Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 28782, 14 pages
doi:10.1155/2007/28782

Research Article

Flexible Triangle Search Algorithm for Block-Based Motion Estimation

Mohamed Rehan, Pan Agathoklis, and Andreas Antoniou

Department of Electrical and Computer Engineering, University of Victoria, P.O. Box 3055, Victoria, Canada BC V8W 3P6

Received 5 October 2005; Revised 24 February 2006; Accepted 7 April 2006

Recommended for Publication by Liang-Gee Chen

A new fast algorithm for block-based motion estimation, the flexible triangle search (FTS) algorithm, is presented. The algorithm is based on the simplex method of optimization adapted to an integer grid. The proposed algorithm is highly flexible due to its ability to quickly change its search direction and to move towards the target of the search criterion. It is also capable of increasing or decreasing its search step size to allow coarser or finer search. Unlike other fast search algorithms, the FTS can escape from inferior local minima and thus converge to better solutions. The FTS was implemented as part of the H.264 encoder and was compared with several other block matching algorithms. The results obtained show that the FTS can reduce the number of block matching comparisons by around 30–60% with negligible effect on the image quality and compression ratio.

1. INTRODUCTION

Motion estimation is one of the key components in several video compression algorithms and standards [1–7]. The main purpose of motion estimation is to reduce temporal redundancy between frames in a video sequence. Motion estimation (ME) algorithms can be classified as block-based, pixel-based, or region-based. Block-based algorithms are the most popular due to their implementation simplicity in both software and hardware.

In block-based motion estimation, each frame is divided into a group of equally sized blocks called macroblocks and a single vector is used to represent motion for each macroblock. This motion vector is obtained by finding the best match between the block in the frame to be compressed, called the current frame, and the reference frame. The main parameters of the block-based ME process are the search window size, the matching criterion, and the search algorithm. The search window is the area in the reference frame in which the search for the best matching block is performed. The search window is defined by the location of its origin (its upper-left corner) and its size. The matching criterion is the evaluation function that measures the degree of matching between two blocks. Different matching criteria are available such as the sum of absolute difference (SAD), the cross correlation (CC), and the mean-square error (MSE). SAD is mostly used because of the simplicity and ease of its implementation and is given by

$$\text{SAD}(V_i) = \sum_{x=0}^{M}\sum_{y=0}^{N} \left| S_l(x,y) - S_{l-1}(x+dx, y+dy) \right|, \quad (1)$$

where M and N are the block width and height, respectively, $S_l(x,y)$ is the pixel value of frame l at relative position x, y from the macroblock origin, and $V_i = (d_x, d_y)$ is the displacement vector.

A wide range of block matching algorithms (BMAs) have been presented in the literature [8–30]. The minimum SAD can be achieved by using a full (i.e., exhaustive) search which has the drawback of high computational complexity. This makes full search (FS) not suitable for real-time video compression applications. Other available block matching algorithms apply fast search techniques such as the 2D logarithmic search (2DS) [9], the cross search (CS) [10], the three-step search (TSS) [11], the new three-step search (NTSS) [12], the hierarchical BMA [13], the hexagon search (HS) [14], the diamond search (DS) [15–17], the zonal search [18, 19], and the simplex search (SS) algorithm [26–30]. In these algorithms, only selected subsets of search positions are used for evaluation leading to reduced computation but can lead to motion vectors corresponding to inferior local minima of the matching criterion. The more recent techniques include more advanced features such as early exit based on a certain threshold value or improved prediction for the search center. These techniques include the motion vector field adaptive search technique (MVFAST) [20], the predictive MVFAST (PMVFAST) [21], and the unsymmetrical cross multihexagon grid search (UMHexagonS) [22, 23] which was proposed for the H.264 encoder and was accepted by the Joint Video Team (JVT).

The group of BMAs presented in [26–30] is based on the simplex optimization algorithm and has been found to yield very promising results. The use of the well-known simplex optimization algorithm for finding the minimum SAD is motivated by the fact that the simplex technique has the capacity to quickly change search direction and perform a coarser or finer search as necessary [24, 25].

In this paper, a new fast BMA is developed by adapting the simplex algorithm to a discrete search grid. This algorithm uses predefined sets of triangles to achieve the best match and it is thus called the flexible triangle search (FTS). Through the use of predefined sets of triangles the search operations can be carried out without floating point operations and without having to adapt the triangle obtained at each step of the algorithm to the discrete search grid. An early version of this algorithm was presented in [31]. The paper is organized as follows: in Section 2, a brief description of the concept of the simplex search algorithm is presented and the motivation for the development of the FTS algorithm is discussed. The proposed FTS algorithm is presented in Section 3 and its performance is evaluated and compared with that of other fast BMAs in Section 4.

2. SIMPLEX SEARCH ALGORITHM

The simplex algorithm is a technique used in optimization when the derivatives of the performance index are not available, or difficult to obtain [25]. In the two-dimensional simplex search, a search triangle is used to locate a minimum of the performance index or error function. This error function is evaluated at the triangle vertices which represent possible minimum locations. The locations of the triangle vertices are modified in a manner that moves the triangle towards possible minimum locations by moving the triangle away from locations of high error function values. During these movements, the search triangle undergoes operations such as reflection, expansion, and contraction. These operations provide the necessary flexibility to efficiently move the triangle towards the minimum location or resize the triangle. Consequently, the search can quickly change direction depending on the search results, or become coarser or finer as necessary. The algorithm's main operations can be briefly described as follows.

Reflection

In this operation, the triangle is reflected away from the vertex with the maximum error value. The vertex with the maximum error value is identified and its new location is calculated by reflecting it with respect to the remaining two vertices. If the value of the error function at the vertex after reflection is less than the value of the error function at the location before reflection, then the reflection operation is considered to be successful and a new triangle with the new vertex instead of the maximum-error vertex is obtained. Thus, the triangle is moved in the direction of the minimum error.

Expansion

After a successful reflection, the possibility of finding a vertex with lower error function value can be further investigated by moving the reflection vertex further in the same direction, a process referred to as expansion. If the value of the error function at the vertex obtained after expansion is lower than the error function value at the vertex after reflection, the vertex obtained after expansion is used as the vertex of the search triangle. Thus expansion increases the size of the triangle allowing it to move faster towards the minimum point using a coarser search.

Contraction

The contraction operation is the opposite of expansion. It is used when both reflection and expansion operations fail. In such a case, the search triangle is close to the minimum point and the size of the triangle is reduced to conduct a finer search and find the minimum point. If the algorithm has already reached a sufficiently small triangle size and no more contraction can be achieved, then the algorithm stops.

The ability of the simplex algorithm to change the search direction and to switch between coarse and fine searches makes it a good candidate to be used for BMA and this has been done in [26–30]. However, during implementation of the simplex algorithm for ME, several possibilities for improvement were observed. Most of these are related to the fact that the original simplex algorithm was intended for continuous variables while BMAs are required to use a discrete grid for the variables. The movement of the triangle is therefore not completely controllable. This sometimes results in the collapse of the triangle into a single vertex due to the integer grid approximation, or search locations may be repeatedly evaluated. Further, the simplex algorithm requires many floating-point calculations which slow down the search relative to the searches in other integer-based algorithms.

3. THE FLEXIBLE TRIANGLE SEARCH ALGORITHM

The FTS was developed based on the simplex method and several modifications are introduced to make it more suitable for the discrete grid required for BMAs.

The FTS is based on using sets of triangles of different sizes to perform the search. The FTS algorithm is switching between levels through expansion and contraction operations. The vertices of these triangles are always on the integer grid and the triangles have different sizes to perform coarse or fine searches. A triangle is defined by its identification ID and its level, that is, T21 stands for triangle T, level 2, and

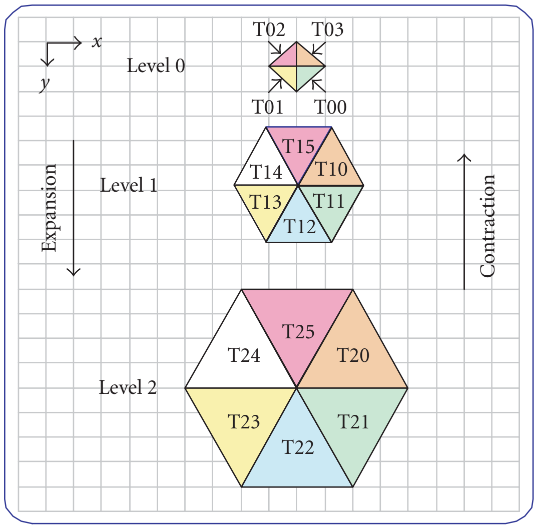


Figure 1: Triangle sets for levels 0 to 2.

ID 1. Figure 1 displays the triangles for levels 0 to 2. The IDs for the three levels are

(i) Level 0 = {T00,T01,T02,T03};
(ii) Level 1 = {T10,T11,T12,T13,T14,T15};
(iii) Level 2 = {T20,T21,T22,T23,T24,T25}.

The vertices of these triangles are denoted as V_0, V_A, V_B, where V_0 is the center point, and V_A and V_B are the vertices in counterclockwise sense from V_0. Thus, the coordinates of the three vertices of a triangle can be obtained from the triangle name and the coordinates of V_0.

More than 3 levels could be used. However, experimental results have shown that 3 to 4 levels are entirely satisfactory for the commonly used window sizes.

Like most other algorithms, the FTS is easily integrated with early termination and motion vector prediction techniques to improve its computational performance. When motion prediction is used, the predictive motion vector is used as the center of the starting triangle group. In addition, an early termination condition based on the SAD value is added to the algorithm.

Based on the above definition of the triangles, the basic operations of the FTS, namely, reflection, expansion, contraction, and translation, can be easily described using look-up tables and can be computed without floating-point operations.

Such a look-up table for reflections and expansions for level 0 triangles is given in Table 1 using the triangle definitions and the origin V_0. The possible reflections and the possible expansions for a level 0 triangle are illustrated in Figures 2 and 3, respectively.

Similar tables can be constructed for reflection and expansion for the other two levels and they are given in the appendix. Through these tables, the FTS algorithm can be implemented using look-up tables and thus its computational efficiency can be greatly increased.

Contraction from level 2 to 1 is straightforward since the triangle orientation does not change. However, contraction from level 1 to level 0 requires some modifications since the number of triangles in level 0 is less than the number of triangles in level 1. Table 2 presents contraction from level 1 to 0.

The FTS algorithm can be described as in Algorithm 1.

The choice of termination condition parameters K max and *ExitSAD* depends on the application and often involves tradeoffs. The value for K max, for example, limits the number of search steps, and thus the amount of computation, but should be higher than the average number of search steps per macroblock to obtain good quality of the reconstructed frames. *ExitSAD* is the minimum SAD value at which the search stops.

The relationship between the FTS operations and triangle levels is illustrated in Figure 4 and the complete flowchart of the algorithm is shown in Figure A.2 in the appendix. The FTS can also be combined with variable block-size mode selection algorithms such as that discussed in [32] to facilitate its use for variable block size motion estimation.

4. ILLUSTRATIVE EXAMPLE

An example of the search pattern using FTS is illustrated in Figure 5. The search starts at the center of the search window and concludes with finding $V_{\min}$ the location with the minimum SAD. The steps involved are as follows.

(1) *Start*

The triangle search starts at level 0; the current triangle is T00 with initial vertices V_1, V_3, and V_2. In this case SAD(V_1) is the maximum and SAD(V_3) is the minimum. Thus, V_1 is set to V_h, V_3 to V_l, and $V_{\min}$ to V_3.

(2) *Reflection*

The triangle vertex V_1 is reflected to V_4. Since SAD$(V_4) <$ SAD(V_1), reflection is successful and should be followed by expansion. The new triangle becomes T02.

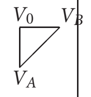
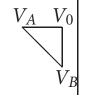
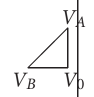
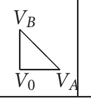
(3) *Expansion*

A test for expansion is performed at vertex V_5 and since SAD$(V_5) <$ SAD(V_4), expansion is successful. The current triangle is then expanded to T14 (based on Table 1) with vertices V_2, V_5, and V_6. V_d is set to $V_e - V_r = (1, 1)$. Since in this case, SAD$(V_5) >$ SAD$(V_{\min})$, $V_{\min}$ will not be updated.

(4) *Translation*

Since the last operation was a successful expansion, translation is attempted. Using the translation vector $V_d = (1, 1)$ from the expansion step, a translation of the current triangle is attempted to V_7, V_8, and V_9. In this triangle, SAD(V_9) is

Table 1: Possible reflections and expansions for Level 0 triangles.

		Results of reflection of V_0 around V_A, V_B		Expansion of V_0 reflection-vertex		Results of reflection of V_A around V_0, V_B		Expansion of V_A reflection-vertex		Results of reflection of V_B around V_0, V_A		Expansion of V_B reflection-vertex	
Current triangle, level 0		New triangle, level 0	Origin shift V_0	Test point V_e	New triangle, level 1	New triangle, level 0	Origin shift V_0	Test point V_e	New triangle, level 1	New triangle, level 0	Origin shift V_0	Test point V_e	New triangle, level 1
T00		T02	(1,1)	(2,2)	T14	T03	(0,0)	$(0,-2)$	T12	T01	(0,0)	$(-2,0)$	T11
T01		T03	$(-1,1)$	$(-2,2)$	T10	T00	(0,0)	(2,0)	T13	T02	(0,0)	$(0,-2)$	T12
T02		T00	$(-1,-1)$	$(-2,-2)$	T11	T01	(0,0)	(0,2)	T15	T03	(0,0)	(2,0)	T14
T03		T01	$(1,-1)$	$(2,-2)$	T13	T02	(0,0)	$(-2,0)$	T10	T00	(0,0)	(0,2)	T15

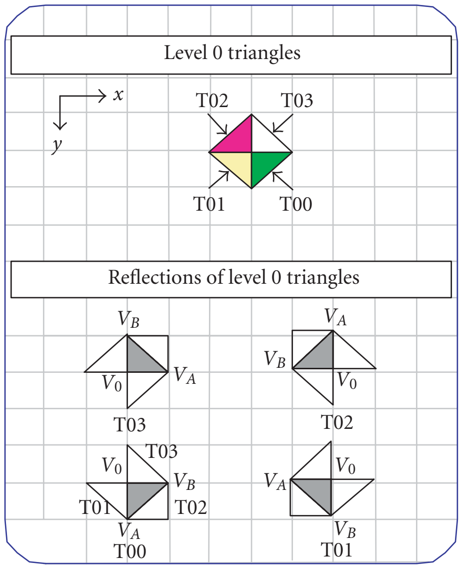


Figure 2: Possible reflections for level 0 triangles. The original triangle is the dark one.

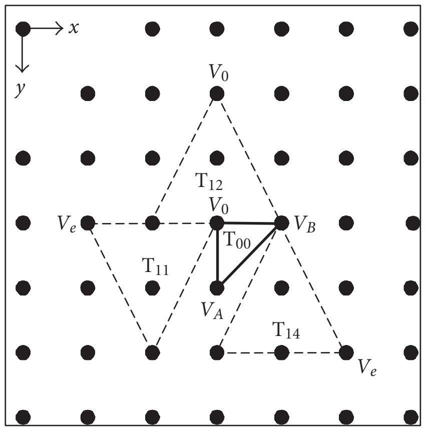


Figure 3: Result of reflection followed by expansion of triangle T00 as outlined in Table 1. The original triangle T00 is shown using a solid line and the resulting level 1 triangles are shown using dotted lines.

the maximum error, $\mathrm{SAD}(V_8)$ is the minimum error and this error is less then $\mathrm{SAD}(V_{\min})$. As a result $V_{\min}$ is updated to V_8. The triangle ID remains T14.

(5) *Reflection*

Since the last operation was a successful translation, another translation is attempted which does not lead to a vertex with a lower error than $\mathrm{SAD}(V_8)$. Thus, reflection is attempted by reflecting V_9 to V_{10}. Since $\mathrm{SAD}(V_{10}) < \mathrm{SAD}(V_9)$, this is a

Table 2: Contraction from level 1 to level 0 triangles.

Level 1, original triangle	Level 0, new triangle
T10	T03
T11	T00
T12	T00
T13	T01
T14	T02
T15	T02

Given a reference frame $S_{l-1}(x, y)$, an $M \times N$ macroblock in the current frame $S_l(x, y)$ finds the displacement vector $V_{\min}$ so that $\mathrm{SAD}(V_{\min})$ in (1) is minimized in the search window. The details of the algorithm are as follows.

Step 1 (initialization). (i) Initialize the current triangle level, current triangle within that set, and initial triangle vertices V_0, V_A, and V_B in the search area. Choose V_0 as the origin of the search window. If motion vector prediction is used, shift V_0 by the predictive motion vector. Initialize the iteration counter $K = 0$ and set translation vector V_d to 0, TranslationFlag to False, and displacement vector $V_{\min}$ to V_0.

Step 2. (i) Check the termination conditions. If any condition is satisfied, then terminate the search.
(ii) Determine the SAD for each new vertex in the current triangle. Identify the vertex with the highest SAD value as V_h, the vertex with the lowest SAD value as V_l, and the vertex with the middle SAD value as V_{mid}.
(iii) If the previous step was a successful expansion or translation operation, go to Step 6, otherwise continue to Step 3.

Step 3 (reflection). (i) Get a new vertex V_r, by reflecting V_h of the current triangle using the table corresponding to the current level, and calculate $\mathrm{SAD}(V_r)$.
(ii) If $\mathrm{SAD}(V_r) < \mathrm{SAD}(V_h)$, go to Step 4, otherwise go to Step 5.

Step 4 (expansion). (i) Locate the expansion vertex V_e for the current triangle using the appropriate triangle level table.
(ii) If $\mathrm{SAD}(V_e) < \mathrm{SAD}(V_r)$, then expansion was successful; increase the triangle level and update the current triangle. Calculate the translation vector between the reflection and expansion vertices as $V_d = V_e - V_r$ and set TranslationFlag to True. If $\mathrm{SAD}(V_e) < \mathrm{SAD}(V_{\min})$, set $V_{\min} = V_e$. Go back to Step 2 with $K = K + 1$.
(iii) If $\mathrm{SAD}(V_e) \geq \mathrm{SAD}(V_r)$, then expansion was not successful. Update the current triangle by replacing V_h by V_r. If $\mathrm{SAD}(V_r) < \mathrm{SAD}(V_{\min})$, set $V_{\min} = V_r$. Go back to Step 2 with $K = K + 1$.

Step 5 (contraction). (i) If the current level is 0, then no more contractions can be done. In this case, terminate the search. Otherwise, contract the triangle by reducing the triangle level. Update the current triangle, set $K = K + 1$, and go to Step 2.

Step 6 (translation). (i) Find a new vertex, V_t, by translating V_l using $V_t = V_l + V_d$ and calculate $\mathrm{SAD}(V_t)$.
(ii) If $\mathrm{SAD}(V_t) < \mathrm{SAD}(V_l)$, then translation was successful; replace V_l by V_t, set $K = K + 1$, and go back to Step 6 if the termination conditions are not met; otherwise stop the search. If $\mathrm{SAD}(V_l) < \mathrm{SAD}(V_{\min})$, set $V_{\min} = V_l$.
(iii) If $\mathrm{SAD}(V_t) \geq \mathrm{SAD}(V_l)$, then translation was not successful; set V_l as the origin of the next search triangle, TranslationFlag to False, and $K = K + 1$. Continue from Step 2.

Termination conditions

The search is terminated if

(i) no more successful contraction operations are possible;
(ii) the number of search iterations reaches a prespecified limit K max;
(iii) the value of SAD becomes less than a prespecified threshold *ExitSAD*.

Algorithm 1

successful reflection. In the reflected triangle $\mathrm{SAD}(V_7)$ is the maximum error. Further, $\mathrm{SAD}(V_{10}) > \mathrm{SAD}(V_8)$ so V_8 remains the minimum point and $V_{\min}$ is not updated. The new triangle becomes T15.

(6) *Reflection*

Expansion is not successful, so reflection is attempted by reflecting V_7 to V_{11}. Since $\mathrm{SAD}(V_{11}) < \mathrm{SAD}(V_8) < \mathrm{SAD}(V_7)$, the reflection was successful and also $V_{\min}$ is updated to V_{11}. The new triangle becomes T12.

(7) *Contraction*

Expansion and reflection are not successful and thus contraction is attempted. Based on Table 2, T12 is contacted to T00. In the new triangle, $\mathrm{SAD}(V_{12})$ is the lowest and is also lower than $\mathrm{SAD}(V_{\min})$. Thus $V_{\min}$ is updated to V_{12}.

(8) *Exit*

An additional reflection does not lead to lower values for SAD. In addition, it is not possible to contract to a lower level. The algorithm will exit with the location of the minimum SAD value in $V_{\min}$.

5. PERFORMANCE ANALYSIS

The FTS was integrated as part of the JVT/H.264 reference encoder. The technique was compared with the NTSS [12], FS, DS [16], and HS [14] algorithms. NTSS is well known for its simplicity while DS and HS are well known for their low computation requirements.

For purposes of comparison, scenes with different kinds of movement have been used. The QCIF (176×144 pixels) and CIF (352×288 pixels) sequences were used. Except

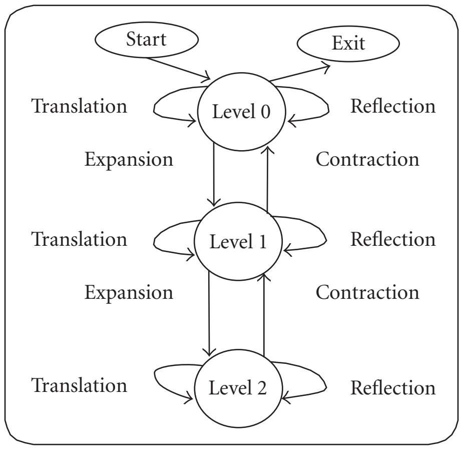


Figure 4: Relation between the FTS operations: reflection, expansion, translation, contraction, and triangle levels.

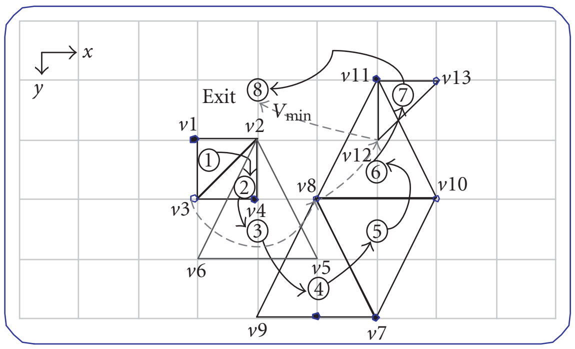


Figure 5: Example of a search pattern using FTS.

for the search algorithm, all other encoding parameters were kept fixed. These parameters include

(i) macroblock size (16 × 16);
(ii) same search area size (16 × 16);
(iii) same rate control algorithm;
(iv) motion vector prediction;
(v) early exit condition parameters K max and *ExitSAD*. In all the simulations, these parameters were chosen to be K max = 25 and *ExitSAD* = 0;
(vi) same number of I and P frames.

The comparison criteria were chosen to be the average number of block matching evaluations to evaluate computational complexity, the compression ratio to evaluate efficiency, and the peak signal-to-noise ratio (PSNR) between the original frames and the reconstructed frames to evaluate quality.

Table 3 lists the average number of block matching comparisons per frame obtained. As can be seen, the average number of block matching comparisons required by the FTS is less than that of the NTSS, FS, DS, or HS. As the average number of block matching comparisons is an indication of the computation complexity, and thus the speed of the algorithm, the results obtained confirmed that the FTS is faster than any of the other three techniques.

The compression ratio results in Table 4 indicate that the FTS produced slightly less compression ratio than the FS and comparable results with those obtained with the DS, HS, and NTSS. In all cases, the difference in compression ratio was within ±1%, which is almost negligible given the reduction achieved in the amount of computation.

The average PSNR results are presented in Tables 5 and 6. As expected, with rate control off all algorithms give similar PSNR values in Table 5. The PSNR values are also very close for all algorithms in Table 6. Figure 6 shows the changes of PSNR at different bit rates while Figure 7 displays the PSNR values for each frame.

From Figure 6, it can be seen that the performance of the FTS is comparable with that of the other algorithms except for the FS. For a qualitative comparison, Figure A.1 shows frames reconstructed from compressed ones using the different BMAs. The visual difference is hardly noticeable. It can be inferred from Tables 5 and 6 and Figures 6 and 7 that the PSNR values obtained using the FTS are comparable with those of the NTSS, DS, and HS and are very close to those of the FS.

From the above comparison, it is clear that the compression ratios, as well as the average PSNR and visual quality of the reconstructed frames using the FTS, NTSS, DS, HS, and FS, are not significantly different. This indicates that the significant reduction in the computational complexity obtained using the FTS did not come at the expense of deterioration in visual quality or compression efficiency.

6. DISCUSSION

The simulation results showed that the FTS is capable of producing almost the same compression ratio and PSNR results as other fast block matching algorithms while reducing the number of block matching computations by around 30–60% depending on the video sequence (Table 7). Further, results indicate that the FTS works well for both slow and fast sequences (see Figure 8). This promising performance of the FTS motivated the implementation of FTS using FPGAs which will be presented in [33].

The performance improvements of the FTS over existing algorithms are related to the following features of the algorithm.

(i) Each reflection operation moves the triangle away from positions of large error using only one SAD calculation while most other fast algorithms require several SAD calculations for each search iteration.

Table 3: Average number of block matching per macroblock.

Sequence	FS	NTSS	DS	HS	FTS
QCIF resolution (176 × 144)					
Miss America	1089	20.00	14.76	12.65	9.04
Akyio	1089	17.34	12.24	11.26	6.51
News	1089	17.49	12.42	11.35	6.83
Silent	1089	17.67	12.59	11.46	7.23
Coastguard	1089	18.15	12.70	11.73	7.37
Foreman	1089	19.11	13.62	12.26	8.20
Carphone	1089	19.33	14.11	12.60	8.44
Stefan	1089	19.03	13.6	12.36	8.33
CIF resolution (352 × 288)					
Coastguard	1089	18.38	12.9	11.86	7.37
Container	1089	18.69	13.55	12.02	7.32
Foreman	1089	20.37	15.09	13.36	9.32
Paris	1089	17.73	12.59	11.52	7.06
Stefan	1089	21.07	15.60	13.69	9.30

Table 4: Compression ratio (no rate control).

Sequence	FS	NTSS	DS	HS	FTS
QCIF resolution (176 × 144)					
Miss America	279.67	279.19	276.33	278.67	280.37
Akyio	312.43	312.50	312.91	313.16	313.00
News	105.21	105.14	105.17	105.34	104.85
Silent	91.64	91.39	91.60	91.40	91.38
Coastguard	38.54	38.50	38.50	38.51	38.42
Foreman	54.42	54.82	54.60	54.49	54.70
Carphone	49.90	49.91	49.86	49.75	49.89
Stefan	13.84	13.80	13.80	13.77	13.70
CIF resolution (352 × 288)					
Coastguard	413.36	413.69	414.13	413.40	413.63
Container	31.94	31.97	31.96	31.97	31.96
Foreman	174.38	173.47	173.88	173.54	173.34
Paris	68.61	68.20	67.95	67.03	67.32
Stefan	64.37	64.24	64.22	64.15	64.04

Table 5: Average PSNR (no rate control).

Sequence	FS	NTSS	DS	HS	FTS
QCIF resolution (176 × 144)					
Miss America	39.62	39.63	39.63	39.64	39.61
Akyio	37.80	37.79	37.77	37.77	37.77
News	36.26	36.26	36.23	36.24	36.25
Silent	35.45	35.46	35.47	35.47	35.47
Coastguard	33.85	33.86	33.86	33.85	33.86
Foreman	34.93	34.91	34.90	34.90	34.90
Carphone	36.03	36.02	36.01	35.99	35.98
Stefan	33.76	33.76	33.76	33.75	33.76
CIF resolution (352 × 288)					
Coastguard	39.31	39.30	39.31	39.31	39.32
Container	34.32	34.30	34.3	34.30	34.29
Foreman	35.54	35.53	35.53	35.53	35.52
Paris	35.89	35.89	35.87	35.86	35.87
Stefan	35.11	35.12	35.11	35.12	35.12

Table 6: Average PSNR (with rate control enabled).

Sequence	FS	NTSS	DS	HS	FTS
QCIF resolution (176 × 144)					
Miss America	44.16	44.14	44.12	44.14	44.15
Akyio	45.39	45.40	45.41	45.38	45.44
News	39.14	39.14	39.13	39.14	39.21
Silent	37.96	37.98	37.95	37.96	37.94
Coastguard	31.51	31.49	31.48	31.49	31.47
Foreman	36.84	36.85	36.84	36.83	36.85
Carphone	38.16	38.13	38.14	38.13	38.13
Stefan	28.55	28.50	28.51	28.50	28.46
CIF resolution (352 × 288)					
Coastguard	43.36	43.35	43.36	43.35	43.33
Container	32.61	32.61	32.60	32.60	32.59
Foreman	36.46	36.46	36.45	36.46	36.44
Paris	35.82	35.77	35.74	35.68	35.72
Stefan	35.07	35.08	35.01	35.04	34.97

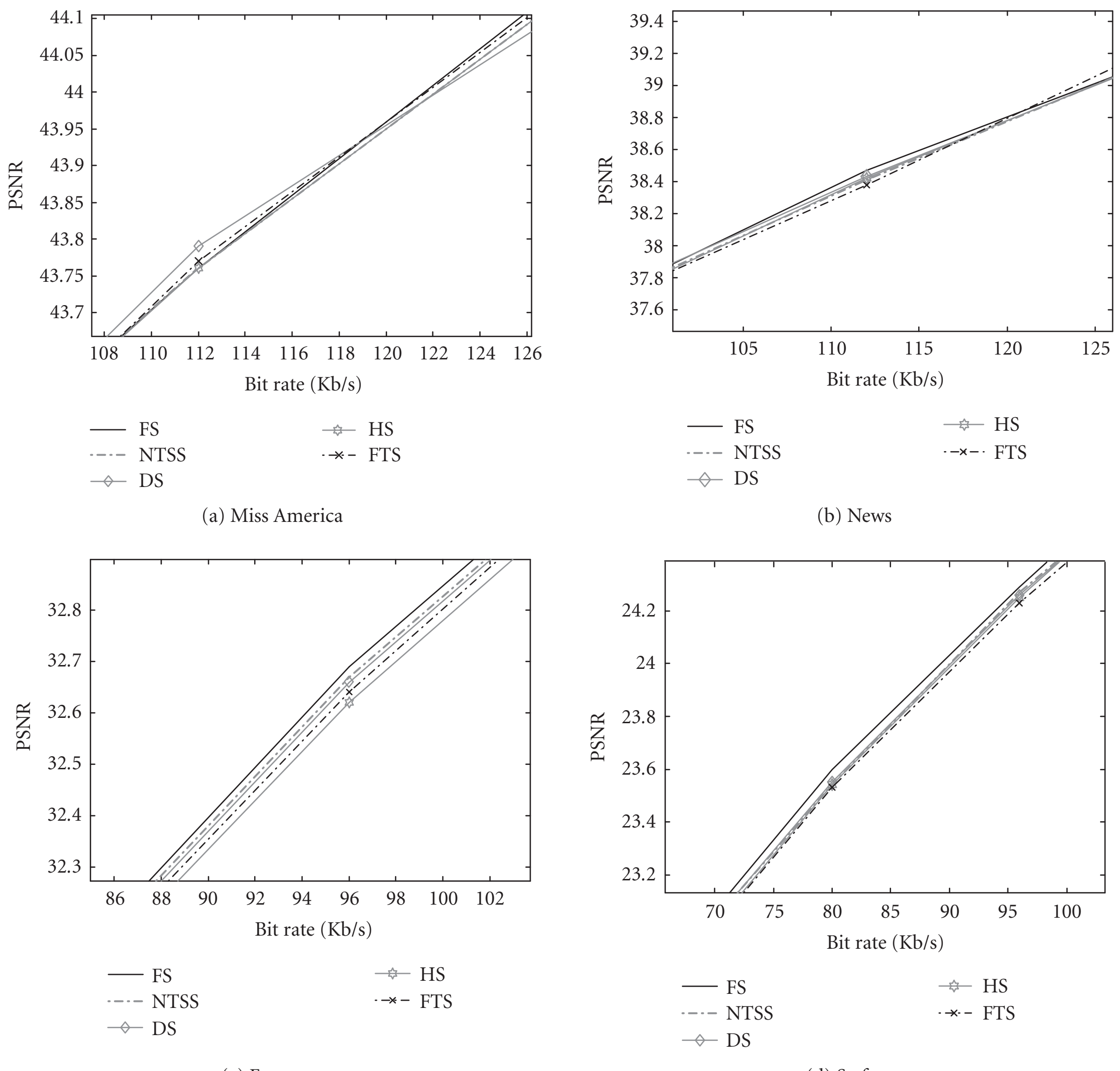
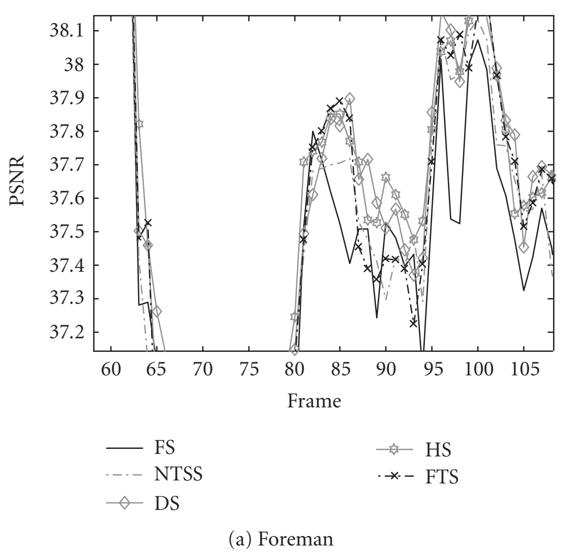


Figure 6: PSNR versus bit rate.



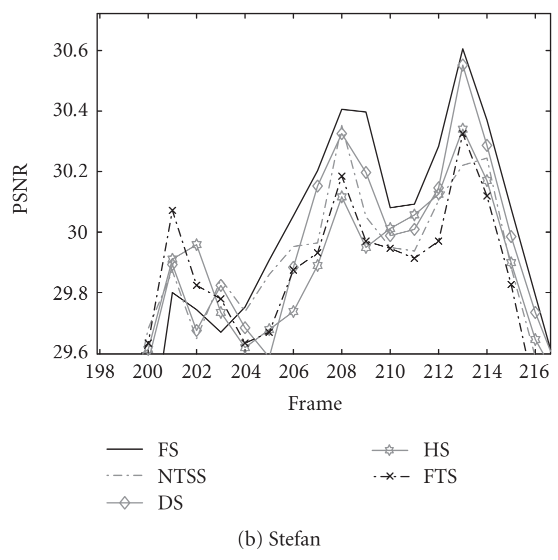


Figure 7: PSNR value per frame.

Table 7: FTS computational improvement percentage.

Sequence	Number of block matching	Percentage improvement with respect to other algorithms		
	FTS	NTSS	DS	HS
Miss America	9.04	54.80%	38.75%	28.54%
Akyio	6.51	62.46%	46.81%	42.18%
News	6.83	60.95%	45.01%	39.82%
Silent	7.23	59.08%	42.57%	36.91%
Coastguard	7.37	59.39%	41.97%	37.17%
Foreman	8.20	57.09%	39.79%	33.12%
Carphone	8.44	56.34%	40.18%	33.02%
Stefan	8.33	56.23%	38.75%	32.61%

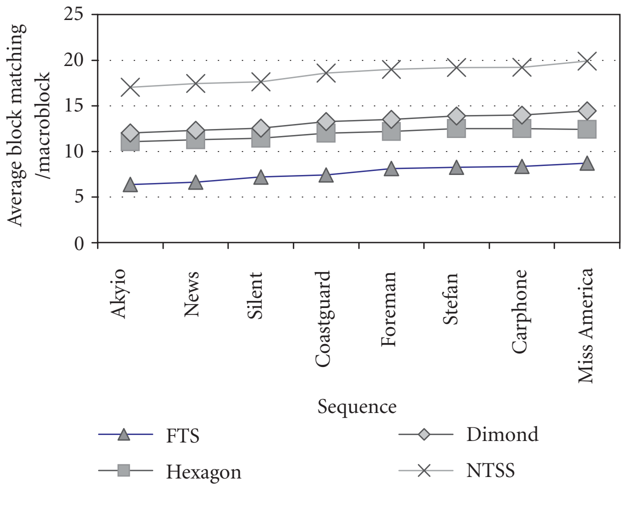


Figure 8: Average number of block matching per macroblock for each algorithm.

(ii) Expansion operations speed up the search by increasing the search step and thus avoiding unnecessary intermediate SAD calculations. The contraction operation reduces the search step to achieve a higher resolution.

(iii) The translation operation is useful when a change of location is detected during the search. The FTS uses the translation operation if a successful expansion follows a successful reflection in which case the chosen direction of expansion may yield a better minimum point.

These operations provide the FTS with excellent search flexibility. Further, the reflection operation can help to avoid local minima.

The use of predefined triangle sets, as shown in Figure 1, leads to the following important features of the FTS.

(i) FTS is optimized for an integer grid and for performing all operations using integer calculations.

Table A.1: Possible reflections and expansions of level 1 triangles.

		Results of V_0 reflection around V_A, V_B		Expansion of V_0 reflection-vertex		Results of V_A reflection around V_0, V_B		Expansion of V_A reflection-vertex		Results of V_B reflection around V_0, V_A		Expansion of V_B reflection-vertex	
Current triangle, level 1		New triangle, level 1	Origin shift V_0	Test point V_e	New triangle, level 2	New triangle, level 1	Origin shift V_0	Test point V_e	New triangle, level 2	New triangle, level 1	Origin shift V_0	Test point V_e	New triangle, level 2
T10	$\overset{V_B}{V_0 \triangle V_A}$	T13	$(3,-2)$	$(5,-3)$	T23	T15	(0,0)	$(-3,-3)$	T25	T11	(0,0)	(1,4)	T21
T11	$\underset{V_A}{V_0 \triangledown V_B}$	T14	(3,2)	(5,3)	T24	T10	(0,0)	$(1,-4)$	T20	T12	(0,0)	$(-3,3)$	T22
T12	$\overset{V_0}{V_A \triangle V_B}$	T15	(0,4)	(0,6)	T25	T11	(0,0)	$(4,-1)$	T21	T13	(0,0)	$(-4,-1)$	T23
T13	$\underset{V_B}{V_A \triangledown V_0}$	T10	$(-3,2)$	$(-5,3)$	T20	T12	(0,0)	(3,3)	T22	T14	(0,0)	$(-1,-4)$	T24
T14	$\overset{V_B}{V_0 \triangle V_A}$	T11	$(-3,-2)$	$(-5,-3)$	T21	T13	(0,0)	$(-1,4)$	T23	T15	(0,0)	$(3,-3)$	T25
T15	$\underset{V_0}{V_B \triangledown V_A}$	T12	$(0,-4)$	$(0,-6)$	T22	T14	(0,0)	$(-4,1)$	T24	T10	(0,0)	(4,1)	T20

(ii) Most of the calculations are predefined in look-up tables, which can easily be accessed during the search process. This comes at a price of having to store approximately 200 8-bit numbers, which is very insignificant given current advances in memory sizes and the total memory needed by the encoder.

(iii) Search triangles are predefined and, consequently, it is not possible for two or three vertices to coincide and thus, the triangle cannot collapse to a line or point.

(iv) The flow control and exit conditions of the FTS are robust.

7. CONCLUSIONS

A new block matching technique referred to as the FTS has been introduced. This new technique is based on the simplex optimization technique and is adapted for a discrete grid. The FTS uses a set of triangles of different sizes to perform the operations of reflection, expansion, contraction, and translation. These operations enable the FTS to quickly change the search direction, switch between coarser and finer searches, and quickly move towards a minimum point. The proposed technique has been implemented as part of an H.264 encoder and compared with some other popular BMA algorithms. Results indicate that the proposed technique requires significantly fewer block matches than other fast BMA algorithms without any reduction in the compression ratio or deterioration of the visual quality of the reconstructed frames.

APPENDIX

Tables A.1 and A.2 present all posible reflections and expansions for level 1 and 2 triangles, respectively.

In Figure A.1, the original frame 255 of the 'Foreman' sequence and the reconstructed frames using different algorithms are presented for comparison. A flowgraph of the FTS algorithm is illustrated in Figure A.2.

Table A.2: Possible reflections and expansions of level 2 triangles.

		Results of V_0 reflection around V_A, V_B		Expansion of V_0 reflection-vertex		Results of V_A reflection around V_0, V_B		Expansion of V_A reflection-vertex		Results of V_B reflection around V_0, V_A		Expansion of V_B reflection-vertex	
Current triangle, level 2		New triangle, level 2	Origin shift V_0	Test point V_e	New triangle, level 3	New triangle, level 2	Origin shift V_0	Test point V_e	New triangle, level 3	New triangle, level 2	Origin shift V_0	Test point V_e	New triangle, level 3
T20	$\overset{V_B}{V_0 \triangle V_A}$	T23	$(6,-4)$	$(8,-5)$	T33	T25	(0,0)	$(-4,-4)$	T35	T21	(0,0)	(2,5)	T31
T21	$\underset{V_A}{V_0 \triangledown V_B}$	T24	(6,4)	(8,5)	T34	T20	(0,0)	$(2,-5)$	T30	T22	(0,0)	$(-4,4)$	T32
T22	$\overset{V_0}{V_A \triangle V_B}$	T25	(0,8)	(0,9)	T35	T21	(0,0)	(6,2)	T31	T23	(0,0)	$(-6,-2)$	T33
T23	$\underset{V_B}{V_A \triangledown V_0}$	T20	$(-6,4)$	$(-8,5)$	T30	T22	(0,0)	(4,4)	T32	T24	(0,0)	$(-2,-5)$	T34
T24	$\overset{V_B}{V_0 \triangle V_A}$	T21	$(-6,-4)$	$(-8,-5)$	T31	T23	(0,0)	$(-2,5)$	T33	T25	(0,0)	$(4,-4)$	T35
T25	$\underset{V_0}{V_B \triangledown V_A}$	T22	$(0,-8)$	$(-8,5)$	T32	T24	(0,0)	$(-6,2)$	T34	T20	(0,0)	$(6,-2)$	T30

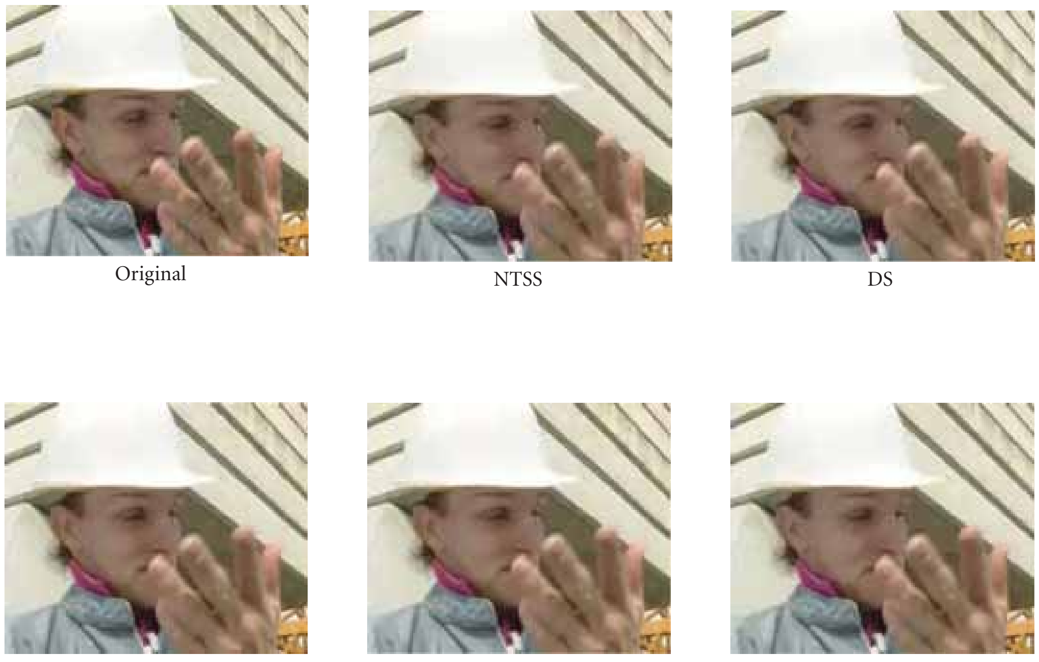


Figure A.1: Reconstructed frame number 255 for Foreman QCIF.

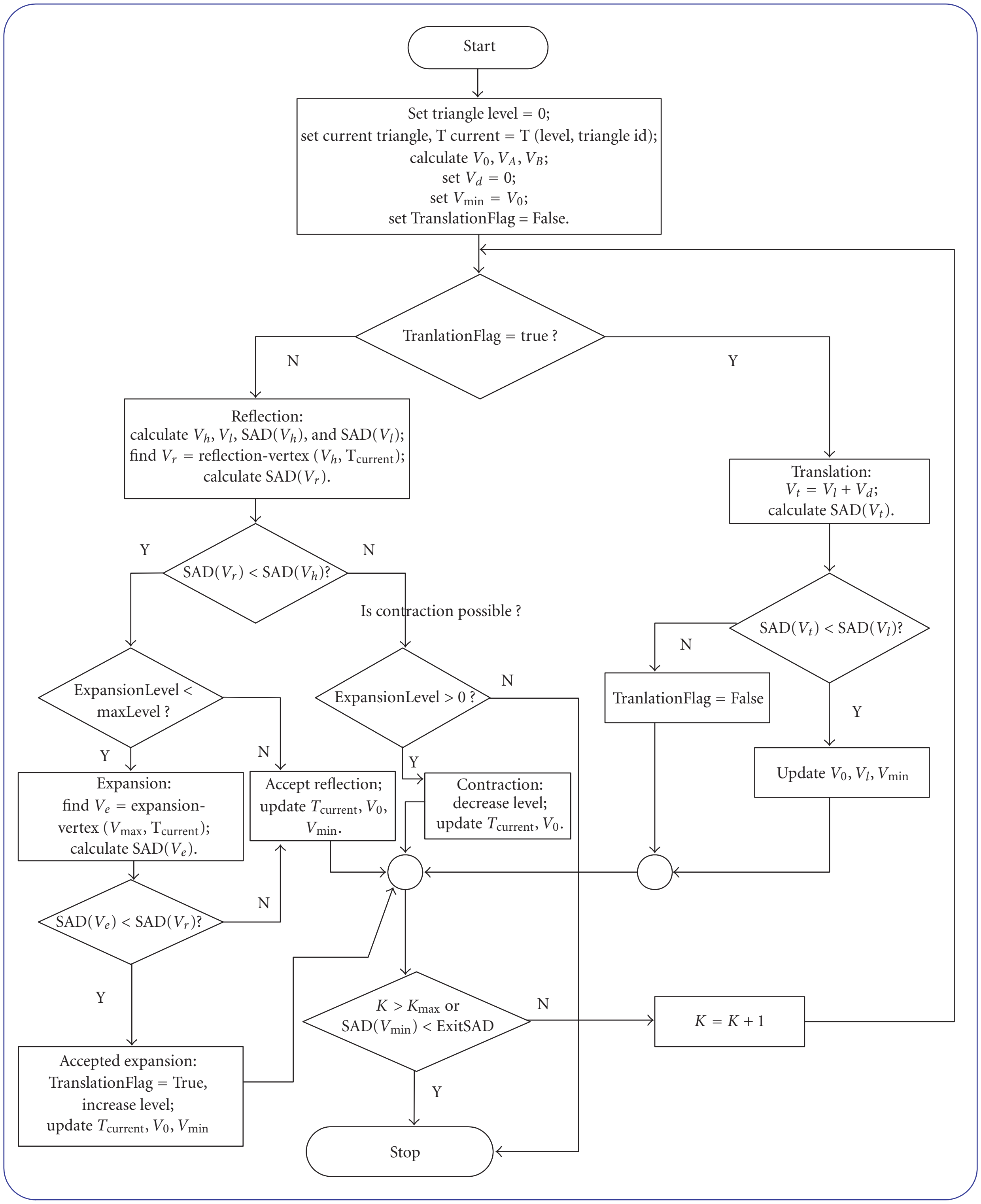

Start
Set triangle level = 0;
set current triangle, T current = T (level, triangle id);
calculate V_0, V_A, V_B;
set $V_d = 0$;
set $V_{\min} = V_0$;
set TranslationFlag = False.
TranlationFlag = true ?
N
Y
Reflection:
calculate V_h, V_l, SAD(V_h), and SAD(V_l);
find V_r = reflection-vertex (V_h, T_{current});
calculate SAD(V_r).
Translation:
$V_t = V_l + V_d$;
calculate SAD(V_t).
SAD(V_r) < SAD(V_h)?
Is contraction possible ?
SAD(V_t) < SAD(V_l)?
ExpansionLevel < maxLevel ?
ExpansionLevel > 0 ?
TranlationFlag = False
Expansion:
find V_e = expansion-vertex ($V_{\max}$, T_{current});
calculate SAD(V_e).
Accept reflection;
update T_{current}, V_0, $V_{\min}$.
Contraction:
decrease level;
update T_{current}, V_0.
Update V_0, V_l, $V_{\min}$
SAD(V_e) < SAD(V_r)?
$K > K_{\max}$ or SAD($V_{\min}$) < ExitSAD
$K = K + 1$
Accepted expansion:
TranslationFlag = True,
increase level;
update T_{current}, V_0, $V_{\min}$
Stop

FIGURE A.2: Flowgraph of the FTS algorithm.

REFERENCES

[1] ISO/IEC 11172, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s," International Organization for Standardization, 1992.

[2] ISO/IEC CD 13818, "Generic Coding of Moving Pictures and Associated Audio," International Organization for Standardization, 1994.

[3] D. E. Le Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.

[4] D. J. Le Gall, "The MPEG video compression algorithm," *Signal Processing: Image Communication*, vol. 4, no. 2, pp. 129–140, 1992.

[5] G. Morrison, "Video coding standards for multimedia: JPEG, H.261, MPEG," in *IEE Colloquium on Technology Support of Multimedia, Digest no. 088*, pp. 2.1–2.4, London, UK, April 1992.

[6] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards Algorithms and Architectures*, Kluwer Academic, Boston, Mass, USA, 1995.

[7] P. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwer Academic, Boston, Mass, USA, 1999.

[8] H. G. Musmann, P. Pirsch, and H.-J. Grallert, "Advances in picture coding," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 523–548, 1985.

[9] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. 29, no. 12, pp. 1799–1808, 1981.

[10] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Transactions on Communications*, vol. 38, no. 7, pp. 950–953, 1990.

[11] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proceedings of National Telecommunications Conference (NTC '81)*, vol. 4, pp. G5.3.1–G5.3.5, New Orleans, La, USA, November 1981.

[12] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[13] B. Paul and E. Viscito, "Hierarchical motion estimation with 2-scale tilings," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '94)*, vol. 3, pp. 260–264, Austin, Tex, USA, November 1994.

[14] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 349–355, 2002.

[15] C.-H. Cheung and L.-M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1168–1177, 2002.

[16] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

[17] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369–377, 1998.

[18] A. M. Tourapis, Q. C. Au, M. L. Liou, and G. Shen, "Fast and efficient motion estimation using diamond zonal-based algorithms," *Journal of Circuits, Systems, and Signal Processing*, vol. 20, no. 2, pp. 233–251, 2001.

[19] A. M. Tourapis, O. C. Au, and M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 934–947, 2002.

[20] P. I. Hosur and K. K. Ma, "Motion vector field adaptive fast motion estimation," in *Proceedings of the 2nd International Conference on Information, Communications and Signal Processing (ICICS '99)*, Singapore, Republic of Singapore, December 1999.

[21] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive motion vector field adaptive search technique (PMVFAST): enhancing block-based motion estimation," in *Visual Communications and Image Processing (VCIP '01)*, vol. 4310 of *Proceedings of SPIE*, pp. 883–892, San Jose, Calif, USA, January 2001.

[22] Z. Chen, P. Zhou, and Y. He, "Fast integer pel and fractional pel motion estimation for JVT," in *JVT-F017r1.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 6th Meeting*, Awaji Island, Japan, December 2002.

[23] Z. Chen, P. Zhou, and Y. He, "Fast motion estimation for JVT," in *JVT-G016.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 7th Meeting*, Pattya II, Thailand, March 2003.

[24] D. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill, New York, NY, USA, 1972.

[25] B. Bunday, *Basic Optimization Methods*, Edward Arnold, London, UK, 1984.

[26] M. Rehan, A. Antoniou, and P. Agathoklis, "A new fast block matching algorithm using the simplex technique," in *Proceedings of the IEEE Symposium on Advances in Digital Filtering and Signal Processing*, pp. 30–33, Victoria, BC, Canada, June 1998.

[27] M. E. Al-Mualla, C. N. Canagarajah, and D. R. Bull, "A simplex minimization for single- and multiple-reference motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1209–1220, 2001.

[28] M. E. Al-Mualla, C. N. Canagarajah, and D. R. Bull, "Simplex minimisation for multiple-reference motion estimation," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '00)*, vol. 4, pp. 733–736, Geneva, Switzerland, May 2000.

[29] M. E. Al-Mualla, C. N. Canagarajah, and D. R. Bull, "Simplex minimisation for fast long-term memory motion estimation," *Electronics Letters*, vol. 37, no. 5, pp. 290–292, 2001.

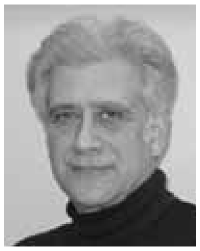
[30] M. E. Al-Mualla, N. Canagarajah, and D. R. Bull, "Simplex minimisation for fast block matching motion estimation," *Electronics Letters*, vol. 34, no. 4, pp. 351–352, 1998.

[31] M. Rehan, P. Agathoklis, and A. Antoniou, "Flexible triangle search algorithm for block based motion estimation," in *Proceedings of the IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing (PACRIM '03)*, vol. 1, pp. 233–236, Victoria, BC, Canada, August 2003.

[32] L. Yang, K. Yu, J. Li, and S. Li, "An effective variable block-size early termination algorithm for H.264 video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 6, pp. 784–788, 2005.

[33] M. Rehan, M. W. El-Kharashi, P. Agathoklis, and F. Gebali, "An FPGA implementation of block based motion estimation using the flexible triangle search algorithm," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 521–524, Island of Kos, Greece, May 2006.

Mohamed Rehan received his B.S. degree with honor in communications and his M.S. degree in computer graphics in 1991 and 1994, respectively, from the Department of Electronics and Communications Engineering, Cairo University, Cairo, Egypt. He is currently working towards his Ph.D. degree. He has been actively involved in the research and development of several projects related to video compression standards including H.263, H.263p, MPEG 1, 2, 4, and H.264.



Pan Agathoklis received the Dipl. Ing. degree in electrical engineering and the Dr.Sc.Tech. degree from the Swiss Federal Institute of Technology, Zurich, Switzerland, in 1975 and 1980, respectively. From 1981 until 1983, he was with the University of Calgary as a Post-Doctoral Fellow and part- time Instructor. Since 1983, he has been with the Department of Electrical and Computer Engineering, University of Victoria, BC, Canada, where he is currently a Professor. He has received a NSERC University Research Fellowship and Visiting Fellowships from the Swiss Federal Institute of Technology, from the Australian National University and the University of Perth, Australia. He has been member of the Technical Program Committee in many international conferences and has served as the Technical Program Chair of the 1991 IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing and the 1998 IEEE Symposium on Advances in Digital Filtering and Signal processing. His fields of interest are in digital signal processing and its applications in control systems and communications.

Andreas Antoniou received his Ph.D. degree in Electrical Engineering from the University of London, UK, in 1966 and is a Fellow of the IEE and IEEE. He served as the founding Chair of the Department of Electrical and Computer Engineering at the University of Victoria, BC, Canada, and is now Professor Emeritus in the same department. He is the author of Digital Filters: Analysis, Design, and Applications (McGraw-Hill, 1993) and Digital Signal Processing: Signals, Systems, and Filters (McGraw-Hill, 2005). He served as Associate Editor/Editor of IEEE Transactions on Circuits and Systems from June 1983 to May 1987, as a Distinguished Lecturer of the IEEE Signal Processing Society in 2003, as General Chair of the 2004 International Symposium on Circuits and Systems, and is currently serving as a Distinguished Lecturer of the IEEE Circuits and Systems Society. He received the Ambrose Fleming Premium for 1964 from the IEE (best paper award), the CAS Golden Jubilee Medal from the IEEE Circuits and Systems Society, the BC Science Council Chairman's Award for Career Achievement for 2000, the Doctor Honoris Causa degree from the Metsovio National Technical University of Athens, Greece, in 2002, and the IEEE Circuits and Systems Society 2005 Technical Achievement Award.