AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

Sprite Region Allocation Using Fast Static Sprite Area Detection Algorithm

[1]Zainab Jawad Ahmed and [2]Loay Edwar George

[1]*Department of Biology Science, College of Science, University of Baghdad, Baghdad, Iraq,*
[2]*Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq,*

Address For Correspondence:
Zainab Jawad Ahmed, Department of Biology Science, College of Science, University of Baghdad, Baghdad, Iraq.

ARTICLE INFO

Article history:
Received 18 February 2017
Accepted 5 May 2017
Available online 10 May 2017

Keywords:
Group of Video, MPEG-4, Standard Deviation, Mean, Sprite.

ABSTRACT

Background: Sprite coding is a very effective technique for clarifying the background video object. The sprite generation is an open issue because of the foreground objects which prevent the precision of camera motion estimation and blurs the created sprite. **Objective:** In this paper, a quick and basic static method for sprite area detection in video data is presented. Two statistical methods are applied; the mean and standard deviation of every pixel (over all group of video frame) to determine whether the pixel is a piece of the selected static sprite range or not. A binary map array is built for demonstrating the allocated sprite (as 1) while the non-sprite (as 0) pixels valued. Likewise, holes and gaps filling strategy was utilized to restore the artifacts happened in the binary map. **Results:** The tests results specified that the proposed method is a fast static sprite area detection algorithm that leads quickly to remarkable sprite location. **Conclusion:** It is found that the proposed strategies can allocate the sprite (survive) areas easily and in appropriate way and distinguish static sprite region which demonstrate survived region.

INTRODUCTION

Sprite coding was introduced and standardized in MPEG-4. The fundamental origination of this coding approach is to fragment the video content of the input video into two fragments; foreground and background object **(**Krutz *et al*., 2008). A sprite is a picture comprised of pixels belonging to a video object that noticeable during a video fragment. Some of this background may not be noticeable in certain frames due to camera motion or the obstruction of the foreground objects. Since sprite area comprises of all parts of the background that were at least visible once. The main problem is actually delivering the sprite itself (Raghavendra *et al*., 2001). The background sprite image is utilized for the reconstruction of scene background for an arrangement of progressive frames. The reference sprite is built by combining several views into one large picture that is further encoded and transmitted (Pastrnak *et al*., 2005). Sprites have been included in MPEG-4 mainly because they offer proper high compression efficiency in some cases. The background VOP will be extracted by warping/cropping this sprite suitably (Ebrahimi and Horne, 2000). Before the encoding process the static sprite is produced off line. The decoder receives every static sprite previously the rest of the video segment. The static sprites are encoded in a proper way such that the reconstructed VOPs can be produced simply, by warping the quantized sprite with the suitable parameters (Shi and Sun 2007).

Numerous algorithms have been introduced to handle the problem of sprite portioning.)Jinzenji *et al*., 2001) proposed a two-layer VOP generation plan with some core algorithms such as GME (Global Motion Estimation)**,** foreground moving object extraction and background sprite generation. A shape information diminishment system for the foreground method is clarified. The background sprite is coded utilizing sprite coding in MPEG-4, while the foreground object is object-coded. On VOP generation and video coding with

To Cite This Article: Zainab Jawad Ahmed and Loay Edwar George., Sprite Region Allocation Using Fast Static Sprite Area Detection Algorithm. ***Aust. J. Basic & Appl. Sci.,*** *11(7): 71-77, 2017*

MPEG-4 the experiments are directed. They compared sprite mode with normal mode. The coding productivity of sprite mode is a few times higher than that of normal mode at the same target picture quality when the foreground proportion is between 10-15%. (Farina *et al*., 2004) proposed an algorithm that supplies an ideal fragmentation of a video sequence into independent background sprites (a multi-sprite); it resulting in an extensive reduction of the included coding cost. In spite of the fact that their sprite generation algorithm made different sprites rather than a solitary background sprite, it is completely suitable with the current MPEG-4 standard. The algorithm has been estimated with a few test-sequences. It was distinguished the aggregate coding cost could be reduced by factors of around 2.7 or considerably higher. (Krutz *et al*., 2010) proposed a video coding method that joins the upsides of sprite coding and H.264/AVC (Advance Video Coding). For that, in H.264/AVC coding environment a set of complicated algorithms for global motion estimation, sprite generation and object fragmentation are combined. The proposed approach exceeds H.264/AVC especially in lower bit rate ranges, reserving up to 21% can be implemented.

In this paper, the algorithm is utilized to separate the foreground and the background objects of the input video, we present a basic, simple and quick algorithm to decide static sprite objects (just) in the reference frame of every Group of Video (GOV) in MPEG-4 video. The system decides the sprite blocks using two basic standards: mean and standard deviation.

MATERIALS AND METHODS

The proposed system has two primary modules. They have the vital stages whose assignments are concentrated in order to deciding the sprite blocks. The design of the proposed system is clarified in Figure 1. Firstly, an arrangement of video processing operations is accomplished. Secondly, the sprite region is assigned and flagged. The main stages of the introduced system are:

Fig. 1: The Designing of Proposed System

Module of load video data:

In this module, the video frames belong to GOV are performed utilizing BMP raster format. Each GOV comprises of N frames (in this paper we considered as 10). At that point, the values of the gray component of the pixels belongs to every frame is resolved using the following equation (from MSDN available at: http:// msdn. microsoft.com/ en-us/ library/ ms893078. aspx.):

$$Y(x,y) = ((66\ R(x,y) + 129\ G(x,y) + 25\ B(x,y) + 128) >> 8) + 16 \quad (1)$$

Module of sprite region allocation:

The video frames are partitioned into foreground and background objects. The point of this module is to generate the static sprite with best quality. A post processing process is important to remove the delivered gaps /pores consequently, improves the sprite allocation process result. There are three main stages for sprite region allocation: (i) Static Sprite Blocks Nomination (SSBN), (ii) Gaps Filling and Islands Remove (GFIR) and (iii) Partitioning the Sprite Area into Blocks (PSAB) (Ahmed and George, 2015).

The stage of static sprite blocks nomination:

The necessary steps of the proposed fast static sprite area detection algorithm are:

1. Load all frames belong to GOV in an array Rec(0 to S_{GOV}-1); S_{GOV} is the number of frames that belongs to GOV.
2. Reserve a binary map array Sprite_Map (,), and its size are set equal to frame size.
3. For each pixel, say at (x,y), do:

A. Determine its mean (Mean) and standard deviation (Std) overall GOV frames.

B. If the value of the standard deviation (Std) is less than a predefined threshold (Th_{StdLow}) then mark the pixel as a static sprite pixel (i.e., set as white pixel) in Sprite_Map (,) array.

C. Else if the value of its standard deviation is greater than a predefined threshold ($Th_{StdHigh}$) then set pixel as non-sprite pixel (i.e., black pixel) in Sprite_Map(,) array.

D. Otherwise, do the steps:

(1) Set the counter value (M) to 0.

(2) For all corresponding pixels belong to GOV frames and located at position (x,y): determine the absolute difference between Mean and each value of the corresponding pixels along the frames in GOV, in case the determined difference is less than a predefine threshold Th_{Dif} then increment the counter (M) by 1.

(3) In case the value of (M) is greater a certain percentage of GOV size, then the Sprite_Map(x,y) value is 1 (i.e., sprite), otherwise it is set 0 (non-sprite) .

The stage of gaps filling & islands removal:

In this work, same Gaps Filling and Islands Removal stage to that introduced in (Ahmed and George, 2015) was executed. Because of the thresholding procedure connected for reason of flagging the I-frame's pixels as sprite or non-sprite pixels, numerous islands (little white areas) and gaps (little black areas) will appear in the sprite-map array as (see Figure 2a). The white color indicates the sprite pixel, and black color refers to non-sprite area.

The tilling procedure tests the black pixel and guarantees that this pixel has number of white neighbors over certain threshold number ($Th_{tilling}$); then the pixel is hailed as sprite pixel in situation its neighbors' number condition is satisfied. Same strategy is followed in the erosion procedure; the distinction is trying the white pixel and if the vast majority of its neighbor is non-sprite pixels utilizing certain threshold number ($Th_{erosion}$), then it is flagged as non-sprite (Phillips, 2000). As next step, the seed filling algorithm is performed to fill the arrangements of connected little black areas that may exist inside extensive sprite regions relying upon certain threshold ($Th_{seed\text{-}filling}$) (Khayal *et al*., 2011). Figure 2.b shows the checked sprite region after tilling erosion and seed filling (post-processing step).

a. Sprite region before Gaps and islands **b.** Sprite region after tiling, erosion and seed filling

Fig. 2: An example on constructed frame

The stage of partitioning sprite area into blocks:

In this work, a partitioning sprite area into blocks stage to that examined in paper article (Ahmed and George, 2015) was executed. The sprite blocks are constructed according to basic regular dividing plan. The following sprite blocks construction steps are implemented:

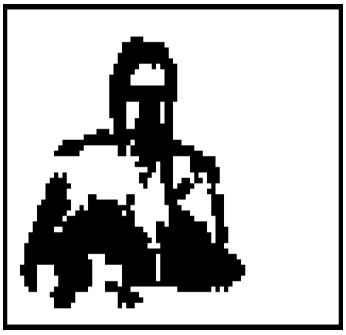
(1) Dividing the constructed binary sprite-map array into non-overlapped blocks; each has size (I×I).

(2) For every block, a pixel-wise scanning is executed for checking the number of white pixels. If the proportion of the counted pixels near block size is progressively a predefined proportion; then the scanned block is considered as a sprite block and all the pixels fall in that block must re-flagged as sprite (whether they are already classified sprite or not); else the block is considered as non-sprite and all its pixels are re-flagged as non-sprite.

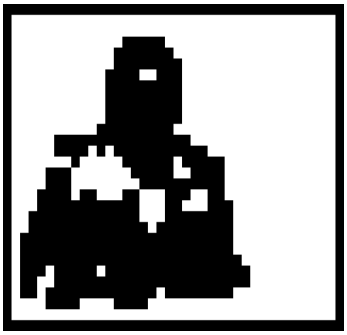
Results and Discussion:

Numerous sets of tests have been performed to evaluate the achievement of the proposed fast static detection algorithm. The system was established utilizing C# programming language. The video test samples Akiyo and Conference, are used (with frame size details: size=320 x240, and size=352 x 288 pixels, respectively and the pixel colour depth =24 bit). These videos were taken from YouTube. Figure 3 demonstrates the frame created by sprite construction steps. Table 1 shows the construction of sprite blocks from static video using the fast static sprite area detection algorithm. The execution was evaluated by using different threshold values with diverse instances of block length [4-8].

Reference frame



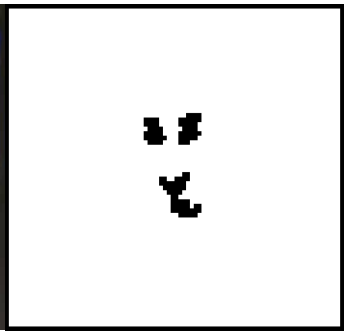
Threshold= 12 and
block size 4×4



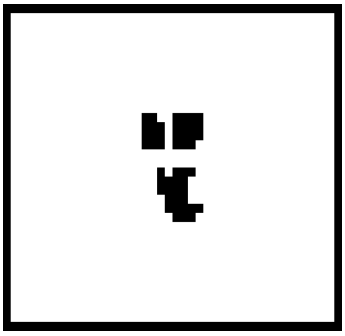
Threshold=64 and
block size 8×8

a. Conference video

Reference frame



Threshold=12 and
block size 4×4



Threshold=64 and
block size 8×8

b. Akiyo video

Fig. 3: Sprite construction

Table 1: the sprite construction using different block size

Conference Video (Static Video)					
Block Size	Threshold	Points Percentage			
		Exact	Missing	Added	Error
4	16	68.833	6.658	0.000	6.658
	15	69.458	6.033	0.042	6.074
	14	70.005	5.486	0.120	5.605
	13	70.902	4.589	0.327	4.915
	12	71.262	4.229	0.447	4.676
5	25	66.276	9.215	0.000	9.215
	24	67.026	8.465	0.031	8.496
	23	67.805	7.686	0.099	7.785
	22	68.349	7.142	0.173	7.315
	21	68.540	6.951	0.210	7.160
	20	69.009	6.482	0.327	6.809
6	36	63.443	12.522	0.000	12.522
	35	63.948	12.018	0.014	12.032
	34	64.349	11.617	0.038	11.655
	33	65.041	10.925	0.101	11.026
	32	65.376	10.590	0.143	10.732
	31	65.701	10.265	0.195	10.460
	30	66.448	9.518	0.345	9.862
7	49	62.288	15.046	0.000	15.046
	48	63.120	14.214	0.017	14.231
	47	63.433	13.900	0.031	13.931
	46	63.986	13.348	0.067	13.415
	45	64.106	13.228	0.077	13.305
	44	64.516	12.817	0.124	12.941
	43	64.803	12.530	0.164	12.694
	42	66.820	10.514	0.500	11.014
8	64	57.667	17.824	0.000	17.824

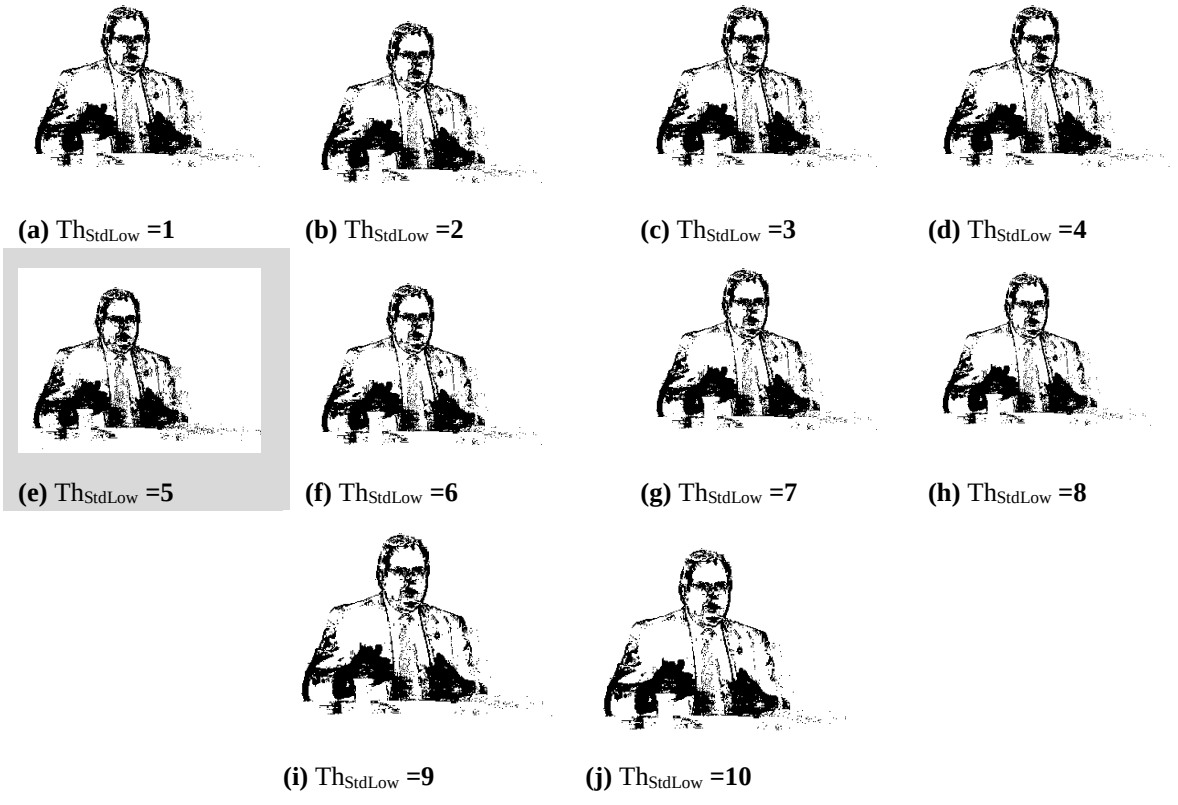
	63	58.323	17.168	0.010	17.178
	62	58.807	16.684	0.026	16.710
	61	59.522	15.969	0.061	16.030
	60	59.600	15.891	0.066	15.957
	59	60.061	15.430	0.105	15.535
	58	60.741	14.750	0.176	14.926
	57	61.112	14.379	0.221	14.600
	56	61.622	13.868	0.294	14.163

There are other control parameters having impacts on the execution of fast static sprite area detection algorithm. The impacts of the following control parameters have been researched: (**i)** the highest permissible standard deviation value Th_{StdLow} for direct indication as sprite point; **(ii)** a threshold Th_{Dif} for deciding whether the determined pixel deviation from mean is small or large; and **(iii)** the lowest number of counted pixels N_{Static} which have small deviation in order to decide the location (x,y) as static sprite pixel. Table 2 presents the adopted default values of the considered control parameters, these qualities are chosen after of making an exhaustive tests and picking the best setup of parameters.

Table 2: The default values of the control parameters in fast static detection algorithm

Parameter	Default Value	Range
Th_{StdLow}	5	[1,10]
Th_{Dif}	4	[1,8]
N_{Static}	7	[1,10]

Figure 4 demonstrates the impact of Th_{StdLow} on the designation rate of sprite point. Figure 5 presents the impact of Th_{Dif} on choosing the decided pixel deflection from mean. Figure 6 illustrates the impact of N_{Static} on the decision of sprite area. The outcomes for the Conference video test allude that the expansion of Th_{StdLow} and Th_{Dif} cause an expansion in sprite area (white range) more than needful; while the outcomes allude that the expansion of N_{Static} causes a reduction in the sprite area more than needful and the corrupted accuracy get to be unsuitable. The estimations of Th_{StdLow} =5, Th_{Dif} =4 and N_{Static} =7 are selected as a best trade- off value because they give a good number of sprite blocks .The effect of $Th_{StdHigh}$ is perceptible on the outcomes; its value was set constant at (40).

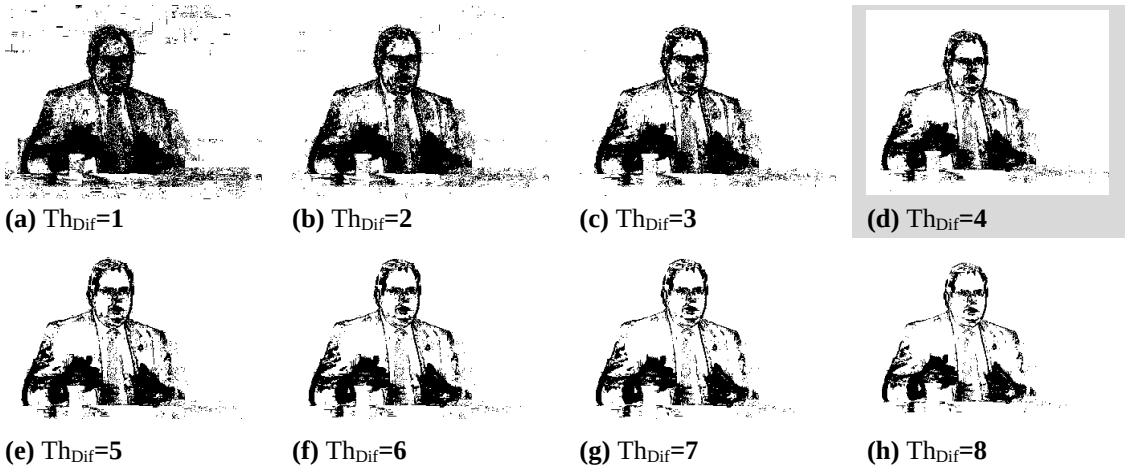


(a) Th_{StdLow} **=1** **(b)** Th_{StdLow} **=2** **(c)** Th_{StdLow} **=3** **(d)** Th_{StdLow} **=4**

(e) Th_{StdLow} **=5** **(f)** Th_{StdLow} **=6** **(g)** Th_{StdLow} **=7** **(h)** Th_{StdLow} **=8**

(i) Th_{StdLow} **=9** **(j)** Th_{StdLow} **=10**

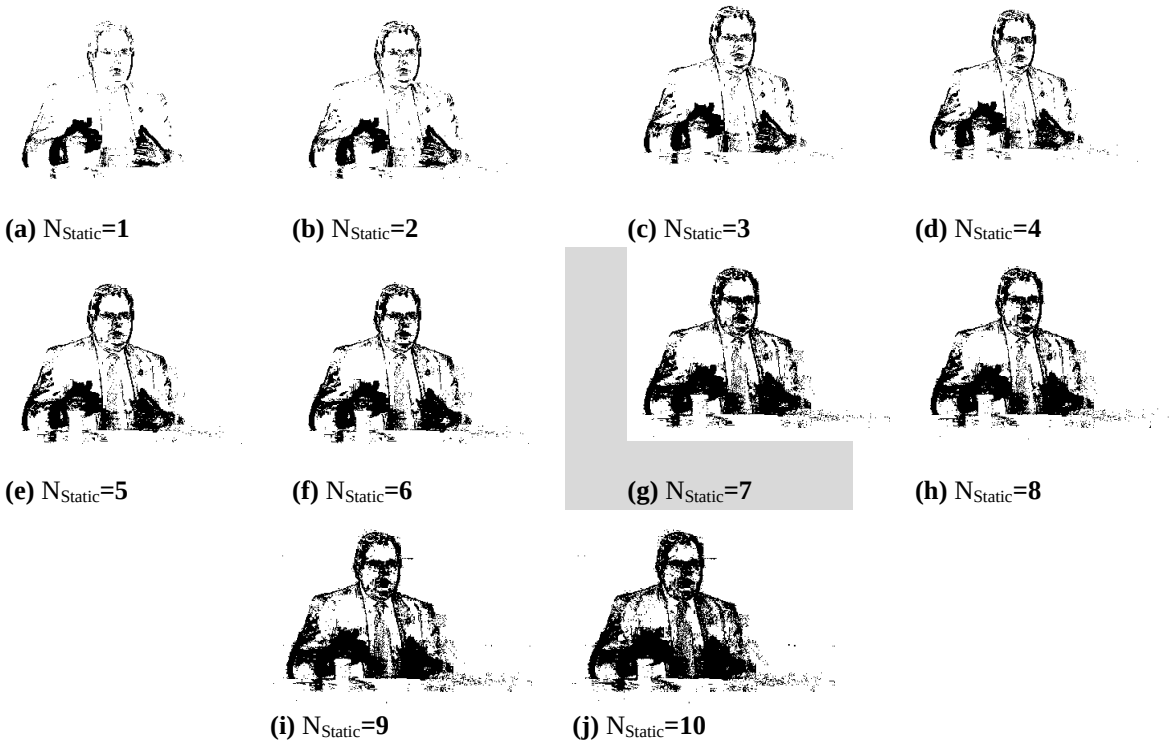
Fig. 4: The effect of threshold Th_{StdLow} in fast static area



(a) Th_{Dif}**=1** **(b)** Th_{Dif}**=2** **(c)** Th_{Dif}**=3** **(d)** Th_{Dif}**=4**

(e) Th_{Dif}**=5** **(f)** Th_{Dif}**=6** **(g)** Th_{Dif}**=7** **(h)** Th_{Dif}**=8**

Fig. 5: The effect of threshold Th_{Dif}



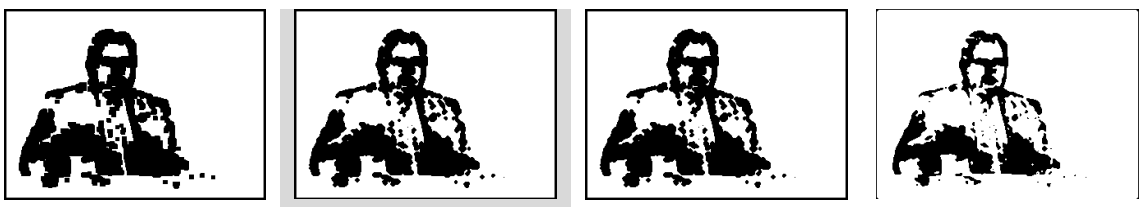
(a) N_{Static}**=1** **(b)** N_{Static}**=2** **(c)** N_{Static}**=3** **(d)** N_{Static}**=4**

(e) N_{Static}**=5** **(f)** N_{Static}**=6** **(g)** N_{Static}**=7** **(h)** N_{Static}**=8**

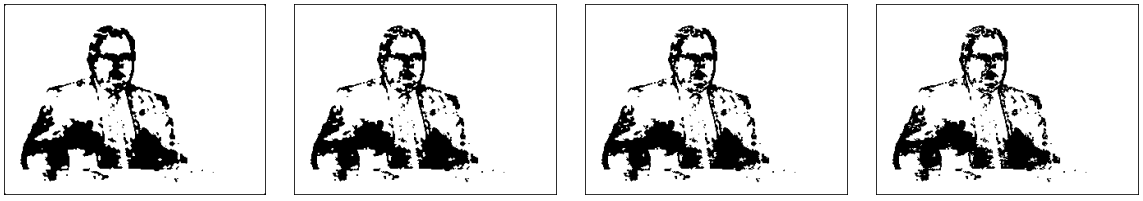
(i) N_{Static}**=9** **(j)** N_{Static}**=10**

Fig. 6: The effect of threshold N_{Static}

For erosion algorithm, the limit $Th_{erosion}$ has huge impact on the islands filling/ expulsion (Phillips, 2000). Figure 7 demonstrates the impact of erosion on isolated island (little white areas) filling. The result demonstrates that the decrease of $Th_{erosion}$ causes increment in filling/expulsion islands. In this way, the most appropriate $Th_{erosion}$ quality is found (2); since it leads to acceptable qualities for isolated island evacuation. The $Th_{filling}$ and $Th_{seed\text{-}filling}$ have same value in motion vector analysis method, so they are set to (5) and (200), respectively (Khayal *et al*., 2011).



(a) $Th_{erosion}$**=1** **(b)** $Th_{erosion}$**=2** **(c)** $Th_{erosion}$**=3** **(d)** $Th_{erosion}$**=4**



(e) $Th_{erosion}$=5 **(f) $Th_{erosion}$=6** **(g) $Th_{erosion}$=7** **(h) $Th_{erosion}$=8**

Fig. 7: The effect of threshold in algorithm

Conclusion:

We have presented a new technique for background sprite generation from the MPEG video. The test outcomes showed that (1) the proposed sprite region system using two statistical methods; the mean and standard deviation is quick; (2) they selected properly static sprite region, (3) the proposed system led to recognized static sprite (survived) detection with better quality and (4) the expended time is enough even for comparatively long GOV. For future work, some appropriate sign decomposers (like, wavelet) can be used to take its advantages to decrease the number of scanning steps to designate the survived region.

Contributions:

This work was contributed in determining static sprite area in video coded with MPEG4 using simple mechanism depending on mean and standard deviation.

REFERENCES

Ahmed, Z.J. and L.E. George, 2015. Sprite Region Allocation Using Motion Compensation Technique. International Journal of Computer Science and Mobile Computing, 4(2): 188-197.

Ebrahimi, T. and C. Horne, 2000. MPEG-4 natural video coding an overview. Signal Processing: Image Communication Elsevier, 15: 365-385.

Farin, D., P.H.N. De and F. Effelsberg, 2004. Minimizing MPEG-4 Sprite Coding-Cost Using Multi-Sprites. SPIE, 5308: 234-245.

Jinzenji, K., H. Watanabe, S. Okada and N. Kobayashi, 2001. MPEG-4 Very Low Bit-rate Video Compression Using Sprite Coding. In IEEE 2001 International Conference on Multimedia and Expo, pp: 5-8.

Khayal, M.S.H., A. Khan, S. Bashir, F.H. Khan and S. Aslam, 2011. Modified New Algorithm for Seed Filling. Journal of Theoretical and Applied Information Technology, 26(1): 28-32.

Krutz, A., A. Glantz and T. Sikora, 2015. Recent Advances in Video Coding Using Static Background Models. In: IEEE 2010 Picture Coding Symposium, pp: 462-465.

Krutz, A., A. Glantz, T. Sikora, P. Nunes and F. Pereira, 2008. Automatic Object Segmentation Algorithms for Sprite Coding using MPEG-4. In IEEE 2008 50th International Symposium ELMAR, pp: 459-462.

Pastrnak, M., D. Farin and P.H.N. De, 2005. Adaptive Decoding Of Mpeg-4 Sprites For Memory-Constrained Embedded Systems. In Proceedings of the 26th Symposium on Information Theory, pp: 137-144.

Phillips. D., 2000. Image Processing in C. R & D Publications.

Shi, Y.Q. and H. Sun, 2007. Image and Video Compression for Multimedia Engineering. CRC Press.