International Scholarly Research Network
ISRN Signal Processing
Volume 2011, Article ID 975462, 11 pages
doi:10.5402/2011/975462

Research Article

Efficient Real-Time Video-in-Video Insertion into a Pre-Encoded Video Stream

Dan Grois,[1] Evgeny Kaminsky,[2] and Ofer Hadar[1]

[1] *Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel*
[2] *Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel*

Correspondence should be addressed to Dan Grois, grois@ee.bgu.ac.il

Received 1 December 2010; Accepted 21 December 2010

Academic Editors: P. G. Bao and W.-L. Hwang

Copyright © 2011 Dan Grois et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This work relates to the developing and implementing of an efficient method and system for the fast real-time Video-in-Video (ViV) insertion, thereby enabling efficiently inserting a video sequence into a predefined location within a pre-encoded video stream. The proposed method and system are based on dividing the video insertion process into two steps. The first step (i.e., the Video-in-Video Constrained Format (ViVCF) encoder) includes the modification of the conventional H.264/AVC video encoder to support the visual content insertion Constrained Format (CF), including generation of isolated regions without using the Frequent Macroblock Ordering (FMO) slicing, and to support the fast real-time insertion of overlays. Although, the first step is computationally intensive, it should to be performed only once even if different overlays have to be modified (e.g., for different users). The second step for performing the ViV insertion (i.e., the ViVCF inserter) is relatively simple (operating mostly in a bit-domain), and is performed separately for each different overlay. The performance of the presented method and system is demonstrated and compared with the H.264/AVC reference software (JM 12); according to our experimental results, there is a significantly low bit-rate overhead, while there is substantially no degradation in the PSNR quality.

1. Introduction

Video-in-Video (ViV) insertion into a pre-encoded video sequence is a very desirable feature for various future applications, including providing various TV services for mobile device users (such as the commercial video insertion, subtitling, and advertising). However, the traditional approaches failed to date in providing an efficient solution for supporting the fast real-time insertion of overlays. The previous works related to the Video-in-Video (ViV) transcoders, such as [1–3] proposed by the Technion Signal and Image Processing Laboratory, use two full decoders to extract the coding domain data (i.e., motion vectors, coding modes, etc.) and to extract raw video sequences from both the compressed original video stream and inserted video content. Upon completing the extraction, the desired video content is inserted into the original video stream, and then the combined video sequence is compressed by an encoder, according to the coding domain data. According to [1–3], the encoder can decrease about 60% in the run time compared to the original JVT encoder (while the picture size of the inserted video content is the 1/4 size of original video stream resolution), which is the saving of about 39% in the run time of the H.264/AVC encoder and decoder CASCADE (based on the Relative CPU (RCPU) performance compared to the conventional H.264 decoder). However, this is still far from being satisfactory, and much more significant run-time reduction has to be achieved.

In this work, we develop and implement an efficient method and system for the fast real-time Video-in-Video (ViV) insertion for H.264/AVC. According to our proposed method, we efficiently insert a video sequence into a predefined location within a pre-encoded video stream for providing various content (e.g., for inserting advertisements into the TV video stream). According to our experimental results, the proposed ViV insertion method enables

achieving a significant performance (in terms of the bit-rate and insertion run time) over the conventional brute-force approaches. In addition, the proposed ViV method enables supporting multiple rectangular overlays of various sizes (e.g., $16N \times 16M$ sizes, where N and M are integers).

According to our ViV method, the video insertion process is performed in two steps.

(a) The first step (i.e., the ViVCF encoder) includes modification of the conventional H.264/AVC video encoder to support the visual content insertion constrained format (CF), including generation of isolated regions without using the FMO slicing, and to support the fast real-time insertion of overlays. This step is computationally intensive, but it should to be performed only once even when different overlays have to be modified (e.g., for different users).

(b) The second step for performing the ViV insertion is relatively simple and substantially not computationally intensive. This step is performed separately for each different overlay.

This work is organized as follows: Section 2 describes the H.264 baseline profile ViVCF, while presenting the IPCM isolation in Section 2.1., inter-isolation in Section 2.2., Luma intra-isolation in Section 2.3., Luma intra-prediction in Section 2.4., generation of the ViV inserter profiles in Section 2.5., and ViV inserter in Section 2.6. In addition, the experimental results and conclusions are presented in Sections 3 and 4, respectively.

2. H.264 Baseline Profile ViVCF

In order to achieve the industry requirements, we focus the development and implementation of our proposed efficient real-time ViV insertion method and system on transferring the majority of all ViV processing to the Encoder 1 (the "Mainstream") and to the Encoder 2 (the "ROAD"/"Region-of-Advertising"), as presented in Figure 1. This simplifies the insertion process to performing direct ViV insertion operations, which in turn enables the ViV insertion process to consume less computational resources.

In the proposed scheme presented in Figure 1, the ROAD 4×4 region (enclosed by a thick line in the "Encoder 1" block) is intra-isolated by Intra-Pulse Code Modulation (IPCM) marcoblocks (MBs). The ROAD IPCM-coded MBs (shown in the grey color within the "ViV Inserter" block) are placed on the top and left ROAD borders for decoding the ROAD MBs independently of the original (ORIG) MBs. In turn, the ORIG IPCM-coded MBs (shown in Figure 2) are placed under the bottom and right ROAD borders for decoding the ORIG MBs independently of the ROAD MBs. The advantage of the proposed IPCM isolation is the relative easy implementation of the ROAD insertion process. By this way, the ROAD inserter is free of any complicated decoder and encoder data operations (such as MC, ME, CAVLC, CAVLD, and CABAC operations). The detailed review of the ViV inserter operations is further presented in Section 2.6.

2.1. IPCM Isolation. The H.264/AVC standard includes the Intra-Pulse Code Modulation (IPCM) macroblock mode [4], in which the values of samples are sent directly, that is, without prediction, transformation, quantization, and the like. An additional motivation for using the IPCM macroblock mode is to allow the regions of the picture to be represented without any fidelity loss. The IPCM isolation is the simplest way to avoid corruption propagation from the ROAD to ORIG MBs, and vice versa. However, the IPCM mode is not efficient by definition (i.e., it requires 384 bytes for each $4:2:0$ 16×16 MB), so we should use it only when the usage of other MB Isolation techniques is not allowed. Thus, we should use the IPCM Isolation to validate the proposed concept of the ROAD insertion (Figure 2 represents the general scheme of the IPCM isolation).

2.2. Inter-Isolation. The main idea of the Inter-Isolation, that is, the usage of inter-modes, is to restrict all MBs outside the ROAD area having motion vectors inside the ROAD area. The motion estimation (ME) method that is currently implemented in the H.264/AVC JM reference software uses three functions: one function for the integer search and two other functions for the subpixel MV (Motion Vector) search (for the 16×16 block partition, and for other partitions separately), while in case when the integer MV points to the ROAD border, then the subpixel ME is disabled for the current MB partition. Figure 3 represents an example [5] of the motion vectors restriction in MB partitions, which originally pointed inside the ROAD area.

As a result, all those vectors were changed to repoint them outside the ROAD area.

2.3. Luma Intra-Isolation. Figure 3 represents available nine 4×4 Luma intra-prediction H.264/AVC modes [5, 6]. The arrows in Figure 4 indicate the direction of prediction in each mode. The encoder may select the prediction mode for each block that minimizes a residual between a predicted block and a block to be encoded.

2.3.1. 4×4 Luma Intra-Isolation. Since the MBs in each Intra-Slice depend one on another, we cannot allow the Mainstream encoder (Encoder 1) to choose particular Intra-Modes. Otherwise, the ROAD area will be affected at least by the left and top-neighbor MBs. For this reason, we restrict the encoder to verify some modes for the ROAD MBs, and MBs which are located below and at the right side of the ROAD area. Figure 5 represents an example of 4×4 Luma intra-isolation process.

According to Figure 5, the following operations are performed.

(a) Applying the VERTICAL, VERTICAL-LEFT, and DIAGONAL DOWN-LEFT, intra-prediction modes for all 4×4 blocks to be adjusted to the RIGHT ROAD border (vertically positioned MBs, as depicted in Figure 5).

(b) Applying the HORIZONTAL and HORIZONTAL-UP intra-prediction modes for all 4×4 blocks to be

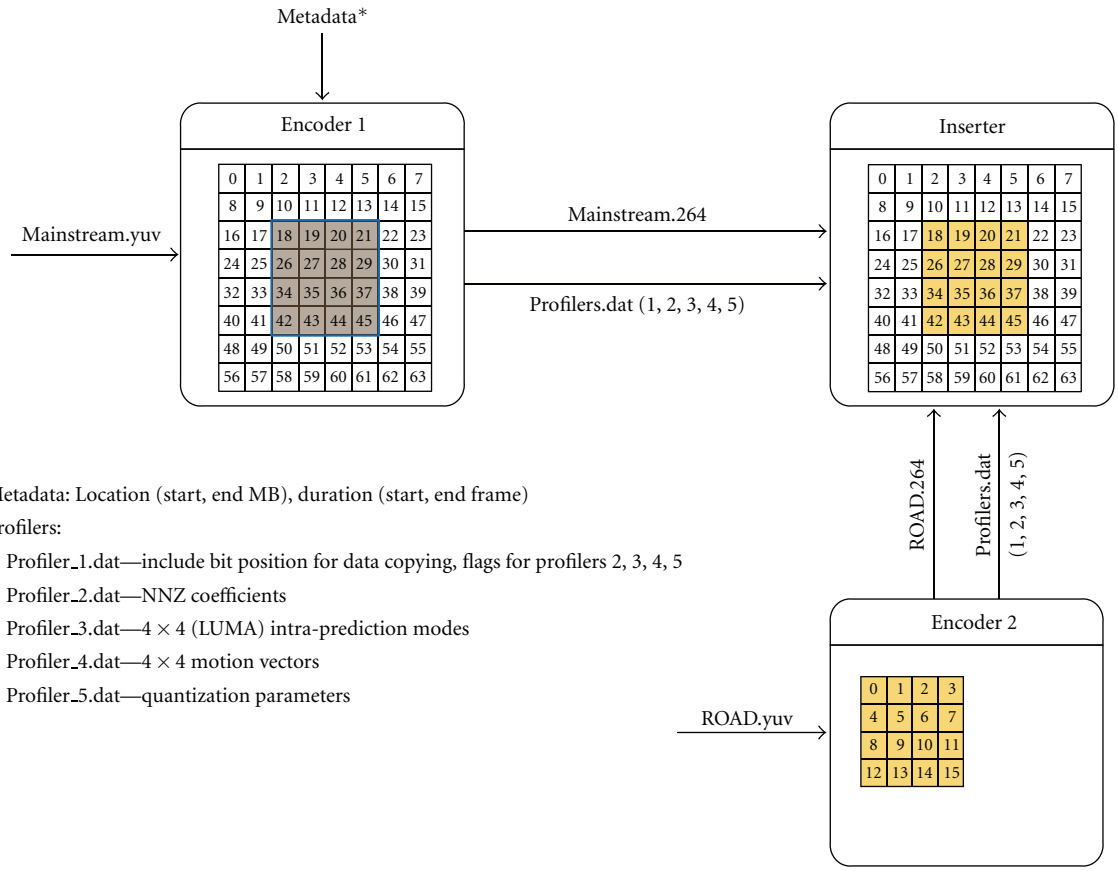


Figure 1: Proposed ViVCF system scheme.

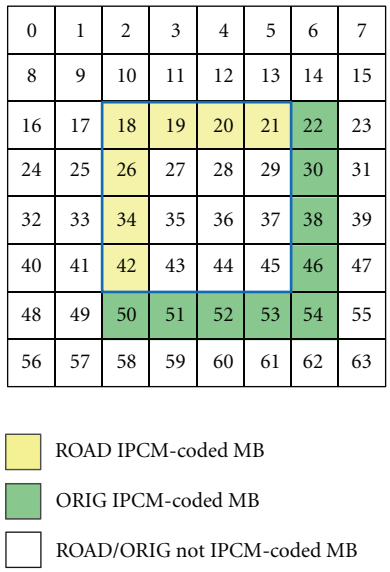


Figure 2: General scheme for the IPCM isolation.

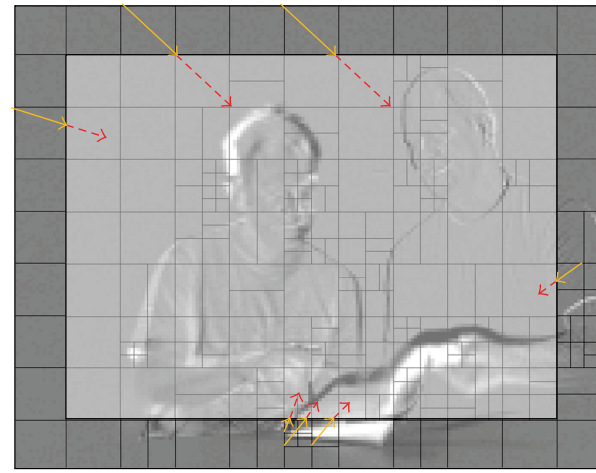


Figure 3: Motion vector restriction of the mainstream.

adjusted to the BOTTOM ROAD border (horizontally positioned MBs, as depicted in Figure 5).

(c) Applying the VERTICAL, VERTICAL-LEFT, DIAGONAL DOWN-LEFT, HORIZONTAL, and HORIZONTAL-UP intra-prediction modes for the 4 × 4 block to be adjusted to the BOTTOM-RIGHT ROAD border (e.g., MB having the "54" value, as depicted in Figure 5).

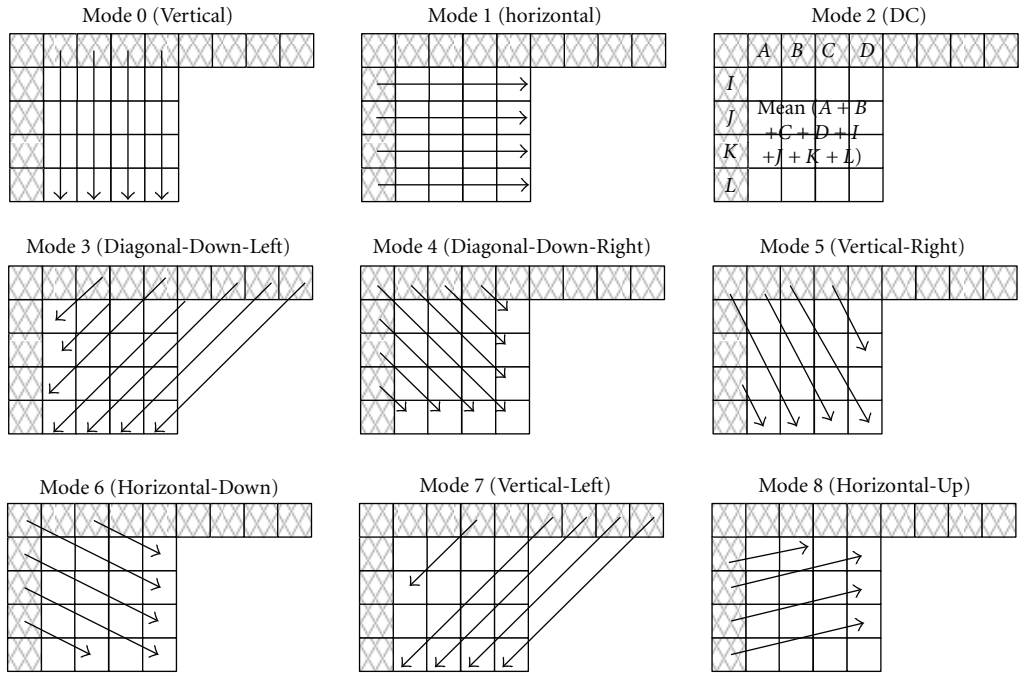


Figure 4: 4 × 4 Luma intra-prediction modes.

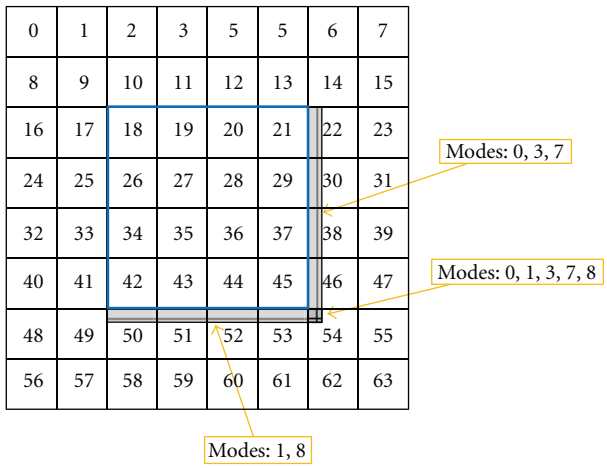


Figure 5: 4 × 4 Luma intra-isolation.

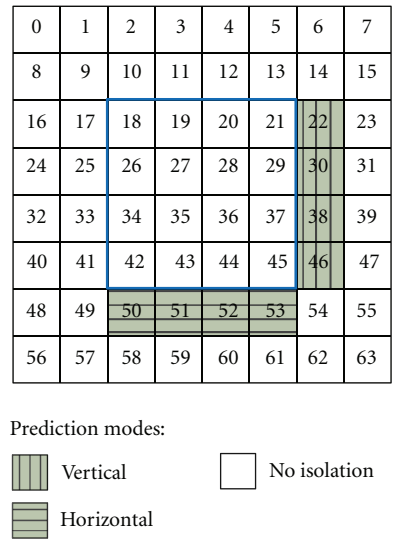


Figure 6: 16 × 16 Luma intra-isolation.

2.3.2. 16 × 16 Luma Intra-Isolation. For the Luma Intra-Isolation, first we disable several 16 × 16 search directions for the MBs, which are positioned near the ROAD area, as shown in Figure 6 which presents an example of 16 × 16 Luma intra-isolation process as follows.

(a) Applying the VERTICAL intra-prediction mode for all MBs to be adjusted to the RIGHT ROAD border (vertically positioned MBs, as depicted in Figure 6).

(b) Applying the HORIZONTAL intra-prediction mode for all MBs to be adjusted to the BOTTOM ROAD border (horizontally positioned MBs, as depicted in Figure 6).

Similarly to the mainstream encoder (Encoder 1), the ROAD encoder (Encoder 2) should also isolate several MBs. For this purpose, we restrict several encoder modes for the ROAD MBs that are adjusted to the left and top image borders, as presented for example in Figure 7.

According to Figure 7, the following operations are performed.

(a) Coding (in the IPCM) the TOP-LEFT MBs for the I-slice, or inter-coding these MBs for the P-slice.

(b) Applying the VERTICAL, VERTICAL-LEFT, and DIAGONAL DOWN-LEFT (i.e., 0, 3, 7 modes) Luma

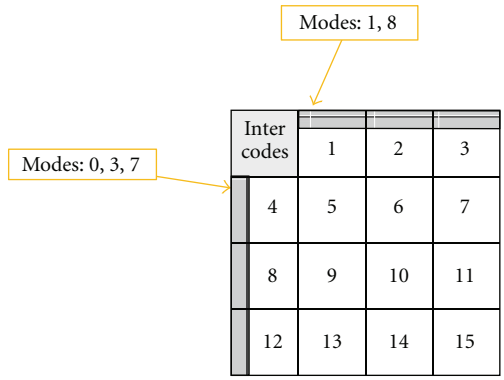


Figure 7: Example of the 4 × 4 Luma intra-isolation.

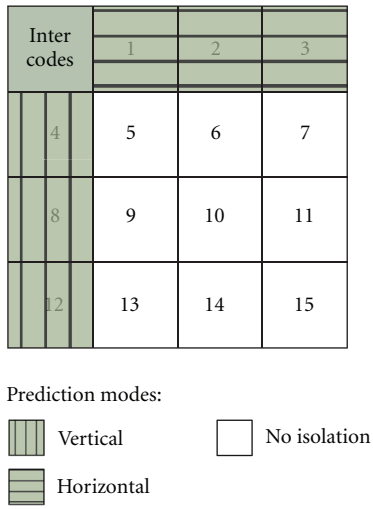


Figure 8: 16 × 16 Luma intra-isolation.

intra-prediction modes for all LEFT ROAD 4 × 4 blocks (vertically positioned MBs, as depicted in Figure 7).

(c) Applying the HORIZONTAL and HORIZONTAL-UP Luma intra-prediction modes (i.e., 1, 8 modes) for all TOP ROAD 4 × 4 blocks (horizontally positioned MBs, as depicted in Figure 7).

Further, we disable several 16 × 16 search directions for the MBs neighbor to the top and left image borders, as shown in Figure 8, which presents an example of the 16 × 16 Luma intra-isolation process as follows.

(a) Coding (in the IPCM) TOP-LEFT MBs for the I-slice, or inter-coding these MBs for the P-slice.

(b) Applying VERTICAL intra-prediction mode for all LEFT ROAD MBs (vertically positioned MBs, as depicted in Figure 8).

(c) Applying HORIZONTAL intra-prediction mode for all TOP ROAD MBs (horizontally positioned MBs, as depicted in Figure 8).

2.4. 16 × 16 Luma Intra-Prediction. As an alternative to the 4 × 4 Luma intra-isolation described in Section 2.3.1. above, the entire 16 × 16 Luma component of a macroblock may be predicted [4, 5]. For this, four modes can be used as shown in Figure 9.

(a) *Mode 0* (vertical): performing an extrapolation from top-positioned samples (denoted in Figure 9 as "H").

(b) *Mode 1* (horizontal): performing an extrapolation from left-positioned samples (denoted in Figure 9 as "V").

(c) *Mode 2* (DC): performing a mean of the top and left-positioned samples ("H + V").

(d) *Mode 3* (Plane): a linear "plane" function is fitted to the top and left-positioned samples ("H" and "V").

2.5. Generation of the ViV Inserter Profiles. According to the proposed method, the profiles are generated (for the "Mainstream" and "ROAD", Figure 1) by the encoder for the enabling to perform the ROAD insertion process.

2.5.1. Mainstream Profiles Generation. For achieving easy and fast operation of the ViV inserter, the mainstream encoder ("Encoder 1", Figure 1) should generate and update at least five different profiles, as illustrated in Figure 10. The first profile (provided as a "profiler_1.dat" file for each compressed frame of the mainstream) includes a set of bit pointers, which determine what portion of a video stream should be copied or skipped. This profile also includes flags to indicate when the remained four profiles (out of five) should be used. The second profile is provided as a "profiler_2.dat" file and includes a number of NNZ (nonzero) DCT coefficients for each 4 × 4 mainstream macroblock that is adjacent to the top-left borders of the ROAD outside area. Also, the third profile is provided as a "profiler_3.dat" file and includes the 4 × 4 Luma intra-prediction modes. In addition, the fourth profile is provided as a "profiler_4.dat" file and is used for the motion vectors, which should be updated according to the predefined motion vectors restrictions. Further, the fifth profile (provided as a "profiler_5.dat" file) is used for the baseline encoder mode and includes the Quantization Parameter (QP) of the left outside borders of the ROAD area.

2.5.2. ROAD Profiles Generation. The ROAD encoder ("Encoder 2", Figure 1) uses the same profiles generations approach as the mainstream encoder ("Encoder 1"). The difference is in the specific macroblocks, which should be updated, as illustrated in Figure 11.

Thus, in *Profile 1*, we specify a bit-counter position of the first ROAD macroblock (MB); this is done for each macroblock line of ROAD frame (i.e., macroblocks no. 0, 4, 8, and 12). Further, in *Profile 5*, we specify the quantization parameter (QP) at the end of each above macroblock line (this should be done because the QP of the ROAD and Mainstream adjacent macroblocks can be different). On the other hand, at the bottom and right borders of the ROAD, we

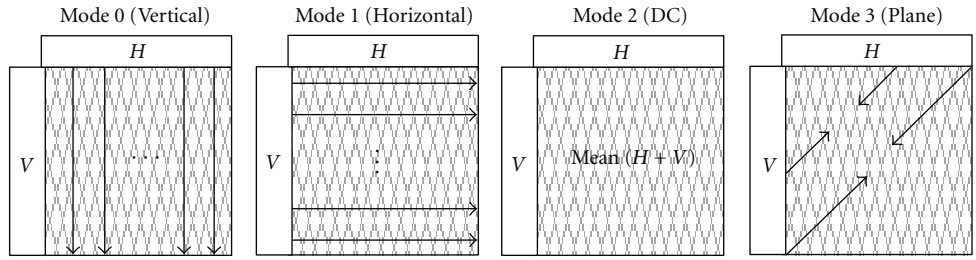


Figure 9: 16 × 16 Luma intra-prediction modes.

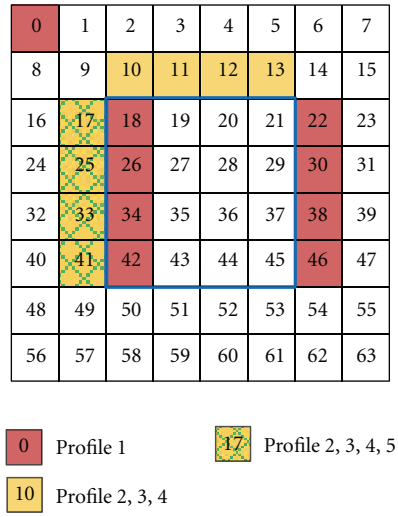


Figure 10: Profiles updating scheme for "mainstream" (Encoder 1, Figure 1).

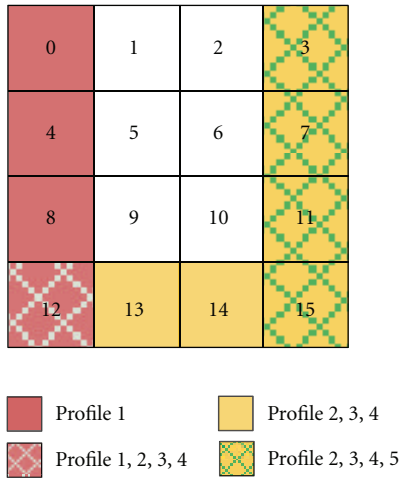


Figure 11: Profiles updating scheme for "ROAD" (Encoder 2, Figure 1).

specify: (a) a number of NNZ (nonzero) DCT coefficients (in *Profile 2*), (b) 4 × 4 Luma intra-prediction modes (in *Profile 3*), and (c) motion vectors (MVs) (in *Profile 4*).

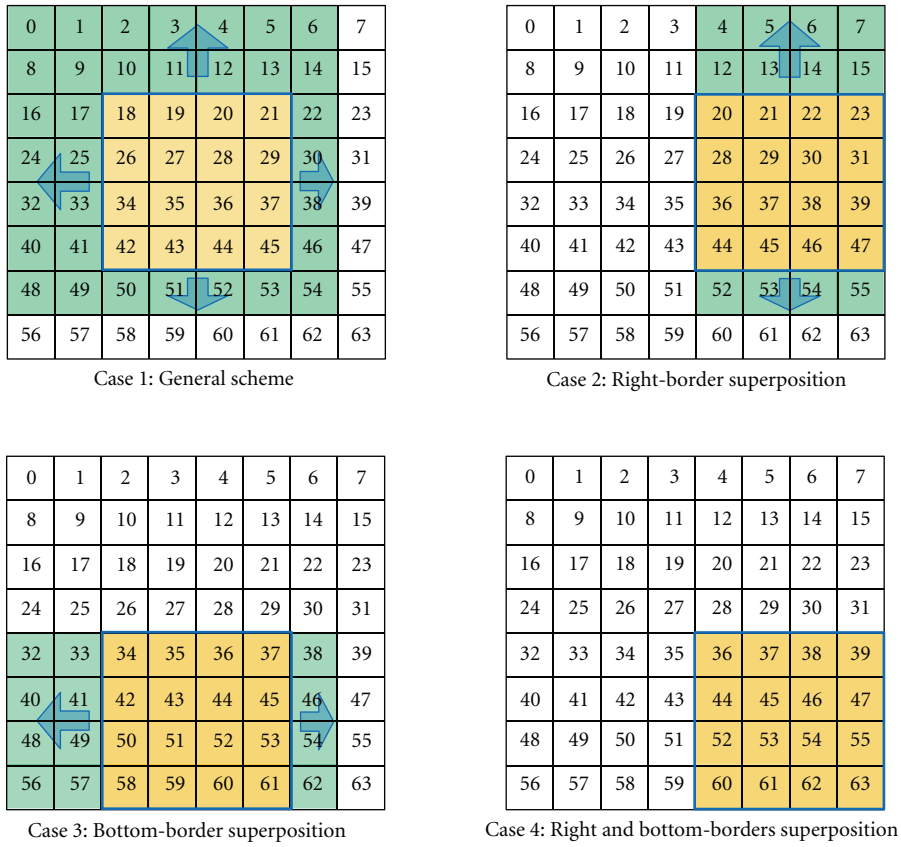
2.6. ViV Inserter. The following Sections 2.6.1. and 2.6.2. present the ViV inserter instructions and the implementation of the ViV inserter, respectively.

2.6.1. ViV Inserter Instructions. The proposed isolation schemes and corresponding profiles generation (as described in Sections 2.1 to 2.5 above), make it possible to distinguish between the four different ViV insertion locations, as depicted in Figure 12, which presents four different cases. The "Case 1" refers to the major and general insertion scheme, where the ROAD video can be placed in majority of mainstreams zones. For this case, we can change the ROAD area location (presented by the bright-colored blocks) according to the dark-colored MBs, as also shown by arrows.

Additionally to the above general case ("Case 1"), we have other three special ROAD position cases. Thus, in "Case 2", the right border of the ROAD area is superposed with the mainstream right border by slightly changing the isolation scheme. The same approach is also observed in the remaining two cases ("Case 3" and "Case 4"): in "Case 3", the bottom-border superposition is used, and in "Case 4", the both right and bottom borders superposition is used. It is noted that the indication regarding a particular case number ("Case 1", "2", "3" or "4") is conveyed to the ViV inserter within the profile ".dat" file. In turn, the ViV inserter uses this indication for selecting the corresponding insertion scheme to be used.

2.6.2. Implementation of the ViV Inserter. According to the proposed ViVCF scheme (as presented in Figure 1), the ViV inserter has four major inputs: two H.264 ViVCF coded video streams (i.e., "mainstream.264" and "ROAD.264"), and two sets of description profiles (i.e., the "profilers.dat" files provided from the Encoder 1 and Encoder 2). When the ViV inserter received the above two streams and their corresponding description profiles, it can initiate the ViV insertion process. Figure 13 below demonstrates the MB map of the typical H.264/AVC ViVCF frame.

This map represents all possible MBs (provided in various gray-scale colors), which may be affected in the isolation process. Only the affected MBs should be specially processed in the ViV insertion process. It is noted that the required ViV insertion process instructions, flags, and relevant coefficients are provided by the set of profiles within the profile ".dat" files, as described in Sections 2.1. to 2.5 above. All other (nonaffected) MBs are copied from the two H.264/AVC CF streams. The copy process is performed



Case 1: General scheme

Case 2: Right-border superposition

Case 3: Bottom-border superposition

Case 4: Right and bottom-borders superposition

Figure 12: Four sample ViV insertion schemes having different ViV insertion locations.

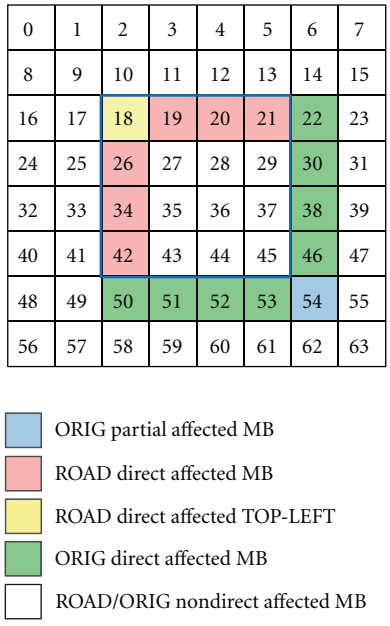


Figure 13: The Frame MBs Map for the ViV inserter.

Table 1: Test conditions for evaluating the presented ViV method and system.

Profile IDC	Baseline
Number of reference frames	5
Fast motion estimation	ON
RD optimization	ON
GOP structure	IPPP
Intra-Period	0
Frame rate	30
Overall number of frames	85
Spatial resolution	CIF (352 × 288)
CODEC	H.264/AVC JM 12

according to the bit counters, which can be provided within the "profile_1.dat" file (Figure 1). As a result, this video-in-video insertion scheme makes it possible to locate the ROAD video stream into any zone (portion) of the mainstream, according to Cases 1–4 presented in Figure 12 above. In turn, the ViV inserter is able to select the corresponding insertion scheme. Further, in Figure 14, we present an example of the proposed ViV inserter implementation by dividing the both given mainstream and ROAD frames into a number of virtual slices. Each virtual slice includes all required data: the start and end bit counter in the given frame, the stream type and corresponding variables, which can be updated from other profiles.

Table 2: Experimental results for the "*NEWS*" video sequence.

No.	The serial number of the top left MB	The serial number of the bottom right MB	Bit rate (Kbits/sec)	Bit rate overhead (%)
1	23	209	309.1	**3.0**
2	27	213	309.1	**3.0**
3	32	218	313.8	**4.6**
4	89	275	308.5	**2.8**
5	93	279	308.9	**3.0**
6	98	284	312.8	**4.3**
7	177	363	310.1	**3.4**
8	181	367	310.3	**3.4**
9	186	372	311.4	**3.8**
	Average bit rate/overhead		**310.4**	**3.5**

Table 3: Experimental results for the "*TEMPETE*" video sequence.

No.	The serial number of the top left MB	The serial number of the bottom-right MB	Bit rate (Kbits/sec)	Bit rate overhead (%)
1	23	209	1785.2	**0.1**
2	27	213	1788.8	**0.2**
3	32	218	1787.9	**0.1**
4	89	275	1788.0	**0.1**
5	93	279	1789.1	**0.2**
6	98	284	1788.9	**0.2**
7	177	363	1794.3	**0.5**
8	181	367	1789.0	**0.2**
9	186	372	1793.1	**0.4**
	Average bit rate/overhead		1789.4	**0.2**

Table 4: Experimental results for the "*CREW*" video sequence.

No.	The serial number of the top-left MB	The serial number of the bottom right MB	Bit rate (Kbits/sec)	Bit rate overhead (%)
1	23	209	991.43	**1.9**
2	27	213	995.74	**2.4**
3	32	218	1009.62	**3.8**
4	89	275	990.46	**1.8**
5	93	279	1001.12	**2.9**
6	98	284	1017.07	**4.6**
7	177	363	987.24	**1.5**
8	181	367	993.15	**2.1**
9	186	372	1001.93	**3.0**
	Average bit rate/overhead		998.6	**2.7**

Table 5: Experimental results for "*NEWS*", "*TEMPETE*", "*PARIS*", "*ICE*", "*CREW*", and "*BUS*", "*MOBILE*" video sequences.

No.	Video sequence	Average bit rate (Kbits/sec)	JM 12 bit rate (Kbits/sec)	Bit rate overhead (%)
1	*NEWS*	310.43	300.01	**3.5**
2	*TEMPETE*	1789.36	1786.08	**0.2**
3	*PARIS*	884.16	869.26	**1.7**
4	*ICE*	514.31	498.28	**3.2**
5	*CREW*	998.64	972.84	**2.7**
6	*BUS*	1825.6	1759.39	**3.8**
7	*MOBILE*	2693.47	2655.23	**1.4**

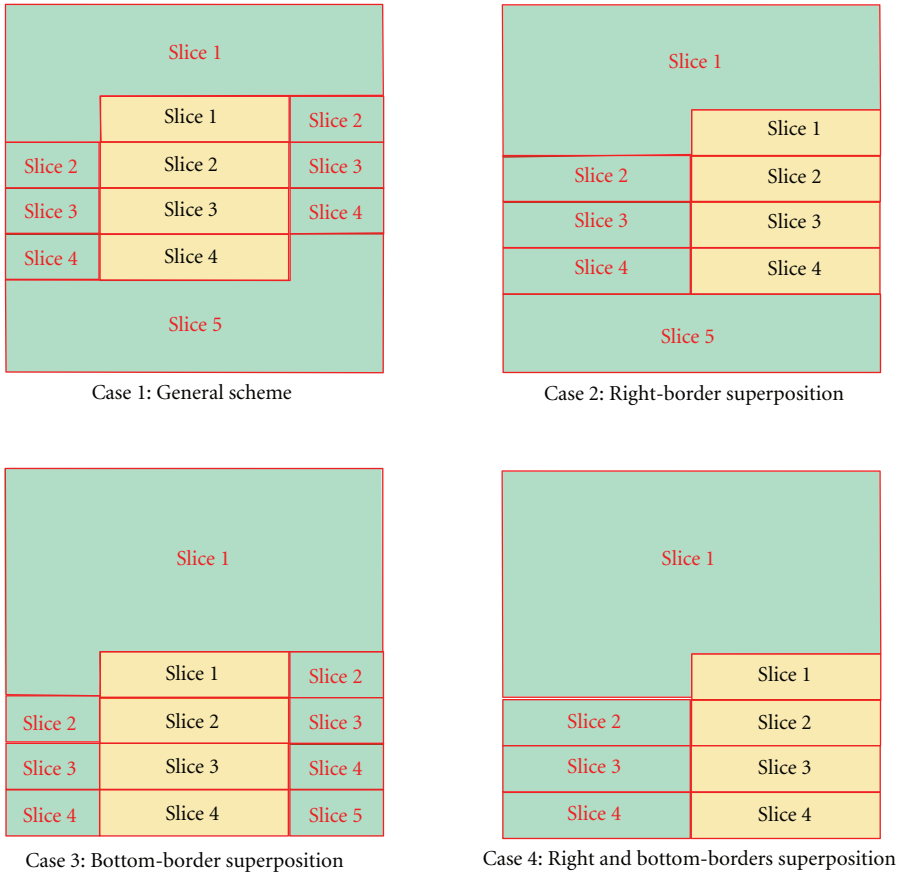


Figure 14: Dividing a frame into a number of slices.

In addition, the following Figure 15 schematically illustrates an example of the ViV insertion process for the MB-Level. This process includes five major steps as follows.

(a) For P_SLICE: since the "mb_skip_run" parameter does not exist in the first ROAD MB, when the previous MB in the mainstream is coded, a corresponding parameter (i.e., the bit "1") is added.

(b) For P_SLICE: the motion vector difference (MVD) should be recalculated according to the motion vectors (MVs) in the left, top, and top-left neighbors MB partitions in "profile_4".

(c) For I_SLICE and P_SLICE: the intra 4×4 modes in all adjusted blocks should be recalculated according to the intra 4×4 modes in the left and top neighbor 4×4 blocks in "profile_3".

(d) For I_SLICE and P_SLICE: the ΔQP should be recalculated according to the current macroblock QP and previous macroblock QP in "profile_5".

(e) For I_SLICE and P_SLICE: all adjusted 4×4 CAVLC coded blocks should be entropy coded according to the number of NNZ (nonzero) coefficients in the left and top neighbor 4×4 blocks in "profile_2".

3. Experimental Results

Table 1 presents general test conditions for evaluating the presented real-time ViV method and system, which is compared to the JM 12 (H.264/AVC reference software [7]). The used test platform is Intel Core 2 Duo CPU, 2.33 GHz, 2 GB RAM with Windows XP Professional operating system, version 2002, Service Pack 3.

Table 2 presents experimental results for the "*NEWS*" video sequence, in which the ROAD area (defined by the top-left and bottom-right MBs) is located at different positions. As is seen from Table 1, the average bit rate is 310.43 Kbits/sec, and the corresponding average bit rate overhead is 3.5% compared to the bit rate of 300.0 Kbits/sec of the JM12 (H.264/AVC Reference Software version 12). Thus, we achieve a significantly low bit rate overhead, thereby enabling to efficiently employ the same channel bandwidth substantially without decreasing the visual quality. Further, according to the proposed ViV insertion method, the run time can be only 5% of the original JVT encoder run time, thereby saving up to 95% in the run-time of the H.264/AVC encoder and decoder CASCADE (based on the Relative CPU (RCPU) performance).

Table 3 presents experimental results for the "*TEMPETE*" video sequence (average bit-rate overhead: 0.2% compared to the bit rate of 1786.1 Kbits/sec of the JM12).

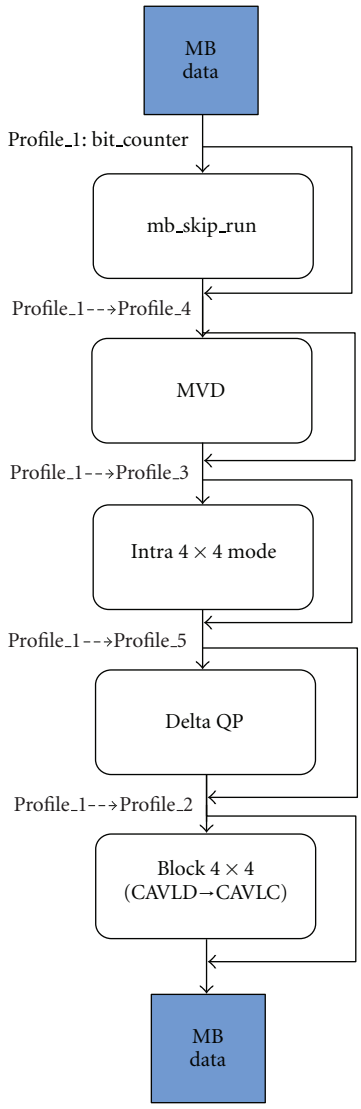


Figure 15: Example of the MB-level ViV insertion process.

Similarly, Table 4 presents experimental results for the "*CREW*" video sequence (average bit rate overhead: 2.7% compared to the bit rate of 972.8 Kbits/sec of the JM12).

Table 5 summarizes experimental results (performed according to the test conditions specified in Table 1) for seven different video sequences: "*NEWS*", "*TEMPETE*", "*PARIS*", "*ICE*", "*CREW*", "*BUS*", "*MOBILE*".

As is clearly seen from the experimental results above, according to the proposed ViV insertion scheme, there is a significantly low bit rate overhead, which varies from 0.1% to 3.8% only. Also, based on the above conducted experiments, the average PSNR quality remains substantially the same compared to JM 12: *NEWS*: 40.1 dB, *TEMPETE*: 37.2 dB, *PARIS*: 37.8 dB, *ICE*: 41.8 dB, *CREW*: 39.3 dB, *BUS*: 37.1 dB, and *MOBILE*: 36.8 dB.

It should be noted that the proposed method for performing the efficient real-time Video-in-Video (ViV) insertion can be implemented in a similar manner for the Scalable Video-Coding (SVC) schemes [8–12], and particularly for the Region-of-Interest (ROI) video-coding systems and applications [13–19].

4. Conclusions

In this work we presented an efficient method for the fast real-time Video-in-Video (ViV) insertion, thereby enabling efficiently inserting a video sequence into a predefined location within a pre-encoded video stream (e.g., for inserting advertisements into the TV stream). The proposed ViV insertion method and system enable achieving a significant performance over the conventional brute-force approaches, in terms of the bit rate and insertion run time, and have a significantly low bit rate overhead. Also, the proposed ViV insertion method and system enable supporting multiple rectangular overlays of various sizes (e.g., $16N \times 16M$ sizes, where N and M are integers). According to the experimental results, there is a significantly low bit rate overhead (up to 3.8%), while there is substantially no degradation in the PSNR quality.

Acknowledgments

This work was supported by the NEGEV consortium, MAGNET Program of the Israeli Chief Scientist, Israeli Ministry of Trade and Industry under Grant no. 85265610. The authors are grateful to Alexey Minevich, Udi Levy, Michael Degtyar, Yoav Naaman, and Maoz Loants for their assistance in testing and evaluation.

References

[1] Y. Bamel and D. Vardi, "H.264 content insertion in the coding domain," SIPL Annual Projects Presentation, 2008, http://sipl.technion.ac.il/new/Archive/Annual_Proj_Pres/sipl2008/Posters/Cont_Ins.pdf.

[2] M. Ziv and D. Brot, "Content insertion into H.264 compressed video," SIPL Annual Projects Presentation, 2008, http://sipl.technion.ac.il/new/Archive/Annual_Proj_Pres/sipl2008/Posters/compres_cont_insrt.pdf.

[3] N. Goldfarb and O. Rottenstreich, "Content insertion into H.264 compressed video: supporting B frames," SIPL Annual Projects Presentation, 2008, http://sipl.technion.ac.il/new/Teaching/Projects/ProjPosts/H264-Cont-Ins.pdf.

[4] "*H.264/AVC*, Draft ITU-T Rec. and Final Draft Intl. Std. of Joint Video Spec," (H.264/AVC), Joint Video Team, Doc. JVT-G050, March 2003.

[5] I. E. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*, John Wiley & Sons, Chichester, UK, 2003.

[6] J. S. Park and H. J. Song, "Selective intra prediction mode decision for H.264/AVC encoders," *Transactions on Engineering, Computing and Technology*, vol. 13, pp. 51–55, 2006.

[7] A. M. Tourapis, "H.264/14496-10 AVC Reference Software Manual," Doc. JVT-AE010, July 2009.

[8] Applications and Requirement for Scalable Video Coding, JVT ISO/IEC JTC1/SC29/WG11, Doc. N6880, Hong-Kong, China, January 2005.

[9] "*H.264/AVC*, Draft ITU-T Rec. and Final Draft Intl. Std. of Joint Video Spec," (H.264/AVC), Joint Video Team, Doc. JVT-G050, March 2003.

[10] T. Wiegand et al., "ISO/IEC 14496-10:200X/Amd.3 Part 10: Advanced Video Coding—AMENDMENT 3: Scalable Video Coding Joint Draft ITU-T Rec. H.264/ISO/IEC 14496-10/Amd.3 Scalable video coding," Joint Video Team, Doc. JVT-X201, July 2007.

[11] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[12] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien, "Joint draft 8 of SVC amendment," Tech. Rep., ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6 9 (JVT-U201), Hangzhou, China, 2006.

[13] D. Grois, E. Kaminsky, and O. Hadar, "Dynamically adjustable and scalable ROI video coding," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB '10)*, pp. 1–5, March 2010.

[14] Z. Lu et al., "CE8: ROI-based Scalable Video Coding," Doc. JVT-O308, Busan, KR, April 2005.

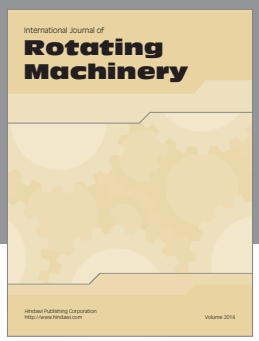
[15] D. Grois, E. Kaminsky, and O. Hadar, "ROI adaptive scalable video coding for limited bandwidth wireless networks," in *Proceedings of the International Federation for Information Processing Wireless Days (IFIP '10)*, pp. 1–5, October 2010.

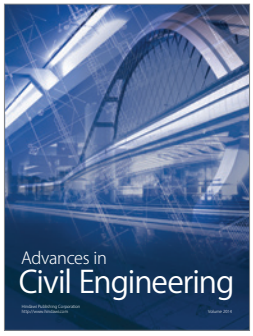
[16] T. C. Thang et al., "Spatial Scalability of Multiple ROIs in Surveillance Video," JVT-O037, Busan, KR, April, 2005.

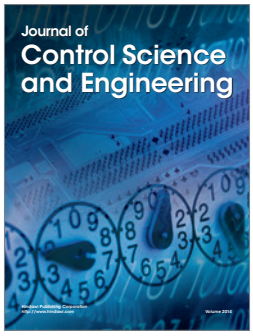
[17] Z. Lu, "Perceptual Region-of-interest (ROI) based Scalable Video Coding," Doc. JVT-O056, Busan, KR, April 2005.

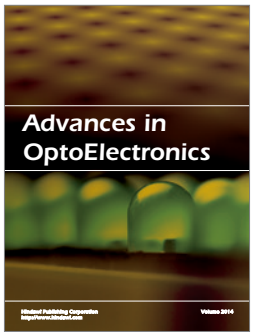
[18] D. Grois, E. Kaminsky, and O. Hadar, "Adaptive bit-rate control for region-of-interest scalable video coding," in *Proceedings of the 26th Convention of IEEE of Electrical and Electronics Engineers in Israel (IEEEI '10)*, pp. 000761–000765, November 2010.

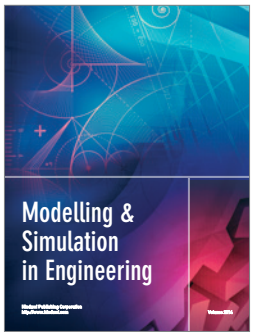
[19] P. Lambert, "A real-time content adaptation framework for exploiting ROI scalability in H.264/AVC," *Advanced Concepts for Intelligent Vision Systems*, pp. 442–453, 2006.

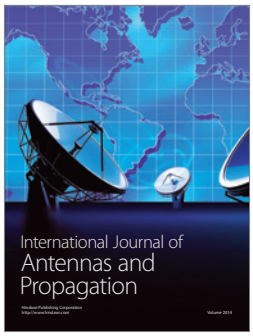


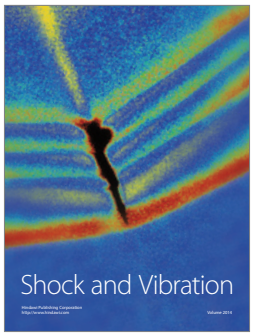


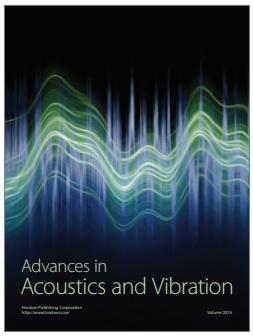












Hindawi

Submit your manuscripts at
http://www.hindawi.com