Progressive Transmission and Display of Static Images

Roger Andrew Packwood

A dissertation submitted to the University of Warwick
for the degree of Doctor of Philosophy

The Department of Computer Science

The University of Warwick

Coventry

England

September 1986

Contents

List of Figures

List of Photographs

Acknowledgements

I would like to express my gratitude to my supervisor, Dr Graham Martin, for his continued support and encouragement throughout my period of study.

I would like to thank Anne, my wife, for her gracious acceptance of the indulgence this thesis represents.

I received financial support from the Science and Engineering Research Council for part of my period of study.

Declaration

The work presented in this thesis is, except where stated, my own original work. None of the material presented has been submitted in support of any other qualification at any other educational establishment.

Summary

Progressive image transmission has been studied for some time in association with image displays connected to remote image sources, by communications channels of insufficient data rate to give subjectively near instantaneous transmission. Part of the work presented in this thesis addresses the progressive transmission problem constrained that the final displayed image is exactly identical to the source image with no redundant data transmitted. The remainder of the work presented is concerned with producing the subjectively best image for display from the information transmitted throughout the progression.

Quad-tree and binary-tree based progressive transmission techniques are reviewed, especially an exactly invertible table based binary-tree technique. An algorithm is presented that replaces the table look-up in this technique, typically reducing implementation cost, and results are presented for the subjective improvement using interpolation of the display images. The relevance of the interpolation technique to focussing the progressive sequence on some part of the image is also discussed.

Some aspects of transform coding for progressive transmission are reviewed, intermediate image resolution and most importantly problems associated with the coding being exactly invertible. Starting with the two-dimensional case, an algorithm is developed, that judged by the progressive display image can mimic the behaviour of a linear transform while also being exactly invertible (no quantisation). This leads to a mean/difference transform similar to the binary-tree technique. The mimic algorithm is developed to operate on n-dimensions and used to mimic an eight-dimensional cosine transform. Photographic and numerical results of the application of this algorithm to image data are presented. An area transform, interpolation to disguise block boundaries and bit allocation to coefficients, based on the cosine mimic transform are developed and results presented.

Abbreviations

CMT	Cosine Mimic Transform
DMA	Direct Memory Access
DPCM	Differential Pulse Code Modulation
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
EDST	Even Discrete Sine Transform
EDSTo	Even Discrete Sine Transform with offset
FWM	Four Way Mean
MSE	Mean Square Error
MST	Monotone Sub-Tree
MT	Mimic Transform
PTD	Progressive Transmission and Display
RAM	Random Access Memory
RGB	Red Green and Blue
VLSI	Very Large Scale Integration
1-D	one-dimensional
2-D	two-dimensional
3-D	three-dimensional

1. Introduction

1.1. Progressive Transmission and Display (PTD)

The data rate available for communication between an image source and an image display device is not always sufficient to update the display as quickly as an observer of the display device might wish. Although this can be true of 'moving pictures', only issues relating to static images are examined in this thesis. Techniques are discussed that are usually described as 'progressive' in that the ordering of information for transmission and the way that it is displayed are carefully considered. The objective is to present to a viewer the most subjectively useful information in the most subjectively acceptable manner as quickly as possible.

The definition of the most subjectively useful information is application dependent, but can be generalised for some broad application areas, typical expected application areas are the searching of remote image databases and remote surveillance. When performing a search of a remote image database it may be that the search is based on only one property of the images, the subjectively useful information. The decision to wait for full resolution transmission of an image or to move on to the next image can be made more quickly if this one property is conveyed early in the transmission. For remote surveillance some spatially coarse approximation to the input scene would give early indication of change. Depending on the position of change within the scene a conventional slow-scan system might not indicate the change until later due to the scanning time. This approach of first transmitting some coarse approximation to the entire image can also be used for searching an image database where no property other than overall image content is important. Such a technique has potentially wide application and forms the basis of the methods investigated in this thesis. For the two applications quoted, the word remote is used to emphasise that progressive techniques are only useful where available communications channels do not have sufficient data-rate to give near instantaneous image transmission. This is most common where long distances are involved, making high data-rate transmission costs prohibitive.

The second and related part of the investigation is concerned with the subjectively acceptable presentation of the transmitted information. If some coarse approximation to an image is transmitted then this can usually be improved for display by being smoothed. For progressive transmission where early transmitted information is being continuously re-accessed and updated, this post-processing must be designed not to corrupt already received data. The overall process of progressive transmission and display will be referred to as PTD.

A simple numerical example would be the transmission of one data value that represented a pixel block of 8×8 elements in a source image. One value for every such block in an image can be transmitted in 1/64 of the time to transmit all the pixels individually. The single block value is then used in some fashion to generate the displayed pixel values for the block it represents. A possibility for this the display phase is to set all the display pixels within the block to the transmitted value, though this is a very unsophisticated display technique. For an image of 512 lines each of 512 pixels the time to transmit the block values is the same as that required to transmit 8 lines of the image at full resolution.

The various techniques that have been proposed for PTD have not in general been designed to present to a subject viewer any particular property or feature of an image. Typically, some form of iterative algorithm is adapted such that intermediate stages produce presentable images. It is not usually possible to change the algorithm in order to extract information that represents anything other than a very low level description of the image. This lack of sophistication is matched by the lack of any clear understanding of what is a useful high level description of an image that would enable a viewer to make some rapid assessment.

For grey scale images, the approach of attempting to extract salient information from an image contrasts with the common approach of effectively low-pass filtering and sub-sampling as achieved by using a transform coding approach [Lohscheller 1984]. Some progressive transmission techniques are very application oriented, especially those for progressive transmission of bi-level images. These bi-level images fall into three different categories. First is the category of progressive facsimile transmission, and here the property of edge information can be used to aid progression, as the images typically contain diagrams or text. The techniques associated with this category can be very specialised when compared with techniques appropriate to all forms of images. Work has been reported in this area by Johnsen and Netravali [1981], and the work of Frank, Daniels and Unangst [1985] is also related to this area. The second category of bi-level displays comprises those images produced by a dithering technique, to render a grey scale image on a bi-level display. These images have redundancy properties dependent on the dithering process and can thus usefully be treated differently to facsimile style images [Yamada & Inamoto 1979, Netravali & Bowen 1981]. The third category of bi-level displays is that of screened images as used by the publishing industry, the conversion from grey scale to bi-level is generally performed by techniques other than dithering. It has been proposed by Farelli and Marangelli [1984a,b], that photographic material may be archived digitally in a screened bi-level format and that searching of the archive may be made easier if an image is retrieved in a progressive display fashion. All the bi-level progressive schemes utilise a sub-sampling approach, this is the most straightforward method for the low level structure of the image, be it facsimile, dithered or screened. This approach can again be contrasted with one based on some knowledge of optimum display content independent of the display mechanism.

Differing degrees of emphasis are placed on the amount of data compression required of a progressive transmission technique. Robinson and Coakley [1983], in a general introduction to progressive transmission, place emphasis on data compression, as do Farelli and Marangelli [1984a,b] because the compressed form is to be stored in an archive, with the minimum of storage space used. However, Johnsen and Netravali [1981] express satisfaction that their technique requires only about 20 per cent more transmission time than that required for non-progressive transmission. Clearly, the greater degree of compression the shorter the transmission time, which is a major objective. If an image is to be stored in its coded and hence possibly compressed form this can reduce the overall amount of storage required for a data-base or archive. The total transmission time of a specific method needs careful interpretation, as the time/fidelity curve may be such that a generally acceptable image is produced at some relatively early point in the progression. The issue of the degree of compression possible for a stored image, in say an archive, is more clear in that it has an obvious effect on the size of store required. Thus it may be that the degree of compression required for image storage may affect the algorithm used for progressive transmission. If an image is to be stored in its coded form, then certainly this coded form should be no larger than the uncoded image, which might be the case for some progressive techniques that transmit essentially redundant information.

The quality of the final image resulting from progressive transmission is application dependent. In comparison of the final image with that presented for coding, Lohscheller [1984] uses subjective criteria. For archiving applications similar to that proposed by Farelli and Marangelli [1984a,b], although complicated by the fact that they address only the problem of screened images, more demanding criteria might be used, possibly that of overall exact reconstruction.

The usefulness of PTD techniques hinges on the criteria of acceptable transmission time, how this relates to the amount of information to be transmitted, and the available data rates for transmission. The ability to perform PTD depends on the data rate, as this affects the required speed of processing, which also influences the choice of algorithm. In practice there is likely to be a choice of data rates available at differing costs and these must be matched against the cost of the PTD implementation for any change in cost/performance.

1.2. PTD System Requirements

Many non-progressive coding schemes have an overall fidelity behaviour that can only be expressed in a statistical sense, such as mean square error. This subjectively useful measure may not be relevant if any further processing is to be performed on the image, after its receipt via such a coding scheme. Ideally for further processing the received image would be identical to the original submitted for transmission. Furthermore if a PTD method were exactly reversible there would be no need for subjective evaluation of the final image, minimising potential problems with such evaluation. In the opinion of the author, subjective evaluation of the intermediate images need not be as critical as that otherwise required for the final image. Part of the argument for this is that different PTD techniques may have 'complete' intermediate images that contain different amounts of transmitted information, thus making comparison more difficult by requiring some notional time/fidelity interpolation. In addition the early approximations will be grossly distorted, and all intermediate approximations are to varying degrees transient and so may evaluate differently out of the progressive context. Because of these problems of subjective evaluation and the potential for applications requiring exactly reversible coding, the author has chosen to concentrate on these techniques. It may be that a fresh insight into such coding will make possible new work on non-reversible coding with a different understanding of the information being discarded.

Data compression is not an important objective in this work, indeed it is difficult to ensure that PTD techniques do not increase the amount of information to be transmitted. With reversible coding, information cannot be discarded as is common in image transmission (by quantisation) and only redundancy reduction can be used to achieve overall compression. This possibility is investigated wherever a clearly skewed probability distribution is found for variable length coding, and when methods such as or similar to run length coding can be used.

The problem of communication channel errors is not discussed in this thesis. It is expected that the PTD techniques will be used in a digital network environment where provision has been made at some lower level for error detection and correction. The issue of error propagation does not arise under these conditions, for exact reconstruction no undetected error can be tolerated and any consideration of behaviour under error conditions is without meaning. If the data for transmission has any demonstrable redundancy then this could be used to aid error detection, but again the author prefers the approach of removing as much redundancy as possible before committing the data to the communications channel and then adding such checking information as appropriate at the channel level.

The following short set of requirements is drawn both from the preceding section and from a desire to make any resulting design as 'non-controversial' as possible. This second aspect will be explained more fully when the requirements are expanded upon. To perform progressive transmission the input image data must be converted to some new set of values that can be re-ordered for transmission, the conversion being performed by some transform. The inverse of this transform is used to re-assemble the image data at the receiver and during this process some intermediate description of the image data will be used. The requirements are stated in terms of this transform as

i) The transformation should be invertible, that is the final displayed image is identical to the original stored image. This is not meant as 'subjectively identical' but, numerically, all pixel values are identical.

ii) No redundant information is transmitted.

iii) The intermediate representation of the image maintained by the display processor, which contains all the information thus far transmitted, should require no more storage space than the original image. It should be possible to generate display values in a very simple fashion from this intermediate representation.

Considering these points individually in the order listed above

i) It can be argued that techniques exist to reproduce subjectively identical images which is normally all that is required and thus there is little motivation to investigate techniques with exact invertibility. However, subjective measures are generally not sufficient in any instance where further processing is undertaken on the transmitted image. Any detailed numerical results will almost certainly differ from those that would be derived from the original data in a possibly confusing manner. There would be further problems if the image were to be retransmitted as the errors would accumulate. Clearly it is desirable for progressive transmission to be exactly invertible, the practical use of such a technique would depend on the costs it incurred.

ii) The requirement of no redundant data transmission is necessary but probably not sufficient for a technique to be readily accepted and used. Redundant information in this sense is information transmitted in the progressive sequence and then simply discarded most probably by display over-writing as new information is received. If redundant information were transmitted then the benefit of progressive display would need to be weighed against the overhead in a possible conflict of requirement. On a simplistic level, if no redundant information is transmitted, the technique is no worse than clear transmission and the progressive transmission is all added value. However as it is easy to compress an image reversibly by simple entropy or run-length coding (and others) the progressive display technique might be required to perform with a similar compression rate. This point is reinforced by the ability of subjectively good techniques without exact inversion to perform with moderate in-built compression. It is not clear how this requirement is related to steering information. With an exact one-to-one coder, improvement might be made by directing initial update to some specific part of the image. While this is not redundant information it is an overhead, needing value judgements of performance increase against extra incurred costs. To avoid having to make these judgements, techniques that require steering information will not initially be investigated.

iii) Ideally the intermediate representation would be directly displayable, each element in the intermediate representation corresponding to a display pixel. If the representation requires no more resolution than the display grey-scale range it can be stored in the display memory. This is an example of a 'non-controversial', no-cost implementation possibility with no extra storage required above that needed to display the image, and it would thus be most acceptable. It does however seem somewhat unrealistic and it is useful to examine the consequences of not meeting this requirement, these are basically of equipment cost. If the intermediate representation requires more resolution than the grey-scale image but display values can be generated from it in real-time by relatively simple hardware, then the resolution of the display memory can be increased and the extra hardware attached. For the extra resolution this is a proportional increase in cost over existing storage and if the extra hardware is similar to say a look-up table, then the total cost increase is not excessive. If it is not possible to generate display values in real-time from the intermediate representation then it must be stored separately from the display with greater cost implications. The total amount of memory to be required is more than doubled and as discussed later there are performance implications in having to maintain the intermediate representation and update the display. Despite stating the effects of relaxing this requirement, meeting it does remain a very firm and worthwhile objective, justifying maximum design effort.

1.3. Potential Application Parameters

The potential for PTD depends on the combination of image resolution and data communication rate which together define the total transmission time. There is a range of total transmission times over which it is reasonable to attempt to apply PTD techniques, though this will to some extent be application dependent. If the time to receive an image is of the order of a few seconds it would not seem that any benefit from PTD would be justifiable because of the inherently short waiting time. This might be investigated more closely if the application was surveillance of some rapidly changing scene, with critical fast response, but this would be exceptional. Periods of several tens of seconds of transmission would benefit from PTD but at some point the total time becomes excessive for waiting, though this might be as long as several minutes, judging from accepted computer response times. Again at this extreme of waiting time, potential for PTD will be application dependent and the possibility of the viewer being completely satisfied with an intermediate rather than the final image must be assessed. A number of different image resolutions and data communications rates are shown in figure 1.1, this is not meant to include all such possibilities but to indicate some typical values together with some extreme combinations as boundary examples.

Data Rate	Data (bits)				Usage
	16kx8	512x512x8	512x512x8x3	2kx2kx8x3	
4800 bits s^{-1}	27s	500s ≈8mins	1500s ≈25mins	24000s ≈48hrs	computer terminal or modem over telephone
64 kbits s^{-1}	2s	32s	96s	1500s 25mins	wide area network
10 Mbits s^{-1}	0·02s	0·25s	0·75s	12s	local area network
	early Picture Prestel insert	monochrome TV	colour TV	very high res. imaging	

Figure 1.1 The transmission times of different combinations of image resolution and data rate.

Considering the data-rates shown in the table, the rate of 4800 bits s^{-1} is typical of computer to terminal rates over RS423 connections and also of modems over telephone type connections. A data rate of 64k bits s^{-1} matches a group of 8 voice channels between digital telephone exchanges, and is used as a target data rate in telecommunications applications, 8k bits s^{-1} might also be considered with this. The 10M bits s^{-1} data rate of a local area network such as Ethernet is slightly contentious in that it is unlikely that a host computer, especially if time-shared, could manage such a continuous data output rate, limited by processor overhead, internal bus and disk bandwidth. This channel data rate is then an upper bound rather than an expected rate, it would be reasonable to consider rates down to an order of magnitude lower for network connection. Such networks are becoming more common and as they are often used to connect powerful work-stations together which are suitable for image processing, the inclusion for consideration of the approximate 10M bits s^{-1} data rate seems reasonable. The issue of further processing in this instance is very important to the use of exactly reversible PTD methods.

Considering the various spatial image resolutions shown in the table, the quantised resolution for all the entries in the table is quoted as 8 bits, this has been used as a commonly accepted standard. It is assumed that the PTD techniques will be usable with any quantised resolution with only minor changes, 6 bits being usually considered as an adequate resolution value, 8 bits is accepted as a standard of good quality quantisation. The smallest image information quantity shown in the table is the size of the memory used for the luminance information of a picture insert in early Picture Prestel investigations [Nicol, Fenn & Turkington 1980]. The design criterion was the size of the memory, rather than any specific insert size. The shape of the insert was variable within the limit of the memory size and is generally consider to be about one ninth of the picture area of a domestic television picture. The Picture Prestel system is colour rather than monochrome using luminance and chrominance information but for the purpose of this discussion the colour components will be ignored. The pixel resolution of 512 square (second column figure 1.1) is rather less than that of broadcast television, having too few lines and rather less horizontal resolution, it is however a common size for image stores and a convenient computational size. This is simply multiplied by three to give the figure in the third column which is based on a naive colour representation, again this is more of an image frame store standard than one for data transmission. The final column is an extreme fictitious example using spatial resolution that might arise in medical or remote sensing, multi-spectral applications. It would be very difficult to display such an image using current display technology.

From the first column of the table, the lowest data rate and the lowest resolution combine to a transmission time ideally suited to PTD, it being this particular application that caused increased interest in PTD. However at the next higher data rate (down column) the transmission time of this relatively small quantity of data is too short for PTD to be applicable.

Considering the second column, 512 pixel square resolution, the lowest data rate (4800 bits s^{-1}) seems impractical due to the long transmission time (8 mins.). However the excess time factor in this case is not large, a factor of four over the previously stated maximum time of two minutes, in comparison with possible simple compression techniques. Without investigating any particular algorithms, compression of two would be a reasonable expectation based on second order pixel entropy as given by Schreiber [1956].

Though there remains some disparity between possible and required compression factors this combination of data rate and image resolution need not be rejected for PTD consideration. An increase in data rate to 64k bits s^{-1} places the transmission time again in the PTD area, though the data rate increase to 10M bits s^{-1} decreases the time below the limit.

Considering the naive colour representation (third resolution column), at the lowest data transmission rate excessive compression above simple pixel entropy would be required to transmit this amount of data in a period suitable for PTD (a factor of approximately 12). However with the middle data rate, 64k bits s^{-1}, PTD could be usable, although without compression the waiting time is 96s.

The fourth column with the highest resolution display figures shows an unacceptable waiting time at 4800 bits s^{-1}, though the total time for transmission does not make it impossible to transfer files of this size (times of several weeks might be considered excessive). Such a system might be regarded as very slow facsimile, increasing the motivation to check the image at the outset of transmission, PTD could be used for this check. Similar remarks apply to the higher data rate of 64k bits s^{-1} which is still a long haul data rate, giving a more reasonable facsimile with PTD used for a visual pre-check. The combination of high resolution image and high data rate network is marginally suitable for PTD, though the existence of this combination is debatable. Displays of 1k pixels square by 8 bits are becoming more common, which the 10M bits s^{-1} network should be able to transfer in approximately one second. However, if as stated the image source fails to match this data rate by a conceivable factor of 10, then PTD might be used if only for 'cosmetic' effect.

From the table, transmission time indicates if it would be appropriate to use PTD techniques, but does not determine the possibility, this depends only on the transmission data rate. At the extreme limiting case, if data sufficient to describe a single pixel arrives at intervals equal to the time required to write a pixel value to the display memory, then there is no processor resource free to perform any decoding or image manipulation. The processor must be matched to the requirement in terms of data rate and algorithm complexity, or more likely, the complexity of the algorithm attempted is dictated by the amount of unused processing power. Special purpose hardware might be constructed to provide support for PTD and algorithms might be optimised for such an application. In general the complexity of the algorithm is very important due to the criticality of execution time and the multiplicative effect of the very large amounts of data involved.

1.4. Display System Architecture

A guide to the ability to perform PTD is the time taken to receive one new data value, depending on the data rate and size of the datum. At 4800 bits s^{-1} with 8-bit data and some small overhead the time between arrival of data values is approximately 2 ms. Further calculation is based on a hypothetical processor with a bus cycle time of 1 μ s and an instruction time of 4 μ s, one cycle for instruction fetch, operand in, action, operand out. Such a processor can execute approximately 500 instructions per datum. This estimate should be considered conservative because of the number of cycles per instruction and cycle time, allowing an arbitrary factor of four tolerance on this gives between 500 and 2000 instructions per datum. Making a very tenuous assumption of 10 instructions to update a single pixel gives figures of 50 and 200 pixels that can be updated in one datum period. These figures are obviously very approximate but do indicate that only a relatively small part of an image can be updated per datum. Whereas the processor speed assumptions may be pessimistic, the number of instructions per pixel may be very optimistic, being dependent on the display architecture and the coding algorithm. If the number of instructions per pixel were to increase to 100, then only as few as 5 pixels per datum might be updated. Therefore the data rate which determines the required throughput is a fundamental and very important design criterion. Using the hypothetical processor the position is much more restricted at 64k bits s^{-1}, giving a per datum time of only 144 μ s, equivalent to 24 instructions per datum. It seems inevitable that some form of hardware assistance would be required to do any significant manipulation in this time period. At 10M bits s^{-1} the hypothetical processor could not receive data under program control, though at this rate it would not normally be expected to do so. Any PTD processing at a rate close to this would have to be performed almost entirely by some purpose designed hardware. This will raise the issue of suitability of various algorithms for hardware implementation. The question of how many pixels can be updated per datum is very important to the decision about pixel grouping in an image and to the question of the resolution of the first approximate image. For some algorithms the first approximation is the worst case iteration for the number of pixels that must be updated per datum. Performance is adversely affected if more pixels need to be changed than the datum period permits, meaning data that could be displayed remains unused, buffered either at the image source or the display local processor.

A subjective factor is the time taken by a subject viewer to comprehend in any useful way an image that is being displayed either instantaneously or progressively. It may be that a suitably chosen PTD technique causes no greater delay in the making of an image based decision, than the delay occurring when an image appears instantaneously. In this case it may be possible to use a lower data rate on a cost basis, with no reduction in performance.

The architecture of the display system is important for progressive display work, which is very demanding of performance independent of the power of the processor used. To update a pixel it must at least be possible to change the contents of the display memory without causing any visible effect other than that expected on the addressed pixel, some low performance systems have unfortunate visual side effects when display memory is accessed. It is also assumed in the following discussion that the display memory is in some way readable by the local processor, though it is conceivable that some display systems may not have this facility The main variation in display architecture is the way in which the processor local to the display accesses the display memory. This divides into two basic categories, memory mapped and register access.

With memory mapped displays the video refresh system accesses memory that is transparently available in the processors address space. This is the easiest system to work with from the programmers viewpoint, and as it requires no program code overhead for access, it is also potentially the fastest for program execution. If access is completely transparent there is no need to localise or pattern accesses to the memory, placing no constraint on the algorithm. There may be some system-wide constraint for doing this if any cache is present and random addressing of large areas of memory can cause problems with some memory mapping systems, but these issues are outside the scope of this discussion.

Register access at its simplest requires functionally three registers, two co-ordinate registers x, y, and a data register. The image is updated by writing x and y co-ordinates followed by a new data value, reading similarly, by writing co-ordinates and reading the data register. This requires more program code than the memory mapped system and is correspondingly slower. If this additional code is not in-line and a subroutine or procedure call is used, the effect on program throughput can be substantial. The greater part of the activity of a PTD program is manipulating the displayed image and register access can mean more effort to update a displayed pixel than required to calculate the new value. Using this simple register access mechanism there is again no incentive to process pixels in blocks rather than individually as the pixel access operation is independent of any locality of operation. Register access can be improved by having an auto-increment or decrement facility in conjunction with the co-ordinate registers, which operates when the data register is accessed. This allows more rapid update of a horizontal or vertical line within the display and is a simple way in which the display architecture can influence the choice of algorithm. In order to achieve the greatest speed of image manipulation it is natural to use algorithms that operate with the fastest access mode of the memory, which in this case is line-by-line. The idea of automatic increment is extended when the display memory is connected to the processor system by a Direct Memory Access (DMA) device. To perform a DMA transfer at least the registers for a buffer pointer, count for transfer and direction for transfer must be loaded, possibly in conjunction with a go command. This is adequate only if the display system is sufficiently intelligent to extract x,y start position information from the data stream being passed by DMA. If this is not possible then further registers are required to instruct the display system on the action to be performed with the data stream. For single point access this method is very slow, having a strong influence on the algorithms used. Some form of block access will give higher through-put than point by point access if highest speed processing is required using DMA transfer.

Inevitably there will be mechanisms that combine some of these features. One such technique maps a single line of the display memory into the processor memory space, depending on some register. Such a line based access mechanism might present some degree of impediment to algorithms that are based on rectangular block working areas.

The ability to rapidly update a display depends not only on the connection between the processor and display memory but also on the way in which the video refresh system accesses the display memory. It is possible to use very fast access display memory that can service both processor and video refresh without penalty, but this is too expensive to be used often. Instead the processor must compete for display memory bandwidth with the video refresh system, which can take either nearly all the bandwidth or very little depending on its design. Usually, individual display memory devices can service either the processor or the display refresh mechanism but not both without causing one to wait. The worst system built with unsophisticated memory devices uses all the memory bandwidth for refresh except during blanking. Such a system is only marginally usable for PTD, giving perhaps only 25% access to the processor, its only justification is its low cost. More complex systems interleave memory devices for increased processor access, thus if the devices are two-to-one interleaved the processor is guaranteed at least 50% access, four-to-one guarantees 75% access, requiring increasing amounts of hardware. Currently the best systems use specialised video RAM, which have built in shift registers that can be loaded from the cell array. Video refresh is then performed entirely from the shift register which is of length the side of the cell matrix, possibly 256 bits long. The memory is available for use by the processor between shift register loads, and figures of greater than 90% are quoted for processor access of these devices.

Progressive display algorithms must have a working copy of the image displayed which will be updated as new information arrives, this new working copy must then be displayed. If the working copy can be kept in display memory then no work is required to update the display from the working copy. However it is possible that the working copy needs to contain values that are not directly displayable and therefore cannot be stored in display memory. As an example the software may wish to store an invalid display value in a pixel to denote that the true pixel value has not been received, possibly a negative number may be used. This implies that the working copy must be kept to greater resolution than the display to permit storage of these extra values. The software must then update the display memory by interpreting the contents of working copy. This arrangement is very expensive in terms of memory and can incur some processor overhead in maintaining the two separate representations. An improvement upon this is to introduce some hardware mapping between the display memory and the part of the display hardware that converts a pixel value to an analogue signal, this may take the form of a look-up table. As an example if the image data were quantised to 8 bits, then a display memory 10 bits deep would permit the storage of positive and negative numbers of twice the range of the data. The corresponding look-up table would have 1024 entries of 8 bits and could be loaded to clamp all negative values to zero and positive values greater memory bandwidth with the video refresh system, which can take either nearly all the bandwidth or very little depending on its design. Usually, individual display memory devices can service either the processor or the display refresh mechanism but not both without causing one to wait. The worst system built with unsophisticated memory devices uses all the memory bandwidth for refresh except during blanking. Such a system is only marginally usable for PTD, giving perhaps only 25% access to the processor, its only justification is its low cost. More complex systems interleave memory devices for increased processor access, thus if the devices are two-to-one interleaved the processor is guaranteed at least 50% access, four-to-one guarantees 75% access, requiring increasing amounts of hardware. Currently the best systems use specialised video RAM, which have built in shift registers that can be loaded from the cell array. Video refresh is then performed entirely from the shift register which is of length the side of the cell matrix, possibly 256 bits long. The memory is available for use by the processor between shift register loads, and figures of greater than 90% are quoted for processor access of these devices.

Progressive display algorithms must have a working copy of the image displayed which will be updated as new information arrives, this new working copy must then be displayed. If the working copy can be kept in display memory then no work is required to update the display from the working copy. However it is possible that the working copy needs to contain values that are not directly displayable and therefore cannot be stored in display memory. As an example the software may wish to store an invalid display value in a pixel to denote that the true pixel value has not been received, possibly a negative number may be used. This implies that the working copy must be kept to greater resolution than the display to permit storage of these extra values. The software must then update the display memory by interpreting the contents of working copy. This arrangement is very expensive in terms of memory and can incur some processor overhead in maintaining the two separate representations. An improvement upon this is to introduce some hardware mapping between the display memory and the part of the display hardware that converts a pixel value to an analogue signal, this may take the form of a look-up table. As an example if the image data were quantised to 8 bits, then a display memory 10 bits deep would permit the storage of positive and negative numbers of twice the range of the data. The corresponding look-up table would have 1024 entries of 8 bits and could be loaded to clamp all negative values to zero and positive values greater than the maximum range of the display to the maximum value. Alternatively the table could be used to compress a greater range of display memory values into the range of displayable values, the fact that this is non-reversible causing no problems. Look-up tables of this sort are not common in display equipment, they are more often used to expand an 8 bit display value to a 24 bit RGB colour value. For monochrome work such a table could be used for mapping 8 bits to 8 bits of monochrome, but this would only be of use if data quantised to less than 8 bits were being used.

The display system used by the author has display memory that is four-to-one interleaved and so has high availability to the local processor. It is possible to alter the display memory with no visible side-effects though on very close inspection some local activity can be seen, possibly caused by the interleaving mechanism. However the connection between the processor and the display is in the form of a DMA interface which has considerable overhead in random point addressing, and for speed in demonstration, block reading and writing are used whenever possible. There is no hardware mapping between the display memory and the digital-to-analogue conversion system thus a pixel store value is converted to a luminous intensity. Each picture store is 6 bits deep with 6-bit conversion thus the resolution is just adequate for image processing work rather than as good as possible. Various processors have been attached to the frame store equipment used by the author during the course of the work reported here, the main issues being the provision of floating point arithmetic and the size of available memory. The memory size, at its largest 256k bytes presents problems in conjunction with the frame store when extended precision for an image is required.

2. Tree Coding

2.1 Simple Quad-tree Coding Methods

A method of relating the successive images which form a progressive display sequence, known as a pyramid data structure has been developed by Sloan and Tanimoto [1979]. The pyramid in conventional orientation has as its base the original image, it is then built up of successive layers, each a more coarse approximation, to the peak, the most coarse. To move up a layer in the pyramid, a group of pixels in the layer to be built on of rectangular shape and size $m \times n$, the 'reduction window', is used to generate the value of a single pixel in the immediately superior layer. The algorithm used to generate the single pixel is called the 'reduction rule'. The technique as described has the topmost layer of the pyramid of unit dimension and the number of layers in the pyramid as L, where the original image has dimension m^L, n^L. The base is described as level L and the peak as level 0, the ordering being based upon the decoding sequence, rather than the coding sequence.

A PTD algorithm is discussed by Sloan and Tanimoto that is independent of the reduction rule used, as no previously transmitted information is used by the receiving process at any stage in the progression. The algorithm is suggested in conjunction with a reduction window of 2×2. For progressive transmission, each layer of the pyramid is transmitted in turn, each layer forming an intermediate image, starting with the topmost layer of a single pixel. Each transmitted layer is expanded by pixel replication within the reduction window to fill the entire display area. Thus the first intermediate image is a monotone grey level over the whole display. The bottom layer of the pyramid is the original image which is transmitted with no processing to form the final image identical to the original. As each layer is transmitted the data simply overwrites that of the preceding layer, causing the total amount of information transmitted to be greater than that of the original image. The amount of information in each layer increases (transmission order) as a geometric progression and the quantity of information overwritten at the display is approximately one third of that of the original image, using a 2×2 reduction window.

A reduction rule called 'SUM' is also described, by this rule elements in the reduction window are summed to produce the new single pixel value. Using the same reduction window size of 2×2, the summation requires two additional bits of accuracy per level, both for transmission and for intermediate storage. The advantage of this rule is that the single pixel summation value can be utilised in the calculation of the four pixel values in the reduction window, rather than being discarded by overwriting. This is done by transmitting three of the values and subtracting these from the sum to find the fourth value. The redundant information transmitted is the extra bits at each level, extending the precision of the mean, for 8-bit data this represents about 12% additional information. The information is redundant because the coding scheme does not utilise the constraint that the fourth pixel value which is calculated must be within the pixel value range rather than the range of the sum, all of which is available to the fourth pixel depending on the other three values. A problem with the SUM rule is that although the total number of bits required for intermediate storage decreases with assent of the pyramid, the increase in required resolution can be quite large. Moving up the pyramid nine levels for a 512×512 original image adds 18 bits to the starting resolution. It would be impractical to give the entire frame-store 18+8 bits of resolution, though software could be used to assemble extended resolution from otherwise unused pixels. These unused pixels could be created if the hardware were capable of dynamic variable pixel replication, not displaying all pixels but repeating some to fill the entire display. This reduction rule, apart from the way in which the scale factor is used, is identical to the taking of an exact mean which would also require two further bits of accuracy per level. With SUM, the most significant part of the stored value equivalent to the range of the display would form the displayed value.

Some mechanism can be used in addition to the reduction rule to utilise for compression any monotone area in an image. If the pixels of a reduction window are all of the same value, and of the same value as their parent pixel which has been transmitted, then the pixel values need not be explicitly transmitted. In the place of the values, some control information is transmitted to indicate the parent value is correct for the whole window and to update the place keeping algorithm appropriately for the next data to be transmitted. The opportunities to do this are image content dependent, and will tend to vary with the number of quantisation levels used. This technique is called 'Explicit Repainting' by Sloan and Tanimoto. The notion of user interaction with the transmission process to specify some area of the image to be preferentially updated was also introduced under the title of 'Interactive Detailing'.

The techniques presented by Sloan and Tanimoto [1979], characterised by the pyramid representation and the different reduction rules form a very basic approach to PTD, with respect to the desired properties set out in section 1.2. The simplest method, full transmission of each layer does not meet the requirement of no redundant information. The SUM reduction rule has better redundancy properties but requires either higher store resolution or increased complexity in the display hardware beyond that specified in section 1.4.

A quad-tree based technique more sophisticated than those reproduced above has been presented by Wilson [1984]. In this technique, value differences between levels in the tree are coded by a predictive scheme augmented by interblock coding and quantisation. Another pyramid based approach has been described by Burt and Adelson [1983] with associated filter functions described by Burt [1981]. In this the image data is represented as a pyramid of differences between a sequence of iteratively sub-sampled images produced by convolution with a gaussian shaped weighting function. The sub-sampling is by a factor of two in each direction, yielding a pyramid of similar shape to that of a quad-tree, but the gaussian weighting function extends over an area of 25 pixels. The technique due to Wilson [1984] is more sophisticated than appropriate for extended consideration at this stage of the investigation, and the problem of sets of differences as used by Burt and Adelson 1983], with respect to exact invertibility, is simply more complex than that presented by the methods of Sloan and Tanimoto [1979].

2.2. Exactly Invertible Coding with Binary-trees

A technique that builds to some extent on the work of Sloan and Tanimoto has been reported by Knowlton [1980]. It addresses the problem of taking the mean of the elements in a reduction window as the reduction rule, which would normally require extra precision. The problem is restricted to two elements, with the proposed technique likened to a linear transformation, which requires only a single difference value to resolve the mean into two data values. An approach analogous to Sloan and Tanimotos SUM reduction rule might be used to transmit the mean or sum of two elements with one extra bit of resolution, followed by one data value to resolve the pair. The second data value can simply be one of the pair of element values to be subtracted from the mean/sum. The overhead is one bit per pair of values, and the attendant problem is increased intermediate storage resolution. A more conventional transform approach would resolve the mean using a difference value, rather than an original element value, the disparity between these two approaches using two values is small but significant, and becomes much greater as more values are considered. A detailed description of a mean and difference transformation is given in appendix A, though a clear distinction is that a true difference value is signed whereas an element value is not.

The technique presented by Knowlton is a hand crafted transformation that requires no extra information to describe an approximate mean, which can then be kept in intermediate store with no additional resolution. The approximate mean can be resolved into two values using a number that behaves somewhat like a difference value but again requires no extra precision. The complete transformation is described by a set of look-up tables. The term 'composite value' is introduced for the value that approximates the mean and the term 'differentiator' for the value that corresponds to the difference. The goodness of the approximation of the composite value to the true mean varies within the input space, and this has been optimised with respect to an assumption of expected data point density within the input space. This assumption is that for typical image data, a pair of adjacent pixels will have similar values and the transformation table is designed for such pixel pairs to have a composite value very close to the true mean. In general as the difference between the two pixel values increases, so does the difference between the composite value and the true mean. For data with a range of three bits, the composite value table is shown in figure 2.1, and the corresponding differentiator table shown in figure 2.2. The tables are shown for only 3-bit data to make their size reasonable for reproduction, none of the important properties of the tables are affected by their size. Note that the differentiator table is a 90 degree clockwise rotation of the composite table and a simple transformation can be used to extract differentiator values from the composite table. It can be seen from figure 2.1 that for points on the diagonal through the origin, which represent equal value pairs, the composite is equal to the true mean or pixel value. The error between the true mean and the composite value is shown in figure 2.3, its behaviour not being completely optimal with respect to the above stated simple assumptions of adjacent pixel value probabilities. These assumptions would indicate that pixel pairs with a difference of the full data range are relatively unlikely yet the composite value for these pairs has minimal error with respect to the mean, but this property of the error has no adverse effects.

The table is stated to have been laid out by putting a unit wide strip of appropriate length (Knowlton shows a table for 4-bit data which is 16 × 16) in the lower left corner of the table, and 'by adding one-unit wide strips as best one can, successively'. The table has a unique composite/differentiator pair for each data combination and so the transformation can be inverted. The tables for the inverse transformation have identical content to those for the forward transformation, figure 2.1 and 2.2. To use these table for inverse transformation, the names 'p_0' and 'composite' are interchanged and also 'p_1' and 'differentiator'. Figure 2.1 becomes the p_0 table with the horizontal index the composite value and vertical index the differentiator, similarly for figure 2.2

	7	3	4	5	6	7	7	7	7
	6	2	3	4	5	6	6	6	7
	5	1	2	3	4	5	5	6	7
	4	0	1	2	3	4	5	6	7
p_1	3	0	1	2	3	4	5	6	7
	2	0	1	2	2	3	4	5	6
	1	0	1	1	1	2	3	4	5
	0	0	0	0	0	1	2	3	4
		0	1	2	3	4	5	6	7
					p_0				

Figure 2.1 The composite values of Knowltons method for 3 bit data, $p_0 p_1$ are two pixel values.

The transform is applied iteratively in a fashion similar to that described by Sloan and Tanimoto [1979] and the result is a binary tree as illustrated in figure 2.4. The first iteration combines pairs of real pixel values to produce a composite value and a differentiator, in figure 2.4 taking the top row to the second row. The notation indicates how the tree is built without requiring extra work-space. Thus pixels a,b are combined and pixel a overwritten with the differentiator value and pixel b overwritten with the composite value. At the next iteration composites b,d are combined with similar overwriting. The final iteration

The table is stated to have been laid out by putting a unit wide strip of appropriate length (Knowlton shows a table for 4-bit data which is 16 × 16) in the lower left corner of the table, and 'by adding one-unit wide strips as best one can, successively'. The table has a unique composite/differentiator pair for each data combination and so the transformation can be inverted. The tables for the inverse transformation have identical content to those for the forward transformation, figure 2.1 and 2.2. To use these table for inverse transformation, the names 'p_0' and 'composite' are interchanged and also 'p_1' and 'differentiator'. Figure 2.1 becomes the p_0 table with the horizontal index the composite value and vertical index the differentiator, similarly for figure 2.2

	7	3	4	5	6	7	7	7	7
	6	2	3	4	5	6	6	6	7
	5	1	2	3	4	5	5	6	7
	4	0	1	2	3	4	5	6	7
p_1	3	0	1	2	3	4	5	6	7
	2	0	1	2	2	3	4	5	6
	1	0	1	1	1	2	3	4	5
	0	0	0	0	0	1	2	3	4
		0	1	2	3	4	5	6	7
					p_0				

Figure 2.1 The composite values of Knowltons method for 3 bit data. p_0 p_1 are two pixel values.

The transform is applied iteratively in a fashion similar to that described by Sloan and Tanimoto [1979] and the result is a binary tree as illustrated in figure 2.4. The first iteration combines pairs of real pixel values to produce a composite value and a differentiator, in figure 2.4 taking the top row to the second row. The notation indicates how the tree is built without requiring extra work-space. Thus pixels a,b are combined and pixel a overwritten with the differentiator value and pixel b overwritten with the composite value. At the next iteration composites b,d are combined with similar overwriting. The final iteration

p_1 \ p_0	0	1	2	3	4	5	6	7
7	0	0	0	0	0	1	2	3
6	0	1	1	1	1	2	3	4
5	0	1	2	2	2	3	4	5
4	0	1	2	3	3	4	5	6
3	1	2	3	4	4	5	6	7
2	2	3	4	5	5	5	6	7
1	3	4	5	6	6	6	6	7
0	4	5	6	7	7	7	7	7

Figure 2.2 The differentiator values of Knowltons method for 3 bit data. $p_0 p_1$ are two pixel values.

p_1 \ p_0	0	1	2	3	4	5	6	7
7	−0·5	0	0·5	1	1·5	1	0·5	0
6	−1	−0·5	0	0·5	1	0·5	0	0·5
5	−1·5	−1	−0·5	0	0·5	0	0·5	1
4	−2	−1·5	−1	−0·5	0	0·5	1	1·5
3	−1·5	−1	−0·5	0	0·5	1	1·5	2
2	−1	−0·5	0	−0·5	0	0·5	1	1·5
1	−0·5	0	−0·5	−1	−0·5	0	0·5	1
0	0	−0·5	−1	−1·5	−1	−0·5	0	0·5

Figure 2.3 The difference between a true mean value and the composite value of Knowltons method.

produces a composite value for the whole image and a differentiator to generate two half areas. The tree is described in one dimension and in order to reduce in two dimensions, the transformation (reduction) has to be applied in alternating directions. Knowlton reports coding from a pixel which has a height to width ratio of 2:3. By performing the first pixel combination in the height direction a pixel of height to width ratio of 4:3 is produced. In this way the extreme ratios of 1:1,2:1,1:1 are avoided to be replaced with 2:3,4:3,2:3. Without this change in aspect ratio the two-dimensional reduction is shown in figure 2.5. The area is shown and marked for a total area of 8×8 pixels rather than the whole image. This particular size, which will be commented upon later, is used partly to show clearly the scanning mechanism over a small area instead of perhaps less clearly over an entire image. The cell contents abcdefg correspond to those in figure 2.4 with the arc notation also corresponding. After coding there is only the single composite value g, all others being differentiators. Decoding is the reverse process of coding, commencing from the entire image area being filled with the same gray-level, the composite value for the whole image, this would correspond to the value g in figure 2.4. The image is then bisected to produce two monotone areas, and again iteratively until the original image is recreated, figure 2.5 can also be interpreted as depicting the decoding process. It is for instance possible to methodically extract the composite value from a block in the display at the same position it would be placed during coding. These blocks will be referred to sometimes as composite pixels in both coding and decoding phases, and when distinction needs to be made, an actual display or data pixel will be referred to as a real pixel. Note that the same tables used for coding can be used for decoding if the table is interpreted correctly.

There is clearly a fundamental difference between Knowlton's method, based on a binary tree, forced to alternate direction, and Sloan and Tanimotos approach with a 2×2 reduction window producing what has come to be known as a quad-tree. Knowlton states that the two way scheme was chosen instead of a four way decomposition for two reasons. Firstly it is a 'gentler' way of manipulating the image and can therefore be carried out more quickly, and secondly it requires much smaller look-up tables. These two assertions need to be examined more critically.

Undoubtedly splitting a composite pixel into two pixels of more correct value is a 'gentler' way of changing an image than splitting a composite pixel into four. The assumption here seems to be that three

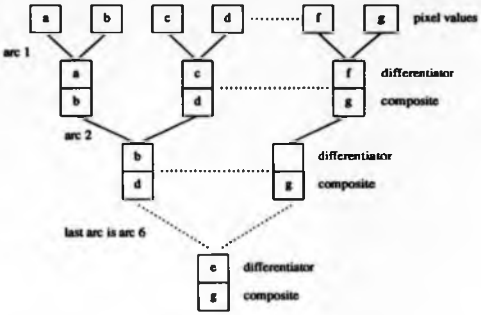


Figure 2.4 Binary tree produced by iterative combination of pairs of values using Knowltons method. The cells are labelled showing how composite and differentiator can overwrite the original data.

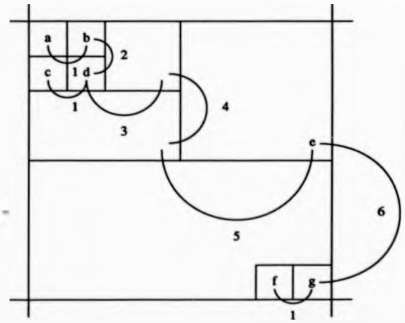


Figure 2.5 A possible scanning order to correspond to the binary tree produced by application of Knowltons method.

values will be supplied simultaneously for one complete update to four pixels, this is the case with Sloan and Tanimotos SUM rule. However if one value were used to update the four pixels then the change would be as 'gentle' if not more so due to a lower input of information per pixel. This would not be possible with the SUM rule, as receipt of one correct pixel value would not allow a four pixel group to be split into two pairs in any useful fashion. To do this requires behaviour similar to that of a linear transform, an extension of the two dimensional table (for use with two values) to four dimensions. The issue of speed here is only a problem if the display system is compute bound rather than bandwidth bound, though there will almost certainly be more computation in total if four pixels are updated with one datum and then re-displayed. The update would need to be repeated three times to completely resolve the four pixels.

Referring to the second point, about table size, this is a definite problem. In the original description, Knowlton gives tables for 4-bit data and does not discuss the size of the tables in general, though they are trivial to calculate. A single table for 8-bit data has 2^8 entries per side so contains 2^{16} or 64k entries which in this case should be bytes. Because of symmetry only one such table is required, and a table of this size, although uncomfortably large for a sixteen-bit processor, is not prohibitively expensive in terms of the overall display system cost. This is not true of a four-dimensional table which would require 2^{32} entries, a completely unrealistic size. Thus to say that the technique using two values requires much smaller look-up tables is if anything, an understatement. The four-dimensional tables required for a four value method could not be implemented. The difficulty of implementation is sufficient cause to cease consideration of these large tables, but another challenging problem remains with respect to the contents of the tables. Proceeding by analogy with Knowlton, this would require the fitting of a number of three-dimensional solids into the vertex of a four-dimensional solid, a somewhat daunting proposition, nor is the three-dimensional case an easy or clearly useful intermediate formulation. In terms of an appropriate linear transformation equivalent, four point (value) transforms are easily derived, especially those based on the Hadamard transform. Such a transform has been used for quad-tree coding by Wendler [1982], though not in an exactly reversible configuration but using quantisation. A composite system using a Hadamard transform over 2×2 elements with level by level differencing, producing a quad-tree structure, has been reported by Yasuda, Tagaki and Awano [1979, 1980], again the emphasis is not on exact invertibility. It seems in general that the quad-tree is more favoured for image processing than the binary-tree, which is occasionally employed to enable Knowlton's transformation to be used. A more extreme example is the use of oct-trees for tomographic representation where an eight point transform might be used, but instead three levels of a binary-tree are required to use Knowlton's transform [Hardas & Srihari 1984]. A generalised approach to transformations similar to but over more points than Knowlton's would find application, but the approach would need to be algorithmic rather than table driven, because of the prohibitive size of the tables. Of course an algorithm was required to fill the original table (based on unit strips), but the filling algorithm may not be usable for coding in any way better than a linear search through the values generated by the algorithm.

The idea of 'Interactive Detailing' proposed by Sloan and Tanimoto is developed by Knowlton into the use of concentric rings of different resolution around some focal point in the image. It is also suggested that this focal point need not be defined by the user but selected at coding time. This topic is addressed to different degrees by most investigations of PTD. With the basic mechanisms for PTD in place the implementation of some reverse channel for user direction should not present any major difficulties.

2.3. Reducing Redundancy within the Binary-tree

Two methods of compression are discussed by Knowlton, one identical to that presented by Sloan and Tanimoto, of being able to mark a sub-tree or cone of the pyramid as monotone and therefore requiring no further information to update it. This technique of marking a subtree of a Knowlton decomposition was investigated to determine if any significant data compression could be performed in this way. For this investigation the full binary tree was calculated up to the root value average for the GIRL image, this image is reproduced in photograph 2.1. The results of the investigation are shown in figure 2.6 and as can be seen there is only one monotone sub-tree (MST) of size 64 pixels and none larger. The method used for finding the MSTs is such that the top entry of a column in figure 2.6 corresponds to the top-most (tree-wise) differentiator that has the value corresponding to no-difference. The number at the top of the column is a count of the occurrences of such a node in the tree. As an example the right-most column corresponds to real pixel pairs as scanned by the algorithm with identical values, the column to the left of this corresponds to pixel quads. The exact degree of compression possible using the MSTs will depend on the coding algorithm in addition to the size of any monotone subtrees. An additional differentiator value can be used to denote the root of a monotone subtree indicating that both children of the node have the same value, and also all below that point requiring no further update. Of course there is already a differentiator value indicating that both children have the same value, this has special significance when marking identical real pixels that are not part of a larger MST. The new differentiator gives no saving for these real pixel pairs and hence some of the obvious redundancy within an image is left unaffected. With this approach each transmitted differentiator would carry the overhead, albeit small, of this extra possible 'root' value. Some more sophisticated coding scheme than simple binary enumeration would have to be used, since for 6-bit data the number of differentiators would increase from 64 to 65. This is only a small increase in information to 6·03 bits, (assuming equal probability) but cannot reasonably be coded by adding an extra bit to the differentiator. The saving in transmitted information per monotone tree increases with the size of the tree, though in a somewhat non-obvious fashion. The binary tree producing n^2 real pixel values has $n^2 - 1$ differentiator nodes, as the root differentiator needs to be transmitted the saving is $n^2 - 2$. These values are shown (figure 2.6) as the row marked 'save/tree' and then multiplied by the instances of the tree at that size to produce the row 'total saved'. Summing across this row the potential number of nodes that need not be transmitted is 16008. The amount of redundant information does not change greatly when the remaining nodes are multiplied by the increase in resolution required to represent the extra differentiator value, so the potential saving remains approximately 6%. This does not seem a worthwhile saving for the extra complication involved.

Another possibility for compression is to increase the differentiator space to a greater extent so that each differentiator as well as containing data to bisect a composite pixel also indicates whether either or both subtrees are monotone below and need no further updating. The overhead would be three extra possible values for the differentiator, being the possible combinations of none, one or the other subtrees being monotone, should both be monotone this would have been marked by the immediately higher level in the tree. The three extra differentiator values approximately treble the increase in information in comparison with the first technique, but this remains small relative to the original 6 bits of data. As bottom level dif-

Block size horz vert	Number of Nodes in Monotone Sub-tree					
8x8	1					
4x8	2	8				
4x4	4	16	77			
4x2	8	32	154	549		
2x2	16	64	308	1098	5667	
2x1	32	128	616	2196	11334	29410
1x1	64	256	1232	4392	22668	58820
%total area	0·02	0·09	0·5	1·6	8·6	22
save/tree	62	30	14	6	2	0
total saved	62	240	1078	3294	11334	0

Figure 2.6 Some measurements of the internal structure of the binary tree produced using Knowltons method with the GIRL image.

ferentiators can now be included in the redundant node count, and one further node in each other sized MST, the count increases to about 20% of the total tree size. Again this does not seem sufficient saving to justify the increased complexity.

These MST based techniques are very sensitive to noise in the original image as any tree-wise higher level coherence is made unusable by the smallest disturbance at the base of the tree. This is likely to become a greater problem as the quantisation resolution increases, but does point to the possibility of some higher level description capable of describing monotone areas of the tree rather than complete sub-trees. Starting with a noisy image, after one combination the resulting composite pixels may have the same value, with any noise appearing in the first differentiator. It is possible the tree may be monotone for some distance above this level. This was investigated using a two-level marking scheme for the tree, differentiators with the zero difference value were marked type A unless either child were marked type A, in which case the node was marked type B. Nodes of type A would then be either isolated identical pairs or leaves of a tree, nodes of type B being internal nodes of a tree. The proportion in the complete tree of nodes of type A was 24%, and nodes of type B was 12%. These findings are very difficult to interpret, for instance, the relative proportions are consistent with the internal and leaf nodes of a fully balanced tree, which is an unlikely possibility. A more sophisticated coding scheme would be required to take advantage of this internal structure than of the the more simple MSTs, being required to describe arbitrary shaped trees. However these figures at least place a bound on the maximum possible compression using the internal structure of 36%. Once again the greater degree of complexity required would not be justified by the redundancy reductions possible using this approach.

The second possible compression technique suggested by Knowlton is the use of variable length coding for the differentiator, as the distribution of values is not uniform. The distribution is biased towards the middle range values which correspond to little or no difference between two pixel (composite or real/original) values. This was investigated and the first order entropy of the differentiator values for the GIRL image was calculated as 2·9 bits. This is comparable with the second order (two pixel) entropy of the same image, which is approximately 2·8 bits, and the entropy of the difference values produced by simple DPCM (no special action at line end) coding at 3 bits. There is some difference in code table size for these techniques, though the differentiator table is small, being the same size as the grey level set. For comparison the first order (single pixel) entropy of the image is 5·2 bits. From these results, useful compression by probability matched coding seems quite feasible, with compression of up to 50%. The usual problem associated with variable length coding of possible loss of synchronisation is ignored as being more properly dealt with by the communications channel rather than the coding scheme. Some related work on the exploitation for compression of the structure of the the information within the tree has been reported by Tamminen [1984a, 1984b].

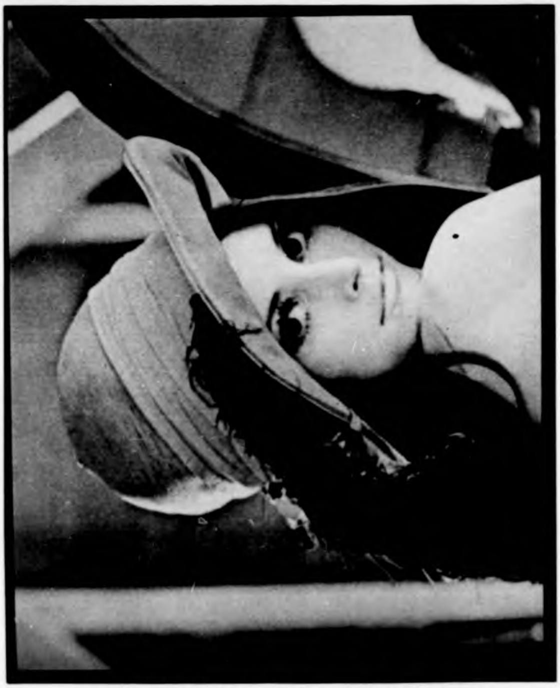
2.4. Binary-tree Implementation and Results

The basic system as described by Knowlton was implemented, no compression was attempted, nor initially any focussed update (Interactive Detailing). The ability to code an image in situ was very useful, also the property of no extra information transmission means that the coded information takes up no more storage than the original image and thus can be kept in an image store. The coding algorithm operates by exchanging pairs of data or part coded data values as in figure 2.4 and 2.5 and temporary storage is only required for one or two values at a time. A difference in the decoding strategy adopted from that of Knowlton was the starting resolution of the image, as described by Knowlton the first image would be a monotone grey level. If the display system cannot update the display sufficiently quickly to keep pace with the data input for the first few iterations, Knowlton suggests the lag be tolerated or special hardware be used. A third possibility is to commence update at some point in the sequence where the amount of work to be done per datum more nearly matches the available processing capability. For demonstration purposes it was decided that data should be made available to the decoding program at a rate equivalent to a 4800 baud serial line. Processing speed was commensurate with a starting resolution based on a composite pixel size of 8×8 real pixels, giving an overall image resolution of 64×64 composite pixels. This level of performance was possible by programming in a high level language giving some special attention to the block filling code, without recourse to extreme optimisation. This was considered to be a reasonable compromise, 4800 baud being a possible data rate as described in section 1.3 and the 64×64 resolution not too far along the progressive sequence to make invalid the progressive nature of the display. The usefulness of an image of 64×64 resolution is very much content dependent and if such an image imparts no useful information there seems no point in expending effort to reach that point in the display sequence progressively. Working with a first composite pixel size of 8×8 has further implications in accessing the display memory in a block-wise fashion due to the architecture of the experimental equipment. It is much more efficient to copy a block of display memory into processor local memory to do a complete update for that block at the level appropriate for the display iteration. The starting composite pixel size is a convenient block size for this operation, affecting the order of information supply and image update. The ordering of update of nodes at any single level in the code tree is not mentioned by Knowlton, unless for some specific focussed update, the simplest approach being some form of raster scan. Using the block access this is modified such that all the updates to a specific block are performed in succession rather than the block needing to be repeatedly fetched as the raster intersects it. The blocks are updated in a raster fashion. This gives rise to a secondary update phenomenon in that the update caused by writing a complete block is visible during the early iterations, and it is possible to follow the sequence because of the raster scan. It may be possible to break-up this pattern by using a pseudo-random updating sequence though this has not been investigated. Using a block access mechanism reduces picture activity in comparison with using the display memory for all accesses and this may be a positive feature. Use of direct access to the display memory was not investigated in detail at this stage because of its extreme penalty in speed of execution on the available hardware. This starting resolution gives seven update iterations, these can be seen in photograph 2.2. In order to fit the entire update sequence into one storage frame, only part of the image, one sixteenth, is displayed from any one intermediate image. The GIRL image has been reduced in size to match the smaller image size by averaging over a 4×4 block of real pixels. While this is in general not an acceptable method of subsampling there are no apparent visual problems in this particular instance, the sub-image shown nearest the bottom-left corner of photograph 2.2 is identical to the coding data. The progression of intermediate images starts at the top-left of photograph 2.2 and proceeds clockwise to the final image which is identical to the original. The first intermediate image has a composite pixel size of 8×8 real pixels. Subsequent composite pixels divide first vertically then horizontally as shown in figure 2.5, where this implementation was the motivation for basing discussion on an 8×8 block. The resulting composite pixel sizes are

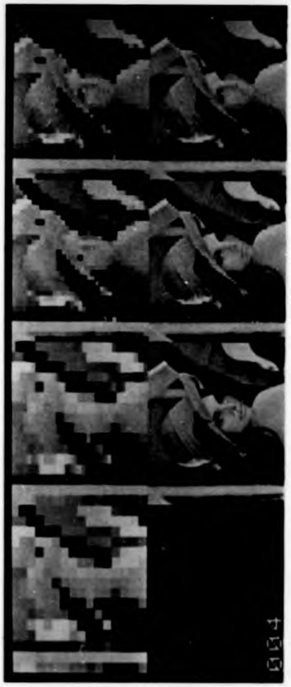
horizontal size/vertical size, 8×8, 8×4, 4×4, 4×2, 2×2, 2×1, 1×1

The peculiar pixel geometry chosen by Knowlton was not adopted because of problems in mapping onto the available display and a desire to have the final image at full available resolution for comparison purposes. The scaling-down of the image gives it finer detail (in pixel terms) than the original full image, which seems to be more locally smooth. This could be a product of the averaging used to produce the small images. It is not a problem though it should be noted that the small images might be a more severe test than the original. While the following comments on image quality are made in an informal manner it is worth remarking on a subjective evaluation issue associated with PTD. Particularly with this algorithm, towards the end of the sequence if there are no aliasing problems, the very small block structure can become invisible. Thus the total error might still be quite large but invisible depending on the picture size and viewing distance, something not always under control for informal tests.

Referring to photograph 2.2 the block structure is very visible in the first image, dominating the image and obscuring any picture content. This may be due simply to there being no picture information at all at this block size, the picture being so sub-sampled by the repeated averages. However the image does improve in an acceptable fashion and as commented on by Knowlton and demonstrated by this example, the approximate nature of the composite value with respect to the true mean does not cause unacceptable degradation of the coarser approximations. It has been assumed from the outset that this technique does work satisfactorily and no particular conclusions are drawn from pictorial results except to confirm this, with further work using this as a basis for comparison. Overall the technique is robust and easy to implement with the greater amount of effort expended in data management rather than operation of the transform. Additionally the technique meets the criterion of no redundant information transmitted, no extra resolution is required in the image store and of course the intermediate representation is directly displayable. This technique thus meets all the requirements set out in section 1.2. Some possible areas for improvement include replacing the look-up table by some computational scheme, and from a subjective viewpoint, minimising the visibility of the block structure in the coarser approximations. A higher level goal would be to develop a similar transform over four or more points to allow a cleaner implementation of quad-tree and oct-tree based algorithms.



Photograph 2.1 The original of the GIRL image, used initially in the monotone sub-tree investigation, and later as the source material for the photographically reproduced results.


Photograph 2.2 An update sequence using Knowlton's method with a starting composite pixel size of 8×8 real pixels. The sequence of sub-images proceeds clockwise from that which is top-left with the image oriented correctly. The final sub-image is identical to the source image used.

3. Two-element Transforms for Binary-tree Coding

One possible improvement to Knowlton's method, mentioned in the previous section, is the replacement of the table look-up by some simple calculation which will yield identical values to those contained in the table. The motivation for this is the very large size of the tables that can be required. For example, using 8-bit data one table contains 2^{16} one byte entries, which is 64k bytes. With the falling cost of read-only-memory, a table this size is not unreasonably expensive, but certainly from a hardware viewpoint costs more than a few bytes of program. However the look-up table technique has also been investigated for use with 24-bit remote sensing data [Hill 1983]. This would require tables of size 2^{48} entries which are completely beyond available appropriate technology.

3.1. Equivalent Algorithms for the Table Driven Transforms

Two very different algorithms have been published which permit simple calculation of the entries in these look-up tables. The first algorithm to be described is that of Packwood and Martin [1983]. The approach adopted is to split the table into four quadrants, dividing at the mid-points of the ranges of the p_0 and p_1 values. Each quadrant has in isolation a very simple structure, which can be illustrated for the composite value table with data in the range 0,N−1 and N assumed even as

$p_0 + p_1 - N/2$	$\max(p_0, p_1)$
$\min(p_0, p_1)$	$p_0 + p_1 - N/2 + 1$

Figure 3.1 has the same contents as the original Knowlton composite table but has been split slightly to make the quadrant structure clearer. Much of the complexity of the algorithm is the decision of in which quadrant a particular data point resides. This results in overall a very simple algorithm, the complexity and space requirements of which are independent of the size (number of bits) of the data. The algorithm, in two parts to calculate the composite value and differentiator for data in the range 0,N−1 is

```
composite value =
    if( $p_1 < N/2$ ) then
        { if( $p_0 < N/2$ ) then min( $p_0, p_1$ )
        else $p_0 + p_1 - N/2 + 1$ }
    else
        { if( $p_0 < N/2$ ) then $p_0 + p_1 - N/2$
        else max( $p_0, p_1$ ) }
```

The 'differentiator' table divided into quadrants is shown in figure 3.2 and an expression for its calculation is:

```
differentiator =
    if( $p_1 < N/2$ ) then
        { if( $p_0 < N/2$ ) then $p_0 - p_1 + N$
        else max( $N - 1 - p_1, p_0$ ) }
    else
        { if( $p_0 < N/2$ ) then min( $N-1-p_1, p_0$ )
        else $p_0 - p_1 + N/2 - 1$ }
```

For three-bit data as given in the various tables N=8. The tables required for decoding are identical to the coding tables as described above in section 2.2, and have similar quadrant based structure. If necessary a rotation of the tables can be used to find the other values from only the composite table. An algorithm similar to the quadrant based algorithm described above but derived independently has been published by Hardas and Srihari [1984]. Although the quadrant algorithm is satisfactory for the precise application of replacing the required look-up tables, there seems to be no link between the algorithms or expressions and the basic requirement of approximating the mean and producing a unique differentiator. If the method of generating the values in the tables is ad-hoc then the methods used for replacing the tables are doubly so.

Another algorithm has been proposed by Hill [1983] which in some ways is more contrived than that described above, though it may be more related to the method originally used to fill the table. The algorithm is known as the 'Ring Neighbour Algorithm' and is based upon consideration of the table as a set of concentric rings. Each element has a 'ring neighbour' which is found by traversing the ring containing that element, for a distance of r elements in an anticlockwise direction, where r is the ring number. A table with the rings marked is shown in figure 3.3, where the table entry at each cell is found by interchanging the row/column indices of the ring neighbour of the cell. The algorithm described here is slightly different to that described by Hill in the sense of the rotation to find the ring neighbour. This is

	7	3	4	5	6	7	7	7	7
	6	2	3	4	5	6	6	6	7
	5	1	2	3	4	5	5	6	7
	4	0	1	2	3	4	5	6	7
P_1	3	0	1	2	3	4	5	6	7
	2	0	1	2	2	3	4	5	6
	1	0	1	1	1	2	3	4	5
	0	0	0	0	0	1	2	3	4
		0	1	2	3	4	5	6	7

P_0

Figure 3.1 Quadrant split version of Knowltons composite table, this table has the same contents as shown in figure 2.1.

due to a differing interpretation of a co-ordinate pair as row/column or column/row indices. The result of Hill's transformation is a table produced by a reflection along the equal valued line of Knowlton's table. Hill stresses that his transformation is an involution (is reflexive) and also that this would be the case for Knowlton's table if it was reflected in the appropriate axis. Neither of these assertions is novel and it is clear that that some confusion has arisen from a trivial difference in procedure. These minor differences aside it is obvious that some different structure within the table is being exploited by this algorithm to that used by the quadrant based algorithm. The unit width rings used hint at some relationship with the filling algorithm but once again there seems no relevance to the fundamental mean and difference problem.

This un-clear relationship between these algorithms and the mean and difference problem makes it very difficult to generate from them some algorithm for a greater number of points. Based on the quadrant algorithm, a four point transform would comprise sixteen fractional parts, and if the same sort of structure exists, the patterns within should not be difficult to describe. However this route would surely lead to a very limited class of transforms with fixed basis vectors, I spent little time trying to construct such

p_1 \ p_0	0	1	2	3	4	5	6	7
7	0	0	0	0	0	1	2	3
6	0	1	1	1	1	2	3	4
5	0	1	2	2	2	3	4	5
4	0	1	2	3	3	4	5	6
3	1	2	3	4	4	5	6	7
2	2	3	4	5	5	5	6	7
1	3	4	5	6	6	6	6	7
0	4	5	6	7	7	7	7	7

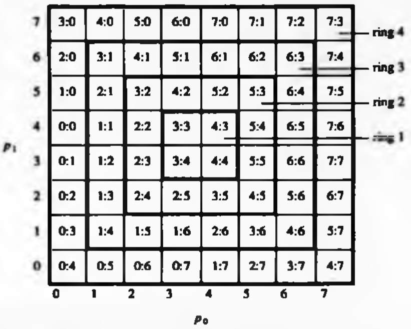
Figure 3.2 Quadrant split version of Knowltons differentiator table, this table has the same contents as shown in figure 2.2.

transforms with no useful result. Use of the ring algorithm over a greater number of points is even more obscure and no attempt has been made to extend this work. In the context described by Knowlton these algorithms can always replace the table where the ability to execute some simple program exists. This removes the only impediment to using the whole PTD system that exists in circumstances with large data spaces.

3.2. New Mean/Difference Transform

A very different algorithm that more accurately mimics the behaviour of a linear transform has been described to me by Prof. M.S. Paterson. Adopting the notation of t_0 for the approximate mean, and t_1 for the difference value, for data in the range $0, 2^r-1$ with $\&$ as the bitwise and operator, assuming a twos-complement representation for negative numbers that has the 2^rth bit set, the algorithm is

$$t_1' = p_1 - p_0 + 2^{r-1}$$
$$t_0' = p_1 + p_0 + \mathrm{abs}\left(2^r \;\&\; t_1'\right)$$
$$t_0 = \left\lfloor \frac{t_0'}{2} \right\rfloor \mod 2^r$$



p_1 \ p_0	0	1	2	3	4	5	6	7
7	3:0	4:0	5:0	6:0	7:0	7:1	7:2	7:3
6	2:0	3:1	4:1	5:1	6:1	6:2	6:3	7:4
5	1:0	2:1	3:2	4:2	5:2	5:3	6:4	7:5
4	0:0	1:1	2:2	3:3	4:3	5:4	6:5	7:6
3	0:1	1:2	2:3	3:4	4:4	5:5	6:6	7:7
2	0:2	1:3	2:4	2:5	3:5	4:5	5:6	6:7
1	0:3	1:4	1:5	1:6	2:6	3:6	4:6	5:7
0	0:4	0:5	0:6	0:7	1:7	2:7	3:7	4:7

ring 4
ring 3
ring 2
ring 1

Figure 3.3 This table illustrates the 'Ring Neighbour Algorithm', the cell contents are of the form composite:differentiator. To find the contents traverse the ring anti-clockwise by the ring number and interchange the co-ordinates.

$$t_1 = t_1' \mod 2^r$$

and the inverse transform

$$p_0 = t_0 - \left\lfloor \frac{t_1}{2} \right\rfloor + 2^{r-2} \mod 2^r$$

$$p_1 = t_0 + \left\lceil \frac{t_1}{2} \right\rceil - 2^{r-2} \mod 2^r$$

The table for t_0 values of this transform is shown in figure 3.4 and for the t_1 values in figure 3.5. In both tables where a cell has contents of the form a:b the a value is the primed value, in figure 3.4 this would be t_0'. All the values are of this form in the t_0 table due to the division by two of t_0' to obtain t_0 whereas many values of t_1' yield t_1 unchanged. The normal limits of these tables are shown by the double bars and the elements emboldened can be considered to have wrapped-around because of the modulus operators. Apart from the points marked in bold, t_0 accurately approximates the mean value of p_0 and p_1, the bold points are completely incorrect approximations. The t_1 values have no special numerical significance but serve only to differentiate the t_0 values much as does a Knowlton style differentiator. An

								15:7			
							13:6	14:7	15:7		
7		15:7	16:0	17:0	18:1	11:5	12:6	13:6	14:7	15:7	
6		14:7	15:7	16:0	9:4	10:5	11:5	12:6	13:6	14:7	15:7
5		13:6	14:7	7:3	8:4	9:4	10:5	11:5	12:6	13:6	14:7
4		12:6	5:2	6:3	7:3	8:4	9:4	10:5	11:5	12:6	
p_1 3		3:1	4:2	5:2	6:3	7:3	8:4	9:4	10:5		
2	17:0	2:1	3:1	4:2	5:2	6:3	7:3	8:4	17:0		
1	16:0	1:0	2:1	3:1	4:2	5:2	6:3	15:7	16:0		
0		0:0	1:0	2:1	3:1	4:2	13:6	14:7	15:7		
			16:0	17:0	18:1						
				16:0							
		0	1	2	3	4	5	6	7		
				p_0							

Figure 3.4 Table of approximate mean values t_0 for Patersons transform, some values are shown as $t_0':t_0$.

important feature of this transform is the consistent use of the floor and ceiling operators to produce the approximations to the true linear vectors. Another feature is the use of the modulus operator to shape the vectors and fit their length to the appropriate space boundaries. This particular algorithm was not investigated in detail due to its seeming complexity and lack of development path but a similar technique is developed later in the text for which this is a useful comparison. It does show that other approximations are possible which are much more closely related to the required mean and difference representation.

								7			
							7	6	5		
7		11:3	10:2	9:1	8:0	7	6	5	4	3	
6		10:2	9:1	8:0	7	6	5	4	3	2	1
5		9:1	8:0	7	6	5	4	3	2	1	0
4		8:0	7	6	5	4	3	2	1	0	
p_1 3		7	6	5	4	3	2	1	0		
2	7	6	5	4	3	2	1	0	−1:7		
1	6	5	4	3	2	1	0	−1:7	−2:6		
0		4	3	2	1	0	−1:7	−2:6	−3:5		
			2	1	0						
				0							
		0	1	2	3	4	5	6	7		
				p_0							

Figure 3.5 Table of difference values t_1 for Patersons transform, some values are shown $t_1{}'$:t_1.

4. Interpolation and Focussed Update for the Subjective Improvement of the Binary-tree Methods

The second of the possible improvements to the basic techniques described by Knowlton relates to the subjective quality of the intermediate images. The progressive display techniques discussed thus far have utilised intermediate images comprised of composite pixels of various sizes but of monotone grey level. The boundaries of these composite pixels are clearly visible, and as they are of uniform size at each iteration a very strong block structure is evident. This block structure can be made less visible by using some form of interpolation. For continued decoding the only requirement is that the 'value' of the composite pixel must in some way be made available to the next iteration of the decoding algorithm. There are two ways this can be achieved. A single individual pixel may be used to retain this value while all other pixels within the composite pixel can be modified. Alternatively all individual pixels within the composite are modified but calculated in such a way that the value of the composite pixel may be recalculated for use in the next iteration. Of these two, only methods based on retaining the pixel value were investigated since they seemed to be the simplest.

4.1. Bilinear Interpolation

First investigations were based on bilinear interpolation [Pratt, 1978, p115]. This can be utilised with the block access method based on the first composite pixel as described in section 2.4, where display access is based on an 8×8 array of display pixels. The method is illustrated in figure 4.1 where the enclosed square outline represents the boundary of an access block. For the first decoding iteration a block has associated with it only one value which is placed in the real pixel marked B. The blocks are accessed in a raster scan fashion from bottom-to-top and left-to-right so that the values in the blocks to the left and below are already in place. This ignores the effects at the left and bottom boundaries which will be discussed later. The interpolation is done in two phases of linear interpolation performed in orthogonal directions.

The first phase is to find interpolated values for the pixels marked a0 and a1 in figure 4.1. The lower set of values will have been generated when the lower block was interpolated and are available from display memory. The upper set of values needs to be calculated and the value of pixel A must be fetched from

A	a0	a0	a0	a0	a0	a0	a0	B
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
D	a1	a1	a1	a1	a1	a1	a1	C

Figure 4.1 Bilinear interpolation with only one data value for an 8 × 8 block 'at' point B. Interpolation is in the order a0,a1,b0....b7.

display memory. By definition the interpolated values are linear combinations of the end point values but linear combination would be an expensive way of calculating them. It is more efficient to use the incremental approach of accumulation from one end point by adding some constant. This constant is the pixel to pixel difference found by subtracting the end point values and dividing the result by the number of pixel differences between them, which is 8. This binary division can be performed by shifting and the accumulation needs three bits of fractional part for full accuracy. The amount of processing can be reduced by keeping the interpolated values for each end-to-end difference in a table. The number of possible end-to-end differences is twice the range of the data, with both ascending and descending sequences. The table need only contain half of these if the mechanism exists to either add or subtract the table value to the end point value. For 8-bit data the table needs to contain 256 sequences. Each sequence needs strictly to be only 7 elements long as the eighth element will always be the sample value. It is however much easier to access the table if the sequences are kept to 8 elements. The table then has 2k entries of the same size as the data, for 8-bit data each entry will be one byte.

The second phase of the interpolation is to generate the values marked b0,b1,b2...b7 (in figure 4.1). This requires 8 separate interpolations performed top-to-botton, using as end points values in the a0 and a1 interpolations. These interpolations are again over 8 elements. The block is then fully interpolated. There is a slight anomaly in this interpolation system at the bottom and left edges of the image. At these edges there are no blocks below or to the left respectively. As implemented, accesses in these areas are detected as special cases. For blocks adjacent to the left boundary, point 'A's value is taken from point B and likewise point 'D's from point C. Similarly at the bottom edge of the image, point C from point B and point D from point A. The corner block has all three remaining values duplicated from point B. These anomalies do not cause any significant visible problems.

At the next decoding iteration a single differentiator value is decoded with the composite value of the block. The composite value for the block is available from display memory at position B. From these two values are generated two new values for the block which will be bisected horizontally to produce two smaller composite pixels. The placing of the composite pixel values in real pixels is shown in figure 4.2 at points B, E. The block to the left has a decoded rather than interpolated value at point F.

A	a0	a0	a0	a0	a0	a0	a0	B
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
	b0	b1	b2	b3	b4	b5	b6	b7
F	a2	a2	a2	a2	a2	a2	a2	E
	b8	b9	b10	b10	b11	b13	b14	b15
	b8	b9	b10	b10	b11	b13	b14	b15
	b8	b9	b10	b10	b11	b13	b14	b15
D	a1	a1	a1	a1	a1	a1	a1	C

Figure 4.2 Bilinear interpolation with two data values per block 'at' B,E. Interpolation is in the order a0,a1,a2,b0.....b15.

The first phase of interpolation now has an additional sequence marked a2 between points E and F. The second phase comprises 16 separate interpolations over 4 elements rather than 8 interpolations over 8 elements. If a table is being used for the interpolation values, the correct value can be found by subsampling by two the values in the table. As decoding proceeds the same principles can be used iteratively. The value for the composite pixel is placed in the top right corner of the pixel, and the interpolation table is subsampled by increasing factors as the required interpolation sequences become shorter. At all times a complete access block is copied from the display memory, fully updated according to the iteration level of the processing and copied back to the display memory. It is simple and useful to retain the right-most column of a block to form the left edge of the next interpolation proceeding right in a raster scan fashion.

The result of using this interpolation scheme is shown in photograph 4.1. The underlying data is the same as that shown in photograph 2.2. It is generally agreed that the interpolation gives a subjective improvement, though no attempt has been made to quantify this. With some qualification regarding the extra processing required, this form of interpolation seems to work well with no hidden problems or drawbacks. It can be recommended for use whenever update based on Knowlton's work is used. Approximately twice as much processing time is required to perform the complete update using interpolation as is required when composite pixels are filled to a monotone value. This performance was achieved only with some careful optimisation. With the image processing equipment available to the author, the starting composite pixel of size 8×8 real pixels could only be interpolated in a time equivalent to receiving data at 2400 baud. If higher rates are required the starting composite pixel size must be reduced. Neither of these changes is realistic and for interpolation to be feasible some increase in efficiency is required. This motivated a search for some simpler interpolation algorithm. The problem with bilinear interpolation as used is not the final value calculation, which can be reduced to the addition of a table element to an end point point value. Rather it is the management overhead of activity such as moving from sequence to sequence. Inevitably sequential access of a two dimensional array by both row and column is expensive as it can only be stored by row or column. This requires explicit address calculation rather than simple linear stepping through a vector. A second problem with the block access interpolation is that it depends heavily on the raster scanning of the blocks. This makes it impossible to use with any pseudo-random update scheme as suggested in section 2.4. If focussed update is required, it may be possible to use a block based regular scanning system. Traversing a ring, of a concentric squares update scheme, there will be an already interpolated block inside the ring, and an updated block 'behind' the scanning point. These blocks would give the two already processed edges required by the algorithm described above, but the overall management needed would be increased as the orientation of these edges would not be fixed. It is the control and management aspects of this interpolation method that would present problems if a hardware or microprogrammed interpolation system were required.

4.2. Four-way Mean Interpolation

With these issues in mind, in conjunction with Prof. M.S Paterson a different algorithm was proposed [Packwood & Martin 1984]. The algorithm which is illustrated in figure 4.3 is based upon calculating the interpolated value of a point as the mean of four neighbours. This is performed in a top-down iterative fashion in a series of passes over the whole image. Thus in figure 4.3 the starting data is available at the points marked 0 which correspond to the points marked A,B,C,D in figure 4.1. From these four points the value of the single point marked 1 is calculated as the mean (the algorithm will be referred to as the Four Way Mean FWM algorithm). At the next iteration the points marked 2 are calculated as the mean of two points marked 0 and two points marked 1 (note that one of these points is off the diagram). At subsequent iterations the points marked 3 and 4 are calculated, for clarity the remaining points marked 5 and 6 are not shown but would be interpolated in the same fashion. This shows how the square composite pixel can be filled in using the interpolation scheme.

At the next iteration of the decoding algorithm the square composite pixel will be bisected horizontally and there will be a new value associated with each half. By placing the value for the top half at position 0 and the value for the bottom half at position 1 the algorithm can then proceed as described with the calculation of the number 2 points. This requires a certain amount of algorithmic licence in the placing of the value at position 1. This causes some spatial distortion of the image though it is not visually obvious as a problem. The situation is more clear at the next decoding iteration when values are available for the pixels marked 0,1,2 and the correct value can be placed in the corner of all the composite pixels. With this algorithm it is easier from a data management viewpoint to assume that all points off the image have the

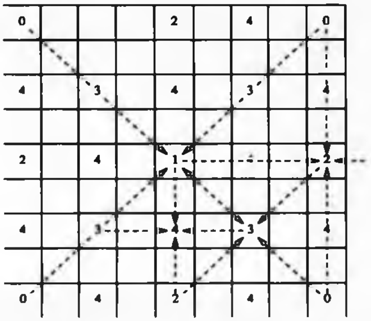


Figure 4.3 Order of interpolation for the Four Way Mean Algorithm. The points are interpolated in the order 0,1,2,3... using transmitted data at the points where it is available.

value zero. This will cause the image to fade to black around its perimeter.

The result of using this interpolation mechanism is shown in photograph 4.2 with the same underlying data as photographs 2.2 and 4.1. An important feature of this sequence is the dots, both white and dark that can be seen in the earlier images in the sequence. These arise because the behaviour of this interpolation scheme, although simple to describe, is not as innocuous as it might first appear. An interpolation algorithm has an equivalent reconstruction filter with an associated impulse response. The impulse response of the interpolation algorithm is easy to calculate. It has an exponential form with significant value out to a distance of 1·5 access blocks. More important is the cusp like nature of the response at the sample point producing a severe discontinuity. It seems this discontinuity is responsible for the visible dots in photograph 4.2. The dots do become less visible as the decoding sequence progresses. Another point of note is that the scheme does not deal any better with the strong boundary in the middle right of the sub-image than the bilinear algorithm. Unfortunately with the equipment available it is impossible to assess any improvement in time required to interpolate the image. This is because the interpolation is based upon point by point access of the image which carries an extreme overhead in accessing the frame store. The point by point algorithm in operation actually takes much longer than the access block based bilinear interpolation due to this overhead.

The bilinear algorithm can be reformulated such that the only overt mathematical operation required is the taking of the mean of two elements. Much more difficult to enumerate is the amount of arithmetic required to do address manipulation. This makes the reduced complexity of the four way mean algorithm more questionable since it is equally difficult to analyse its addressing overhead. Some of the data management issues depend on whether an interpolation is performed locally as in the block access method, or globally over the whole image. It is possible to use the four way mean algorithm in a local fashion, but more difficult than use in a global manner. This is due to the large number of pixels which should be changed when any one pixel is updated. Thus interpolation must follow at a distance behind update to ensure correct behaviour. An easier approach with the four way algorithm is to update the whole image before the interpolation and then interpolate level by level over the whole image. This greatly increases the amount of visible image activity over the block access method. The slow speed of execution of the program makes it very difficult to judge what subjective effects this visible activity would have if more suitable (better random pixel access) equipment were available.

4.3. Focussed Update

Despite these problems, as most of the software was implemented it was decided to investigate briefly a focussed update based upon the four way mean algorithm. The first problem this presents is that the state of each pixel as data or interpolated value must be readily available to the interpolating process. This prevents the overwriting of received data by an over-zealous interpolating algorithm. The option of storing one bit to represent this state explicitly was too difficult to implement and would violate the requirement of increased storage. It was decided to combine the existing software base with focussed update by setting the spatial granularity of changes in update level to be an access block. The update level of a block could then be used to decide to interpolate any given pixel at any level within that block. It is possible to calculate the update level of a block from its spatial position within the image without having to store it explicitly. The approach suggested by Knowlton was followed, using concentric rings two access blocks wide, with resolution decreasing radially by one update level per ring. The rings are two access blocks wide to give an acceptable overall change per complete update, the more obvious choice of one block gives a very slow rate of change. The result can be seen in photograph 4.3. The number of data values used for each image is :

1, 26, 108, 304, 711, 1492, 3062, 6146

This corresponds to a single central access block with concentric squares increasing in width by four access blocks. For example the second sub-image comprises 25 blocks each having been updated with one value at that iteration. Together with the one value for the one block at the first iteration this gives a total of 26 values. The increase in data values per image which starts as a high relative value rapidly declines to an approximate doubling per image at the end of the sequence. Although by no means conclusive, considering just this one example and the crude methods used, focussed update and interpolation seem to operate well in combination. For such as a head and shoulders image, the reduction in resolution away from the centre of interest gives a useful reduction in the amount of data required and the reduction is well disguised by the interpolation used.

4.4. Piece-wise Cubic Interpolation

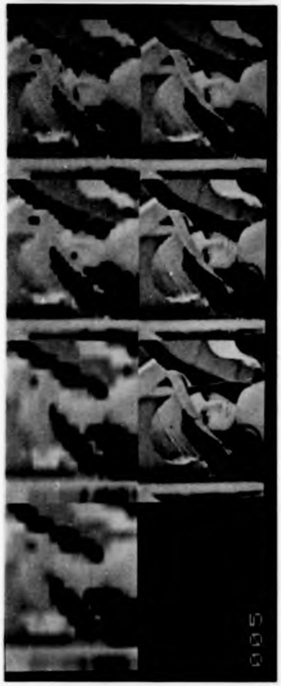
Some other work has been reported on interpolation for Knowlton style update [Sanz,Munos & Garcia 1982]. Bi-cubic splines were used but found to require excessive computation, cubic convolution interpolation [Rifman & Mckinnon 1974] was also investigated with more useful results. Cubic convolution interpolation is based on a piece-wise cubic approximation to a sinc pulse given by:

$$f(x) = \begin{array}{ll} 1-2|x|^2+|x|^3 & 0 \le |x| \le 1 \\ 4-8|x|+5|x|^2-|x|^3 & 1 \le |x| \le 2 \end{array}$$

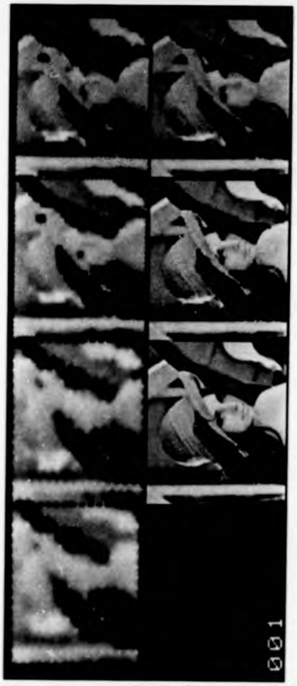
To use this in a conventional sense as an interpolation filter would seem to require excessive computation. It is possible to use table look-up to replace some of the computation, requiring a table for a quadrant of the function. For the worst case sub-sampling of 8×8 blocks the table needs to be of side 16 elements as the function is defined to $x \leq 2$. The table size depends on the data set size, for 6-bit data the table has $16 \times 16 \times 64$ entries, which is 16k entries, for 8-bit data the table size is 64k entries. For 6-bit data the table size is quite acceptable, but becomes excessive for 8-bit data. If a table were used, then without performing detailed calculations per point, the effect of sixteen sample points must be considered. The taking of a table value and summing the result for each sample does not appear prohibitively expensive computation, but extra arithmetic will be required to calculate the position in the table for each sample point. Special action needs to be taken at the image boundaries. Use of a table takes no advantage of the simplifying nature of the cubic approximation as the table could as easily contain values calculated from the true sinc function. The most obvious problem with using the approximation directly is calculating the radial distance from the sample point. The radial distance has a range of 22 values. To use tables for evaluation of the exponentiation and multiplication of individual terms of the formula requires four tables with a total size of approximately one third of the single table described above. This techniques requires additional complexity to deal with the different sample spacing that occurs at alternate stages of a Knowlton update. I have not implemented this interpolation method in any form but would expect the results to be good at the price of increased computation. Considering that all the interpolation algorithms stretch the computation budget this one seems to be not possible without extensive hardware support.

A technique for reducing the visibility of the block boundaries, by interleaving some elements of adjacent blocks has been reported by Pearson and Whybray [1984]. As with interpolation, this technique is largely independent of the coding algorithm. The technique has been extended by the investigation for progressive display of the relationship between the quantity of information used to update a block and the degree of interleaving required for optimum picture quality [Cazin, Pearson & Whybray 1985]. Spriggs and Nightingale [1986] have reported some work in which interpolation is used together with a quad-tree approach.

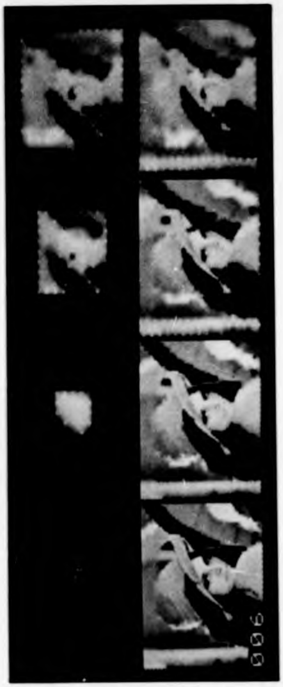
The four way mean algorithm has questionable subjective and computational performance and thus little to recommend it. The visible effects of large area level by level interpolation remain uninvestigated and it is not clear how interpolation could be used with focussed update in any simple fashion. The block access method using bilinear interpolation gives a useful increase in image quality if there is sufficient processor resource to cause no reduction in data through-put. The method has no adverse properties other than the increased processing requirement.



Photograph 4.1 An update sequence based on Knowlton's method, enhanced using bilinear interpolation over an access block of 8 × 8 pixels. The sequence is clockwise from top-left with the final image identical to the source image used.



Photograph 4.2 An update sequence based on Knowlton's method, enhanced using the Four Way Mean interpolation scheme. The sequence is clockwise from top-left with the final image identical to the source image used.



Photograph 4.3 A focussed update sequence, based on Knowlton's method enhanced using the Four Way Mean interpolation scheme. For each new image, all the blocks (8×8 pixels) are updated one level in the tree, and an extra circum-square of blocks each with one data value is added. The final image contains approximately 27% of the source image data and thus corresponds very nearly to the 5th image in the earlier photographic sequences.

5. Transform Coding for PTD

5.1. Transforms over more than Two Elements

The techniques discussed in the preceding sections have been criticised for not producing the best quality images for the quantity of information transmitted, which can be improved using transform coding [Lohscheller 1984, Baronetti, Guglielmo & Riolfo 1979, Bisherurwa 1981,1982,1983]. Transform coding for PTD has been investigated, but usually differs from work similar to Knowlton's in that it is not exactly invertible. Knowlton's work is based loosely on a two point linear transform and as stated in section 2.4 there are reasons to extend this for specific PTD applications. To perform transform coding in the more general sense requires a change of emphasis. The number of values over which the transform operates needs to be increased to the block sizes commonly used for image coding such as eight or sixteen. With this increase in number of values comes the requirement to work with a large set of basis vectors. With two values there is only one useful set of basis vectors, the mean and difference. The applications mentioned earlier with quad-trees and oct-trees would require only extended versions of this transform, being based on sets of differences. These sets of differences would be characterised by basis vectors with identical coefficients apart from their signs, such basis vectors are characteristic of the Hadamard transform and of the Haar transform [Shore 1973, Harmuth 1969]. The shape of the basis vectors of the Haar transform is such that when used over a two-dimensional area, the transform behaves similarly to a hierarchical application of a differencing transform over fewer points. More general transform coding would require arbitrarily related vector coefficients. The non-invertible property of transform coding derives from the quantisation operation, the transformed coefficients are approximated to some more limited set of values. The requirement for this operation needs to be closely investigated so that it can be replaced or in some way made redundant so that the coding can be made exactly invertible.

Knowlton's method is essentially transform based, and the potential improvements in use of a more general transform over more points may not be clear. One difference is the way in which the quantity of information per update iteration varies. Knowlton's method has an exponential increase in information per update with the effect that the time for each update increases by a factor of two. In a straightforward implementation of a transform technique each successive update would require the same amount of additional information and thus take the same time. There is no evidence to suggest whether one of these is subjectively preferable to the other. The potential for improved quality is the greater degree of control available in recreating different patterns in the image, that is possible with transform coding. Effectively the only pattern usable by two point coding is the linear slope across a block, which is reproduced well with interpolation. With transform coding it is possible to coarsely quantise different coefficients to spread the update across several basis vectors for improved quality. This should also be possible with Knowlton's method if differentiators where initially transmitted at lower quantised resolution, but there is little formal basis for doing this. There may also be potential for increased data compression with transform coding, depending on the probability distributions of the transform coefficients. Transform methods for progressive display were first suggested by Sloan and Tanimoto [1979], mentioning the Fourier and Hadamard transforms, though no work was reported. The first real results were obtained by British Telecom [Nicol, Fenn & Turkington 1980] as part of the Picture Prestel investigation. Picture Prestel was developed to have the facility in addition to normal Prestel capabilities of inserting a small colour picture into a Prestel page. The Hadamard transform was used with an 8×1 block-size giving potentially eight updates, one per coefficient. Little more information is available than this, no doubt in part due to the commercial sensitivity of the work. One important omission is any mention of the quantisation scheme used, though in conversation B. Fenn suggested that a scheme similar to that developed at Essex University by Ghanbari and Pearson [1978] was used. This underlines one of the problems of transform coding, it is really insufficient to know simply that a Hadamard transform is being used without knowing the quantisation strategy also. The choice of quantisation strategy is in general still the subject of theoretical study with respect to distortion measures. Progressive display is further complicated by the method of intermediate storage of the image, transform coding techniques will produce intermediate images with non-integer pixel values. No information appears to be available concerning how the intermediate images for Picture Prestel were stored. Picture Prestel hardware is described as having only a small amount of memory, and it is not clear if the transform method was ever implemented on the hardware that was initially developed to demonstrate DPCM techniques.

5.2. The Hadamard Transform Coefficients

The issues relating to intermediate storage can be illustrated by reference to the Hadamard transform. These are more easily demonstrated with this transform due to the binary fractional nature of the coefficients. Consider the matrix H_8 which shows the sign changes of the Hadamard transform of order 8, +/– indicating terms of value +1/–1.

$$\begin{bmatrix} + & + & + & + & + & + & + & + \\ + & + & + & + & - & - & - & - \\ + & + & - & - & - & - & + & + \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & - & - & + & - & + & + & - \\ + & - & + & - & - & + & - & + \\ + & - & + & - & + & - & + & - \end{bmatrix} \quad \mathbf{H_8}$$

For data blocks of size 8 elements by 1 element with an individual element denoted p_j the transform coefficients q_i are given by

$$q_i = \frac{1}{\sqrt{8}} \sum_{j=0}^{7} p_j \, H_{i,j} \quad i = 0,7$$

In this case the rows of the matrix H_8 together (with the factor $1/\sqrt{8}$) form the basis vectors of the transform which is both orthogonal and unitary. The inverse of the matrix H_8 is then its transpose, however because the matrix is symmetrical the transformation is reflexive

$$p_i = \frac{1}{\sqrt{8}} \sum_{j=0}^{7} q_j \, H_{i,j} \quad i = 0,7$$

Sometimes the Hadamard is described in a non-unitary form as

$$q_i = \frac{1}{8} \sum_{j=0}^{7} p_j \, H_{i,j} \quad i = 0,7$$

$$p_i = \sum_{j=0}^{7} q_j \, H_{i,j} \quad i = 0,7$$

where the factor $1/\sqrt{8}$ has been moved back from the inverse transform to the forward transform. From an information viewpoint the position of these factors is irrelevant, it does not affect the number of possible values the transform coefficients may take. The limits of the range of the transform coefficients are easily calculated. Disregarding any multiplicative factor, the maximum value is generated using the first (zeroth) basis vector and is eight times the maximum value of the data. However care must be taken here, if the data is in the range 0,(n−1) having n distinct values, taking $\log_2 n$ bits to represent the value, the maximum value of the sum in this case is 8(n−1). The number of possible values of the sum is not eight times the number of possible values of each component, thus requiring just less than three extra bits to represent the sum. The range of the remaining coefficients is +/− four times the maximum value of the data, but again the number of possible different values is 8(n−1). If the first and remaining coefficients are not considered separately then the number of possible values is greater than this to meet the differing extremes. Because of the (n−1) factor even an exact coding technique might introduce a small amount of extra redundancy. The difficulties of describing how these coefficients inter-act to produce an inverse transform result inside the input space can be inferred from the description of the simple two point case (appendix A). The two point case also shows how the extra precision required for the transform coefficients is independent of any multiplicative factor, there is no difference in information between an accurate sum or mean. The additional precision of the coefficients is redundant because the number of described possible points is far greater than the set of required points. As with the mean and difference transform, fractional results are possible, and the range of reachable values greater than the input range. In order to reduce the amount of information required to describe the transform coefficients, to the same quantity as the input data or lower, the coefficient is quantised. That is the coefficient is approximated by one of a predetermined smaller set of values. This leads to a loss of information and the result may be an error in the final image depending on the way the errors sum and any final rounding. The quantisation process is usually designed to minimise the subjective effects of any such error. This error may be statistically small but is nonetheless present and so methods using quantisation do not meet the exact invertibility criterion. If the quantisation stage is omitted and the coefficient transmitted at full resolution then with the H_8 transform, three extra bits per coefficient are required. This approach would cause failure to meet the 'no extra information' criterion.

As mentioned above the range of the intermediate data values for progressive display can exceed the range of the data, this is essentially a geometric problem. For progressive transmission, as each coefficient is received its contribution to the final image is calculated and added to that already available. Again using the Hadamard transform and the definition of H_8 given above

$$h_{i,j} = \frac{1}{\sqrt{8}} H_{i,j}$$

$$\begin{bmatrix} p_0 \\ \cdot \\ \cdot \\ p_n \end{bmatrix} = \begin{bmatrix} h_{0,0} & \cdot & \cdot & h_{0,n} \\ \cdot & \cdot\cdot & \cdot & \cdot \\ \cdot & \cdot\cdot & \cdot & \cdot \\ h_{n,0} & \cdot & \cdot & h_{n,n} \end{bmatrix} \begin{bmatrix} q_0 \\ \cdot \\ \cdot \\ q_n \end{bmatrix}$$

$$= q_0 \begin{bmatrix} h_{0,0} \\ \cdot \\ \cdot \\ h_{n,0} \end{bmatrix} + q_1 \begin{bmatrix} h_{0,1} \\ \cdot \\ \cdot \\ h_{n,1} \end{bmatrix} + \ldots q_n \begin{bmatrix} h_{0,n} \\ \cdot \\ \cdot \\ h_{n,n} \end{bmatrix}$$

and rewritten due to the symmetry of the transform matrix

$$\begin{bmatrix} p_0 \\ \cdot \\ \cdot \\ p_n \end{bmatrix} = q_0 \begin{bmatrix} h_{0,0} \\ \cdot \\ \cdot \\ h_{0,n} \end{bmatrix} + q_1 \begin{bmatrix} h_{1,0} \\ \cdot \\ \cdot \\ h_{1,n} \end{bmatrix} + \ldots q_n \begin{bmatrix} h_{n,0} \\ \cdot \\ \cdot \\ h_{n,n} \end{bmatrix}$$

where the column vectors are more clearly rows of the transform matrix and the basis vectors of the transformation. It is not immediately clear from this formulation but if the multiplicative factors are considered the resolution of the intermediate display of 1/8 can be calculated. Less easy to calculate is the possible range of the intermediate display, though it is easy to generate examples with pathological data that illustrate the behaviour. For example as the reconstruction example in figure 5.1 shows, the intermediate values can be negative; the example is based on a data range of 0,7. In this reconstruction however the intermediate sequence has no value greater than the greatest value in the data sequence, this can be demonstrated in a more extreme example, shown in figure 5.2. This behaviour should not be unexpected, intuitively there can be values in the reconstruction sequence that are greater than the maximum value of the original data set. However the absolute maximum data value is not built into the transform in any way so if it occurs in the data then a greater value may occur in the reconstruction sequence.

As has already been mentioned an advantageous place to store the intermediate images is in the display memory and to do this it is at least necessary to know the limits of the values to be stored. To find these limits there seems to be no better method than one close to enumeration. Since the transform coefficients are linear combinations of the data values and the reconstruction points linear combinations of the coefficients it follows that the most extreme behaviour occurs at the vertices at the limits of the data space. For an eight point transform there are only 2^8 vertices so the enumeration is not expensive nor

data	0·000	0·000	0·000	0·000	0·000	0·000	0·000	7·000
code	0·875	−0·875	0·875	−0·875	0·875	−0·875	0·875	−0·875
0	0·875	0·875	0·875	0·875	0·875	0·875	0·875	0·875
1	0·000	0·000	0·000	0·000	1·750	1·750	1·750	1·750
2	0·875	0·875	−0·875	−0·875	0·875	0·875	2·625	2·625
3	0·000	0·000	0·000	0·000	0·000	0·000	3·500	3·500
4	0·875	−0·875	−0·875	0·875	0·875	−0·875	2·625	4·375
5	0·000	0·000	0·000	0·000	1·750	−1·750	1·750	5·250
6	0·875	−0·875	0·875	−0·875	0·875	−0·875	0·875	6·125
7	0·000	0·000	0·000	0·000	0·000	0·000	0·000	7·000

Figure 5.1 Hadamard forward and inverse transform example showing that intermediate values can be negative.

data	0·000	7·000	7·000	0·000	0·000	7·000	7·000	7·000
code	4·375	−0·875	0·875	−0·875	−2·625	−0·875	0·875	−0·875
0	4·375	4·375	4·375	4·375	4·375	4·375	4·375	4·375
1	3·500	3·500	3·500	3·500	5·250	5·250	5·250	5·250
2	4·375	4·375	2·625	2·625	4·375	4·375	6·125	6·125
3	3·500	3·500	3·500	3·500	3·500	3·500	7·000	7·000
4	0·875	6·125	6·125	0·875	0·875	6·125	9·625	4·375
5	0·000	7·000	7·000	0·000	1·750	5·250	8·750	5·250
6	0·875	6·125	7·875	−0·875	0·875	6·125	7·875	6·125
7	0·000	7·000	7·000	0·000	0·000	7·000	7·000	7·000

Figure 5.2 Hadamard forward and inverse transform example showing how intermediate values can be greater than the range of the data.

difficult to perform. The reconstruction sequence of a vertex is calculated and each point value compared to find the maximum and minimum values at each position in the vector. For the Hadamard transform using a normalised co-ordinate system with maximum value one then the maximum value in the reconstruction is 1·375 and the minimum value −0·375, these values occurring for all positions in the vector. The two symmetries here, that of all positions in the vector having the same maximum and minimum, and the under and over-range extension being the same have not escaped notice but have not been investigated further. In order to exactly reconstruct the original data, for data normalised to the range 0,1 the intermediate store must be capable of storing values in the range −0·375, 1·375. This is less than double the range of the data and so would require one extra bit to resolve the increase.

This single extra bit representation can only be realised if some mapping is possible between the intermediate store and the display. Without this it is clearly impossible to display values that are less than zero or greater than the maximum value of the display hardware. A simple look-up table is the most elegant solution to this problem. Less elegant would be sign and magnitude bits with the simple mapping of forcing all negative values to zero, detected by the sign bit, and all over-range values to the maximum as detected by the most significant bit. If a look-up table were used then a simple clamping algorithm could be adopted or some attempt to compress the information into the display range made. It is not clear if any visibly useful information would be gained by doing this, though geometrically each additional coefficient will improve the approximation. Attempts to store the overflow and sign information without affecting the display result in severe wrap-around effects when the value goes over-range though the effects can be less pronounced when the value is negative depending on the representation used for negative numbers. The alternative solution to these problems is to store intermediate images somewhere other than the picture store, requiring additional memory equivalent to the picture store. This might be expected to be a more expensive solution than those proposed above.

6. Sine and Cosine Transforms for PTD

Some work on progressive picture transmission has been reported by Bisherurwa [Bisherurwa & Coakley 1982] using both sine and cosine transform variants, and by Lohscheller [1984] based on the cosine transform.

6.1. Computation of the Sine and Cosine Transforms

Bisherurwa seems to have been originally attracted to use of the Even Discrete Sine Transform (EDST) because of its reduced computational complexity in some forms in comparison with the Discrete Cosine Transform (DCT). A sparse matrix factorisation of the DST has been reported [Bisherurwa 1981] which over seven elements requires only six multiplications and eighteen additions/subtractions. The best factorisation for the DCT over eight elements is given as 14 multiplications and 26 additions/subtractions. The EDST defined by Jain [1979] is

$$X(u) = \sqrt{\frac{2}{N+1}} \sum_{i=1}^{N} x(i) \sin\left[\frac{i\,u\,\pi}{N+1} \right]$$

$$u = 1, 2, \ldots N$$

The critical component of this with regard to computational complexity is the fraction within the sine term which gives rise to the values

$$\sin\left[\frac{\pi}{2} \right] = 1$$

and

$$\sin\left[\pi \right] = 0$$

when N is odd. A multiplication is not required to evaluate the components of the transform containing these values, producing the low multiplication count for the EDST with data vectors of odd length. The DCT is defined by Ahmed, Natarajan and Rao [1974] as

$$X(0) = \frac{\sqrt{2}}{N} \sum_{i=0}^{N} x(i)$$

$$X(u) = \frac{2}{N} \sum_{i=0}^{N-1} x(i) \cos\left[\frac{\left[2i+1 \right] u\,\pi}{2N} \right]$$

$$u = 1, \ldots N-1$$

Note that the vectors in this definition are not normalised and that the inverse transform is not identical to the forward transform. The more pertinent property in this discussion is that the fraction in the cosine term cannot take the fractional values of π that yield the values of zero and one. Both transforms do have a high degree of symmetry which makes possible the sparse factorisations such as those derived by Bisherurwa.

Bisherurwa also states that the EDST produces pictures that are subjectively better than those produced by other fast discrete transforms when used for progressive display. This statement relates particularly to the quality of the first displayed image in the progressive sequence. The coefficients before normalisation are illustrated in figure 6.1, the zeroes and ones in the vectors are visible and the shape of the zeroth vector should be noted. The zeroth vector of the DCT has uniform coefficients and the zeroth transform coefficient is related to the mean over the data vector. In this way all the 'DC' energy of the data is 'collected' by the zeroth coefficient. The term 'collected' was used in this sense by Clarke [1983] in a criticism of the DST based on the non-uniform nature of its zeroth basis vector. Some of the other basis vectors of the DST have a non-zero mean and collect some of the 'DC' energy. This adversely affects the behaviour of the transform for data compression as it does not optimally compact information into the low-order transform coefficients. In a later publication Clarke further criticises the DST [1984] showing that it can be derived from the Karhunen-Loeve transform of a first order Markov source in the limiting condition as the correlation tends to zero. This is not a useful model for image processing, a better model is the limiting condition as the correlation tends to unity, from which is derived the DCT. Indeed in a later paper Bisherurwa confirms [Bisherurwa & Coakley 82] that the EDST coefficients have residual correlation, thus casting more doubt on the suitability of the EDST for this application.

6.2. The Sine Transform First Image

There is also the claim for improved subjective performance from the EDST, especially in a variant that introduces an offset into the block structure. This technique used in conjunction with the innovative fast algorithm Bisherurwa and Coakley label 'FEDSTo'. This assertion is supported by pictorial evidence [Bisherurwa & Coakley 1981] where the inverse transformed images from the first DCT, DST and DSTo images are compared. The subjective quality of the first image of the EDSTo transform is due entirely to

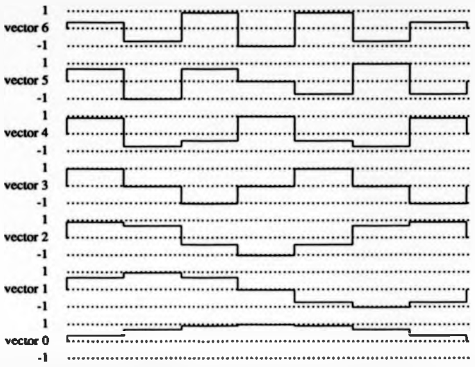


Figure 6.1 Graphical representation of the basis vectors of the 7-D Sine transform

the first basis vector ($u = 1$), which has the coefficient values

0·191 0·354 0·462 0·500 0·462 0·354 0·191

The sum of these coefficients is 2·154 which combines with the largest coefficient 0·5 to give a greatest possible display value of 1·257 times full scale. Considering also the other basis vectors the enumerated minimum and maximum values possible during complete inverse transformation are

min.	−0·385	−0·302	−0·390	−0·354	−0·390	−0·302	−0·385
max.	1·303	1·337	1·343	1·403	1·349	1·337	1·303

Generally the overall range is similar to that of the Hadamard transform given in section 5.2 but the values are no longer constant with position in the vector. The maximum values are symmetrical about the middle coefficient, which has the largest maximum, the structure being comparable to the first basis vector. The set of minimum values is however bi-modal, the reason for this has not been sought though it is somewhat unexpected. Bisherurwa does not state the exact numerical approach used to calculate the display values for the EDST, certainly there is clear potential for the first image to have over-range values. To investigate the subjective quality of the EDSTo it was used to generate the upper-left quadrant of the composite image shown as photograph 6.1. To ensure that all the values are in range and that the display is suitable for full reconstruction, the inverse transformed values have been divided by 1·5 to give the display values. Unfortunately for the comparison available in photograph 6.1, this action has produced a dark image, possible exaggerated due to the photographic process. The validity of some of the following comments may not be obvious from the photograph but are made on the basis of observations of the images directly on the equipment used with control over brightness and contrast to change the image quality. It should be noted that the basic cause of this problem is that the display of EDST intermediate images is not completely straight-forward.

The top left sub-image of photograph 6.1 produced using the EDSTo transform should be compared initially to the lower right, which shows the mean values over the same data blocks described above, again with offset. The mean values are offset in an attempt to factor out this operation in the comparison of the two images. As the offset must break up the block structure it is assumed that the mean with offset is subjectively preferable to the mean without offset. The offset is included in the DST image as the claim made by Bisherurwa and Coakley is for optimum performance with the offset. Although not obvious from the photograph it is agreed that the image produced by the EDSTo is subjectively preferable to that produced using the mean value [Bisherurwa & Coackley 1981]. This appears to be because the sharp boundary between adjacent horizontal blocks that is apparent in the lower right image (mean) does not exist in the upper left image (EDSTo). This is due to the shape of the zeroth basis vector of the DST as shown in figure 6.1, the values of the coefficients corresponding to the ends of a block are smaller than those corresponding to the center. The values of the first display image are calculated as the product of the first transform coefficient and the appropriate basis vector coefficient. This causes individual blocks to have higher pixel values at the centre than at the ends. Since this shaping occurs in adjacent blocks the visual disparity between them is reduced, and as the fall-off of values to the black edge is relatively smooth, the overall result is that the boundary visibility is very much reduced. This property of the basis vector is also partly responsible for the overall difference in 'brightness' between the two compared sub-images. It is not known what the precise effect of this will be in the comparison. The interpolatory aspect is illustrated in the upper-right sub-image of photograph 6.1 in which the mean has been taken over the same block size and used as a multiplicative factor with the zeroth DST basis vector, again with offset. As the DST basis vector has a greatest value of 0·5 the mean value was multiplied by 2 to use the full range of the display and to equalise 'brightness' levels. It can be seen that this produces a result very similar to that of the DSTo (upper left) and indicates that it is the interpolatory effect which is dominant rather than the linear combination of pixel values used to produce the first transform coefficient. The final quadrant of photograph 6.1 to be discussed is the lower left, which is a simple first order interpolation across a horizontal block again of 7 elements with offset. For reconstruction the true mean has been positioned at the right-most end of the pixel block. The appearance of the reconstructed block is very much different to that based upon the envelope of the DST zero vector and it is consequently difficult to subjectively compare the results. It does appear though that the linear interpolation method is at least as good as, if not better than the use of the DST envelope effect. As has been stated previously the first DCT coefficient is related to the mean value of the data points. Therefore if linear interpolation were used after inverse transformation of the first coefficient, the result would be identical to the interpolation quadrant of photograph 6.1. The mean value would have to be recovered from the interpolated data and replicated throughout the block for decoding to continue in the normal fashion. Such a simple approach cannot be used with any subsequent DCT coefficients as all the values within the block must be retained and it is not easy to modify these values in such a way that they can be recovered. The minimum and maximum values in complete DCT inverse transformation were also calculated, they are

min.	−0·283	−0·361	−0·360	−0·388	−0·388	−0·360	−0·361	−0·283
max.	1·283	1·361	1·360	1·388	1·388	1·360	1·361	1·283

Like the limits for the Hadamard transform these values have equal under and over-range components but the mechanism behind the relationship between the values across the block is not at all clear.

6.3. Complexity of the Inverse Transform

Bisherurwa and Coakley have also reported computationally efficient techniques for use in the display phase of progressive reconstruction [1983]. A conventional 'fast' algorithm cannot be used simply to advantage as the computational complexity is independent of the number of coefficients available. The algorithm exists only for the case of all coefficients present. It is possible to use a 'fast' algorithm if coefficients not available (progressively) are set to zero, however the real cost computationally would depend on how individual low-level algorithms and implementations react to the level zero. If coefficients become available one by one then the incremental computation by single vector multiplication and addition will be cheaper than any fast algorithm. Bisherurwa and Coakley have however pointed out the symmetry in the basis vectors and that there are fewer coefficients for multiplication than the degree of the transform, thus saving in multiplications. They also state that the computation can be facilitated by table look-up and that the DST is better than the DCT in terms of the table size required by a factor of two. However assuming a naive implementation of a transform over eight elements there are at most eight coefficients for each of eight vectors and with a data set size of 256 this gives a total table size of 16k entries. With a pessimistic assumption of 16 bits per entry the table size is only 32k bytes, this is not expensive in terms of overall system costs and a DCT (or DST) could be implemented in this way.

Other work in the area of improved algorithms for progressive display has been reported by Takikawa [1984] relating to the use of the Hadamard transform. The technique reported uses data replication to permit the use of fast transforms on an increasing block size, the iterations require progressively 1, 4, 16, 64, 256 data values. This implies that the update would be subjectively different from the coefficient by coefficient approach, possibly more like the exponential progression of the tree update algorithms. As the algorithm is related to the Fast Fourier Transform the author states that it may be possible to extend the algorithm to the DCT and DST.

6.4. Two-Dimensional Transforms

Bisherurwa and Coakley have reported work on two-dimensional transforms of both separable and non-separable types [1982]. Only small area transforms were used with nine elements arranged in a 3×3 square array, with both the DST and DCT being investigated. Non-separable transforms were generated by ordering the pixels in the two-dimensional array to produce a one-dimensional block of nine elements to which the DCT and DST were applied in a conventional manner. Some results were obtained regarding the optimum ordering of the elements to form the one-dimensional block, nearest neighbour ordering being generally optimal. The DCT was preferred to the DST for non-separable transforms generated in this manner. The fact that the non-separable two-dimensional DST does produce the same stripe effect as the one-dimensional DST is stated as an advantage, though the interpolatory effect must be lost. At larger block sizes the non-separable DST does exhibit a checked pattern and the smaller block-size will have an inherently reduced block effect so it is difficult to compare these results with those derived for larger block sizes. This problem also occurs with the investigation of the separable DST and DCT over the small block-size. It is not clear whether use of the DST and DCT at such small block sizes, vectors of length three, has any relationship to larger block-sizes. The properties relating to the shape of the basis vectors must only become apparent at larger block-sizes. Results are stated for the optimum transmission ordering of the two-dimensional array of coefficients. A diagonal scanning of the array that has components horizontal, vertical and diagonal equally distributed is reported as the best approach. Similar work has been reported by Ngan [1984] on the ordering of an 8×8 block of coefficients. A technique that gave good results is based on ordering the coefficients by a mean squared error criterion which would require the overhead of transmission of the sequence information. After this method in performance ranking the results depend on individual picture statistics with an expected correspondence between the orientation of edges within the image and the best ordering of transform coefficients.

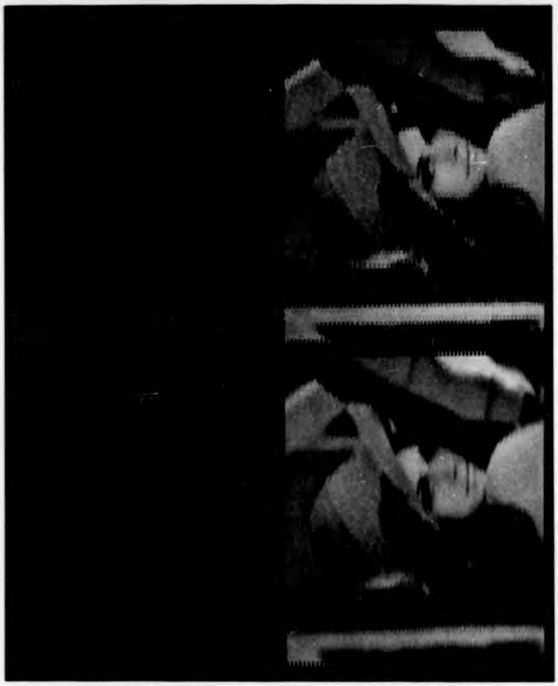
Good results have been reported by Lohscheller [1984] using the separable DCT over blocks of 8×8 pixels. He stresses the importance of early picture content recognition as the primary motivation for progressive transmission and also the constraint that no redundant information should be transmitted. His aims differ from those of myself in that the final image should be subjectively unimpaired rather than identical to the original. Given this criterion it is possible to use the DCT without facing some of the problems stated above, though to obtain optimum quality advanced techniques are used. The basic technique involves independent quantisation of the transform coefficients and transmission ordered by coefficient variance to give the best mean square error performance at each iteration. The effects of rounding within the image store are discussed and it is stated that the required accuracy depends mainly on the transform block size within the context of one transform. It is stated that for a block size of 8×8 elements a storage word size of twelve bits (starting with 8-bit data) results in a rounding error lower than 0·1% of the signal variance. The performance of the technique is improved by investigation of the visbility thresholds of the set of DCT coefficients, these thresholds are used to decide whether or not to transmit a coefficient. This is further developed by use of a classification algorithm that matches the pattern of above threshold coefficients against reference patterns that define a class. Good results are claimed for a compression ratio of 30 with only slight image quality deterioration compared to a compression factor of 4 without the classification algorithm. This combination of techniques sets a very high standard which must be difficult to match with less complex methods. For example the small block size approach used by Bisherurwa and Coakley does not have the potential for such compression ratios. A 3×3 block with original 8-bit quantisation would need to be reconstructed using less than 3 bits in total, this would seem to be a very difficult challenge.

6.5. Exact Invertibility

As indicated by the work of the various researchers discussed above it is possible to use transform coding techniques to good effect in PTD. Intermediate results must be stored to greater precision and range than the original quantisation with some form of mapping between the intermediate representation and the displayable values. The object of exact reconstruction is not addressed by any of the reported transform work and in this sense it does not meet the criteria set out in section 1.2.

A fundamental problem is the precision to which the basis vectors must be described, which of course depends on their desired shape. The basis vectors of the DCT will in all cases be an arbitrary approximation to the desired cosine curve and it does not seem intuitively clear that these approximated vectors would even be exactly orthogonal. However with the DCT, because of the high degree of symmetry, only a small amount of care with symmetry will result in orthogonal vectors and this may accidentally occur within a high-level language program. This would not be true in general without the symmetry and approximation might produce a set of non-orthogonal vectors. The 'precision' of the basis vectors required to produce an exact result is not a meaningful requirement when used in association with a transform such as the DCT. If the inverse transform is a true inverse of the forward transform rather than the transpose of a not quite orthogonal matrix the final results will be exactly identical to the data supplied. It is only in the intermediate calculations that there can be any idea of closeness of approximation to the desired transform. Consideration of this property is important to PTD as it is these intermediate results that are viewed. The quality of the intermediate results will depend on the information compaction properties of the transform used and it is here that the DCT may be expected to be better than say the Hadamard transform. The more coarse the approximation between the real algorithm and the desired transform, the worse will be this compaction property resulting in the image quality not improving as quickly as it might. This situation is less clear for non-progressive transmission with a single resulting image. Assuming orthogonal approximate vectors, any reduction in the compaction property although theoretically detrimental, is likely to be so small as to be completely masked by the operation of the quantiser mechanism. This is also likely to be true of any small irregularity in the orthogonality of the basis vectors. Considering the DCT it would be be possible to use coarse approximations to the basis vectors with manageable properties for PTD. As an extreme example each term in the basis vector could be approximated by plus/minus one depending on the sign of the term. This produces a set of vectors similar but not identical to that of the Hadamard transform, which would for exact reproduction have the same requirements for extended range and precision. Unfortunately, despite being derived from the DCT its performance might be expected to be little different to that of the Hadamard transform. The basis vectors might be made better approximations to those of the DCT. If a binary fractional approach is used, the value 0·5 could be used to approximate some terms if symmetry and orthogonality are retained. The addition of this one bit fraction to the vector approximation increases the precision required to represent the transform coefficients by the same amount, one extra bit. The precision of the basis vectors could be increased in respect to the desired cosine shape to any degree in this manner with a corresponding increase in required precision. The increase in coefficient resolution would need to be matched by an increase in intermediate storage precision. This redundancy results in a large number of possible output values which are not valid data values. Increasing the precision of the basis vectors when a quantiser is used would not have any similar effect.

With the constraint of no redundant data transmission, a conventional approach to transform coding as described above cannot be used for exact reconstruction and a fundamentally different approach must be found.



Photograph 6.1 Each sub-image of this photograph is made up of individually processed blocks of 7 × 1 pixels, horizontally aligned. The top-left image is calculated using the first seven point sine transform coefficient and the first basis vector of the seven point sine transform, with overall division by 1·5. The top-right image uses the same basis vector but multiplied by the mean over the data block with an additional multiplication by two. The bottom-right image has each block filled with the mean value over the block and the bottom-left image uses the same mean values but with first order linear interpolation to fill the blocks.

7. The Two Point Mimic Transform

In this section is reported the initial stage of the design of an algorithm that retains the desired properties of a linear transform for PTD but is also exactly invertible. This is to be done without compromising the suitability for display of the intermediate representation of the algorithm as inherited from the linear transform. In some ways Knowlton's method does this for a specific transform and what is sought may be similar but usable with a wide range of sets of basis vectors over a greater number of points. The idea of approximating the behaviour of a linear transform can be clarified by stating that the criterion of approximation is based on the comparison of the sets of images comprising the PTD sequences produced by the new algorithm and a conventional transform. The idea of 'mimicking' seems appropriate, an algorithm is sought that can mimic the operation of linear transformation, and the outline algorithm will be called the Mimic Transform (MT). To mimic the behaviour of any specific linear transform, such an algorithm would obviously need to use the basis vectors of that transform and thus become a very particular instance of a MT. The additional constraints to be applied to the algorithm are those originally set out in general for PTD techniques to be developed in this thesis. Firstly the algorithm should not require the transmission of any information that is made redundant by any subsequent transmission. Additionally no extra precision should be required to store the intermediate images. Knowlton's method and the new mean/difference transform described in section 3.2 can be taken as indications that a general solution might be possible, rather than negative indications, although do not give significant direction to any design.

7.1. The First Reconstruction Set

As it is the displayed images by which any algorithm will be judged, rather than anything more internal to the algorithm, it seemed reasonable to design the algorithm by consideration of the displayed images. To make the investigation easier it was decided to begin with a two point example, it being considered very difficult to 'visualise' the vectors in any more than three dimensions and easiest in two. However to avoid the immediate trap of a special case, the mean and difference vectors were not used, but an almost equally simple set of vectors was adopted. The two basis vectors B_0, B_1 are defined on the two axis vectors A_0, A_1 as

$B_0 = 0{\cdot}6A_0 + 0{\cdot}8A_1$ 7.1.1

$B_1 = 0{\cdot}8A_0 - 0{\cdot}6A_1$ 7.1.2

By definition the axis vectors are unit orthogonal vectors and by the above definition so are the basis vectors. Note that this gives B_0 a slope of 4/3 which is conveniently related to the 3:4:5 right-angled triangle. The basis vectors are numbered by order of reproduction such that the component due to B_0 would be displayed first in the PTD sequence. The notation adopted for the coefficients of the basis vectors is

$b_{\text{vector number, axis number}}$

With this notation the basis vectors are

$B_0 = b_{0,0}A_0 + b_{0,1}A_1$ 7.1.3

$B_1 = b_{1,0}A_0 + b_{1,1}A_1$ 7.1.4

Using a layout similar to that of the already reproduced tables, the basis vectors are shown in figure 7.1. For the diagrams that are essentially tables, the values on the A_0 and A_1 axes are labelled p_0, p_1 respectively, for the geometrical diagrams the axes are marked A_0, A_1. For figure 7.1, as the points space has been divided up as for a table and labelled with values, the 'axes' have been marked p_0, p_1. The boundary of the point space is reached by a linear distance of 10 along B_0 illustrating one aspect of the quantisation problem in that this value is greater than the range of the data values. The introduction of a single point **p** at co-ordinates 5,3 is also illustrated in figure 7.1. That this is a geometric figure rather than a table is slightly confusing but consistent if the lower left corner of a cell is taken to be the geometric point associated with the cell. The point **p** has transform coefficients of T_0, T_1 on the vectors B_0, B_1 respectively, given by

$T_0 = 0{\cdot}6 \times 5 + 0{\cdot}8 \times 3 = 5{\cdot}4$ 7.1.5

$T_1 = 0{\cdot}8 \times 5 - 0{\cdot}6 \times 3 = 2{\cdot}2$ 7.1.6

By common practice these coefficients would be quantised before transmission. If this were omitted for PTD, T_0 would be transmitted and inverse transformed to produce a first approximation. This inverse transformation with the notation $q_{a,b}$ as the bth coefficient of the ath approximation is

$q_{0,0} = 0{\cdot}6 \times T_0 = 3{\cdot}24$ 7.1.7

$q_{0,1} = 0{\cdot}8 \times T_0 = 4{\cdot}32$ 7.1.8

producing the co-ordinates of point **s** on B_0. Only integer values can be displayed and the point must be

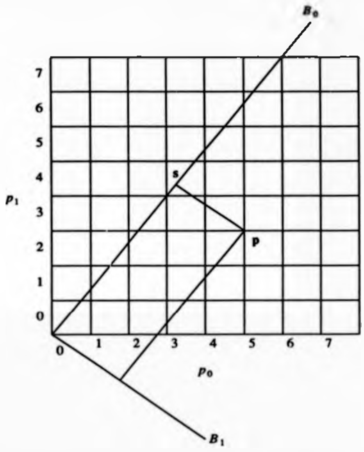


Figure 7.1 An example of a two point transform. Point p represents two data values p_0, p_1 and projects to point s on the basis vector B_0.

approximated, simple truncation or rounding for this point gives the result 3,4. Therefore there is one unavoidable approximation from the inverse transformed co-ordinate to the display value.

The B_0 vector intersects 12 different cells and by inspection at least one point will project onto the B_0 vector in each cell. This means that if the co-ordinates of the projected points are simply truncated to give displayable values there are 12 distinct points that might be produced by a first coefficient. Concentrating on development based on the intermediate images, an algorithm could be developed with 12 transmission codes corresponding to this set of 12 points on B_0. This generates two problems, firstly 12 codes does not require an integer number of bits. Secondly as 12 does not exactly divide 64, the total number of possible codes, the second transform coefficient or specifier cannot be optimally encoded leading to redundant data transmission. A minor consequence of this approach is that the first transform coefficient or specifier would contain more information than the second when without further elaboration it would seem better if they were more nearly equal. Considering these points the best approach is for the intermediate representation to have eight distinct values or codes, the second specifier also then has eight values. This splits the information evenly between the two specifiers and permits integral bit descriptions. The first display will then be one of eight points, each of which approximates a total of eight points. Associated with each data point there is then what will be called a first reconstruction point, which is a member of a first reconstruction set, the same as Knowlton's method. A possible set of eight reconstruction points is shown in figure 7.2, generated by

$$p_0 = \left\lfloor \frac{0{\cdot}6}{0{\cdot}8} p_1 \right\rfloor \qquad p_1 = 0,1,2,3,4,5,6,7 \qquad 7.1.9$$

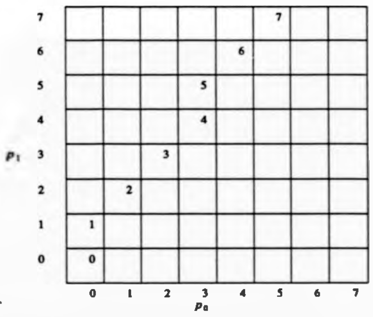


Figure 7.2 A possible first reconstruction set based on the p_1 values.

The set is generated from the A_1 axis and the cell content value is its p_1 value which is a simple way of indexing the point within the set. An undesirable property of the point set is that a decision has to be made concerning to which axis the basis vector is angularly closest for it to be used to generate the point set. The alternative point set

$$p_1 = \left\lfloor \frac{0{\cdot}8}{0{\cdot}6} p_0 \right\rfloor \qquad p_1 = 0,1,2,3,4,5 \qquad 7.1.10$$

is shown in figure 7.3. Without any further action this set contains only six points and the decision of which axis to use is required to achieve the optimum set size.

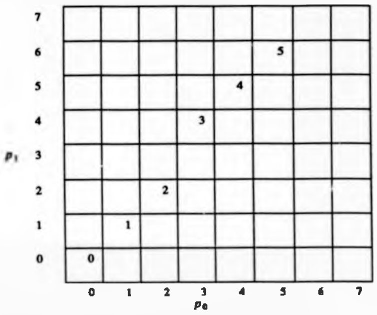


Figure 7.3 A possible first reconstruction set based on the p_0 values.

7.2. The Projection Mechanism

For input points that are in the first reconstruction set shown in figure 7.2, the first approximation can be exactly correct. If the p_1 co-ordinate is used to describe the reconstruction point then this can also be used as the first 'transform' coefficient. Thus the transform coefficient is no longer a measure of the distance along a vector but rather a reconstruction point specifier. The simple correspondence between p_1 value and 'transform coefficient' is only valid for points in the set shown in figure 7.2 though these points are easily found by the property

$$p_0 - \left\lfloor p_1 \frac{b_{0,0}}{b_{0,1}} \right\rfloor = 0 \qquad 7.2.1$$

which gives

$$p_0 - \left\lfloor p_1 \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor = 0 \qquad 7.2.2$$

The axis used to generate or enumerate the first reconstruction set will be referred to as the 'related axis'.

which in this case is A_1. At the next stage in the coding/decoding sequence it is necessary to associate seven more points with each point in the first reconstruction set. This completes a set of eight points with a clear first approximation point, a selection from this set of points can be made by the second specifier to give an individual point. A possible approach to this is to assume that these extra points are linearly arranged parallel to B_1, which is how a linear transform behaves. As has been shown, more points are fitted to the vector if the points are generated from the axis closest to the vector. A set of approximations of vectors parallel to B_1 can be defined by

$$x = p_1 - \left\lfloor p_0 \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor \quad x = 7,6,5.. \qquad 7.2.3$$

These vector approximations are shown in figure 7.4, the cells marked 'a' constitute the set generated by x = 7, marked 'b' correspond to x = 6 etc.. The points in the first reconstruction set are labelled as in figure 7.2. Each alphabetically labelled set should ideally have some correspondence to one point in the first approximation set. However, as may be seen, neither set 'c' nor set 'e' intersects the first approximation set. This is an example of the general problem of intersecting the approximations to two lines, and no further progress was made in attempting to usefully associate a set of points with each reconstruction point in this manner.

A new view-point was adopted, that the first reconstruction set can be likened to the points that project parallel to B_0 to the point zero on the B_1 vector. This same set of points of course projects to zero on the A_0 axis. As sets of points similar to the first reconstruction set tessellate along the A_0 axis, by using modulo arithmetic it is possible to separate the complete space into eight sets of eight points as shown in figure 7.5. The sets can be numbered by their intercept with the A_0 axis and this number can be considered the second approximate transform coefficient or specifier t_1, which can be calculated by what will be called projection as

$$t_1 = \left[p_0 - \left\lfloor p_1 \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor \right] \quad \mathrm{mod}\ 0,7 \qquad 7.2.4$$

where mod x,y indicates the modulus operation with the set of residues, the integers between x and y inclusive.

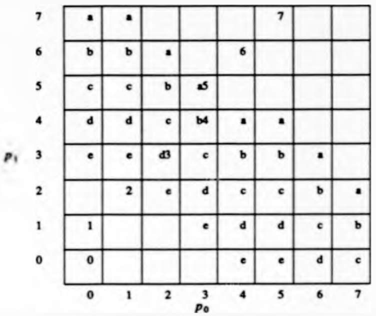


Figure 7.4 The first reconstruction set, labelled numerically, intersects with equivalent approximations that should be orthogonal to it. Each orthogonal set is labelled with a different letter.

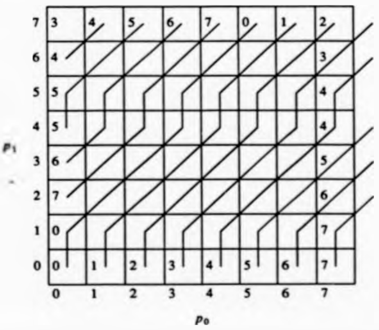


Figure 7.5 Partition of the point space into 8 sets of 8 points based on the projected value.

7.3. The Correction Factor

The projection mechanism segments the point space into the required 8 sets, each of 8 points, and the p_1 co-ordinate of a point uniquely defines the point within a set. This means that a transform from p_0 , p_1 to p_1 , t_1 is invertible and thus the p_1 co-ordinate itself has a property required of the transform coefficient t_0. As stated earlier, for points in the first reconstruction set the first approximation can be exactly correct. This implies that for these points the p_1 co-ordinate is in some way related to the T_0 value. This is a simple geometrical relationship in that the p_1 co-ordinate is the A_1 component of the distance T_0 along B_0. Effectively distance along B_0 is being measured by its projection onto A_1 the related axis. For points not in the first reconstruction set the p_1 co-ordinate is not simply related to the T_0 value measured on A_1, and some method of calculating t_0 with this property is required for these points. It is simple geometrically to calculate the required value, but this gives non-integer results requiring some form of rounding. There is the additional constraint that the 8 possible t_0 values occur in each point set specified by a t_1 value and thus the t_1 value must be included in the t_0 calculation. With all these constraints it seemed very difficult to calculate the t_1 value for a single point in a simple geometrical fashion.

Taking a slightly different approach the t_1 value of a point represents its projection onto A_0, giving the projected point. A point on A_0 other than the origin has a non-zero T_0 value which can be used to calculate the t_0 value of the not-projected original point. Illustrated in figure 7.6 is the construction required to find the t_0 value for this original point p that projects parallel to B_0 to the point r. The point p projects parallel to B_1 to the point s and has the same A_1 co-ordinate as the point q (equals p_1). Distances along B_0 are being measured by their projection onto A_1, the related axis, so the t_0 value of point q is measured as $|ox|$ and similarly the t_0 value of point p is measured as $|oy|$. The t_1 value of point p will approximate the distance $|or|$ and the distance $|ox|$ is known as the A_1 co-ordinate (p_1) of the point p. Before considering the approximation of $|or|$ by t_1 the geometric problem is to find the distance $|oy|$ in terms of the known distances $|ox|$ and $|or|$.

As constructed

st is parallel to uv is parallel to A_1

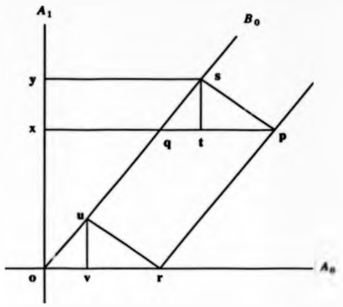


Figure 7.6 Construction to find for a two point transform the correction factor **xy** for the point **p** that projects to the point **r**.

ys is parallel to **xp** is parallel to A_0

sp is parallel to **ur** is parallel to B_1 (not shown)

rp is parallel to B_0

$\|st\| = \|yx\|$	(parallel lines) 7.3.1
$\|sp\| = \|ur\|$	(parallel lines) 7.3.2
$\|qp\| = \|or\|$	(parallel lines) 7.3.4
$\angle spq = \angle uro$	(corresponding angles para. lines) 7.3.5
$\Delta sqp \equiv \Delta uor$	(side angle side) 7.3.6
$\Delta uov \equiv \Delta sqt$	(angle,angle,corres. side) 7.3.7
$\therefore \|uv\| = \|st\|$	7.3.8

The problem can now be restated as finding the distance $|uv|$ in terms of $|or|$. From the construction and the orthogonality of A_0, A_1

$$\angle ovu = 90^\circ \tag{7.3.9}$$

and similarly from the orthogonality of B_0, B_1

$$\angle our = 90^\circ \tag{7.3.10}$$

$$\angle ouv = 90^\circ - \angle uov \tag{7.3.11}$$

$$\angle \mathbf{var} = 90^\circ - \angle \mathbf{ouv} \qquad 7.3.12$$

$$\therefore \angle \mathbf{uov} = \angle \mathbf{var} \qquad 7.3.13$$

If B_0 has direction cosines $b_{0,0}$ and $b_{0,1}$ on axes A_0 and A_1 respectively then

$$\frac{b_{0,1}}{b_{0,0}} = \frac{|\mathbf{uv}|}{|\mathbf{ov}|} = \frac{|\mathbf{vr}|}{|\mathbf{uv}|} \qquad 7.3.14$$

$$|\mathbf{vr}| = |\mathbf{uv}| \frac{b_{0,1}}{b_{0,0}} \qquad 7.3.15$$

$$|\mathbf{ov}| = |\mathbf{uv}| \frac{b_{0,0}}{b_{0,1}} \qquad 7.3.16$$

$$|\mathbf{or}| = |\mathbf{ov}| + |\mathbf{vr}| \qquad 7.3.17$$

$$= |\mathbf{uv}| \frac{b_{0,1}{}^2 + b_{0,0}{}^2}{b_{0,0}\, b_{0,1}} \qquad 7.3.18$$

As B_0 is a unit vector $b_{0,1}{}^2 + b_{0,0}{}^2 = 1$

$$|\mathbf{or}| = \frac{|\mathbf{uv}|}{b_{0,0}\, b_{0,1}} \qquad 7.3.19$$

$$|\mathbf{uv}| = |\mathbf{or}|\, b_{0,0}\, b_{0,1} \qquad 7.3.20$$

Alternatively as vectors

$$B_0 . \overline{\mathbf{or}} = b_{0,0}\, |\mathbf{or}| \qquad 7.3.21$$

$$= |\mathbf{ou}| \qquad 7.3.22$$

$$|\mathbf{uv}| = b_{0,1}\, |\mathbf{ou}| \qquad 7.3.23$$

$$|\mathbf{uv}| = b_{0,1}\, b_{0,0}\, |\mathbf{or}| \qquad 7.3.24$$

This finally gives the desired value $|\mathbf{oy}|$ as

$$|\mathbf{oy}| = |\mathbf{ox}| + b_{0,0}\, b_{0,1}\, |\mathbf{or}| \qquad 7.3.25$$

To translate this into the required form some approximations need to be made. Firstly the distance $|\mathbf{or}|$ is to be approximated by the t_1 value of the point $\mathbf{p}$. The distance $|\mathbf{ox}|$ is known for $\mathbf{p}$ as p_1 but the final result $|\mathbf{oy}|$ must be rounded to an integer value as it will be used to specify a point in the reconstruction set as the value t_0. For conformity with the projection mechanism and as an investigative step, rounding by taking the floor will be used. As p_1 has integer values, only the contribution from t_1 needs to be rounded, giving

$$t_0 = p_1 + \left\lfloor b_{0,0}\, b_{0,1}\, t_1 \right\rfloor \qquad 7.3.26$$

The expression within and including the floor operator will be called the 'correction factor' (cf), being understood as the value to be added to the co-ordinate on the related axis to find for a point the specifier on the basis vector being approximated. From figure 7.6 the correction factor must be positive for the point p, and all its three components are positive, confirming that the correction factor is added to the co-ordinate on the related axis. For a point to the left of B_0, the distance corresponding to $|xy|$ must be subtracted from $|ox|$ to find $|oy|$. This will correspond to the correction factor being negative, requiring t_1 to be negative, a problem because as yet the only values permitted for t_1 are 0 to 7. Before this problem is addressed the results of the derivation of t_0 need to be checked in the context of the example.

7.4. The Modulus Limits

Using the example basis vectors from equations 7.1.1 and 7.1.2 gives for t_0

$$t_0 = p_1 + \left\lfloor 0{\cdot}48 t_1 \right\rfloor \qquad 7.4.1$$

By reference to figure 7.5, for the point 4,7 which has a t_1 value of 7 this gives a t_0 value of 10, whereas only the values 0-7 can be used to specify a point in the first reconstruction set. The point with t_1 value 7 that should have the lowest t_0 value is 7,0 which has a t_0 value of 3. As the t_0 value can only increase from this point, for t_1 equal to 7 the t_0 values 0,1,2 cannot be used. Thus for points with t_1 equal to 7, the t_0 values 0,1,2 are unused and the values 8,9,10 at points 2,5 3,6 and 4,7 are generated but cannot be used. This problem can be rectified by slightly changing the definition of t_0 to be

$$t_0 = \left(p_1 + \left\lfloor b_{0,0} b_{0,1} t_1 \right\rfloor \right) \quad \bmod 0,7 \qquad 7.4.2$$

where mod x,y is the conventional modulus operation with the set of residues the integer values between and including x and y. The values for this latest formulation of t_0 are shown in figure 7.7 with the corresponding full set of values for t_1 shown in figure 7.8. From these two figures together it can be seen that the transform has the required property of 8 sets of points based on the t_1 values. Within each set there is a correspondence with the points in the first reconstruction set which is the set with t_1 equal to 0. However, although the t_0 values are satisfactory for points in and to the right in the table of the set with t_1 equal to 0, they are not very good for points to the left. In normal use of transform coding, data points are expected to be grouped around the basis vectors and in this case might be expected to be distributed equally about B_0. For points to the left of B_0 the described t_0 values are very bad approximations to the appropriate T_0 values. As an example the point 0,2 has a t_0 value of 5, this is due to it projecting to a t_1 value of 7 when it would clearly be more accurately projected to a value of -1. The correction factor is responsible for this behaviour, since as stated above the factor needs to be negative for points to the left of B_0. Improvement can be made by changing the set of residues permitted for t_1, from 0,7 to a set more equally distributed about zero, and the set $-4,3$ is investigated giving

$$t_1 = \left[p_0 - \left\lfloor \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor \right] \quad \text{mod} \; -4,3 \qquad 7.4.3$$

p_1								
7	0	0	1	1	2	7	7	7
6	7	0	0	1	6	6	6	7
5	7	7	0	5	5	5	6	6
4	6	6	7	4	4	4	5	5
3	5	6	3	3	3	4	4	5
2	5	2	2	2	3	3	4	4
1	1	1	1	2	2	3	3	4
0	0	0	0	1	1	2	2	3
p_0	0	1	2	3	4	5	6	7
cf	0	0	0	1	1	2	2	3

Figure 7.7 Table of t_0 values showing the correction factor, based on a projection modulus of 0,7.

The t_0 values with the new set of residues is shown in figure 7.9 where the table has been extended to show the operation with t_1 negative and how the correction factor combines with the modulus operator. It can be seen how some of the points at the lower right of the table which did have correct t_0 values now have t_0 values that do not approximate T_0. These points project due to the modulus operator to the range $-4,0$ and in the table is shown how the t_0 value is calculated, including how if the t_0 value is negative, the modulus operator returns it in range at the higher values. The corresponding t_1 values are shown in figure 7.10, with the values to the left of the first reconstruction set being negative, though the total

	7	3	4	5	6	7	0	1	2
	6	4	5	6	7	0	1	2	3
	5	5	6	7	0	1	2	3	4
	4	5	6	7	0	1	2	3	4
p_1	3	6	7	0	1	2	3	4	5
	2	7	0	1	2	3	4	5	6
	1	0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
		0	1	2	3	4	5	6	7
					p_0				

Figure 7.8 Table of t_1 values based on a projection modulus of 0,7.

number of different values is still eight. The changes in sign of the t_1 values at places other than $t_1 = 0$ indicate the areas of the table that do not contain good approximations. The modulus operation is still required for the t_0 values though its effect is now only apparent in the incorrect approximation parts of the table. It can for example be seen to have effect in the lower-right portion of the table shown in figure 7.9.

The size of the correctly approximated area can be changed slightly by using different sets of residues. A table similar to that of figure 7.9 is shown in figure 7.11 with a different residue set of −3,4 which gives 12 points in error compared with 16 points in error in figure 7.9. The effect of different residue sets will be returned to later, but for the immediate investigation the set −4,3 will be retained as it has computational advantages when used with twos-complement arithmetic. A three-bit twos-complement number can be used to store the values −4,3 and the modulus with this range can be taken simply by masking the appropriate number of bits.

That some of the data space is not correctly approximated is a very considerable flaw in the transform as described. Changing the transform to remove this flaw would seem to require making it more like Knowlton's method, where extreme data is not well approximated but neither completely in error. The

p_1 \ p_0	−4	−3	−2	−1	0	1	2	3	4	5	6	7	
7					0	5	5	6	6	7	7	7	0
6					4	4	5	5	6	6	6	7	
5				3	3	4	4	5	5	5	6	3	
4				2	2	3	3	4	4	4	5	2	
3			1	1	2	2	3	3	3	4	1	1	
2		0	0	1	1	2	2	2	3	0	0	1	
1	−1 7	−1 7	0	0	1	1	1	2	7	7	0	0	
0	−2 6	−2 6	−1 7	−1 7	0	0	0	1	6	6	7	7	
cf	−2	−2	−1	−1	0	0	0	1					

Figure 7.9 Table of t_0 values based on a projection modulus of −4,3 showing how the wrapped-around point values are calculated.

p_1 \ p_0	0	1	2	3	4	5	6	7
7	3	−4	−3	−2	−1	0	1	2
6	−4	−3	−2	−1	0	1	2	3
5	−3	−2	−1	0	1	2	3	−4
4	−3	−2	−1	0	1	2	3	−4
3	−2	−1	0	1	2	3	−4	−3
2	−1	0	1	2	3	−4	−3	−2
1	0	1	2	3	−4	−3	−2	−1
0	0	1	2	3	−4	−3	−2	−1

Figure 7.10 Table of t_1 values based on a projection modulus of −4,3.

	7				0	0	5	6	6	7	7	7	0	0
	6				7	4	5	5	6	6	6	7	7	
	5				3	4	4	5	5	5	6	6		
	4				2	3	3	4	4	4	5	5		
p_1	3			1	2	2	3	3	3	4	4	1		
	2		0	1	1	2	2	2	3	3	0	1		
	1	−1 7	0	0	1	1	1	2	2	7	0	0		
	0	−2 6	−1 7	−1 7	0	0	0	1	1	6	7	7		
		−3	−2	−1	0	1	2	3	4	5	6	7		
								p_0						
	cf	−2	−1	−1	0	0	0	1	1					

Figure 7.11 Table of t_0 values based on a projection modulus of −3,4 showing a different correctly approximated area to modulus limits −4,3.

transform that has been described cannot be simply changed to effect the improvement and it seems substantially more work would be required to do this. A transform different in this respect will not be better at approximating a linear transform than that described. Over much of the data space it is likely to be worse in order to give useful performance at the extremes where the current transform fails. As this simple two element transform is only the first step towards the goals set out at the start of this section it was decided to proceed with the investigation of this transform. There were several reasons for this. It may be that subjective impairment due to this problem will be rare as image data points do not often lay in the badly approximated areas. These errors may be tolerable when they occur, or possibly maskable. Extended investigation of this transform may discover further problems which make the technique completely unworkable, and any further development will certainly aid in the design of an improved transform. It should be possible to evaluate the performance of this transform in such a way that the major flaw is isolated, allowing investigation of other aspects of performance.

7.5. The Inverse Transform

Having completed a design of the transform in a forward direction it may not be clear that what has been described is invertible and if so how. The coefficients t_0 and t_1 are defined

$$t_1 = \left[p_0 - \left\lfloor p_1 \frac{b_{0,0}}{b_{0,1}} \right\rfloor \right] \quad \bmod -4,3 \qquad 7.5.1$$

$$t_0 = \left[p_1 + \left\lfloor b_{0,0} b_{0,1} t_1 \right\rfloor \right] \quad \bmod 0,7 \qquad 7.5.2$$

Decoding is performed by step-wise reversal of the coding sequence

$$p_1 = \left[t_0 - \left\lfloor b_{0,0} b_{0,1} t_1 \right\rfloor \right] \quad \bmod 0,7 \qquad 7.5.3$$

$$p_0 = \left[t_1 + \left\lfloor p_1 \frac{b_{0,0}}{b_{0,1}} \right\rfloor \right] \quad \bmod 0,7 \qquad 7.5.4$$

Although the application of the modulus limits seems arbitrary, the invertibility of the transform depends on the number of different values the coefficients can take rather than the absolute values. Also the inverse should be considered step-wise, not as a whole. The calculation of t_0 and the inversion to find p_1 is performed mod 0,7 with the correction factor only a constant. The steps involving t_1 and p_0 are based on the identity that for a number x in the range 0,n

(x mod p,q) mod 0,n = x for any q − p = n

Having completed the design of the transform, the progressive sequence associated with it can be described. The first value transmitted is t_0 which is used to generate a point in the first reconstruction set. This is in effect the same as performing the inverse transform assuming that t_1 has the value zero. If $q_{n,i}$ is the i th component of the n th approximation then the first inverse transformation gives

$$q_{0,0} = \left\lfloor \frac{b_{0,0}}{b_{0,1}} t_0 \right\rfloor \quad \bmod 0,7 \qquad 7.5.6$$

$$q_{0,1} = t_0 \quad \bmod 0,7 \qquad 7.5.7$$

After the first display the coefficient t_1 is transmitted and the point updated to its original data values. To do this the value t_0 is required and this is available directly from display memory as $q_{0,1}$. The second inverse transformation proceeds as

$$q_{1,1} = \left[t_0 - \left\lfloor b_{0,0} b_{0,1} t_1 \right\rfloor \right] \quad \bmod 0,7 \qquad 7.5.8$$

$$q_{1,0} = \left[t_1 + \left\lfloor q_{1,1} \frac{b_{0,0}}{b_{1,0}} \right\rfloor \right] \quad \text{mod } 0,7 \qquad 7.5.9$$

The modulus operation in the calculation of the first display values is not strictly necessary. It is used because from an implementation viewpoint it is easier to generalise all arithmetic to the same precision as the operation is required for the second phase of the decoding.

Example 7.5

$$p_0 , p_1 = 2 , 5$$

$$\begin{aligned} t_1 &= \left[2 - \left\lfloor 0.75 \times 5 \right\rfloor \right] \quad \text{mod } -4,3 \\ &= -1 \end{aligned}$$

$$\begin{aligned} t_0 &= \left[5 + \left\lfloor 0.48 \times -1 \right\rfloor \right] \quad \text{mod } 0,7 \\ &= 4 \end{aligned}$$

The first reconstruction is

$$\begin{aligned} q_{0,0} &= \left\lfloor 0.75 \times 4 \right\rfloor \\ &= 3 \end{aligned}$$

$$q_{0,1} = 4$$

the second reconstruction is

$$\begin{aligned} q_{1,1} &= \left[4 - \left\lfloor 0.48 \times -1 \right\rfloor \right] \quad \text{mod } 0,7 \\ &= 5 \end{aligned}$$

$$\begin{aligned} q_{1,0} &= \left[-1 + \left\lfloor 0.75 \times 4 \right\rfloor \right] \quad \text{mod } 0,7 \\ &= 2 \end{aligned}$$

reproducing the original data.

End of Example 7.5

7.6. A Useful Mean/Difference Transform

It is not anticipated that the vectors used so far for the development of the transform would be of any practical value, but the same process can be applied to the more useful mean and difference transform.

The basis vectors for this transform are

$$B_0 = \frac{1}{\sqrt{2}}\left[A_0 + A_1 \right] \qquad 7.6.1$$

$$B_1 = \frac{1}{\sqrt{2}}\left[A_0 - A_1 \right] \qquad 7.6.2$$

giving the set of coefficients

$$\begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{bmatrix} = \begin{vmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{vmatrix} \qquad 7.6.3$$

Assuming the B_0 component of the two data values is transmitted first, the first approximation will be close to the mean of the data. As B_0 is equidistant from the two axes A_0, A_1, either axis can be used as the related axis to generate the first reconstruction set. If the axis A_1 is used the transform is

$$t_1 = \left\lfloor p_0 - p_1 \frac{\frac{1}{\sqrt{2}}}{\frac{1}{\sqrt{2}}} \right\rfloor \quad \text{mod } -4,3 \qquad 7.6.4$$

$$= \left\lfloor p_0 - p_1 \right\rfloor \quad \text{mod } -4,3 \qquad 7.6.5$$

$$t_0 = \left[p_1 + \left\lfloor \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} t_1 \right\rfloor \right] \quad \text{mod } 0,7 \qquad 7.6.6$$

$$= \left[p_1 + \left\lfloor 0{\cdot}5\, t_1 \right\rfloor \right] \quad \text{mod } 0,7 \qquad 7.6.7$$

The t_0 values for this transform are shown in figure 7.12 and the t_1 values in figure 7.13. The diagonal structure produced by the equal axis components of the B_0 vector are clearly visible in both figures and the areas of breakdown of approximation are visible as in earlier tables. By the inverse transformation of t_0 the first approximation is calculated as

$$q_{0,1} = t_0 \quad \text{mod } 0,7 \qquad 7.6.8$$

$$q_{0,0} = \left\lfloor \frac{b_{0,0}}{b_{0,1}} t_0 \right\rfloor \quad \text{mod } 0,7 \qquad 7.6.9$$

$$= \left\lfloor \frac{\frac{1}{\sqrt{2}}}{\frac{1}{\sqrt{2}}} t_0 \right\rfloor \quad \text{mod } 0,7 \qquad 7.6.10$$

$$= t_0 \quad \text{mod } 0,7 \qquad 7.6.11$$

p_1 \ p_0	−4	−3	−2	−1	0	1	2	3	4	5	6	7			
7					7	0	0	5	5	6	6	7	7	0	0
6					7	7	4	4	5	5	6	6	7	7	
5					6	3	3	4	4	5	5	6	6		
4					2	2	3	3	4	4	5	5			
3				1	1	2	2	3	3	4	4	1			
2			0	0	1	1	2	2	3	3	0	0			
1		$\frac{-1}{7}$	$\frac{-1}{7}$	0	0	1	1	2	2	7	7	0			
0	$\frac{-2}{6}$	$\frac{-2}{6}$	$\frac{-1}{7}$	$\frac{-1}{7}$	0	0	1	1	6	6	7	7			

Figure 7.12 Table of approximate mean values t_0 for the mean/difference transform.

p_1 \ p_0	0	1	2	3	4	5	6	7
7	1	2	3	−4	−3	−2	−1	0
6	2	3	−4	−3	−2	−1	0	1
5	3	−4	−3	−2	−1	0	1	2
4	−4	−3	−2	−1	0	1	2	3
3	−3	−2	−1	0	1	2	3	−4
2	−2	−1	0	1	2	3	−4	−3
1	−1	0	1	2	3	−4	−3	−2
0	0	1	2	3	−4	−3	−2	−1

Figure 7.13 Table of difference values t_1 for the mean/difference transform.

After this display, t_1 will be received and the second display calculated, again t_0 is available from display memory as $q_{0,1}$.

$$t_0 = q_{0,1} \qquad 7.6.12$$

$$q_{1,1} = \left[\, t_0 - \left\lfloor 0{\cdot}5\, t_1 \right\rfloor \right] \quad \text{mod } 0,7 \qquad 7.6.13$$

$$q_{1,0} = \left[\, t_1 + q_{1,1} \right] \quad \bmod 0,7 \tag{7.6.14}$$

Example 7.6

$$p_0 \, , p_1 = 3 \, , 6$$

Coding

$$t_1 = \left[\, p_0 - p_1 \right] \quad \bmod -4,3$$
$$= 3 - 6 \quad \bmod -4,3$$
$$= -3$$
$$t_0 = \left[\, p_1 + \left\lfloor 0{\cdot}5 t_1 \right\rfloor \right] \quad \bmod 0,7$$
$$= \left[\, 6 + \left\lfloor 0{\cdot}5 \times -3 \right\rfloor \right] \quad \bmod 0,7$$
$$= \left[\, 6 - 2 \right] \quad \bmod 0,7$$
$$= 4$$
$$t_1 \, , t_0 = -3 \, , 4$$

First display

$$q_{0,0} = q_{0,1} = t_0 = 4$$

Second display

$$t_0 = q_{0,1}$$
$$q_{1,1} = \left[\, 4 - \left\lfloor 0{\cdot}5 \times -3 \right\rfloor \right] \quad \bmod 0,7$$
$$= 6$$

$$q_{1,0} = \left[\, 6 + -3 \right] \quad \bmod 0,7$$
$$= 3$$

End of Example 7.6

This transform is almost identical to that described in section 3.2, which was included partly for the comparison now being made. The t_0 values are identical except for the interchange of the $p_0 \, , p_1$ values. The patterns of the t_1 values in the two transformations are exactly the same, only the numbering scheme is different. The patterns in both the t_0 and t_1 values are easy to see in figures 7.12 and 7.13, being clear lines at 45 degrees to the table boundaries. The method described immediately above for generating these tables is more simple than that described in section 3.2, illustrating how different algorithms can yield the same result and the similarity between the two might be further investigated. The t_0 values of the transform are in a sense optimal, for the correctly approximated area they are equal to the truncated mean value of the data points. The inverse transformation giving the first displayed approximation introduces no further error so this approximation is as accurate as possible. The correctly approximated area is also as large as possible, to within the choice of modulus limits −4,3 or −3,4 which in this case makes no difference to the area. All the 2,3,4,5 values of t_0 are in the area of the table with good approximations, the approximation failing for the points that would ideally have these values but for which one is not 'available'. Though the transform has good accuracy for some points, again part of the space, 25%, has an incorrect approximation. The trade-off for this undesirable behaviour is the correctness of the rest of the space compared to other methods such as Knowlton's technique. Knowlton's method has no badly approximated points, but the better approximated are not as well approximated as this transform makes possible.

This transform can be used directly in place of Knowlton's transform and the replacement was made in the programs that produced photographs 2.2 and 4.1. Little difference was apparent and it was not considered worthwhile reproducing the results because of the similarity. This does show that the impairments due to the incorrect areas in the tables do not necessarily lead to subjective impairments, this is true for this image but it is not possible to generalise. The new transform is more simple to implement than even the algorithmic equivalent to Knowlton's tables, in use it may even be faster than table look-up. To find the t_1 value requires only a subtraction and mask; t_0 requires a one bit shift, add and mask. It must surely be difficult to achieve any more useful transform in fewer operations. For its simplicity it might be considered for implementation directly in hardware if the potential problems can be tolerated.

7.7. The Correctly Approximated Area

Returning to the more general development of the MT, an early decision in the design of the transform was to use the point set generated by

$$p_0 = \left\lfloor \frac{0{\cdot}6}{0{\cdot}8} p_1 \right\rfloor \quad p_1 = 0,1,3,..7 \qquad 7.7.1$$

for possible first display approximations, the first reconstruction set. The rejected alternative based on the use of p_0 for generation was dismissed because only 6 points were generated as opposed to the 8 points using p_1. This was before the introduction of the modulus operator, with which the 8 points that would be described by the t_0 coefficient generated by p_0 are shown in figure 7.14. This point set does not seem useful but in addition to the first approximation set the total number of points approximated is also important. The complete transform space with the t_1 coefficient is shown in figure 7.15. The number of points correctly approximated is not as great as that using p_1 for generation, though the transform does not break down with the less than optimum choice. Interestingly, the correction factor as originally derived with A_1 as the related axis

$$\mathrm{cf} = \left\lfloor b_{0,0} b_{0,1} t_1 \right\rfloor \qquad 7.7.2$$

has only B_0 coefficients, this becomes

$$\mathrm{cf} = \left\lfloor b_{1,0} b_{1,1} t_1 \right\rfloor \qquad 7.7.3$$

using A_0 as the related axis. These two expressions have with the values of the main example used in this section the same magnitude but different signs. Using the A_1 axis any point will have a different t_1 value so the correction factor for any point will be different, but this shows the high degree of symmetry involved. Investigating the use of the A_0 axis also makes more clear how the axis choice affects the approximated area. As the range of projected values is fixed as the same as the t_1 values, there is a well defined length of axis onto which points can project to be correctly approximated. The area that is correctly approximated is that swept out by translating this portion of the axis parallel to the basis vector being approximated. The more nearly normal to this basis vector the axis chosen, the greater is the swept out area and consequently the area of correct approximation. For a set of orthogonal axes this is equivalent to choosing the axis not to be projected onto for the related axis by angular closeness.

This concludes the investigation into exactly reversible transforms of over two points. Although the two point transform is an essential first step, the overall objective is for transforms over more points. The principles of the two point transform must be assessed to see how they might be extended. These principles are

i) The choice of the set of points to be used for the first approximation and the method of enumerating them by the co-ordinate on the related axis.

ii) Projecting back to the remaining axis by using the co-ordinate on the related axis, to give the second transform coefficient t_1.

iii) Using the projected value t_1 to calculate a correction factor which is used together with the co-ordinate on the related axis to give the first transform coefficient t_0.

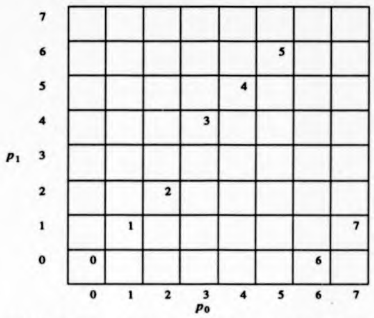


Figure 7.14 A first reconstruction set based on p_0 values with the modulus operation used to bring 8 points into the set.

									3
								3	2
								2	1
							3	1	0
						3	2	0	−1
	7	−1	−2	−3	3	2	1	−1	−2
	6	−2	−3	−4	2	1	0	−2	−3
	5	−3	−4	3	1	0	−1	−3	−4
	4	−4	3	2	0	−1	−2	−4	3
p_1	3	3	2	1	−1	−2	−3	3	2
	2	2	1	0	−2	−3	−4	2	1
	1	1	0	−1	−3	−4	3	1	0
	0	0	−1	−2	−4	3	2	0	−1
	−1	−1	−2	−3					
	−2	−2	−3	−4					
	−3	−3	−4						
	−4	−4							
		0	1	2	3	4	5	6	7

p_0

Figure 7.15 Table of t_1 values using A_0 as the related axis with modulus limits −4,3.

8. The Three Point Mimic Transform

8.1. The First Reconstruction Set and Projection

The transformation developed in the previous section must now be extended to operate over more points. A three point example is used, much as the two point example was used, to retain a clear understanding of the developing mechanism. The following basis vectors were chosen in an attempt to keep at least some of the vectors easy to visualise and again attempting to avoid a special case. The orthogonal vectors were generated first and the multiplicative factor is simply the inverse of the modulus. Although the decimal fractions are quoted to a few digits, the precision is not rigidly adhered to, any later calculations use precision as stated. The vectors chosen are

$$B_0 = 0{\cdot}1857\left[\,2A_0 + 3A_1 + 4A_2\right] \qquad 8.1.1$$
$$B_1 = 0{\cdot}0909\left[\,9A_0 + 2A_1 - 6A_2\right] \qquad 8.1.2$$
$$B_2 = 0{\cdot}0169\left[\,-26A_0 + 48A_1 + 23A_2\right] \qquad 8.1.3$$

where

$$B_0 = b_{0,0}A_0 + b_{0,1}A_1 + b_{0,2}A_2 \qquad 8.1.4$$
$$B_1 = b_{1,0}A_0 + b_{1,1}A_1 + b_{1,2}A_2 \qquad 8.1.5$$
$$B_2 = b_{2,0}A_0 + b_{2,1}A_1 + b_{2,2}A_2 \qquad 8.1.6$$

The set of vectors is ordered by the required approximation sequence, the first decoded approximation is based on the single vector B_0. The second and third approximations use additional information related to the B_1 and B_2 vectors respectively. Adopting the same approach as the two point transform case, an axis must be choosen as the related axis for B_0 by the criterion of smallest angle between the axis and the vector. This then generates the first approximation set in a very simple fashion. At this stage the related axis is found as the axis with the greatest component in B_0, as the axes are unit orthogonal. With the example under investigation this is A_2. The first reconstruction set is generated in as similar fashion as possible to the two point case to be

$$\left[p_2\frac{2}{4}, p_2\frac{3}{4}, p_2\right] \quad p_2 = 0,1,2...7 \qquad 8.1.7$$

The next step in the development is to consider how to project the point back to the plane defined by

A_0, A_1, by reducing the A_2 component to zero. A slight change of notation is required as more than one iteration will be required to produce three rather than two transform coefficients. The notation $p_{n,x}$ is used for the co-ordinate on the xth axis at the n th iteration. The first projection using the same set of residues as the two point case is given by

$$p_{1,0} = \left(p_{0,0} - \left\lfloor p_{0,2} \frac{b_{0,0}}{b_{0,2}} \right\rfloor \right) \quad \text{mod } -4,3 \qquad 8.1.8$$

$$p_{1,1} = \left(p_{0,1} - \left\lfloor p_{0,2} \frac{b_{0,1}}{b_{0,2}} \right\rfloor \right) \quad \text{mod } -4,3 \qquad 8.1.9$$

$$p_{1,2} = \left(p_{0,2} - \left\lfloor p_{0,2} \frac{b_{0,2}}{b_{0,2}} \right\rfloor \right) = 0 \qquad 8.1.10$$

The result of this projection is a co-ordinate pair in the A_0, A_1 plane. Some properties of this transform are shown in figure 8.1. The direction of the B_0 vector through the origin is shown, and also indicated is the way in which some of the component solids produced by the modulus limits fit together to affect the correctly approximated area.

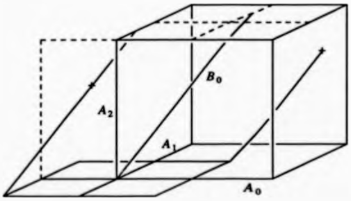


Figure 8.1 Outline diagram of projection in 3-D. The cube represents the point space and the angled lines are parallel to B_0, showing projection to the displaced square at the base of the cube.

The next step is to calculate t_0 using the co-ordinate on the related axis A_2 and the correction factor which will now be based on two values, $p_{1,0}$ and $p_{1,1}$.

$$t_0 = p_{0,2} + \text{cf}\left(p_{1,0}, p_{1,1} \right) \quad \text{mod } 0,7 \qquad 8.1.11$$

For this, the correction factor must be derived, and this can be performed in a manner analogous to the two point case.

8.2. The First Correction Factor

The construction required for the derivation of the first correction factor is shown in figure 8.2 which has some similarity with figure 7.6 that illustrates the two point case. The correction factor is to be derived for the point p which has the same A_2 co-ordinate as the point q. The main difference from figure 7.6 is that the vectors A_2, B_0 and the point p need not be all co-planar, though any pair will be.

In the plane of A_2, B_0 the lines yes xq du are constructed normal to A_2 and the line cq parallel to A_2.

$\lvert cq \rvert = \lvert xy \rvert$	(parallel lines) 8.2.1

In the plane of B_0, p the line rp is constructed parallel to B_0, and or parallel to qp as q and p have the same A_2 co-ordinate.

$\lvert sp \rvert = \lvert ur \rvert$	(parallel lines) 8.2.2
$\lvert qp \rvert = \lvert or \rvert$	(parallel lines) 8.2.3
$\angle spq = \angle uro$	(corresponding angles) 8.2.4
$\Delta spq = \Delta uro$	(side angle side) 8.2.5
Δscq similar Δudo	(corresponding angles parallel lines) 8.2.6
$\lvert sq \rvert = \lvert uo \rvert$	(from 8.2.5 congruent triangles) 8.2.7
$\Delta scq = \Delta udo$	(angle angle corres. side) 8.2.8
$\lvert od \rvert = \lvert cq \rvert$	(congruent triangles) 8.2.9

The points p and q have the same A_2 co-ordinate of $\lvert ox \rvert$, to find the t_0 value of p the distance $\lvert xy \rvert$ must be added to this. The distance $\lvert xy \rvert$ is equal to the distance $\lvert od \rvert$ which depends on the point r which has co-ordinates on the A_0, A_1 axes.

$$\lvert ou \rvert = \overline{or} . B_0 \qquad 8.2.10$$

$$\lvert od \rvert = b_{0,2} \lvert ou \rvert \qquad 8.2.11$$

$$\lvert od \rvert = b_{0,2} \left(\overline{or} . B_0 \right) \qquad 8.2.12$$

This scalar product can be expanded in terms of the two non-zero co-ordinates and with the appropriate rounding gives

$$cf\left(p_{1,0}, p_{1,1} \right) = \left\lfloor {}^{e}b_{0,2} \left({}^{e}b_{0,0} p_{1,0} + {}^{e}b_{0,1} p_{1,1} \right) \right\rfloor \qquad 8.2.13$$

This gives t_0 as

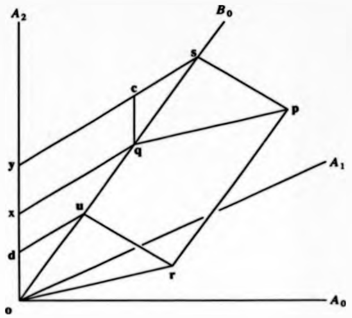


Figure 8.2 Construction to find for a three point transform the correction factor xy for the point p that projects to the point r.

$$t_0 = p_{0,2} + \left\lfloor {}^0b_{0,2}\left[{}^0b_{0,0}\,p_{1,0} + {}^0b_{0,1}\,p_{1,1} \right] \right\rfloor \qquad 8.2.14$$

Again the correction factor is added to the co-ordinate on the related axis as shown in figure 8.2. Depending on the position of the points p and r the distance |ou| may be negative along B_0 which is a valid result of the scalar product. In this case the correction factor will reduce the value on the related axis (the related value), to give the transform coefficient.

8.3. Projection to New Axes

The projection to co-ordinate $p_{1,0}, p_{1,1}$ and the calculation of t_0 can be viewed as one iteration of the coding algorithm. This has produced one transform coefficient and ideally would have removed from further consideration one axis A_2 and one basis vector B_0, reducing the dimensionality of the problem from three to two. If this were true then the two point transformation already developed might be applied and the overall transformation completed. Unfortunately this is not quite straight-forward. The problem is that the projected point $p_{1,0}, p_{1,1}$ lies in the plane defined by A_0, A_1 rather than the plane defined by B_1, B_2 which would be the case for a true linear transform. Any attempt to project to the plane defined by B_1, B_2 meets with the same sort of intersection problems as the two-dimensional problem shown in figure 7.4. It is easy to define a set of points to constitute an approximation to a plane, but difficult to project a point in three dimensions to intersect a point in this set. This problem means that without further effort A_2 and B_0 have not been eliminated from further consideration. The solution is illustrated in figure 8.3, though the figure is not related to the three point numerical example. In figure 8.3 the point P projects to the point Q in the plane A_0, A_1 and to the point R in the plane B_1, B_2. The points m and n mark the unit vectors on the axes A_0, A_1 and it can be seen how these vectors can be considered as some vector in the plane B_1, B_2 plus some component in the direction B_0.

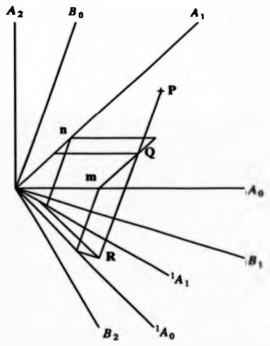


Figure 8.3 The relationship between the co-ordinates of the point Q in the plane A_0, A_1 and the point R in the plane ${}^1A_0, {}^1A_1$ when both project from the point P.

If

$$A_0 = {}^1A_0 + a_0 B_0 \qquad 8.3.1$$

$$A_1 = {}^1A_1 + a_1 B_0 \qquad 8.3.2$$

and

$$\mathbf{Q} = p_{1,0} A_0 + p_{1,1} A_1 \qquad 8.3.3$$

then

$$\mathbf{Q} = p_{1,0}\,{}^1A_0 + p_{1,1}\,{}^1A_1 + \left[p_{1,0} a_0 + p_{1,1} a_1 \right] B_0 \qquad 8.3.4$$

If point **R** is in the plane defined by B_1, B_2 then it has no B_0 component and thus

$$\mathbf{R} = p_{1,0}\,{}^1A_0 + p_{1,1}\,{}^1A_1 \qquad 8.3.5$$

Therefore the co-ordinates $p_{1,0}, p_{1,1}$ can be retained as co-ordinates in the correct plane without modification. The condition required to do this is that they are now co-ordinates on what can be considered a new set of axes ${}^1A_0, {}^1A_1$. Since for subsequent iterations it is anticipated that more new sets of axes might be required, the axis notation can be modified so that the original axes A_0, A_1, A_2 can be relabelled ${}^0A_0, {}^0A_1, {}^0A_2$. This notation is also extended to the basis vector coefficients such that the basis vectors B_i are defined in terms of axis vectors ${}^0A_i$ by coefficients ${}^0b_{i,j}$. The vectors ${}^1A_0$ and ${}^1A_1$ are easily found from ${}^0A_0$ and ${}^0A_1$ which themselves can be found by inverting the first basis matrix. This first basis matrix contains the original basis vector coefficients :-

$$\begin{bmatrix} {}^0b_{0,0} & {}^0b_{0,1} & {}^0b_{0,2} \\ {}^0b_{1,0} & {}^0b_{1,1} & {}^0b_{1,2} \\ {}^0b_{2,0} & {}^0b_{2,1} & {}^0b_{2,2} \end{bmatrix} \begin{bmatrix} {}^0A_0 \\ {}^0A_1 \\ {}^0A_2 \end{bmatrix} = \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix} \qquad 8.3.6$$

as this matrix is unitary orthogonal it can be inverted by transposition to give the axis matrix

$$\begin{bmatrix} {}^0b_{0,0} & {}^0b_{1,0} & {}^0b_{2,0} \\ {}^0b_{0,1} & {}^0b_{1,1} & {}^0b_{2,1} \\ {}^0b_{0,2} & {}^0b_{1,2} & {}^0b_{2,2} \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} {}^0A_0 \\ {}^0A_1 \\ {}^0A_2 \end{bmatrix} \qquad 8.3.7$$

Hence the vectors ${}^1A_0$ and ${}^1A_1$, which are respectively the vectors A_0 and A_1 with their B_0 component removed, are generated by deleting the left column of the axis matrix. As A_2 is no longer required then the bottom row also is deleted.

$${}^1A_0 = {}^0b_{1,0} B_1 + {}^0b_{2,0} B_2 \qquad 8.3.8$$

$${}^1A_1 = {}^0b_{1,1}B_1 + {}^0b_{2,1}B_2 \qquad 8.3.9$$

Putting in some numerical values from the three point example, this gives

$${}^1A_0 = 0{\cdot}8182\,B_1 - 0{\cdot}4389\,B_2 \quad \text{with modulus } 0{\cdot}9286 \qquad 8.3.10$$

$${}^1A_1 = 0{\cdot}1818\,B_1 + 0{\cdot}8103\,B_2 \quad \text{with modulus } 0{\cdot}8313 \qquad 8.3.11$$

Although these vectors ${}^1A_1, {}^1A_0$ will be referred to as axis vectors since the projected points have co-ordinates on them, they have, or more properly lack, two properties that are normally associated with axes. These vectors are neither orthogonal nor of unit length. Although the problem is now reduced to having two values, it is not the same problem dealt with in the two point case. The main difference is that the axes are not orthogonal and at this stage the axis vectors are defined in terms of the basis vectors which is not adequate to perform the projection part of the algorithm. To perform the projection operation the basis vectors must be known in terms of the axes and this requires the inversion of the axis matrix to find the basis matrix. The matrix to be inverted, drawn from equations 8.3.8 and 8.3.9 is

$$\begin{bmatrix} {}^0b_{1,0} & {}^0b_{2,0} \\ {}^0b_{1,1} & {}^0b_{2,1} \end{bmatrix} = \begin{bmatrix} 0{\cdot}8182 & 0{\cdot}4389 \\ 0{\cdot}1818 & 0{\cdot}8103 \end{bmatrix} \qquad 8.3.12$$

In this example the matrix is not singular and as a two by two matrix it is simply inverted. The more general question of singularity at this point in the algorithm will be discussed later. The inverse matrix which defines the basis vectors B_0, B_1 in terms of ${}^1A_0, {}^1A_1$ is denoted

$$\begin{bmatrix} {}^1b_{1,0} & {}^1b_{1,1} \\ {}^1b_{2,0} & {}^1b_{2,1} \end{bmatrix} = \begin{bmatrix} 1{\cdot}091 & 0{\cdot}5909 \\ -0{\cdot}2448 & 0{\cdot}1803 \end{bmatrix} \qquad 8.3.13$$

giving

$$B_1 = {}^1b_{1,0}\,{}^1A_0 + {}^1b_{1,1}\,{}^1A_1 \qquad 8.3.14$$

$$B_2 = {}^1b_{2,0}\,{}^1A_0 + {}^1b_{2,1}\,{}^1A_1 \qquad 8.3.15$$

Note that the subscripts for the coefficients describing the basis vectors are the transpose of those of the coefficients describing the axis vectors, and the pre-super-script has changed. The subscripts are kept this way to make manipulations across iterations and between basis and axis vectors easier, for verification of the algorithm in a more algebraic fashion as will be shown later. Clearly the pre-super-script needs to change to uniquely identify the values. The change is perhaps less clear than it might be because the deletion of a row and column from the axis matrix created a new axis matrix without changing any of the remaining coefficients or their pre-super-script. When the example is enumerated, the resulting values are shown in statement 8.3.13, it might be that B_0 and B_1 seem no longer unitary since

$${}^1b_{1,0}^2 + {}^1b_{1,1}^2 \neq 1 \qquad 8.3.16$$

However, it must be emphasised that ${}^1A_0$ and ${}^1A_1$ are not unitary and that the inequality does not have its usual significance. This represents an unusual change of emphasis in that the basis vectors are not being manipulated as might be expected, but to avoid any irreversible operations on the numerical co-ordinates it is the axis vectors that are being operated upon.

8.4. The Choice of Related Axis

Having performed these operations on the basis and axis vectors, the operation of projection, which is fundamental to the technique, must be re-examined. This in turn depends on the choice of related axis, which needs again to be investigated. The concept of the related axis was introduced initially for the generation of a set of points from which a first approximation might be chosen. However the choice of the related axis also implicitly decides the set of axes to be projected onto, as the remaining axes. It is the orientation of these axes that affects the correctly approximated area. This is to some extent demonstrated by figure 7.15 and 7.9 where the 'width' of the correctly approximated area, orthogonal to the vector being approximated, is affected by the axis being projected onto. The size of the correctly approximated area is increased when the axis projected onto is more nearly orthogonal to the basis vector being approximated. The limit of this would be the 'projected onto' axis being co-linear with a basis vector not being approximated. If the choice of related axis were sought via maximisation of the correct area, the choice would be the axis least orthogonal to the approximated basis vector. This is, of course, the axis which is angularly closest to the approximated vector, the same axis that would be chosen by consideration of the first approximation set. Calculation of the angle between a basis vector and the axes is not as straight-forward as in section 8.1 where the cosine of the angle was directly available from the basis coefficient matrix. With orthogonal and unitary axes and unit basis vectors, the basis matrix contains as coefficients the cosine of the angle between basis vector and axis. To minimise the angle requires the maximum cosine value, so the related axis is chosen to have the greatest coefficient in the equation defining the basis vector in terms of the axes. With the change in properties of the axis vectors the angle must be calculated more explicitly using the orthogonality and unit length properties of the basis vectors.

The next basis vector to be approximated is B_1 and the angle between this and the two remaining axes ${}^1A_0, {}^1A_1$ is required to choose the next related axis. By using equations 8.3.8 and 8.3.9 and taking the scalar product with each axis, ${}^1A_0$ and ${}^1A_1$, in turn, and dividing the result by the length of the axis vector yields the cosine of the required angle giving

$$ {}^1A_0 \quad \frac{{}^0b_{1,0}}{\sqrt{{}^0b_{1,0}^{\,2} + {}^0b_{2,0}^{\,2}}} \qquad 8.4.1 $$

$$ {}^1A_1 \quad \frac{{}^0b_{1,1}}{\sqrt{{}^0b_{1,1}^{\,2} + {}^0b_{2,1}^{\,2}}} \qquad 8.4.2 $$

Of these two values, the greater must be chosen to give the closest axis to be the related axis. It might be expected that the modulus component of these expressions would obviate any anomaly due to the modulus of the axes not being one. For the three point example being used in this section, the values for ${}^1A_0$ and ${}^1A_1$ are respectively 0·8812 and 0·2189 by which ${}^1A_0$ is chosen as the related axis. In general some care with the sign of the coefficients will be required, the choice strictly needs to be made on the modulus of the values calculated in expressions 8.4.1 and 8.4.2.

Having seen that the choice of related axis causes no extraordinary problems with the new axes, there is nothing in the projection mechanism that requires attention. The new axes cause the co-ordinate spacing on which the points are located to be distorted and no longer rectilinear, in terms of the basis vectors, but as the basis vectors are expressed in terms of these axes for projection there is no inconsistency.

Unfortunately, this geometrical interpretation, the choosing of the axis vector closest to the basis vector is not correct. The problem becomes apparent when using this criterion, as some apparently anomalous behaviour is discovered. This is illustrated with a rather contrived example that commences with an axis matrix, that is the axis vectors defined in terms of the basis vectors. This corresponds to the coefficients in equations 8.3.8 and 8.3.9, but as it is not intended to represent any one iteration the extended axis notation is not used, and for clarity the axes are labelled B_0, B_1. The axis matrix is

$$ \begin{bmatrix} \frac{1}{2} & \frac{1}{8} \\ \frac{-5}{8} & 1 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} \qquad 8.4.3 $$

which can be inverted to give the basis matrix

$$\begin{bmatrix} b_{0,0} & b_{1,0} \\ b_{1,0} & b_{1,1} \end{bmatrix} \text{ defined in } \frac{64}{37}\begin{bmatrix} 1 & \frac{-1}{8} \\ \frac{5}{8} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} A_0 \\ A_1 \end{bmatrix} = \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} \qquad 8.4.4$$

If the axis to be approximated is B_1, expressions equivalent to 8.4.1 and 8.4.2 yield the approximate value for A_0 of 0·2425 (approx. 76°), and for A_1 of 0·8478 (approx. 32°). Clearly axis A_1 is closer to B_0 than A_0 and should be used for the related axis. A point $p_{0,0}, p_{0,1}$ is projected using coefficients from the basis matrix given in equation 8.4.4. This gives

$$p_{1,0} = p_{0,0} - \left\lfloor p_{0,1} \frac{b_{1,0}}{b_{1,1}} \right\rfloor \qquad 8.4.5$$

with numerical values

$$p_{1,0} - p_{0,0} - \left\lfloor p_{0,1} \frac{5}{8}\frac{2}{1} \right\rfloor \qquad 8.4.6$$

The aspect of this expression that is anomalous, is the factor within the floor operator which is greater than unity, though at this point in the development of the algorithm the expected value of this factor has not been discussed. However, in the first two point example, with the 'best' related axis the factor was 0 6/0 8, this is less than unity, whereas with the other axis being used as related axis, this factor was inverted to be greater than unity. This throws suspicion on the new factor with value greater than one, which is justified when the point space is shown with the basis and axis vectors in figure 8.4. This diagram differs from previous diagrams in that the origin is in the centre of the space, consistent with point data after a projection with modulus limits –4,3. From figure 8.4 it can be seen that vector B_1 intersects a boundary of the point space defined by the A_0 axis. This means that using the A_0 axis, eight points can be defined on B_0 whereas fewer points can be defined using A_1. Therefore, by the criterion of point set, A_0 should be used as the related axis. Investigating the problem of correctly approximated space, the limits are shown approximately in figure 8.5. The dotted lines show the limits of projection onto A_1 with A_0 as the related axis, the dashed lines the limit of projection onto A_0 with A_1 the related axis. Of the two areas enclosed by these sets of boundary lines, that associated with projection onto A_1 enclosed by the dotted lines is the greater. Thus by the criterion of correctly approximated area, A_1 should be the axis projected onto, leaving A_0 as the related axis. Again the two requirements of the first reconstruction set and correctly approximated area are not in conflict and it remains to find some aspect of the geometry that appropriately indicates A_0 as the related axis.

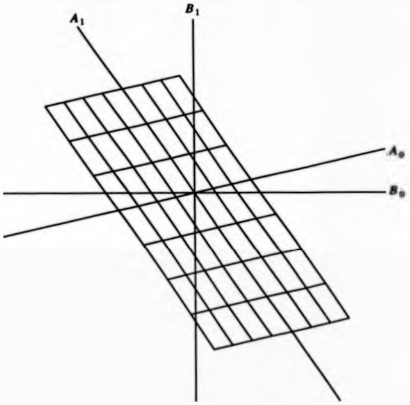


Figure 8.4 Illustration of how the choice of related axis for B_1 by the angle between B_1 and the axis A_1 can be incorrect.

The basic understanding of the related axis is as the axis that sets the boundary of the point space that the approximated basis vector first intersects. The distance along the approximated axis corresponding to the different boundaries of the axes can be calculated from the basis matrix. If

$$B_1 = x\,A_0 + y\,A_1 \qquad 8.4.7$$

then with a maximum value on the axis of *range*, the intercept points of B_1 with the boundaries are

$$A_0 \text{ boundary intercepts } \frac{range}{x}\,,\ A_1 \text{ boundary intercepts } \frac{range}{y} \qquad 8.4.8$$

The first boundary to be intercepted is associated with the smaller of these values corresponding to the greater of x and y. In this case, by referring to equation 8.4.4, the greater coefficient value is associated

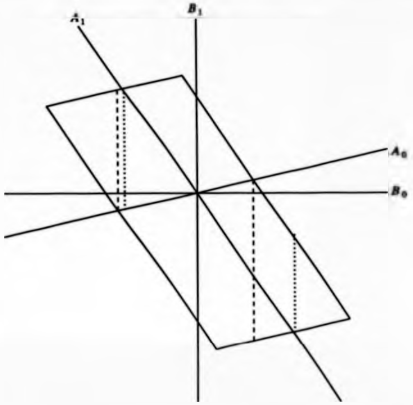


Figure 8.5 The limits of the correctly approximated point space, the dashed line corresponds to A_0 as the related axis, the dotted line corresponds to A_1 as the related axis.

with A_0, the desired result. Therefore in general, the related axis should be chosen as that having the greatest coefficient when the basis vector being approximated is expressed in terms of the axes. This method does not conflict with the naive choice made in section 7.1, nor with the equally naive choice made at the start of the three point example and investigation in section 8.1. At the more suspect choice, the point at which this departure was made with expressions 8.4.1 and 8.4.2, the basis matrix is given as statement 8.3.13. From this, axis ${}^1A_0$ has the greater coefficient in B_1 and the choice is again unchanged.

Before leaving the investigation of the related axis, the anomaly of the factor within the projection algorithm that indicated the problem can be reviewed. As described earlier, depending on the basis matrix and choice of related axis, the factor within the projection part, the projection factor can be greater than unity. The underlying projection mechanism is simply

$$p_{n+1,i} = p_{n,i} - \left\lfloor \frac{{}^n b_{n,i}}{{}^n b_{n,j}} p_{n,j} \right\rceil \qquad 8.4.9$$

It might be assumed that in an analogous fashion to a linear transform, as n increases the $p_{n,i}$ become small. However this is only true for a point near to some subset of the axes that are either approximated first, or with a linear transform those first in the sequence describing the point. For an arbitrary point, its linear transform coefficients will not have any particular distribution of magnitudes. Similar behaviour will also be expected of the MT, about which some fairly simple observations can be made. For a point close to the basis vector being approximated, from the projection equation 8.4.9

$$p_{n,i} \approx \frac{{}^n b_{n,i}}{{}^n b_{n,j}} p_{n,j} \qquad 8.4.10$$

and equation 8.4.9 thus represents a decrease in magnitude of $p_{n,i}$. However as the point moves away from the basis vector such that the sign of $p_{n,j}$ or $p_{n,i}$ changes then the expression must cause an increase in magnitude of $p_{n+1,i}$. If the point is close to the origin then the distance moved to effect this change can be small, though the final value of $p_{n+1,i}$ will still be small. For quite large areas or volumes, 'quadrants' perhaps, depending on the signs of the co-ordinates, the projected value must increase, which seems counter-intuitive. Even in 'quadrants' where the co-ordinate signs match with the projection factor, the space where the point projects closer to the origin is reduced by the possibility of

$$\frac{{}^n b_{n,i}}{{}^n b_{n,j}} p_{n,j} > 2 p_{n,i} \qquad 8.4.11$$

causing $p_{n,i}$ to change sign and increase in magnitude. Thus projection can be expected sometimes to increase the magnitude of the co-ordinates independently of the magnitude of the projection coefficient. As a sub-optimal choice of the related axis does not cause breakdown of the algorithm it seems this coefficient affects only in some sense the stability of the algorithm. The co-ordinate values are more likely to decrease if the projection coefficient is less than unity.

Returning to direct consideration of the three point transform and the calculation of the second coefficient, re-stating the choice of related axis as ${}^1A_0$ the next step is a further projection. Projection parallel to B_1 is straightforward :-

$$p_{2,0} = \left[p_{1,0} - \left\lfloor p_{1,0} \frac{{}^1 b_{1,0}}{{}^1 b_{1,0}} \right\rceil \right] \bmod -4,3 \qquad 8.4.12$$

$= 0$

$$p_{2,1} = \left[p_{1,1} - \left\lfloor p_{1,0} \frac{{}^1b_{1,1}}{{}^1b_{1,0}} \right\rfloor \right] \bmod -4,3 \qquad 8.4.13$$

8.5. The Second Correction Factor

Having performed the second projection, a correction factor is required to enable calculation of the second coefficient, with the problem now reduced to having two values. The derivation of the next correction factor is slightly more complicated than the earlier two point case as the axis vectors are not orthogonal nor of unit length. The case is illustrated in figure 8.6 using the same notation for various points as used in figure 8.2. The layout of axis and basis vectors approximates the actual configuration as represented in equations 8.3.8 and 8.3.9. The choice of position for point p, giving the point a negative ${}^1A_1$ co-ordinate was made only to make more clear the diagram and increase its similarity with figure 8.2. The consequences of the negative co-ordinate on the ${}^1A_1$ axis are discussed below. Referring to figure 8.6 once again the points p and q have the same ${}^1A_0$ co-ordinate, both points are on the same line parallel to ${}^1A_1$. The point p projects parallel to B_1 to the point s on the B_2 vector giving a difference in B_1 co-ordinates between p and q of |sq|. When the distance |sq| is measured on the related axis ${}^1A_0$ it becomes the distance |xy|. The distance |xy| is the correction factor when measured in units of length the modulus of ${}^1A_0$, an important qualification not obvious from figure 8.6. A difference from figure 8.2 is that the distance |oy| is less than the distance |ox|, this is linked to the sign of the ${}^1A_1$ co-ordinate which is discussed later. The derivation of the correction factor follows that described in section 8.2, though all the points in figure 8.6 are co-planar. The correspondence follows to the point at which the distance |od| is shown to be equal to the distance |cq| in equation 8.2.9. After this stage a slightly different approach is used for this derivation to make more clear the different axes and basis coefficients being used.

From equation 8.3.9

$${}^1A_1 = {}^0b_{1,1}B_1 + {}^0b_{2,1}B_2 \qquad 8.5.1$$

As $p_{2,1}$ is the co-ordinate of the projected point r on axis ${}^1A_1$ in units of modulus ${}^1A_1$, noting the approximation this involves as the projection is not geometrically precise :-

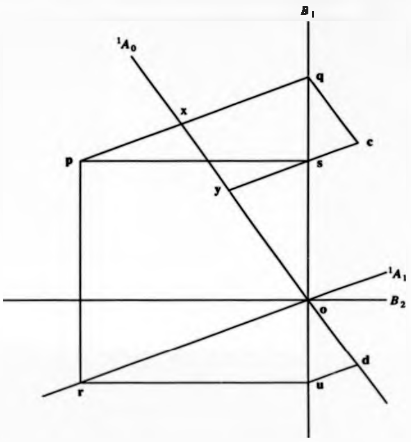


Figure 8.6 Construction to find for a two point transform the correction factor xy for the point p that projects to the point r with axes that are not orthogonal.

$$\overline{\mathbf{or}} = p_{2,1}\,{}^1A_1 \qquad 8.5.2$$

$$= p_{2,1}\,{}^0b_{1,1}\,B_1 + p_{2,1}\,{}^0b_{2,1}\,B_2 \qquad 8.5.3$$

and the component of this parallel to B_1 equal to the distance |ou| measured in unit modulus is

$$\overline{\mathbf{ou}} = p_{2,1}\,{}^0b_{1,1}\,B_1 \qquad 8.5.4$$

from equation 8.3.13

$$B_1 = {}^1b_{1,0}\,{}^1A_0 + {}^1b_{1,1}\,{}^1A_1 \qquad 8.5.5$$

and from figure 8.6 od is the ${}^1A_0$ component of ou and so

$$\overline{\mathbf{od}} = p_{2,1}\,{}^0b_{1,1}\,{}^1b_{1,0}\,{}^1A_0 \qquad 8.5.6$$

As od has the same direction on the axis ${}^1A_0$ as xy, the distance |od| must be added as the correction facto In units of modulus ${}^1A_0$ with again the appropriate rounding in the form of the floor operator

$$\mathrm{cf} = \left\lfloor p_{2,1} \, {}^{0}b_{1,1} \, {}^{1}b_{1,0} \right\rfloor \qquad 8.5.7$$

As stated above, for the example shown in figure 8.6, the correction factor must reduce the distance $|ox|$ to the distance $|oy|$ and hence must be negative. As the coefficients in the expression are positive, and in the diagram $p_{2,1}$ is negative, the correction factor is also negative. This agrees with the process of adding the correction factor to the related coefficient.

More generally at whatever iteration in the sequence depending on the number of starting values, the two point case at the n th iteration with $p_{n+1,j}$ measured on ${}^{n}A_{i}$ and the j th axis being the related axis

$$\mathrm{cf} = p_{n+1,j} \, {}^{0}b_{n,j} \, {}^{n}b_{n,j} \qquad 8.5.8$$

This can be compared with the more restricted result of the earlier two point transform derivation shown in equation 7.3.24. At the zeroth iteration with t_1 equivalent to $p_{1,0}$ and A_1 for the related axis, giving $n = 0, i = 0, j = 1$ the correction factor is

$$\mathrm{cf} = p_{1,0} b_{0,0} b_{0,1} \qquad 8.5.9$$

showing the results of the two point transform derivations to be consistent.

Having derived the correction factor the next transform coefficient is

$$t_1 = p_{1,0} + \left\lfloor p_{2,1} \, {}^{0}b_{1,1} \, {}^{1}b_{1,0} \right\rfloor \qquad 8.5.10$$

and finally

$$t_2 = p_{2,1} \qquad 8.5.11$$

which completes the coding part in the derivation of the three point MT.

8.6. Decoding

Decoding a complete set of 'transform' coefficients is quite straightforward, being simply the reversal of the coding process, as follows

input t_2

$$p_{2,1} = t_2 \qquad 8.6.1$$

input t_1

$$p_{1,0} = t_1 - \left\lfloor p_{2,1} \, {}^{0}b_{1,1} \, {}^{1}b_{1,0} \right\rfloor \qquad 8.6.2$$

$$p_{1,1} = p_{2,1} + \left\lfloor p_{1,0} \frac{{}^1b_{1,1}}{{}^1b_{1,0}} \right\rfloor \qquad 8.6.3$$

input t_0

$$p_{0,2} = t_0 - {}^0b_{0,2}\left[{}^0b_{0,0}p_{1,0} + {}^0b_{0,1}p_{1,1} \right] \qquad 8.6.4$$

$$p_{0,0} = p_{1,0} + \left\lfloor p_{0,2} \frac{{}^0b_{0,0}}{{}^0b_{0,2}} \right\rfloor \qquad 8.6.5$$

$$p_{0,1} = p_{1,1} + \left\lfloor p_{0,2} \frac{{}^0b_{0,1}}{{}^0b_{0,2}} \right\rfloor \qquad 8.6.6$$

The decoding of a partial set of coefficients as produced during a progressive transmission is more complicated and requires some extra computation when compared with a conventional linear transform. At the start of transmission t_0 is received and reconstruction from this is well defined, as one point in the first reconstruction set as shown in equation 8.1.7. The same result can also be drawn from equations 8.6.4, 8.6.5 and 8.6.6 with the assumption that the correction factor is zero, in turn produced by the assumption of zero co-ordinates on the ${}^1A_0$ and ${}^1A_1$ axes. This is equivalent to assuming that t_1 and t_2 are zero. The assumption is normal for standard transform coding as zero is the expected or mean value of these coefficients. Thus to produce the first approximation, strictly only equations 8.6.4-6 are required, though if t_1 and t_2 were set to zero, the entire sequence commencing with equation 8.6.1 through to equation 8.6.6 could be used. The first set of display values is

$$q_{0,2} = t_0 \qquad 8.6.7$$

$$q_{0,0} = \left\lfloor t_0 \frac{{}^0b_{0,0}}{{}^0b_{0,2}} \right\rfloor \quad \text{mod } 0,7 \qquad 8.6.8$$

$$q_{0,1} = \left\lfloor t_0 \frac{{}^0b_{0,1}}{{}^0b_{0,2}} \right\rfloor \quad \text{mod } 0,7 \qquad 8.6.9$$

After the first display has been generated the next coefficient t_1 will be received. The decoding level represented by equations 8.6.2 and 8.6.3 must now be used as these contain references to t_1. The correction factor component of equation 8.6.2 is assumed zero since $p_{2,1}$ is assumed zero as a result of t_2 being assumed zero. In general the correction factor, at the lowest level (highest level index) with a non-zero coefficient, will have a zero value, and any offsets due to lower level co-ordinates (in this case $p_{2,1}$) will be zero. From equations 8.6.1 and 8.6.3 will come values for $q_{1,0}$ and $q_{1,1}$ which are shown in equations

8.6.10 and 8.6.11.

$$q_{1,0} = t_1 \qquad 8.6.10$$

$$q_{1,1} = \left\lfloor q_{1,0} \frac{{}^1b_{1,1}}{{}^1b_{1,0}} \right\rfloor \quad \text{mod } 0,7 \qquad 8.6.11$$

These values will be passed to the next level which must also obtain a value for t_0 and this will be available from display memory as $q_{0,2}$ as shown in equation 8.6.7. At this stage, in order not to make the notation over-cumbersome, an algorithmic approach is adopted and the assignment operator ':=' appears in statements 8.6.12, 8.6.13 and 8.6.14

$$q_{1,2} := q_{0,2} - {}^0b_{0,2}\left[{}^0b_{0,0}q_{1,0} + {}^0b_{0,1}q_{1,1} \right] \quad \text{mod } 0,7 \qquad 8.6.12$$

$$q_{1,0} := q_{1,0} + \left\lfloor q_{1,2} \frac{{}^0b_{0,0}}{{}^0b_{0,2}} \right\rfloor \quad \text{mod } 0,7 \qquad 8.6.13$$

$$q_{1,1} := q_{1,1} + \left\lfloor q_{1,2} \frac{{}^0b_{0,1}}{{}^0b_{0,2}} \right\rfloor \quad \text{mod } 0,7 \qquad 8.6.14$$

It is important to note that if t_2 were coded to have a non-zero value then the values available at this stage for $q_{1,0}$ and $q_{1,1}$, are not necessarily correct in the sense that they are identical to those generated in the coding sequence. This causes two different types of error when these values are used to generate the final values for display at this iteration. The first error is that the correction factor is not correct causing an imprecise correction of $q_{0,2}$ making the display value $q_{1,2}$ incorrect. However the final values of $q_{1,0}$ and $q_{1,1}$ (assignments 8.6.13 and 8.6.14) are doubly incorrect since they are based on an incorrect value and are projected an incorrect distance. This is not to say the display values are greatly in error, though it is not completely clear whether the two error mechanisms interact. The overall decoding technique differs completely from that of a normal linear transform where newly received coefficients simply cause addition of a vector related set of values to the already decoded data. This is made more obvious at the next and final (for three points) iteration when t_2 is supplied to the decoding process. The first step follows the model of 8.6.1, however the second step based on equation 8.6.2 requires a t_1 value. The t_1 value is no longer available directly, it was assigned to $q_{1,0}$ in 8.6.10, but $q_{1,0}$ was subsequently modified in 8.6.13. To find t_1 the three values $q_{1,0}, q_{1,1}, q_{1,2}$ must be used with the forward coding algorithm to regenerate t_0 and t_1, with t_2 to be appended. This produces a full set of coefficients required for inverse transformation. Although this progressive display system depends on the intermediate display data, the data cannot be used in its displayed form except in the trivial case when a t_0 value is directly available at the second iteration. For transforms over more points, the majority of decoding iterations must re-code the display to extract the already transmitted coefficients. In terms of computation required this is expensive compared with the complexity of progressive decoding using a conventional linear transform. With this algorithm the inverse transformation requires extra computation because it is iterative yet even this is doubled by the need to find previous coefficients. The re-coding can be viewed as part of the cost of exact invertibility. The intermediate display structure needs to be more sophisticated and more carefully maintained than the simple process of summation used for conventional linear transforms.

8.7. Examples

In figure 8.7 are shown some examples of the progressive sequences for sets of three data points using the algorithm developed above. The range of the data values is 0,7 and the consequent modulus limits are built in to the MT algorithm, though none of the examples exhibit behaviour caused by these limits being exceeded. For comparison a normal linear transform sequence is also given. The linear transform values are produced without quantisation and the intermediate values retained to full machine precision with just a few digits reproduced for convenience. The values are reproduced to three digits as it is not clear how, in practice they would be rendered displayable, by truncation or rounding. However, either process can be readily performed by the reader, this approach permitting the best possible result for the linear transform. Despite this flexibility the differences between the two algorithms are nowhere greater than one digit in these examples. Though it must be admitted that this represents a potentially high proportional error, this is due only to the restricted range used.

To summarise, in this section the principles developed for the two point MT have proved with the appropriate elaboration to be suitable for a three point MT. The projection mechanism has become more sophisticated, requiring the definition of new axis vectors. These vectors lie within the space of co-dimension one defined by the remaining set of basis vectors, as successive basis vectors are deleted. These new axis vectors are not orthogonal nor of unit modulus and the correction factor has been reformulated for use with these axis vectors. These new axis vectors have also had some effect on the way in which the related axis is chosen, though in practice the result is unchanged from the earlier two point

	New Algorithm			Linear Transform		
data	4	2	5	4·00	2·00	5·00
code	5	2	−2	6·31	0·91	−2·08
first display	2	3	5	2·34	3·52	4·69
second display	4	4	5	3·09	3·68	4·19
final display	4	2	5	4·00	2·00	5·00
data	4	4	4	4·00	4·00	4·00
code	4	2	0	6·69	1·82	−0·07
first display	2	3	4	2·48	3·72	4·97
second display	4	4	4	3·97	4·05	3·97
final display	4	4	4	4·00	4·00	4·00
data	0	0	4	0·00	0·00	4·00
code	3	−2	−1	2·97	−2·18	−1·55
first display	1	2	3	1·10	1·66	2·21
second display	0	1	4	−0·68	1·26	3·40
final display	0	0	4	0·00	0·00	4·00

Figure 8.7 Some examples of reconstruction sequences for both the MT and the underlying linear transform over three points.

case.

There may be applications for a three point transformation such as developed above. The most obvious area is that of colour rendition by using red, green and blue values. It should be possible to derive a monochrome luminance value by taking the mean of the three values and then resolving the colour with two further co-ordinates. This might be useful for making the colour component of an image progressive.

9. The Mimic Transform over more than Three Points

In this section the remaining issues relevant to generalising the MT to operate over more than three points will be examined, the resulting transform will be referred to as an n point transform. The transform will be expanded algebraically for verification of its iterative behaviour and some early assumptions about rounding and modulus limits will be reassessed. Two areas require further attention for operation over an increased number of points, these are the correction factor, and issues around the inverting of the axis matrix as described in section 8.3. The basic projection mechanism does not require any further development, nor does the choice of related axis.

9.1. The Correction Factor

The first point for consideration is the correction factor. The construction required for the derivation is shown in figure 9.1 which has similarities with all the previous diagrams relating to the correction factor. Illustrated is the configuration at some iteration n with the basis vector B_n being approximated and the axis ${}^nA_j$ as the related axis. In common with figure 8.2 the vectors ${}^nA_j, B_n$ and point p may not all be co-planar, though any pair will be co-planar. If all three components were co-planar then pq would be co-linear with xq, unfortunately figure 9.1 lacks the simple perspective of figure 8.2. The lines pq and xq are in the sub-space defined by a constant ${}^nA_j$ co-ordinate. By working in the plane of B_n, p the same proof as associated with figure 8.2 can be used to show that |aq| is equal to |uo|. Similarly, working in the plane of ${}^nA_j, B_n$ showing |od| equal to |qc|. The problem is again then to find the distance |od| in terms of the distance |or| with due regard to the signs of various coefficients.

The point r which has been constructed to have a zero co-ordinate on ${}^nA_j$ may have non-zero co-ordinates on the remaining axes, giving the vector $\overline{or}$ as

$$\overline{or} = \sum_i p_{n+1,i}\,{}^nA_i \qquad \text{(i ranging over the remaining axes) 9.1.1}$$

The vector $\overline{ou}$ is the B_n component of $\overline{or}$ which will be the summation of the B_n components due to each axis ${}^nA_i$. By reference to the appropriate axis matrix (an example would be equation 4.3.6) the B_n component of ${}^nA_i$ is ${}^0b_{n,i}$, giving $\overline{ou}$

$$\overline{ou} = \sum_i {}^0b_{n,i}\,p_{n+1,i}\,B_n \qquad \text{(i ranging over the remaining axes) 9.1.2}$$

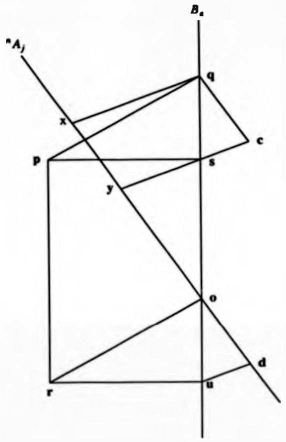


Figure 9.1 Construction to find for an n point transform the correction factor **xy** for the point **p** that projects to the point **r**, where the axes are not orthogonal and **p**, ${}^nA_j$ and B_n are not co-planar.

The component of $\overline{\mathbf{ou}}$ parallel to ${}^nA_j$ is found using a coefficient from the appropriate basis matrix (an example would be equation 4.3.2) giving the ${}^nA_j$ component of B_n as ${}^nb_{n,j}$, giving

$$\overline{\mathbf{od}} = {}^nb_{n,j} \sum_i {}^0b_{n,i}\, p_{n+1,i}\, {}^nA_j \qquad \text{(i ranging over the remaining axes) } 9.1.3$$

and the correction factor is $|\mathbf{od}|$ in units of ${}^nA_j$ so the ${}^nA_j$ is dropped and the required rounding included to give

$$\mathrm{cf} = \left\lfloor {}^nb_{n,j} \sum_i {}^0b_{n,i} p_{n+1,i} \right\rfloor \qquad \text{(i ranging over the remaining axes 9.1.4)}$$

Again with this example it may not be clear if the correction factor is to be added or subtracted, the related co-ordinate **x** is shown in the figure being reduced by the correction factor to the point **y**. From the figure, if the various components of or are on the positive parts of the axis then the corresponding ${}^0b_{n,j}$ coefficients must be negative. Alternatively as represented in the figure by the intended similarity with figure 8.6 if the ${}^0b_{n,j}$ coefficient is positive then the point r is on the negative part of an axis. The resulting correction factor is negative in either case, requiring it to be added to the co-ordinate on the related axis. However, this does not seem a very rigorous derivation and it is cross checked in a purely algebraic fashion later to confirm the result.

The result is consistent with the correction factor derived for the zeroth iteration of the three point example in equation 8.2.13. Using ${}^0A_2$ as the related axis giving $j = 2$, the zeroth iteration giving $n = 0$, and working over three points leaves two remaining axes $i = [0,1]$ to give

$$\text{cf} = \left\lfloor {}^0b_{0,2} \left[{}^0b_{0,0}p_{0,1} + {}^0b_{0,1}p_{1,1} \right] \right\rfloor \qquad 9.1.4$$

which agrees with the original derivation. This new formulation can also be tested against the next (first) iteration of the same three point example. The related axis for this iteration is ${}^1A_1$, producing the parameters $j = 0, n = 1, i = [1]$ giving the correction factor

$$\text{cf} = \left\lfloor {}^1b_{1,0}\,{}^0b_{1,1}p_{2,1} \right\rfloor \qquad 9.1.5$$

identical to result 8.5.7. This shows the new derivation of the correction factor at least to be consistent with these specific examples derived less generally. The formulation of equation 9.1.4 can be considered as a scalar product between the projected point set and some part of the basis vector, modified by a multiplicative constant. The ${}^0b_{n,j}$ coefficients within the summation are the original basis vector coefficients and this seems to show that the algorithm still retains some feature of the simple linear transform mechanism. Exactly how this partial linear transform operates on the projected rather than the original co-ordinates does not seem clear, but will certainly make a very great difference, so perhaps the similarity is superficial.

9.2. The Matrix Sequence and Inversion of the Axis Matrices

The second point for consideration is the inversion of the axis matrix as described in section 8.3. Before discussing this in detail it is useful to look at the overall sequence of operations used to derive the various matrices and coefficients. The starting point for the algorithm is the straight-forward definition of the basis vectors in terms of the set of axes, the axis on which the co-ordinates of any point would be given. This matrix denoted ${}^0b$ will be called a basis matrix and is defined as the square matrix in this equation

$${}^0b \text{ matrix defined in } \begin{bmatrix} {}^0b_{0,0} & {}^0b_{0,1} & {}^0b_{0,2} & \cdots & {}^0b_{0,n} \\ {}^0b_{1,0} & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ {}^0b_{n,0} & \cdot & \cdot & \cdots & {}^0b_{n,n} \end{bmatrix} \begin{bmatrix} {}^0A_0 \\ \cdot \\ \cdot \\ {}^0A_n \end{bmatrix} = \begin{bmatrix} B_0 \\ \cdot \\ \cdot \\ B_n \end{bmatrix} \qquad 9.2.1$$

As this matrix is unitary orthogonal then its transpose is its inverse and what will be called an axis matrix is defined as the transpose of ${}^0b$ to give ${}^0a$ defined in

$${}^0a \text{ matrix defined in } \begin{bmatrix} {}^0b_{0,0} & {}^0b_{1,0} & {}^0b_{2,0} & \cdots & {}^0b_{n,0} \\ {}^0b_{0,1} & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ {}^0b_{0,n} & \cdot & \cdot & \cdots & {}^0b_{n,n} \end{bmatrix} \begin{bmatrix} B_0 \\ \cdot \\ \cdot \\ B_n \end{bmatrix} = \begin{bmatrix} {}^0A_0 \\ \cdot \\ \cdot \\ {}^0A_n \end{bmatrix} \qquad 9.2.2$$

The projection process requires at each iteration a new set of axis vectors which do not have a component in one of the basis vectors. This basis vector is the one being approximated and conventionally the basis vectors are indexed by approximation sequence. This indexing results in top to bottom, row by row selection from the basis matrix. As can be seen in equation 9.2.2, the operation of removing the first basis vector component from all the axes is simply performed with the axis matrix by deleting the left-most column of coefficients and B_0 from the top of the basis vectors. In addition after each projection iteration one of the axis vectors, the related axis, is no longer required. As has been shown, in general the ordering of the related axes is not fixed, and the axis will be within the body of the matrix rather than at an edge. To remove the axis vector from the axis matrix the appropriate row is deleted and the axis vector deleted on the right-hand side. After the first iteration the basis vector zero is deleted and the related axis ${}^0A_j$ is removed giving the matrix ${}^1a$ defined as

$${}^1a \text{ matrix defined in } \begin{bmatrix} | & {}^0b_{1,0} & {}^0b_{2,0} & {}^0b_{3,0} & \cdots & {}^0b_{n,0} \\ | & {}^0b_{1,1} & \cdot & \cdot & \cdots & \cdot \\ | & \cdot & \cdot & \cdot & \cdots & \cdot \\ | & not\ j & - & - & - - & - \\ | & {}^0b_{1,n} & \cdot & \cdot & \cdots & {}^0b_{n,n} \end{bmatrix} \begin{bmatrix} - \\ B_1 \\ \cdot \\ \cdot \\ B_n \end{bmatrix} = \begin{bmatrix} {}^1A_0 \\ \cdot \\ \cdot \\ not\ j \\ {}^1A_n \end{bmatrix} \qquad 9.2.3$$

Simply by repeatedly deleting entries from the axis matrix, the sequence of matrices ${}^0a$, ${}^1a$, ${}^2a$, $\cdots$ can be created. This gives at each iteration the definition of the axis vectors in terms of the basis vectors. However for projection the basis vectors must be known in terms of the axis vectors, which requires the inversion of an axis matrix at each iteration. Actually only one basis vector needs to be known for projection but this fact has not been made use of, the most simple approach is to invert the matrix. As both a row and column of the axis matrix have been deleted, the set of equations represented by the matrix is neither under-determined nor over-determined. Although the columns of ${}^0a$ were orthogonal the deletion of a row means that the property of orthogonality no longer holds and the matrix cannot be inverted by transposition. It is not then immediately clear that the matrix can be inverted. The deletion of a row and column is a very extreme operation on a matrix and on the vectors it represents. Assuming temporarily that the matrix so produced can be inverted, that is the matrix is non-singular, at the first iteration the inversion will produce the matrix ${}^1b$ defined as

$${}^1b \text{ matrix defined in } \begin{bmatrix} - & - & - & - & | & - \\ {}^1b_{1,0} & {}^1b_{1,1} & {}^1b_{1,2} & \cdot & not\ j & {}^1b_{1,n} \\ {}^1b_{2,0} & \cdot & \cdot & \cdot & | & \cdot \\ \cdot & \cdot & \cdot & \cdot & | & \cdot \\ \cdot & \cdot & \cdot & \cdot & | & \cdot \\ {}^1b_{n,0} & \cdot & \cdot & \cdot & | & {}^1b_{n,n} \end{bmatrix} \begin{bmatrix} {}^1A_0 \\ \cdot \\ \cdot \\ not\ j \\ {}^1A_n \end{bmatrix} = \begin{bmatrix} - \\ B_1 \\ \cdot \\ \cdot \\ \cdot \\ B_n \end{bmatrix} \qquad 9.2.4$$

The indexing of this matrix and all the basis matrices, matches that of the ${}^0b$ matrix and is the transpose of the indexing of the axis matrices in order to maintain the identities

$$\sum_i {}^n a_{j,i}\, {}^n b_{k,i} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases} \qquad \text{(i ranging over the remaining axes) } 9.2.5$$

The problem remains of the singularity or otherwise of the series of matrices ${}^n a$, and this is dependent on the determinants of the matrices. The problem can be resolved by consideration of a factorisation of the determinant of a matrix. The determinant of a matrix can be expressed in terms of a summation over a single row or column of the matrix. The summation is of the products of each row or column element with its associated cofactor and a sign term. The cofactor of an element is the determinant of the matrix (called the 'minor' of the element) formed by deleting the row and column, of which the element is a member, from the larger matrix. The sign term simply depends on the row/column position in the matrix of the particular element in the product term. By considering the summation, if the determinant of a matrix is non-zero then for any expansion of the determinant, at least one cofactor must be non-zero. Consider the first axis matrix, which is invertible and hence has a non-zero determinant. This determinant can be expanded by the left-most column, which will be deleted to form the next axis matrix. The expansion will use the set of matrices produced by deleting this column and each row in turn. One of this set of matrices produced by row deletion will be the next axis matrix. For the summation to be non-zero, the determinant of at least one of these matrices must be non-zero. This implies that at least one matrix produced by the deletion of the left-most column and one row, must have a non-zero determinant and be invertible. This does not mean the row corresponding to the related axis chosen by the criterion described earlier can be deleted without producing a singular matrix. If the matrix is singular then the next best choice for the related axis must be tried. This process can be repeated until an axis is found which can be deleted to produce an invertible matrix, which is guaranteed. As has been shown in section 7.7, albeit not in a truly general fashion, use of a sub-optimal related axis does not cause the transformation to break down in respect of invertibility. A similar conclusion could be reached by consideration of the operation of the modulus arithmetic. The argument that from a non-singular matrix, the left column (any column) and one row can be deleted to produce a non-singular matrix, can be applied iteratively to generate a sequence of non-singular matrices.

Therefore, starting with a unitary orthogonal matrix there is at least one 'path' through the matrix generated by a set of related axes for the operation of the algorithm. The existence of this 'path' seems to be the strongest property that can be attached to the series of matrices due to the extreme nature of the operation of deletion of a row and column. Although the starting matrix is unitary orthogonal, only the property of non-singularity is required to show the existence of the 'path', the unitary and orthogonality properties disappear at the first iteration.

9.3. Data range and Modulus Limits for Projection

Although the issues relating to the order of the MT have been addressed, the issue of the range of the data has rather been over-looked. All the examples thus far have been based on a range of eight values 0-7, this was purely for convenience in respect of the reproduced table size. The algorithm developed should be usable with any range of data, but the 'power of two' ranges have generally been expected to be used.

For these ranges of the form 2^n, the modulus limits for projection $-2^{n-1}, 2^{n-1} - 1$ have been used consistently. Alternatively for a range of data 0,N−1 the modulus limits can be shown as −N/2,N/2−1 though this implies N is divisible by 2. The modulus limits for inverse transformation are the same for all iterations but the last, the final iteration giving the display values, for which the modulus limits are $0, 2^{n-1}$. With these modulus limits the transformation has been used successfully with ranges other than eight for the data. Although the range of the data defines the number of possible projection values, it was shown earlier in section 7.7 that changing the modulus limits can affect the size of the correctly approximated area. This was demonstrated for a point space with only positive co-ordinates and orthogonal axes. For most iterations of the MT the point space is nearly symmetrical about zero and the axes are not orthogonal. Considering these differences the geometry that describes the size of the correctly approximated area requires further investigation. It should be possible to use mixed sets of residues in the MT, only quite obviously requiring that at each iteration of the decoding operation the residue set matches that used for coding. No further work has been done on the modulus limits but there may possibly be scope for improvement.

Projection in any number of geometrical dimensions presents no problems, at the n th iteration using ${}^nA_j$ as the related axis. The co-ordinates before projection on the remaining axes are $p_{n,i}$, with i ranging over the remaining axes. The co-ordinates after projection are $p_{n+1,i}$, with data in the range $0, N-1$ and N divisible by 2 these co-ordinates are given by

$$p_{n+1,i} = p_{n,i} - \left\lfloor p_{n,j} \frac{{}^nb_{n,i}}{{}^nb_{n,j}} \right\rfloor \quad \bmod -N/2, N/2-1 \qquad \text{(i over the rem. axes) 9.3.1}$$

This gives $p_{n+1,j} = 0$ and this co-ordinate can be omitted from further consideration.

9.4. Expansion of the Projection Iterations

In this section, the correction factors that have been derived in previous sections are expanded algebraically. This has been done for two reasons. The first is as a cross-check on the geometry that has been used in the derivations. The second is to look more closely at the way in which the various approximations affect the overall transform result. It was not possible to perform the expansion in any useful way if the floor and modulus operators were retained. The geometry that has been used did not take account of these approximations and the cross-check remains valid. The effect of the approximations can only be gauged by looking at various stages in the expansion and seeing how the different components are affected.

The expansions are derived from an n point transform, the first expansion is of t_0, using the j'th axis as the related axis and approximating B_0.

The projected points are

$$p_{1,i} = p_{0,i} - p_{0,j} \frac{{}^0b_{0,i}}{{}^0b_{0,j}} \quad i = 0,n \tag{9.4.1}$$

The coefficient t_0 is the co-ordinate on the related axis plus the correction factor

$$t_0 = p_{0,j} + {}^0b_{0,j} \sum_{i=0}^{n} {}^0b_{0,i} p_{1,i} \tag{9.4.2}$$

$$= p_{0,j} + {}^0b_{0,j} \sum_{i=0}^{n} {}^0b_{0,i} \left[p_{0,i} - p_{0,j} \frac{{}^0b_{0,i}}{{}^0b_{0,j}} \right] \tag{9.4.3}$$

$$= p_{0,j} + {}^0b_{0,j} \sum_{i=0}^{n} {}^0b_{0,i} p_{0,i} - \frac{{}^0b_{0,j}}{{}^0b_{0,j}} p_{0,j} \sum_{i=0}^{n} {}^0b_{0,i} {}^0b_{0,i} \tag{9.4.4}$$

The fraction in front of the second summation has the value one, as does the second summation itself. The set of coefficients ${}^0b_{0,i}$ makes up the first basis vector which is of unit length and the summation can be considered either as the scalar product of the vector with itself or the square of the modulus.

$$t_0 = p_{0,j} + {}^0b_{0,j} \sum_{i=0}^{n} {}^0b_{0,i} p_{0,i} - p_{0,j} \tag{9.4.5}$$

$$= {}^0b_{0,j} \sum_{i=0}^{n} {}^0b_{0,i} p_{0,i} \tag{9.4.6}$$

Showing that to within a constant factor, and without approximations being taken, t_0 has the form of the true transform coefficient. The constant factor is to be expected from the way in which an MT coefficient is measured on the related axis. Otherwise, as no approximations were made this confirms the derivation of the correction factor. Perhaps disturbing is the way in which the co-ordinate on the related axis is removed by part of the correction factor, which contains the summation giving the conventional transform.

For further investigation, coefficient t_1 is expanded with the second related axis index k. The projection is

$$p_{2,i} = p_{1,i} - p_{1,k}\frac{{}^1b_{1,i}}{{}^1b_{1,k}} \quad i = 0,n \; i \neq j \qquad 9.4.7$$

and the expansion

$$t_1 = p_{1,k} + {}^1b_{1,k}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}p_{2,i} \qquad 9.4.8$$

$$= p_{1,k} + {}^1b_{1,k}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}\left[p_{1,i} - p_{1,k}\frac{{}^1b_{1,i}}{{}^1b_{1,k}}\right] \qquad 9.4.9$$

$$= p_{1,k} + {}^1b_{1,k}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}p_{1,i} - \frac{b_{1,k}}{b_{1,k}}p_{1,k}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}{}^1b_{1,i} \qquad 9.4.10$$

The following identity can be used to reduce expression 9.4.10.

$$\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}{}^1b_{1,i} = 1 \qquad 9.4.11$$

This can be clarified by looking at equations 9.2.1 to 9.2.5. The row of the ${}^0b$ matrix in the summation also forms a column of the ${}^0a$ matrix. Taking the j value out of the index effectively makes this a column of the ${}^1a$ matrix. The ${}^1b_1$ part forms a row of the ${}^1b$ matrix which is derived as the inverse of the ${}^1a$ matrix. The indexes operate such that this row and column correspond in the matrices and as the product is a unit matrix, the product of this row and column is one. The ability to show this identity relatively easily is part of the motivation for the otherwise unusual indexing used for the axis matrices.

$$t_1 = p_{1,k} + {}^1b_{1,k}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}p_{1,k} - p_{1,k} \qquad 9.4.12$$

$$= {}^1b_{1,k}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}p_{1,i} \qquad 9.4.13$$

$$= {}^1b_{1,k}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}\left[p_{0,i} - p_{0,j}\frac{{}^0b_{0,i}}{{}^0b_{0,j}}\right] \qquad 9.4.14$$

$$= {}^1b_{1,k}\left[\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}p_{0,i} - \frac{p_{0,j}}{{}^0b_{0,j}}\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}{}^0b_{0,i}\right] \qquad 9.4.15$$

From the original definition of ${}^0b$ there is the orthogonality property of two rows of the matrix

$$\sum_{i=0}^{n} {}^0b_{1,i}{}^0b_{0,i} = 0 \qquad 9.4.16$$

$$\sum_{\substack{i \neq j \\ i=0}}^{n} {}^0b_{1,i}{}^0b_{0,i} + {}^0b_{1,j}{}^0b_{0,j} = 0 \qquad 9.4.17$$

$$\sum_{\substack{i\neq j\\ i=0}}^{n} {}^0b_{1,i}\,{}^0b_{0,i} = -\,{}^0b_{1,j}\,{}^0b_{0,j} \tag{9.4.18}$$

Substituting this into the expression for t_1 in equation 9.4.15 gives

$$t_1 = {}^1b_{1,k}\left(\sum_{\substack{i\neq j\\ i=0}}^{n} {}^0b_{1,i}\,p_{0,i} - \frac{p_{0,j}}{{}^0b_{0,j}}\left[-\,{}^0b_{1,j}\,{}^0b_{0,j}\right]\right) \tag{9.4.19}$$

$$= {}^1b_{1,k}\left(\sum_{\substack{i\neq j\\ i=0}}^{n} {}^0b_{1,i}\,p_{0,i} + p_{0,j}\,{}^0b_{1,j}\right) \tag{9.4.20}$$

$$= {}^1b_{1,k}\sum_{i=0}^{n} {}^0b_{1,i}\,p_{0,i} \tag{9.4.21}$$

This result shows that the second coefficient has the correct format when no approximations are made. Though this does not constitute any sort of proof for the remaining coefficients, the layout of the above expansion shows that the remaining coefficients should have the appropriate form.

Unfortunately this has not revealed much of the effects of the rounding. For the higher index coefficients the contribution of the correction factor directly is small due to its being calculated from a reduced set of projected values. More of the transform information is embedded in the projected points, and hence in the related value. However the projected points degrade cumulatively due to the rounding involved so the error mechanism changes with the iteration, but this information is of no great utility. The expansion also confirms that the choice of related axis cannot cause, if wrongly made, the complete breakdown of the transform as no special properties were ascribed to the related axes.

9.5. Rounding of the Correction Factor

An early decision in the development of the MT was the rounding of the correction factor by taking the floor of the value as was done in the projection part. This was an investigative decision based on the expected methods of calculating the inverse. The decision has been left unchanged and at no stage in the development has any other aspect been affected by or required to affect this method of rounding. Any form of rounding could then be used and it would seem most appropriate to round to the nearest integer value rather than take the floor. Conceptually this is simply performed by adding 0.5 to the correction factor and taking the floor, though practically this may not be optimum. There remains a potential problem with the correction factor associated with the accuracy of its calculation. The absolute value affects the accuracy of the approximation to the linear transform but does not affect invertibility. The only requirement for invertibility is that the value calculated during coding is exactly matched by the value calculated during decoding. It is conceivable that if coding and decoding are carried out on different equipment using floating-point arithmetic, the values may not be identical before rounding. If this difference occurs at the rounding decision value then the final correction factors could be different. This can be prevented by using a standard floating-point implementation with careful matching of algorithms. Alternately, as discussed later, the algorithm can be arranged to be table driven, requiring only integer operations and if identical tables are used then all the calculated values should match exactly. This rounding problem can also affect the projected values as they are also sensitive to calculation error at the rounding decision boundary value. The same options that would solve the problem for the correction factor will also solve the problem for the projected values.

9.6. Summary of Operation

The MT has now been extended to work over a larger point set and its operation has to some extent been verified by the above expansion. The expansion did not take account of the various rounding operations and there seems to be no other way than implementation of a specific MT to investigate the problems that these might cause. The rounding is very much associated with the way the MT has been developed by designing it to have the required behaviour first over two points, then three, and finally to a lesser extent over n points. Unfortunately it is not really possible to visualise the operation and desired behaviour with point sets of size greater than three and in some ways the investigation of a specific MT is still developmental.

10. Potential Improvements to the Mimic Transform

After developing the MT fully to operate over any number of points, some of the issues raised during the design stage need to be reconsidered, and the overall algorithm consisting of several iterations considered. The problem of cumulative error was not obvious during design as it arises only from the iterative nature of the algorithm, which is not overly apparent in a step-by-step design. The problem of wrap-around error has been noted throughout the development but even now no solution is offered and only tentative proposals can be made.

10.1 Projection with Reduced Cumulative Error

The algebraic comparison of the MT and a conventional linear transform in section 9.4 did not recognise either the floor operations or the modulus operations within the MT. The use of the floor operator in the MT will cause the rounding of many values in all instances, whereas the effect of the modulus operator is more data dependent, and any effect it has is of a gross nature. During the operation of the MT without distortion the floor operator has the dominant effect of the two. The floor operator is first encountered operationally in the projection part of the MT, causing a rounding of the projected point. As it is the rounded projected point that is used in the calculation of the correction factor this causes a difference between the actual and ideal correction factors. This difference is increased as the floor operator is also applied to the output of the calculation of the correction factor. In this way the immediate result of the operator within one iteration is the changing of the correction factor and therefore also the output coefficient at that iteration. Overall this is a minor effect as the rounded projected values are passed on to the next iteration where the process is repeated, increasing the divergence of the MT values from the linear transform in a cumulative fashion.

The only way to prevent or alleviate this increasing divergence of the two algorithms is to make the values carried over between iterations more accurate representations of the projected values. To keep the amount of information constant at each iteration, the number of different values the points can take must remain fixed. If the numerical spacing between values is reduced for increased accuracy then the overall range is also reduced. Assuming the modulus operator is used to control the range of the projected values, a smaller set of points will project to the reduced range and more points will wrap-around into the range. This will reduce the size of the correctly approximated area, which makes it doubtful whether or not such a technique would be at all useful. However some variation of an improved resolution projection might be of value if the MT is re-designed to obviate the errors caused by the modulus operation.

Returning to the basics of an earlier example, figure 10.1 shows two sets of points. One set similar to those shown in figure 7.2 is marked 'O' and constructed from

$$p_0 = \left\lfloor p_1 \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor$$

and the second set marked 'X' is constructed from the initially obscure identity

$$p_0 = 0{\cdot}5 \left\lfloor p_1 \frac{1{\cdot}2}{0{\cdot}8} \right\rfloor$$

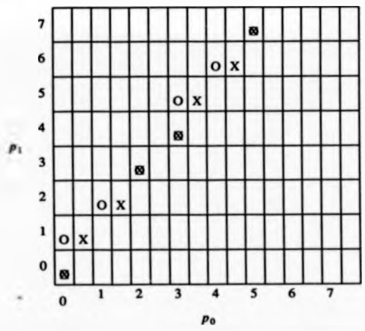


Figure 10.1 Two different sets of points approximating the straight line.

The second construction generates some values with a fractional part of one half, and extra resolution has been indicated on the p_0 scale of figure 10.1 to show this. The second construction has the factor within the floor multiplied by two and a corresponding division outside the floor. This is a clumsy way of generating a 'rounding down to half unit boundary' operation which can be denoted more cleanly as

$$\lfloor\ \rfloor_{0{\cdot}5}$$

With this notation the second set of points in figure 10.1 can be better described by

$$p_0 = \left\lfloor p_1 \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor_{0{\cdot}5}$$

The rounding down to half unit causes some of these points to be placed one half unit to the right of the fully rounded point. It can be seen that the new points marked 'X' more closely approximate the straight line which otherwise has some distinct steps. Where the two points, one from each set, are not coincident within a one-unit co-ordinate square, a data point which must have a unit co-ordinate will project to the value −0·5 using the modified projection

$$p_{1,0} = p_{0,0} - \left\lfloor p_{0,1} \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor_{0{\cdot}5}$$

with the extended iteration notation for the co-ordinates. All the points in a row of the table will project to the same sort of value, either integer or fractional, causing some rows to be offset by 0·5. As mentioned above the points can take only a fixed number of distinct values, and in this small example the number is 8. Working from the original projection target set of −4 to 3, dividing by two gives the set

−2, −1·5, −1, −0·5, 0, 0·5, 1, 1·5

The modulus notation can be extended to work with similar sets and the modified operator '$\mathrm{mod}_{0{\cdot}5}\, x,y$' has as its set of residues the integers in the range x,y inclusive, and all these values with one half added. The projection can now be more fully specified with the correct set of residues as

$$p_{1,0} = p_{0,0} - \left\lfloor p_{0,1} \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor_{0{\cdot}5} \quad \mathrm{mod}_{0{\cdot}5}\, -2,1$$

The set of projected values is shown in figure 10.2 with the correspondence

−2	−1·5	−1	−0·5	0	0·5	1	1·5
a	b	c	d	e	f	g	h

As can be seen, the projected value is not unique within any one row, as all the projected values occur twice, and because of this the projection is not invertible. A reconstruction will always find the point that projects into the range −2,1·5, and never reach the other point that projects there via the modulus operator. To make the projection invertible it is necessary to force half of each row to project to integer values

p_1										
7	g	a	c	e	g	a	c	e	g	a
6	h	b	d	f	h	b	d	f	h	b
5	b	d	f	h	b	d	f	h	b	d
4	c	e	g	a	c	e	g	a	c	e
3	e	g	a	c	e	g	a	c	e	g
2	f	h	b	d	f	h	b	d	f	h
1	h	b	d	f	h	b	d	f	h	b
0	a	c	e	g	a	c	e	g	a	c
p_0	−2	−1	0	1	2	3	4	5	6	7

Figure 10.2 Projection to reduced length with increased precision.

and half to project to fractional values, only half the line will then project accurately. Early attempts to do this were based on a fixed segmentation of the data space in turn based on producing accurate projected values in areas of expected high data point occurrence. However, early efforts were found to be reliant on some symmetric properties of the example being used for investigation. One such property of the example is that it has equal numbers of integer and fractional rows, an example without this property would be a transform based on a vector of slope 6/7. For this line the set of values indicating the type of value on each row is

x	0	1 -	2	3	4	5	6	7
$\left\lfloor x \cdot \frac{6}{7} \right\rfloor_{0 \cdot 5}$	0	1	1·5	2.5	3	4	5	6

To guarantee in a general way that overall there are equal numbers of points that project to integer and fractional values, it is necessary to force each row to have equal numbers of the two types of points. This can be done by forcing a change of type on the point that does not project onto the interval $-2, 1{\cdot}5$, the type change can be performed by adding 0·5 to the projected value. The projection, in an algorithmic form is

$$t_1 := p_{0,0} - \left\lfloor p_{0,1} \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor_{0.5}$$
$$\text{if } \left[\left[t_1 < -2 \right] \text{ or } \left[1{\cdot}5 < t_1 \right] \right] \text{ then } t_1 := \left[t_1 + 0{\cdot}5 \right] \quad \text{mod}_{0.5}\, -2,1$$

Without considering the correction factor, $p_{0,0}$ can be recovered by using the following

$$s1 = \left[t_1 + \left\lfloor p_{0,1} \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor_{0.5} \right] \quad \text{mod}_{0.5}\, -4,3$$
$$s2 = \left[\left[\left[t_1 - 0{\cdot}5 \right] \ \text{mod}_{0.5}\, -2,1 \right] + 4 + \left\lfloor p_{0,1} \frac{0{\cdot}6}{0{\cdot}8} \right\rfloor_{0.5} \right] \quad \text{mod}_{0.5}\, -4,3$$

and then taking whichever of s1 or s2 is integral to be the correct value for $p_{0,0}$.

This line of investigation was taken much further, and included an implementation as part of a complete MT such as that described later. The results of its use indicate that any further detailed description is not justified, though some general comments are in order. Increased resolution projection would probably not be used for each iteration of an MT but mixed with the normal projection method. The two types can be freely mixed as the output of an increased resolution projection has the same format as the normal projection. At the appropriate iteration, the increased resolution values are interpreted differently, but this is transparent to any succeeding stages. Using twos-complement arithmetic it is easy to move between two representations for either integer only or fractional representations. This is performed simply by multiplication or division as expected and can be performed practically by shifting. When working with the fractional representation it is only necessary to maintain the one extra bit of precision in a straight-forward fashion. However when a full PTD implementation is used there becomes apparent a problem with this technique that may not be solvable. It occurs when a truncated coefficient sequence is being decoded, which causes the projected values at a decoding iteration to be slightly different from those that existed during coding. With the normal projection method this slight difference is propagated with no difficulty. With the increased resolution projection, an incorrect interpretation of what is the least significant bit as indicating a result to be either integer or fractional can cause a large displacement in the resulting value. Effectively the inaccuracy can cause the s1 and s2 values to be interchanged, giving in the case of the example a 4 element shift along a row. This has an extremely detrimental effect on the decoded values, making the algorithm unusable in its described form.

As mentioned above, this is a general form of increased resolution projection and a more specific approach would likely meet with more success. Such a specific method may be difficult to design for more than two points, but remains a possibility and represents an increased foundation on which to construct an improved MT.

10.2. Fundamental Problems

Much thought has been given to the problem of wrap-around in the mimic transformation but no real progress has been made in improving performance. The problem is apparently more difficult than the derivation of the MT itself, it is not clear how much of the MT mechanism will carry over and how it would fit into a new transformation. Some ideas on possible avenues of investigation are presented below.

The MT does not draw heavily on Knowlton's original work aside from the concept but a new transformation would be expected to have more similarity, thus exhibiting the same distorted transformation that is not forced into gross error to accommodate extreme data values. A rather vague observation based on the quadrant nature of Knowlton's tables is that for two quadrants of the differentiator table, figure 3.2 lower left and upper right, the differentiator value behaves identically to the projected value. In these quadrants some composite values are close to the mean, but others less accurate. In the remaining two quadrants the relationship between the differentiator and projected value disappears thought the mean/composite is good in some areas. The author for a long time held the opinion that Knowlton's approach was based on 'lines' orthogonal to the vector of interest whereas the projection mechanism is based on 'lines' parallel to the vector. By these assumptions the two are very different though it now seems the distinction is less well defined. An obvious place to review the MT is the subtraction within the projection part that actually produces the results that go out of range. Perhaps the subtraction could be replaced by some operation with more moderate behaviour, the final value being more dependent on the related value. A controlled difference transformation is required for this and might be provided by Knowlton's work which once again becomes relevant. Knowlton's table is usually viewed as providing a composite value close to the mean, to be resolved by a differentiator. A similar transformation could provide a good approximation to the difference to be resolved by a value similar to the mean. A problem with this approach is that the result of a difference transform cannot be a single value but must be two values, and the stumbling block is what to do with the second value, as all the subtractions within a projection iteration use the same related value. A somewhat less than attractive idea is to perform all the subtractions in a fixed sequence and allow the related value to be changed by each differencing operation. It would surely be very difficult to analyse the behaviour of such an operation.

A more fundamental approach is to return to the basic idea suggested by Knowlton, discarded at the start of the derivation of the MT, of fitting lines or surfaces or solids into the transform space. The MT could be used to locate a point with integer transform coefficients within some space and it may be possible to produce a secondary transform based on regions within the point space that maps coefficients to new values. This approach is attractive in that it does not require fundamental re-working of the MT but is complicated by having to operate with data that may have already been garbled by the original projection mechanism. Although the way forward is not clear, what is apparent, is that a considerable amount of further work is required to modify the MT, and the question arises as to whether the anticipated product justifies the effort. This question may be partially answered by investigating a realistic coding exercise to find out how the MT performs, and if there are any aspects of its behaviour of interest other than the features that have been deliberately designed into it. The additional problem this presents is the difficulty of judging future performance by using a transform so flawed, but the amount of effort required for the investigation is not great compared with that required to re-work the MT, and some results are presented in the next section.

11. The Cosine Mimic Transform (CMT)

The Mimic Transformation, having been developed to operate over any number of points, can now be tested with some more practical and demanding set of basis vectors than used in the preceding examples. In earlier sections two different approaches to PTD were discussed, one based on a hierarchical application of a very simple transform, the second on a more straight-forward application of a transform over a greater number of points. There would seem to be a choice between these two areas for further investigation of a Mimic Transformation. For the hierarchical tree based algorithms, Knowlton's method and its variants satisfactorily address the two point transform problem, leaving the four point transform to be investigated for quad-tree decompositions and the eight point transform for the oct-tree decompositions. In the area of transforms over the greater number of points there seems to be a concentration of effort on the cosine transform over block sizes of the order of 8 and 16, as exemplified by the work of Lohscheller [1984]. As the MT has known flaws which make it unlikely to be usable without further work, it was decided that the best way forward was, if possible to provoke the manifestation of these flaws to better understand the problems they present. The following material should then be viewed as precedent for further work on the MT rather than the evaluation of a finished new transform. At this stage in the development there are parts of the MT that are not fully understood and these will become apparent in the evaluation. After considering these factors it was decided to investigate an eight point cosine transform. The hierarchical transforms would perhaps better represent the possible applications of the MT but were considered to present too many special case conditions for an exhaustive test. A general linear transform of which the cosine transform is representative was considered more useful in this respect. The mimic of the DCT will be referred to as the Cosine Mimic Transform or CMT. Undoubtedly interweaved in this decision is the idea that the MT might arouse more wide-spread interest if associated with a particularly often used linear transform. The choice of transformation over 8 elements matches the work already performed and is otherwise commonly used. A 16 element transform may give opportunity for greater compression but this was considered not to justify a change from the 8 element block length.

11.1. Formulation of the CMT

The DCT was given in section 6.1, although not in a normalised fashion and is repeated here after proper normalisation. With data values $x(i)$ and transform coefficients $X(i)$

$$X(0) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x(i)$$

$$X(u) = \sqrt{\frac{2}{N}} \sum_{i=0}^{N-1} x(i) \cos\left| \frac{\left[2i+1 \right] u \pi}{2N} \right|$$

$$u = 1, \ldots, N-1$$

Numerically this produces for $N = 8$ the set of vectors shown in figure 11.1, this matrix being used for the further investigation. The slight difference between the modulus of the coefficients of vectors 0 and 4 is a numerical anomaly arising from the different methods used to calculate the vectors. This was not considered to be a problem and was left unchanged. The vectors are also shown graphically in their normalised form to a common scale in figure 11.2. The sign changes of each vector are visible, showing the vectors are ordered by the number of sign changes along the set of coefficients. The number of sign changes is commonly referred to as the sequency of the vector. Although the vectors are based on a sinusoidal function, the vectors do not exhibit this shape and it is dangerous to draw many parallels with frequency spectrum and the continuous and discrete Fourier transforms.

B_0	coeffs.	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535
B_1	"	0·4904	0·4157	0·2778	0·0975	−0·0975	−0·2778	−0·4157	−0·4904
B_2	"	0·4619	0·1913	−0·1913	−0·4619	−0·4619	−0·1913	0·1913	0·4619
B_3	"	0·4157	−0·0975	−0·4904	−0·2778	0·2778	0·4904	0·0975	−0·4157
B_4	"	0·3536	−0·3536	−0·3536	0·3536	0·3536	−0·3536	−0·3536	0·3536
B_5	"	0·2778	−0·4904	0·0975	0·4157	−0·4157	−0·0975	0·4904	−0·2778
B_6	"	0·1913	−0·4619	0·4619	−0·1913	−0·1913	0·4619	−0·4619	0·1913
B_7	"	0·0975	−0·2778	0·4157	−0·4904	0·4904	−0·4157	0·2778	−0·0975
		on axis A_0	on axis A_1	on axis A_2	on axis A_3	on axis A_4	on axis A_5	on axis A_6	on axis A_7

Figure 11.1 Coefficients of the 8-D Discrete Cosine Transform, this is the ${}^{0}b$ matrix.

The first step in constructing the transformation that will mimic the DCT is to select the related axis for the first basis vector B_0. The matrix shown in figure 11.1 is what has previously been referred to as a basis matrix, and from the coefficients of B_0 the position of the coefficient of greatest magnitude is used

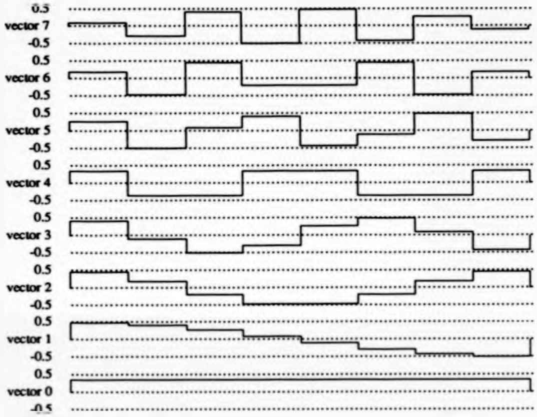


Figure 11.2 Graphical representation of the basis vectors of the 8-D Discrete Cosine Transform.

to choose the related axis. Of course in this case, as in others where the first vector is related to the mean, all the coefficients have the same value, and some other criterion must be used. Consider the mechanism of the transform. The data value at the location in the data vector corresponding to the related axis, called from here-on the related value, will be used as the starting value to construct the first transform coefficient. Therefore the closer the related value is to the mean, the smaller will be the correction factor and perhaps the better the approximation. Assuming a very simple model of the data in the form of a linear ramp then the middle value of the sequence will be approximately equal to the mean. With a vector of 8 elements there is no strictly middle element and based on the linear model an arbitrary choice of axis 4 is made for the related axis. However as shown in section 9.4, without considering the modulus limits, the transform coefficients are exactly correct independent of the choice of related axis. The choice of related axis cannot affect the accuracy of the transformation by the mechanism of the correction factor.

Instead, with the simple linear model, the projection mechanism dictates the choice of related axis at the first iteration. Adopting the earlier notation for values in the transformation, the starting data values are

$$p_{0,i} \quad i = 0,7$$

with range $0, N-1$. If the related axis is axis j then the projected values

$$p_{1,i} \quad i = 0,7$$

are given by

$$p_{1,i} = \left[p_{0,i} - \left\lfloor \frac{{}^0b_{0,i}}{{}^0b_{0,j}} p_{0,j} \right\rfloor \right] \quad \text{mod } -N/2, N/2-1 \quad i = 0,7$$

and as

$$b_{0,i} = b_{0,j} \quad i = 0,7$$

the fraction within the floor operator has the value one. As the $p_{0,j}$ co-ordinates can take only integer values, the floor operator is redundant giving

$$p_{0,i} = p_{0,i} - p_{0,j} \quad i = 0,7$$

For equal value basis coefficients, simply stated, projection is the subtraction in turn of the related value from the remaining data values. As an example, assume that the overall range of data values permitted is 0,7 and the values within the data block make up the ordered sequence 0-7. Some of the possible projections, depending on the choice of related axis are

	$p_{0,i} =$	0	1	2	3	4	5	6	7
$j = 0$	$p_{1,i} =$	0	1	2	3	4	5	6	7
$j = 7$	"	−7	−6	−5	−4	−3	−2	−1	0
$j = 4$	"	−4	−3	−2	−1	0	1	2	3

Considering that the modulus limits for projection are $-N/2, N/2-1$ giving in this case $-4,3$, choosing $j = 0$ will cause 4 values to exceed the modulus limits. Choosing $j = 7$ will cause 3 values to exceed the modulus limits and choosing $j = 4$ causes no values to exceed the limits. Although in many ways this is an unreal example, it seems reasonable to expect full scale changes to occur across an 8 element block. Less realistic in the simple linear model, though the overall result is clear, the transform can operate with full-scale change across one block but requires the related value to have an near exact mid-range value.

Although this requirement of the related value can be relaxed when the range across a block is less than full-scale, small local variations or noise on the related value can cause the projected values to go outside the modulus limits. Especially in areas of near full-scale image activity, the transform is vulnerable to operation in its regions of total breakdown. The only positive step to alleviate this problem is to choose a middle position in the block as the co-ordinate position corresponding to the related axis.

Having selected the related axis as $^0A_4$, to use the full notation, consideration moves to the axis matrix from which this and the vector B_0 will be deleted. This aspect of the development of the MT can be confusing, as it is the axis matrix and not the original basis matrix from which all the other matrixes are derived. From the axis matrix the basis vectors and individual axes are deleted to create the succession of axis matrices. The transform starts with a basis matrix as shown in figure 11.1, and as a first and unique step this is inverted to derive the starting axis matrix shown in figure 11.3; note that this defines the ^0A axes. The inversion is easy to perform as the initial basis matrix is unit orthogonal and can be inverted by transposition. This can be verified by comparing figure 11.1 and figure 11.3. From the axis matrix shown in figure 11.3, the axis $^0A_4$, and all components of the remaining axes on the basis vector B_0, are deleted. The matrix produced is shown in figure 11.4 where the ^1A axes are defined with no B_0 component.

$^0A_0$	coeffs.	0·3536	0·4904	0·4619	0·4157	0·3536	0·2778	0·1913	0·0975
$^0A_1$	"	0·3536	0·4157	0·1913	−0·0975	−0·3536	−0·4904	−0·4619	−0·2778
$^0A_2$	"	0·3536	0·2778	−0·1913	−0·4904	−0·3536	0·0975	0·4619	0·4157
$^0A_3$	"	0·3536	0·0975	−0·4619	−0·2778	0·3536	0·4157	−0·1913	−0·4904
$^0A_4$	"	0·3536	−0·0975	−0·4619	0·2778	0·3536	−0·4157	−0·1913	0·4904
$^0A_5$	"	0·3536	−0·2778	−0·1913	0·4904	−0·3536	−0·0975	0·4619	−0·4157
$^0A_6$	"	0·3536	−0·4157	0·1913	0·0975	−0·3536	0·4904	−0·4619	0·2778
$^0A_7$	"	0·3536	−0·4904	0·4619	−0·4157	0·3536	−0·2778	0·1913	−0·0975
		on vector B_0	on vector B_1	on vector B_2	on vector B_3	on vector B_4	on vector B_5	on vector B_6	on vector B_7

Figure 11.3 The first axis matrix for the CMT ^0a, the inverse of the matrix ^0b.

The ^1a matrix shown in figure 11.4, although derived from an easily invertible matrix is no longer unitary or orthogonal and must be inverted by a generalised inversion algorithm. There is no guarantee that a matrix produced in this way will be invertible but in this particular case the inversion is possible and a standard library program was used to produce the inverse. The new inverse forms the next basis matrix

$^1A_0$	coeffs.	0·4904	0·4619	0·4157	0·3536	0·2778	0·1913	0·0975
$^1A_1$	"	0·4157	0·1913	−0·0975	−0·3536	−0·4904	−0·4619	−0·2778
$^1A_2$	"	0·2778	−0·1913	−0·4904	−0·3536	0·0975	0·4619	0·4157
$^1A_3$	"	0·0975	−0·4619	−0·2778	0·3536	0·4157	−0·1913	−0·4904
$^1A_5$	"	−0·2778	−0·1913	0·4904	−0·3536	−0·0975	0·4619	−0·4157
$^1A_6$	"	−0·4157	0·1913	0·0975	−0·3536	0·4904	−0·4619	0·2778
$^1A_7$	"	−0·4904	0·4619	−0·4157	0·3536	−0·2778	0·1913	−0·0975
		on vector B_1	on vector B_2	on vector B_3	on vector B_4	on vector B_5	on vector B_6	on vector B_7

Figure 11.4 The second axis matrix for the CMT ^1a.

shown in figure 11.5.

B_1	coeffs.	0·5879	0·5133	0·3753	0·1951	−0·1802	−0·3182	−0·3928
B_2	"	0·9239	0·6533	0·2706	0·0000	0·2706	0·6533	0·9239
B_3	"	0·1379	−0·3753	−0·7682	−0·5556	0·2126	−0·1802	−0·6935
B_4	"	0·0000	−0·7071	−0·7071	0·0000	−0·7071	−0·7071	0·0000
B_5	"	0·6935	−0·0747	0·5133	0·8315	0·3182	0·9061	0·1379
B_6	"	0·3827	−0·2706	0·6533	0·0000	0·6533	−0·2706	0·3827
B_7	"	−0·3928	−0·7682	−0·0747	−0·9808	−0·9061	−0·2126	−0·5879
		on axis $^1A_0$	on axis $^1A_1$	on axis $^1A_2$	on axis $^1A_3$	on axis $^1A_5$	on axis $^1A_6$	on axis $^1A_7$

Figure 11.5 The second basis matrix for the CMT ^1b, the inverse of the matrix ^1a.

The generation of these matrices is not part of either the coding or decoding operation and the computational cost, unless very great, is not important. Only the top row of the ^1b matrix is used, as the next vector to be approximated is B_1 the top row gives the $^1b_{1,a}$ values for projection. These coefficients are also used to choose the related axis, which in this case is axis $^1A_0$, as it has the largest coefficient value. This choice can be used to illustrate the relationship between the magnitude based method, and the earlier described angle based method for choosing the related axis. From the axis matrix shown in figure 11.4 the axes $^1A_0$ and $^1A_7$ are identical in terms of their basis vector descriptions apart from the differing signs of some of the coefficients. These vectors have the same modulus and angle to the basis vector to be approximated, B_1, yet when the matrix is inverted axis $^1A_0$ has a clearly greater component. The difference between the two numerical values $^1b_{1,0}$ and $^1b_{1,7}$ is caused by the orientation and modulus of the other axes in the set. This is a case where the choice of related axis is not intuitively clear, because in some respects the axes appear equally good. However the method of choice seems clear and has not been challenged. Several instances of symmetrical axes arise during the calculation of the MT matrices for the cosine transform, but their significance, if any, is not known and these instances are ignored.

The series of operations of row and column deletion, followed by inversion to find the new basis matrix is repeated and the full series of matrices is given in appendix B. The resulting set of related axes is given by the indices

4,0,7,2,5,1,6,3

The choice in each case was by the largest coefficient and no choice produced a singular matrix that could not be inverted, in no case was there more than one coefficient with the greatest value. For the MT, only the top row of each basis matrix is required and these are shown collectively in figure 11.6. In turn each row of this matrix is used for projection and there is no value in the column for the related axis in subsequent rows. It is perhaps not obvious that some of these values should be greater than one, though no particular meaning has been attached to this. Although there are eight rows shown in figure 11.6 the last row is only present to simplify the algorithms that use a data structure containing exactly these values within the transformation programs. Altogether, a transformation program requires the basis vector information in figure 11.1 to calculate the correction factor, the projection information of figure 11.6 and the index sequence shown above defining the related axes.

B_0	=	${}^0A_i$	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535
B_1	=	${}^1A_i$	0·5879	0·5133	0·3753	0·1951	——	−0·1802	−0·3182	−0·3928
B_2	=	${}^2A_i$	——	−0·1533	−0·3192	−0·3066	——	0·5538	1·1533	1·5412
B_3	=	${}^3A_i$	——	−0·5556	−0·9808	−0·721	——	0·471	0·3444	——
B_4	=	${}^4A_i$	——	−0·3066	——	0·5198	——	−1·0467	−0·9554	——
B_5	=	${}^5A_i$	——	−0·8504	——	0·7804	——	——	0·5272	——
B_6	=	${}^6A_i$	——	——	——	−0·9932	——	——	−1·7534	——
B_7	=	${}^7A_i$	——	——	——	1·0000	——	——	——	——
	i		0	1	2	3	4	5	6	7

Figure 11.6 All the values used for projection, each row of this matrix is the top row of a basis matrix.

11.2. Some Example Results

A transformation program based on the numerical development above produced the approximation sequence shown in figure 11.7. The data part of the figure should be obvious but the code part can be misleading as it is in, essentially, twos-complement form and the first coefficient is offset by half the range of the data. The modulus for all coefficients and projections except the first is $-N/2, N/2-1$ whereas the first must have limits $0, N-1$. This can be made more consistent by subtracting $N/2$ from the data and adding it back just before display at each iteration. The range used for the data in this example is 0,63 for N equal to 64. Thus the first coefficient value, 61, is a twos-complement number that represents -3 calculated as $61-64$ and when the offset is added $-3+32=29$, the mean value which gives the first monotone display value. The code has been presented in its twos-complement form to emphasise the limited precision modulo arithmetic used extensively in the MT. Each display row corresponds to the addition of one code value (coefficient) to the displayable information. The program that produced this example also recoded the display information to find the already used coefficients in a full operation as would be required in a usable implementation. The adoption of the simple ramp pattern for the data is not linked to the above description of problems with projection over full-scale data, but is chosen only as a pattern whose approximation can be evaluated to some extent without immediate recourse to the mean square error values given. The transformation behaves at least superficially as expected, the approximation converges on the input data in an apparently acceptable fashion. The mean square error (mse) figures are perhaps not quite as expected, rising slightly just prior to the end of the sequence and being nearly constant for three iterations before this, showing that by this measure the approximation did not improve with extra information.

data		16	16	24	24	32	32	40	48		
code		61	47	5	6	62	0	62	59		
display	0	29	29	29	29	29	29	29	29	MSE	111·0000
"	1	15	17	21	26	32	37	41	43	"	8·2500
"	2	17	17	20	25	31	36	41	45	"	5·7500
"	3	14	17	23	26	29	33	40	48	"	2·5000
"	4	15	16	23	26	30	31	39	48	"	1·5000
"	5	15	16	23	26	30	31	39	48	"	1·5000
"	6	15	15	24	26	30	33	39	49	"	1·6250
"	7	16	16	24	24	32	32	40	48	"	0·0000

Figure 11.7 An example of the CMT operation.

These results can be compared with those of figure 11.8 which shows the results of applying a conventional DCT to the same data, each extra row corresponding to the addition of one extra coefficient. For this example the coefficients and intermediate values were stored at single real precision and only the output values converted to integer by the addition of 0·5 and truncating the result. In a real coding/decoding environment these coefficients would be quantised, and intermediate storage would be of reduced precision, thus the results are possibly better than might be achieved by a more realistic implementation. Both the integer result and the value with one decimal place are shown to make clear the effect of rounding on the way in which the sequence of display values converges. A surprising result in the comparison between the two methods is that the mean square error values for the MT are equal to or smaller than those of the DCT used directly. No conclusion should be drawn from a single example in this respect, it was never anticipated that an MT should perform better than its underlying transform. The matter is ascribed to a quirk of the rounding used, it may be that the rounding inherent in the MT can lead to consistently better results, but this topic has not been pursued. By the analysis of section 9.4, the coefficients produced by the DCT and CMT can be compared. Ideally the MT coefficients should be the product of the appropriate ${}^{n}b_{n,j}$ value and the corresponding linear transform coefficient. The ${}^{n}b_{n,j}$ value is the constant at the front of the correction factor for coefficient/iteration n and related axis j. When tested, the match between MT and linear transform coefficients is quite good, it should be noted that the appropriate basis vector coefficients at the end of the sequence are negative, accounting for the differing signs of the coefficients. The divergence of the CMT coefficients from the DCT coefficients caused by the rounding in the MT is not very large in this example.

It is quite easy to provoke the transformation into complete break-down of its mimic behaviour, although the final result is always equal to the input data. An example is shown in figure 11.9, where the mean square error figures are very large and meaningless in this context. There is apparently no connection at all between successive display iterations and it is an indication of the complicated internal machinations of the MT that the correct result can be obtained from even the penultimate line with the apparently harmless coefficient value 4. Note that this is simply the value 4.

Some further examples of the eight point approximation sequence are given in appendix C. From these, the above example and more detailed diagnostic dumps of the transform not reproduced here, it is possible to infer some pattern in the transform coefficients. The coefficients do not tend to become small or

data		16	16	24	24	32	32	40	48		
code		82	−28	4	−4	3	0	2	4		
		29·0	29·0	29·0	29·0	29·0	29·0	29·0	29·0		
display	0	29	29	29	29	29	29	29	29	MSE	111·0000
		14·9	17·1	21·0	26·2	31·8	37·0	40·9	43·1		
display	1	15	17	21	26	32	37	41	43	"	8·2500
		16·6	17·8	20·3	24·5	30·1	36·2	41·6	44·8		
display	2	17	18	20	24	30	36	42	45	"	6·7500
		14·6	18·2	22·7	25·8	28·7	33·9	41·1	46·8		
display	3	15	18	23	26	29	34	41	47	"	3·1250
		15·6	17·2	21·7	26·8	29·7	32·9	40·1	47·8		
display	4	16	17	22	27	30	33	40	48	"	2·3750
		15·3	17·8	21·6	26·3	30·3	33·0	39·5	48·1		
display	5	15	18	22	26	30	33	40	48	"	2·2500
		15·6	17·1	22·3	26·0	30·0	33·7	38·8	48·4		
display	6	16	17	22	26	30	34	39	48	"	2·2500
		16·0	16·0	24·0	24·0	32·0	32·0	40·0	48·0		
display	7	16	16	24	24	32	32	40	48	"	0·0000

Figure 11.8 An example of the DCT operation.

data		0	0	0	0	63	0	0	0		
code		32	0	0	1	2	63	2	4		
display	0	0	0	0	0	0	0	0	0	MSE	496·1250
"	1	0	0	0	0	0	0	0	0	"	496·1250
"	2	0	0	0	0	0	0	0	0	"	496·1250
"	3	0	0	1	0	1	0	0	1	"	480·7500
"	4	0	0	1	63	0	1	0	1	"	992·6250
"	5	-1	63	1	63	0	0	63	0	"	1984·7500
"	6	1	0	1	0	0	0	1	0	"	496·5000
"	7	0	0	0	0	63	0	0	0	"	0·0000

Figure 11.9 An example of the CMT operation with complete breakdown of approximation.

zero, even for naively patterned data such as the ramp values. This property is expected of a conventional linear transform and taken account of in any quantisation strategy. The MT coefficients cannot take fractional values so, in a sense, the comparison with the linear transform is incorrect. However there seems to be a tendency for the coefficients to become small in the middle indexes and increase again near the end of the sequence. A rather vague explanation has been developed for this. During the coding phase, the point represented by the coefficients thus far calculated does converge on the data point. However due to the rounding/truncation inherent in the transform it cannot converge precisely but must 'miss'. It is the increased activity at the end of the sequence that moves a point around in a small locus to effect the exact recovery of the data. No parallel can be drawn with a conventional transform, as the keeping of intermediate results as integer only values would cause the final result to be incorrect.

11.3. The CMT Applied to Image Data

Ultimately the transformation must be tested on actual image data with the results presented for subjective comment. Again, the GIRL image, reduced in size as in the earlier evaluations is used. The sequence of 8 images based on the CMT is shown in photograph 11.1. An 8×1 horizontally aligned block is used. The first sub-image is formed by inverse-transforming only the first coefficient, and results in a block of monotone grey level equal to the rounded mean of the data block. To create the second image the first image is re-transformed, to yield the first coefficient. The second coefficient is appended to the first and the pair inverse transformed to produce the second image. As both coding and decoding phases of the CMT are iterative they need only be carried out to the level appropriate to the number of coefficients available. This reduces the total amount of computation, which has been increased relative to a conventional linear transform, by the coding of the stored image to obtain already available coefficients. The sequence of coding/decoding progresses by the addition of one extra coefficient per image to the final image which is identical to the original data.

Subjectively the overall result is not good, displaying some gross errors where the transformation is obviously operating with values outside its projection modulus. No statistical measures or differences have been taken because the method could not be used for practical purposes in this form, as it is dominated by the flaw in the algorithm. Some measurements were taken of the frequency of occurrence of these flaws and they are discussed later. For comparison a conventional DCT has been used to generate the sequence of images in photograph 11.2 based on the same data, with no quantisation. The intermediate values were stored to real precision and the conversion to integer display values did not clamp under and over-range values to the limits of the display, but took the modulus with respect to the display range. This operation is similar to that internal to the MT and results in the black dots that are visible in the otherwise peak white areas where the stored value is greater than the maximum displayable value. The corresponding effect of white in black areas is not evident. The image is biased towards high pixel values and there are no negative values produced in the reconstruction sequence.

11.4. CMT Error Statistics

Some numerical counts of the excess value activity for the DCT, demonstrated in the images described above, are shown in figure 11.10. The number of blocks with excess values decreases to zero, with the two iterations at the end of the sequence having no occurrences of excess values. The blocks have only over-range values, limited to about one pixel over-range per block. No blocks had under-range values, and although not shown in the table, as expected the blocks with over-range values had no under-range values.

	number of blocks		number of values	
iteration	under-range	over-range	under-range	over-range
0	0	0	0	0
1	0	19	0	21
2	0	11	0	12
3	0	11	0	11
4	0	5	0	5
5	0	5	0	5
6	0	0	0	0
7	0	0	0	0

Figure 11.10 The number of out-of-range values for the DCT reconstruction with the reduced size GIRL image.

This shows that some of the errors produced by the CMT may be due to the underlying DCT producing out-of-range values for truncated sequences of coefficients. At this stage the interaction between the DCT and the MT is not sufficiently well understood to address this problem, but similar measurements were also taken for the CMT yielding the results shown in figure 11.11. These figures indicate very different behaviour to the DCT, perhaps the greatest difference, though not itself particularly visible, is that there are many blocks that have under-range values. These are probably occurring in the dark detailed areas of the image where the related value in the first projection is unrepresentative of the block causing early failure of the transformation. In other aspects as shown in the table, the CMT is generally much worse than the DCT. Many more blocks go over-range, and if it is relevant due to the nature of the transformation, more elements are out of range in any one faulty block. The bottom row of figure 11.11 would normally be expected to show zeroes, but for the numerical results, the final modulus operator was omitted effectively recording the number of blocks for which the first projection during coding goes out of range.

	number of blocks		number of values	
iteration	under-range	over-range	under-range	over-range
0	0	0	0	0
1	9	58	16	100
2	40	57	109	77
3	53	63	140	106
4	46	68	113	106
5	42	63	75	89
6	36	62	68	85
7	39	25	55	35

Figure 11.11 The number of out-of-range display values for the CMT reconstruction with the reduced size GIRL image.

Three decode error conditions within the CMT need to be differentiated. These are

i) error due to the original data causing projection error during coding

ii) error due only to the behaviour of the MT with truncated set of coefficients when the mimicked transform would operate correctly with a similar truncated sequence

iii) error due to the behaviour of the mimicked transform with truncated coefficients, in this case the DCT

The first error condition might be investigated by carrying over extra information from the coding phase to the decoding phase to isolate blocks with this property. To differentiate between the second two error types requires a clearer understanding of the geometry involved. The worst case under and over-range values for the DCT have been determined earlier as factors −0·388 and 1·388, the total being less than double the original range. With the MT it should be possible to add one extra bit to the intermediate value representation to accommodate this increase in range. A potential problem with this is conflict within an improved MT in which projection is moderated to never exceed the modulus limits, removing the projection wrap-around problem. If the MT can be modified in this way the possibility of excess values must be designed in at an early stage.

Some of these problems are only associated with the decoding phase and do not manifest themselves during coding, which can be investigated independently. Measurements were taken to determine in a more quantitative fashion the performance of the coding phase of the transform, including the more and less desirable properties. For this study the original full area GIRL image was used. This was a straightforward way of increasing the data for investigation over that of the smaller image and obviated any doubts over the simple averaging mechanism used to construct the sm ller images. With the 8×1 block-size the full image consists of 32768 (32k) blocks. The entire image was coded while recording the operation of the projection mechanism. For each iteration and position within the block, a count was kept of the projected values that exceeded the modulus limits, the modulus limits were then imposed before the remaining iterations. A count was also kept of the total number of blocks that produced at least one projection that exceeded the modulus limits. The result was 263 blocks, equivalent to approximately 0·8% of the total image. The total number of errors caused by projection out of range was 963 which corresponds to an average of approximately 4 errors per block out of a potential 27 projection operations. The complete set of error figures is shown in figure 11.12, but this table does not give any information about error occurrence in any one block. This means that below the first row of the table it is impossible to distinguish between the first error within a block and an error that might be caused primarily by a preceding error possibly in a different column (different index) in the same block. The distribution of errors shown in the top row of the table is as expected, with the greater number at each end of the block where the middle value becomes a poor predictor. Of the total number of errors, those in the top row occurring at the first projection account for approximately 30%. Because of the above caveat regarding the figures below the top row of the table it is difficult to draw further conclusions from the table. There seems to be no skew on the distribution of values, probably indicating that no unexpected error mechanism is operating. The reason behind the relatively high number of errors at the base of the two longer columns is not clear, possibly indicating breakdown of the transform in some cases, perhaps caused by some instability. Again not clear are the reasons behind some of the very high values not in the top row.

depth	0	127	68	18	0	0	0	11	61
"	1	0	73	83	4	—	0	24	74
"	2	—	0	13	3	—	4	90	0
"	3	—	5	0	43	—	3	5	—
"	4	—	7	—	29	—	0	75	—
"	5	—	0	—	62	—	—	11	—
"	6	—	—	—	46	—	—	0	—
"	7	—	—	—	0	—	—	—	—

Figure 11.12 Projection statistics for the original GIRL image, the number of times the modulus limits were exceeded for each position in the vector at each iteration.

Additional measurements were also taken to determine by what amount outside the modulus limits were the projected values that did exceed the limits. The results of this investigation are shown in figure 11.13, showing the mean value of excess, 'miss', in the projected value. The values in the top row of the table indicate that the first projection does not in general go out of range by any great degree; as expected the values decrease towards the position corresponding to the related axis. The relatively large value in row 1/column 1 cannot be explained, also the value at position row 2/column 6. These large values correspond to large error counts in figure 11.12, though the converse correspondence does not always exist with some high error counts corresponding to unexceptional mean 'miss' values. The zero values at the base of the columns indicate only the position of the related axis and might have been omitted.

mean miss	0	4·88	4·10	2·50	0·00	0·00	0·00	2·64	3·18
"	1	0·00	20·33	9·48	2·75	—	0·00	3·04	4·93
"	2	—	0·00	1·92	2·67	—	1·50	13·53	0·00
"	3	—	2·80	0·00	4·97	—	6·33	2·40	—
"	4	—	1·71	—	5·34	—	0·00	5·00	—
"	5	—	0·00	—	4·68	—	—	3·27	—
"	6	—	—	—	5·06	—	—	0·00	—
"	7	—	—	—	0·00	—	—	—	—

Figure 11.13 The average by which the modulus limits were exceeded, for the values that exceeded the limits, for the original GIRL image.

These then are the negative aspects of the transform, which cause the apparently poor subjective performance. There may be some statistical difference between the original GIRL image and the version reduced in size which causes the performance on the small image to differ from that anticipated by analysis of the larger data set. The small image contains 2048 blocks and if the 1% error rate is maintained then approximately 20 blocks will exhibit errors. As can be seen from figure 11.11 the actual number is far greater than this, as 64 blocks are shown by the last row to be in error at the first projection, a greater proportion than predicted by the large image statistics. This shows that the statistical value of a property important to the CMT has changed. The simple averaging used to create the small image may produce the pronounced discontinuities in the image to which the CMT is vulnerable causing the increased number of errors. Some of the visible block errors are caused by the underlying DCT, thus the number of faults caused by projection errors will be smaller than the number of observed faults. However as shown in figure 11.10 the number of errors due to the underlying DCT is relatively small, this being the third error type as described above. This leaves either the coding projection mechanism causing the bulk of the errors or the truncation of the coefficient sequence during decoding. To clarify this issue, measurements were made for the coding of the small image, which show the total number of blocks in error at the coding phase to be 131. Tables corresponding to figures 11.12 and 11.13 are shown in figure 11.14 and 11.15. By comparison it can be seen that the worst figures for the smaller image are only reduced by a factor of 4 compared with the area reduction which is 16. For the reduced size image the total number of blocks with at least one projection that exceeds the modulus limits is 131. This represents approximately 6.4% of the small image compared with a figure of 0·8% for the large image. The figure of 131 blocks with projection errors matches the statistics of figure 11.11 where the number of errors is constant, for many iterations, at approximately 100. The 'mean miss' figures for the small image are in much closer agreement with those for the large image which is perhaps surprising considering the number of projection errors. The one slight anomaly in the 'miss' values shown in figure 11.15 is the 7 in the top row, and it can be seen by reference to figure 11.14 to have occurred only once and can be discounted.

depth	0	33	16	8	1	0	3	11	18
"	1	0	26	18	3	—	3	34	68
"	2	—	2	0	1	—	10	50	0
"	3	—	5	0	7	—	4	2	—
"	4	—	1	—	6	—	0	42	—
"	5	—	0	—	10	—	—	12	—
"	6	—	—	—	8	—	—	0	—
"	7	—	—	—	0	—	—	—	—

Figure 11.14 Projection statistics for the reduced size GIRL image, the number of times the modulus limits are exceeded for each position in the vector at each iteration.

mean miss	0	4·45	3·94	4·25	7·00	0·00	3·33	4·09	3·50
"	1	0·00	15·81	13·00	6·00	——	6·67	5·00	6·50
"	2	——	1·50	0·00	2·00	——	5·20	15·16	0·00
"	3	——	6·80	0·00	6·00	——	5·50	6·50	——
"	4	——	1·00	——	3·17	——	0·00	6·17	——
"	5	——	0·00	——	6·40	——	——	3·17	——
"	6	——	——	——	4·38	——	——	0·00	——
"	7	——	——	——	0·00	——	——	——	——

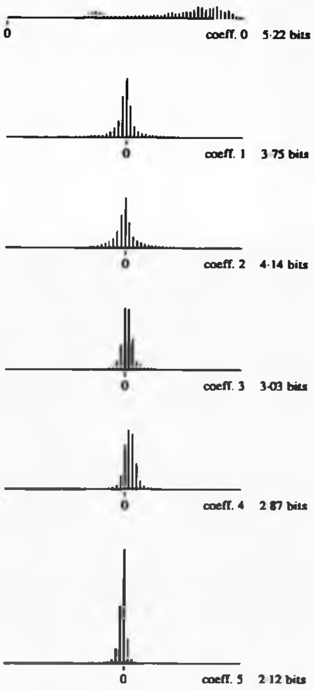
Figure 11.15 The average by which the modulus limits were exceeded, for the values that exceeded the limits in the reduced size GIRL image.

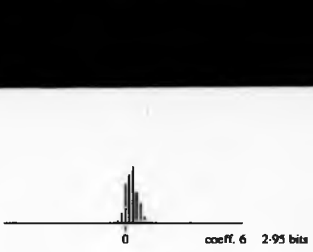
Clearly the averaging used to produce the small image, or in some sense the amount of detail in the small image, has adversely affected the operation of the CMT. The number of blocks with projection errors for the large image represented 0·8% of the image, which seems a good result, but this increased to 6·4% for the small image. It is possible that projection errors are largely causing the display errors rather than the cause being primarily the truncated coefficient sequences. Some of the measurements presented above have been of less than optimum usefulness due to the iterative nature of the transform. It would be useful to know more about the occurrence of the first projection error in the block when it occurs after the first iteration. However, no matter how many statistics are gathered the problems will not be reduced and the overall intent is still to re-design the algorithm thus limiting the desirability of further probing of the existing CMT.

11.5. CMT Coefficient Entropy

The projection mechanism is that part of the algorithm that causes the faulty behaviour, though any overall evaluation of the transformation looks at intermediate image values. A major part of the process to create these is the operation of the correction factor, which in conjunction with the projection mechanism produces the transformation coefficients. Information on the behaviour of the transform can also be gained by investigating these coefficients. Again the complete GIRL image was used as source material, and after coding, the histograms of the different coefficients values were constructed. These histograms are shown in figure 11.16. It is clear from these histograms that the distribution of values within the coefficients is skewed, the entropy values for each coefficient are also shown in the figure. All the coefficients except the first are based on the range $-N/2, N/2-1$; thus zero is not quite central and this might produce a slight bias towards the -1 value in the histograms. The histograms of the coefficients of higher index are however biased to a slightly positive result.

Figure 11.16





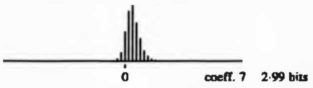
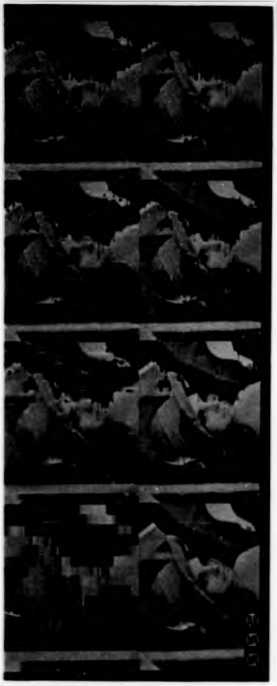


Figure 11.16 Histograms of the values of the CMT coefficients, negative values are conventionally represented as left of zero, the number of bits each coefficient represents is also shown.

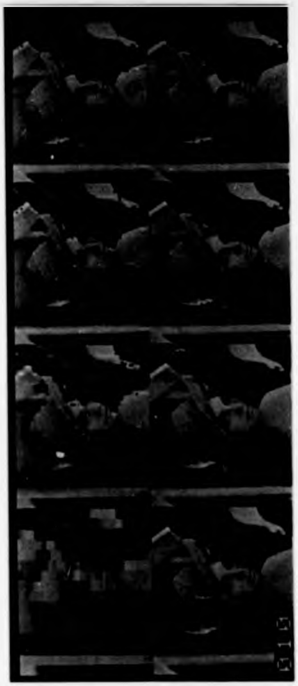
Summing over the entropy values of the CMT coefficients gives a total amount of information of 27·07 bits, the original data of 6 bits per pixel and 8 pixels per block gives 48 bits, and 8 times the second order entropy of 2·8 bits gives 22·4 bits. Comparing these figures, the performance of the CMT in terms of potential for redundancy reduction is not especially great. Transform coding is used because the information is segmented in such a way that some can be discarded without commensurate penalty in image quality. For exact reconstruction this cannot be done, but this is an issue of the problem specified rather than the MT. The entropy results do not seem perturbed by the those blocks with projection errors, but at a 1% occurrence rate the effect is negligible.

11.6. Extending the Range of the Display Values

The major result of the subjective evaluation is that the wrap-around effect in the projection mechanism causes severe visible degradation. Secondary but important factors are, that the underlying linear transform and the MT itself behave in such a way that truncated sequences of coefficients also cause overflow and subsequent visible wrap-around. The second problem can be countered in normal use by keeping the intermediate results to extended precision as suggested by Lohscheller [1984], 12 bits total for 8-bit data to attain the required error performance. The problem with the MT that makes a similar approach difficult, is that the MT expects intermediate values to go transiently out of range but then uses the modulus operator to bring those values back within range. It seems to the author that under normal operation with the MT, a value only goes over-range when its projection sequence was in error or because of a truncated coefficient sequence, and in either case the result would be a highly visibly distorted block. A possible approach is then to decode a truncated coefficient sequence using one extra bit of range and clamp this extra range at the extremes of the displayable range only for the display, by some method such as a look-up table. The immediate problem with this is that the final image of the sequence would need to be decoded with the correct range to give exact invertibility. A second and perhaps less clean solution is to store somewhere not in display memory a set of offsets which correct the excess range behaviour. A further problem with these solutions is that they are very unlikely to be workable if major modifications to the algorithm are made to solve the problems associated with the projection mechanism. There are also several problems with the implementation that make these suggested changes awkward, as the number of minor modifications increases, it becomes much more difficult to control their side-effects and interaction. For these reasons this direction has not been followed but does remain open as one useful next step in the investigation.



Photograph 11.1 Using the CMT taken over horizontal blocks of 8 × 1 pixels, each image is calculated by inverse transformation from an increasingly large set of coefficients. The first image, inverse transformed from one coefficient is at top-left, the sequence proceeds clockwise increasing the coefficient set by one per image. The final image with 8 coefficients is identical to the source image data.



Photograph 11.2 Using the DCT taken over horizontal blocks of 8 x 1 pixels, each image is calculated by inverse transformation from an increasingly large set of coefficients. The first image, inverse transformed from one coefficient is at top-left, the sequence proceeds clockwise increasing the coefficient set by one per image. Values outside the display range are brought into the range by the modulus operation rather than clamping, some of these out-of-range values values are visible as black dots in other wise white areas.

12. Extensions of the Cosine Mimic Transform

12.1. A Two-Dimensional CMT

Before moving on to some more novel techniques, a common coding method associated with linear transforms, 2-D coding will be discussed. A 2-D separable transform is created by successive applications of a 1-D transform in two passes over a 2-D block. The alignment of the 1-D blocks taken for transform is rotated through ninety degrees between the two passes. The natural choice of block-size following the 1-D work is 8×8 and a basically unmodified CMT was used to produce in the 2-D separable fashion the numerical reconstruction sequence shown in figure 12.1. There is no particularly preferred way of adding coefficients to produce a new displayable iteration of a 2-D transform, the main criterion seems to be the number of different displayed images required. With the MT, each iteration incurs more overhead than a conventional linear transform as the display data needs to be recoded. Eight iterations was an arbitrary choice, it gives the same number of images as the 1-D transform, having more gives an increasingly unwieldy amount of intermediate display information for anything other than viewing only. Each iteration is decoded from the coefficients contained within a square subset of the complete set of coefficients. The subset square of coefficients is taken from the corner of the full square array of coefficients that contains the overall mean value, and increases in linear dimension by one coefficient per image starting from a subset of one coefficient. By this method the first image uses only coefficient 0,0 the second 0,0 0,1 1,0 1,1. The fraction of total information available at each iteration is the rather unusual sequence

1·5%, 6·2%, 14%, 25%, 39%, 56%, 76%, 100%

The 2-D ramp data was chosen to make it possible to evaluate the numerical results as in earlier examples, and it was not considered worthwhile to produce a comparative DCT sequence. The 2-D MT separates cleanly the two components of the ramp and presents no surprises in the reconstruction sequence. No error figures or direct comparison with the DCT are given, it being considered they would be difficult to interpret, a single mean square error value for an iteration producing 64 values would not be useful if the error within the block had visible correlation.

16	16	16	16	24	24	24	24
16	16	16	16	24	24	24	24
24	24	24	24	32	32	32	32
24	24	24	24	32	32	32	32
32	32	32	32	40	40	40	40
32	32	32	32	40	40	40	40
40	40	40	40	48	48	48	48
40	40	40	40	48	48	48	48

data

32	26	31	29	32	33	34	31
49	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0

code

32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32

display 0

15	15	17	19	21	22	24	25
16	16	18	20	22	23	25	26
20	20	22	24	26	27	29	30
25	25	27	29	31	32	34	35
30	30	32	34	36	37	39	40
34	34	36	38	40	41	43	44
38	38	40	42	44	45	47	48
40	40	42	44	46	47	49	50

display 1

14	14	16	18	21	22	23	24
16	16	18	20	23	24	25	26
19	19	21	23	26	27	28	29
24	24	26	28	31	32	33	34
30	30	32	34	37	38	39	40
33	33	35	37	40	41	42	43
37	37	39	41	44	45	46	47
39	39	41	43	46	47	48	49

display 2

16	14	15	17	22	22	23	23
19	17	18	20	25	25	26	26
23	21	22	24	29	29	30	30
27	25	26	28	33	33	34	34
32	30	31	33	38	38	39	39
35	33	34	36	41	41	42	42
39	37	38	40	45	45	46	46
42	40	41	43	48	48	49	49

display 3

16	14	15	17	22	22	23	23
17	15	16	18	23	23	24	24
23	21	22	24	29	29	30	30
26	24	25	27	32	32	33	33
31	29	30	32	37	37	38	38
35	33	34	36	41	41	42	42
38	36	37	39	44	44	45	45
42	40	41	43	48	48	49	49

display 4

16	15	15	16	23	24	23	23
16	15	15	16	23	24	23	23
24	23	23	24	31	32	31	31
26	25	25	26	33	34	33	33
31	30	30	31	38	39	38	38
36	35	35	36	43	44	43	43
40	39	39	40	47	48	47	47
42	41	41	42	49	50	49	49

display 5

16	15	15	16	23	23	23	23
16	15	15	16	23	23	23	23
23	22	22	23	30	30	30	30
27	26	26	27	34	34	34	34
31	30	30	31	38	38	38	38
35	34	34	35	42	42	42	42
40	39	39	40	47	47	47	47
42	41	41	42	49	49	49	49

display 6

16	16	16	16	24	24	24	24
16	16	16	16	24	24	24	24
24	24	24	24	32	32	32	32
24	24	24	24	32	32	32	32
32	32	32	32	40	40	40	40
32	32	32	32	40	40	40	40
40	40	40	40	48	48	48	48
40	40	40	40	48	48	48	48

display 7

Figure 12.1 A 2-D area CMT example.

The 2-D transform has been used to produce the sequence of images shown in photograph 12.1 using the same number of coefficients at each iteration as in the numerical example. The degree of visible distortion is increased over the 1-D transform, as an error within any 1-D transform in the first pass can corrupt the entire 2-D block in which it operates. The non-distorted areas however show the expected behaviour with some smooth block filling evident, and use of an improved MT in a 2-D separable manner should be possible with useful results.

12.2. Bitwise Partition of the CMT Coefficients for Transmission

A potential enhancement of the MT involves the segmentation in a bit-wise fashion of individual coefficients for transmission at different stages in a progressive sequence. The high order bits would be transmitted first as an approximation to the coefficient, to be followed later in the update sequence by the low order bits to give the coefficient its original full accuracy. It is not difficult to perform the segmentation and recombination with the MT, though it adds to the overall complexity and calculation time required there are no difficult problems to overcome. The process does not seem to have been attempted with a conventional transform, though there is probably less motivation in that quality can usually be improved by increased quantiser sophistication which is not possible with the MT. A potential problem in attempting to split a coefficient across several updates with a conventional transform is the linearity of the quantiser used. If the quantiser is non-linear then as the low order bits are received they cannot simply be inverse transformed and added to the display values. It would only be possible to integrate the new data with the display by using the forward transform to find the original coefficients and updating these with the additional data bits. This is not practical as the repeated transformation using a conventional transform would cause increasing error. The inability to simply add new data to the display with the MT is inherent as is the need to re-transform to find the transmitted coefficients. The only extra operation required to accommodate the additional data is to 'or' the newly received bits into the coefficient.

The operation is not quite as simple as it might be, due to the nature of the range of coefficient values, their representation as twos-complement numbers and how they are affected by being bitwise segmented. If the twos-complement numbers used are ordered by their bit pattern representation, the sequence is

0,1,2,...31,−32,−31,...,−1

Simply performing bitwise 'and' of the twos-complement number with some bit-mask has undesirable results for negative numbers. As an example, if the top two bits only are masked in, the value −1 represented by 63, when masked gives 48 which represents −16. Thus the truncation which is causing rounding towards zero for the bit pattern, causes rounding away from zero for the negative numbers when rounding towards zero would seem desirable. The solution to this problem is to convert negative numbers to sign and magnitude format, mask only the magnitude and then restore the sign. This process is used during both the coding and decoding phases as the extra bits are inserted. There is inevitably in this method one bit used to represent the sign of the coefficient, and whereas this is correct it means at least two bits are required in any coefficient for it to be meaningful. The coefficients are effectively being quantised by this truncation with a set of well defined decision levels and well defined but less correct reconstruction levels. By this method the reconstruction levels are identical to the decision levels, whereas ideally they should be placed in-between. Assuming an even probability distribution of the coefficients for simplicity, the reconstruction levels should at the half way points between the decision levels. Separating the decision and reconstruction levels has not been attempted but there does not seem to be any major obstacle to the attempt, this may even make one bit coefficients meaningful. Additional sophistication of this form will of course increase the amount of computation required. With simple bit operations the bit quantities shown in figure 12.2 were used to produce the images shown in photograph 12.2. The number of reasonable permutations for this operation is very great and only the first three coefficients have been used with three different bit totals. It is important to remember that these images are not being assessed by their obvious flaws but to gauge the potential for an improved MT that will not exhibit these flaws. The following comments are made on the understanding that these flaws in the images are being ignored. The first three images (marked 1,2,3 in figure 12.2) all have coefficient data summing to 6 bits. The first image has 6-bit resolution in the first coefficient only, with all the other coefficients set to zero. In image 2, moving two of the bits into the second coefficient gives perhaps a slight improvement as the image becomes smoother. At the third image however the reduction in resolution of the first coefficient has increased the contouring to an unacceptable degree with the result that quality is reduced. Images 4 and 5 each have 9 bits in total and again it seems advantageous to move some resolution away from the first coefficient into the second. This is confirmed by the third set of images with 12 bits, in this case the number of possible permutations is far greater than those shown, but the trend can be gauged. The final image, 8, shows that a certain amount of contouring is tolerable if the bits moved from the first coefficient can be used to reduce the visibility of the block edges by permitting greater variation of the other coefficients.

coeff. 0	6 bits	coeff. 0	4 bits	coeff. 0	3 bits	coeff. 0	6 bits
coeff. 1	0 bits	coeff. 1	2 bits	coeff. 1	3 bits	coeff. 1	3 bits
coeff. 2	0 bits	coeff. 2	0 bits	coeff. 2	0 bits	coeff. 2	0 bits
total	6 bits	total	6 bits	total	6 bits	total	9 bits
1		2		3		4	
coeff. 0	5 bits	coeff. 0	6 bits	coeff. 0	6 bits	coeff. 0	4 bits
coeff. 1	4 bits	coeff. 1	6 bits	coeff. 1	3 bits	coeff. 1	4 bits
coeff. 2	0 bits	coeff. 2	0 bits	coeff. 2	3 bits	coeff. 2	4 bits
total	9 bits	total	12 bits	total	12 bits	total	12 bits
5		6		7		8	

Figure 12.2 Bit allocations for the bit partitioned CMT coefficients.

12.3. Interpolation of the Displayed Image

The second potential enhancement to the CMT is interpolation, this proved useful with the hierarchical methods discussed earlier, and if possible would address a fundamental problem associated with transform coding of highly visible block boundaries. This problem has previously been approached in different ways, sometimes by using a very different transform that takes account of the end point values and superimposes transform information on-top of a basic linear model. Using PTD the block boundary will eventually disappear and the problem can be viewed as that of predicting the values of some of the coefficients that will be transmitted. These predicted values can be appended to the coefficients already transmitted before inverse transformation to find the display value. This is possible with the MT as after re-coding the display data to recover the coefficients there is a clearly defined distinction between those coefficients that have been transmitted and those that have been appended for interpolation. Certainly with the MT these two sets of coefficients will not mutually degrade as decoding iterates. The problem is to choose the coefficients to be appended in order to minimise the boundary effect. A very simple and pragmatic approach was taken to the solution of this problem using the CMT based on 8×1 blocks. Again a linear ramp model was used and interpolation only performed in one dimension. The first assumption made was that for a linear model, the mean value, available as the first coefficient, occurs near the middle of the block. Assuming the end-point value of the preceding block (to the left in scanning order) is known, a pixel to pixel difference can be found by dividing the known mean and end-point values by the number of pixel intervals which in this case is 4. This difference can be used to generate all 8 points within the interpolated block using as a starting value, either the block mean assumed mid-way, or the known end-point of the preceding block. Ideally these values could then be displayed as the first approximation, being constructed to have the correct mean value. However constructing a block with differences in this way, cannot, due to rounding, be guaranteed to preserve the mean value. A more sophisticated approach using the symmetry involved might keep the mean correct, however this approach does not combine satisfactorily with the overflow clamping required that is discussed below. Instead, it is necessary to forward transform the derived data, implant the original mean and inverse transform the entire coefficient set to find the first display approximation. At the next iteration the displayed value must be forward transformed to find all the coefficients as not only the first is valid but the remainder represent the calculated linear ramp values. As the second transmitted coefficient is received it overwrites the second interpolation coefficient and the set is inverse transformed to find the second display value. This process causes an increase in the computation required for coding and decoding, previously the number of iterations within either of these phases could be matched to the number of coefficients available. With recovery of all coefficients required at each stage, this is not possible, causing a doubling of the total amount of computation required. This may not be a problem depending on the way in which the available processing power is matched to the decoding process. Without interpolation it is necessary for production of the final display to transform the displayed data to produce all but one coefficient, insert the final coefficient and inverse transform the entire set. A hardware engine that can do this will have idle time during early display iterations and the extra computation for interpolation will merely keep the engine fully active, not requiring any extra performance.

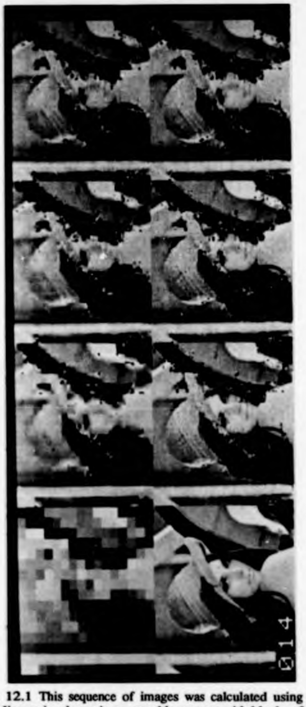
As pragmatic as is this approach, there are equally pragmatic, small but none-the-less tedious problems associated with it. The first problem apparent in early tests is the stability of the interpolation sequence, as each block depends on its predecessor a cumulative effect is possible. Additionally, as more received coefficients are available the result depends on the interaction between the two-types of coefficient. The early results were, in luminance profile, somewhat saw-tooth shaped and as a result the method was changed slightly. The pixel differences were calculated not using the end point value but instead the mean of the previous block. The mean of the previous block is available as its first coefficient and the new pixel difference is found by dividing the difference between the two mean values by 8. This makes worse the closeness of the match at the edge of the block but does stabilise the overall sequence. A further detail problem is associated with clamping the interpolated values, the simple interpolation algorithm used can produce values outside the display range. With the CMT these values must be clamped at the display limit before coding. This itself does not cause problems, but if the inserted mean is slightly greater than the calculated mean, for a block that has clamped values the reconstruction goes into error even though the original data projects without problem.

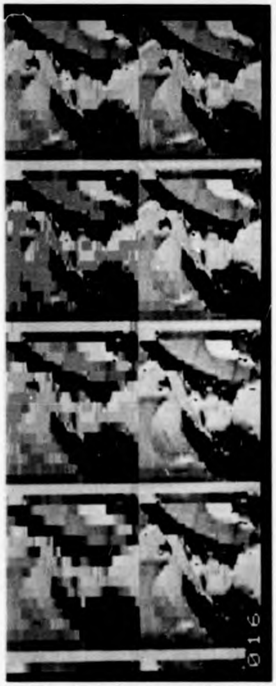
A more general issue relevant to the CMT is the degree of interpolation provided by the transform itself. Although not linear, the basis vector B_1 as shown in figure 11.2, will produce a similar effect to the linear interpolation and it seems likely that the two will interact. It may be that simple linear interpolation has no effect after the coefficient for B_1 has been transmitted as this would have had the greatest contribution to the linear interpolation.

All these problems aside, the results of using the above described interpolation are shown in photograph 12.3. For this sequence of images the final modulus operation of the decoding sequence has been omitted and the resulting values clamped at the display limits, in order to minimise the visible effects of wrap-around. This is done in part because of the increased occurrence of the errors caused by the mean insertion in the clamped blocks.

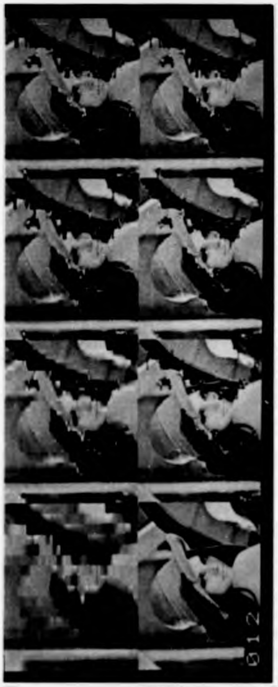
The results shown in photograph 12.3 do not exhibit the desired interpolative effect to any useful degree, and despite clamping the final iteration of the decoding process a large number of errors are still visible. From consideration of the algorithm used, the first image at least should show interpolation and by comparison with the first image of photograph 11.1 it is possible to see the variation in pixel values across some of the blocks. However the interpolation algorithm described does not guarantee edge-to-edge matching of the blocks and clearly some greater effort is required to achieve the desired effect. The algorithm will also have reduced effect for iterations after the first, as the interpolation does not take into account changes occurring as the interpolation coefficients are overwritten by received data. These two properties most probably account for the poor performance. It is not clear how to immediately proceed with improving the interpolation algorithm. It seems the first step in an interpolation algorithm is to find the difference between the display values produced from the transmitted coefficients and the desired interpolative values. This presents the question of what the desired values are, given that all the pixels in the block have display values. If linearly related values were required then the difference could be found between between these values and the decoded values. This difference needs to be represented as far as possible in terms of the coefficients not yet transmitted. It is not clear how well this can be achieved with the MT as the MT relies heavily on the ordering of the basis vectors and the representation needs to be in terms of the vectors at the end of the conventional transmission sequence. Some fundamental geometry is required in order both to better understand the approximation mechanism and to derive some technique that is possible in conjunction with a MT.

A different approach, motivated by the performance of the sine transform in reducing block boundary visibility as reviewed in section 6.1 may be possible. The method functions by making the pixel values at the ends of the blocks take near zero values, and from the shape of the second cosine basis vector as shown in figure 11.2 this may be possible with the CMT. A further property of the basis vectors that may be of use is the single slope of the first basis vector as mentioned earlier in this section. It may be possible to calculate a value for this based on the available mean values and hold transmission of the coefficient until later in the sequence. There are altogether many possible avenues of investigation for increasing the subjective quality of these images specifically by reducing the visibility of the block boundaries. The ability to manipulate the received coefficients without affecting the fidelity of the final image is unique to the MT and being so novel cannot be expected to be used immediately to best effect.


Photograph 12.1 This sequence of images was calculated using the CMT over a two-dimensional area in a separable manner with blocks of 8×8 pixels. The number of coefficients used for each image follows the sequence 1,4,9,16 etc., starting with the top-left image and proceeding clockwise. The final image is identical to the source image data.



Photograph 12.2 These images have been calculated using the CMT over blocks of 8 × 1 pixels with some coefficients having less than the full 6-bit resolution. Only up to the first three coefficients are used with up to a total of 12 bits of data. The full bit allocation is given in the text.



Photograph 12.3 The CMT based sequence shown here uses horizontal blocks of 8 × 1 pixels. The display data of the first image is interpolated and transformed to find a set of coefficients that are used instead of zero for coefficients not yet transmitted at any stage. The final image is identical to the source data.

13. A Table Based Implementation of the Cosine Mimic Transform

In the earlier sections concerned with hierarchical PTD techniques, implementation issues related to the table equivalent transforms and the interpolation algorithms were considered important. The precise implementation of these algorithms affects the possible throughput and hence the usability of the techniques. Little has been said of the implementation of the MT or CMT other than their increased computational requirement over conventional linear transforms. Obviously the coefficients of the cosine transform have non-integer values and these are used in the CMT, some of the examples have at least quoted mean square error figures with several places of precision. The reader may have concluded quite rightly that floating-point arithmetic has been used extensively in the implementation of the CMT.

A production implementation of any long precision algorithm can often use fixed-point arithmetic rather than floating-point but this is excessively tedious to implement for an exploratory investigation. More useful, less laborious and quite common, is an investigation of the transform to see what parts can be implemented by table look-up. Table look-up will always give a speed increase over floating-point arithmetic and probably be faster than fixed-point arithmetic depending on the availability of a multiplication instruction and the precise operation required.

13.1. Table Based Projection

Part of the MT and consequently the CMT is very easy to implement by table look-up and this was done at an early stage in the investigation. This is part of the projection mechanism, an operation of which looks like

$$p_{n+1,i} = \left[p_{n,i} - \left\lfloor \frac{b_{n,i}}{b_{n,j}} p_{n,j} \right\rfloor \right] \quad \text{mod} \; -N/2, N/2-1 \qquad i \text{ over remaining axes}$$

Of this the part of the expression inside the floor operator is of interest. The potential for table look-up depends on the number of variables contained in the expression to be evaluated by table, and their range of values. The index j is fixed for each iteration indexed n and for a transform over 8 elements there are 8 iterations. There are also 8 different values of i for each iteration and so the fraction within the floor can take 8 different values per iteration giving 64 values for the fraction in total. The value $p_{n,j}$ within the floor operator has the same range as the data, which in the authors implementation is 64. Thus the total number of combinations is 64×64, giving 4096 values which is a manageable size. If the data range were increased to 256, a common value of 8 bits, the table size would increase to 16k entries which is still manageable. Another important issue is the size of the entries within the the table and their format. The expression is very easy to tabulate in this respect, because the floor operator guarantees the result to be an integer value and the modulus operator gives it clearly defined limits. Using twos-complement arithmetic, the modulus with limits $-N/2, N/2-1$ and N a power of two is very easy to find. It is only necessary to take the appropriate number of low-order bits to find the modulus. This can be done conveniently with a bit-wise 'and' operation with the value $N-1$. Using twos-complement representation a range of values N equal to 64 or 256 will fit into an 8-bit byte. Consequently the table entries are bytes and with a range of 64 values for the coefficient in the expression the table is of size 4k bytes.

Using the table gives the projection operation a very clean and simple formulation. When the projection operation is implemented there is no need to keep copies of $p_{n,i}$ for each iteration n, and only one set is kept which is modified from iteration to iteration. Working within a basic frame-work of the C programming language the resulting program segment might be

```
rel_val = p[ rel_axis_index ] ;

for( i = 0; i <= range - 1; i = i + 1 )
      p[ i ] = ( p[ i ] - projection_table[iteration][rel_val][i] )
                                    & ( range - 1 ) ;
```

This in itself is not adequate as the simple loop mechanism cannot control the decreasing set size with iteration. Somewhere this decreasing set size must be built into the algorithm for it to operate correctly. Spurious values in the co-ordinate (p) values are not important if they are not counted in the evaluation of the correction factor, and set size could be built in at the correction factor evaluation. The co-ordinate values could be forced to zero if the projection table were zeroed out for appropriate iterations and indexes allowing the correction factor to be taken across all co-ordinates. Both these solutions require more operations per iteration than are strictly necessary, and some improvement is possible in the loop control. With an example based on the second iteration of the CMT after axis 4 has been deleted, the approach adopted by the author is very similar to

```
int set[1][9] = { 0, 1, 2, 3, 5, 6, 7, -1 } ;

rel_val = p[ rel_axis_index ] ;

for( ptr = &set[1][0]; *ptr >= 0; ptr++ )
      p[ *ptr ] = ( p[ *ptr ] - projection_table[1][rel_val][*ptr] )
                                          & ( range - 1 ) ;
```

This controls the loop more correctly at the expense of some extra indirection with cost dependent on the ability of the compiler. An optimum implementation would of course need to be adjusted for the precise instruction set or language available but it is hoped that this has shown that the projection mechanism can be implemented quite elegantly. The inverse of projection, referred to by the author as the build operation is almost identical to projection, the difference being an addition rather than subtraction at the heart of the operation. An issue of detail is that it should be possible, if necessary, to improve the naive layout of the projection table to reduce its size as the number of entries strictly required at each iteration is reduced. This will increase the computation required to perform the look-up operation and is a low-level trade-off.

13.2. A Table Based Correction Factor

Table look-up for projection was implemented at an early stage in the investigation, but the other candidate for re-implementation, the correction factor, was calculated in a very naive way using floating-point for all the examples and images shown in this thesis. It was however always desirable that this should be implemented more efficiently, and considered important to show that this part of the algorithm could be better performed.

The correction factor with j fixed for any n is generally

$$cf = \left\lfloor {}^{n}b_{n,j} \sum {}^{0}b_{n,j}\, p_{n,i} \right\rfloor \qquad i \text{ ranging over remaining axes}$$

The first thing to note is that the correction factor requires the ${}^{0}b$ set of coefficients rather than the ${}^{n}b_{n}$ coefficients used for projection. Without taking advantage of the decreasing range of i dealt with above, there are 64 ${}^{0}b_{n}$ coefficients and if data takes the range 64 this gives a table size of 4k entries for the product inside the summation. As the product is summed it must be stored to a greater than integer precision, and adequate precision is obtained by using two bytes, giving a table size of 8k bytes. If 8-bit data were used the table size would increase to 32k bytes which, with the projection table, is large for the 64k byte address space of a 16-bit processor. Considering this, some way was sought of reducing the space requirement of this new table. The Discrete Cosine Transform basis vectors ^0b_n exhibit a high degree of symmetry, and disregarding the sign, are comprised of only 7 different valued coefficients. It is possible to map from an index into the full coefficient array to one of these 7 values using only an 8×8 array containing the values +/-(1 to 7). The magnitude of an entry of this table can then be used to index into a table containing the products of the coefficients with the data range, and the sign can be used to modify the looked-up product. The 7 coefficients fit, more easily into a table of size 8, and with 64 data values, the multiplication table requires only 512 entries. There seems no reason why the $^nb_{n,j}$ coefficient should not be moved inside the summation and hence into the table to remove the problem of the multiplication after the summation. However for consistency and clarity, the implementation follows the above definition and performs the summation first. The longest summation is performed over 8 elements and for the summation to have a correct integer result, precision to 1/8 is required for each summed value needing three fractional bits. The result is modified by the factor $^nb_{n,j}$ which can be greater than one, potentially requiring more precision from the summation, but the greatest factor is less than two so no further precision is required. Thus the table containing the products of the coefficient values $^0b_{n,j}$ and the data value $p_{n,j}$ must have three bits of precision for the summation to be accurate. As it is impractical to store other than byte aligned data, the 512 entry table has a byte integer and a byte fractional part making the table size 1k bytes. To summarise the first part of the calculation, the product within the summation is found using n, i to index an 8×8 table to find which of the 7 different coefficient values is to be used together with its sign. This identifier is then used with $p_{n,j}$ to index a 512 element table containing the 16-bit product. No special action is required for the summation, the 16-bit table entries are summed and the result remains with one byte integer and one byte fraction.

A very simplistic approach has been taken to the multiplication of the result of the summation with the factor $^nb_{n,j}$. This requires yet another table holding 16-bit products between each $^nb_{n,j}$ factor of which there are 8, and numbers in the range 0 to 256. This table is of size 8×256 entries, each of two bytes giving 4k bytes in total. These entries are used in a form of long multiplication with each byte of the sum taken in turn. First the integer part of the sum is 'looked up' with the appropriate factor and the result shifted left by one byte (multiplied by 256). Then the fraction part is 'looked up' with the same factor index and the table entry added to the previously shifted result. This gives the integer result and an accurate 8-bit fractional part.

This method is less than optimal for several reasons. One reason is that the full 8-bit fraction result of the sum is not required and the size of the final table could be reduced. Another reason is that the integer part of the "$b_{n,j}$ factor is only ever zero or one, so the first part of the long multiplication is really redundant and could be replaced by a simple test and load. Throughout these operations, a record needs to be kept of the signs of various stored values as the operation uses a mixture of sign/magnitude and twos-complement arithmetic. For data with a range of 64 values, some of the tables are too large but otherwise much shifting of values is required.This unnecessarily complicates the algorithm for the resulting saving in space, unless there is an extreme restriction on this. The total amount of table space required to calculate the correction factor for 6-bit data is 64 + 512 + 4k bytes, with and additional 4k bytes used for projection.

It is likely that the evaluation of the correction factor could be performed more elegantly, but it has been shown to be possible without recourse to floating-point arithmetic and only a few fixed point operations. Even integer multiplication is not directly required though it is often tacitly assumed to be available for array index calculations. The expected range of the correction factor has not been discussed, partly because it is not fully understood. Certainly it cannot be greater than the range of the data and a result of this range is allowed for by the above described algorithm. However the MT can be implemented in an efficient form as it has a simple iterative form with most of the arithmetic being possible using table look-up.

14. Conclusion

In this section the results of the different lines of enquiry pursued in this thesis are summarised and the potential areas for further work discussed.

Progressive transmission and display can only be used, and indeed is only useful in very specific circumstances. As the cost of the equipment required to perform image display falls and as distributed computing environments based on the work-station concept become more common the potential applications area for PTD becomes larger. The two most important factors in determining the feasibility of using PTD in any single application are the quantity of data that makes up the image and the data transmission rates available. Several different combinations of these two parameters were considered in section 1 with some comment on the way in which the data rate affects the processing rate required to maintain throughput. One conclusion stated is that for conventional single processor based systems it is impossible to update large parts of the display in the time taken to receive one data value at even low data rate transmission. Some aspects of the display hardware were discussed with reference to the way in which the architecture might affect the choice of method of access of the display memory and hence the overall PTD algorithm. PTD requires much more processor access of the display memory than more conventional line-by-line update methods and the complete display system must be configured to optimise these operations. As the potential applications area for PTD changes with increasing emphasis on processing of the received image, as described by Hill [1983], the fidelity requirements of the PTD methods used must also change. Subjective measures of image quality are often used in the design of PTD techniques, but when post-processing for numerical results is to be performed, the final display data should be numerically equal to the source data.

The new algorithms described in this thesis address only the problems of exactly invertible PTD techniques. An additional constraint placed on the algorithms is that the intermediate representation maintained in the display system, must be either directly displayable or displayable with only a small amount of processing that can be performed for display in real time. An example of the processing that may be performed is a table look-up between the read-out of the intermediate representation and the conversion to analogue signal levels. If this constraint on the intermediate representation cannot be met then the intermediate representation must be kept separately and in addition to display memory with consequent increase in hardware and processing cost as discussed in sections 1.2 and 1.4. A constraint was also placed on the the total amount of data transmitted for a PTD sequence. In order to maximise the acceptability of a PTD technique it should not involve transmission of more data than would be needed to ordinarily transmit the image, in a non-progressive fashion.

The methods reviewed first in this thesis were the well known quad-tree methods described by Sloan and Tanimoto [1979]. These techniques were exactly invertible, more as a result of their simplicity than as a specific design aim, though techniques that were not exactly invertible were mentioned by Sloan and Tanimoto, not explicitly but in reference to transform coding. The intermediate representations of the different quad-tree based methods were suitable for display either directly or with simple hardware manipulation. This was primarily because the display system was anticipated to have very little processing ability. The problem associated with these methods is the transmission of redundant information. Using one method, a new image completely overwrites all previously transmitted information. An improvement on this technique was based on a mean/difference transform, which became a recurring theme throughout this thesis. The simple approach used required extra precision to represent the mean, and this caused problems with the intermediate representation. However the main objection to its use by the criteria adopted in this thesis is again that it requires the transmission of essentially redundant information as the extra precision of the mean value. The mean/difference transform was improved upon by Knowlton [1980], who produced a two element transform that gave an approximate mean value that did not require extra precision. This technique meets all the criteria laid down for techniques to be developed in this thesis, only its definition in terms of a hand-crafted table limits its application. In some cases the table required for operation of the method is too large for reasonable implementation and some algorithm was sought to replace the table look-up operation. An algorithm for this purpose was described in section 3.1, the algorithm is a complete solution to the problem and is itself so simple that it is unlikely to be improved upon. Relating also to Knowlton's work, a more realistic approach was taken to the resolution of the first image displayed in a progressive sequence. Knowlton originally specified that the entire display be updated by first and then second data values to be received. The fraction of the display rewritten per data value received, although it decreases exponentially with iteration, remains relatively large in terms of an assumed processing capability, for several iterations. The solution offered to this problem is to commence true progressive display with an image that for the experimental equipment was made up of composite pixels of size 8×8 real pixels. With the algorithm to replace table look-up and the change to the starting resolution, Knowlton's method can be used in a wide range of PTD applications.

Although not a definite obstacle to the operation of Knowlton's method, the intermediate images have a strong block structure. In order to make this aspect of the method more attractive, an attempt was made to improve the subjective quality of the intermediate images. Knowlton's method can be combined with a simple interpolation technique such as bilinear interpolation as described in section 4.1. Photographic results of this combination were given at the end of section 4. Some attempt was made to produce an interpolation technique that did not require the data management overhead of bilinear interpolation and the result was the four way mean algorithm described in section 4.2. While being algorithmically more simple than bilinear interpolation and thus more suitable for implementation in VLSI or micro-sequenced hardware, the four way mean algorithm has questionable subjective performance. However its derivation has also indicated that the bilinear interpolation mechanism might be reformulated to produce an algorithm more similar to the four way mean algorithm. Undoubtedly the interpolation techniques do improve Knowlton's method, at the expense of some increase in computation, and should be considered whenever Knowlton's method is used.

With many PTD techniques it is possible to perform some focussing of the update sequence, preferentially updating certain parts of the image corresponding to areas of main interest. An update sequence might focus always on the centre of the image, at a point specified at coding time, or at some point specified by the viewer at decoding time. Focussed update causes an interpolation scheme to become more complicated, different areas of the image are updated to different levels, and therefore must be interpolated to different degrees. The bilinear interpolation scheme described in section 4.1 relies to some extent on the scanning order of the blocks, which is very different for focussed update. Nonetheless a focussed update sequence was shown in section 4.3 based on the four way mean algorithm, demonstrating that focussed update has potential for use and further development, especially in conjunction with interpolation.

Knowlton, and Sloan and Tanimoto commented on the possibility of reducing redundancy within the tree representations of images by marking monotone sub-trees and thus not transmitting a series of identical values. With Knowlton's method the trees are binary-trees and the results of an investigation into the size of these monotone sub-trees was reported in section 2.3. For the image investigated, the potential for reducing redundancy by coding sub-trees did not seem to justify the complexity required. The potential for coding internal monotone trees which should not be as affected by noise and quantiser resolution as trees which contain bottom level leaf differentiators was also investigated. Again the results did not indicate a worthwhile saving for this particular image. An investigation of the entropy of the differentiators yielded more encouraging results, as suggested by Knowlton. The values produced are approximately equivalent to the second order entropy of the image data and also to the entropy of the pixel-to-pixel difference values. Variable length coding of differentiators requires a smaller code-book than coding based on either of the other two measures, and if in any application variable length coding is possible, then a useful increase in performance may be obtained.

In its more developed form, Knowlton's method is clearly useful for PTD although in some ways the approach is not completely satisfactory. As the method is based on a binary-tree decomposition, the time taken to transmit each new update for the entire image, increases by a factor of two with each update. This is difficult to reconcile with a notion of constant increase in picture detail, though the point may be argued, and a constant update period might be preferable. With a simple mean/difference transform and a scanning order dictated by the binary-tree approach, the only pattern within the image that can be detected usefully for coding is the monotone area. Some better method of decomposing the image is required, into more components to give greater flexibility of update. Transform coding has this ability and good results are achieved in its use for image coding. Although Knowlton's method is based on a binary-tree, a similar exactly invertible method is not known for quad-tree coding. A simple transform technique over four points would enable the investigation of quad-tree methods. Transform techniques have a coefficient by coefficient update method which gives a steady perceived increase in picture quality and the intermediate representation using transform coding is suitable for direct display. However transform techniques, as previously published, do not meet the criteria laid down in this thesis in that they are not usually exactly invertible, and extra resolution is required to maintain the intermediate representation. Some comments were presented in section 5 and section 6 regarding the actual range of values required in the intermediate representation, for some well known transforms in order that the representation should not be subject to overflow. There was also some discussion of the precision required for the transform coefficients in order to retain the property of exact invertibility. Conventional transform coding used in such a way as to be exactly invertible would require transmission of coefficients with precision that is ultimately redundant, and this is not acceptable. A further problem is that the intermediate representation also needs to be retained to greater precision than that of the source image data, as a separate factor to the increase in range. Some method of adapting transform coding to obviate these problems while retaining the desirable features of update period and flexibility of update was sought.

The algorithm produced as the result of this adaptation is known as the mimic transform in that its intermediate images are similar to those of the linear transforms that it mimics. The Mimic Transform should be able to mimic any linear transform, the basis vectors of which are required for development of that particular Mimic Transform. The development of the Mimic Transform commenced with the two point case in order to make the first steps in the development as simple as possible. The first principle of the MT as developed was the measurement of distance along a basis vector, effectively a transform coefficient, by a corresponding distance along one of the axes. This is possible because the criterion of mimicry is the intermediate image in which only integer values are possible and the rounding normally required for display is simply incorporated in the coding phase. The second principle developed in the two point case is that of projection. This is used to reduce the dimensionality of the problem, much as a linear transform can be considered as a set of projections into successively fewer dimensions. In two dimensions, for a two point transform, only one projection is required to find the second co-ordinate. The third principle in the two point Mimic Transform is the calculation of the correction factor, this is used to give a consistent approximation to a transform coefficient for a set of points from which one can be selected using one of the new transform coefficients. The fourth and final principle is the use of modulo arithmetic to ensure the Mimic Transform is invertible. As part of the investigation, a mean/difference transform was developed that required very little computation. It might be implemented in very few machine instructions in which case the resulting program would be deceptively simple.

The Mimic Transform was then modified for use over three points, thus requiring an extra iteration over the two point case to completely transform the data. The use of an axis (the related axis) to measure the distance along a basis vector is fundamental to the Mimic Transform and did not need to be changed for operation over three points. The use of the projection operation was less straightforward; in the two geometrical dimensions of the two point transform the projection was only to an axis rather than a basis vector, which was then sufficient. In the three dimensions of a three point transform the first projection must be onto a plane defined by two basis vectors whereas the original mechanism would project onto the plane defined by two axes. This apparently requires a further operation to find a corresponding point on the correct plane, but this is very difficult to achieve without forcing the co-ordinates to take fractional values and possibly defeating the goal of exact invertibility. The novel approach finally adopted involved not modifying the co-ordinate values, but considering them instead to be on a new set of axes in the correct plane. This requires some manipulation of the matrices defining the basis vectors, and their inverses, which are the matrices defining the axes in terms of the basis vectors. It was shown that the required manipulation is always possible when the first basis matrix is non-singular, as is always true of a linear transform basis matrix. This manipulation resulted in the axes being not of unit length if the basis vectors are considered unit length, and the axes are no longer orthogonal. The calculation of the correction factor at the first iteration of the three point transform is more complicated than the two point case, as not all the relevant components of the calculation are co-planar. However, the calculation can be resolved by working in two different planes with a common line, making the calculation possible. At the second iteration of the three point transform, all the components are co-planar but the axes are no longer orthogonal and again the derivation of the correction factor is slightly different to the two point case. Despite these changes the correction factor is not difficult to derive and the resulting expression is not difficult to evaluate. The transform coefficients throughout are measured along the related axes although the simple method of choosing the related axis by the angle between the axis vector and the basis vector was found at this stage to be not optimal. Further investigation clarified the point to give a clear choice of the best axis to choose for the related axis. The modulo arithmetic used is also fundamental to the operation of the transform and its use was continued from the two point to three point transforms in a consistent fashion.

A Mimic Transform over three points is only marginally, if at all more useful than a two point Mimic Transform, and the concept was extended for maximum flexibility by operation over any number of points. This required a check on the operation of the correction factor mechanism which needed only a simple generalisation from the three point case. The projection mechanism required no modification to operate satisfactorily within a transform over a greater number of points. Two other facets of the Mimic Transform were also finalised at this stage. The modulus limits were discussed including the way in which they affect the correctly approximated area, and the rounding method for the correction factor clarified. In order to check that the Mimic Transform had been devised correctly, in terms of its relationship with the underlying linear transform, two iterations of the MT were expanded back through the projection mechanism to check the exact formulation of the coefficients. This is otherwise not immediately clear as the coefficients are normally expressed in terms of projected values. This expansion shows that without the floor operation, and also without the modulus limits, the MT has the form of the underlying linear transform. Of course the floor operation and the modulus limits do make the MT behave differently to the linear transform, but the floor operation causes only slight differences of the form of rounding or approximation. It is the modulus operation that gives rise to the main problem with the MT, that some parts of the data space are not well approximated, and the point needs further attention.

Having developed the MT to operate over any number of points, it was consider useful to look briefly at the outstanding problems that had been noted early in the development of the MT. As the projection mechanism is not geometrically precise, the co-ordinate values can degrade cumulatively with respect to the geometrical ideal represented by the linear transform. The only way to improve performance in this respect is to retain the projected values to greater precision. As only a fixed number of projection values are possible, increasing the precision requires a reduction in the range of possible values, and this may not be desirable. However, projection with increased precision was shown to be possible, although under certain circumstances a very general implementation causes some problems by its susceptibility to the decoding error caused by truncated coefficients sequences.

Only tentative proposals were made for changes to minimise the problem associated with the modulus limits, and it was considered important to gain experience of using the MT in a real coding situation before undertaking further investigation.

There were two options for the main direction to take in evaluating the MT in its first realistic coding application. One possibility was an approach using a hierarchical tree based technique, as the requirement for an exactly invertible transform suitable for quad-tree coding was part of the motivation for the design of the MT. An eight point transform for oct-tree coding as used in tomographic applications might also have been suitable for a similar approach. These transforms would be formulated with simple sets of differences and would probably not exhibit the desired flexibility of update required while presenting an over simple set of basis vectors for use with the MT. The alternative approach was to base the MT on a linear transform that is commonly used for image coding, which might give more flexible update and also permit comparison of the MT with that equivalent linear transform. It was considered important to test the MT as much as possible by the coding example, and so the conventional linear transform approach was chosen and furthermore the Discrete Cosine Transform was chosen as being often used in image coding. The union of the MT and the cosine transform produced the Cosine Mimic Transform (CMT), with no unexpected complications in its development. In simple numerical tests, using data designed not to present the CMT with difficulty, the CMT performed well with mean square error values for the intermediate values that are similar to those of the DCT used for comparison. By using data designed to cause break-down of the approximation, it is possible to force the CMT into operation where the intermediate do not have the desired properties, though the final result is always equal to the input data. When used with image data, the results of which are reproduced photographically at the end of section 11, it can be seen that the CMT does break-down with some image data. Much of the image approximates the behaviour of the DCT, for which equivalent image based results were also presented. The processing of the image data by the CMT was investigated to find out if any particular part of the transform was susceptible to break-down. One aspect of conventional transform coding that was not built into the MT is the behaviour of a linear transform when decoding a truncated coefficient sequence, as the values produced exceed the range of the data. Some of the errors in the image produced using the CMT are caused by the underlying transform having undesirable behaviour in this respect. The projection values at coding time were investigated and although the modulus limits were exceeded, the co-ordinate values were not far outside the limits as indicated by the 'miss' values reproduced in section 11. The results of the investigation of the general operation of the CMT did not give a clear indication of the behaviour of the MT itself with truncated coefficient sequences and this might be further investigated. The MT should only require small changes to permit storage of values outside the range of the data, as is required for storage of the intermediate values of even a conventional linear transform. Although this violates the constraint of no extra storage for the intermediate display values, it is clear from the behaviour of the MT that it is impossible to meet this constraint without extreme visible degradation, when using transform coding. Although the range of values must be increased, the precision remains unchanged and it should be possible to use a look-up table to convert intermediate values to display values, with only a simple table structure equivalent to clamping the out-of-range values at the display limits.

Even without making any improvements to the MT, it was considered worthwhile investigating some further possibilities for its use while acknowledging that they might be difficult to assess with the existing flaws. An obvious possibility for investigation was a 2-D transform, using a separable transform and block-size the same as the 1-D case for simplicity. A simple numerical example indicated that the overall approach was operationally sound, and the transform was applied to the GIRL image. The problems associated with the 1-D transform are again obvious in the results of the 2-D transform, but over some of the image the transform can be seen to have behaved satisfactorily. This shows that if the problem with the modulus limits can be remedied, most of the errors shown by the 2-D transform will no longer be produced and the technique will be fully viable.

A technique thought to be completely new as applied to transform coding was subsequently investigated. This involved segmenting a transform coefficient for transmission with different segments in different updates. This allows the most significant part of a coefficient to be transmitted early in the update sequence, with the finer resolution transmitted later. An alternative way of considering this is that it permits a fixed number of bits available for an update to be made up of different coefficients to achieve the best image quality. The segmentation is possible without compromising the overall exact invertibility of the algorithm. Although the practical investigation of the partitioning of the coefficients was extremely limited, it does show that there are immediate potential benefits in taking the information for one update from several coefficients. The method is related to the practice of quantising some coefficients to a more coarse degree, as used with conventional transform coding, although the methods are not exactly parallel.

A second technique, also considered to be novel to transform coding was investigated. This technique was specifically referred to as interpolation, although the general area is simply that of post-processing to improve the quality of the displayed images. Using transform coding the boundaries of the blocks over which the transform is taken are usually visible during PTD. To minimise the visibility of the block boundaries, the display information needs to be modified without corrupting the already received data. Using a conventional linear transform, the received coefficients would need to be stored separately from the display data. They would then be available for inverse transformation at each iteration. It is not practical to repeatedly transform the display information as the process would increase degradation of the image. The image quality will not degrade using the mechanism within the MT. The difficult aspect of the post-processing in combination with the MT is the choice of the coefficients not yet transmitted in order to minimise the visibility of the block boundaries. Although the interpolative method, used to produce the photographic results in section 13 was not very successful, some suggestions were made for its direct improvement. Other techniques not directly related to interpolation were also suggested, and the way in which these post-processing techniques interact with the CMT itself was also discussed. There is certainly potential for further investigation on this post-processing topic.

It is clear that even without the extensions discussed above, more computation is required to perform the CMT than the linear transform upon which it is based. From the review in section 1.3 it can be seen that any increase in the complexity of PTD algorithms is to be avoided, and a sophisticated 'fast' implementation of the CMT would be required for it to be useful. As with most image coding algorithms, it is important to consider what techniques are required for the implementation of the algorithm, specifically if floating-point arithmetic is required. This problem was addressed in section 13, where it was shown that the CMT could be implemented entirely by table look-up, though hiding to some extent the complexity of calculating addresses for the look-up. The existence of such an implementation makes the use of the CMT much more feasible using current and future technology.

The MT does require further effort for it to be generally useful for PTD, but it is suggested that some of the required changes are straightforward. The solution of the problem associated with the correctly approximated area should be justified by the potential for use of the MT. It would be worthwhile to investigate quad-tree and oct-tree decompositions with the MT, and the possibility of post-processing promises further improvements.

Overall, this thesis has shown that where image transmission is not at a subjectively adequate rate for conventional slow-scan methods, there are simple progressive techniques offering improvement without sacrificing ultimate image quality with little or no cost. More sophisticated techniques are possible which will give improved performance, both in terms of increasing the subjective usefulness of the early transmitted data, and by disguising any image segmentation by using post-processing techniques.

References

Ahmed, N., Natarajan, T. & Rao, K.R. 1974. 'On Image Processing and a Discrete Cosine Transform' *IEEE Transaction on Computers*, C-23, January 1974, 90-93

Baronetti, G., Guglielmo, M. & Riolfo, B. 1979. 'A Frozen Picture Transmission System employing Orthogonal Transformations' *Proceedings of the 1979 Picture Coding Symposium, Ipswich, UK*

Bisherurwa, E.J.K. 1983. 'Picture Coding Techniques for Videotex and Facsimile' Thesis no. D50739'84, Univ. of Essex

Bisherurewa, E.J.K. & Coakley, F.P. 1981. 'New Fast Discrete Sine Transform With Offset' *Electronics Letters*, 17, No. 21, 803-805

Bisherurwa, E.J.K & Coakley, F.P. 1982. 'Improvements in the Quality of Pictures Produced From a Few Low-Sequency DCT and DST Coefficients' *IEE Conference Publication*, 214, 230

Bisherurwa, E.J.K. & Coakley, F.P. 1983. 'Transform Coding Techniques for Photovideotex' *IEE Colloq. Transform Techniques in Image Processing*, May 1983, Digest no. 1983/50, Section 9

Burt, P.J. 1981. 'Fast Filter Transforms for Image Processing' *Computer Graphics and Image Processing*, 16, 20-51

Burt, P.J. & Adelson, E.H. 1983. 'The Laplacian Pyrammid as a Compact Image Code' *IEEE Transactions on Communications*, COM-31, No. 4, 532-540

Cazin, J., Pearson, D.E. & Whybray, M. 1985. 'Progressive Picture Transmission Using Interleaved Blocks' *Electronics Letters*, 21, 1181

Clarke, R.J. 1983. 'Performance of Karhunen-Loeve and Discrete Cosine Transforms for Data Having Widely Varying Values of Intersample Correlation Coefficients' *Electronics Letters*, 19, No. 7, 251-253

Clarke, R.J. 1984. 'Relation Between the Karhunen-Loeve and Sine Transforms' *Electronics Letters*, 20, No. 1, 12 -13

Fanelli, A.M. & Marangelli, B. 1984. 'Data Compression for Progressive Transmission of Screened Images' *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-14, No. 3, 519-524

Fanelli, A.M. & Marangelli, B. 1985. 'Progressive Transmission of News Photos' *Computer Vision, Graphics and Image Processing*, 27, 239-245

Frank, A.J., Daniels, J.D. & Unangst, D.R. 1980. 'Progressive Image Transmission Using a Growth-Geometry Coding' *Proceedings of the IEEE*, 68, No. 7, 897-909

Ghanbari, M. & Pearson, D.E. 1978. 'Probability Density Functions for Hadamard Coefficients in Transformed Television Pictures' *Electronics Letters*, 14, No. 8, 252-254

Hardas, D.M. & Srihari, S.R. 1984. 'Progressive refinement of 3-D Images Using Coded Binary Trees: Algorithms and Architecture' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6, No. 6, 748-757

Harmuth, H.F. 1969
Transmission of Information by Orthogonal Functions , Springer, New York

Hill, F.S. 1983. 'Interactive Image Query System using Progressive Transmission' *ACM Computer Graphics*, **17**, No. 3, 323-330

Jain, A.K. 1979. 'A Sinusoidal Family of Unitary Transforms' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-1**, No. 4, 356-365

Johnsen, O. & Netravali, A.N. 1981. 'Progressive Transmission of Two-Tone Images' *IEEE Transactions on Communications*, **COM-29**, No. 12, 1934-1941

Knowlton, K. 1980. 'Progressive Transmission of Grey-Scale and Binary Pictures by Simple, Efficient, and Lossless Encoding Schemes' *Proceedings of the IEEE*, **68**, No. 7, 885-896

Lohscheller, H. 1984. 'A Subjectively Adapted Image Communication System' *IEEE Transactions on Communications*, **COM-32**, No. 12, 1316-1322

Netravali, A.N. & Bowen, E.G. 1981. 'A Picture Browsing System' *IEEE Transactions on Communications*, **COM-29**, No. 12, 1968-1976

Ngan, K.N. 1984 'Image Display Techniques Using the Cosine Transform' *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASSP-32**, No. 1, 173-177

Nicol, R.C. & Fenn, B.A. 1979. 'Coding for a Viewdata "Picture in Picture" Facility' *Proceedings of the 1979 Picture Coding Symposium, Ipswich, UK*

Nicol. R.C., Fenn, B.A. & Turkington R.D. 1980. 'Transmission Techniques for Picture Viewdata' *IEE Conference Publication*, **191**, 109-113

Onoe, M., Yasuda, Y. & Inamoto, Y. 1979. 'Time Sequential Coding of Dithered Binary Pictures' *Proceedings of the 1979 Picture Coding Symposium, Ipswitch, UK*

Packwood, R.A. & Martin, G.R. 1983. 'Coding Technique for Progressive Transmission of Static Images' *Electronics Letters*, **19**, No. 11, 412-413

Packwood, R.A. & Martin G.R. 1984. 'A coding technique with post-processing for progressive display applications' *Proceedings of the 1984 Picture Coding Symposium 84, CCETT, France*

Pearson, D.E. & Whybray, M.W. 1984. 'Transform Coding of Images Using Interleaved Blocks' *IEE Proceedings*, **131**, No. 5, 466-472

Pratt W.K. 1978. *Digital Image Processing*. Wiley Interscience

Rifman, S.S. & Mckinnon D.M. 1974. 'Evaluation of Digital Correction Techniques for ERTS Images' TRW report no. 20634-6003-TU-00

Robinson, J.A. & Coakley, F.P. 1983. 'Picture Coding for Photovideotex' *Computer Communications*, **6**, No. 1, 3-13

Sanz, A. Munos, C. & Garcia, N. 1982. 'On the Use of Splines in Hierarchical Image Transmission' *IEEE International Conference on Acoustics Speech and Signal Processing*, **1**, 456-459

Schreiber, W.F. 1956. 'The Measurement of Third Order Probability Distributions of Television Signals' *IRE Transactions on Information Theory*, **IT-2**, No. 3, 94-105

Shore, J.E. 1973. 'On the Application of Haar Functions' *IEEE Transactions on Communications*, COM-, No. , 209-216

Sloan, K.R. & Tanimoto, S.L. 1979. 'Progressive Refinement of Raster Images' *IEEE Transactions on Computers*, C-28, No. 11, 871-873

Spriggs, H. & Nightingale, C. 1986. 'Recursive Binary Nesting, A Quadtree Approach to Image Compression' *Proceedings of the 1986 Picture Coding Symposium, Tokyo, Japan*

Takikawa, H. 1984 'Fast Progressive Reconcstruction of a Transformed Image' *IEEE Transactions on Information Theory*, IT-30, No. 1, 111-117

Tamminen, M. 1984a. 'Encoding Pixel Trees' *Computer Vision, Graphics and Image Processing*, 28, 45-57

Tamminen, M. 1984b. 'Comment on Quad- and Octtrees' *Communications of the ACM*, 27, No. 3, 248-249

Wendler, T. & Meyer-Ebrecht, D. 1982 'Proposed standard for variable format picture processing and a codec aroach to match diverse imaging devices' *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, 318, 298-305

Wilson, R. 1984. 'Quad-Tree Predictive Coding: A New Class of Image Data Compression Algorithms' *IEEE International Conference on Acoustics Speech and Signal Processing*

Yasuda, Y., Tagaki, M. & Awano, T. 1979. 'Hierarchical Coding of Still Images' *Proceedings of the 1979 Picture Coding Symposium, Ipswich, UK*

Yasuda, Y., Tagaki, M. & Awano, T. 1980. 'Step by Step Image Transmission and Display from Gross to Fine Information Using Hierarchical Coding' *The Transactions of the IECE of Japan*, E-63, No. 4, 305-306

Appendix A. The Two-element Mean and Difference Transform

In this appendix the problem of combining two data values into a mean and difference representation is considered, with emphasis on exact invertibility. This invertibility means calculation of the original data values from the mean and difference representation. Specifically detailed below are the number of mean and difference values for a given range of data values and how they combine to produce valid data points during the inversion process.

There are several different ways of formulating transforms with similar behaviour all related to the mean and difference of two data values. If the data values are x, y and the values related to the mean and difference are A, B a linear transform approach would give

$$A = \frac{1}{\sqrt{2}}\left[x+y\right] \qquad B = \frac{1}{\sqrt{2}}\left[x+y\right]$$
$$x = \frac{1}{\sqrt{2}}\left[A+B\right] \qquad y = \frac{1}{\sqrt{2}}\left[A-B\right]$$

Moving the $1/\sqrt{2}$ fraction from the inverse to the forward transform gives the semi-sum (mean), semi-difference formulation

$$A = \frac{1}{2}\left[x+y\right] \qquad B = \frac{1}{2}\left[x-y\right]$$
$$x = A + B \qquad y = A - B$$

This gives the A value the mean of the data values, making it suitable for display as an approximation to the data values. A third variant

$$A = \frac{1}{2}\left[x+y\right] \qquad B = x - y$$
$$x = A + \frac{B}{2} \qquad y = A - \frac{B}{2}$$

and finally

$$A = x + y \qquad B = x - y$$
$$x = \frac{1}{2}\left[A+B\right] \qquad y = \frac{1}{2}\left[A-B\right]$$

Even this set of transforms is not symmetrically complete, there could be sum with semi-difference and any number of others with the fraction 1/2 factorised in different ways as it is by $1/\sqrt{2}$. For x, y having a fixed range of values, the number of distinct values of A and B is the same for each of these mini-transforms, although the actual values will be different. The number of distinct values is almost twice the range of the data as indicated by the summing of the data values for those transforms having A equal to the sum of x and y. Thus the mean value, although having a range equal to the data range, must have half value precision, keeping the number of distinct values constant. The difference (B) has the same number of values as the mean (A) but the range is extended by including negative values. With both the A and B values having approximately twice the range of the data, the number of points they can represent is approximately four times the number of possible data points. It is this increased number of obtainable points that makes any intermediate representation based on one of the variants of A, B have some redundancy. Given this increased point space it is useful for further investigation of mean/difference transforms, to investigate the relationship between the A and B values and exactly what their range of values is for a given range of data.

For data values in the range $0, n-1$ the mean must have a range $0, n-1$ with a resolution of one half giving $2n-1$ possible mean values.The difference must have a corresponding range of $0, n-1$ but of either sign again giving $2n-1$ difference values. Note that this is not the same as simply adding half-unit resolution to the mean as this would allow the mean value $n-1+0{\cdot}5$, which is not a valid mean value. Neither is it the same as adding a sign bit to the difference as this would permit a double representation of zero. The number of valid mean difference/combinations is $(2n-1)^2$ of which n^2 are valid input points.

$2n-1$ possible difference values	$-(n-1), \ldots, -1, 0, 1, \ldots, n-1$
$2n-1$ possible mean values	$0, 0{\cdot}5, 1, 1{\cdot}5, 2, \ldots, \ldots, (n-1)$

The number of difference values which give a valid data point for each mean value varies as a symmetric linear progression. Consider a mean of zero, then the only possible difference value is zero, for a mean value of 0·5, two differences are possible, plus/minus 0·5. For a mean of one, three differences are possible, plus/minus one or zero. The difference value of zero is only possible for integer mean values, not for fractional mean values. This combines with the symmetry of plus/minus differences to give a linear progression with difference and starting values of one. This progression continues up to the mean value of $(n-1)/2$ which has the maximum number of possible differences associated with it, that is n differences.

As the mean value increases from this point, the number of possible difference values decreases symmetrically with the increase. This continues to the mean value $(n-1)$ which has one possible difference, zero. The correspondence between mean values and possible differences is then

mean value	0	0·5	1	...	$(n-1)/2$	...	1
no. of possible differences	1	2	3	...	n	...	1

An arithmetic progression such as this is summed by the expression

$$\frac{m}{2}\left[\, 2a+\left[\, m-1\,\right] d\,\right]$$

where m is the number of terms, a is the value of the first term, and d is the difference between terms. In this instance omitting the middle value of the sequence leaves two identical progressions of $(n-1)$ terms from 1 to $(n-1)$ which individually sum as

$$\frac{n-1}{2}\left[\, 2+\left[\, n-2\,\right]\right]=\frac{n^2-n}{2}$$

from the symmetry this must be doubled and the middle value of the progression added to the sum

$$2\left[\,\frac{n^2-n}{2}\,\right]+n=n^2$$

which is the expected number of data points. The behaviour of the varying number of differences that may be associated with the mean value, matches that which can easily be seen in a geometrical interpretation. The true linear transform represented geometrically has the square data space (in a plane) surrounded by a square at 45° with sides intersecting the vertices of the data space.

This relatively detailed investigation of an often dismissed problem was originally undertaken as the precursor to an investigation of how mean and difference values combine for transforms over three and ultimately four points. A four point transform would be required for quad-tree coding. The detail is necessary as when investigating exactly invertible coding methods with no redundant data transmission, it is important to know that the number of possible mean values is $2n-1$ as described above, rather than the often assumed $2n$. A full investigation of value combinations over more points was never undertaken as it was considered too difficult a method of gaining insight into a more global problem. As a first consideration, over three points the difference values occupy in a geometrical sense a 2-D space of rhombic shape; the equivalent space occupied for a four point transform was not determined. Even finding a mean and difference representation is a simplification of the problem that with transform coding has vectors oriented not through the vertices of the data space. Perhaps more useful than an investigation of a transform over a greater number of points, would be an investigation similar to that above for a two point transform with arbitrary but invertible combinations of data values.

Appendix B. The Full Series of Matrices for the CMT

This appendix contains the complete set of basis matrices calculated in the production of the 8 point CMT. The only purpose of this appendix is to show the behaviour of the process that generates these matrices as the set of inverses of the axes matrices. The first basis matrix is generated from the transform equations that take data values $x(i)$ to transform coefficients $X(i)$, the equations here are as given in section 11·1

$$X(0) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x(i)$$

$$X(u) = \sqrt{\frac{2}{N}} \sum_{i=0}^{N-1} x(i) \cos\left[\frac{\left(2i+1\right) u \pi}{2N} \right]$$

$$u = 1, \ldots N-1$$

with $N = 8$. The resulting first basis matrix is

basis matrix ${}^{0}b$

0·3535	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535	0·3535
0·4904	0·4157	0·2778	0·0975	−0·0975	−0·2778	−0·4157	−0·4904
0·4619	0·1913	−0·1913	−0·4619	−0·4619	−0·1913	0·1913	0·4619
0·4157	−0·0975	−0·4904	−0·2778	0·2778	0·4904	0·0975	−0·4157
0·3536	−0·3536	−0·3536	0·3536	0·3536	−0·3536	−0·3536	0·3536
0·2778	−0·4904	0·0975	0·4157	−0·4157	−0·0975	0·4904	−0·2778
0·1913	−0·4619	0·4619	−0·1913	−0·1913	0·4619	−0·4619	0·1913
0·0975	−0·2778	0·4157	−0·4904	0·4904	−0·4157	0·2778	−0·0975

The above matrix is inverted by transposition to give the first axis matrix which is

axes matrix ${}^{0}a$

0·3536	0·4904	0·4619	0·4157	0·3536	0·2778	0·1913	0·0975
0·3536	0·4157	0·1913	−0·0975	−0·3536	−0·4904	−0·4619	−0·2778
0·3536	0·2778	−0·1913	−0·4904	−0·3536	0·0975	0·4619	0·4157
0·3536	0·0975	−0·4619	−0·2778	0·3536	0·4157	−0·1913	−0·4904
0·3536	−0·0975	−0·4619	0·2778	0·3536	−0·4157	−0·1913	0·4904
0·3536	−0·2778	−0·1913	0·4904	−0·3536	−0·0975	0·4619	−0·4157
0·3536	−0·4157	0·1913	0·0975	−0·3536	0·4904	−0·4619	0·2778
0·3536	−0·4904	0·4619	−0·4157	0·3536	−0·2778	0·1913	−0·0975

To produce the sequence of axis matrices of which the basis matrices are the inverses, for each new axis matrix the left-most column of the preceding matrix is deleted and row is also deleted. The index of the row to be deleted is taken from the list

4,0,7,2,5,1,6,3

Note that these indexes refer to the rows of the first axis matrix ${}^{0}a$ and are not used by counting down the rows of any other matrix. The first axis matrix is considered for the purposes of developing the MT as a set of row vectors, and each of these row vectors has modulus one. As successive columns are deleted from the first axis matrix to form the sequence of axis matrices, the row vectors have decreasing dimension. As each coefficient is lost there is a corresponding reduction in modulus of the row vector. As these axes matrices are inverted to find the basis matrices there is a corresponding increase in the coefficients of the basis matrices to maintain the unit matrix product. The remaining matrices of the sequence are given below.

basis matrix ${}^{1}b$

0·5879	0·5133	0·3753	0·1951	−0·1802	−0·3182	−0·3928
0·9239	0·6533	0·2706	0·0000	0·2706	0·6533	0·9239
0·1379	−0·3753	−0·7682	−0·5556	0·2126	−0·1802	−0·6935
0·0000	−0·7071	−0·7071	0·0000	−0·7071	−0·7071	0·0000
0·6935	−0·0747	0·5133	0·8315	0·3182	0·9061	0·1379
0·3827	−0·2706	0·6533	0·0000	0·6533	−0·2706	0·3827
−0·3928	−0·7682	−0·0747	−0·9808	−0·9061	−0·2126	−0·5879

basis matrix ${}^{2}b$

−0·1533	−0·3192	−0·3066	0·5538	1·1533	1·5412
−0·4958	−0·8562	−0·6013	0·2549	−0·1056	−0·6013
−0·7071	−0·7071	0·0000	−0·7071	−0·7071	0·0000
−0·6801	0·0705	0·6013	0·5308	1·2815	0·6013
−0·6047	0·4090	−0·1270	0·7706	−0·0635	0·6384
−0·4252	0·1761	−0·8504	−1·0266	−0·4252	−0·8504

basis matrix ${}^{3}b$

−0·5556	−0·9808	−0·7210	0·4710	0·3444
−0·7071	−0·7071	0·0000	−0·7071	−0·7071
−0·6203	0·1951	0·7210	0·3147	0·8315
−0·5412	0·5412	0·0000	0·5412	−0·5412
−0·5098	0·0000	−1·0196	−0·7210	0·2112

basis matrix ${}^{4}b$

−0·3066	0·5198	−1·0467	−0·9554
−0·7308	0·5776	0·4084	0·9000
−0·8478	−0·3978	0·8011	−0·3512
−0·5098	−1·0196	−0·7210	0·2112

basis matrix ${}^{5}b$

−0·8504	0·7804	0·5272
1·0824	0·0000	−1·0824
−0·2986	−1·3776	0·8693

basis matrix ${}^{6}b$

−0·9932	−1·7534
−1·6516	0·6841

Appendix C. Some CMT Numerical Examples

This appendix contains some numerical examples of use of the 8 point CMT, based on a data range of 0-63. Some of the data sequences used have obviously been generated as test sequences. The remaining sequences were taken from different images over a period of time and have no special properties. They are not intended to illustrate any particular behaviour and serve only as simple examples. They may be of use to anyone attempting to implement the CMT. The CMT results are in the same format as those in section 11.2, the 'code' values are essentially twos-complement values and the first coefficient is offset by half the range of the data.

	data	16	16	24	24	32	32	40	40		
	code	60	49	63	2	2	62	2	57		
	display	28	28	28	28	28	28	28	28	MSE	80.0000
	display	16	17	21	26	31	35	39	41	"	3.2500
	display	15	17	20	25	31	34	38	40	"	3.5000
CMT	display	15	18	22	26	31	34	38	41	"	2.8750
	display	15	16	22	25	30	34	37	41	"	3.0000
	display	15	15	23	25	30	35	39	41	"	2.3750
	display	15	15	22	26	30	34	39	41	"	2.5000
	display	16	16	24	24	32	32	40	40	"	0.0000

	data	16	16	24	24	32	32	40	40		
	code	79	−24	0	0	0	1	0	5		
	display	28	28	28	28	28	28	28	28	MSE	80.0000
	display	16	18	21	26	30	35	38	40	"	4.2500
	display	16	18	21	26	30	35	38	40	"	4.2500
DCT	display	15	18	22	26	30	34	38	41	"	3.2500
	display	15	18	22	26	30	34	38	41	"	3.2500
	display	16	17	22	26	30	34	39	40	"	2.2500
	display	16	17	22	26	30	34	39	40	"	2.2500
	display	16	16	24	24	32	32	40	40	"	0.0000

	data	0	8	16	24	32	40	48	56		
	code	60	34	0	6	1	0	3	2		
	display	28	28	28	28	28	28	28	28	MSE	336·0000
	display	3	6	13	23	33	42	49	53	"	4·7500
	display	3	6	13	23	33	42	49	53	"	4·7500
CMT	display	1	6	17	24	32	39	47	56	"	1·0000
	display	1	6	18	24	32	39	47	56	"	1·3750
	display	1	6	18	24	32	39	47	56	"	1·3750
	display	0	8	16	23	32	38	48	55	"	0·7500
	display	0	8	16	24	32	40	48	56	"	0·0000

	data	0	8	16	24	32	40	48	56		
	code	79	−51	0	−4	0	−1	0	0		
	display	28	28	28	28	28	28	28	28	MSE	336·0000
	display	3	7	14	23	33	42	49	53	"	3·7500
	display	3	7	14	23	33	42	49	53	"	3·7500
DCT	display	0	7	16	24	32	40	49	56	"	0·2500
	display	0	7	16	24	32	40	49	56	"	0·2500
	display	0	8	16	24	32	40	48	56	"	0·0000
	display	0	8	16	24	32	40	48	56	"	0·0000
	display	0	8	16	24	32	40	48	56	"	0·0000

	data	54	53	53	53	42	48	19	33		
	code	12	16	50	1	61	4	46	24		
	display	44	44	44	44	44	44	44	44	MSE	138·6250
	display	58	55	52	47	42	37	33	31	"	47·2500
	display	54	53	53	50	46	37	30	26	"	39·5000
CMT	display	53	53	53	50	46	36	30	27	"	40·8750
	display	54	52	52	51	47	35	29	28	"	40·6250
	display	54	54	53	50	50	36	26	30	"	34·5000
	display	56	49	59	46	48	42	21	32	"	22·7500
	display	54	53	53	53	42	48	19	33	"	0·0000

	data	54	53	53	53	42	48	19	33		
	code	125	27	−8	0	3	−5	12	−10		
	display	44	44	44	44	44	44	44	44	MSE	138·6250
	display	58	56	52	47	42	37	33	31	"	47·8750
	display	53	54	54	51	46	39	31	27	"	35·5000
DCT	display	53	54	54	51	46	39	31	27	"	35·5000
	display	54	53	53	52	47	37	30	28	"	36·6250
	display	53	55	52	50	50	38	27	30	"	31·5000
	display	55	50	57	48	47	44	22	32	"	12·7500
	display	54	53	53	53	42	48	19	33	"	0·0000

	data	38	36	37	32	33	31	31	28		
	code	1	5	0	63	3	63	1	62		
	display	33	33	33	33	33	33	33	33	MSE	10·5000
	display	38	37	36	34	33	31	30	29	"	1·0000
	display	38	37	36	34	33	31	30	29	"	1·0000
CMT	display	38	36	35	33	33	30	29	28	"	1·2500
	display	38	37	37	32	33	32	30	28	"	0·3750
	display	38	36	37	32	33	32	30	28	"	0·2500
	display	38	36	37	32	33	32	31	28	"	0·1250
	display	38	36	37	32	33	31	31	28	"	0·0000

	data	38	36	37	32	33	31	31	28		
	code	94	9	0	1	0	0	1	3		
	display	33	33	33	33	33	33	33	33	MSE	10·5000
	display	37	37	36	34	32	31	30	29	"	1·2500
	display	38	37	36	34	32	31	30	29	"	1·1250
DCT	display	38	37	35	34	33	31	30	29	"	1·3750
	display	37	37	36	33	32	32	30	28	"	0·8750
	display	38	37	36	33	32	32	31	28	"	0·6250
	display	38	37	36	33	32	32	30	28	"	0·7500
	display	38	36	37	32	33	31	31	28	"	0·0000

	data	34	35	36	36	35	38	37	37		
	code	4	62	63	1	3	63	1	5		
	display	36	36	36	36	36	36	36	36	MSE	1·5000
	display	35	35	35	36	37	37	38	38	"	1·1250
	display	34	34	35	36	37	36	37	38	"	1·3750
CMT	display	34	34	36	36	37	35	36	38	"	2·0000
	display	33	34	37	35	36	37	37	38	"	0·8750
	display	34	34	37	35	36	37	37	37	"	0·6250
	display	34	34	37	35	36	37	38	37	"	0·7500
	display	34	35	36	36	35	38	37	37	"	0·0000

	data	34	35	36	36	35	38	37	37		
	code	102	−2	0	0	0	0	1	0		
	display	36	36	36	36	36	36	36	36	MSE	1·5000
	display	35	35	35	36	36	37	37	37	"	0·5000
	display	34	35	35	36	36	37	37	37	"	0·3750
DCT	display	34	35	35	36	36	37	37	37	"	0·3750
	display	34	35	36	35	36	37	38	37	"	0·5000
	display	34	35	36	36	36	37	38	37	"	0·3750
	display	34	35	36	35	36	38	37	37	"	0·2500
	display	34	35	36	36	35	38	37	37	"	0·0000

	data	46	47	46	44	41	30	25	19		
	code	5	16	48	0	0	0	5	1		
	display	37	37	37	37	37	37	37	37	MSE	105·5000
	display	51	48	45	40	35	30	26	24	"	13·1250
	display	46	46	46	44	39	30	23	18	"	1·2500
CMT	display	46	46	46	44	39	30	23	18	"	1·2500
	display	46	46	46	44	39	30	23	18	"	1·2500
	display	46	46	46	44	39	30	23	18	"	1·2500
	display	47	47	46	44	42	30	24	19	"	0·3750
	display	46	47	46	44	41	30	25	19	"	0·0000

	data	46	47	46	44	41	30	25	19		
	code	105	27	−9	0	1	0	−1	2		
	display	37	37	37	37	37	37	37	37	MSE	105·5000
	display	51	49	45	40	35	30	26	24	"	13·5000
	display	46	47	47	45	39	32	24	19	"	1·3750
DCT	display	46	47	46	44	39	32	24	19	"	1·1250
	display	46	46	46	45	40	32	24	19	"	1·0000
	display	46	47	46	44	40	32	24	20	"	0·8750
	display	46	47	45	45	40	31	25	19	"	0·5000
	display	46	47	46	44	41	30	25	19	"	0·0000