Dynamic Threshold and Fast Motion Estimation Algorithm based on PSO for H.264's Scene Change Detection

L.Koteswara Rao[1] , Dr.D.Venkata Rao[2]

[1]Associate Professor, Dept of ECE, KGReddy College of Engg.& Tech. Hyderabad, AP, INDIA
[2]Principal, PPD College of Engg.& Tech. Vijayawada, AP, INDIA

ABSTRACT
Many scene change detection algorithms proposed so far use fixed thresholds for identifying the scene change. These thresholds are obtained empirically or they must be calculated before the detection once the whole sequence is obtained. For videos having high scene complexity and variation, the performance of most of the scene change algorithms decrease considerably. In this paper, we study the correlation between local statistical characteristics, scene duration and scene change. Based on this analysis, we further propose and implement a scene change algorithm for H.264 codec, defining an automated, dynamic threshold model with fast motion estimation algorithm having low complexity which can efficiently trace out scene changes. Experimental results on QCIF videos indicate very good performance with significantly improved accuracy combined with minimum complexity.

General Terms
Video signal processing, Encoder.

Index Terms
Dynamic threshold model (DTM), scene change detection, automated threshold, scene duration, Motion Estimation, H.264.

1. Introduction

Now a days, for number of video applications the scene change detection is of great importance. Scene change detection is a part of video encoder to improve its efficiency. Computational schemes of algorithms proposed previously defines a similarity measure between two consecutive frames. When this measure reveals a big enough change, a scene change is declared. These schemes define a fixed threshold. If the value of the measure exceeds the threshold, a scene change is detected. However, a fixed threshold value cannot perform well for all videos mainly due to the diversity of their characteristics. The key problem is to obtain an optimal value for such fixed threshold. If it is set too high, there is high probability that some cuts remain undetected. On the other hand if it is too low, the detection scheme produces false detections. In real-time videos, we can have both cases simultaneously.

To solve the threshold selection problem, many approaches have been proposed. In order to overcome the detection problem, a double threshold (high – low) was proposed to eliminate missed scene changes and dismiss false ones [1].

Although it improved the efficiency, results are not sufficient, especially in real- world videos with high motion like sport games. In addition to this method, a function-based lowering of the threshold, after a scene change was used to decay from high to lower threshold [2]. This technique was used to avoid false detections close to a real scene change, assuming that scene changes cannot occur immediately after each other. However, in most of these methods an optimal threshold (or two thresholds) is required to be determined for each video in advance. Other methods were proposed to find automatically an optimal static threshold e.g. using histogram differences [3], entropy [4] or the Otsu method [5], but they still have the disadvantage of a static threshold and therefore they are not suitable for real-time applications. A truly dynamic threshold is presented in [6], where the input data is filtered by a median filter and then a threshold is set using the filtered output and standard deviation. However, it is not suitable for real-time applications, as the median filter uses future frames as well. A different approach for variable bit rate video is presented in [7], where the bit-rate used in each frame is the "change" metric. It uses statistical properties of the metric values in a single shot, together with the shot's length, to define a threshold.

In this paper we focus on a dynamic threshold model w i t h fast motion estimation algorithm for real-time scene change detection in different video sequences. The method we use is based on the extraction of the sum of absolute differences between consecutive frames by using proposed motion estimation technique from the H.264 codec. These differences serve as a criterion for the choice of the compression method as well as for the temporal prediction. We use a sliding window to extract local statistical properties (mean value, standard deviation) which we further use to define a continuously updating automated thresholds.

Manuscript received May 5, 2011
Manuscript revised May 20, 2011

This paper is organized as follows. In Section 2, the compression standard H.264 and its features are described. In Section 3, we define the motion estimation technique based on PSO. In section 4, we describe the dynamic threshold model for video streams. Section 5 includes the simulation results, while in Section 6 conclusions and final remarks can be found.

2. H.264 Encoder

H.264 [8] is the newest high compression digital video codec standard proposed by the ITU-T Video Coding Experts Group together with the ISO/IEC Moving Picture Experts Group as the product of a collective partnership effort known as the Joint Video Team. This standard is also known as Advanced Video Coding (AVC). The H.264 encoder has three different types of frames defined: spatially predicted frames (I), from previous frames predicted frames (P), bi-directionally predicted frames (B). Each color input frame contains both chrominance and luminance data. Each frame is tiled in macro blocks which are separately encoded spatially or temporally. Macro blocks in H.264 are further tiled in smaller blocks. Each block can be compared with the respective block in the previous frame by using the following motion estimation technique.

3. BI-Directional Motion Estimation Algorithm based On Particle Swarm Optimization

To impose smoothness constraints on the estimation motion field, the image domain is divided in to non overlapping small regions called Blocks assuming that the motion within each block can be characterized by a simple parametric model [9]. Generally, the most straightforward Block Matching Algorithms called full search (FS) simply compares the given macro block (MB) in the anchor frame with all candidate MBs in the target frame exhaustively within a predefined search region. This is not fit for real-time applications because of its unacceptable computational cost. To speed up the search, various fast algorithms for block matching which reduce the number of search candidates have been developed. Well known examples are three-step search (TSS), Four Step Search (FSS), block- based gradient descent search (BBGDS) and diamond search (DS) have been proposed to reduce computational efforts, based on fixed search pattern and greedy search method.

Over the last few years, promising computational intelligence methods, called evolutionary computing techniques such as genetic algorithm (GA), particle swarm optimization (PSO) have been successfully applied to solve motion estimation problem [10]. Such approaches are suitable for achieving global optimal solution, which traditional fast BMAs are not able to obtain easily. The GA needs to set some key parameters such as population size, probability of mutation, probability of crossover, etc. If these parameters are not prefixed properly, efficiency of GA becomes lower and also it is time consuming process. So here we are adopted PSO procedure in order to do the bidirectional motion estimation [11]. Motivated by the potential improvement attainable by switching from independent search to joint search for the motion vector estimation, and by the practical requirement of avoiding an excessively high search complexity, the proposed method is an iterative technique to jointly optimize the motion vectors by using particle swarm optimization. PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality.

Bidirectional ME [12] forms a major computation bottleneck in video processing applications such as detection of noise in image sequences, interpolation/ prediction of missing data in image sequences and de interlacing of image sequences.

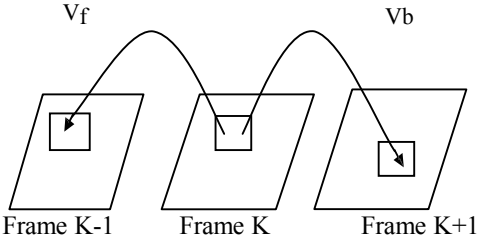


Fig 3.1 .Bidirectional motion estimation

In general, Bidirectional motion estimation is performed by following the steps:

1. Finding forward motion vector Vf by taking past frame as reference frame.
2. Finding backward motion vector Vb by taking future frame as reference.
3. Find the matching error for both methods and find the average motion vector position and its matching error.
4. Compare all the three errors and take the motion vector which is giving the least error.

Generally, the objective of a motion estimation algorithm is to minimize a cost function that measures the interpolation error in the macro block. Examples are the popular sum of absolute difference (SAD).The process of finding robust motion vectors using minimal computations is a heavily researched area, and various fast algorithms have been proposed.

In order to reduce the overall processing time in some video processing applications, the complexity of the bidirectional ME algorithm used [13]. The proposed novel technique to do the bidirectional motion estimation is based on PSO. The Block matching algorithm based on PSO is giving good results in terms of quality and less number of computations. Our idea is not to find the forward and backward motion vectors individually but to find the minimum matching macro block each time when PSO is finding for a minimum matching block. So it will reduce the number of computations involved in finding out the minimum matching point. PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest. The variables pbest and gbest and their increments are both necessary. Conceptually pbest resembles autobiographical memory, as each individual remembers its own experience (though only one fact about it), and the velocity adjustment associated with pbest has been called "simple nostalgia" in that the individual tends to return to the place that most satisfied it in the past. On the other hand, gbest is conceptually similar to publicized knowledge, or a group norm or standard, which individuals seek to attain. The updating formula for each particles velocity and position in conventional standard PSO is written as

$$V_{id}(t+1) = W \times V_{id}(t) + c1 \times rand(.) \times (pbest - X_{id}(t)) + c2 \times rand(.) \times (gbest - X_{id}(t))$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1)$$

where i = 1, 2…. N, N is the number of particles in the swarm, d = 1, 2….D, and D is the dimension of solution space; Vi = (Vi1 , Vi2 , ...Vid), V id ε[−Vmax , Vmax] is the velocity vector of particle i which decides the particle's displacement in each iteration. Similarly, Xi = (Xi1 , Xi2 ,…Xid), Xid ε [−Xmax , Xmax] is the position vector of particle i which is a potential solution in the solution space. the quality of the solution is measured by a fitness function, W is the inertia weight which decreases linearly during a run and c1, c2 are both positive constants, called the acceleration factors which are generally set to 2 and rand(.) and rand(.) are two independent random number distributed uniformly over the range [0, 1]; and Pg , Pi are the best solutions discovered so far by the group and itself respectively. The termination criterion for iterations is determined according to whether the presetting maximum generation or a designated value of the fitness is reached.

Particle's velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.

3.1 Algorithm Steps

The proposed algorithm can be summarized in the following steps

1. Initialization. Assume c1 = 2, c2 = 2, and W be from 0.9 to 0.4 linearly.
2. Perform block matching algorithm based on PSO.
3. Each time find the minimum matching error (SAD) point in the past frame and the current frame as shown in fig.3.1.
4. Take the minimum out of both matching error (SAD), this is considered as the Cost function of our algorithm.
5. For each generation we are getting the minimum error point in the two reference frame at a time.
6. Until it reaches the stopping criteria it will continue the above steps.
7. Save the final motion vector point for motion compensation.

Since we are performing the Block matching procedure at a time in two reference frames, our objective function is to minimize the mean of the two matching errors between two frames.

$$\text{Cost fun} = \min(SADP, SADF)$$

Here SADP and SADF are the sum of absolute difference of the past frame and future frames respectively.

3.2 Experiments and Simulation results

The performance of the proposed Bidirectional motion estimation block matching algorithm based on Particle Swarm Optimization(POS) is evaluated in terms of Average Mean Square Prediction Error (AMSPE) in table 3.1, and Average Search points per frame in table 3.2 is computed for quality measurement.

Sequence	Bi-Ds	Bi-Pso
Zib Sport	125.1636	80.3798
Euro Sport	32.4141	29.5195
Akiyo	11.8697	16.3192
Quiet	76.9223	51.4913

Table 3.1.Average mean square prediction error

Sequence	Bi-Ds	Bi-Pso
Zib Sport	21.6463	10.9515
Euro Sport	21.3988	11.3266
Akiyo	19.7825	9.9448
Quiet	19.905	7.5776

Table 3.2.Average search points per frame

4. Automated Dynamic Threshold Model

A scene is a block of frames between two scene changes. The sequence of SAD values that are computed has statistical properties that are valuable in our effort to detect scene changes. On scene changes we obtain a high SAD value which makes them detectable. However, we can also have high SAD values during rapid movement, sudden light changes and transition effects such as zoom in/out, dissolve etc. Moreover, we can have scene changes with very different value levels.

A scene break, where both scenes have similar background does not give a peak as high as

if they had different ones. Consequently, a thresholding function is needed, which will be able to adapt to the character of the scene without the need for a previous input. If we want to preserve the real-time character of the algorithm, we should only use the previous sequence values, without looking in the future. For the proposed thresholding function, we use local statistical properties of the sequence. Let Xi be a random variable which models the SAD value in frame i. We use a sliding window of length N. If the actual frame number is n, then the window lies between [n- (N+1), N-1].We compute the empirical mean value mn and the standard deviation σn of Xi in the defined window as follows.

$$m_n = \sum_{i=n-N-1}^{N-1} Xi$$

$$\sigma_n = \sqrt{\frac{1}{N-1}\sum_{n=n-N-1}^{N-1}(Xi - mn)^2}$$

We use the above two equations together with Xn-1 to define the threshold T (n) as follows:

$$T(n)=a.X_{n-1}+b.m_n+c.\sigma_n$$

Alternatively we can read the above equation as T(n) =a(Xn-1- mn)+(b+a) mn+c.σn which is more illustrative for the following discussion about appropriate choice of the constants. Constants a, b, c are very important because they determine the character of the thresholding function. If b takes high values, the threshold will become more rigid, keeping values without approaching the Xi sequence. This avoids wrong change detections like in case of intense motion scenes; but on the other hand, the detector can miss some low valued, difficult scene changes. A low value of b allows us to detect regions with high activity, which can provide valuable information to other applications (e.g. bit-rate control, semantics, error concealment etc.). As σn is the standard deviation, high values of c prevent from detecting intense motion as a scene change. On the contrary, a must have a small value because, as we have already discussed, the properties of Xi are not always welcome in scene change detection. The whole procedure of choosing a, b, c is a tradeoff and should be performed with respect to the intended application.

In addition to the dynamic threshold model described, a function based lowering of the threshold found in [2] is employed to avoid false detections immediately after a scene change. When a scene change is detected in frame p, the SAD value of this frame is assigned to the threshold. In this case, for the next K frames we further use the threshold Te(n) which is decaying exponentially, in order to avoid false scene change detection very close to the previous change

$$T_e(n)=X_{n-1} \exp(-s(n-p)),$$

where 's' controls the speed of decaying.

5. Experimental Results

In this section, we validate the proposed scheme for real-time encoding application. For this purpose we use the H.264 codec v9.2 found in [14] modified to extract the SAD values. We used video sequences with football content. They were chosen because they have scenes with intense motion, change of light conditions, high complexity and different types of scene changes. There are changes between the field and the crowd, which are easy to detect but there are also scene changes with the same background (the playground) which are more complicated. In football sequences also many visual effects like zoom in/out and transition effects like dissolve, fade in/out can be found.

All these characteristics make football videos very challenging for a scene change detector. The first video sequence ("zibsport") is a collage of highlights from the Austrian league. It includes the cup ceremony, celebrations, football highlights and even some violent incidents in the watching crowd. Its resolution is 320 x

240, and its length is 4,000 frames. It was encoded with the H.264 codec with one I frame at the start and P and B frames in the format PBPBPBPB. The number of true scene changes in this sequence was 36. We obtained them manually by watching the video and counting them.
The second video sequence ("eurosport") and it is a typical football match recorded in 2004. Its resolution is 172 x 144 (QCIF) and it is encoded in the same way as the previous video. Its length is 20,000 frames. The number of true scene changes in this sequence was 64. The evaluation of the proposed method is performed by comparing with other methods and the ground truth. For this reason we employ the "recall" and "precision" ratios

Recall =Nc/(Nc+ Nm)
Precession= Nc/(Nc+ Nf)

where Nc is the number of correct detections,
Nf the number of false ones and
Nm the number of missed ones.

Zib sport (320X480)	True	False	Missed	Recall	Precession
Fixed threshold	29	10	7	.8	.74
Dynamic threshold without exponential decaying	35	2	1	.97	.94
Dynamic threshold with exponential decaying	35	1	1	.97	.97

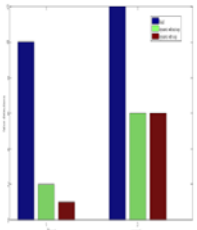
Table 5.1 shows the results of the scene change detection with a fixed threshold, with the dynamic threshold with and without exponential decaying for Zib sport

The fixed threshold used for comparison was chosen optimally (the best case) after having the SAD values for the whole sequence, we took the value that minimizes the number of missed and false detections. Dynamic threshold parameters a, b, c we set to the following values: a=-1, b=2, c=2. These values were chosen the same for both sequences although they have different resolution to test their sensitivity. We used a sliding window of size 20 frames and speed of exponential decaying s=0.02. Please note, that scene change detection is more difficult in small resolutions like QCIF and thus the results are better for the "zib sport" sequence. We could do better if we set the DTM parameters separately for the first and second sequence.However, despite the use of an optimal fixed threshold the DTM performs significantly better in both cases.

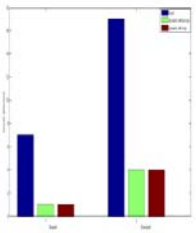
Euro sport(320X480)	True	False	Missed	Recall	Precession
Fixed threshold	43	12	17	.67	.78
Dynamic threshold without exponential decaying	60	6	4	.93	.91
Dynamic threshold with exponential decaying	60	6	4	.93	.91

Table 5.2 shows the results of the scene change detection with a fixed threshold, with the dynamic threshold with and without exponential decaying for Euro sport

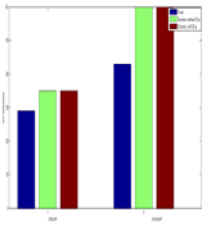
Optimal setting of the DTM parameters may also be slightly different for sequences with another character (for instance a TV discussion, video clip, etc.). Exponential decaying improves the results slightly, but the added value is rather low compared to the contribution of DTM itself against the fixed threshold.



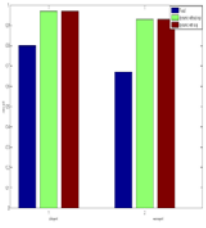
FalseDetections



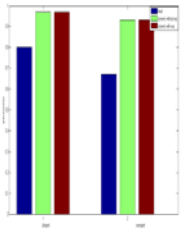
Missed Detections



True Detections



Recall



Precession

6. Conclusions

In this paper we presented a novel automated dynamic threshold model for scene change detection. It is based on the local statistical properties of the video sequence. The method was designed and implemented for H.264 codec, but the idea could be used for any other codec or even for raw video sequences as well. Proposed method has the advantage of low complexity. Moreover, it uses only previous frames for the detection, which makes is suitable for real time applications. The method performs significantly better than an optimum fixed threshold setting and gives very good results also for low resolutions. It can be further enhanced to recognize and handle different kinds of transitions.

REFERENCES

[1] C.L.Huang, B.Y.Liao, A Robust Scene-Change Detection Method for Video Segmentation, IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no.12, pp.1281-1288, Dec.2001.
[2] S.Youm, W.Kim, Dynamic Threshold Method for Scene Change Detection, ICME, vol.2, pp. 337- 340, 2003.
[3] X.Wang, and Z.Weng, Scene Abrupt Change Detection, Canadian Conference on Electrical and Computer Engineering, vol. 2, pp. 880-883, 2000.
[4] K.W.Sze, K.M.Lam, G.Qiu, Scene Cut Detection using the Colored Pattern Appearance Model, ICIP, vol.2, pp. 1017-1020, 2003.
[5] P. K.Sahoo and S.Soltani, A. K.C.Wong and Y.C.Chen, A survey of thresholding techniques, CVGIP, vol. 41, pp.233-260, 1988.
[6] H.C.Liu, G.Zick, Automatic Determination of Scene Changes in MPEG Compressed Video, ISCAS, vol.1 , pp.764-767, 1995.
[7] H.Li, G.Liu, Z.Zhang, Y. Li, Adaptive Scene Detection Algorithm for VBR Video Stream, IEEE Transactions on Multimedia, vol. 6, no. 4, pp. 624-633, Aug. 2004.
[8] H.264 and MPEG-4 video compression by Iain E. G. Richardson.
[9] Video Processing and Communications by Yao Wang, Joern Ostermann,Ya-Quin Zhang.
[10] R.C.Eberhart, Y.Shi, "Comparison between genetic Algorithms and particle swarm optimization," in Proc. IEEE Int. Conf. Evol. Comput., Anchorage, AK, pp. 611616, May 1998.
[11] Y.H.Shi and R.C.Eberhart. Empirical study of particle swarm optimization. In Proc. IEEE Congress on Evolutionary Computation, 1999.
[12] M.-K.Kim, and J.-K. "Efficient motion estimation Algorithm for bidirectional l pre-diction scheme," Electronic Letters, vol.30, no.8, pp.632-633, April 1994.
[13] X.Li, Y.Lu, D.Zhao, W.Gao, S.Ma, "Enhanced direct coding for bipredictive pictures," Proc. IEEE ISCAS, vol.3, pp.785-788, 2004.
[14] H.264/ AVC Software Coordination, JM Software, v9.2, available in http://iphome.hhi.de/suehring/tml/.