Hill, P. R., Chiew, T-K., Bull, D. R., & Canagarajah, C. N. (2006). Interpolation free subpixel accuracy motion estimation. IEEE Transactions on Circuits and Systems for Video Technology, 16(12), 1519 - 1526. 10.1109/TCSVT.2006.885722

Link to published version (if available):
10.1109/TCSVT.2006.885722

Transactions Letters

Interpolation Free Subpixel Accuracy Motion Estimation

P. R. Hill, T. K. Chiew, D. R. Bull, and C. N. Canagarajah

***Abstract*—Subpixel motion estimation plays an important role in compression efficiency within modern video codecs such as MPEG2, MPEG4, and H.264. Subpixel motion estimation is implemented within these standards using interpolated values at 1/2 or 1/4 pixel accuracy. Such interpolation gives a good reduction in residual energy for each predicted macroblock and, therefore, improves compression. However, this leads to a significant increase in computational complexity at the encoder. This is especially true for H.264 where the cost of an exhaustive set of macroblock segmentations need to be estimated in order to obtain an optimal mode for prediction. This paper presents a novel interpolation-free scheme for subpixel motion estimation using the result of the full pixel sum of absolute difference distribution of each motion compensated block applied to an H.264 encoder. This system produces reduced complexity motion estimation with a controllable tradeoff between compression performance and encoder speed. These methods facilitate the generation of a real time software H.264 encoder.**

***Index Terms*—Motion estimation, video coding.**

I. Introduction

HYBRID video compression algorithms rely on block based motion compensation in order to reduce temporal redundancy and, therefore, facilitate compression. Although whole pixel motion compensation provides a degree of redundancy reduction, it has been found that motion compensation with subpixel accuracy progressively reduces spatial redundancy as the level of subpixel accuracy increases. This is clearly shown in Fig. 1 where the displaced frame difference (DFD) residual for a frame within the foreman test sequence has progressively smaller energy as subpixel accuracy increases. This is also reflected in the rate-distortion performance of an encoder. Figs. 5–7 show the considerable difference in rate-distortion performance between whole pixel and quarter pixel motion compensation in an H.264 encoder.

Subpixel accuracy for motion estimation is traditionally made possible using interpolated reference frames. The creation and use of such interpolated reference frames has a significant implication on the computational load of the encoder and memory bandwidth requirements. For example, Table I shows the memory requirements of a 4:2:0 reference frame for 1/2 and 1/4 subpixel interpolation.

Manuscript received April 25, 2005; revised November 1, 2005. This paper was recommended by Associate Editor I. Ahmad.

The authors are with the Department of Electrical and Electronic Engineering, The University of Bristol, Bristol BS5 1UB, U.K. (e-mail: paul.hill@bristol.ac.uk; dave.bull@bristol.ac.uk; nishan.canagarajah@bristol.ac.uk).

Digital Object Identifier 10.1109/TCSVT.2006.885722

The use of multiple reference frames within the H.264 encoder offers enhanced prediction (resulting in better compression) and error resilience. The use of multiple reference frames will obviously have a considerable impact on memory requirements when using interpolated reference frames. This paper introduces an interpolation-free method for subpixel block based motion estimation which reduces memory bandwidth requirements and improves computational efficiency. This will facilitate real time encoding in situations with limited memory and memory bandwidth.

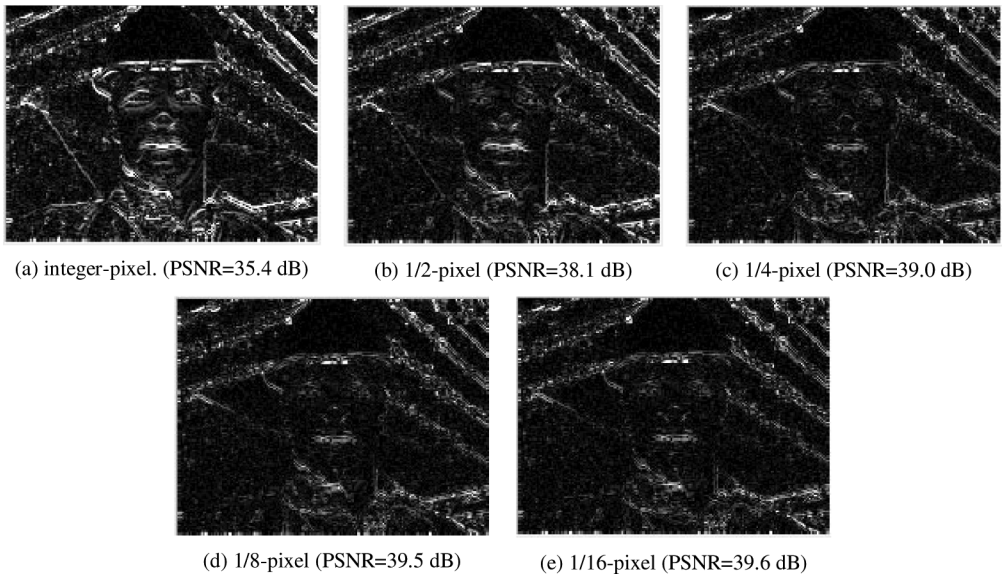
In Section II of this paper, a parabolic model of the subpixel resolution motion estimation cost is described that uses the sum of absolute difference (SAD) cost at the best whole pixel resolution position and its neighbors. Section III describes the generation of the parameters for the model described in Section II using whole pixel resolution results. Once the parameters for the model are generated, the minimum value of the model is estimated using the techniques described in Section IV. This method, when used in a simple and direct way, was found to give worse results (in a rate-distortion sense) than the fully interpolated case. In order to improve the location of the actual minimum, an interpolated fall-back was developed as shown in Section V. The check used for the fallback mechanism is described in Section V-A supported by the results given for the two fallback checks given in Section VI. A comparison of all the presented techniques for three sequences is presented in Section VII. Finally, conclusions and a summary of the developed methods are provided in Section VIII.

II. Model Description

In most standard video encoders, the motion vector field is estimated in a coarse to fine fashion. The coarse estimation is obtained at integer pixel displacements using standard block matching algorithms, i.e., the frame $r(\mathbf{z})$ is tessellated into nonoverlapping blocks (macroblocks or segmented macroblocks) and the best match between these blocks and a reference block is generated, thus producing a motion vector for each block. The coarse motion vector field is generated by minimizing a cost function defined and evaluated at each integer displacement $\boldsymbol{\zeta}$.[1] The cost function adopted (as is most common) is the SAD defined as

$$\mathrm{SAD}_i(\boldsymbol{\zeta}) = \sum_{\mathbf{z}\in B_i} |r(\mathbf{z}) - s(\mathbf{z}+\boldsymbol{\zeta})| \tag{1}$$

[1]The proposed system actually employs a more complex multihexagonal integer search pattern [1] and then evaluates the integer cost function at all neighbors surrounding the minimum value.



(a) integer-pixel. (PSNR=35.4 dB) (b) 1/2-pixel (PSNR=38.1 dB) (c) 1/4-pixel (PSNR=39.0 dB)

(d) 1/8-pixel (PSNR=39.5 dB) (e) 1/16-pixel (PSNR=39.6 dB)

Fig. 1. Illustration of the benefits of subpixel motion estimation. There is a gradual reduction in the high-energy pixels in the DFD as subpixel resolution increases, as quantified by the PSNR.

where $r(\mathbf{z})$ is the current frame block, $s(\mathbf{z})$ is the corresponding block in the reference frame, $\boldsymbol{\zeta}$ is the displacement between the current frame block and the reference block, B_i stands for the block extent in the neighborhood of $\mathbf{z}$, and i is the block number.

The SAD cost function has been shown to give near optimal efficiency versus signal-to-noise ratio (SNR) values, in comparison to alternative cost functions (sum of squared difference, multiply and add correlators, etc.) [2], [3]. A transformed SAD has also been used as a cost function [4]. The transformation used is often the Hadamard transform as it is relatively easy to compute and gives good results [4]. While such transformed cost functions will increase complexity they could be used in conjunction with our methods to produce a better rate-distortion result.

For the fine estimation, instead of using direct interpolation from the reference frame pixel values, our method uses a 2-D parabolic interpolation between the estimated ambiguity samples, approximating the actual behavior in the presence of pure translations [5], [6] in order to fit the continuous domain uncertainly ellipsoid of the ambiguity function [7].

A parametrically controlled parabolic surface is used to estimate the subpixel SAD values and is defined by [8]

$$\mathrm{SAD}_i(\zeta) \approx A\zeta_x^2 + B\zeta_y^2 + C\zeta_x\zeta_x + D\zeta_x + E\zeta_y + F \quad (2)$$

where $\mathrm{SAD}_i(\zeta)$ is the estimated SAD value of the ith block. The ζ_x and ζ_y values are the coordinates of the estimation, centred at the best motion vector at whole-pixel resolution (they vary from 1.0 to -1.0) with the coordinate (0,0) being the best motion vector at whole-pixel resolution. Therefore, for the quarter-pixel case used in our experimental H.264 encoder, the parameters ζ_x and ζ_y take the values $[-1, -3/4, -1/2, -1/4, 0, 1/4, 1/2, 3/4, 1]$. This model is reasonable for any stationary 2-D signal and extremely close to the actual interpolation surface for sources with Gaussian-shaped autocorrelation functions (ACFs) (see appendix).

After the motion vector at whole-pixel resolution is obtained, the SAD values of its nearest neighbors are made available (calculated or retrieved) giving the eight nearest SAD neighbors from which the parameters A, B, C, D, E, and F can be estimated. These neighbors are shown in Fig. 2. For the subsequently described techniques, the 8 neighbors are either labelled near neighbors (with even indices) or far neighbors (odd indices).

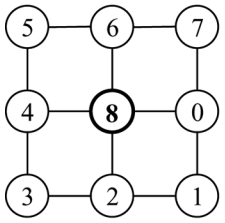
There are nine 9 potential points for the estimation of the 6 parameters within (2). The system is, therefore, overdescribed and there are, therefore, many possible estimation methods. Chiew and Bull [9], [10] presented three models for this parameter estimation. The first method used an under-determined model based just on the near neighbors [defined as the near-neighbors model (NNM)]. A second method used an overcomplete system model (OSM) that used all the 9 neighbor points and a pseudo-matrix inverse method for obtaining A-F. These methods proved to give inferior results to our chosen method: the complete-system model (CSM).

III. CSM for Parabolic Parameter Estimation

The CSM parameter estimation model has been found to be the most efficient computationally and in terms of compression [9], [10]. In this model, the parameters A, B, D, E and F are calculated from (3) using (4). Equation (3) is derived from the simple insertion of neighbor values into (2) shown in Fig. 2. The value of C is firstly set to zero in the under-determined case (NNM). However, within the CSM method the value of C is chosen from the set C_1, C_3, C_5, C_7 where C_k is the value of C found with the complete system of equations using points 0, 2, 4, 6, 8 and k (as defined in Fig. 2). Then the value of C is chosen using (5), where $\hat{S}_{ki}$ is the estimate of S_i using (2) with parameter set A, B, C_k, D, E, F. This model determines which of the far-neighbors best fits the model and ignores the other 3, thus removing the effect of outliers. Its complexity is similar to

TABLE I
MEMORY REQUIREMENTS FOR INTERPOLATED REFERENCE FRAMES

	Original Picture	$^1/_2$-pel interpolated Picture	$^1/_4$-pel interpolated Picture
QCIF	38 kB	152 kB	608 kB
CIF	152 kB	608 kB	2.43 MB



Pixel co-ordinates with respect to centre pixel:
Pixel 8: (0,0) Pixel 2: (0,1) Pixel 5: (-1,-1)
Pixel 0: (1,0) Pixel 3: (-1,1) Pixel 6: (0,-1)
Pixel 1: (1,1) Pixel 4: (-1,0) Pixel 7: (1,-1)

Centre pixel: Pixel 8
Near neighbours: Pixels 0, 2, 4, 6
Far neighbours: Pixels 1, 3, 5, 7

Fig. 2. Illustration of position of neighbor pixels.

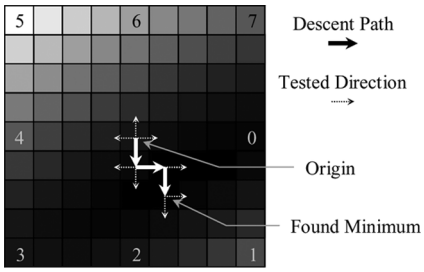


Fig. 3. Location of the parabolic function minimum.

that of the NNM method whilst also guaranteeing the existence of a minimum point

$$
\begin{aligned}
S_0 &= S(1,0) = A + D + F \\
S_1 &= S(1,1) = A + B + C + D + E + F \\
S_2 &= S(0,1) = B + E + F \\
S_3 &= S(-1,1) = A + B - C - D + E + F \\
S_4 &= S(-1,0) = A - D + F \\
S_5 &= S(-1,-1) = A + B + C - D + E + F \\
S_6 &= S(0,-1) = B - E + F \\
S_7 &= S(1,-1) = A + B - C + D - E + F \\
S_8 &= S(0,0) = F
\end{aligned} \tag{3}
$$

$$
A = -S_8 + \frac{1}{2}(S_0 + S_4); \quad B = -S_8 + \frac{1}{2}(S_2 + S_6)
$$

$$
D = \frac{1}{2}(S_0 - S_4); \quad E = \frac{1}{2}(S_2 - S_6); \quad F = S_8 \tag{4}
$$

$$
C = \underset{k=1,3,5,7}{\operatorname{argmin}} \sum_{i=1,3,5,7} |S_i - \hat{S}_{ki}|. \tag{5}
$$

IV. OBTAINING THE PARABOLIC SURFACE MINIMUM (FOR 1/4 PIXEL RESOLUTION)

A parabolic surface (i.e., second-order polynomial) as defined in (2) will only have one minimum in continuous space. However, this minimum is not guaranteed to be within the $(-1,1) \times (-1,1)$ area. Therefore, analytical methods are not appropriate. The technique adopted in [9] and [10] is to evaluate the value of $S(x,y)$ for all subpixel locations within the $(-1,1) \times (-1,1)$ area. This achieves the desired result but has an associated high computational complexity due to its exhaustive search and the large number of multiplications associated with (2). The global effect on computational complexity is compounded as the quarter pixel motion estimation is one of the most used computational modules within the encoder.

We now present three methods for calculating the minimum of the parabolic surface without an exhaustive search.

Algorithm 1 Find Parabolic Surface Minimum

1: **Define** origin **o** to be best result from whole pixel motion estimation
2: **Define** the 4-connected near neighbours of the origin **o**
3: **Discount** any 4-connected near neighbours that have been estimated before or are outside the (-1,1)(-1,1) area
4: **Calculate** $S()$ for the origin **o** and its non-discounted 4-connected near neighbours
5: **Set** $m4$ to the minimum $S()$ value for all the non-discounted 4-connected near neighbours and the origin **o**
6: **if** the value of $S()$ at the origin m4 is less than **o** **then**
7: move the origin **o** the position of the minimum $S()$ value in the non-discounted 4 connected near neighbours.
8: **Goto** line 2
9: **end if**
10: **Finish** minimum position = origin

A. Obtaining Parabolic Minimum Method 1: Simple 4-Connected Gradient Descent

The basis of this method is to start at an origin and check the $S()$ value of its near (4-connected) neighbors (at quarter pixel accuracy). If any of these values are less than the value of $S()$ at the origin then move the origin to the position of the new value and then repeat until the minimum is found. This method is shown in Fig. 3 and described in detail in the pseudo code shown in Algorithm 1.

This method is similar to the block based gradient descent motion estimation scheme utilized within the H.263 standard reference software and defined by Liu and Feig [11]. However, this only uses interpolated values whereas our method uses the estimated SAD values derived from the parameterized parabolic surface (2). This method is not guaranteed to find the minimum value of $S(k)$ within the $(-1,1) \times (-1,1)$ area as the descent of the gradient can get caught in a local minimum. This problem

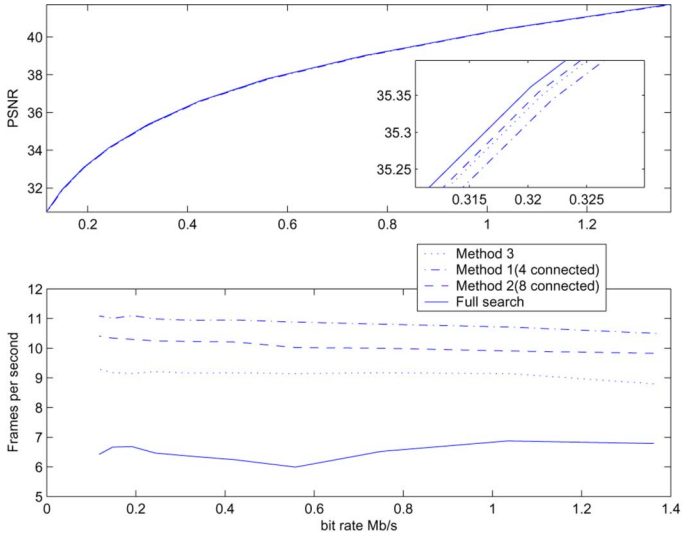


Fig. 4. Results for obtaining the parabolic minimum methods (Foreman Sequence: 100 frames, CIF). Top: Rate-distortion results. Bottom: Encoder speed in fps. (Color version available online at: http://ieeexplore.ieee.org.)

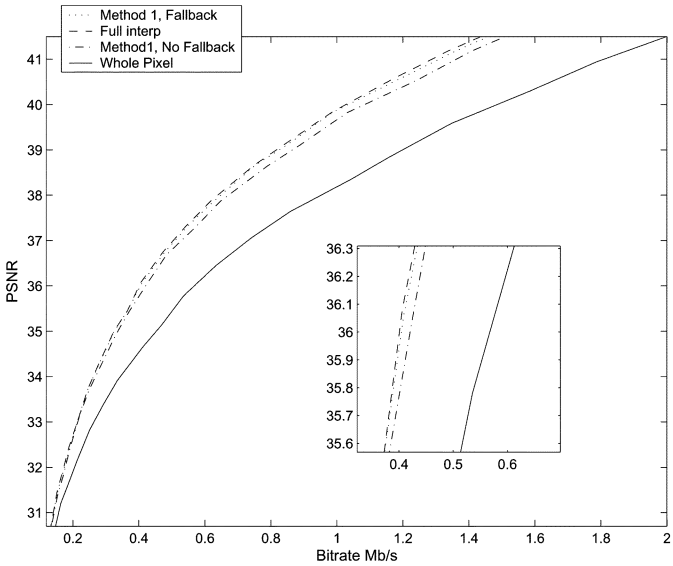


Fig. 5. Comparison of rate-distortion results for subpixel methods: Foreman.

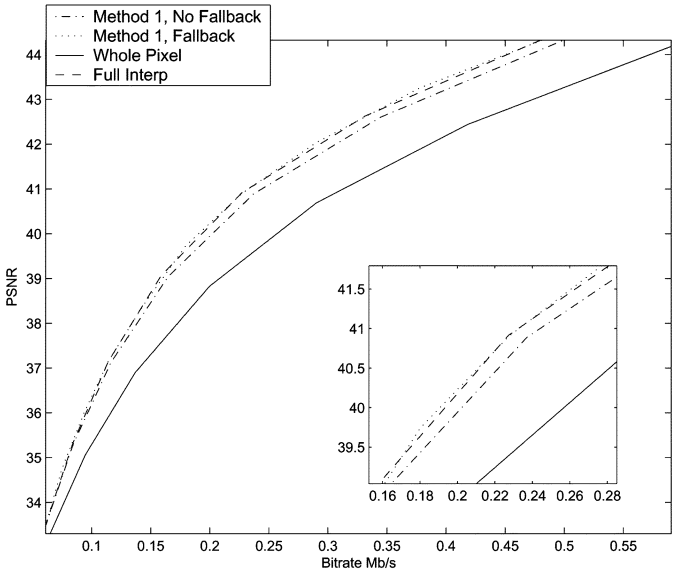


Fig. 6. Comparison of rate-distortion results for subpixel methods: Akiyo.

is reduced using an 8-connected search. Method 1 used in conjunction with an 8-connected search is defined as method 2.

B. Obtaining Parabolic Minimum Method 3: Two-Stage Algorithm

The three-stage algorithm [12], logarithmic search [13] and four step search [14] are all common hierarchical methods for motion estimation. In the case of the location of the quarter pixel parabolic minimum there is only space for two levels of hierarchy. All of these methods, therefore, simplify to the two stage algorithm described below.

The two stage algorithm checks the value of the 8-connected 1/2 pixel resolution positions (and the origin) relative to the origin. The final minimum of $S()$ is then the minimum of an 8-connected 1/4 pixel resolution check centred on the minimum at the 1/2 pixel resolution (including the minimum at the 1/2 pixel resolution). This method is identical to the method for obtaining the interpolated quarter pixel position in the H.264 reference encoder.

Finally, the small size of the quarter pixel area means that it is difficult for any of the standard one or two dimensional minimum location analytical methods [15] to provide any speed improvement.

C. Results for Obtaining Parabolic Minimum Methods

Fig. 4 shows the results of using the three methods described above. The top figure shows an insignificant difference in rate-distortion efficiency between using any of these methods. However, when examined closely (see inset) it can be seen that all three methods have a slight cost involved when compared to the full search. The cost for each method is variable over the rate-distortion curve but is approximately 0.2% of the rate at a fixed PSNR value. This cost is the highest for method 1 using a 4-connected search followed by the 8-connected search of method 2 and finally method 3. However, the speed of these methods shown in the bottom figure of Fig. 4 shows the significant speed improvement using method 1. As the cost for all the methods is insignificant compared to the full search (top figure) method 1 is preferred.

V. Interpolated Fall-back

Method 1 gives an excellent speed improvement with an insignificant reduction in bit rate and is, therefore, the method subsequently used. The results from method 1, provides a reduction in bit rate when compared to the simple whole pixel case. However, this method is measurably worse (in terms of rate-distortion performance) compared to the fully interpolated case (as shown in shown in Figs. 5–7). This can be accounted for through the fact that the parabolic surface estimated using the CSM method may not give a good representation of the actual interpolated map in the presence of outliers or statistical anomalies.

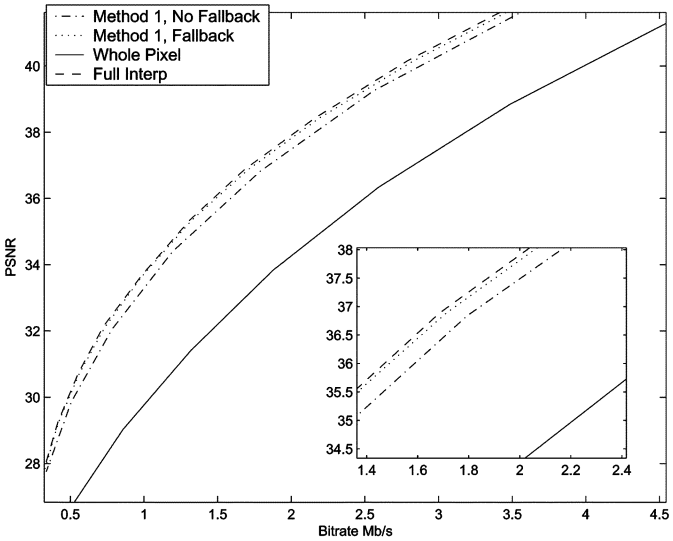


Fig. 7. Comparison of rate-distortion results for subpixel methods: Stefan.

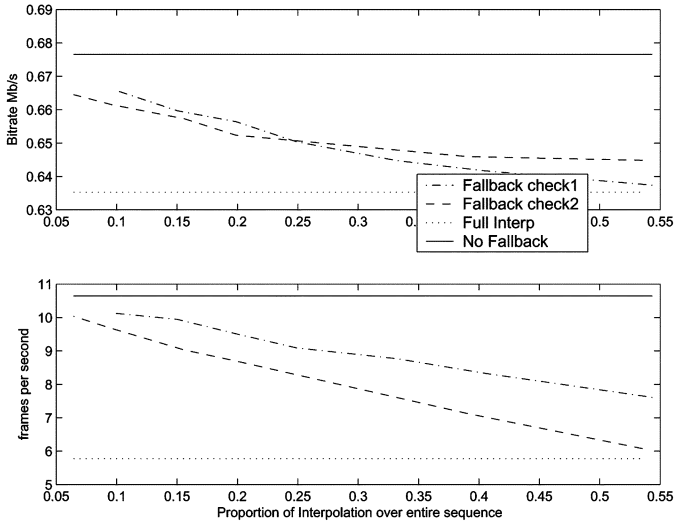


Fig. 8. Comparison of threshold methods. Foreman sequence.

We, therefore, adopt the concept of interpolated fall-back. This involves a check that measures the quality of the result from method 1. The motion vector from method 1 is kept without any further refinement if the result of the consistency check (see below) is positive. However, if the fallback check is negative then the usual interpolation method is used. This has an impact on computationally efficiency; the greater the use of interpolation the better the bit rate but higher the computational load. This, therefore, allows a tradeoff between computational efficiency and compression according to a quality threshold.

A. Fallback Check

The fallback check should reflect how good method 1 (4-connected) is at achieving a SAD minimum similar to that which would have been obtained by interpolation. Two methods were adopted for this.

Fallback check 1: The CSM method for obtaining the correct parameters for the parabolic minimum surface does not give an exact match for the far neighbors, i.e., $S()$ in (2) does not always equal the actual values of SAD values for the far neighbors. Fallback check 1 is, therefore, a measure of how well the parabolic surface model fits the SAD values for the far neighbors. This is achieved by obtaining a measure of the divergence from the model *DivMod*. *DivMod* is defined as the average difference between the predicted value of $S()$ [using (2)] with the actual SAD values for the far neighbors [see (7)]. Fallback check 1 is, therefore, a check to see if this value is over a threshold [see (7)]

$$\text{DivMod} = \sum_{i=1,3,5,7} |S_i - \hat{S}_{ki}| \tag{6}$$

$$\text{Fallback check1}: \quad \frac{\text{DivMod}}{(B_h \times B_w)} > \text{thresh1} \tag{7}$$

where S_i is the predicted SAD values at far neighbor positions 1,3,5 and 7, $\hat{S}_{ki}$ is the actual SAD values at far neighbor positions 1, 3, 5, and 7, and B_w and B_h define the block size for motion estimation (block size varies from 16×16 to 4×4).

B_w and B_h are included to normalize the effect of different sized blocks. Fig. 8 shows how the variation of the threshold changes the percentage of interpolation fallback and its associated compression and timings. Obviously there is a tradeoff between compression and speed according to the value of the threshold.

Fallback check 2: A more direct method of checking how good method 1 is, is obtained using the actual interpolated SAD value at the quarter pixel motion vector position indicated by the minimum found by method 1. The absolute difference between the actual interpolated SAD value at this point and the predicted SAD value [$S()$ in (2)] using method 1 is compared to a threshold [as in (7)] to form fallback check 2. As with fallback check 1, there is also a tradeoff between compression and speed according to the value of the threshold. This is also shown in Fig. 8.

VI. Fallback Check Results

Fig. 8 shows the results of the two fallback check methods described above. The top line of the top graph shows the bit rate of a system without any interpolation and the bottom line of the same graph shows the bit rate of the system with full interpolation. Between these two lines, the two threshold methods decrease the bit rate as the proportion of interpolation increases. However, this is a much greater increase than would be expected from randomly selected interpolation.

The lower graph in Fig. 8 shows the frame rate of the systems with the top and bottom lines similarly delimiting the non and full interpolation modes. The graph shows how both systems speed up when less interpolation is performed. Fallback check 1 offers the best tradeoff between compression performance and complexity and is, therefore, the recommended method and was selected for the further experiments in this paper.

As bit rate conservation was chosen as a priority, a threshold value of 2.0 (threshold 1) was chosen giving approximately 40%

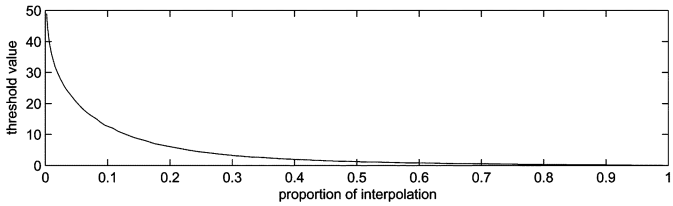


Fig. 9. Variation of interpolation proportion with threshold.

TABLE II
DIFFERENCE BETWEEN RATE-DISTORTION CURVES (MEASURED IN % PSNR DIFFERENCE OVER LOGARITHMIC SCALE [16]) COMPARED TO FULL INTERPOLATION CURVE

	Fallback	No Fallback	Whole Pixel
Foreman	0.4004	1.5703	10.9342
Akiyo	0.0027	0.2316	0.7219
Coast	0.2333	0.7102	8.5228
Hall	0.0000	0.0378	0.4380
Stefan	0.1249	1.9357	16.5843
Container	0.0000	0.5626	9.2982
Mobile	0.2066	3.5145	21.3099

interpolation. As shown in Fig. 8, this offers a significant reduction in bit rate compared to the noninterpolated case but also offers a speed-up of 8 fps. This is the method and threshold used for all the subsequent experimental results. Other threshold values could be chosen (the proportion of interpolation versus threshold value is shown in Fig. 9) but threshold $= 2.0$ was decided upon as giving a good compromise between compression performance and speed improvement.

It should be noted that all methods include the cost of the motion vector together with the SAD measure in order to optimize the rate-distortion efficiency of the encoder [4]. Minimum location methods have been presented by Giunta and Mascia [2], however, these do not take into account this factor and, therefore, would be less efficient in a rate-distortion sense.

VII. RESULTS FOR ALL METHODS

Figs. 5–7 show the excellent performance of the interpolation free method (no fallback) compared to the whole pixel case. These figures also show that the normal interpolated results are slightly better than the interpolation free results. The same figures show how an interpolation free system using a fall back method can approximate the rate-distortion performance of a fully interpolated system. Table II shows the relative performance of many different test sequences. This table shows the relative difference in rate-distortion performance of the three methods: fallback, no fallback, and whole pixel, where the rate-distortion curves are compared to the fully interpolated case. The metric used is the difference between rate-distortion curves. This is the average difference in PSNR over the range calculated as defined by [16]. Table III shows the relative average speeds of the experiments depicted in Figs. 5–7.

These tables show that the fallback method always provides a speed improvement compared to the fully interpolated case, with negligible loss of quality.

TABLE III
SPEEDS OF METHODS IN FRAMES PER SECOND OF ENCODER

	Full Interp	Fallback	No Fallback	Whole Pixel
Foreman	7.3034	8.5351	10.5877	14.9931
Akiyo	9.7557	11.7719	12.2695	15.6095
Coast	7.1907	7.6270	12.1715	16.2123
Hall	10.902	13.197	13.7286	17.8572
Stefan	6.4978	7.2846	10.3925	14.9289
Container	14.4546	15.9767	18.2345	22.4146
Mobile	5.8278	8.0765	11.3227	15.1986

A. Experimental Conditions

The platform the experiments were run on was a desktop pentium 4 2.8 GHz. The experiments were carried out using a baseline implementation of the H.264 encoder (i.e., only I- and P-modes, CAVLC, deblocking filter and a single reference frame). All sequences consisted of 100 frames at CIF resolution encoded at 25 fps and encoded with an I-frame frequency of 10.

The cost function of each motion estimation is returned using the actual interpolated SAD value, i.e., at least one interpolation is carried out for each block. This significantly reduced the bit rate compared to using the estimated SAD value from (2). Additionally, if the actual interpolated cost is greater than the whole pixel result, then the whole pixel result is used.

The experiments that resulted in Fig. 8 were performed with a set macroblock quantization parameter (QP) of 26. The bit rate variation was, therefore, indicative of rate-distortion performance as the PSNR stayed approximately constant in all cases.

VIII. CONCLUSION

Quarter pixel motion estimation within H.264 provides a key contribution to the standard's high performance in a rate-distortion sense (quarter pixel motion compensation gives a reduction in bit rate of between 10% and 20% compared to whole pixel motion compensation). However, the interpolation required to give an accurate estimation for quarter pixel resolution is computationally intensive. This is compounded by the exhaustive search used by the H.264 encoder to check all possible prediction modes. Interpolation-free quarter pixel motion estimation is able to give a substantial reduction in computational complexity. This is shown in Table II where the speed improvement of the non-fallback method over full interpolation ranges from 25% to 60%. However, as is shown in Figs. 5–7, this method is inferior (in a rate-distortion sense) to a fully interpolated system. The fallback methodology introduced in this paper improves the rate-distortion performance to a level close to full interpolation, with only a slight increase in complexity. It is recommended to use fallback check 1 to implement the fallback system.

Table II shows that the fallback system is faster (in terms of frames per second) than a fully interpolated system by a factor of between 12% and 20%.

There are many possibilities for mode selection and early termination (early skip for example). While none of these techniques were used within this work, their incorporation would further increase the speed of the encoder and complement the approach presented.

APPENDIX
ACCURACY OF PARABOLIC MODEL FOR SUBPIXEL INTERPOLATION

A second-order parabolic surface function has been adopted as an alternative to the interpolation required for subpixel motion estimation (2). In order to examine the suitability of such a model, we consider the 1-D case followed by the 2-D case.

1-D Case: Assuming the model of two signals containing two differently delayed and scaled versions of the same signal $s(t)$ and two measurement noises $n_1(t)$ and $n_2(t)$

$$\begin{aligned} x_1(t) &= s(t) + n_1(t) \\ x_2(t) &= As(t-D) + n_2(t). \end{aligned} \tag{8}$$

These signals can be characterized by the their autocorrelation functions $R_s(\tau) = E\{s(t)\cdot s(t-\tau)\}$, $R_{n1}(\tau)$, and $R_{n2}(\tau)$. The task is, therefore, to find a model of the interpolation between the signals and, therefore, obtain the displacement D.

Two coarse level estimators to obtain D are the direct correlator (DC) estimator and the average magnitude difference function (AMDF) estimator and can be defined as

$$\hat{R}_{\mathrm{DC}}(\tau) \equiv \frac{1}{N}\sum_{k=1}^{N} x_1(k)x_2(k+\tau)$$

$$\hat{R}_{\mathrm{AMDF}}(\tau) \equiv \frac{1}{N}\sum_{k=1}^{N} |x_1(k) - x_2(k+\tau)|.$$

D is, therefore, obtained as the minimum and maximum values of τ for R_{AMDF} and R_{DC} respectively. The estimated correlation function $E\{\hat{R}_{\mathrm{DC}}(\tau)\}$ can be expressed as (in the case of uncorrelated noise and and a single source)

$$E\{\hat{R}_{\mathrm{DC}}(\tau)\} = AR_s(\tau - D). \tag{9}$$

This is band limited [17] and presents a symmetric peak around $\tau = D$ (due to the nature of the autocorrelation). This suggests that this function is, therefore, able to be approximated using a convex second order parabola

$$\hat{R}_{\mathrm{DC}}(\tau) \approx C\tau^2 + B\tau + C \tag{10}$$

where A, B, and C are the parabolic parameters The AMDF estimator is more difficult to similarly evaluate as it is not band limited [17]. However, under the assumption that $s(t)$ is a zero mean Gaussian stationary process (denoting $\sigma_X^2 = R_X(0)$)

$$E\{\hat{R}_{\mathrm{AMDF}}(\tau)\}^2 = \frac{2}{\pi}\left(\sigma_s^2(1+A^2) + \sigma_{n1}^2 + \sigma_{n2}^2 - 2AR_s(\tau - D)\right). \tag{11}$$

which also suggests an approximation using a simple parabola, but in this case (due to the inverse sign) a concave parabola [but can also be approximated using (10)].

Jacovitti and Scarano [17] evaluate these expressions further to obtain measures of the "parabolic misfit." An extremely close fit of the parabolic model when using Gaussian sources is shown in [17, Fig. 2].

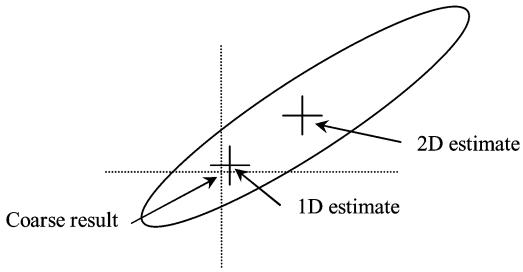


Fig. 10. Effect of systematic bias of the 1-D estimator for the detection of a 2-D paraboloid minimum.

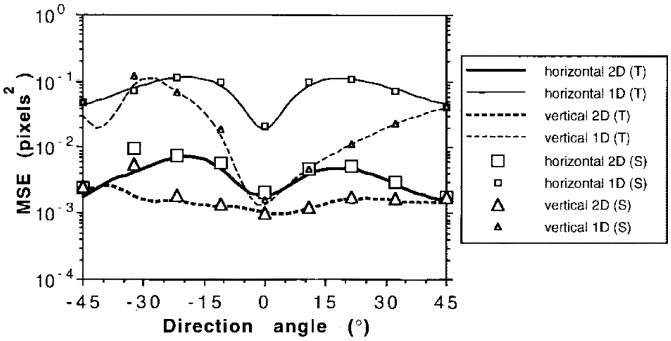


Fig. 11. Theoretical (T) curves and simulated (S) points of horizontal and vertical misplacement MSEs (in pixels ^{2}) versus the direction angle, obtained by the 2-D and two 1-D estimators of the actual shift $x = y = 0.25$ from 16×16 random blocks with directional Gaussian-shaped ACF (eccentricity factor $= 3$) for $\mathrm{SNR} = 40$ dB (taken from [6]).

2-D Case: Fig. 10 shows the systematic bias of the 1-D estimator when used to find the minimum of a 2-D-paraboloid. The 1-D estimator parabola (justified for the 1-D case) must be extended to an equivalent 2-D estimator. A 2-D parabolic surface [i.e., (2)] model is the 2-D extension of (10). Fig. 11 shows the theoretical and simulated misplacement of the minimum found using the 1-D and 2-D parabolic surface models as obtained by Giunta [6].

The theoretical results were obtained using second order Taylor expansions using the simulated input Gaussian sources (see below). The simulated results were obtained using 1000 test runs with a directional 2-D Gaussian-shaped ACF $R_s(t_1, t_2)$

$$R_s(t_1,t_2) = e^{\left((t_1\cos\theta + t_2\sin\theta)^2/2\sigma_1^2\right) - \left((t_1\sin\theta + t_2\cos\theta)^2/2\sigma_2^2\right)} \tag{12}$$

where θ is the direction angle of the simulated function, and σ_1 and σ_2 control the width of the Gaussian function. Two independent 2-D white Gaussian noises were added each with an SNR of 40 dB. Fig. 11 shows that the simulated and theoretical results give a good match. It also shows that the 2-D estimator [and, therefore, the parabolic model (2)] are superior to two 1-D estimators.

All the above mathematical formalism is taken from [17], however, the AMDF estimator is equivalent to a normalized SAD (used in (2)). A simple second order parabolic estimation

is, therefore, justified in the 1-D case for SAD values and by extension to a 2-D parabolic surface as shown above.

REFERENCES

[1] C. Zhu, X. Lin, and L. P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 5, pp. 349–355, May 2002.

[2] G. Giunta, "Fast estimators of time delay and doppler stretch based on discrete-time methods," *IEEE Trans. Signal Process.*, vol. 46, no. 7, pp. 1785–1797, Jul. 1998.

[3] Y. S. Jehng, L. G. Chen, and T. D. Chiueh, "An efficient and simple VLSI tree architecture for motion estimation algorithms," *IEEE Trans. Signal Process.*, vol. 41, no. 2, pp. 889–900, Feb. 1993.

[4] *H.264. Draft ITU-T Rec. and FDIS of Joint Video Spec. (H.264:ISO=IEC 14496-10 AVC)*, JVT of MPEG and VCEG, Doc. JVT-G050r1, 2003.

[5] C. Cafforio and F. Rocca, "Methods for measuring small displacements of television images," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 9, pp. 573–579, Sep. 1976.

[6] G. Giunta, "Fine estimators of 2-D parameters and application to spatial shift estimation," *IEEE Trans. Signal Process.*, vol. 46, no. 12, pp. 3201–3207, Dec. 1999.

[7] H. M. Kim and B. Kosko, "Neural fuzzy motion estimation and compensation," *IEEE Trans. Signal Process.*, vol. 45, no. 10, pp. 2315–2332, Oct. 1997.

[8] G. Giunta and U. Mascia, "Estimation of global motion parameters by complex linear regression," *IEEE Trans. Signal Process.*, vol. 8, no. 11, pp. 1652–1657, Nov. 1999.

[9] T. K. Chiew, "Rapid segmentation for video processing," Ph.D. dissertation, Dept. Elect. Elcetron. Eng., Univ. Bristol, Bristol, U.K., 2004.

[10] T. K. Chiew, J. T. H. Chung-How, D. R. Bull, and C. N. Canagarajah, "Interpolation-free subpixel refinement for block-based motion estimation," *Proc. SPIE*, vol. 5308, pp. 1261–1269, 2004.

[11] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technology*, vol. 6, no. 4, pp. 419–423, Aug. 1996.

[12] B. Zeng, R. Li, and M. L. Liu, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, Aug. 1994.

[13] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, no. 8, pp. 1799–1808, Aug. 1981.

[14] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast blockmatching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[15] W. T. Vetterling, W. H. Press, S. A. Teukolsky, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1993.

[16] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *VCEG document VCEG-M33*, Mar. 2001.

[17] G. Jacovitti and G. Scarano, "Discrete time techniques for time delay estimation," *IEEE Trans. Signal Process.*, vol. 41, no. 2, pp. 525–533, Feb. 1993.