

Description

Image data compression has been a topic of considerable interest in recent years because of the burgeoning fields of information and communication. Although a picture is worth a thousand words, with unadorned techniques it may take more transmission capacity to handle a single picture than many thousands of words. Compared to text, the information content in an image is normally quite large and, therefore, the efficient transmission and/or storage of pictorial information has been subjected to extensive research for a number of years.

A number of factors have come into play that assist in that research. First, in many instances, the images stored or transmitted are static. No movement needs to be communicated, and that relaxes the storage and transmission requirements. Second, users find it acceptable for motion to have a limited temporal resolution. That too relaxes the storage requirements. Third, sometimes the information need not be handled on a real-time basis. Consequently, there is time to indulge in rather sophisticated processing techniques.

Another reason why image data compression is possible is that images contain a considerable amount of superfluous information. Generally, one can identify two kinds of superfluous information. Statistical redundancy, having to do with similarities in the data representing the image; and subjective redundancy, having to do with data similarities which can be removed without user complaints. Statistical redundancy is illustrated by the transmission of a white page which may be depicted without repeatedly specifying (at every pixel) the color of the page. Subjective redundancy is illustrated by the fact that in depicting movement one can ignore events that occur faster than a certain speed, because the human eye cannot discern those events.

Researchers have tried to encode images in a number of ways that realize transmission and storage savings by eliminating the aforementioned redundancies. These encoding approaches can be broadly divided into predictive Pulse Code Modulation, Interpolative or Extrapolative Coding, Transform Coding, and Vector Quantization.

In predictive coding, such as in the Differential Pulse Code Modulation (DPCM) approach, an attempt is made to predict the pixel to be encoded. The prediction is made by using the encoded values of the previously encoded pixels. Generally, these pixels are combined to generate a predicted value; and the signal that is quantized, encoded and transmitted is the difference between the actual value and the generated predicted value. Adaptive DPCM, where the prediction algorithm is based on local picture statistics, is a variation on this approach.

In interpolative or extrapolative coding, only a

subset of the pixels is sent to the receiver. The receiver must then interpolate the available information to develop the missing pixels.

In transform coding, instead of coding the image as discrete intensity values on sets of sampling points, an alternative representation is made first by transforming blocks of pixels into a set of coefficients. It is the coefficients that are quantized and transmitted. Several transformations have been used in the art, such as the Hadamard, the Karhunen-Loeve, and the Discrete Cosine Transforms. These transforms, being unitary, conserve the signal energy in the transform domain but, typically, most of the energy is concentrated in relatively few samples. Predominately, it is the samples representing the low frequencies that are non-zero. Samples that are zero, or ones that are very close to zero, need not be transmitted. Still additional samples of low value can be quantized coarsely, resulting in substantial compression of the data that needs to be either stored or transmitted.

Yet another encoding technique is Vector Quantization, where an image block is decomposed into a set of vectors. From the possible (or experienced) signal patterns and corresponding vectors, a subset of representative vectors is selected and included in a code book. When encoding, the developed sets of vectors are replaced with the closest representative vectors in the code book, and compression is achieved by further replacing the representative vectors with labels. At the receiver, the inverse operation takes place with a table look-up which recovers the representative vectors. A facsimile of the image is reconstructed from the recovered representative vectors.

The decoding process in vector quantization is quite simple, but that is not the case on the encoding side. Primarily this is because one must develop the code book, and that becomes impractical when the image block is large or when good fidelity is desired. Also, to develop those representative vectors one must use a "training" set of data.

The present invention seeks to provide methods and encoders for encoding data signals in an efficient manner, both as regards the number of bits used to describe the data and as regards the encoding process itself.

According to one aspect of this invention there is provided a method as claimed in claim 1.

According to another aspect of this invention there is provided a block list encoder as claimed in claim 10.

An embodiment of the invention is realized with a Block List Transform (BLT) approach to image coding. As with the Block Transform Coding and the Vector Quantization approaches, the BLT encodes two dimensional blocks of pixels. Furthermore, it uses predictive coding to represent the pixels of each block

which is processed. The encoding process is different, however, because the pixels are not treated in the normal order of left-to-right, top-to-bottom. Instead, they are coded in the order prescribed by a predetermined "List". The code for each pixel is developed by computing a prediction based on the previously coded pixel values in the neighborhood of the pixel being coded and subtracting this prediction from the true value of the present pixel. This results in a sequence of predominantly non-zero valued pixel prediction errors at the beginning of the encoding process (of each block), and predominantly zero valued pixel prediction errors towards the end of the encoding process. Enhancements to this approach include adaptive encoding, where pixel sequences having expected zero prediction errors are not encoded, and run length encoding of the signals to be transmitted/stored for augmenting the encoding process.

The invention will now be described with reference to the accompanying drawings in which:

FIG. 1 depicts a square image and a plurality of blocks dividing the image;

FIG. 2 shows one of the blocks of FIG. 1 in detail and identifies the pixel coordinates within the block;

FIG. 3 is a table specifying the prediction pixels employed in developing a predicted value for each of the pixels in FIG. 2;

FIG. 4 identifies the quantization groupings in the table of FIG. 3;

FIG. 5 is a table identifying the quantization interval and rounding values for the groupings of FIG. 4;

FIG. 6 is a table that presents the pixels of FIG. 2 in an order, as close as possible, according to geometrical proximity of the pixels;

FIG. 7 is a table that introduces the prediction orders and thresholds used in an adaptive encoding method pursuant to the invention;

FIG. 8 is a block diagram of an encoder embodying the invention;

FIG. 9 is a block diagram illustrating a decoder embodiment adapted to decode the signals developed by the FIG. 8 encoder;

FIG. 10 is a block diagram illustrating an encoder embodying the invention and employing adaptive encoding; and

FIG. 11 is a block diagram illustrating a decoder embodiment adapted to decode the signals developed by the FIG. 10 encoder.

An image to be encoded may be divided into overlapping blocks, or non-overlapping blocks as shown in FIG. 1. Typically the blocks are small enough so that there is reasonable correlation between the pixels in the block. FIG. 2 shows an 8x8 block of pixels, such as block 10 in FIG. 1, along with some pixels from previously coded blocks above and to the left of block 10. The first pixel subscript is the row

number, and the second is the column number. The pixels to be coded are in rows 1-8, columns 1-8, while row 0 and column 0 are from previously transmitted blocks, and need not be coded. FIG. 2 can also be viewed as an overlapping block, having one overlapping row and one overlapping column.

FIG. 3 is a table which shows one complete list of the pixels in block 10. The list of FIG. 3 specifies the order of coding of the pixels in the block of FIG. 2 and the pixels used in developing the predicted value of each pixel. It should be understood, of course, that the list of FIG. 3 is merely illustrative and that many other lists can be created which achieve essentially the same encoding results. Also included in the table of FIG. 3 are the previously transmitted pixels which are to be used in the prediction of the corresponding pixel to be coded.

The method proceeds as follows. The first pixel to be encoded and sent is the lower right hand pixel 88 (in FIG. 2), for which a prediction is formed from pixels 08 and 80. Pixels 08 and 80 have values that have been previously developed in the course of the encoding process of previous blocks. The difference between the actual value of pixel 88 and its prediction is quantized, coded and transmitted. This is the quantized differential prediction error. A coded value for pixel 88 is then formed by adding the prediction to the quantized differential error. This is also the value which is reconstructed at the receiver. The coded value of pixel 88 is substituted in block 10 for the original value of the pixel. Next, a prediction for pixel 84 is developed using pixels 80 and 88 and the difference between the prediction value and the actual value is encoded and sent. Pixel 84 is then replaced in block 10 by its coded value. This procedure continues until all remaining pixels in the list have a zero quantized differential prediction error. At that time, an End-Of-Block (EOB) message is sent, and the coder moves onto the next block of pixels to be encoded.

Actually, in some realizations it is advantageous (as will become more apparent infra) to use a small buffer where a few values are saved prior to transmission. Also, a zero counter keeps track of strings of zero values. Zero values are not sent right away and may not be sent at all, if they occur immediately before an EOB.

It may be observed that as the algorithm progresses further down the list, the prediction pixels become spatially closer and closer to the pixel to be coded. This leads to a prediction error that, on average, becomes smaller and smaller toward the end of the list.

It may also be noted that, as in Block Transform Coding, in BLT coding the last few non-zero data are often preceded by many zero values. Encoding and sending all those zero values in order to read the few non-zero data points can cause an unnecessarily high bit rate. Therefore, if the last few nonzero values

(stored in the aforementioned small buffer) are small, they can usually be truncated to zero without seriously affecting coded image quality, and the overall bit rate is then reduced.

Another property for which the BLT can be used to good advantage is that quantization of the prediction error can become coarser and coarser as the encoding process continues down the list. This occurs for two reasons. First, nonzero BLT data towards the end of the list are caused by edges or texture in the picture; and in these regions it is well known that quantization noise is less visible than in flat, low detail areas of the image. Secondly, for pixels further down the list, only large magnitude intensity transitions are able to cause sizable BLT data, because the neighboring points are, presumably, close to the actual average of the image. Therefore, large quantization errors will not be as visible for these pixels. Thirdly, the feedback quantization of the BLT results in the quantization noise being fairly localized and of high frequency. This further contributes to a lessening of visibility. This, by the way, is in contrast to Block Transform Coding approaches (e.g., Cosine Transform) in which quantization of one coefficient affects all pixels in the block.

Taking advantage of the above-described inherent advantages of the BLT approach, an enhanced algorithm allows for different quantization levels in different groups of pixels. FIG. 4 shows the pixels of FIG. 3 grouped into six classes {1,2,3,4,5,6} for the purpose of defining quantizations. Pixels in the lower classes have their prediction errors quantized finer than those of the upper classes. FIG. 5 shows one possible allocation for uniform quantization. The quantization can be carried out by simply adding the rounding value to the magnitude of the prediction error, dividing by the quantization interval size and multiplying by the sign of the prediction error. This yields a signed integer. Reconstruction is accomplished through multiplication by the interval size.

When encoding certain classes of images, it turns out that clusters of nonzero data points occur along edges of objects and in textured regions. This clustering of BLT data points suggests that savings may be possible if the pixels are sent not in a fixed order, but instead are sent according to a criterion of whether or not nearby data points are zero. The first step in implementing such an adaptive procedure is to reorder the prediction table of FIG. 4 so that pixels which are geometrically close to each other in the block are also close to each other in the table. One possibility of such reordering is shown in the prediction table of FIG. 6. It may be noted that the FIG. 4 quantization classes are retained, and that in some cases the pixels used to develop the prediction values had to be altered in order to utilize only previously coded pixels. This is necessary because the receiver has available only the previously encoded pixels.

The next step is to devise a criterion for determining when there are no more significant data points to be sent in the small region currently being coded. An example of such a procedure is shown in the table of FIG. 7.

The first two columns of the FIG. 7 table are the same as in the prediction table of FIG. 6. The next two columns are entitled "Next Pixel If Skip" and specify which coded pixel to skip to if it is determined that coded pixels can be skipped. If a "Small Skip" is to be attempted, the first pixel entry is used. If a "Large Skip" is to be attempted, the second pixel entry is used.

In general, whenever a zero quantized prediction error is transmitted, a skip of pixels is attempted. Whether or not to attempt a large or a small skip is determined by the previously coded reference pixels. If the coded prediction errors for these reference pixels are sufficiently small then a large skip is attempted. Otherwise, a small skip is attempted. If there are no reference pixels, then only a small skip is attempted.

It may be in certain circumstances that not all skips are in the best interests of maintaining good coded picture quality. For example, in some image blocks pixels 44, 40 and 80 may well have zero quantized prediction error, which would normally imply a large skip from pixel 44 to coded pixel 48. However, in some other image blocks there may be large nonzero quantized prediction errors at some of the intervening coded pixels in the list. For this reason a "Cancel Skip (CS)" message capability is included which may be transmitted by the coder. The occurrence of this message means that the skip attempt which would normally be made because of zero differential values is to be ignored at the coder and the decoder. Decoding then simply resumes with the next pixel, as if the previous pixel had a nonzero differential value. The cancel skip code rarely needs to be sent, however, with well designed Block Lists.

Columns 5-8 of the skip criterion table determine whether or not a CS message is to be transmitted. If a large skip is being attempted, columns 4, 6 and 8 are used; otherwise, columns 3, 5 and 7 are used. The way these parameters are used is as follows. Following the decision to attempt a skip, the coder begins calculating quantized differential values for succeeding pixels up to, but not including, the pixel specified in column 3 or 4, whichever is being used. During this process the nonzero quantized values are counted in accordance with their value. When the absolute value of the quantized differential values is small, e.g., equal to the quantization interval itself, then a "small pixel" counter is incremented. If the nonzero value is large, a "large pixel" counter is incremented. For each pixel examined before a possible CS, a zero differential value is coded for that pixel even if the actual quantized differential value is nonzero and one of the

counters is incremented. If during the skip attempt, the number in the small pixel counter exceeds the value in column 5 or 6, whichever is being used, or the number in the large pixel counter exceeds the value in column 7 or 8, whichever is being used, then a CS message is sent followed by a zero value for each pixel examined so far in the attempted skip. Coding of pixels then resumes in the normal manner. It may be noted that the letter "I" in the table of FIG. 7 indicates "Infinity", which represents an arbitrary large number; and EB stands for End of Block.

By the above-described adaptive means, small areas of zero differential values within a block can be coded without requiring any information to be sent to the receiver. The decoder assumes simply that in the absence of CS message, skipped pixels have zero differential value and can be replaced by their prediction values in the reproduced image.

If skipped pixels occur at the end of the ordered list of coded pixels, then the End-of-Block message can also be deleted from the transmission, as long as the first pixel value in the following block is forcibly transmitted independently of whether it has a zero value. However, it only pays to delete the end-of-block message if less than a certain number of zero values exist. If a long string of zero pixels must otherwise be sent at the end of a block, then sending the end-of-block message takes fewer bits.

As simple as the encoding approach is, the decoding approach is even simpler. To decode an image block with the aid of received signals one must only consult the column and row of two previous blocks, as in FIG. 2, (which is effect seed the process) and the Block List used in the encoding process. A prediction signal is developed based on the prediction pixels specified in the list, and the prediction signal is augmented by the quantized prediction error received from the encoder. The augmented signal is stored in the block and the decoding process continues with the next pixel in the list.

FIG. 8 shows one possible implementation of the basic coding operation.

For a simple arithmetic averaging predictor, all that is required is an accumulator 120, a pixel-RAM 130, a ROM 140 for the controlling list, counters 150 and 160, and a coder 170.

Although the method of the invention deals with blocks, such as the 8x8 block of FIG. 2, one must realize that in many applications one is presented with entire scan lines at a time. Before accepting the first pixel in the 8th scan line, therefore, one must dispose of (or store) the lines of the previous seven scans. Eight scan lines comprise a strip of pixel blocks and, accordingly, the first step is to load an entire strip of pixel blocks (appearing on line 121), into the pixel-RAM. RAM address for this operation are provided by input counter 160. The bottom-most row (8) of pixels from the previously transmitted strip of pixel blocks is

kept in the pixel-RAM and becomes the top-most row (0) of the present strip of pixel blocks. The leftmost column (0) of the leftmost block is filled with some suitable predetermined value, e.g., the value of pixel 01, or in the case of color difference blocks, simply zero.

Coding commences next. Block counter 150 outputs the address of pixel 00 for the block presently being coded and all other addresses are referenced to it with the aid of adder 157. Accumulator 120 contains a register 122 and that register is cleared. Thereafter, list-ROM 140 outputs, for example four addresses corresponding to the four pixels which are employed to develop the prediction value. List ROM 140 contains the entire list of the block, e.g., the list of FIG. 3. These are added by the accumulator (adder 123), and the sum is divided by four (divider 124) to obtain the coded pixel value. Thence, list-ROM 140 outputs the address of the pixel to be coded, the encoded pixel is passed to subtractor 171 along lead 126 where it meets the predicted value on lead 172. A differential error is computed in subtractor 171 and passed on to quantizer 173. Quantizer 173 develops the quantized prediction error (QPE) whose value is added to the prediction value in adder 174 to obtain the coded pixel value. The coded pixel is then written into the pixel-RAM to replace the uncoded pixel value. This coded pixel value is later be used in the predictions of other pixels in the same block, or for pixels in blocks to the right and below in the image. Succeeding pixels of the block are coded similarly, under control of counter 160 which provides addresses to list-ROM 140 and pixel-RAM 130.

The quantized prediction error values may be coded in any one of several ways prior to transmission or storage; most of which are fairly well known. For example, FIG. 8 shows a variable word-length coder 180 for the non-zero values, in parallel with a zero counter 190 which is used to implement run-length coding of zero values. If the end-of-block arrives after a string of zero values (as it usually will), then the final run-length of zeros is not sent. Instead, an end-of-block message is transmitted.

The decoder for the basic BLT is shown in FIG. 9 and it is essentially the inverse of the encoder. The variable word-length decoder (180) produces either nonzero prediction error values or zero-run lengths, depending on what was transmitted. In the case of zero-run lengths, the zero counter is used to produce a string of zero quantized prediction error (QPE) values to be added (in adder 174) to the corresponding prediction values (from accumulator 120) in order to reproduce the coded pixels. Upon reception of an end-of-block signal, the zero counter is used to produce zero QPE values for the remaining pixels of the block. After decoding all blocks of a strip, the data is output from pixel-RAM 130 via lead 126, with the pixel-RAM addresses being provided by output

Counter 160. List-ROM 140 and counters 150 and 160 operate in a manner that corresponds to their operation in the FIG. 8 encoder.

One implementation for an adaptive BLT encoder is shown in FIG. 10. First, all registers are reset to zero (e.g., within accumulators 48 and 64 and elements 69, 70, 71), and image data enters pixel-RAM 41 in the same way as in FIG. 8, with addresses being provided by input counter 42. Coding then commences as in FIG. 8. Pixel addresses within a block are provided by counter 43 and list-ROM 44, and fed to adder 45. The beginning address of each block is output by block-counter 47 and fed to adders 45 and 46. The output of adder 45 then provides the pixel addresses necessary for prediction and coding. Every set of four prediction pixel addresses is followed by one coded pixel address.

Coding commences as in FIG. 8 using accumulator-divide-by-4 circuit 48, subtractor 49, quantizer 50, adder 51, and variable word-length coder 59 together with zero counter 30. The output of quantizer 50 is the Quantized Prediction Error (QPE).

Adaptive BLT coding begins as soon as a zero value for QPE is transmitted by the coder 59. A QPE=0 causes a STRT signal to appear at the output of NOR gate 52, which in turn causes register 53 to be set to 1. The output of Register 53 is the skip attempt indicator (S), which enables certain operations described below. The S signal is fed back to NOR gate 52 in order to disable further STRT signals during this skip attempt. Small/Large indicator 54 checks the magnitude of each nonzero QPE and passes the result to counters 55 and 56. If S=1, indicating that a skip attempt is in progress, then small counter 55 counts the number of "small" nonzero QPEs, while large counter 56 counts the number of "large" nonzero QPEs. Whenever S=0, counters 55 and 56 are reset to zero.

During a skip attempt, S=1 also causes the Coded Prediction Error (QPE) to be forced to zero by means of inverter 57 and AND gate 58. Zero QPEs generated during a skip attempt are counted by zero counter 60 for possible later transmission. Whenever S=0, counter 60 is reset to zero.

Coded Prediction Error (QPE) values are added to the prediction values by adder 51, and the resulting coded pixel values are inserted into pixel-RAM 41 with addresses being provided by adder 45, in the same way as in FIG. 8. However, with adaptive BLT the magnitude of the QPE values (which are provided by gate 58 and modified by ABS circuit 61) are inserted into the prediction-error-RAM 62. Addresses are provided by reference-pixel-ROM 63 and adder 46 in exactly the same way as with list-ROM 44 and adder 45, except that here four reference pixel addresses appear at the output of adder 46 instead of four prediction pixel addresses.

While accumulator 48 is calculating a prediction

value, accumulator 64 is summing the magnitudes of the corresponding reference pixel QPEs. After this is done, circuit 65 examines the resulting value to determine if a large skip or a small skip should be attempted, and passes the answer to ROMs 66, 67 and 68. Memories 66, 67, and 68 contain the information of columns 3-8 in FIG. 7. At the start of a skip attempt, as indicated by the STRT output of NOR gate 52, the outputs of ROMs 66, 67 and 68 are stored in registers 69, 70 and 71, respectively, for use as follows.

Register 69 outputs the number of small nonzero QPEs allowed before a premature cancellation of the skip attempt, and similarly register 70 outputs the number of allowed large nonzero QPEs. Register 71 outputs the next pixel to be encoded if the skip attempt is successful.

The output of counter 55, which is the number of small nonzero QPEs encountered so far, is compared with the maximum allowed (in register 69) by comparator 72. If the number is too large then comparator 72 causes a CANCEL signal to appear at the output of OR gate 74. Similarly, the number of large nonzero QPEs (in counter 56) is compared with the maximum allowed (in Register 70) by comparator 73, and if it is too large the CANCEL signal is generated.

The CANCEL signal is fed to variable word-length coder 59, whereupon it causes a cancel-skip word to be sent to the channel. Following this, the zero run-length output of zero counter 60 is sent, and counter 60 is reset to zero. The CANCEL signal is also fed to OR gate 75, which generates a STOP signal which, in turn, causes the skip indicator output (S) of register 53 to be reset to zero, thus prematurely ending the skip attempt.

During a skip attempt, S=1 enables comparator 76, which compares the output of pixel-counter 43 with the next-pixel-if-skip value in register 71. When they become equal the skip attempt is successful, and the END signal is generated. The END signal is fed to OR gate 75 which resets register 53 to S=0. Coding then recommences in the normal way.

An implementation of an adaptive BLT Decoder is shown in FIG. 11, where most of the modules perform exactly the same function as their counterparts in FIG. 10. In normal operation, variable word-length decoder 89 receives QPE values and passes them to adder 81 for construction of the decoded pixels. However, if a CANCEL message is received, then immediately following may be a zero run-length value. Decoder 89 then outputs the CANCEL signal during the entire time that zero counter 80 is producing zero QPE values corresponding to the received zero run-length value. The CANCEL signal of decoder 89 is fed to both OR gate 95 and to NOR gate 82 in order to prevent a skip, as described below. Following this, normal operation resumes.

If a zero QPE value is received which is not

Immediately followed by a CANCEL signal, then a skip can be made, and a STRT signal is generated at the output of NOR gate 82. Skip indicator register 83 is then set to $S=1$, which disables further operation by decoder 89. Pixels are then decoded using zero QPE values until the END signal is generated by comparator 76, which causes register 83 to be reset to $S=0$. Following this, normal operation again resumes.

Upon reception of an End-of-Block signal, decoder 89 simply outputs a very large zero run-length number to zero-counter 80. This cause zero QPE values to be generated until the end of the block, at which time pixel-counter 43 generates a Start-of-Block signal resetting the system. Frequently, it is not necessary to send an End-of-Block signal, since in many cases the transmission of a few zero QPE values causes automatic skips to the end of the block.

The above-described method and implementation are illustrative of the invention and variations are, of course, possible.

For example, what has been described is an approach where there is effectively a one row and one column overlap between adjacent blocks. It should be realized that any desired overlap is possible.

The obvious tradeoff is that with greater overlap the prediction accuracy improves, but each new block either encodes fewer pixels or is larger.

Another enhancement may be to use a number of pixels other than four in developing the predicted value of a pixel, or of using other than a simple arithmetic average, e.g., median. Again, there is a tradeoff between accuracy and processing effort.

Yet another variation relates to the block list. In some applications (e.g., where the block is very large) it may be easier to develop the block list "on the fly" rather than rely on a large table. Instead of a list-ROM element one would have a processing element that would develop the required sets of prediction pixels.

Still another enhancement relates to the order in which the blocks of FIG. 1 are processed. In facsimile transmission where sequential scanning is the norm (e.g., from top to bottom and from left to right), blocks would most conveniently be encoded in a similar sequence. This is the approach employed in the above illustrative embodiments, with a pixel-RAM storing strips of data. If the pixel-RAM is large enough, however, the entire image may be stored first and the order of encoding can then be different. One could, for example, start with the block in the center of the image and work outwards, or vice versa.

Although the description above relates to encoding and decoding of image information, as has been indicated initially invention relates to the field of data compression generally. The aforesaid pixels are mere illustrative examples of data points, or values, and the method and apparatus described herein in connection with an image compression environment is equally applicable to other data compression tasks, such as

compression of speech signals.

Claims

1. A method for predictively encoding data signals divided into two-dimensional blocks containing a plurality of data point signals, CHARACTERISED BY the steps of:

(a) selecting a data point signal to be encoded, said data point being chosen in sequence from a given ordered list where some adjacent data points in said list are from different, non-adjacent, lines of a block,

(b) obtaining a set of prediction data point signals for the selected data point signal from among encoded data point signals and forming a neighborhood that is substantially centered in the two dimensions about the corresponding data point signal to be encoded,

(c) developing a predicted value for said selected data point signal in accordance with said set of prediction data point signals,

(d) developing a quantized prediction error signal from a quantized arithmetic difference between said data point signal to be encoded and said predicted value, and

(e) developing an encoded data point signal based on said predicted value of said selected data point signal and on said quantized prediction error signal, to add to said encoded data point signals,

and wherein said ordered list is such that as encoding progresses through the list each set for prediction data point signals obtained in step (b) becomes spatially closer and closer to its selected data point signal.

2. A method as claimed in claim 1 comprising the step of initially storing the data point signals of a block to be encoded in a storage element (130), and wherein

step (e) comprises the step of replacing in said storage element each selected data point signal with its corresponding encoded data point signal.

3. A method as claimed in claim 1 or 2 wherein step (c) combines said prediction point signals obtained in step (b).

4. A method as claimed in claim 1, 2 or 3 comprising the step of encoding said quantized prediction error signal in accordance with a preselected variable word length encoding method.

5. A method as claimed in any preceding claim wherein steps (a), (b), (c) and (d) are iterated for each data point signal in each of said blocks.

6. A method as claimed in claim 5 wherein step (a) adaptively progresses through said list in order, or skips over preselected data point signals based on specific characteristics of said data.



immediately followed by a CANCEL signal, then a skip can be made, and a STRT signal is generated at the output of NOR gate 82. Skip Indicator register 83 is then set to S=1, which disables further operation by decoder 89. Pixels are then decoded using zero QPE values until the END signal is generated by comparator 76, which causes register 83 to be reset to S=0. Following this, normal operation again resumes.

Upon reception of an End-of-Block signal, decoder 89 simply outputs a very large zero run-length number to zero-counter 80. This cause zero QPE values to be generated until the end of the block, at which time pixel-counter 43 generates a Start-of-Block signal resetting the system. Frequently, it is not necessary to send an End-of-Block signal, since in many cases the transmission of a few zero QPE values causes automatic skips to the end of the block.

The above-described method and implementation are illustrative of the invention and variations are, of course, possible.

For example, what has been described is an approach where there is effectively a one row and one column overlap between adjacent blocks. It should be realized that any desired overlap is possible.

The obvious tradeoff is that with greater overlap the prediction accuracy improves, but each new block either encodes fewer pixels or is larger.

Another enhancement may be to use a number of pixels other than four in developing the predicted value of a pixel, or of using other than a simple arithmetic average, e.g., median. Again, there is a tradeoff between accuracy and processing effort.

Yet another variation relates to the block list. In some applications (e.g., where the block is very large) it may be easier to develop the block list "on the fly" rather than rely on a large table. Instead of a list-ROM element one would have a processing element that would develop the required sets of prediction pixels.

Still another enhancement relates to the order in which the blocks of FIG. 1 are processed. In facsimile transmission where sequential scanning is the norm (e.g., from top to bottom and from left to right), blocks would most conveniently be encoded in a similar sequence. This is the approach employed in the above illustrative embodiments, with a pixel-RAM storing strips of data. If the pixel-RAM is large enough, however, the entire image may be stored first and the order of encoding can then be different. One could, for example, start with the block in the center of the image and work outwards, or vice versa.

Although the description above relates to encoding and decoding of image information, as has been indicated initially invention relates to the field of data compression generally. The aforesaid pixels are mere illustrative examples of data points, or values, and the method and apparatus described herein in connection with an image compression environment is equally applicable to other data compression tasks, such as

compression of speech signals.

Claims

1. A method for predictively encoding data signals divided into two-dimensional blocks containing a plurality of data point signals, CHARACTERISED BY the steps of:

(a) selecting a data point signal to be encoded, said data point being chosen in sequence from a given ordered list where some adjacent data points in said list are from different, non-adjacent, lines of a block,

(b) obtaining a set of prediction data point signals for the selected data point signal from among encoded data point signals and forming a neighborhood that is substantially centered in the two dimensions about the corresponding data point signal to be encoded,

(c) developing a predicted value for said selected data point signal in accordance with said set of prediction data point signals,

(d) developing a quantized prediction error signal from a quantized arithmetic difference between said data point signal to be encoded and said predicted value, and

(e) developing an encoded data point signal based on said predicted value of said selected data point signal and on said quantized prediction error signal, to add to said encoded data point signals,

and wherein said ordered list is such that as encoding progresses through the list each set for prediction data point signals obtained in step (b) becomes spatially closer and closer to its selected data point signal.

2. A method as claimed in claim 1 comprising the step of initially storing the data point signals of a block to be encoded in a storage element (130), and wherein

step (e) comprises the step of replacing in said storage element each selected data point signal with its corresponding encoded data point signal.

3. A method as claimed in claim 1 or 2 wherein step (c) combines said prediction point signals obtained in step (b).

4. A method as claimed in claim 1, 2 or 3 comprising the step of encoding said quantized prediction error signal in accordance with a preselected variable word length encoding method.

5. A method as claimed in any preceding claim wherein steps (a), (b), (c) and (d) are iterated for each data point signal in each of said blocks.

6. A method as claimed in claim 5 wherein step (a) adaptively progresses through said list in order, or skips over preselected data point signals based on specific characteristics of said data.



7. A method as claimed in claim 5 or 6 wherein step (a) selects data point signals in list order when said quantized prediction errors are large, and selects data point signals out of order when said quantized prediction errors are small.

8. A method as claimed in any preceding claim wherein at least one of said blocks comprises a data signal overlap with other blocks.

9. A method as claimed in claim 8 wherein said data overlap appears along one horizontal edge and one vertical edge of each of the blocks that include a data signal overlap.

10. A block list encoder for encoding data point signals by a method as claimed in claim 1, CHARACTERISED BY a data point memory (130) for storing input data point signals belonging to data blocks,

a block list memory (140) for storing a list of candidate data points in a block and storing for each candidate data point a corresponding set of prediction data point designations where each of the designated prediction data points of a candidate data point is also a candidate data point earlier in said list,

an address counter (160),
means (153) for addressing said data point memory with said address counter and said block list memory,

a predictor (120) responsive to output signals of said data point memory,

computation means (171,173) responsive to output signals of said data point memory and to said predictor output signals for developing quantized prediction error signals,

combining means (174) responsive to said computation means and to said predictor for developing coded values,

means for storing said coded values in said data point memory, and

output means (180) responsive to said computation means for delivering said quantized prediction error signals.

11. An encoder as claimed in claim 10 wherein said output means comprises an encoder for compressing said quantized prediction error signals.

12. An encoder as claimed in claim 10 wherein said output means comprises a variable word length encoder (180) and zero run length encoder (190) for compressing said quantized prediction error signals.

13. An encoder as claimed in claim 10,11 or 12 wherein said predictor comprises averaging means (123,124).

14. An encoder as claimed in any one of claims 10 to 13 comprising

means (74) for inhibiting said output means, and

means (72;73), responsive to said quantized prediction error signals, for controlling said means for inhibiting.

15. An encoder as claimed in any one of claims

10 to 13 comprising means (72,73,74) for blocking said delivering of said quantized prediction error signals for sequences of said data points, based on pre-selected thresholds of said quantized prediction error signals.

Patentansprüche

1. Verfahren zum prädiktiven Codieren von Datensignalen, die in zweidimensionale Blöcke eingeteilt sind, wobei jeder Block eine Vielzahl von Datenpunktsignalen enthält, gekennzeichnet durch folgende Schritte:

(a) Auswählen eines zu codierenden Datenpunktsignals, wobei der Datenpunkt der Reihe nach aus einer in einer gegebenen Weise geordneten Liste ausgewählt wird, in der einige benachbarte Datenpunkte stehen, die aus verschiedenen, nicht benachbarten Zellen eines Blocks sind,

(b) Gewinnen eines Satzes von Voraussagedatenpunktsignalen für das ausgewählte Datenpunktsignal aus codierten Datenpunktsignalen und Bilden einer Nachbarschaft, die im wesentlichen das entsprechende zu codierende Datenpunktsignal zweidimensional umgibt,

(c) Ermitteln eines vorausgesagten Wertes für das ausgewählte Datenpunktsignal in Übereinstimmung mit dem Satz Voraussagedatenpunktsignalen,

(d) Ermitteln eines quantisierten Voraussagefehlersignals aus einer quantisierten arithmetischen Differenz zwischen dem zu codierenden Datenpunktsignal und dem vorausgesagten Wert und

(e) Ermitteln eines codierten Datenpunktsignals auf der Grundlage des vorausgesagten Wertes des ausgewählten Datenpunktsignals und auf der Grundlage des quantisierten Voraussagefehlersignals, das zu den codierten Datenpunktsignalen addiert wird, und ferner dadurch gekennzeichnet,

daß die geordnete Liste dergestalt ist, daß, solange das Codieren in der Liste fortgesetzt wird, sich jeder bei dem Schritt (b) gewonnene Satz von Voraussagedatenpunktsignalen räumlich mehr und mehr seinem ausgewählten Datenpunktsignal nähert.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet,

daß die Datenpunktsignale eines zu codierenden Blocks zuerst in einem Speicherelement (130) gespeichert werden und

daß bei dem Schritt (e) jedes ausgewählte Datenpunktsignal in dem Speicherelement durch sein entsprechendes, codiertes Datenpunktsignal ersetzt wird.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet,



daß im Verfahrensschritt (c) die bei Schritt (b) gewonnenen Voraussagedatenpunktsignale kombiniert werden.

4. Verfahren nach Anspruch 1, 2 oder 3, gekennzeichnet durch

Codieren des quantisierten Voraussagefehlersignals in Übereinstimmung mit einem Verfahren zum Codieren einer vorher ausgewählten variablen Wortlänge.

5. Verfahren nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, daß die Schritte (a), (b), (c) und (d) für jedes Datenpunktsignal in jedem Block wiederholt werden.

6. Verfahren nach Anspruch 5, dadurch gekennzeichnet,

daß die Anwendung des Schrittes (a) durch die Listenreihenfolge bestimmt wird oder auf der Grundlage spezifischer Eigenschaften der Daten zu vorher ausgewählten Datenpunktsignalen übergeht.

7. Verfahren nach Anspruch 5 oder 6, dadurch gekennzeichnet,

daß bei Schritt (a) Datenpunktsignale in der Reihenfolge der Liste ausgewählt werden, wenn die quantisierten Voraussagefehler groß sind, und daß Datenpunktsignale außerhalb der Reihenfolge ausgewählt werden, wenn die quantisierten Voraussagefehler klein sind.

8. Verfahren nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet,

daß wenigstens ein Block eine Datensignalüberlappung mit anderer Blöcke aufweist.

9. Verfahren nach Anspruch 8, dadurch gekennzeichnet,

daß die Datenüberlappung entlang eines horizontalen Randes und eines vertikalen Randes eines jeden Blocks, der eine Datenüberlappung einschließt, auftritt.

10. Blocklistencodierer zum Codieren von Datenpunktsignalen nach einem Verfahren gemäß Anspruch 1, gekennzeichnet durch einen Datenpunktspeicher (130) zum Speichern von Eingangsdatenpunktsignalen, die zu Datenblöcken gehören,

einen Blocklistenspeicher (140), der eine Liste von Kandidatendatenpunkten in einem Block speichert und für jeden Kandidatendatenpunkt einen entsprechenden Satz gekennzeichnete Voraussagedatenpunkte speichert, wobei jeder gekennzeichnete Voraussagedatenpunkt eines Kandidatendatenpunktes ebenso ein früherer Kandidatendatenpunkt in der Liste ist,

einen Adressenzähler (160),

eine Einrichtung (153) zum Adressieren des Datenpunktspeichers mit Hilfe des Adressenzählers und des Blocklistenspeichers,

einen Prädiktor (120), der auf die Ausgangssignale des Datenpunktspeichers anspricht,

eine Berechnungseinrichtung (171, 173), die unter Ansprechen auf die Ausgangssignale des Datenpunktspeichers und die Ausgangssignale des

Prädiktors quantisierte Voraussagefehlersignale erzeugt,

eine Kombinationseinrichtung (174), die unter Ansprechen auf die Berechnungseinrichtung und den Prädiktor codierte Werte ermittelt, eine Einrichtung zum Speichern der codierten Werte in dem Datenpunktspeicher und

eine Ausgangseinrichtung (180), die unter Ansprechen auf die Berechnungseinrichtung die quantisierten Voraussagefehlersignale abgibt.

11. Codierer nach Anspruch 10, dadurch gekennzeichnet,

daß die Ausgangseinrichtung einen Codierer (118) zum Komprimieren der quantisierten Voraussagefehlersignale aufweist.

12. Codierer nach Anspruch 10, dadurch gekennzeichnet,

daß die Ausgangseinrichtung einen Codierer (180) für eine variable Wortlänge und einen Codierer (190) für eine Null-Lauflänge zum Komprimieren der quantisierten Voraussagefehlersignale enthält.

13. Codierer nach Anspruch 10, 11 oder 12, dadurch gekennzeichnet,

daß der Prädiktor eine Einrichtung (123, 124) zum Bilden des Mittelwertes aufweist.

14. Codierer nach einem der Ansprüche 10 bis 13, gekennzeichnet durch

eine Einrichtung (74) zum Blockieren der Ausgangseinrichtung, und

eine Einrichtung (72; 73), die unter Ansprechen auf die quantisierten Voraussagefehlersignale die Blockiereinrichtung steuert.

15. Codierer nach einem der Ansprüche 10 bis 13, gekennzeichnet durch

eine Einrichtung (72, 73, 74), die das Abgeben der quantisierten Voraussagefehlersignale für Folgen von Datenpunkten auf der Grundlage von vorher ausgewählten Schwellwerten der quantisierten Voraussagefehlersignale blockiert.

Revendications

1. Procédé pour coder de manière prédictive des signaux de données divisés en blocs bidimensionnels contenant une pluralité de signaux de point de données, caractérisé par les étapes suivantes :

(a) sélection d'un signal de point de données à encoder, ledit point de données étant choisi en séquence dans une liste ordonnée donnée dans laquelle certains points de données adjacents proviennent de lignes non adjacentes différentes d'un bloc,

(b) obtention d'un ensemble de signaux de points de données de prédiction pour le signal de point de données sélectionné parmi des signaux de points de données encodés et formation d'un voisinage qui est essentiellement centré dans les



deux dimensions autour du signal de point de données correspondant à encoder,

(c) développement d'une valeur prédictive pour le signal de point de données sélectionné d'après ledit ensemble de signaux de points de données de prédiction,

(d) développement d'un signal d'erreur de prédiction quantifié à partir d'une différence arithmétique quantifiée entre le signal de point de données précité à encoder et ladite valeur prédite, et

(e) développement d'un signal de point de données encodé basé sur ladite valeur prédite du signal de point de données sélectionné et sur le signal d'erreur de prédiction quantifié, pour l'additionner auxdits signaux de points de données encodés,

et dans lequel la liste ordonnée est telle que, à mesure que l'encodage progresse dans la liste, chaque ensemble de signaux de points de données de prédiction obtenus au cours de l'étape (b) devient spatialement de plus en plus rapproché de son signal de point de données sélectionné.

2. Procédé selon la revendication 1, comprenant l'étape consistant à emmagasiner initialement les signaux de points de données d'un bloc à encoder dans un élément de mémoire (130), et dans lequel l'étape (e) comprend l'étape consistant à remplacer dans ledit élément de mémoire chaque signal de point de données sélectionné par son signal de point de données encodé.

3. Procédé selon la revendication 1 ou 2, dans lequel l'étape (c) combine lesdits signaux de points de données de prédiction obtenus au cours de l'étape (b).

4. Procédé selon la revendication 1, 2 ou 3, comprenant l'étape consistant à encoder le signal d'erreur de prédiction quantifié d'après un procédé d'encodage de longueur de mot variable présélectionné.

5. Procédé selon l'une quelconque des revendications précédentes, dans lequel les étapes (a), (b), (c) et (d) sont répétées pour chaque signal de point de données dans chacun desdits blocs.

6. Procédé selon la revendication 5, dans lequel l'étape (a) progresse de manière adaptative dans la liste précitée dans l'ordre ou saute des signaux de points de données présélectionnés sur base de caractéristiques spécifiques de ces données.

7. Procédé selon la revendication 5 ou 6, dans lequel l'étape (a) sélectionne des signaux de points de données dans l'ordre de la liste lorsque les erreurs de prédiction quantifiées précitées sont importantes, et sélectionne des signaux de points de données dans le désordre lorsque les erreurs de prédiction quantifiées sont faibles.

8. Procédé selon l'une quelconque des revendications précédentes, dans lequel au moins un des blocs présente un recouvrement de signal de don-

nées avec d'autres blocs.

9. Procédé selon la revendication 8, dans lequel ledit recouvrement de données apparaît le long d'un bord horizontal et d'un bord vertical de chacun des blocs qui présentent un recouvrement de signaux de données.

10. Codeur de liste de blocs pour encoder des signaux de points de données par un procédé selon la revendication 1, caractérisé par une mémoire de points de données (130) pour emmagasiner des signaux de points de données d'entrée appartenant à des blocs de données,

une mémoire de liste de blocs (140) pour emmagasiner une liste de points de données candidats dans un bloc et, pour chaque point de données candidat, pour emmagasiner un ensemble correspondant de désignations de points de données de prédiction dans lequel chacun des points de données de prédiction désigné d'un point de données candidat est également un point de données candidat précédent dans la liste précitée,

un compteur d'adresse (160),

des moyens (153) pour adresser ladite mémoire de points de données avec le compteur d'adresse et la mémoire de liste de blocs,

un prédicteur (120) répondant aux signaux de sortie de la mémoire de points de données,

des moyens de calcul (171, 173) répondant aux signaux de sortie de la mémoire de points de données et aux signaux de sortie du prédicteur pour développer des signaux d'erreur de prédiction quantifiés,

des moyens de combinaison (174) répondant aux moyens de calcul et au prédicteur pour développer des valeurs codées,

des moyens pour emmagasiner les valeurs codées dans la mémoire de points de données, et

des moyens de sortie (180) répondant aux moyens de calcul pour délivrer les signaux d'erreur de prédiction quantifiés précités.

11. Codeur selon la revendication 10, dans lequel les moyens de sortie comprennent un codeur pour comprimer les signaux d'erreur de prédiction quantifiés.

12. Codeur selon la revendication 10, dans lequel les moyens de sortie comprennent un codeur à longueur de mot variable (180) et un codeur de longueur de traitement de zéros (190) pour comprimer les signaux d'erreur de prédiction quantifiés.

13. Codeur selon la revendication 10, 11 ou 12, dans lequel le prédicteur comprend un dispositif de moyenne (123, 124).

14. Codeur selon l'une quelconque des revendications 10 à 13, comprenant un moyen (74) pour inhiber les moyens de sortie précités, et un moyen (72, 73) répondant aux signaux d'erreur de prédiction quantifiés pour commander ledit moyen d'inhibition.

15. Codeur selon l'une quelconque des revendi-



cation 10 à 13, comprenant des moyens (72, 73, 74) pour bloquer la délivrance des signaux d'erreur de prédiction quantifiés pour des suites des points de données précités, sur base de seuils présélectionnés des signaux d'erreurs de prédiction quantifiés.

5

10

15

20

25

30

35

40

45

50

55

11