PERGAMON

Available at www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 1451 – 1458

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

# A novel two-phase Hilbert-scan-based search algorithm for block motion estimation using CTF data structure

Kuo-Liang Chung*,1, Lung-Chun Chang

*Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Rd., Taipei 10672, Taiwan, ROC*

Received 30 April 2003; received in revised form 29 October 2003; accepted 1 December 2003

**Abstract**

Motion estimation is one of the kernel issues in MPEG series. In this correspondence, a novel two-phase Hilbert-scan-based search algorithm for block motion estimation is presented. First in the intra-phase, a segmentation of the Hilbert curve is applied to the current block, then a novel coarse-to-fine data structure is developed to eliminate the impossible reference blocks in the search window of the reference frame. In the inter-phase, a new prediction scheme for estimating the initial motion vector of the current block is presented. Experimental results reveal that when compared to the GAPD algorithm, our proposed algorithm has better execution time and estimation accuracy performance. Under the same estimation accuracy, our proposed algorithm has better execution time performance when compared to the FS algorithm. In addition, when comparing with the TSS algorithm, our proposed algorithm has better estimation accuracy performance, but has worse execution time performance.
© 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

*Keywords:* Block matching; Coarse-to-fine data structure; Estimation accuracy; Hilbert curve

## 1. Introduction

Motion estimation plays an important role in video coding [1,2]. The block matching (BM) algorithm is widely used in motion estimation. In BM algorithm, the encoder first divides the current frame into many fixed-size blocks, and the motion vector for the current block with size $b\times b$, $b=2^m$, is determined by finding the best matching reference block in the reference frame according to the defined matching criterion. Commonly, a search window of size $(2W+1)\times(2W+1)$ is used to confine the search range in the reference frame. Throughout this paper, the matching criterion used is the sum of absolute difference (SAD) which is defined by

$$\mathrm{SAD}(v_x,v_y)=\sum_{x=0}^{b-1}\sum_{y=0}^{b-1}|B_c(x,y)-B_r(x+v_x,y+v_y)| \quad (1)$$

for $-W\leqslant v_x,v_y\leqslant W$ where $B_c(x,y)$ denotes the gray value of the pixel at position $(x,y)$ in the current block and $B_r(x+v_x,y+v_y)$ denotes the gray value of the pixel at position $(x+v_x,y+v_y)$ in the reference block. The best matching reference block within the search window is the one with the minimal $\mathrm{SAD}(v_x,v_y)$.

The full search (FS) algorithm is the most well-known brute-force BM algorithm and can find the best matching block in the search window. Although the FS algorithm can obtain the global optimal result, however the considerable computational cost limits its practical applications. To reduce the computation time requirement, several more efficient BM algorithms have been developed. These developed BM algorithms include the improved FS algorithm [3], the three-step search algorithm [4] and its improved variants [5–9], the binary pyramid search algorithm [10], and the globally adaptive pixel-decimation (GAPD) algorithm [11] which is superior to the other pixel-decimation-based algorithms [12–14].

In this correspondence, a novel two-phase Hilbert-scan-based search algorithm for block motion estimation is presented. First in the intra-phase, a segmentation of the Hilbert

* Corresponding author. Tel.: +886-2-27376771; fax: +886-2-27376777.
*E-mail address:* klchung@cs.ntust.edu.tw (K.-L. Chung).
1 Supported by the National Science Council of R.O.C. under contracts NSC90-2213-E-011-056 and NSC92-2213-E-011-080.

curve is applied to the current block based on the look-up table (LUT) technique, then a novel coarse-to-fine (CTF) data structure is developed. For the current block, the proposed CTF data structure is used to eliminate the impossible reference blocks in the search window of the reference frame. In the inter-phase, a new prediction scheme for estimating the initial motion vector of the current block is presented. Under five different kinds of real video sequences, experimental results reveal that when compared to the GAPD algorithm [11], our proposed algorithm has better execution time and estimation accuracy performance. Under the same estimation accuracy, our proposed algorithm has better execution time performance when compared to the FS algorithm. In addition, when comparing with the TSS algorithm [4], our proposed algorithm has better estimation accuracy performance, but has worse execution time performance.

The remainder of this paper is organized as follows. Section 2 presents the intra-phase of our proposed algorithm. Section 3 presents the inter-phase of our proposed algorithm. Section 4 presents our proposed whole motion estimation algorithm. Experimental results are demonstrated in Section 5. Some concluding remarks are addressed in Section 6.

## 2. Intra-phase: segmentation of Hilbert curve and CTF data structure

Hilbert curve [15] is a curve passing through all pixels in the image domain and it scans the neighbouring pixel continuously. Consider a $2^r \times 2^r$ gray image represented as an array in the integer domain $\{x, y \mid 0 \leqslant x \leqslant 2^r - 1,\ 0 \leqslant y \leqslant 2^r - 1\}$. Two simple Hilbert curves on the $2 \times 2$ image domain and the $4 \times 4$ image domain are shown in Fig. 1(a) and (b), respectively. Each entry along this curve is denoted by an integer called the Hilbert order denoted by the symbol $o$. For example, the corresponding Hilbert orders of points $(0, 0)$ and $(1, 0)$ in Fig. 1(a) are 1 and 2, respectively.

In motion estimation, since the size of each block is fixed, usually $16 \times 16$, an LUT-based array $OC_p$ is built up to keep the relationship between the Hilbert order and the coordinate $(x, y)$ of a pixel in the block. According to the $OC_p$ array, the Hilbert order of a pixel at coordinate $(x, y)$ in the current block can be obtained in $O(1)$ time and vice versa. Three sub-arrays are used to implement the $OC_p$ array. The first sub-array $O_p[x, y]$, $0 \leqslant x \leqslant 15$ and $0 \leqslant y \leqslant 15$, is used to record the Hilbert order at position $(x, y)$. Further, the $x$-coordinate and $y$-coordinate of each pixel with Hilbert order $o$, $0 \leqslant o \leqslant 255$, are stored in the second sub-array $C_{p_x}[o]$ and the third sub-array $C_{p_y}[o]$, respectively.

In Section 2.1, a segmentation for the Hilbert curve of the current block is presented. Our proposed segmentation leads to the design of novel CTF data structure.

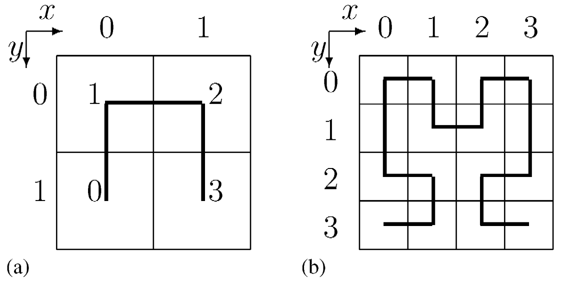

Fig. 1. Two simple Hilbert curves. (a) $2 \times 2$ domain. (b) $4 \times 4$ domain.

### 2.1. Segmentation of Hilbert curve

For the current block with size $2^m \times 2^m$, the obtained Hilbert curve is stored in the array $H = \langle (0, g_0), (1, g_1), \ldots, (2^m \times 2^m - 1, g_{2^m \times 2^m - 1}) \rangle$ where $(o, g_o)$ denotes the gray level $g_o$ for the pixel with Hilbert order $o$. Initially, the stored Hilbert curve is viewed as a 1-D segment with $2^{2m}$ entries.

The segmentation of Hilbert curve is a recursive process. At each subdivision step, the 1-D segment with $2^k$ entries, $2 \leqslant k \leqslant 2m$, is subdivided into two $2^{k-1}$ 1-D sub-segments. If the sub-segment is not homogeneous, it is subdivided into two equal-sized sub-segments until all homogeneous sub-segments are obtained. A sub-segment is called a homogeneous sub-segment if the estimated gray level of each pixel in the sub-segment is in some vicinity of its real gray level. Suppose the indices of the two end-points of the sub-segment are $i_1$ and $i_2$ and their corresponding gray levels are $g_{i_1}$ and $g_{i_2}$, respectively. Using the linear interpolation method, the estimated gray level of the pixel with index $i$, $i_1 \leqslant i \leqslant i_2$, in the sub-segment is calculated by

$$\bar{g}_i = g_{i_1} + \frac{g_{i_2} - g_{i_1}}{i_2 - i_1}(i - i_1). \tag{2}$$

Given a specified error tolerance $\varepsilon$, if the quality condition, $|g_i - \bar{g}_i| \leqslant \varepsilon$ for $i_1 \leqslant i \leqslant i_2$ holds, then the sub-segment is homogeneous.

Let us take an example to demonstrate how the above segmentation method works. Let $S_1^1$ be the approximated segment of the Hilbert curve from the Hilbert order $A$ to $B$ and let $d_1$ be the maximal absolute difference between $S_1^1$ and the original segment from $A$ to $B$. If $d_1 > \varepsilon$, the original segment from $A$ to $B$ is subdivided into two equal-sized sub-segments. Here, $C$ is the first division point. $S_1^2$ ($S_2^2$) is the new approximated sub-segment of the Hilbert curve from $A$ to $C$ ($C$ to $B$). Further, let $d_2$ be the maximal absolute difference between the original segment from $A$ to $C$ and $S_1^2$. Suppose $d_2 > \varepsilon$, then $D$ is the second division point and we have $S_1^3$ and $S_2^3$ with respect to the two Hilbert curves from $A$ to $D$ and from $D$ to $C$, respectively. Let all the sub-segments be homogeneous up to here. Finally, the Hilbert curve from $A$ to $B$ can be approximated by $\hat{H}(\varepsilon) = S_1^3 \cup S_2^3 \cup S_2^2$.

| | | | |
|---|---|---|---|
| 93 | 87 | 110 | 121 |
| 105 | 100 | 98 | 116 |
| 95 | 82 | 96 | 102 |
| 79 | 88 | 92 | 84 |

Fig. 2. A $2^2 \times 2^2$ block.

## 2.2. CTF data structure

In this subsection, a novel CTF data structure is presented to store $\hat{H}(\varepsilon)$. Suppose the Hilbert curve $H$ has been divided into $N$ segments, say $S_1, S_2, \ldots,$ and $S_N$, and $H$ is approximated by $\hat{H}(\varepsilon) = S_1 \cup S_2 \cup \cdots \cup S_N$.

For each segment $S_j$, $1 \leqslant j \leqslant N$, it has $L_j$ entries and $S_j$ is denoted by $S_j = \langle (O_{j-1},\ \bar{g}^j_{O_{j-1}}), (O_{j-1}+1, \bar{g}^j_{O_{j-1}+1}), \ldots, (O_{j-1}+L_j-1, \bar{g}^j_{O_{j-1}+L_j-1}) \rangle$ where $O_{j-1} = \sum_{k=1}^{j-1} L_k$ and $(i, \bar{g}^j_i)$, $O_{j-1} \leqslant i < O_{j-1}+L_j$, denotes the estimated gray level $\bar{g}^j_i$ of the pixel with Hilbert order $i$ in $S_j$. At top level, i.e. level 0, of the CTF data structure, it saves all the Hilbert orders of the end-points in $\hat{H}$. For each point with Hilbert order $i$ in $S_j$ except the two end-points, we save the Hilbert order $i$ into level $l = \lceil (\varepsilon - d^j_i + 1)/q \rceil$ where $d^j_i = |g_i - \bar{g}^j_i|$ and $\lceil t \rceil$ denotes the smallest integer which is greater than or equal to $t$ and $q$ denotes the gap between level $l$ and $l+1$ for $l \geqslant 1$. In our experiments (see Section 5), the CTF data structure has nine levels, namely level 0, level $1, \ldots,$ level 7, and level 8. In this case, if $\varepsilon = 24$, $q$ is set to be 3 because of $q = \frac{\varepsilon}{8} = \frac{24}{8} = 3$. Similarly, if $\varepsilon = 16$, $q$ is set to be 2. Specially, if $\lceil (\varepsilon - d^j_i + 1)/q \rceil$ is greater than 8, the Hilbert order is saved into the bottom level, i.e. level 8. In summary, in the CTF data structure, the difference $d^j_i$ $(=|g_i - \bar{g}^j_i|)$ is quantized by $q$ and the Hilbert order is assigned to level $l = \lceil (\varepsilon - d^j_i + 1)/q \rceil$. The larger the difference between the original gray level and the estimated gray level is, the sharper the difference between them. Excepting level 0, the Hilbert order of the pixel is assigned to the upper level of the CTF data structure when the original gray level of the pixel is more different from the estimated gray level of the pixel.

We now take an example to explain how to construct the CTF data structure. As shown in Fig. 2, a $2^2 \times 2^2$ block is given where the integer value in each entry denotes the gray level. After scanning the block via the Hilbert scanning order, we obtain $H$ and $\hat{H}(\varepsilon = 16) = S_1 \cup S_2 \cup S_3$ as shown in Table 1. Each difference $d^j_i$, $1 \leqslant j \leqslant 3$ and $0 \leqslant i < 16$, is listed in the last column of this table. Next, we want to construct the CTF data structure of Table 1. Initially, six end-points of the three segments, 0, 7, 8, 11, 12, and 15, are assigned to the top level, i.e. level 0, of the CTF data structure. In fact, it is enough to save 0, 7, 11, and 15 to the top level since for two neighbouring end-points, it is enough to store the corresponding smaller Hilbert order. From the relationship between $d^j_i$'s and the Hilbert order $o$'s in Table 1, when $q = 2$, the reduced CTF data structure is shown in Table 2. For exposition, note that in the segment $S_3$, for the pixel with Hilbert order 13, we have $d^3_{13} = 0$, so the level of that pixel is set to 8.

For each current block, two arrays are enough to implement the reduced CTF data structure. First, an array CTF[$i$], $0 \leqslant i \leqslant 15$, is used to store all the Hilbert orders from the top level to the bottom level. Thus, we

Table 1
The obtained $H$ and $\hat{H}$

| Hilbert order | $H$ | $\hat{H}(\varepsilon = 16)$ | $=$ | $S_1$ | $\bigcup$ | $S_2$ | $\bigcup$ | $S_3$ | $d^j_i$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | ( 0, 79) | ( 0, 79) | | ( 0, 79) | | | | | 0 |
| 1 | ( 1, 88) | ( 1, 82) | | ( 1, 82) | | | | | 6 |
| 2 | ( 2, 82) | ( 2, 85) | | ( 2, 85) | | | | | 3 |
| 3 | ( 3, 95) | ( 3, 88) | | ( 3, 88) | | | | | 7 |
| 4 | ( 4,105) | ( 4, 91) | | ( 4, 91) | | | | | 14 |
| 5 | ( 5, 93) | ( 5, 94) | | ( 5, 94) | | | | | 1 |
| 6 | ( 6, 87) | ( 6, 97) | | ( 6, 97) | | | | | 10 |
| 7 | ( 7,100) | ( 7,100) | | ( 7,100) | | | | | 0 |
| 8 | ( 8, 98) | ( 8, 98) | | | | ( 8, 98) | | | 0 |
| 9 | ( 9,110) | ( 9,104) | | | | ( 9,104) | | | 6 |
| 10 | (10,121) | (10,110) | | | | (10,110) | | | 11 |
| 11 | (11,116) | (11,116) | | | | (11,116) | | | 0 |
| 12 | (12,102) | (12,102) | | | | | | (12,102) | 0 |
| 13 | (13, 96) | (13, 96) | | | | | | (13, 96) | 0 |
| 14 | (14, 92) | (14, 90) | | | | | | (14, 90) | 2 |
| 15 | (15, 84) | (15, 84) | | | | | | (15, 84) | 0 |

Table 2
The reduced CTF data structure

| Level | Segment | Hilbert order |
|---|---|---|
| 0 | End-points | 0, 7, 11, 15 |
| Level | $d_i^j$ | Hilbert order |
| 1 | 16, 15 | |
| 2 | 14, 13 | 4 |
| 3 | 12, 11 | 10 |
| 4 | 10, 9 | 6 |
| 5 | 8, 7 | 3 |
| 6 | 6, 5 | 1, 9 |
| 7 | 4, 3 | 2 |
| 8 | 2, 1, 0 | 5, 13, 14 |

have CTF[0..13] = [0, 7, 11, 15, 4, 10, 6, 3, 1, 9, 2, 5, 13, 14]. In addition, we need another array Counter[0..8] = [4, 0, 1, 1, 1, 1, 2, 1, 3] to record the number of Hilbert orders at level $l$. For example, Counter[0] = 4 means that there are four Hilbert orders, 0, 7, 11, and 15, saved at level 0. From Counter[0] = 4 and CTF[0..3] = [0, 7, 11, 15], we can derive that 8 (=CTF[1] + 1 = 7 + 1) and 12 (=CTF[2] + 1 = 11 + 1) are also the two end-points at level 0 in the reduced data structure.

## 3. Inter-phase: predicting initial motion vector

In the intra-phase mentioned in the last section, we have described how to construct the reduced CTF data structure in order to eliminate the impossible candidates in the search window. In this section, a new prediction strategy is presented to predict the initial motion vector of the current block.

For exposition, we take a 352 × 256 current frame as a representative. First we think each block with size 16 × 16 as a shrinking point. Then, there are 22 × 16 shrinking points in the current shrinking frame. In this shrinking frame as shown in Fig. 3, we first tile a maximal square block $MB_1$ with size 16 × 16. Then four square blocks, each with size 4 × 4, are tiled. Finally, eight square blocks, each with size 2 × 2, are tiled. Considering one possible entrance–exit sequence (see Fig. 3), where the symbol $E_I$ denotes the entrance of the maximal block and the symbol $E_O$ denotes the exit, the Hilbert curve of Fig. 3 is depicted in Fig. 4.

For each frame, since the number of blocks is fixed, a LUT technique can be used to keep the relationship between the Hilbert order and the coordinate $(b_x, b_y)$ of any shrinking point in the shrinking frame. Three arrays are used to implement the LUT and by the three arrays, the Hilbert order of the current block's neighbouring block at the coordinate $(b_x, b_y)$ can be obtained by $O(1)$ time and vice versa. The

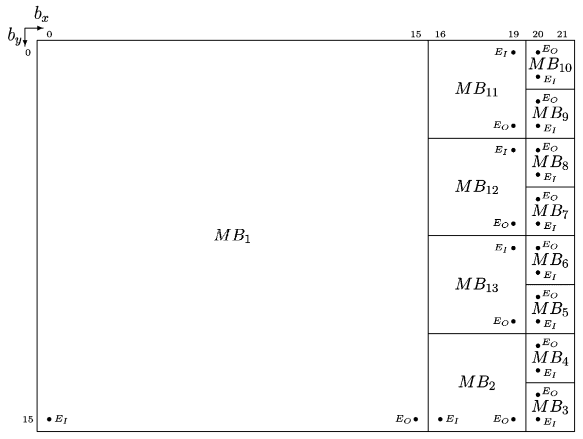

Fig. 3. The decomposition of an image with size 22 × 16.

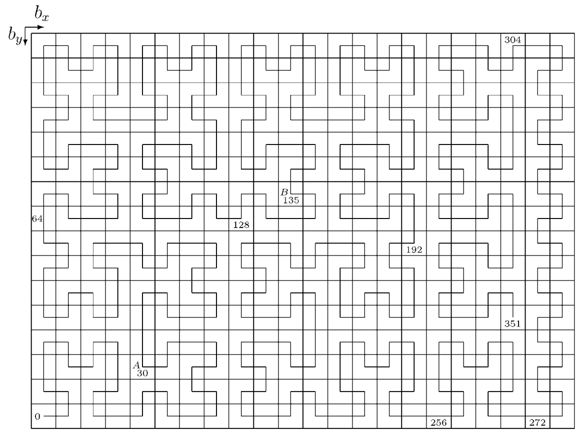

Fig. 4. The Hilbert orders of Fig. 3.

first array used is $O_b[b_x, b_y]$, $0 \leqslant b_x \leqslant 21$ and $0 \leqslant b_y \leqslant 15$, which is used to record the Hilbert order at position $(b_x, b_y)$. For example, the Hilbert order of the shrinking point $A$ at (4, 13) is $O_b[4, 13] = 30$. The $b_x$-coordinate and $b_y$-coordinate of each shrinking point with the Hilbert order $o$, $0 \leqslant o \leqslant 351$, are stored in the second array $C_{b_x}[o]$ and the third array $C_{b_y}[o]$, respectively. For example, the Hilbert order of the shrinking point $B$ at (10, 6) is $O_b[10, 6] = 135$ and the $b_x$-coordinate and $b_y$-coordinate of $B$ with Hilbert order 135 are $C_{b_x}[135] = 10$ and $C_{b_y}[135] = 6$, respectively. After constructing the three LUT-based arrays, for each current block, the Hilbert orders of its eight neighbouring blocks can be obtained in $O(1)$ time. For example, let us enlarge the current block $B$ (see the shrinking point $B$ in Fig. 4) and its eight neighbouring blocks. Fig. 5 demonstrates the enlarged 3 × 3 window containing nine blocks. The Hilbert order of $B$ at (10, 6) is $O_b[10, 6] = 135$ and the Hilbert orders of its eight neighbouring blocks are $O_b[9, 6] = 130$, $O_b[9, 7] = 131$,

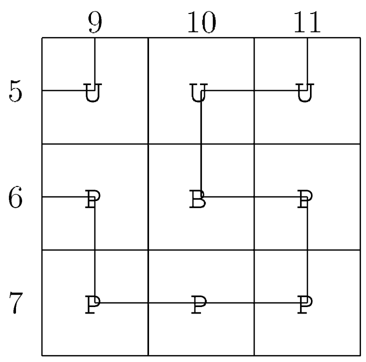

Fig. 5. The current block $B$ and its eight neighboring blocks.

| | | | |
|---|---|---|---|
| 96 | 89 | 106 | 118 |
| 104 | 105 | 98 | 117 |
| 93 | 85 | 97 | 100 |
| 80 | 89 | 90 | 86 |

Fig. 6. The reference block $B_r$.

$O_b[10,7]=132$, $O_b[11,7]=133$, $O_b[11,6]=134$, $O_b[10,5]=136$, $O_b[11,5]=137$, and $O_b[9,5]=141$.

In Fig. 5, from the Hilbert orders of the current block $B$ and its eight neighbouring blocks, we know that the five blocks at (9, 6), (9, 7), (10, 7), (11, 7), and (11, 6) have obtained their motion vectors before the current block $B$ since their Hilbert orders are less than 135. Therefore, these five processed blocks can be used to predict the initial motion vector of $B$. For convenience, in Fig. 5, the symbol $P$ is used to denote the type of each processed block and the symbol $U$ denotes the type of the unprocessed block. In our proposed prediction scheme, we use the mean of the five motion vectors of the five processed blocks as the predicted initial motion vector of the current block $B$.

In practice, each shrinking point is of size $16 \times 16$ and the size of each frame is $352 \times 256$. In Fig. 5, the coordinate of the shrinking point $B$ is (10, 6) in the current shrinking frame domain, but the coordinate of $B$ in the original current frame is $(160, 96)$ $(=(10 \times 16, 6 \times 16))$. From Fig. 5, it is known that $B$ is the current block, i.e. $B_c = B$, and the $33 \times 33$ search window whose center is located at $(160, 96)$ in the reference frame. Suppose the motion vectors of the five processed neighbouring blocks marked by $P$s are $(6,-2), (5,-2), (6,-3), (7,-3)$, and $(6,-4)$, then the mean motion vector of the five processed neighbouring blocks is equal to $(v_x, v_y) = (6,-3)(=((6+5+6+7+6)/5, ((-2)+(-2)+(-3)+(-3)+(-4))/5))$. According to the predicted initial motion vector, we move the reference block $B_r$ from the coordinate $(160,96)$ to the coordinate $(166, 93)$ $(=(160+6, 96-3))$. By Eq. (1), the initial minimal SAD of the current block $B_c$ and the reference block $B_r$ can be calculated and denoted by $\text{SAD}_{min}$. The value $\text{SAD}_{min}$ will be used in our proposed motion estimation algorithm.

## 4. The proposed two-phase motion estimation algorithm

Before presenting our proposed two-phase motion estimation algorithm, we first present a CTF-based successive elimination strategy in order to speed up the searching work in the search window.

Suppose the CTF data structure of the current block $B_c$ has been constructed. In the CTF data structure, for each Hilbert order $o$ at level $l$, $l = 0, 1, \ldots, 8$, the gray value of the pixel at $(C_{p_x}[o], C_{p_y}[o])$ in $B_c$ is compared to that in the reference block $B_r$. Let $\text{PSAD}(B_c, B_r; l)$ denote the accumulated absolute differences between the pixels with the Hilbert orders at level $l$ in $B_c$ and the corresponding pixels in $B_r$. Considering levels from 0 to $l$, the accumulated PSAD for the first $(l+1)$ levels is denoted by $\text{APSAD}(B_c, B_r; 0, l) = \sum_{k=0}^{l} \text{PSAD}(B_c, B_r; k)$. We take an example to illustrate how to calculate the values of $\text{PSAD}(B_c, B_r; l)$ and $\text{APSAD}(B_c, B_r; 0, l)$. The $2^2 \times 2^2$ block in Fig. 2 is used as the current block $B_c$. The reference block $B_r$ with size $2^2 \times 2^2$ is assumed to be Fig. 6. In Table 2, it implies that the Hilbert orders at level 0 of the CTF data structure are 0, 7, 8, 11, 12, and 15 although only 0, 7, 11, and 15 are saved at level 0. Using the $OC_p$ array, the $x$- and $y$-coordinates with Hilbert orders 0, 7, 8, 11, 12, and 15 are (0, 3), (1, 1), (2, 1), (3, 1), (3, 2), and (3, 3), respectively, as shown in Fig. 1(b). We thus have $\text{PSAD}(B_c, B_r; 0) = 11$ $(=|79-80|+|100-105|+|98-98|+|116-117|+|102-100|+|84-86|)$. Since there is no Hilbert order at level 1, we have $\text{PSAD}(B_c, B_r; 1) = 0$. When $l = 2$, we have $\text{PSAD}(B_c, B_r; 2) = 1$ and $\text{APSAD}(B_c, B_r; 0, 2) = 12$ $(=11+0+1)$. From $\text{APSAD}(B_c, B_r; 0, l) = \text{APSAD}(B_c, B_r; 0, l-1) + \text{PSAD}(B_c, B_r; l)$, we have

$$\text{APSAD}(B_c, B_r; 0, 0) \leqslant \text{APSAD}(B_c, B_r; 0, 1) \leqslant \cdots \leqslant \text{APSAD}(B_c, B_r; 0, 8). \qquad (3)$$

We now describe how Eq. (3) can be used to speed up the searching process in the search window. For the current block $B_c$ and the reference block $B_r$ in the search window, it starts from the top level, i.e. level 0, of the CTF data structure and goes downward to the bottom level, i.e. level 8. According to the predicted initial motion vector of $B_c$, suppose the initial minimal SAD, i.e. $\text{SAD}_{min}$, has been calculated. We first calculate $\text{APSAD}(B_c, B_r; 0, 0)$. If $\text{APSAD}(B_c, B_r; 0, 0) > \text{SAD}_{min}$, then $\text{APSAD}(B_c, B_r; 0, l) > \text{SAD}_{min}$ will hold for $1 \leqslant l \leqslant 8$.

Table 3
Execution time and estimation accuracy performance comparison

| | | Salesman | | Garden | | Calendar train | | Football | | Table tennis | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | ε | TIME | PSNR | TIME | PSNR | TIME | PSNR | TIME | PSNR | TIME | PSNR |
| FS | | 45.64 | 34.44 | 66.14 | 27.02 | 66.20 | 31.67 | 66.22 | 26.06 | 66.20 | 28.38 |
| TSS | | 0.75 | 34.16 | 1.13 | 26.00 | 1.05 | 31.31 | 1.10 | 25.44 | 1.13 | 26.80 |
| GAPD | 16 | 32.95 | 32.99 | 35.95 | 25.06 | 48.72 | 31.19 | 44.36 | 25.08 | 32.77 | 27.24 |
| | 24 | 38.14 | 33.56 | 42.51 | 26.11 | 54.50 | 31.30 | 54.61 | 25.43 | 41.87 | 27.74 |
| | 32 | 41.64 | 33.82 | 47.44 | 26.34 | 59.14 | 31.40 | 60.98 | 25.58 | 49.39 | 27.90 |
| OURS | 16 | 19.73 | 34.44 | 34.61 | 27.02 | 27.63 | 31.67 | 35.17 | 26.06 | 39.86 | 28.38 |
| | 24 | 18.47 | 34.44 | 33.39 | 27.02 | 26.51 | 31.67 | 34.94 | 26.06 | 38.23 | 28.38 |
| | 32 | 18.45 | 34.44 | 32.60 | 27.02 | 26.22 | 31.67 | 35.96 | 26.06 | 37.98 | 28.38 |

Thus, the reference block $B_r$ will not be the best matching block and can be rejected. If $\mathrm{APSAD}(B_c, B_r; 0, 0) \leqslant \mathrm{SAD}_{min}$, $\mathrm{APSAD}(B_c, B_r; 0, 1)$ is calculated. Similarly, if $\mathrm{APSAD}(B_c, B_r; 0, 1) > \mathrm{SAD}_{min}$, then the reference block $B_r$ can be rejected; otherwise, $\mathrm{APSAD}(B_c, B_r; 0, 2)$ is tested further. This multilevel pruning process is repeated until $B_r$ is rejected or the bottom level of the CTF data structure has been reached. If the bottom level is reached, $\mathrm{APSAD}(B_c, B_r; 0, 8)$ is calculated by Eq. (3). Then we check whether $\mathrm{SAD}_{min}$ should be replaced or not. If $\mathrm{SAD}_{min}$ is replaced, then the current best matching block is set to be $B_r$. Consequently, for block motion estimation, the above multilevel pruning process leads to computation-saving effect.

Given a video sequence as the input and the specified error tolerance $\varepsilon$, for the $(t-1)$th reference frame and the $t$th current frame, $t \geqslant 2$, our proposed two-phase algorithm for motion estimation consisting of four steps is shown below.

### 4.1. Intra-phase

*Step 1*: For each current block $B_c$ in the $t$th frame, according to the segmentation of Hilbert curve described in Section 2.1, we obtain the array $\hat{H}(\varepsilon)$ to save the approximate segments.

*Step 2*: Based on the description of Section 2.2, we construct the reduced CTF data structure.

### 4.2. Inter-phase

*Step 3*: According to the three arrays, $O_b[,]$, $C_{b_x}[]$, and $C_{b_y}[]$, defined in Section 3 and these processed neighbouring blocks of $B_c$, the initial motion vector of $B_c$ is predicted using the mean of its processed neighbouring blocks' motion vectors. Then, the initial minimal SAD, $\mathrm{SAD}_{min}$, is calculated.

*Step 4*: Based on the row-major order, for each reference block $B_r$ in the search window, the searching process consisting of four steps is shown below.

*Step 4.1*: We calculate $\mathrm{APSAD}(B_c, B_r; 0, 0)$.

*Step 4.2*: By Eq. (3), if $\mathrm{APSAD}(B_c, B_r; 0, 0)$ is greater than $\mathrm{SAD}_{min}$, $\mathrm{APSAD}(B_c, B_r; 0, l) > \mathrm{SAD}_{min}$ will hold for $1 \leqslant l \leqslant 8$. Thus, the reference block $B_r$ will not be the best matching block and go to Step 4.4. If $\mathrm{APSAD}(B_c, B_r; 0, 0)$ is less than or equal to $\mathrm{SAD}_{min}$, $\mathrm{APSAD}(B_c, B_r; 0, 1)$ is calculated by Eq. (3). Similarly, if $\mathrm{APSAD}(B_c, B_r; 0, 1) > \mathrm{SAD}_{min}$, then the reference block $B_r$ is not the best matching block and go to Step 4.4; otherwise, $\mathrm{APSAD}(B_c, B_r; 0, 2)$ is tested, and so on. This comparison process is repeated until $B_r$ is rejected or the bottom level of the CTF data structure is reached.

*Step 4.3*: If the bottom level of the CTF data structure is reached, $\mathrm{APSAD}(B_c, B_r; 0, 8)$ is calculated and we check whether $\mathrm{SAD}_{min}$ should be replaced or not. If $\mathrm{APSAD}(B_c, B_r; 0, 8)$ is less than $\mathrm{SAD}_{min}$, the current minimal SAD, i.e. $\mathrm{SAD}_{min}$, is replaced and the current best matching block is set to be $B_r$; otherwise, go to Step 4.4.

*Step 4.4*: The reference block $B_r$ is rejected. Further, the next reference block $B_r$ in the search window is compared to $B_c$ and go to Step 4.1.

## 5. Experimental results

In this section, five video sequences, namely the salesman, garden, calendar train, football, and table tennis, are used to evaluate the performance among the FS algorithm, three-step search (TSS) algorithm, the GAPD algorithm [11], and our proposed algorithm (OURS for short). Under the five video sequences, the first 21 frames of the salesman sequence and the first 30 frames of the other four sequences are used. Each frame in the video sequence is of size 352 × 256. The block size and the size of the search window are selected as $16 \times 16$ and $33 \times 33$, respectively. All the concerning algorithms are implemented using

Borland C++ builder and the IBM compatible personal computer Pentium 4 microprocessor with 1.4$G$ MHz.

The peak signal-to-noise ratio (PSNR) are used to measure the estimation accuracy of each concerning BM algorithm. The PSNR is defined by

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right)\ \mathrm{d}B,$$

where MSE is the mean square error between the estimated frame and the original frame. For each video sequence, given three kinds of error tolerance, say $\varepsilon = 16, 24$, and 32, Table 3 lists the average PSNR and the required execution time for each error tolerance where the symbol TIME, which is the summation of preprocessing time and searching time, is used to measure the execution-time performance in terms of millisecond ($=10^{-3}$ s). Here, the preprocessing time includes the time required in Steps 1–3 in our proposed OURS algorithm and the searching time denotes the time required in Step 4.

Let the execution time improvement ratio of our proposed OURS algorithm over the GAPD algorithm be defined by $(T_{GAPD} - T_{OURS})/T_{GAPD} \times 100\%$, where $T_{GAPD}$ and $T_{OURS}$ denote the execution time required in the GAPD algorithm and the proposed OURS algorithm, respectively. When comparing with the GAPD algorithm, Table 3 reveals our proposed algorithm OURS has 33.1% execution time improvement ratio and has 0.79 dB estimation accuracy improvement ratio.

In our proposed OURS algorithm, the estimation accuracy is independent of the error tolerance $\varepsilon$ since all the reference blocks in the search window must be checked. Therefore, the estimation accuracy of our proposed algorithm is almost the same as that of the FS algorithm. However, due to use of the proposed CTF data structure and applying Eq. (3), our proposed algorithm can reject more reference blocks early and has better execution time performance when compared to the FS algorithm. In addition, when comparing with the TSS algorithm, our proposed algorithm has better estimation accuracy performance, but has worse execution time performance.

## 6. Conclusion

This paper has presented the novel two-phase Hilbert-scan-based search algorithm for block motion estimation. In the intra-phase, after presenting the segmentation method for the Hilbert curve, then a novel reduced CTF data structure has been developed in order to eliminate the impossible reference blocks in the search window. In the inter-phase, a new prediction scheme for estimating the initial motion vector of the current block has been presented. Under five different kinds of real video sequences, experimental results reveal that when compared to the GAPD algorithm, our proposed algorithm has better execution time and estimation accuracy performance. Under the similar estimation accuracy, our proposed algorithm has better execution time performance when compared to the FS algorithm. In addition, when comparing with the TSS algorithm, our proposed algorithm has better estimation accuracy performance, but has worse execution time performance.

Instead of storing all the look-up tables for keeping the relation between Hilbert orders and $(x, y)$-coordinates, it is an interesting research issue to derive a general formula to cover different frame sizes, such as CCIR601 ($720 \times 480$), CIF ($352 \times 288$), and so on.

## References

[1] K.R. Rao, J.J. Hwang, Techniques and Standards for Image, Video, and Audio Coding, Prentice-Hall, Englewood Cliffs, NJ, 1996.

[2] A.M. Tekalp, Digital Video Processing, Prentice-Hall, Englewood Cliffs, NJ, 1995.

[3] M.J. Chen, L.G. Chen, T.D. Chiueh, One-dimensional full search motion estimation algorithm for video coding, IEEE Trans. Circuits Syst. Video Technol. 4 (1994) 504–509.

[4] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, Motion compensated interframe coding for video conferencing, in: Proceeding of the National Telecommunications Conference, November/December 1981, pp. G.5.3.1–G.5.3.5.

[5] M. Ghanbari, The cross-search algorithm for motion estimation, IEEE Trans. Commun. 38 (1990) 950–953.

[6] L.M. Po, W.C. Ma, A novel four-step algorithm for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 6 (1996) 313–317.

[7] L.K. Liu, E. Feig, A block-based gradient descent search algorithm for block motion estimation in video coding, IEEE Trans. Circuits Syst. Video Technol. 6 (1996) 419–422.

[8] F.H. Cheng, S.N. Sun, New fast and efficient two-step search algorithm for block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 9 (1999) 977–983.

[9] S. Zhu, K.K. Ma, A new diamond search algorithm for fast block-matching motion estimation, IEEE Trans. Image Process. 9 (2000) 287–290.

[10] X. Song, T. Chiang, X. Lee, Y.Q. Zhang, New fast binary pyramid motion estimation for MPEG2 and HDTV encoding, IEEE Trans. Circuits Syst. Video Technol. 10 (2000) 1015–1028.

[11] Y. Wang, Y. Wang, H. Kuroda, A globally adaptive pixel-decimation algorithm for block-motion estimation, IEEE Trans. Circuits Syst. Video Technol. 10 (2000) 1006–1011.

[12] M. Bierling, Displacement estimation by hierarchical block matching, in: Proceedings of the SPIE Conference Visual Communication, Image Processing'88, Vol. 1001, November 1988, pp. 942–951.

[13] Y.L. Chan, W.C. Siu, New adaptive pixel decimation for block motion vector estimation, IEEE Trans. Circuits Syst. Video Technol. 6 (1996) 113–118.

[14] B. Liu, A. Zaccaring, New fast algorithms for the estimation of block motion vectors, IEEE Trans. Circuits Syst. Video Technol. 3 (1993) 148–157.

[15] D. Hilbert, Üeber die stetige abbildung einer linie aufein flächenstück, Math. Ann. 38 (1891) 459–460.

**About the Author**—KUO-LIANG CHUNG received the B.S., M.S., and Ph.D. degrees in Computer Science and Information Engineering from National Taiwan University in 1982, 1984, and 1990, respectively. From 1984 to 1986, he was a soldier. From 1986 to 1987, he was a research assistant in the Institute of Information Science, Academic Sinica. He has been a Professor in the Department of Computer Science and Information Engineering at National Taiwan University of Science and Technology since 1995. Now he is the Chairman. Prof. Chung received the Distinguished Professor Award from the Chinese Institute of Engineers in May 2001. He is also an IEEE senior member. Prof. Chung received the Outstanding I.T. Elite Award from the R.O.C. Information Month in November 2003. His research interests include image compression, image processing, video compression, coding theory, and algorithms.

**About the Author**—LUNG-CHUN CHANG received the B.S. degree in Mathematics from Dong-Hai University. Dr. Chang received the Ph.D. degree from National Taiwan University of Science and Technology in 2002. His research interests include image and video processing, video coding, and algorithms.