INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 N4344

Sydney, July 2001

Title: Text of 14496-7 PDTR

Source: Video Group

Optimization Model

Version 3.0

July 2001

Contents

Foreword

to be provided by ISO CS

Information technology — Coding of audio-visual objects — Part 7: Optimised software for MPEG-4 visual tools

1. Fast Motion Estimation

1.1 Introduction to Motion Adaptive Fast Motion Estimation

The optimization of fast motion estimation is essentially a multi-dimensional problem. The key dimensions concerned in this problem are: Rate, Quality (PSNR), Speed-up (or Computational Gain), Algorithmic Complexity, Memory Size and Memory Bandwidth (see Fig. 1). There always exists a trade-off among all these five key dimensions. Therefore, it is highly desirable to have an adaptive fast motion estimation core algorithm with scalable structure, which can be adaptively optimized with respect to all or selected aspects for various coding environment and requirements. Since the rate control is used to fix the bit-rate, the optimization problem is reduced by one dimension to four dimensions.

Motion Vector Field Adaptive Search Technique (MVFAST) [1,2,3] is a generic algorithm of the family of *motion-adaptive* fast search techniques, originally proposed by Kai-Kuang Ma and Prabhudev Irappa Hosur from Nanyang Technological University (NTU), Singapore. The MVFAST (M5453, M5851) offers high performance both in quality and speed and does not require memory to store the searched points and motion vectors. The MVFAST has been adopted by MPEG-4 Part 7 in the Noordwijkerhout MPEG meeting (March 2000) as the *core technology* for fast motion estimation.

A derivative of MVFAST, called *Predictive* MVFAST (PMVFAST) [4], is considered as an *optional approach* that might benefit in special coding situations. PMVFAST incorporates a set of thresholds into MVFAST to trade higher speed-up at the cost of memory size, memory bandwidth and additional algorithmic complexity. In PMVFAST, the threshold values are adjusted based on the 54 test cases specified by MPEG-4. However, the coding performance and sensitivity of PMVFAST using these thresholds for the video sequences and encoding conditions outside the MPEG-4 test set has *not* been studied and verified.

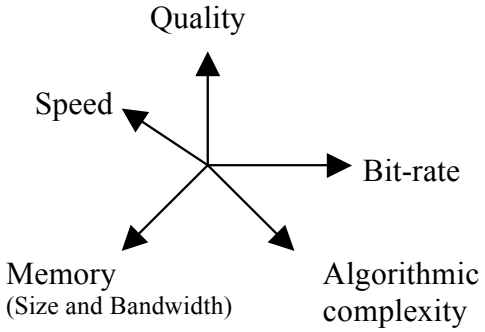


Figure 1.1: Five dimensional optimization problem of fast motion estimation

1.2 Technical Description of Core Technology —MVFAST

1.2.1 Detection of stationary blocks

A large number of MBs in the video sequences (e.g., “talking head” video sequences) with low-motion content tend to have motion vectors equal to (0,0). Such MBs in the regions of no-motion activity can be detected simply based on the sum of absolute difference (SAD) at the origin. Therefore, we exploit an optional phase, called *early elimination of search*, as the first step in MVFAST as follows. The search for a MB will be terminated immediately, if its SAD value obtained at (0,0) is less than a threshold T, and the motion vector is assigned as (0,0). Through extensive simulations, we found that among those zero-motion blocks identified, about 98% of them have their SAD at position (0,0) less than 512. Hence, we choose T = 512 to

enable the mechanism of early elimination of search. Since this early elimination of search phase is optional, it can be turned off or disabled by imposing $T = 0$.

1.2.2 Determination of local motion activity

The *local motion vector field* at a macroblock (MB) position is defined as the set of motion vectors in the *region of support* (ROS) of that MB. The ROS of a MB includes the n neighborhood MBs. In MVFAST, the ROS with $n = 3$ is shown in Fig. 2. Let $V=\{V_0, V_1, ...V_n\}$, where $V_0 = (0,0)$, and V_i (and $i \neq 0$) is the motion vector of MB_i in the ROS (see Fig. 3). The cityblock length of $V_i=(x_i, y_i)$ is defined as $l_{vi} = |x_i| + |y_i|$. Let $L = MAX\{l_{vi}\}$ for all V_i . The motion activity at the current MB position is defined as follows.

$$\begin{aligned} \text{Motion Activity} &= \text{Low, if } L \leq L_1; \\ &= \text{Medium, if } L_1 < L \leq L_2; \\ &= \text{High, if } L > L_2; \end{aligned} \qquad (1)$$

where L_1 and L_2 are integer constants. We choose L_1 and L_2 as the cityblock distance from the center point of the pattern to any other point on the small and large search patterns (see Fig. 4), respectively. Thus, $L_1 = 1$ and $L_2 = 2$.

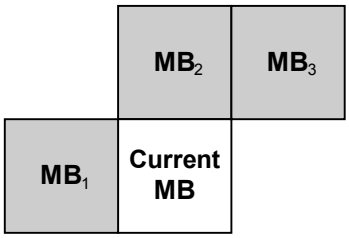


Figure 1.2: Region of support (ROS) for the current MB consists of MB_1, MB_2 and MB_3.

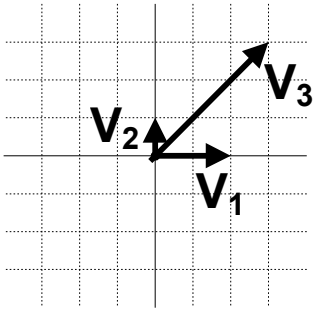


Figure 1.3: Example of distribution of motion vectors belonging to set V. In this case, $l_{v1} = 2$, $l_{v2} = 1$, $l_{v3} = 6$; thus $L = MAX\{l_{v1}, l_{v2}, l_{v3}\} = 6$.

1.2.3 Search Center

The choice of the search center depends on the local motion activity at the current MB position. If the motion activity is low or medium, the search center is the origin. Otherwise, the vector belonging to set V that yields the minimum sum of absolute difference (SAD) is chosen as the search center.

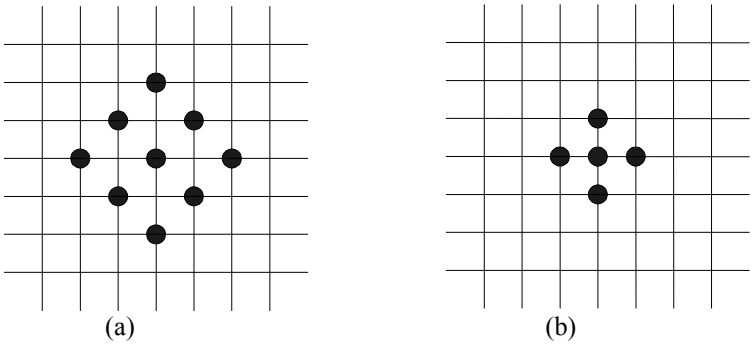


Figure 1.4: (a) Large Diamond Search Pattern (LDSP) and (b) Small Diamond Search Pattern (SDSP).

1.2.4 Search Strategy

A local search is performed around the search center to obtain the motion vector for the current MB. The search patterns employed for the local search are shown in Fig. 4. Two strategies are proposed for the local search and their choice depends on the motion activity identified. If the motion activity is low or high, we employ small diamond search (SDS). Otherwise, we choose large diamond search (LDS).

i) *Small Diamond Search* (SDS)

Step 1: Small diamond search pattern (SDSP) is centered at the search center, and all the checking points of SDSP are tested. If the center position yields the minimum SAD (i.e., no motion), then the center represents the motion vector; otherwise, go to Step 2.

Step 2: The center of SDSP moves to the point where the minimum SAD was obtained in the previous step, and all the points on SDSP are tested. If the center position yields the minimum SAD, then the center represents the motion vector; otherwise, recursively repeat this step.

ii) *Large Diamond Search* (LDS)

Step 1: Large diamond search pattern (LDSP) is centered at the search center, and all the checking points of LDSP are tested. If the center position gives the minimum SAD, go to Step 3; otherwise, go to Step 2.

Step 2: The center of LDSP moves to the point where the minimum SAD was obtained in the previous step, and all the points on LDSP are tested. If the center position gives the minimum SAD, go to Step 3; otherwise, recursively repeat this step.

Step 3: Switch the search pattern from LDSP to SDSP. The point that yields the minimum SAD, is the final solution of the motion vector.

Table 1 summarizes the methodology for selection of search center and search strategy depending on the motion activity at the current MB position.

Table 1: The search modes for MVFAST

Motion Activity	Search Center	Search Strategy
Low	Origin	SDS
Medium	Origin	LDS
High	The position of the vector in set V that yields minimum SAD	SDS

1.2.5 Perspectives on implementing MVFAST

The MVFAST algorithm can be structured in terms of *profiles*. The MVFAST itself as described above can be viewed as the **main profile**. The low, medium and high motion activity cases in Table 1 can be considered individually as three other different profiles of MVFAST. Depending on the video coding applications, any one of these individual profiles can be turned "ON" simply by adjusting the two parameters, L_1 and L_2, in Equation (1). If we set $L_1 = L_2$ = Search Range, we obtain "low motion activity" profile. The "medium motion activity" profile (which is the same as Diamond Search, as described in VM Version 14) can be obtained, if we set $L_1 = -1$ and L_2 = Search Range. For "high motion activity" profile, we can set $L_1 = L_2 = -1$. Note that in this case, Search Range = 2*N, if the search in either coordinate is in the range [–N, N-1].

Although MVFAST is implemented in an intelligent way such that the overlap of search points is minimized when the search pattern moves, few search points are visited more than once. This overlap can be avoided by keeping the record of all the search points visited and testing if the current search point is visited earlier. Thus further improvement over speed-up can be achieved.

The search point (0,0) is always tested in MVFAST. However, some improvement in computational gain is obtained by testing (0,0) point only, if any of the motion vectors in the ROS has motion vector = (0,0).

Through extensive experiments using MVFAST, it is found that further improvement in objective quality can be achieved when interlaced CCIR sequences with high global motion are coded in progressive mode, by including the motion vector of

collocated block on the previously coded non-intra frame in the set *V*. During the motion estimation of interlaced pictures, each frame prediction of macroblock motion is performed before field motion estimation. Therefore, for field motion estimation of current macroblock, its frame motion vector is included in set *V*.

From hardware implementation viewpoint, to restrict the total number of search points for a block in the worst case to be *N*, an additional stopping criterion — "stop the search when the number of search points visited so far is equal to *N*", can be included in SDS and LDS given in Section 2.4.

1.2.6 Special Acknowledgements

Kai-Kuang Ma and Prabhudev Irappa Hosur would like to sincerely acknowledge tremendous support from Professor Meng Hwa Er, Dean, School of Electrical and Electronic Engineering, and Deputy President of Nanyang Technological University, Singapore, who plays a vital role on promoting and directing all Singapore MPEG activities. For independent verification efforts, the following individuals are greatly acknowledged: Dr. Weisi Lin, Mr. Chengyu Xiong, Dr. Ee Ping Ong, all from Institute of Microelectronics (IME), Singapore.

CONTACT PERSON:

Dr. Kai-Kuang Ma, School of Electrical and Electronic Engineering, Nanyang Technological University, Block S2, Nanyang Avenue, Singapore 639798. Tel: +65-790-6366; Fax: +65-792-0415; Emails: ekkma@ntu.edu.sg and kaikuang@hotmail.com.

1.3 Technical Description of PMVFAST

1.3.1 Introduction

This section provides the technical description of the *Predictive Motion Vector Field Adaptive Search Technique* (PMVFAST[4]) which adds some techniques from the *Advance Predictive Diamond Zonal Search* (APDZS) [5] proposed by the Hong Kong University of Science and Technology (HKUST) to the MVFAST core mentioned above to achieve larger speed up. The PMVFAST was contributed by Prof. Ming L. Liou, Dr. Oscar C. Au, and Alexis Tourapis of HKUST. PMVFAST is faster than MVFAST at the expense of higher hardware complexity. The earliest documentation related to APDZS and MVFAST can be found in [6] and [1-3] respectively.

Several independent parties, Optivision Inc., Sarnoff Co., Mitsubishi Electric Information Technology Center America, National Technical University of Athens (NTUA), and Beijing University of Aeronautics and Astronautics (BUAA), conducted evaluation throughout the entire adoption process. For independent verification efforts, the following individuals are greatly acknowledged: Dr. Weiping Li (from Optivision), Dr. Hung-Ju Lee and Dr. Tihao Chiang (from Sarnoff), Mr. Anthony Vetro and Dr. Huifan Sun (from Mitsubishi), Mr. Gabriel Tsechpenakis, Mr. Yannis Avtithis and Prof. Stefanos Kollias(from NTUA), and Prof. Bo Li, Yaming Tu (from BUAA).

1.3.2 Technical Description of PMVFAST

PMVFAST combines the 'stop when good enough' spirit, the thresholding stopping criteria and the spatial and temporal motion vector prediction of APDZS and the efficient large and small diamond search patterns of MVFAST. Let the *refBlock* be the block in the reference frame at the same spatial location as the current block. Without loss of generality, the distortion criterion is assumed to be the Sum-of-Absolute-Difference (SAD), though it can be other measures. The predicted motion vector in PMVFAST is the median of the motion vectors of three blocks spatially adjacent to the current block (left, top and top right), as in MPEG motion vector predictive coding.

Firstly, the PMVFAST computes the SAD of the predicted motion vector (PMV), and stops if any one of two stopping criteria is satisfied. The first criterion is that the PMV is equal to the motion vector of refBlock and the SAD of PMV is less than that of refBlock. The second criterion is that the SAD of PMV is less than a threshold.

Secondly, the PMVFAST computes the SAD of some highly-probable motion vectors (MV of left, top and top right spatially neighboring blocks, MV of (0,0) and MV of refBlock) and stops if any one of two stopping criteria is satisfied. The first

criterion is that the best motion vector so far is equal to the MV of refBlock and the minimum SAD so far (MinSAD) is less than that of refBlock. The second criterion is that the MinSAD is less than a threshold.

Thirdly, the PMVFAST selects the MV associated with minSAD and performs a local search using techniques of MVFAST. If PMV is equal to (0,0) and the motion vectors of the three spatially adjacent blocks are identical with large associated SAD, the large diamond search of MVFAST is applied. Otherwise, if the motion vectors of the three spatially adjacent blocks are identical and are the same as the MV of refBlock, small diamond search is applied with the simplication that only one small diamond pattern is examined. Otherwise, the small diamond search of MVFAST is applied.

Here is the step-by-step algorithm of PMVFAST: The variables *thresa*, *thresb* are integers used as thresholds in the stopping criteria.

(Initialization)

Step 1: Set thresholding parameters (*thresa* & *thresb*). These are set as follows:
If first row and column, *thresa* = 512, *thresb* = 1024
Else *thresa* = minimum value of the sad of left, top and top-right blocks. *thresb* = *thresa* + 256;
If *thresa*<*512, thresa = 512.* If *thresa* > 1024, *thresa* = 1024.
If *thresb* > 1792, *thresb* = 1792.
Set *Found*=0 and *PredEq*=0
Compute the predicted MV according to the Median rule.
Select previous MV, above, and above-right and calculate median.
If block is an edge block, depending to the position, do the following:
If block is on the first column, assume previous MV to be equal to (0,0).
If block is on the first row, select previous MV as the prediction.
If block is on the last column, assume above right MV to be equal to (0,0).
If left MV = top MV = top-right MV then set *PredEq*=1;

(Initial prediction calculation)

Step 2: Calculate *Distance*= $|\text{MedianMV}_X| + |\text{MedianMV}_Y|$ where MedianMV is the motion vector of the median. If *PredEq*=1 and $\text{MV}_{\text{predicted}}$ = Previous Frame MV, set Found=2

Step 3: If *Distance*>0 or *thresb*<1536 or *PredEq*=1
Select small Diamond Search. Otherwise select large Diamond Search.

Step 4: Calculate SAD around the Median prediction. MinSAD=SAD
If Motion Vector equal to Previous frame motion vector and MinSAD<PrevFrmSAD goto Step 10.
If SAD<=256 goto Step 10.

Step 5: Calculate SAD for motion vectors taken from left block, top, top-right, and Previous frame block. Also calculate (0,0) but do not subtract offset.
Let MinSAD be the smallest SAD up to this point. If MV is (0,0) subtract offset.

Step 6: If MinSAD <= *thresa* goto Step 10.
If Motion Vector equal to Previous frame motion vector and MinSAD<PrevFrmSAD goto Step 10.

(Diamond Search)

Step 7: Perform Diamond search, with either the small or large diamond. If *Found*=2 only examine one Diamond pattern, and afterwards goto step 10

Step 8: If small diamond, iterate small diamond search pattern until motion vector lies in the center of the diamond. If center then goto step 10.

Step 9: If large diamond, iterate large diamond search pattern until motion vector lies in the center. Refine by using small diamond and goto step 10.

(Final step. Use best MV found.)

Step 10: The motion vector is chosen according to the block corresponding to MinSAD.
By performing an optional local half-pixel search, we can refine this result even further.

1.3.3 Special Acknowledgement

M.L. Liou, O.C. Au, A.M. Tourapis would like to sincerely acknowledge the tremendous support from Prof. Ishfaq Ahmad, the Department of Electrical and Electronic Engineering of the Hong Kong University of Science and Technology (HKUST), the Hong Kong Telecom Institute of Information Technology (HKTIIT), the Hong Kong Reseach Grants Council (RGC).

CONTACT PERSON: Dr. Oscar C. Au, Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China. Tel: +852 2358-7053; Fax: +852 2358-1485; Emails: eeau@ust.hk.

1.4 Conclusions

The comparison of MVFAST vis-à-vis PMVFAST is given in Table 2.

Table 2: Comparison of MVFAST and its derivative algorithm PMVFAST

	MVFAST	PMVFAST
Threshold comparisons	No threshold or one optional threshold	A set of compulsory thresholds. Coding performance and sensitivity of PMVFAST using these thresholds for the video sequences and encoding conditions outside the MPEG-4 test set has *not* been studied and verified.
Algorithmic complexity	Less complex	Higher complexity
Memory size	No need to store either search points or motion vectors	Memory is compulsory. Up to 4 Mbytes of memory for a search window size of +- 1024 to store search points and up to 1.3 kilobytes for storing motion vectors
Memory bandwidth	Not applicable (since no memory is needed)	Memory bandwidth is wasted for accessing the memory when each search point is visited
Objective quality (PSNR)	On average, about 0.2 dB less than Full Search	On average, about 0.1 dB less than Full Search
Speed-up		About 50% faster than MVFAST
Scalability	Scalable to three different profiles; where each profile is obtained by simply assigning values to two parameters in the beginning of the algorithm.	Not scalable

The video group recommends MVFAST as the core technology for fast motion estimation, since it is a generic solution suitable for all encoding environments. However, if issues such as memory, algorithmic complexity and threshold sensitivity are not of concern, then PMVFAST algorithm can be used. Therefore, MVFAST is integrated as the core mode in Part 7.

1.5 References

[1] Prabhudev Irappa Hosur and Kai-Kuang Ma, "Motion Vector Field Adaptive Fast Motion Estimation," *Second International Conference on Information, Communications and Signal Processing* (ICICS '99), Singapore , 7-10 December 1999.

[2] Prabhudev Irappa Hosur and Kai-Kuang Ma, "Report on Performance of Fast Motion Estimation using Motion Vector Field Adaptive Search Technique (MVFAST)," ISO/IEC JTC1/SC29/WG11 M5453, Maui (USA), December 1999**.**

[3] Kai-Kuang Ma and Prabhudev Irappa Hosur, "Performance Report of Motion Vector Field Adaptive Search Technique (MVFAST)," ISO/IEC JTC1/SC29/WG11 M5851, Noordwijkerhout (Netherlands), March 2000**.**

[4] Alexis M. Tourapis, Oscar C. Au and Ming L. Liou, "Fast Block-Matching Motion Estimation using Predictive Motion Vector Field Adaptive Search Technique (PMVFAST)," ISO/IEC JTC1/SC29/WG11 M5866, Noordwijkerhout (Netherlands), March 2000**.**

[5] A. M. Tourapis, O. C. Au, and M. L. Liou, "Fast Block-Matching Motion Estimation using Advanced Predictive Diamond Zonal Search (APDZS)", ISO/IEC JTC1/SC29/WG11 MPEG2000/m5865, Mar 2000, Noordwijkerhout, Netherland.

[6] A. M. Tourapis, O. C. Au, and M. L. Liou , 'Fast Motion Estimation using Circular Zonal Search', ISO/IEC JTC1/SC29/WG11 MPEG2000/m4038, Oct 1998, Atlantic City, USA.

2. Fast Global Motion Estimation

2.1 Introduction to Feature-based Fast and Robust Global Motion Estimation Technique

Sprite coding is an important technology in MPEG-4 encoder, but sprite coding could be hardly applied in real-time application because the global motion estimation (GME) used in sprite coding is a time-consuming task. In order to overcome this problem a feature-based fast and robust GME technique (FFRGMET) [3][4][5] is proposed by Tsinghua University, which improves the original GME method [2]. Comparison experimental results [6] show that FFRGMET improves the speed substantially. There are three significant improvements of FFRGMET compared with original GME method [2].

(1) More Accurate Outlier Detection on the Base Level

Three-level pyramid is applied in the GME calculation. Local motion will affect GME when there is local motion. Pixels undergoing local motion are the outliers in GME. Residual block based outlier detection method is used in the base level of GME in FFRGMET. The original outlier detection method in VM is residual histogram based, which could not represent the spatial distribution of the residuals. At the base level of the estimation, the pixels belonging to the foreground object appear to show large residuals and concentrate together to a compact region, so the residual block based outlier detection could help to locate the outliers more accurately.

(2) More Robust Object Function for Parameter Estimation

The robust object function is used to replace the quadratic object function in VM. This object function is adaptive to the variance of all pixels to be estimated, and is more robust to outliers than the quadratic form. Robust object function is very important in FFRGMET because there are fewer pixels in calculation compared with GME method in VM. Robust object function restrains the effect of outliers in GME. Robust object function considers the residual and variance of the pixels set in the estimation.

(3) Fewer Pixels Used in GEM Calculation

On the intermediate level and base level of pyramid, it does not need to use all the pixels in the object region to participate the GME calculation. There is much redundant information in whole background. So in FFRGMET, only some feature pixels are selected as representatives in GEM. The feature pixels are selected based on the spatial edge and the temporal difference, which could contribute more to the motion estimation than other pixels. The speed of GME is accelerated because fewer pixels are used.

2.2 Technical Description of FFRGMET

2.2.1 Outlier Exclusion

Residual-block based method to exclude the outliers in FFRGMET calculation. The image is divided into 16×16 sized blocks. The block is regarded as a potential block to be rejected if the SAD of this block belongs to the top 30% ordered by the SAD of a block.

$$SAD = \sum_{i=0}^{16}\sum_{j=0}^{16}\left|\gamma_{ij}\right|;\ \ \gamma_{ij} \text{ is the residual of pixel (i,j).}$$

There are two steps to determine whether to reject the blocks in GME calculation or not. (a) Firstly if there is at least four potential blocks to be rejected in the eight nearest neighbor blocks of current potential block to be rejected, then the current potential block to be rejected will be rejected, otherwise the current potential block to be rejected is reserved. (b) Secondly if there is a rejected block in the eight nearest neighbor of remainder potential block to be rejected, then this potential block to be rejected will be rejected in the calculation of global motion estimation.

2.2.2 Robust Object Function

The following robust object function is used in the Levenberg-Marquadet calculation of FFRGMET:

$$F = \sum_i \frac{r_i^2}{(\sigma^2 + r_i^2)},\ \sigma = 1.253\ \mathrm{E}(r)$$

E(r) is the mean of absolute value of residual r. The weight function for the object function is:

$$\text{weight}(r) = 1/(\sigma^2 + r^2)$$

2.2.3 Feature Selection

Feature pixels are selected in GME calculation according to the following conditions (Condition 7-1).

$$\{(x, y) \mid (UPTHRESH \cdot E(|I_x| + |I_y|)) > (|I_x| + |I_y|) > (DOWNTHRESH \cdot E(|I_x| + |I_y|)) \quad \text{AND} \quad (|I_t| > THRESHOLD_t \cdot E(I_t))\} \tag{7-1}$$

I_x and I_y are the spatial gradient components of luminance. I_t is the temporal gradient of luminance. E(x) is the mean value of the set of x. The pixel that belongs to motion edge must meet the following two conditions. The first is that its spatial gradient value must be larger than a predefined down threshold and less than a predefined up threshold. Differential operator is sensitive to noise when the gradient value is small. So large gradient value can reduce the influence of noise. And large gradient value means that there is an edge of luminance. The second condition is that the absolute value of temporal gradient value must be larger than a predefined threshold. I_t of that pixel will be small if the direction of global motion is perpendicular to the direction of luminance gradient of the point. The pixels belonging to motion edge are used in the intermediate and base levels of pyramid calculation of global motion estimation.

2.2.4 Algorithm Description

Following is the complete description of FFRGMET algorithm (Note that only Y-component data is used in the following steps). A 6-parameter affine model is used in the FFRGMET.

Step1: Set parameters:
Maximum iterative steps for GN (Gauss-Newton) and LM (Levenberg-Marquadet) calculation (MAX_TIMES=32)
Resistant factor of LM (RESISTANCE = 0.002) and amplificatory factor of LM (AMPLIFIER = 10.0)
Number of parameters to be estimated (PARNUM = 6)
Threshold of gradient value (DOWNTHRESH =1.25, UPTHRESH = 1.65)
Threshold of temporal gradient value ($THRESHOLD_t$=1.0)

Step2: Generate the three-level pyramid:
Use Gaussian down-sampling filter [1/4, 1/2, 1/4] on the original image to generate the images of the three-level pyramid.

Step3: Calculate the motion parameters of top level:
Use three-step search method to calculate the initial two translational parameters of 6-parameter affine motion model.
Exclude the top 10% of total pixels in the residual histogram.
Estimate the parameters based on the left pixels using GN method.

Step4: Project the parameters of the top level onto the intermediate level.

Step5: Calculate the motion parameters of the intermediate level:
Exclude the top 10% of total pixels using residual histogram.
Select the pixels according to condition 7-1.
Estimate the parameters of second level with the selected pixels using LM optimizing method.

Step6: Project the parameters of the intermediate level onto the base level.

Step7: Calculate the motion parameters of the base level:
If (VOP.Shape = Rectangle) Then
    Exclude the residual blocks with top 1/3 SAD value using the residual-block based method.
Else
    Exclude the top 10% of total pixels using residual histogram.
End If
Select the pixels according to condition 7-1.
Estimate the parameters of base level with the selected pixels using LM optimizing method.

Note: GN means Gauss-Newton optimizing method, LM means Levenberg-Marquadet optimizing method.

2.2.5 Perspectives on implementing FFRGMET

FFRGMET is faster than the original GME method in MPEG-4 VM, but it is more complex and the PSNR of coding will have a little loss. The original GME method can be used if there is no requirement for speed, otherwise FFRGMET can be used. All those predefined parameters can be changed, which is gotten from the experimentation.

2.2.6 Special Acknowledgements

Shiqiang Yang, Yuzhuo Zhong, Wei Qi and Yuwen He would like to sincerely acknowledge tremendous support from Professor Tihao Chiang and Dr. Huifang Sun, who are charge of Encoder Optimization. For independent verification efforts, the following individuals are greatly acknowledged: Dr. Oscar C. Au, Dr. Alexis M. Tourap, from the Hong Kong University of Science and Technology, China; Dr. Feng Wu, from Microsoft Research China.

CONTACT PERSON: Prof. Shiqiang Yang, Department of Computer Science and Technology, Tsinghua University, Beijing100084, China. Tel: +8610 6278-4141; Fax: +8610 6277-1138; Email: yangshq@mail.tsinghua.edu.cn.

2.3 Conclusions

The feature-based fast and robust global motion estimation technique (FFRGMET) is about 7 times faster than the original global motion estimation (GME) algorithm in MPEG-4 verification model(VM). But the PSNR decreases about 0.06 dB on the average for luminance component. The total GMC coding speed is accelerated about 3.5 times. FFRGMET for sprite coding is much faster than then original global motion estimation method in MPEG-4 VM from the comparison at the cost of a little loss in PSNR, which is negligible. Therefore, FFRGMET is integrated in Part 7.

2.4 References

[1] ISO/IEC WG11 MPEG Video Group, Encoder Optimization Core Experiment Descriptions, ISO/IEC/JTC1/SC29/WG11, No.MPEG00/N3523, Beijing, July 2000.

[2] Janusz Konrad, Frederic Dufaux, Digital Equipment Corp., Improved Global Motion Estimation for N3, ISO/IEC JTC1/SC29/WG11, MPEG97/M3096, San Jose, February 1998.

[3] Qi Wei, Yuzhuo Zhong, Shiqiang Yang, A New Robust Global Motion Estimation Approach, Seoul Meeting of ISO/IEC JTC1/SC29/WG11, MPEG99/M4423, March 1999.

[4] Wei Qi, Yuzhuo Zhong, Shiqiang Yang, Improved Method for Global Motion Estimation, Vancouver Meeting of ISO/IEC JTC1/SC29/WG11, MPEG99/M4687, July 1999.

[5] Yuwen He, Wei Qi, Shiqiang Yang, Yuzhuo Zhong, Feature-based Fast and Robust Global Motion Estimation Technique for Sprite Coding, ISO/IEC JTC1/SC29/WG11, MPEG00/M6226, Beijing, July 2000.

[6] Yuwen He, Wei Qi, Shiqiang Yang, Yuzhuo Zhong, Experiment Results for Feature-based Fast and Robust Global Motion Estimation Technique for Sprite Coding, ISO/IEC JTC1/SC29/WG11, MPEG00/M6548, La Baule, October 2000.

3. Fast and Robust Sprite Generation

3.1 Introduction to Fast and Robust Sprite Generation

Just as those techniques for fast motion estimation and fast global motion estimation, the technique for sprite generation is also very important for sprite coding. This section dedicates to describe the algorithm for fast and robust sprite generation. Firstly, the described algorithm can significantly speed up the sprite generation compared with the method provided in MPEG-4 video VM [1], meanwhile only the little extra memory is necessary. Secondly, the described algorithm can provide better subjective visual quality as well. The visual quality is another key point for static sprite coding because the background object is reconstructed by directly warping the sprite according to the definition of static sprite coding in MPEG-4 standard. Furthermore, when no auxiliary mask information is available, the segmentation technique proposed in [2] is simplified and applied to this algorithm. It can not only accelerate the motion estimation process but also improve the visual quality of generated sprite.

3.2 Algorithm Description

3.2.1 Outline of Algorithm

The described algorithm bases on that in Appendix D of MPEG-4 Video VM. However, in order to achieve fast and robust sprite generation, some novel features are introduced as shown in Figure 3-1. Instead of estimating the global motion of the current image directly from the previous sprite, the described algorithm first warps the previous sprite and then calculates the global motion referencing the warped sprite image. This long-term motion estimation method can greatly decrease the error accumulations caused by the individual frame. The extra cost in memory is also reasonable because the size of warped sprite is the same as that of current frame. Static sprite coding is normally used for object-based video coding, however sometimes auxiliary segmentation information is either unavailable or not accurate enough to mask out all moving objects from the scene. The rough segmentation technique developed in [2] is incorporated into the proposed sprite generation, which is usually used in this algorithm when no auxiliary segmentation masks are available.

The main goal of the described algorithm is to rapidly generate the background sprite with better visual quality. Assume that the video sequence comprises n frames, $\mathbf{I}_k$, k = 0, 1, …, n-1. The sprite $\mathbf{S}_k$ is generated using $\mathbf{I}_i$, i = 0, 1, …, k-1. $\mathbf{P}_k$ denotes the motion parameter estimated at the kth time instant. The complete sprite generation algorithm is described using pseudo C as following:

1. $\mathbf{S}_0 = \mathbf{I}_0$; $\mathbf{P}_0 = 0$;
2. For (k = 1; k < n; k++) {

   Divide $\mathbf{I}_k$ into reliable (**R**), unreliable (**UR**), and undefined (**U**) regions;

   Fast and robust estimate global motion parameter $\mathbf{P}_k$ between $\mathbf{I}_k$ and $\mathbf{S}_{k-1}$;

   If no auxiliary segmentation is available, then segment $\mathbf{I}_k$;

   Warp image $\mathbf{I}_k$ towards sprite using $\mathbf{P}_k$;

   Blending the warped image with $\mathbf{S}_{k-1}$ to obtain $\mathbf{S}_k$.

   }

There are five modules used for processing each frame in the sprite generation, including Image Region Division, Fast and Robust Motion Estimation, Image Segmentation, Image Warping, and Image Blending. Bilinear interpolation is used for image warping, which is the same as that in MPEG-4 Video VM. Each module of the described algorithm is discussed in detail.

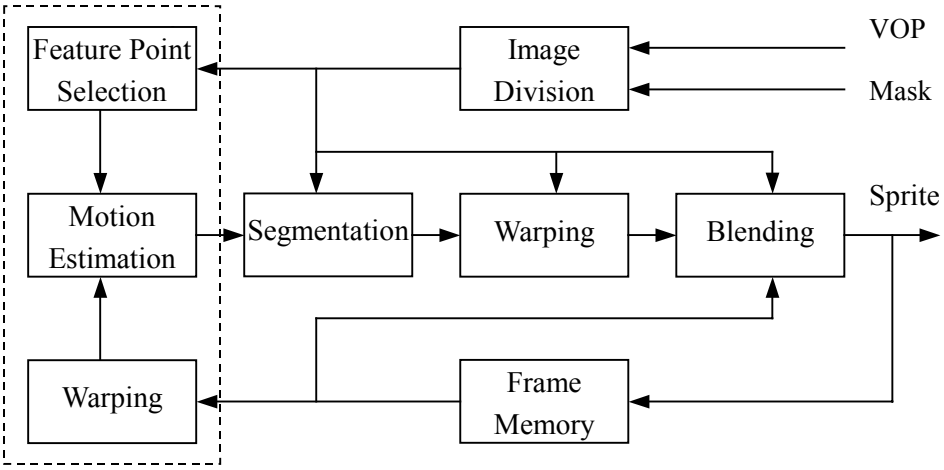
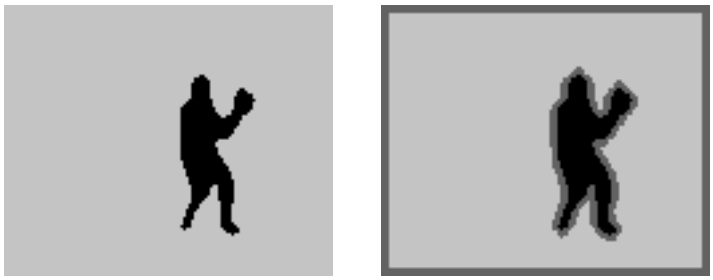


Figure 3-1: Block diagram of fast and robust sprite generation

3.2.2 Image Region Division

According to the visual part of MPEG-4 standard, static sprite coding is normally used for object-based video coding. The described algorithm first derives the reliable mask from the segmentation mask by excluding some pixels along the borders of background object, as well as the frame borders. The excluded areas are defined as unreliable image region (**UR**); the rest region in background object is defined as the reliable image region (**R**). Furthermore, the areas masked as foreground objects are defined as undefined image region (**U**). The core technique of image division is to mask **UR** image region, which can be implemented by scanning the background region from four directions, i.e., left to right, top to bottom, right to left, and bottom to end. After each scanning, start points of the background region can be singled out, and then the subsequent n pixels are masked as **UR** image region. The sprite image is correspondingly divided into **R**, **UR**, and **U** regions as well. Reliable sprite region has been constructed from **R** image region. Unreliable sprite region was the visible part of **UR** image region. And undefined sprite region was not yet visible part of background object in previous images. An example of image division is

shown in Figure 3-1. The image division can contribute to sprite generation on two aspects. Firstly, only **R** region participates the motion estimation, which can speed up the motion estimation processing and eliminate the effect of foreground objects and frame borders. Secondly, **R**, **UR**, and **U** regions are differently dealt with in image blending, which improves the visual quality of generated sprite.



(a) Original segmentation (b) Image division

Figure 3-1: Illustration of image division. Light gray: reliable region (R). Dark gray: unreliable region (UR). Black: undefined region (U).

3.2.3 Fast and Robust Motion Estimation

For the purpose of sprite generation, motion estimation aims at obtaining the motion parameters between the current image and sprite image. The sprite normally severs as the reference image in the motion estimation. In the described algorithm, instead of directly estimating the motion parameters between the current image and sprite, the pre-processing is used to accelerate motion estimation and eliminate the effect of foreground objects. Figure 3-1 illustrates the processing of motion estimation.

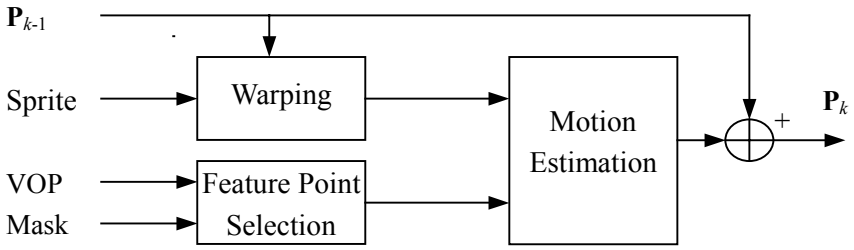


Figure 3-1: Illustration of motion estimation

The pre-processing mainly focuses on the following two aspects. Firstly, feature point selection is performed in order to decrease the pixels involving in motion estimation. The selection is based on the fact that pixels with large gradient values dominantly contribute to motion estimation in the background object, rather than those located in smooth areas. The Sobel operator is first done on the current raw image, and then top 10% of total pixels located in **R** image regions are applied to motion estimation where the image magnitudes have the largest values in the Sobel image. Secondly, the sprite is warped using the previous estimated motion parameters. The great displacement between the current image and the sprite may decrease the accuracy of motion estimation, or even leads to wrong estimation. Using the warped sprite as reference, the current image can increase the robustness of global motion estimation, and even speed up motion estimation. Therefore the proposed fast and robust motion estimation consists of four steps:

Step1: Extract feature points **F** from **R** region of the current image $\mathbf{I}_k$. Sobel operator is performed, and then 10% pixels of the **R** image region are selected.

Step2: Warp previous sprite $\mathbf{S}_{k-1}$ using motion parameter $\mathbf{P}_{k-1}$ to get reference image **S'**.

Step3: Estimate motion parameter **P'** based on gradient decent method using **F** and **S'**.

Step4: $\mathbf{P}_k = \mathbf{P}_{k-1} + \mathbf{P}'$.

$\mathbf{P}_k$ is the final result of global motion estimation. The gradient decent method in step 3 is the same as that described in MPEG-4 Video VM [1].

3.2.4 Image Segmentation

Segmentation information is an essential part for sprite generation and compression. However, the auxiliary segmentation masks are sometimes unavailable. The rough segmentation for sprite generation has been proposed in [2]. In the sprite generation, the simplified segmentation method by using a simple *open* operator with the window size of 9x9 is instead of the time-exhausting iterative morphological operations. Figure 3-1 illustrates the processing of image segmentation. The proposed algorithm can be described in detail using the following steps:

Step1: Warp previous sprite $\mathbf{S}_{k-1}$ using motion parameter $\mathbf{P}_k$ to get predicted image **S'**.

Step2: Subtract current image $\mathbf{I}_k$ from **S'** to get the absolute difference image **D**.

Step3: Filter **D** using *open* operator to detect the motion zones.

Step4: Extract the final segmentation masks by binarizing the filtered image. A threshold is dynamically selected to retain

80% points in background object.

The detailed morphological operation can be found in [3]. Although the segmentation is not accurate with some pixels in the background masked, it is enough only for the purpose of eliminating the effect of foreground objects in the proposed sprite generation. After image segmentation, the background areas are marked as reliable (**R**) image region, and the foreground areas are marked as unreliable (**UR**) image region. The image division results will contribute the following image blending process.

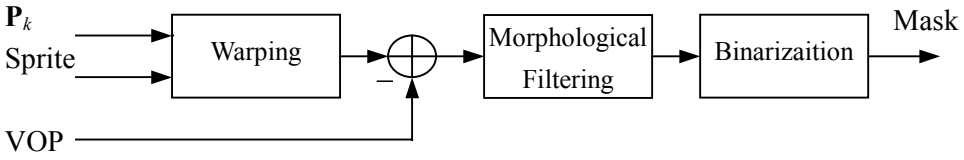


Figure 3-1: Illustration of image segmentation

3.2.5 Image Blending

In the described algorithm, **R** and **UR** regions are differently dealt with in the sprite updating. The pixels located in **R** region are weighted to update the corresponding sprite pixels. However, for those pixels located in **UR** region, the corresponding sprite pixels are updated only when they have never been updated by **R** region pixels. **U** region doesn't participate in this process. The proposed algorithm can provide the sprite with better visual quality. Firstly, the **R** region ensures that reliable image information contributes more to sprite updating because **R** region normally corresponds to the background object. Secondly, the **UR** region tackles the aperture problem when no reliable image information is available. The good trade-off between **R** and **UR** region division can provide better visual quality of sprite than simply averaging images. Suppose ***p*** and ***s*** are a pair of pixels located in warped image and sprite, respectively. The weighting variable relative to ***p*** and ***s*** is denoted by w. The processing of ***p*** and ***s*** can be described using pseudo C as follows,

```
If (p is reliable) {
        If (s is not reliable) {
                Set s as reliable;
                w = 0;
        }
        s = (w * s + p) / (w + 1);
        w = w + 1;
}
Else if (p is unreliable) {
        If (s is not reliable) {
                s = (w * s + p) / (w + 1);
                w = w + 1;
                If (s is undefined)
                        Set s as unreliable;

        }
}
```

3.3 Conclusions

The described algorithm for fast and robust sprite generation in this section can speed up the sprite generation up to 7 times compared with the method in MPEG-4 Video VM with slight improvement in coding efficiency. Moreover, the described algorithm can significantly improve the visual quality of sprite image and reconstructed background object.

CONTACT PERSON:
Prof. Wen Gao, Institute of Computing Technology, Chinese Academy of Science, Beijing 100080, China. Tel: +8610 82649208; Fax: +8610 82649298;
Prof Hanqing Lu, Institute of Computing Technology, Chinese Academy of Science, Beijing 100080, China, Email:hqlu@ict.ac.cn.
Mr. Yan Lu, Department of Computer Science, Harbin Institute of Technology, Harbin 150001, China, Email: ylu@ieee.org.

3.4 References

[1] ISO/IEC WG11 MPEG Video Group, "MPEG-4 Video Verification Model version 16.0", *ISO/IEC JTC1/SC29/WG11 MPEG00/N3312*, Noordwijkerhout, March, 2000.
[2] Yan Lu, Wen Gao, Feng Wu, Hanqing Lu, Xilin Chen, "A robust offline sprite generation approach", *ISO/IEC JTC1/SC29/WG11 MPEG01/M6778*, Pisa, January, 2001.

[3] H. Heijmans, “Morphological image operators”, *Academic Press*, Boston, 1994.

4. Contact Information

For specific technology, please refer to the contacts at the end of each section. For general information, please contact the editors as follows.

Dr. Tihao Chiang
Department of Electronics Engineering
1001 Ta-Hsueh Road, HsinChu
Taiwan 30050, R.O.C.
Tel: +886-3-5712171 ext. 54135
Fax: +886-3-5724361
Email: tchiang@cc.nctu.edu.tw

Dr. Huifang Sun
Mitsubishi Electric Research Laborotoris
Murray Hill Lab.
571 Central Avenue
Murray Hill, NJ 07974
Hsun@merl.com
Tel: 908-665-1200 ext. 35
Fax: 908-665-2414