Arbitrarily Shaped Virtual-Object Based Video Compression

Naresh Sharma*, Yuan F. Zheng*, Eric J. Balster†

*Dept. Electrical and Computer Engineering

The Ohio State University, Columbus, OH, 43210, USA

Email: {sharman, zheng}@ece.osu.edu

†Dept. Electrical and Computer Engineering

University of Dayton, OH, 45469, USA

Email: balsteej@notes.dayton.edu

Abstract

Object based compression techniques are widely believed to have the potential to give the best compression results for a given signal quality. However, true object tracking and extraction is difficult and computationally very expensive. Therefore, an arbitrarily shaped virtual-object compression method is developed. The method is similar to the object based compression methods in that it separates the changing portion of the video from the stationary portion, and encodes each independently. The changing portion of the video is grouped as a 3D arbitrarily shaped virtual object whereas the unchanged portion of the video is grouped as background. The arbitrarily shaped virtual object is coded using 3D wavelet compression whereas stationary background is coded as a single frame using 2D wavelet compression. Experimental results demonstrate that the newly developed method outperforms 3D wavelet compression and the rectangular virtual-object compression by achieving higher compression ratio at a higher PSNR.

Index Terms

Arbitrarily shaped virtual-object, background, shape adaptive wavelet transform, 3D wavelet compression, shape coding, texture coding, multiple virtual-objects, group of frames.

Arbitrarily Shaped Virtual-Object Based Video Compression

I. Introduction

With the arrival of MPEG-4 [1, 2], object based video compression has received more and more attention [3–5]. Previous compression standards such as MPEG-1 [6], and MPEG-2 [7] were block based compression method where the primary element of processing was a macroblock. In object based compression schemes, arbitrarily shaped video objects are identified, extracted, and coded separately providing new functionalities and more flexibility. More importantly, object based compression looks promising to achieve better compression while maintaining good video quality. Therefore, with the focus shifted to object based compression methods, much research has been conducted in the area of shape coding [8–10], texture coding [3, 11–14], and methods of object identification and tracking [15, 16].

The most critical step in object based compression is accurate object extraction. At the same time this is the most difficult part. Given the variety of video scenes, accurate object identification as well as tracking across the frames is difficult. Moreover, such object extraction methods are typically computationally expensive, and no single method can be applied to all types of video scenes. This makes object identification and extraction in the real-time really difficult.

Another issue to consider is whether to use Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) for video coding. DWT based methods have been shown to outperform DCT based methods in terms of PSNR [17]. Also, DWT based methods do not suffer from blocking artifacts as usually is the case with DCT based methods at low bit-rates [18]. In addition, lifting scheme [19, 20] can be utilized to further accelerate the wavelet coefficients computation process. It has been shown that DWT has lower computational complexity of O(N) in comparison to computational complexity of O(NlogN) in case of DCT [21]. Because of these advantages of DWT based methods over DCT based methods, many wavelet based compression techniques [4, 22–29] have been designed targeting video applications. Karlsson et. al. proposed for the first time a 3D wavelet transform based video compression method [29]. He et. al. also proposed a 3D wavelet transform based video coding method [28]. These 3D wavelet transform

based compression methods divide a given video sequence into groups of frames (GoF) and then, 3D wavelet compression is applied to each GoF. However, these methods fail to utilize the temporal redundancy across the GoF, and thus, result in coding more wavelet coefficients than actually needed.

Balster et. al proposed a wavelet transform based rectangular virtual object (RVO) compression method where a RVO covering the points in motion is identified and extracted instead of tracking and extracting the real objects [4]. The extracted RVO is compressed in three dimensions using 3D wavelet compression [28]. The remaining non-changing video portion is grouped as background and coded using 2D wavelet compression. The method achieves good compression results in comparison to the 3D wavelet compression method as it partially utilizes the temporal redundancy by separating the changing and non-changing portions of the video. Additionally, this method overcomes the difficulty in identification and tracking of real objects by restricting the virtual-object shape to be rectangular. However, a significant part of the non-changing video portion is extracted as a part of the RVO, and is compressed in 3D because of this restriction on the shape of the virtual-object. As a result, even this method can not utilize the temporal redundancy to the maximum extent possible.

Because a rectangular object restriction is used in [4], potentially large portions of true background are mis-classified as object information in the separation process. Therefore, a part of true background is encoded using 3D wavelet compression degrading the compression ratio. The compression performance can be further improved by removing rectangular object restriction which in turn necessitates the availability of wavelet transform methods that can be applied to any shape. In recent years, there has been significant amount of research to develop wavelet transform techniques that can be applied to arbitrary shapes and now there are many such methods available [11, 12, 30]. One such method known as Shape Adaptive Wavelet Transform was developed by Li et al. [12]. This shape adaptive wavelet transform technique works by identifying a segment of pixels in the given arbitrarily shaped object and then, transforming it after applying symmetrical extension.

Thus, an arbitrarily shaped virtual-object (ASVO) compression method is developed that can provide some benefits of the object-based compression without the difficulties of real object based compression. Moreover, the method also overcomes the shortcomings of RVO compression [4] in that it does not restrict the shape of virtual-object to be rectangular. The virtual object can

be of any arbitrary shape, and can be coded using shape adaptive wavelet transform based 3D wavelet compression.

To separate the ASVO from the stationary portion, nondecimated wavelet transform is applied in the temporal domain. Applying the wavelet transform in temporal domain results in large wavelet coefficient values corresponding to locations where pixel values are changing significantly across frames. Once the wavelet coefficients are computed, a binary mask is created by applying a motion threshold. It is the binary mask which defines the shape of the virtual object in each frame, and therefore, virtual object can be of any arbitrary shape corresponding to only the changing portion in the video.

Furthermore, idea of multiple arbitrarily shaped virtual objects (MASVO) is presented. The RVO compression method proposed in [4] limits the number of virtual objects to be one for any video scene. However, a given video scene can have multiple moving objects. Moreover, if these multiple moving objects are far apart from each other in the video frame then enclosing all these moving objects in one RVO will degrade the compression performance. In such a case, a large non-changing portion of the video is compressed in 3D as it will be considered as a part of the virtual object. Therefore, MASVO are used to effectively segregate the moving portion of the video from stationary background, improving the compression ratio.

The rest of the paper is organized as follows. Following the introduction, Section II gives a brief description of shape adaptive wavelet transform [12, 30] needed for spatial decomposition of the ASVO and a brief description of RVO compression [4]. Section III describes the ASVO compression and discusses various aspects of it. Section IV presents the experimental results of the new compression method. A comparison is made to the pure 3D wavelet compression as well as to the RVO compression. Section V concludes the paper.

II. Shape Adaptive Wavelet Transform and RVO Compression

A. Shape Adaptive Wavelet Transform

Conventional wavelet transform methods can only be applied to rectangular shapes. However, in MPEG-4 [1], the basis of processing is an audio-visual object which can be of any arbitrary shape. With more and more research being carried out in the field of object based compression, there is a need to have methods to effectively decompose any arbitrary shape. There are already few such methods available today [11, 12, 30, 31]. In this subsection, we briefly review one such

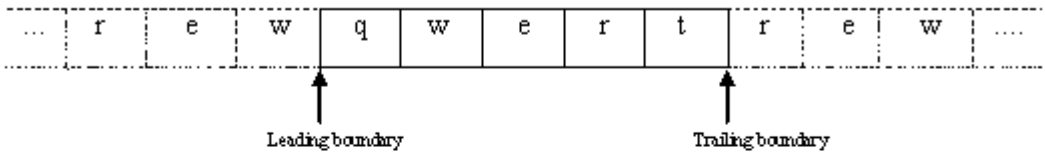


Fig. 1. Symmetric extension for an odd symmetric biorthogonal wavelet filter at the leading and trailing boundaries

method known as shape adaptive wavelet transform [12, 30] which we are using extensively for our ASVO compression method.

The shape adaptive wavelet transform for the spatial decomposition of the frames consists of mainly three steps – 1) *boundary extension*, 2) *wavelet transform*, and 3) *subsampling*. The method is described here for odd symmetric biorthogonal wavelet filter such as a 5/3 wavelet filter which is also the filter we use in our ASVO compression method. Other types of filter follow the same steps though with some minor differences [12].

1) Boundary Extension: In order to apply shape adaptive wavelet transform to a given signal, the segment belonging to the arbitrarily shaped object is identified first in the given row/column for horizontal/vertical wavelet transform. The identified signal segment is then extended at the leading and trailing boundaries. The undefined pixel locations of the signal segments are filled with relevant data from inside the signal segment to maintain the perfect reconstruction property of wavelet transform. Figure 1 demonstrates the symmetrical signal extension of a length 5 signal segment.

2) Arbitrary Length Wavelet Transform : Once a signal segment is boundary extended, wavelet transform is easily applied. To take the wavelet transform, we need to identify the segment start and the segment end in a given row or column. Moreover, there could be multiple segments in the same row or column. Each of these segments are transformed separately.

Let the object be defined using a binary mask denoted by *m(i)* where a value of 1 means the pixel belongs to the object. Let the start and end of first segment be defined by *seg_strt* (index of first *1* in object mask *m(i)* after a *0*) and *seg_end* (index of last *1* after *seg_strt* and before a *0* in the object mask *m(i)*) respectively. The length of the signal segment is then given by *N=seg_end - seg_strt+1*.

If *N=1*, the single element of the segment is repeatedly extended, and either low pass or high

pass filter is applied to the signal. The decision whether to apply a low pass or high pass wavelet filter depends upon the *subsampling strategy* explained next. The resulting wavelet coefficient is then placed in either low pass or high pass band depending upon which filter was applied.

If *N* is greater than *1*, symmetrical signal extension is applied as before. To the symmetrically extended signal segment, low pass and high pass filters are applied at the alternate pixel locations starting from *seg_strt* and ending at *seg_end*. Decision about which filter should be applied first at position *seg_strt* depends upon the subsampling strategy. The decomposition generates same number of low and high pass coefficients for an even length segment (*N = even*), and generates one more low pass or high pass coefficient for an odd length segment(*N = odd*), depending upon the subsampling strategy.

3) Subsampling Strategy: There are two types of subsampling strategies namely *local subsampling* and *global subsampling* that decide which pixels in the segment will be used in low pass filtering and which will be used for high pass filtering. In *local subsampling*, subsampling positions are decided as per the positions relative to the start of a signal segment whereas in *global subsampling*, subsampling positions refer to the positions relative to the beginning of the bounding box of the visual object [12]. *Local even subsampling* refers to the subsampling strategy where the filtering operation is applied at the even locations with respect to the start of a signal segment whereas in local odd subsampling, filter is applied at the odd positions with respect to the start of the signal segment. By locally fixing subsampling positions, we can ensure that number of low pass coefficients are always greater than or equal to the number of high-pass coefficients. However, by locally fixing subsampling position, phases of some of the low pass and some of the high-pass wavelet coefficients are skewed by one sample [12]. As a result, the local correlation among the coefficients is not preserved, and the efficiency of the wavelet transform in the second direction is degraded in case of 2D wavelet transform. Also during the decode process, the exact location of the decoded pixel can not be determined accurately. There can always be an offset of 1 from the exact location.

Fixing the subsampling positions globally, i.e. applying *global subsampling* strictly preserves the spatial relations and therefore, improves the efficiency of wavelet transform in the 2^{nd} spatial dimension. In this case, phase of the filter is fixed with respect to the global positions. Also, to achieve global even or odd subsampling, local subsampling positions have to be adjusted for each segment as starting position of each segment may not be always at even or odd positions

relative to the bounding box [12] of the arbitrarily shaped object. For example, to achieve global even subsampling in low-pass bands and global odd subsampling in high-pass bands, local even subsampling needs to be applied in the low pass band and local odd subsampling in the high pass band if the signal segment is starting at even index with respect to the bounding box of the visual object whereas local odd subsampling needs to be applied in the low pass band and local even subsampling in the high pass band if the segment is starting at odd index with regard to the bounding box.

Wavelet decomposition with globally fixed subsampling positions results in same number of low and high pass band coefficients if segment length is even but if segment length is odd, this can sometimes result in more number of high pass wavelet coefficients than number of low pass coefficients.

In our newly proposed ASVO video compression method, we use global even subsampling in low-pass bands and global odd subsampling in high-pass bands as such a strategy preserves the pixel locations, and results in more signal processing gain [12].

4) 2-D Shape Adaptive Wavelet Transform: Based on the above discussions, 2D shape adaptive wavelet transform process can be described as below:

1) *Find out the shape information of the object as well as the rectangular box enclosing the arbitrarily shaped object.*
2) *Using the arbitrary shape information calculated in step 1, identify the first segment of consecutive pixels in the current row.*
3) *Symmetrically extend the signal segment at the leading and trailing boundaries as shown in Figure 1.*
4) *Apply the low-pass and high-pass wavelet analysis filters to the symmetrically extended segment. Low pass analysis filter is applied such that center of the filter is always at the even indices of the original segment. High pass analysis filter is applied such that its center is always at the odd indices of the original segment.*
5) *Store the wavelet coefficients in their respective low pass and high pass band. If within a frame/bounding box, original segment starts at index 2i in a given row, the low pass as well as the high pass coefficients are stored from index i onwards in their respective low pass and high pass band. On the other hand, if the original segment starts at index 2i+1, low pass coefficients are stored from index i onwards in the low pass band whereas the*

high-pass coefficients are stored from index i+1 onwards in the high pass band.

6) *Update the shape information as per the wavelet coefficients position. This is done so that we have the wavelet coefficients position information and using this information, wavelet transform in the vertical direction or at the next higher scale level can be taken.*
7) *Apply the steps 3-6 on the next identified segment of consecutive pixels in this row.*
8) *Perform the above steps for the next row of pixels.*
9) *Once done with all the rows, follow the above steps for each column of the 1-D transformed frame to get the 2-D transform of arbitrary shaped object.*
10) *Repeat all the steps as many number of times as the number of levels of wavelet decomposition desired to the low pas-low pass band of the 2D decomposed object.*

B. RVO Compression

RVO compression scheme works by separating the motion exhibiting portion from the stationary portion of the video. The portion of the video that exhibits motion is extracted as a RVO. To extract the virtual-object, nondecimated wavelet transform is applied in the temporal domain. Applying the wavelet transform in the temporal domain results in large wavelet coefficients corresponding to locations where pixels are changing significantly.

Using the resulting wavelet coefficients values, a binary mask is created identifying changing and non-changing locations in the video. The value of a wavelet coefficient is compared to pre-determined motion threshold value. If the coefficient value is greater than the threshold, corresponding pixel belongs to an object in motion, and the binary mask value at the corresponding location is set to 1 otherwise it is set to 0. Therefore,

$$M_{vo}[x,y,z] = \begin{cases} 1, & |\lambda_{vo}[x,y,z]| > \tau_{vo} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $M_{vo}[.,.,z]$ is the resulting binary mask for each frame z, τ_{vo} is the experimentally determined motion threshold value, and $\lambda_{vo}[x,y,z]$ are the wavelet coefficients. A better estimate of motion threshold can be calculated using the method described in [32].

Once the binary mask is known for each frame, a rectangle covering all the 1s across all the frame is found. This rectangle defines the starting and ending position of the virtual object in each frame. The portion of the frame outside the rectangle is treated as part of background. Following RVO extraction, RVO is compressed using 3D wavelet compression.

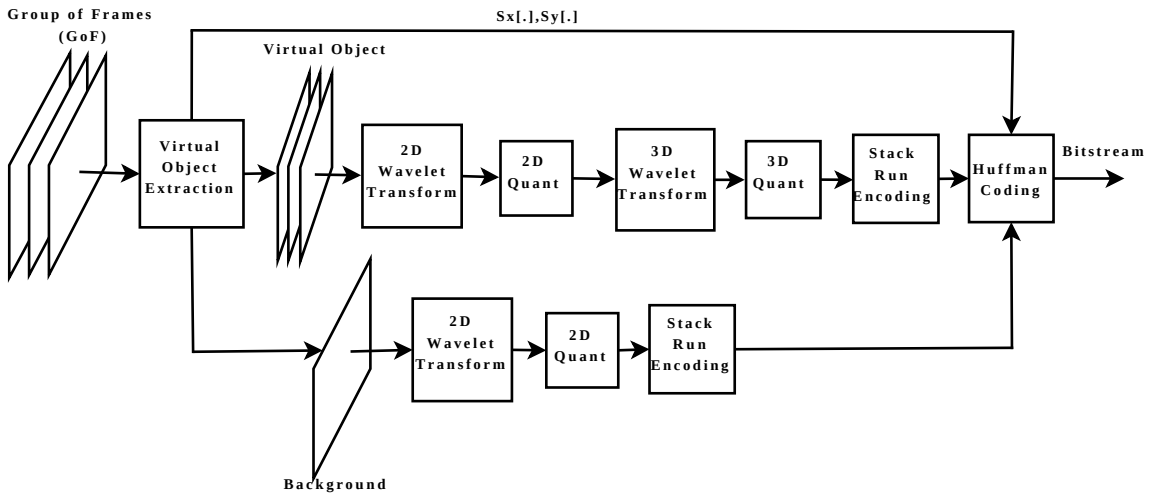


Fig. 2. RVO video compression where $S_x[.], S_y[.]$ indicate the position of the virtual-object.

The background is compressed using 2D wavelet compression as only spatial redundancy can be exploited. It should be noted that corresponding to a group of frames, a single 3D virtual object and only one background frame is extracted. By definition background is the non-changing portion of the video and therefore, does not change across the group of frames. Thus, only one background frame needs to be sent for a given group of frames. Figure 2 gives the overall flow of the virtual-object compression method.

III. ASVO Based Video Compression

An ASVO compression method is proposed that further explores the benefits of the object-based compression without the difficulties of real object based compression. Moreover, the method overcomes the shortcomings of RVO compression [4] in that it does not restrict the shape of virtual-object to be rectangular. Therefore, our newly proposed method better exploits temporal domain redundancies in the video by effectively separating stationary background from ASVO.

Furthermore, idea of MASVO is presented. The RVO compression method proposed in [4] does not consider the case of multiple virtual-objects. A given video scene can have multiple moving objects. Moreover, if these multiple moving objects are far apart from each other in the video frame then enclosing all these moving objects in one RVO degrades the compression

performance. In such a case, a large non-changing portion of the video is considered as a part of the RVO and compressed using 3D wavelet transform thereby degrading compression ratio. Therefore, to effectively segregate the moving portion of the video from stationary background, and to further improve the compression ratio, idea of multiple virtual objects is also presented.

A. ASVO

The ASVO corresponds to the portion of the video which is changing and needs 3D compression. As it could be of any arbitrary shape, it is defined by a binary object mask where a value of 1 means the corresponding location is the part of the virtual object. The dimensions of the object mask are given by W_f and H_f where W_f is the width and H_f is the height of the frame.

Let us define $f(.)$ as an image sequence of a video scene. To determine the ASVO, 3D nondecimated wavelet transform is applied in the temporal domain given by

$$\psi_{asvo}[x, y, z] = \sum_{k} f(x, y, k) g_{asvo}[k - z] \tag{2}$$

where ψ_{asvo} are the resulting coefficient values, and g_{asvo} are the wavelet filter coefficients

Haar Wavelet function is used to determine the ASVO because of its compact support, and a well known fact that biorthogonal Haar wavelet function gives the best results for motion identification [4]. In fact, applying the Haar wavelet filter is equivalent to a pixel to pixel difference between consecutive frames. Therefore, spatial and temporal location of the motion can be easily located by testing the wavelet coefficients' values against a motion threshold.

A 3D binary mask determining motion from non-motion is computed by applying a motion threshold check on the wavelet coefficient values:

$$TM_{asvo}[x, y, z] = \begin{cases} 1, & |\psi_{asvo}[x, y, z]| > \tau_{asvo} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $TM_{asvo}[., ., z]$ is the resulting binary mask for each frame z, τ_{asvo} is the experimentally determined motion threshold value. An estimate of τ_{asvo} can be calculated using method described in [32].

As with the RVO approach [4], binary mask is further refined by applying the spatial support criteria defined in [33]. Therefore,

$$BM_{asvo}[x,y,z] = \begin{cases} 1, & |S_{asvo}[x,y,z]| > s_{asvo} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $S_{asvo}[.]$ and s_{asvo} are determined from $TM_{asvo}[.]$ using the method given in [32, 33].

The resulting 3D refined binary mask provides information about where a pixel value in a given frame is changing significantly from its previous frame in the GoF. These significantly changing locations are subjected to 3D compression [28] across the whole GoF to preserve temporal domain changes. Therefore, if the value 1 appears in BM_{asvo} at a location specified by $[x,y,z]$, pixel values at corresponding location $[x,y]$ are transformed in the temporal domain irrespective of frame number z. Complete information of the points needing 3D compression, or the 2D shape of the ASVO in each frame is therefore given by a 2D mask $OM_{asvo}[x,y]$ which is computed simply by applying logical OR operation to the 3D mask BM_{asvo} across the z-direction as:

$$OM_{asvo}[x,y] = \begin{cases} 1, & \sum_z BM_{asvo}[x,y,z] > 0 \\ 0, & \text{else.} \end{cases} \quad (5)$$

Thus, the object mask is a 2D mask giving the pixel locations that belong to the ASVO in each frame. Again, it should be noted that virtual object is a 3D object with the object mask defining 2D arbitrary shape of the 3D object in each frame, the third dimension being the number of frames in the current group of frame.

Using the binary shape mask information, ASVO can be defined as:

$$ASVO(x,y,z) = f(x,y,z)OM_{asvo}[x,y] \quad (6)$$

where $ASVO(x,y,z)$ is the 3D ASVO corresponding to the image sequence $f(x,y,z)$ with total number of frames being F, and $0 \leq x < W_f$, $0 \leq y < H_f$, and $0 \leq z < F$.

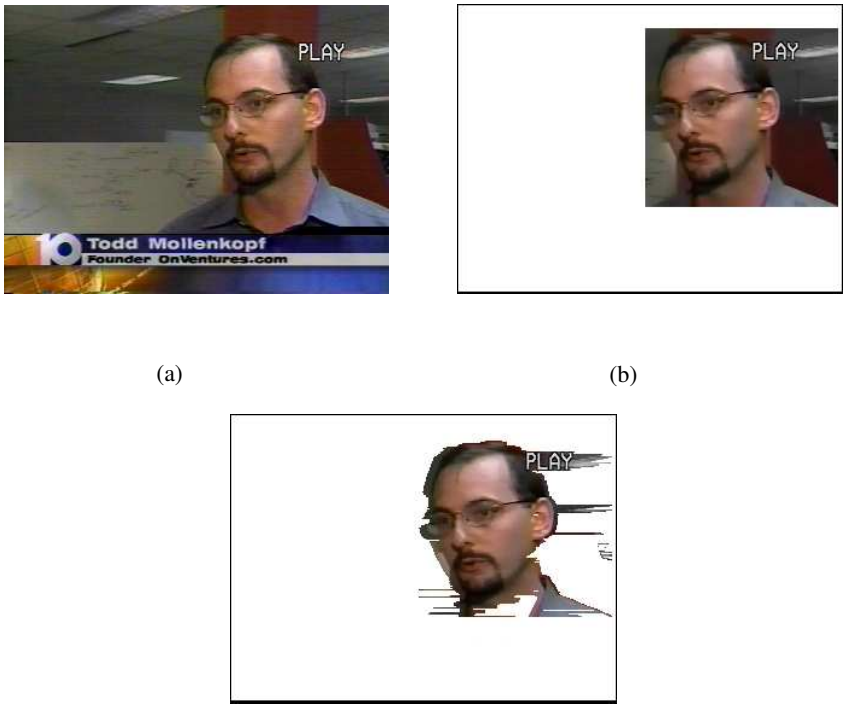
The 2-D background $b_{asvo}(x,y)$ is defined as:

$$b_{asvo}(x,y) = \begin{cases} \frac{\sum_{z=0}^{F-1} f(x,y,z) \times \alpha[x,y,z]}{\sum_{z=0}^{F-1} \alpha[x,y,z]}, & \sum_{z=0}^{F-1} \alpha[x,y,z] \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where

$$\alpha[x,y,z] = \begin{cases} 1, & OM_{asvo}[x,y] = 0 \\ 0, & \text{else.} \end{cases} \quad (8)$$

Figure 3 shows an original frame of "10TV" sequence, the RVO and the newly proposed ASVO.



(a) (b)

(c)

Fig. 3. ASVO Extraction (frame #22 of "10TV" image sequence is shown). (a) Original Frame, (b) RVO, and (c) ASVO. Note that the stationary background is intentionally shown white for clarity purpose. Only the colored portion shown in (b) and (c) belongs to the virtual-object and is compressed in 3D. As colored portion is much less in (c), compression ratio will be higher in the ASVO compression method.

B. ASVO Based Video Compression

Object mask computation defines the ASVO region as well as the background. The background is compressed using conventional 2D wavelet compression where first the 2D wavelet transform of the background is taken followed by 2D quantization [28, 34], stack-run encoding [35] and Huffman coding. The ASVO compression uses 3D wavelet compression method with one significant difference.

In 3D wavelet compression [28, 29], spatial decomposition of the frame is done using conventional wavelet transform, one that can only be applied to rectangular frames. However, in the proposed method, the conventional 2D wavelet transform cannot be directly applied due to the arbitrary shape of the virtual object. Therefore, the shape adaptive wavelet transform [11, 12, 30] is used to code the virtual object. The method is also briefly described in II-A. Spatial

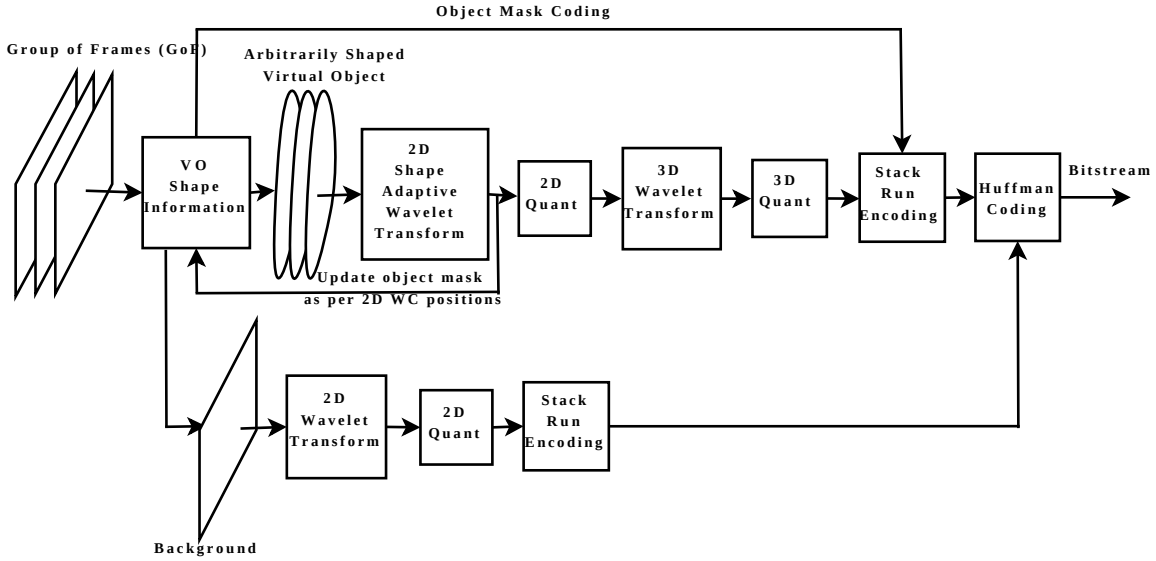


Fig. 4. ASVO Video Compression

decomposition of the ASVO using shape adaptive wavelet transform technique in each frame is followed by 2D quantization [28, 34], temporal domain decomposition, 3D quantization [28, 34], stack run encoding [35] and Huffman coding.

In addition to background and virtual-object coding, shape coding of the virtual object or the binary mask coding is also accomplished to enable the decoder to correctly decode virtual-object and the background. Since the shape information is merely a mask of 1s and 0s, it is directly subjected to stack run encoding followed by Huffman coding. Figure 4 shows the design flow of the basic ASVO compression method.

C. Number of Virtual Objects

ASVO extraction identifies the significantly changing pixels across the GoF. Spatial decomposition of such pixels is done using 2D shape adaptive wavelet transform in each frame. As stated in Subsection II-A and discussed in detail in [12], a rectangular box enclosing the arbitrarily shaped object is needed to compute the position where a given wavelet coefficient should be stored. There are many ways to find this bounding box. Number of such bounding boxes defines number of ASVOs. Accordingly, there can be two types of compression mechanism – *single*

arbitrarily shaped virtual object (SASVO) video-compression mechanism, and *multiple arbitrarily shaped virtual objects (MASVO)* video-compression mechanism. Each of these methods has their own advantages and disadvantages which are discussed here.

1) SASVO Video Compression: This is the most straightforward method where the frame itself can be considered as a bounding box. The arbitrary shaped defined by the object mask is considered to be belonging to a single object bounded by the frame boundaries. No additional computations are needed to compute the bounding box in this case, and the global subsampling positions to apply wavelet filters [12] are also easily defined with respect to the start of the row/column of the original frame. For example, to achieve global even subsampling in low-pass bands and global odd subsampling in high-pass bands, local even subsampling needs to be applied in the low pass band and local odd subsampling in the high pass band if the signal segment is starting at even index with respect to the start of row/column whereas local odd subsampling needs to be applied in the low pass band and local even subsampling in the high pass band if the segment is starting at odd index with respect to the start of row/column. Figure 4 is a basic design-flow of an SASVO video-compression.

There are few problems with this approach however. In a given video sequence, there could be many real moving objects. Corresponding to these moving objects, the extracted object mask have clusters of 1s at positions belonging to these moving objects. Enclosing all these clusters of 1s (different moving objects) by a single rectangular box and then carrying out the shape adaptive spatial decomposition results in high pass band coefficients with relatively large magnitudes, and in low pass band coefficients with relatively large differences in magnitude. At the next level of wavelet decomposition, this again results in relatively large high-pass coefficient values and so on. As a result, in case of video scenes with multiple moving objects, the coding efficiency degrades and the compression ratio is slightly reduced.

2) MASVO Video Compression: In MASVO video compression mechanism, all the 1s of the object mask are not enclosed inside a single rectangular box. Instead an attempt is made to enclose all different objects in different bounding boxes. Usually, multiple moving objects in the video sequence results in multiple clusters of 1s in the object mask. As discussed above, if all these different clusters of 1s corresponding to different objects are enclosed within a single rectangle, coding efficiency is degraded. At the same time, a bounding box is needed to define subsampling strategy for shape adaptive wavelet transform. Therefore, to effectively distinguish

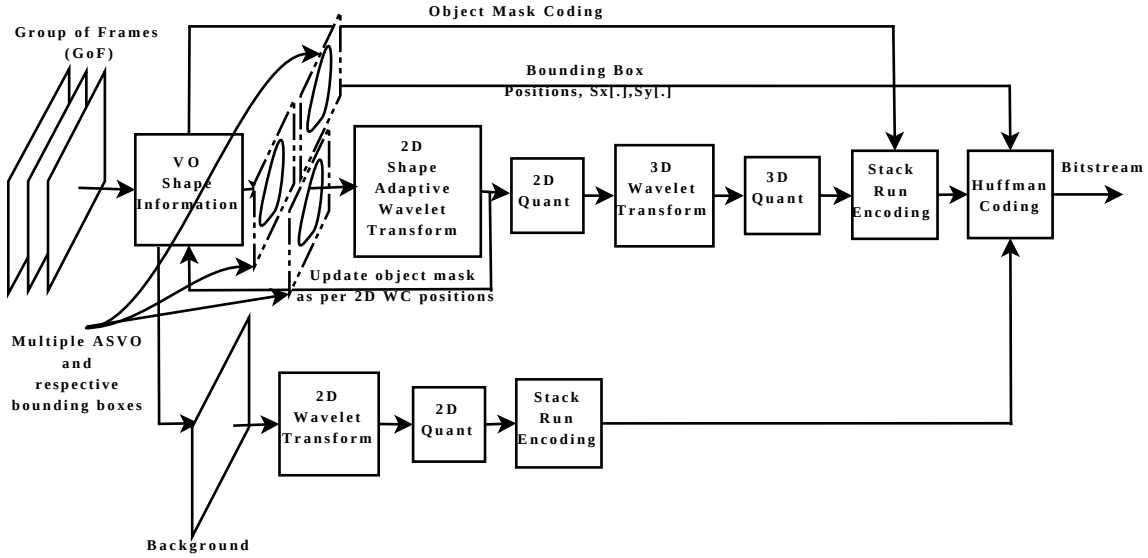


Fig. 5. MASVO Video Compression

objects from one another to increase coding efficiency, and to facilitate shape adaptive wavelet transform, a different bounding box is found corresponding to each cluster of 1s in the object mask. It should be noted that the bounding box boundaries separate one ASVO from the other ASVO, and define subsampling locations for low-pass and high-pass filters; the shape of the virtual object is still arbitrary defined by the object mask values within each bounding box. Figure 5 shows the design flow of MASVO video compression.

Though the MASVO video compression method achieves better coding efficiency in case of a video scene with multiple moving objects, it has additional computational overhead to determine the multiple bounding boxes. Moreover, along with the object mask of the virtual object, starting and ending positions of the bounding boxes are coded and stored so that the decoder side can place the decoded pixels at their correct positions. The coding of the bounding box positions increase the compressed file size and decrease the compression ratio. On the other hand, coding different objects separately results in an increase in coding efficiency, and the compression ratio is increased. Depending upon the scene type, the compression ratio either increases or decreases through the use of MASVO compression, though in general, the compression ratio increases with a video sequence with multiple moving objects.

D. Group of Frames

ASVO extraction works by computing the object mask corresponding to a GoF. The size of the GoF greatly impacts the compression performance, and this is what is discussed in this subsection.

As stated before, a single background frame needs to be sent for every GoF. Therefore, it seems obvious to choose a large GoF size to achieve best possible compression ratio. On the other hand, the shape of the ASVO is given by a 2D mask which is computed simply by applying logical OR operation to the 3D mask $BM_{avo}[x, y, z]$ across the z-direction over the GoF and is given by:

$$OM_{avo}[x, y] = \begin{cases} 1, & \sum_z BM_{asvo}[x, y, z] > 0 \\ 0, & \text{else.} \end{cases} \tag{9}$$

Therefore, as the number of frames in a GoF increases, the probability of obtaining a large virtual object increases as well. In general for a given video sequence, increasing the number of frames in the GoF results in a decrease in the number of background frames needed to be sent thereby increasing the compression ratio. At the same time, increasing the number of frames in the GoF results in an increase in the virtual object size thereby decreasing the compression ratio. Therefore, there is a tradeoff between the compression ratio and the size of the group of the frame. Thus, the GoF size needs to be carefully selected to achieve the best possible compression ratio for a given video sequence.

It should also be noted that the optimal GoF size depends upon the video sequence too. If the video sequence is quite stable with not many moving objects, large number of frames can be grouped together with not much increase in the size of ASVO. In such a case, compression ratio will be increased by increasing the size of GoF as just one background frame needs to be sent corresponding to a large number of frames with very small increase in the virtual object size.

On the other hand, only a small number of frames should be grouped together if the video is having a large number of moving objects in a stationary background. This is because, grouping large number of frames together in such a case will considerably increase the virtual object size reducing the compression gain achieved by ASVO extraction. For example, in a traffic surveillance video, the object could be in one portion of the frame, and at the very next moment

it could be at some other portion of the frame. With multiple such objects and a large GoF size, object mask will be 1 at most of the location resulting in a large virtual object and decreased compression ratio. In the worst case, a large GoF size and multiple moving objects in the video can degenerate the ASVO compression to 3D wavelet compression. Therefore, GoF size should not be a very large value.

Furthermore, memory constraints need to be considered when choosing a maximum GoF. A larger GoF needs more memory to store all the frames and therefore, one can not increase the GoF size endlessly even if the video is not changing significantly. Therefore, a practical upper limit defined by memory constraints is set upon the GoF size. Actual GoF size is found by simple pixel to pixel difference across frames. Starting from the first frame, difference is taken across consecutive frames. As long as the sum of differences across frame is less than a pre-determined threshold value and the total number of frames in the current GoF is less than the upper limit defined by memory constraint, the new frame is not much different than the previous frame, and is added to the GoF containing the previous frame.

Once the GoF is changed, object mask is computed again, previous background frame is discarded, and ASVO compression mechanism is repeated for the new GoF.

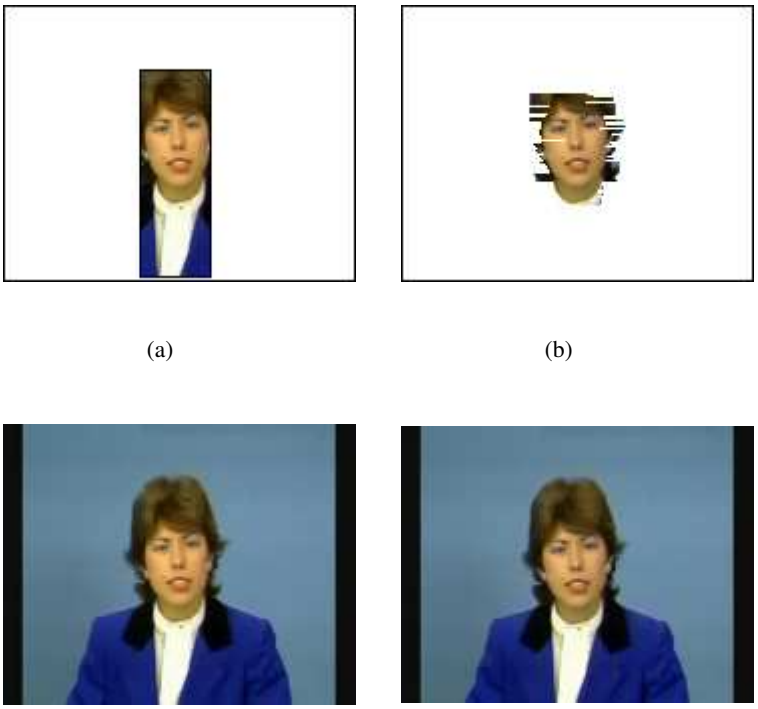
IV. Experimental Results

The ASVO compression method is compared to the 3D wavelet compression and to RVO compression. Standard image sequences such as “CLAIRE”, “AKIYO” and “10TV” are used for comparisons. Results are provided for both the SASVO compression as well as for MASVO compression. Compression ratio has been used as a measure of performance for a given PSNR.

A. RVO and ASVO Compression

SASVO compression is compared to RVO compression [4]. “CLAIRE” sequence is used for comparison. Figure 6 gives the compression results using “CLAIRE” sequence. The maximum GoF size is set to 16 frames.

As shown in Figure 6, SASVO compression gives better compression performance compared to RVO compression. Therefore, the newly proposed method outperforms RVO compression. And since RVO compression outperforms 3D wavelet compression, the method is also better than 3D wavelet compression as also shown in Figure 7.



(a) (b)

(c) (d)

Fig. 6. Comparison between RVO compression and SASVO compression using "CLAIRE" sequence (frame #22 is shown). (a) Extracted RVO (the colored portion only) in RVO compression, and (b) extracted SASVO (colored portion in the frame). (c) RVO compression. Compression Ratio = 53:1, Average PSNR = 26.16, and (d) SASVO Compression. Compression Ratio = 60.02:1, Average PSNR = 37.12.

Additional comparisons are made using "10TV" and "AKIYO" sequences. SASVO compression as well as MASVO compression outperforms RVO compression consistently as shown in Figure 8. In addition, SASVO compression gives similar results to the MASVO compression. This is understandable since all three video sequences have only one moving object. However, the performance of MASVO compression is slightly lower than that of SASVO compression. This is due to the fact that there is only one object in each of the scenes, so there is no benefit to MASVO, and the additional overhead associated with multiple objects results in slightly lower PSNR.

Table I summarizes the results of comparison ratio (CR) and PSNR comparison between 3D wavelet, RVO, SASVO, and MASVO compression methods.

(a)

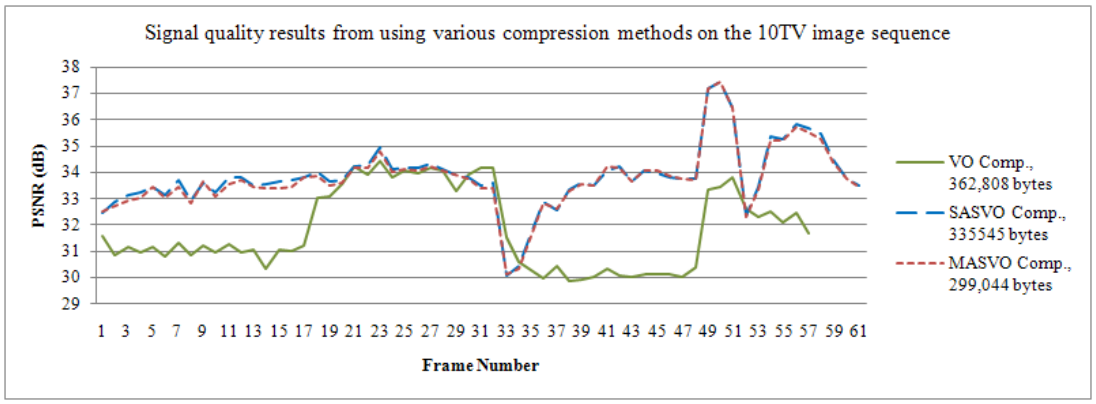
(b) (c)

Fig. 7. Comparison between 3-D compression and SASVO Compression using "10TV" sequence (frame #22 is shown). (a) 3D wavelet compression. Compression ratio = 23:1, Average PSNR = 31.53, (b) extracted SASVO, the whole colored region is considered to be belonging to the same SASVO and is bounded by single rectangular box of dimensions same as the frame size (background intentionally shown white for clarity), and (c) SASVO Compression. Compression Ratio = 42.22:1, Average PSNR = 33.83.

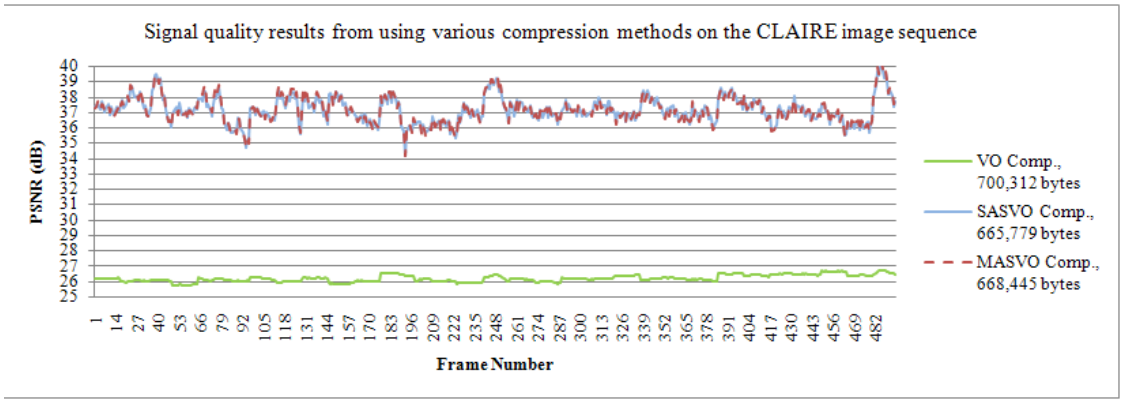
B. Video Sequence with Multiple Moving Objects

We have seen that ASVO compression outperforms RVO compression and 3D wavelet compression consistently for image sequences having just one moving object. Moreover, if we apply MASVO compression to such video sequences, results are similar to SASVO compression as expected because there is only one moving object in the video scene. However, if there are multiple moving objects in the sequence, then MASVO compression performs better than the SASVO compression. Therefore, it is interesting to test the behavior of these compression methods for video sequences involving multiple moving objects. One such sequence involving multiple automobiles is shown in Figure 9.

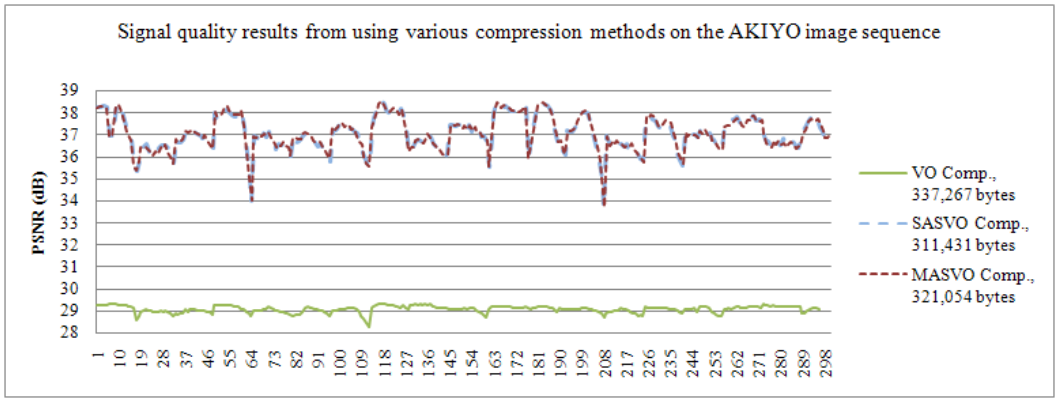
RVO, SASVO as well as MASVO compression methods are applied to this image sequence



(a)



(b)



(c)

Fig. 8. PSNR comparison between RVO, SASVO, and MASVO compression methods

Images Sequence	3D Wavelet Compression		RVO Compression		SASVO Compression		MASVO Compression	
	CR	PSNR	CR	PSNR	CR	PSNR	CR	PSNR
10TV	23.00:1	31.53	23.00:1	31.87	42.22:1	33.83	47.00:1	33.76
AKIYO	46.00:1	29.44	67.00:1	29.10	73.24:1	37.06	72.05:1	37.08
CLAIRE	43.00:1	26.23	53.00:1	26.16	60.02:1	37.12	58.19:1	37.16

TABLE I

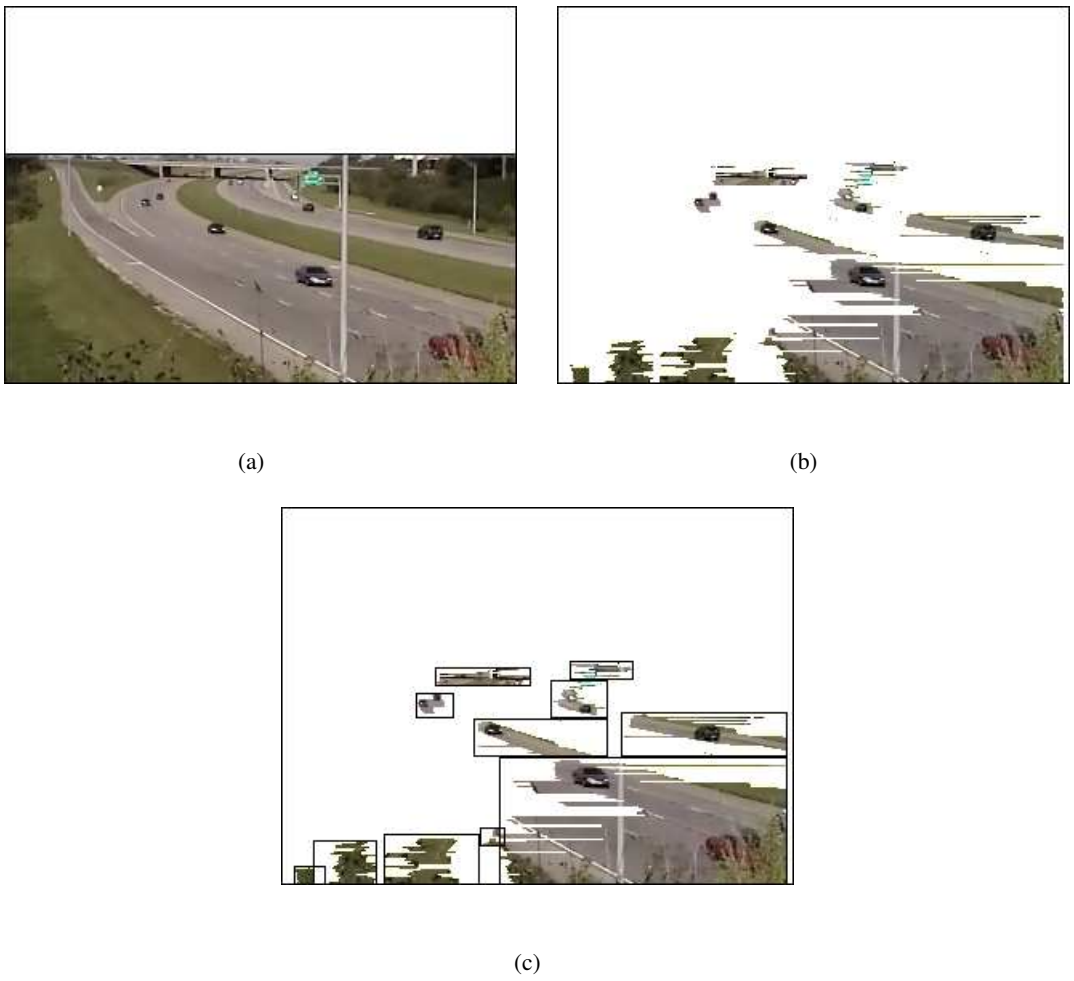
SUMMARY OF COMPRESSION RESULTS

(a)

Fig. 9. Image sequence with multiple moving objects

separately. Figure 10 shows the virtual-objects extracted using various compression methods. Virtual-object portion is shown in real image colors whereas stationary background portion is shown in white for the purpose of clarity.

Clearly from Figure 10, the area belonging to the virtual-object in the spatial domain is largest for RVO extraction. As a result compression ratio will be less for RVO compression method. Figure 11 shows the compression ratio and the corresponding average PSNR for the various virtual-object based compression methods. MASVO compression gives the best compression results in this case and a higher average PSNR value. Overall, ASVO compression is much



(a) (b)

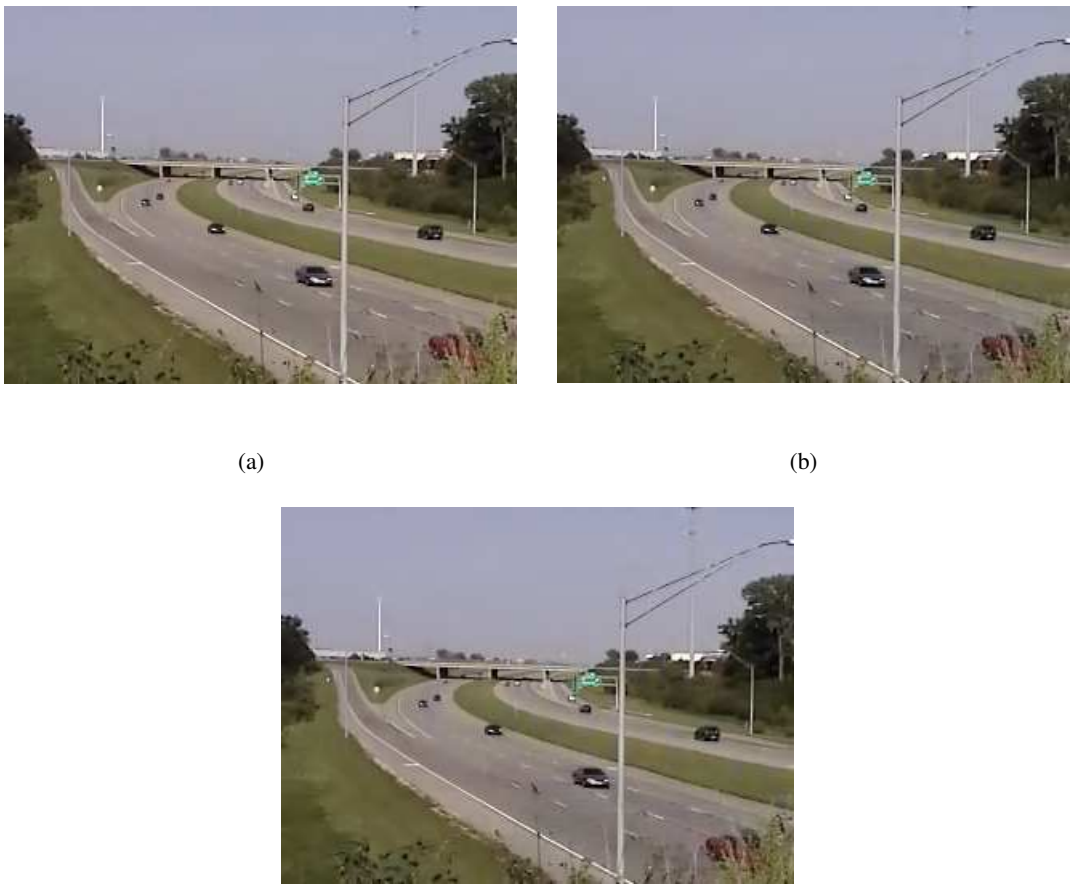
(c)

Fig. 10. Virtual-object extraction. (a) RVO, (b) SASVO, (c) MASVO, and their bounding boxes

superior to the RVO based compression.

V. Conclusions

In this paper, a shape adaptive wavelet transform based ASVO video compression method is presented. The method further explores wavelet transform based video compression and provides some of the benefits of the object based compression at a low computational cost. The newly presented method works by separating ASVO from the stationary background. The ASVO is compressed using 3D wavelet compression with only difference from the 3D wavelet compression being that the spatial decomposition is done using shape adaptive wavelet transform. The background is compressed using 2D wavelet compression. The method provides better



(a) (b)

(c)

Fig. 11. Performance Comparisons. (a) RVO compression. Compression Ratio = 25:1, Average PSNR = 29.85, (b) SASVO compression. Compression Ratio = 47.91:1, Average PSNR = 31.21, and (c) MASVO compression. Compression Ratio = 49.45:1, Average PSNR = 31.44.

compression performance than the pure 3D wavelet compression as well as the RVO compression for the same PSNR.

Furthermore, a MASVO compression method is presented. In general, encoding each arbitrarily shaped object separately from others results in an increase in coding efficiency and therefore, results in an increase in compression ratio. However, in the MASVO compression method, we also need to encode the positions of the bounding boxes of the different arbitrarily shaped objects which can decrease the compression ratio. Therefore, compression ratio will increase or decrease in comparison to SASVO compression depending upon the image sequence being

compressed. In any case though, ASVO compression outperforms 3D wavelet compression and RVO compression.

REFERENCES

[1] I. 14496-10, *Information technology – Coding of Audio-Visual Objects : MPEG-4 Systems*, Oct 1998.

[2] O. Avaro, A. Eleftheriadis, C. Herpel, G. Rajan, and L. Ward, *MPEG-4 Systems: Overview*, Jun 2000.

[3] G. Xing, J. Li, S. Li, and Y. Q. Zhang, "Arbitrarily shaped video-object coding by wavelet," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 10, pp. 1135–1139, Oct 2001.

[4] E. J. Balster and Y. F. Zheng, "Virtual-object video compression," in *Proc. 48th Midwest Symposium on Circuits and Systems*, Aug 2005, pp. 1700–1704.

[5] T. M. Strat, "Object-based encoding: Next-generation video compression," in *Proc. Workshop and Exhibition. MPEG-4*, Jun 2001, pp. 53–57.

[6] I. 11172-2, *Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s - Part 2: Video*, Mar 1993.

[7] I. 13818-2, *Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Video*, Mar 1995.

[8] Z. Shen, M. R. Frater, and J. F. Arnold, "Quad-tree block-based binary shape coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 18, no. 6, pp. 845–850, Jun 2008.

[9] L. Zhou and S. Zahir, "A novel shape coding scheme for MPEG-4 visual standard," in *Proc. Intl. Conf. Innovative Computing, Information and Control*, vol. 3, Sep 2006, pp. 585–588.

[10] Y. T. Hwang, Y. C. Wang, and S. S. Wang, "An efficient shape coding scheme and its codec design," 2001, pp. 225–232.

[11] A. Abu-Hajar and R. Sankar, "Integer-to-integer shape adaptive wavelet transform for region of interest image coding," in *Proc. IEEE 10th Digital Signal Processing Workshop, and the 2nd Signal Processing Education Workshop*, Oct 2002, pp. 94–97.

[12] S. Li and W. Li, "Shape-adaptive discrete wavelet transforms for arbitrarily shaped visual object coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, no. 5, pp. 725–743, Aug 2000.

[13] T. S. Bindulal and M. R. Kaima, "Object coding using a shape adaptive wavelet transform with scalable WDR method," in *Proc. IEEE Intl. Conf. Image Processing*, vol. 2, Oct 2007, pp. 325–328.

[14] Y. Liu, K. N. Ngan, and F. Wu, "3-D shape-adaptive directional wavelet transform for object-based scalable video coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 18, no. 7, pp. 888–899, Jul 2008.

[15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. 511–518.

[16] C. He, Y. F. Zheng, and S. C. Ahalt, "Object tracking using the gabor wavelet transform and the golden section algorithm," *IEEE Trans. Multimedia*, vol. 4, no. 4, pp. 528–538, Dec 2002.

[17] Z. Xiong, K. Ramchandran, M. T. Orchard, and Y. Zhang, "A comparative study of DCT and wavelet based coding," in *Proc. IEEE Intl. Symp. Circuits and Systems*, vol. 4, Jun 1998, pp. 273–276.

[18] H. Jozawa, H. Watanabe, and S. Singhal, "Interframe video coding using overlapped motion compensation and perfect reconstruction filter banks," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, vol. 4, Mar 1992, pp. 649–652.

[19] W. Sweldens, "The lifting scheme: A custom design construction of biorthogonal wavelets," *Appl. Comput.Harmon. Anal*, vol. 3, no. 2, pp. 186–200, Apr 1996.

[20] C. Lin, B. Zhang, and Y. F. Zheng, "Packed integer wavelet transform constructed by lifting scheme," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, no. 8, pp. 1496–1501, Dec 2000.

[21] H. Guo and C. Burrus, "Wavelet transform based fast approximate fourier transform," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, Apr 1997, pp. 1973–1976.

[22] J. Y. Tham, S. Ranganath, and A. A. Kassim, "Highly scalable wavelet-based video codec for very low bit-rate environment," *IEEE Trans. Selected Areas in Communications*, vol. 16, no. 1, pp. 12–27, Jan 1998.

[23] W. Tan, E. Chan, and A. Zalchor, "Real time software implementation of scalable video codec," in *Proc. Intl. Conf. Image Processing*, vol. 1, Sep 1996, pp. 17–20.

[24] A. Polzer, H. Klock, and J. M. Buhrmann, "Video coding by region-based motion compensation and spatio-temporal wavelet transform," in *Proc. Intl. Conf. Image Processing*, vol. 3, Oct 1997, pp. 436–439.

[25] S. A. Martucci, I. Sodagar, T. Chiang, and Y. Zhang, "A zerotree wavelet video coder," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 109–118, Feb 1997.

[26] R. Adhami, "Video compression technique using wavelet transform," in *Proc. IEEE Conf. Aerospace Applications*, 1996.

[27] U. Y. Oktiawati and V. V. Yap, "Video compression using dual tree complex wavelet transform," in *Proc. Intl. Conf. Intelligent and Advanced Systems*, Nov 2007, pp. 775–778.

[28] C. He, J. Dong, Y. F. Zheng, and Z. Gao, "Optimal 3-D coefficient tree structure for 3-D wavelet video coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 10, pp. 961–972, Oct 2003.

[29] G. Karlsson and M. Vetterli, "Three dimensional sub-band coding of video," in *Proc. Intl. Conf. Acoustics, Speech, and Signal Processing*, Apr 1988, pp. 1100–1103.

[30] J. Li and S. Lei, "Arbitrary shape wavelet transform with phase alignment," in *Proc. IEEE Intl. Conf. Image Processing*, vol. 3, Oct 1998, pp. 683–687.

[31] T. Sikora and B. Makai, "Shape-adaptive DCT for generic coding of video," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 5, no. 1, pp. 59–62, Feb 1995.

[32] E. J. Balster, Y. F. Zheng, and R. L. Ewing, "Combined spatial and temporal domain wavelet shrinkage algorithm for video denoising," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 220–230, Feb 2006.

[33] ——, "Feature-based wavelet shrinkage algorithm for image denoising," *IEEE Trans. Image Processing*, vol. 14, no. 12, pp. 2024–2039, Dec 2005.

[34] E. J. Balster, "Video compression and rate-constrained control methods based on the wavelet transform," Ph.D. dissertation, The Ohio State University, 2004.

[35] M. J. Tsai, J. D. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 5, pp. 519–521, Oct 1996.