CCITT SGXV
Working Party XV/1
Specialist Group on Coding
for Visual Telephony

Doc # 200
March 1987

TITLE: DIFFERENTIAL MOTION VECTORS 1.

SOURCE: UK

OBJECTIVE.

The merits of using a differential motion vector scheme within the 'Okubo Reference Model 3' are examined. This is compared with an absolute value scheme. System aspects of the two schemes are discussed.

DIFFERENTIAL VECTORS.

Within the hardware specification document [1] the use of differential motion vectors is detailed but not fully specified. An earlier specialist group document [2] describes the study of differential motion vectors within the 'Okubo Reference Model 2'. Unfortunately the exact nature of the differential vectors in [2] is not described, it is however assumed that they are as described herein.

It is conceivable to have several different types of differential vectors. One type (Type 1) can be generated by taking the differences between the current and previous absolute motion vectors. A second (Type 2) could be generated by taking the difference between the current vector and the last differential vector. Each 'absolute' vector is composed of 2 orthogonal components representing horizontal and vertical motion. In the following text the meaning of 'vector' is used generally with reference to the orthogonal components. Figure 1 illustrates the principle. The blocks represent the sub-blocks of 8*8 pels within an image and example motion vectors are annotated within these blocks. For simplicity only one row of blocks is shown. It is assumed that a fictitious zero valued motion vector exists to the left outside the image frame and that the differential vectors are calculated on a per row of blocks basis. In the case where an absolute motion vector has zero horizontal and vertical components then the differential vector is not transmitted. If a non-zero absolute vector follows a zero absolute vector the differential is calculated as if the previous vector was zero ie the differential vector is the same as the absolute vector.

Only the Type 1 scheme is examined here.

Once the differential vectors have been generated several scenarios exist for the coding of the vectors.

1] A fixed length code can be used. This is used as a reference for comparison purposes.

2] A single variable length code can be used to code both horizontal and vertical vectors separately.

3] Two variable length codes can be used to code horizontal and vertical vectors.

4] A single two dimensional variable length code can be used to code both horizontal and vertical vectors.

Cases 1 and 2 are examined here. Cases 3 and 4 have been examined [2] and been shown to yield only small gains over case 2.

Assuming that the absolute motion vectors have a dynamic range of +/-7 then the differential vectors have a dynamic range of +/- 14. It is possible to modify the requirements of the differential vector coding by using the knowledge that any absolute vector must be with the range of +/- 7. This modified coding is known here as truncated differential vector coding (TDVC) (first illustrated in [2]). For example if the previous absolute vector is +7 and the current absolute vector is -7 then the differential is -14. If one assumes that the maximum differential transmitted is +/- 7 then if one were to transmit +7 (which is the inverted absolute vector) in this case the decoder would normally calculate a the current vector as +14. However it is known that the maximum absolute vector is +/7 thus the decoder may re-calculate the vector by inverting the sign of the apparent differential and use this value as the absolute value of the current vector. A few more examples illustrates the mechanism.

Previous vector	Current vector	Diff. vector	tx/rx vector	decoded vector	decoded and modified
7	-7	-14	7	+14	-7
-7	6	+13	-6	-13	+6
-6	7	+13	-7	-13	+7

RESULTS

The 'Costi' and 'Miss America' sequences were used to generate results for this document. The results shown represent an average result using the data indicated.

The ultimate performance which can be attained by coding the vectors may be measured by the entropy of the data. Table 1 indicates the entropy of the absolute vectors, differential vectors and the TDVs for the source data used. Careful attention was given to the data generation in the case of TDVC in that the 'tx/rx vector' was used for entropy purposes.

Figures 2 to 7 illustrate the distribution of the use of absolute vectors and differential vectors for the sequences used. The graphs represent the composite of the results over the frame range indicated.

The entropy measures shown in Table 1 can usually never be attained with 'real codes' in practice and therefore some estimate of the attainable bit usage per vector is now given. This is calculated as:

$$\text{Bits required for } V(n) = INT(-\log_2 (P(n)) + 0.5)$$

Where V(n) is the vector 'n'.
P(n) is the probability of V(n) occuring
INT(X) implies integer of (X).

Whilst this technique for bit allocation is not truely optimal it is thought to be sufficiently accurate for guidance purposes. For each case examined a bit allocation was generated and a subsequent mean word length for the vectors was calculated. The mean word length (MWL) was calculated using:

$$MWL = P(n)*INT(-\log_2 (P(n)) + 0.5)$$

Where V(n) is the vector 'n'.
P(n) is the probability of V(n) occuring
INT(X) implies integer of (X).

Table 2 illustrates the bit allocation for the Split Screen segment of 'Costi' for TDVC. Table 3 illustrates the bit allocation for the Split Screen segment of 'Costi' for absolute vectors. Table 4 illustrates the MWLs determined as above.

DISCUSSION

In general the results illustrated here are similar to the results of [2].

The distributions of Figures 2 to 7 all exhibit some peak towards the zero vector value, therefore one could reasonably expect to obtain some gain over fixed length coding of the vectors. No distributions of the TDVC method have been included, however the distributions are very similar to Figures 5 - 7 (but vectors truncated to +/- 7). The distributions of the absolute vectors (Figs 2 - 4) exhibit an unexpected peak at motion vector values of +/- 7. This is reflected in the bit allocation table (table 3) for the absolute vectors where vectors +/- 7 are allocated less bits than say vectors +/- 6. The results illustrated in [2] do not exhibit this feature. The peak may result from the motion search being limited to +/- 7 and the occurance of the vector |7| representing the cumulative total of true vector occurrences of |7| and greater.

The entropy values in Table 1 all indicate that on average the vectors theoretically can be coded with less than 4 bits. The difference between the entropies of the absolute and differential methods are small, independent of the source material. This small difference suggests that some criterion other than simple bit efficiency should be used to choose between these methods. The TDVC method is consistently more efficient than the absolute or differential method, however the gain over a fixed bit allocation of 4 bits/vector is still small.

Tables 2 and 3 illustrate bit allocations assigned to the motion vectors using absolute vectors and TDVC for the 'Split Screen' segment of the 'Costi' sequence. The allocation of bits is very similar to those presented in [2].

Table 4 illustrates the average word length of the vectors used. These figures allow a comparison of using an optimal vlc as opposed to a fixed length code. Compared to Table 1 after appropriate bit allocation, the mean word length has increased by about 13%. This increase seems reasonable. In all cases but the TDVC the use of a vlc is worse than a fixed length allocation. In the case of TDVC the gain over fixed length coding is very small. This would typically save about 100 - 200 bits per frame.

It is clear that the overall efficiency of using relative motion vectors within the 'Reference Model 3' yields only a small gain in coding efficiency (approx 0.5%). The gain in efficiency is likely to be a function of the motion estimation algorithm used. In particular, if some type of object based algorithm were to be used then many similarly valued vectors may be generated. In this case a differential method is likely to be significantly superior to an absolute scheme. Further work may be done here. It may also be conceivable that some form of vector post-processing may occur such that the resulting 'motion field' would be much smoother than the original estimated 'motion field'. Then the resulting differential vectors are likely to have small entropies. If the source data were to contain a significant amount of movement which could be described by pure translation (eg panning) then a differential scheme is likely to be superior to an absolute scheme. Although there is provision within the evolving hardware specification [1] to include the concept of global motion vectors to overcome the problems of panning motion, as yet no proposal has been made on how to generate this global motion vector.

The primary disadvantage of differential motion vectors would appear to be one of error propagation. If an error occurs during transmission and is not detected or corrected then that error can affect all subsequent motion vectors on the same row of blocks. This error can then propagate spatially in the following frames of data. The result of the process is however somewhat speculative and a proper study is required to make any quantitative evaluation.

A further though minor disadvantage would be the additional hardware required to generate and decode the differential vectors.

CONCLUSION

The current specification allows the provision of differential vectors of two types. The first is a differential with respect to two absolute block vectors. The second is a differential with respect to a global vector and a block absolute vector. If the global vector is zero then the specification has in effect provision to code the motion vectors in a differential or absolute manner. The problems of error propagation can then be investigated more fully in realistic conditions.

Therefore the current recommendation in [1] should remain until further more substantive evidence can be provided.

REFERENCES

[1].......Doc #182 CCITT WP XV/1 Specialist Group Meeting on Video telephony. Nov 86 Nurnberg.

[2].......Doc #153 CCITT WP XV/1 Specialist Group Meeting on Video telephony. Nov 86 Nurnberg.

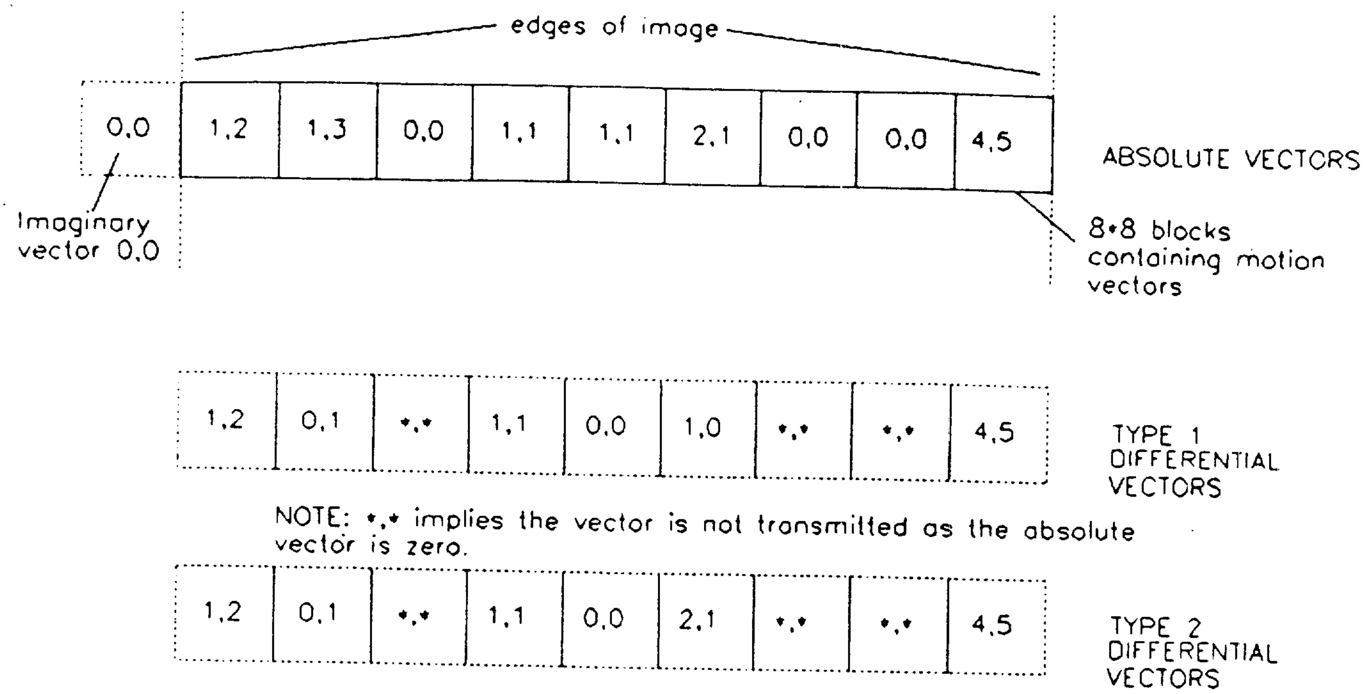

FIGURE 1. EXAMPLE OF ABSOLUTE AND DIFFERENTIAL VECTORS

VECTOR	BITS NEEDED	BITS ALLOCATED
−7	4.84	5
−6	5.41	5
−5	5.31	5
−4	4.7	5
−3	4.45	4
−2	3.75	4
−1	2.77	3
0	1.89	2
1	2.97	3
2	3.98	4
3	4.52	5
4	4.89	5
5	5.27	5
6	5.53	6
7	5.18	5

TABLE 2 BIT ALLOCATION FOR SPLIT SCEEN SEGMENT USING TDVC

		ENTROPY (BITS)		
SEQUENCE	CODED FRAMES	ABSOLUTE VECTORS	DIFFERENTIAL VECTORS	TDV
SPLIT SCREEN	2 −19	3.65	3.64	3.4
TREVOR	21 −48	3.72	3.68	3.45
MISSA	ALL	3.57	3.52	3.29

TABLE 1 ENTROPY OF VECTORS

VECTOR	BITS NEEDED	BITS ALLOCATED
−7	4.45	4
−6	5.88	6
−5	5.07	5
−4	4.49	4
−3	4.01	4
−2	3.56	4
−1	2.74	3
0	2.64	3
1	3.09	3
2	4.04	4
3	4.18	4
4	4.48	4
5	4.67	5
6	5.08	5
7	4.10	4

TABLE 3 BIT ALLOCATION FOR SPLIT SCEEN SEGMENT USING ABSOLUTE VECTORS

		MEAN WORD LENGTH (BITS)		
SEQUENCE	CODED FRAMES	ABSOLUTE VECTORS	DIFFERENTIAL VECTORS	TDV
SPLIT SCREEN	2 −19	4.15	4.14	3.9
TREVOR	21 −48	4.18	4.22	3.95
MISSA	ALL	4.07	4.02	3.79

TABLE 4 MEAN WORD LENGTHS OF VECTOR CODES

FIGURE: 2

DATE: 28/1/87
SOURCE: BTRL UK
AUTHOR: G SEXTON

CODER SOFTWARE: CODEC6 V1.2 (OKUBO REFERENCE MODEL 3)

MODIFICATIONS:
SOURCE SEQUENCE: COSTI CIF. SPLIT SCREEN SEGMENT
FRAME SIZE: 352 * 288
BITRATE: 300KBITS/SEC
BUFFER SIZE: 30000

FRAME RATE: 10 FRAMES / S
CODED FRAMES: 2 - 19

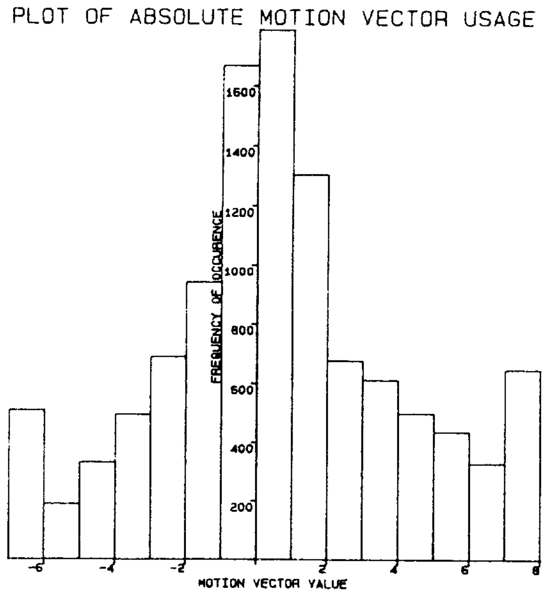

FIGURE: 3

DATE: 28/1/87
SOURCE: BTRL UK
AUTHOR: G SEXTON

CODER SOFTWARE: CODEC6 V1.2 (OKUBO REFERENCE MODEL 3)

MODIFICATIONS:
SOURCE SEQUENCE: COSTI CIF. TREVOR SEGMENT
FRAME SIZE: 352 * 288
BITRATE: 300KBITS/SEC
BUFFER SIZE: 30000

FRAME RATE: 10 FRAMES / S
CODED FRAMES: 21 - 48

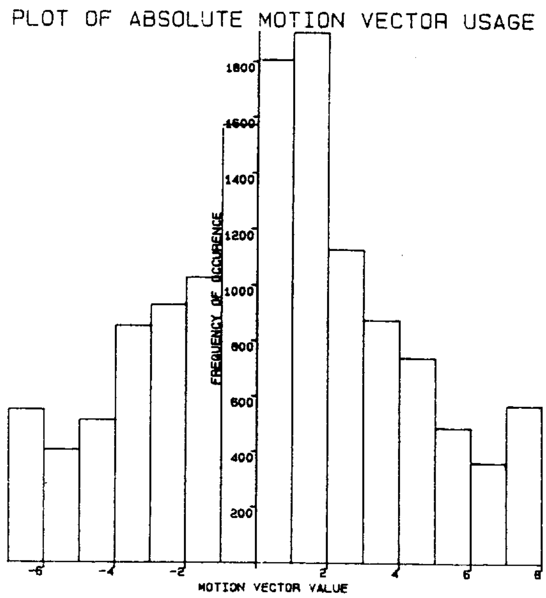

FIGURE: 4

DATE: 28/1/87
SOURCE: BTRL UK
AUTHOR: G SEXTON

CODER SOFTWARE: CODEC6 V1.2 (OKUBO REFERENCE MODEL 3)

MODIFICATIONS:
SOURCE SEQUENCE: MISSA

FRAME SIZE: 352 * 288
BITRATE: 300KBITS/SEC
BUFFER SIZE: 30000

FRAME RATE: 10 FRAMES / S
CODED FRAMES: 2 - 38

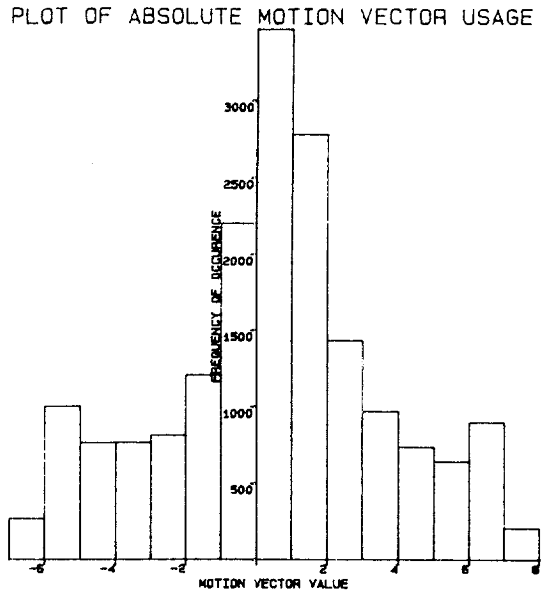

FIGURE: 5

DATE: 28/1/87
SOURCE: BTRL UK
AUTHOR: G SEXTON

CODER SOFTWARE: CODEC6 V1.2 (OKUBO REFERENCE MODEL 3)

MODIFICATIONS:
SOURCE SEQUENCE: COSTI: SPLIT SCREEN SEGMENT

FRAME SIZE: 352 * 288
BITRATE: 300KBITS/SEC
BUFFER SIZE: 30000

FRAME RATE: 10 FRAMES / S
CODED FRAMES: 2 - 19

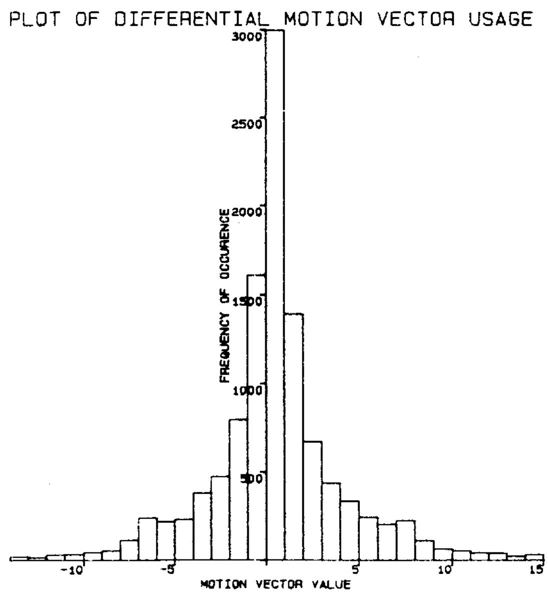

FIGURE: 6

DATE:	28/1/87
SOURCE:	BTRL UK
AUTHOR:	G SEXTON
CODER SOFTWARE:	CODEC6 V1.2 (OKUBO REFERENCE MODEL 3)
MODIFICATIONS:	
SOURCE SEQUENCE:	COSTI: TREVOR SEGMENT
FRAME SIZE:	352 * 288
BITRATE:	300KBITS/SEC
BUFFER SIZE:	30000
FRAME RATE:	10 FRAMES / S
CODED FRAMES:	21 - 48

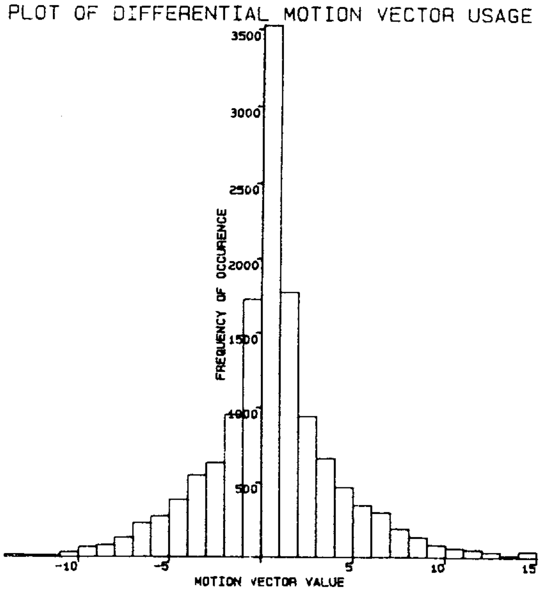

FIGURE: 7

DATE:	28/1/87
SOURCE:	BTRL UK
AUTHOR:	G SEXTON
CODER SOFTWARE:	CODEC6 V1.2 (OKUBO REFERENCE MODEL 3)
MODIFICATIONS:	
SOURCE SEQUENCE:	MISSA
FRAME SIZE:	352 * 288
BITRATE:	300KBITS/SEC
BUFFER SIZE:	30000
FRAME RATE:	10 FRAMES / S
CODED FRAMES:	2 - 38