Object based Hybrid Video Compression

Rhoda Gbadeyan and Chris Joslin
Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada

Keywords: Video Compression, Object-based Coding, Background Subtraction, Object Detection.

Abstract: Standard video compression techniques have provided pixel-based solutions that have achieved high compression performance. However, with new application areas such as streaming, ultra-high definition TV(UHDTV) etc., expectations of end user applications are at an all-time high. Never the less, the issue of stringent memory and bandwidth optimization remains. Therefore, there is a need to further optimize the performance of standard video codecs to provide more flexibility to content providers on how to encode video. In this paper, we propose replacing pixels with objects as the unit of compression while still harnessing the advantages of standard video codecs thereby reducing the bits required to represent a video scene while still achieving suitable visual quality in compressed videos. Test results indicate that the proposed algorithm provides a viable hybrid video coding solution for applications where pixel level precision is not required.

1 INTRODUCTION

With more sophisticated end user devices constantly being introduced onto the market at rapid pace, there is an increasing demand for higher quality video e.g. UHDTV, high quality streaming services etc. The challenge for meeting these requirements necessitate creating video content at higher spatial resolutions and frame rates resulting in the need for a higher bitrate to transmit, distribute and/or store this content. Overtime, research has focused on the development of both standard and non-standard video compression techniques to handle this content. Of these two areas of research, standard video compression solutions have been by far the more successful solution.

For the purposes of video processing, to the human observer, it is safe to assume that there are only minimal identifiable changes as we navigate within a video scene from one frame to the next. However, when observed at the pixel level by a standard video compression algorithm, there can be significant differences when one frame is compared to the next sequence of frames. These pixel variations could be due to object motion, lighting variations, shadowing or occlusion etc. and tracking all of these changes at the pixel level results in a lot of data, which even after compression, take up a lot of valuable memory space as well as transmission bandwidth.

By handling scene variations at the pixel or block level and storing this information, even in its encoded form, the limited memory available is quickly exhausted making compression mandatory. Video compression standards have demonstrated increasingly high performance starting from the H.261 and MPEG-1 standards to the HEVC standard. However, their existing uniform and lengthy schema is fast becoming unable to meet all the changing needs and dynamic nature of the video coding community (Mattavelli, Jorn, & Mickaël, 2019) and there is a need to find other ways to improve compression by adding new functionalities or including content adaptation methods which may boost their coding efficiency. In this paper, we propose a hybrid video compression solution that replaces pixels with objects as the unit of compression while still harnessing the advantages of standard video codecs.

This paper is presented as follows: in section 2, a literature review of existing work in this area of research is provided. Section 3 outlines the details of the proposed algorithm. Results are presented in section 4 and the paper concludes in section 5.

2 LITERATURE REVIEW

Video coding solutions can be categorized into 3 groups on the basis of their approach: standard video coding, non-standard video coding and a hybrid of both standard and non-standard video coding. Both

Gbadeyan, R. and Joslin, C.
Object based Hybrid Video Compression.
DOI: 10.5220/0010207607850792
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 785-792
ISBN: 978-989-758-488-6
Copyright © 2021 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

standard and non-standard video compression solutions have been actively investigated for years resulting in the implementation of several techniques. Standard video compression solutions have been the more successful of the two solutions by far.

While standard codecs have enjoyed large scale success, there have also been a few drawbacks, some of which are that:

- While block-based solutions offer great flexibility, they also suffer the draw back of blocking artifacts in some implementations
- Existing block-based coding standards, such as H.264 and H.265, do not consider the arbitrary shape of moving objects and, as a result, their prediction efficiency is compromised

While standard codecs remain the gold performance criterion for video compression technologies, there is still room for further improvement as well as a need for alternative solutions as these codecs are not always suitable for every application. There is also a need to explore non-pixel/block-based alternatives either as a standalone solution or in combination with existing standard codecs as they may offer a more efficient way to utilize the stringent memory available. Several video compression solutions have been proposed which either adapt existing standard solutions or propose new ways to solve the video compression problem, these are reviewed next.

Mosaics are created by stitching together images from a given scene taken at different instances in time, thereby creating a panoramic view of that scene. Mosaics often contain spatially overlapping information about the scene. Authors in (Hsu & Anandan, 1996) & (Irani & Anandan, 1998) proposed a hybrid mosaic-based video compression algorithm that combines mosaics with standard video compression. This solution replaces pixels with mosaics as the basic unit for compression. By using mosaics, the authors were able to address untapped redundancies still present in video scenes.

Pattern based coding (PBC) for video was standardized by the introduction of predefined pattern code books in the H.263 and H.264 standards (Sullivan & Wiegand, 2005). The problem with predefined codebooks is that using them does not achieve optimal coding efficiency (Manoranjan, Murshed, & Dooley, 2003). Authors in (Manoranjan, Murshed, & Dooley, 2003) & (Murshed & Manoranjan, 2004) introduced the use of dynamically extracted patterns from video content thereby achieving superior coding efficiencies. Their hybrid video coding solution combined the use of customized pattern code books with existing standard video codecs.

Authors in (Galpin, Balter, Morin & Pateux, 2004) & (Balter, Gioia & Morin, 2006) proposed a non-standard model-based approach to video compression. Without utilizing any standard codecs, the authors developed a scalable compression method for scenes with unknown content. They harnessed the redundancy in the 3D models by only intra coding the key frames while all others were compressed in a predicted mode. Their approach avoided having to transmit texture coordinates by computing the texture information at the decoder side.

Over the last few years, deep learning-based video coding has been an actively developing area of research (Dong, Jianping, Li & Wu, 2020). This emerging area of research has yielded techniques that can be considered to be hybrid deep learning-based video coding methods as well as non-standard deep learning-based coding methods. Non-standard deep learning can be divided into either pixel probability (PP) modeling-based techniques or auto-encoder (A-E) based techniques. PP modeling uses deep learning to solve prediction problems and is representative of predictive coding. Some representative work includes (Wu, Singhai, Krahenbuhl, 2018) & (Chen, He, Jin & Wu, 2019). Conversely, A-E based techniques work by training a deep learning network to encode by converting high dimensional signals to low dimensional ones and decode by recovering the high dimensional signals on the basis of the low dimensional signals generated by the encoder. A-Es are representative of transform coding. Some representative work includes (Chen, Liu, Shen, Yue, Cao, & Ma, 2017) & (Lu, Ouyang, Xu, Zhang, Cai, & Gao, 2019).

Hybrid deep learning-based video coding methods work by utilizing trained deep networks as tools within standard coding schemes or in conjunction with standard coding tools such as HEVC. As demonstrated in (Dong, Jianping, Li & Wu, 2020), deep trained networks can replace almost all the modules in standard video codecs. Authors in (Li, Li, Xu, Xiong, & Gao, 2018) & (Hu, Yang, Li, & Liu, 2019) have proposed replacing the traditional intra-coding module with a deep trained network version while authors in (Lin, Liu, Li, & Wu, 2018) & (Zhao, Wang, Zhang, Wang, Ma, & Gao, 2019) utilized unique inter-coding modules.

Today, while deep learning-based video coding continues to show promising results in this field, it is important to note that it is an area of research that is still in its infancy with a lot of room to develop into a mature area of research (Dong, Jianping, Li & Wu, 2020). Next, we review the proposed technique.

3 OVERVIEW OF PROPOSED OBJECT SEGMENTATION AND DETECTION METHOD

When there are changes in a video scene, such changes are often associated with specific objects, affecting either the whole object or only a subsection of the pixels that compose the object. Therefore, rather than pixels, objects form the foundation for each phase of this proposed algorithm.

We propose a background subtraction (BGS) object-based hybrid video compression scheme. The algorithm is made up of three key phases:

1. BGS Phase
2. Object Detection and Tracking Phase
3. Compression and Scene Reconstruction Phase

In each phase, an object-based algorithm was implemented. To achieve a high level of consistency, regardless of the phase, a consistent color space was used. The Hue Saturation Value (HSV) color space was chosen as it is closely aligned with the human visual system. An overview of the proposed hybrid object-based algorithm is provided in Figure 1. Next, each phase of the algorithm is reviewed.

3.1 BGS Module

An adaption of the Statistical and Knowledge Based Object Detection (SAKBOT) algorithm which was proposed by authors in (Cucchiara, Grana, Piccardi & Prati, 2003) as well as the improved SAKBOT approach proposed by authors in (Calderara, Melli, Prati & Cucchiara, 2006) was implemented. The SAKBOT BGS algorithm was chosen as the primary BGS method because it combines the advantages of statistical methods with those of adaptive methods to generate a resilient BGS algorithm. While this BGS approach met the basic requirements for this implementation, there were a few issues that had to be remediated to improve the performance of the overall object-based compression algorithm. These include its inability to (1) provide a measure of variance, (2) take advantage of the spatial information inherent in a video scene and (3) apply post processing to the achieved results. When combined, these factors resulted in a higher incident of false positives (Prati, Mikic, Grana & Trivedi, 2001).

Our implementation of the SAKBOT algorithm is referred to as the adapted SAKBOT algorithm. It added an additional data point to the computation which was the spatial location of the moving visual objects (MVO), their shadows and any ‘ghosts’ (i.e. the set of connected points identified as being in motion by the BGS process but which do not belong to any real moving objects) in the video scene. This change contributed to the reduction of the instances of false positives. This adapted SAKBOT has 3 key components:

1. Background Initialization aka Bootstrapping
2. Foreground Detection
3. Selective Background Maintenance

The background was initialized by building the background model B^t from the video sample. The initialization buffer size was set to 120 frames. This was optimized by initially using a buffer size of ~ 50 frames and then subsequently updating the model as required throughout the computation. The advantage of this modification to the SAKBOT algorithm was a reduction in the memory required upfront to create the BGS model. The generated background model was designated as the background subtractor object and served as the input to the next phase.

The foreground was computed on the frame level as the difference between the current frame I^t and the background model B^t thereby generating the foreground mask $M_t(i,j)$ containing the grey-level information of the foreground. The selective model updates were achieved by applying a temporal median to a circular buffer that stored pixel values over time. This approach benefits from the ability to capture statistical dependencies between color channels. The input video was converted from its native RGB color space to HSV to support the implementation of the shadow detection module.

Next, a binarized motion mask was computed and MVO objects at time t were extracted from this final binarized motion mask. This generated the first list of candidate objects and shadows to be tracked. This list was subsequently vetted in the object detection module enabling the generation of the final list of objects which was later fed as input to the object tracking module.

3.2 Object Detection Module

As no one feature can provide invariance to all scene changes, a multi-feature-based approach was used to achieve reliable object identifications. The local features selected in this phase were object color and object motion. These features are considered to be mutually independent as one feature cannot be predicted on the basis of another feature (Khan & Shah, 2001). The HSV color space was used because it is able to explicitly separate luminosity and chromaticity. Within the HSV color space, the hue of a pixel was assigned a larger weight than its saturation and brightness and was used for clustering pixels into regions for the purpose of segmentation and object detection. This involved extracting this feature as well as generating an HSV Histogram of Oriented Gradient (HOG) based on the hue channel which was then used to detect and segment objects in the foreground.

The object detection problem was considered to be a form of the classification problem in which classifications were considered to be good if they were homogeneous with respect to a specific feature within the region of interest (ROI) and dissimilar outside the focus area. Therefore, since the hue of a particular color is considered to be at its largest value in the center of an object where it is least sensitive to colors from adjacent regions, the center hill of the histogram was identified as the center of the cluster of the color which was then determined to be the center of the object to be segmented. As the HSV color space has the added advantage of being able to identify color shade and intensity value variations in the area of the object's edge, thereby sharpening the object boundaries, it was possible to accurately identify the object of interest from its identified center to its boundary thereby increasing the accuracy of the object segmentation.

To apply object motion as a feature, thereby validating the information provided by the color feature, there was a need to compute the motion parameters. By assuming that the color segment of the detected object was a superset of the motion segment, this computation focused only on generating the motion parameters for pixels identified as belonging to a candidate object/ROI via the dense Farneback optical flow-based approach thereby simplifying the otherwise cumbersome computation process.

Assuming that the motion field within each candidate object's ROI was smooth and that the optical flow constraint assumption that the intensity of a pixel in an image is constant along the trajectory of the pixel's motion holds true, then, motion estimation was computed by generating a parametric motion model for each candidate object in the scene. The motion segmentation approach used key representations highlighted in (Chang, Tekalp & Sezan, 1997) because by defining the segmentation based on a parametric motion model, physically meaningful ROIs could be achieved.

As such, the motion field was jointly represented as both the sum of a parametric field 'p' as well as a residual field 'r'. From the current frame g_k to the search frame g_{k-1}, a 2-D motion vector $d(m,n)$ for pixel location (m,n) was defined as

$$d(m,n) = [u(m,n), v(m,n)] \tag{1}$$

Where u and v are the vectors for each pixel location associated with candidate objects in each frame. Then, k, the set of independently moving objects was set equal to the list of candidate objects generated in the previous phase. Provided that a segmentation label $x(m,n)$ is used to assign motion vector $\boldsymbol{d}(m,n)$ to a pixel belonging to one of the objects in the set k, the motion of each object was approximated by a 6 to 8 parameter affine parametric mapping of ϕ, the result of which produced the parametric motion vector component $\boldsymbol{d}_p$ as well as the residual motion vector $\boldsymbol{d}_r$ at pixel (m,n) as expressed below

$$\boldsymbol{d}(m,n) = \boldsymbol{d}_p(m,n) + \boldsymbol{d}_r(m,n) \tag{2}$$

Where $\boldsymbol{d}_p(m,n)$ is dependent on the segmentation label $x(m,n)$, thereby computing both the parametric motion vectors as well as the residual motion vectors.

The Bayesian rule was applied to the a-posteriori Probability Density Function (PDF) of *u, v,* and *x,* given g_k and g_{k-1} to obtain the Maximum A-Posteriori (MAP) estimates. Therefore,

- Given the best estimates of the motion and segmentation fields, the mapping parameters ϕ where obtained by computing a least squares procedure
- Given the best estimate of the parametric field, constrained by the assumption that the motion field is smooth within each segment, the motion field was updated through an estimation of the minimum-norm residual field and
- Given the best estimate of the motion field via Gibbsian priors, the segmentation field was updated to yield the minimum-norm residual field

A conditional pdf was then used to quantify how well the estimates fit the given candidate objects. After computation, the following was obtained: a parametric model per object, residual motion vectors for each 4 by 4 block of pixels belonging to the object, a dense motion field, a motion segmentation field and a set of mapping parameters. The resulting parametric motion model as well as the motion segmentation field were overlaid on to the foreground, if there was a match between the identified motion areas as well as the candidate object, this object was identified as a real detected object else it was discarded as a false positive. Every detected object was then segmented based on the outlined edges identified. The segmented object was added to the object list and a candidate blob was added to the blob list. A blob was generated on the basis of its connectivity and stored in the blob list. Once this task was completed, the color space was converted back from the HSV color space to the RGB color space for the next phase of processing.

The tracking module received as input the list of detected objects $Alist$ in addition to the list of candidate blobs $Blist$ which were considered to be associated with the tracking objects. A list, $Clist$ was created to track the number of consecutive frames that each object in the detected object list had not appeared in the scene. If an object was identified as a tracking object at some point in the scene but had disappeared from the scene and over the next 50 consecutive frames, the object had not reappeared in the scene, then that object was assumed to have exited the scene and such an object was then removed from $Alist$ and only stored in $Clist$. However, if the i-th object in $Alist$ is recognized as a tracking object, a new blob was added to $Blist$ and the loop was repeated. We assigned a candidate blob in $Blist$ to each object in $Alist$ by computing a distance matrix from $Alist$ to $Blist$ on the basis of 3 measures - position distance, blob distance and color variance measure. The position distance, i.e. the distance from the center of the current tracking object to the center of the previous tracked object was computed. On the basis of this known value, the distance between the position of the current tracking object and its next position was then estimated using the Unscented Kalman Filter (UKF). A similar distance metric was computed between the blobs as well to effectively match blobs to each object's position in the scene. The color variance measure was computed as the square of the distance between the histogram of Hue of the HSV color space of the previous tracking object and that of the current tracking object. Finally, after computing the position, blob and color measures, the overall distance D_{ij} was calculated as

$$D_{i,j} = Color\ Measure * sqrt\ (Position\ Distance) * Blob\ Distance \qquad (3)$$

The value of D_{ij} for each object and blob position x, y was stored in the D array D = $\begin{bmatrix} & D_{ij} & \end{bmatrix}$ (4)

where, D_{ij} is the distance from A_i to B_j

In cases where a tracked object A_i was never recognized as a blob B_j , the distances were individually set to infinity and D_{ij} was assigned to infinity. To solve this problem of assigning a tracked object to a blob, the Hungarian algorithm (Kuhn, 1955) was used. Based on the distance matrix D, this algorithm was used to achieve the optimal assignment thereby associating blobs in $Blist$ with tracking objects in $Alist$. The achieved results were updated based on $Clist$ which contained skipped objects. There were candidate blobs in $Blist$ which had not yet been matched to any tracking object as the blobs were not recognized. The blobs in this category were then matched to the objects in $Clist$ and based on their distance and statistics information, the object list was updated with the newly matched objects.

The tracking algorithm developed in this research is able to handle single object tracking as well as multi-object tracking. In the single object tracking case, it was identified that due to possible computational errors in earlier phases of the algorithm, a single object may have incorrectly been identified as two tracking objects. To address this, a distance threshold value 'd' was defined and set equal to 0.5 such that if the distance between any two trackers was found to be less than d, the objects previously identified to be two distinct tracking objects were considered to be just one object, and the objects were merged into a single tracking object. On the other hand, the multi-tracking algorithm was triggered if the computed distance between any two objects was found to be greater than d.

To enable reconstruction, the spatial location of each object relative to other objects in a frame was computed for each instance of that object across all frames. To generate stable tracks of objects across the scene, each newly identified object was compared to the list of existing validated objects, if a new object was matched to an existing object, it was tagged with the same tracking ID as the existing object, else it was tagged with a new unique tracking ID. Finally, all stable object tracks were written into object files.

3.3 Compression and Object Layer Reconstruction Module

The main goal of this module of the proposed algorithm was to iteratively recombine the objects in a frame, sequentially compress each recombined frame, feed the recombined frame as input into the prediction loop of a video codec and ultimately, reconstruct the original input video. As the proposed algorithm is an object-based hybrid compression algorithm, an H.265 codec was utilized for the compression phase. This reconstruction module received as input a color image of the background, scene objects and their stable tracks as well as the tracking text file containing the statistics of each frame in addition to the objects in the frame. Individual object bitstream files were created for each object's stable track across the scene on the basis of this data and objects were properly placed in the resulting bitstream relative to their location details.

Once the object files were created, the next step was to encode and reconstruct the video sample by recombining the object files after which the video sample was decoded. The algorithm worked as follows:

```
Step 1 – Initialize the video sample
reconstruction phase by reading the
colored background image.

Step 2 – Pass the background as input to
the standard codec, encode this file as
a key frame and store the encoded file
in the buffer. The number of bitstream
files in the buffer is set = 1,

    set Current bitstream = Recombined
    Bitstream; set bitstream# = 1

    If bitstream# < Total # of
    bitstream -1

Step 3 – Pass the previously obtained
Recombined Bitstream as input to the
inter-frame prediction loop.

Step 4 - Read the next object bitstream
from the folder, pass the object
bitstream as input to the standard codec
to be encoded and pass its associated
metadata to the buffer.

Step 5 – If the number of bitstreams in
the buffer is > 1, then

Using the associated metadata
information, combine the New Object
bitstream with the current bitstream to
form the new Recombined Bitstream

    Set Current bitstream = Recombined
    Bitstream; bitstream# ++,

    repeat steps 3-4,

else

Step 6 – Write overall recombined
bitstream to file. This bitstream was
then passed on to the decoder and the
video sample was decoded

end
```

4 RESULTS

This algorithm uses objects as the minimum unit of processing, as such, pixel level changes from one frame to another were not fully captured. Therefore, the Structural Similarity Index Measure (SSIM) proposed by in (Wang, Bovik, Sheikh & Simoncelli, 2004) was used. This metric takes into account the similarity of the edges between the original frame and the reconstructed frame. According to the authors in (Hore & Ziou, 2010), there is a correlation between SSIM and the quality perception of the human visual system (HVS) and is computed by modeling any image distortions resulting from the comparison of the reference image against the test image as a combination of three factors – Contrast distortion, loss of correlation and luminance distortions.

This algorithm has been designed to integrate with any standard video compression algorithm. Therefore, its performance when integrated with HEVC was assessed against the standard HEVC implementation using a few sample videos. The results obtained are promising while identifying areas for improvement. The test results shown in Figure 2 are from the pedestrian traffic video sample which is an outdoor video scene of medium complexity, showing pedestrians in motion, with partial and full occlusion at various points in the scene and the people in the scene moving in unusual patterns. The scene has moderate lighting variations resulting in shadows being associated with the objects in motion. As shown in Table 1, this algorithm achieved comparable SSIM results compared to HEVC while achieving bit reduction over H.265. Testing results have demonstrated that when the complexity of the scene is high, i.e. in scenes where 80% or more of the pixels are changing over the course of the scene, this proposed algorithm did not achieve the level of bit reduction compared to HEVC as it did with less complex scenes. A contributing factor is that the BGS algorithm which forms the foundation of this technique is not as effective at handling the complexity of this type of scenes.

5 CONCLUSIONS

The main contribution of this work of research is a non-pixel-based approach to video coding that harnesses the strength of existing pixel based standard codecs while eliminating the requirement to process a video sample on the basis of pixels, rather offering an opportunity to process such samples on the basis of scene objects. This algorithm is able to efficiently encode all the background areas of each video frame i.e. areas observed for the first time; areas revisited by the capture camera's Field of View (FOV) following a long period of time when that portion of the scene was unavailable as well as areas of the scene that have been uncovered by a moving object. Due to the fact that this algorithm does not track scene changes at the pixel level, it is suitable for use in applications where performance is not measured at the pixel level.

Future work is still required to optimize the performance of this algorithm when processing complex scenes. This algorithm offers an alternative solution suitable for use in bit constrained environments.

REFERENCES

Mattavelli, Marco, Jorn W. Janneck, and Mickaël Raulet, 2019. "MPEG reconfigurable video coding." In *Handbook of signal processing systems*, pp. 213-249. Springer, Cham, 2019.

Hsu, S., & Anandan, P., 1996. Hierarchical representations for mosaic-based video compression. In *Proc. Picture Coding Symp* (Vol. 395).

Irani, M., & Anandan, P., 1998. Video indexing based on mosaic representations. *Proceedings of the IEEE*, *86*(5), 905-921.

ITU-T Rec. H.263 (01 /2005) Video Coding For Low Bit Rate Communications

Sullivan, Gary J., and Thomas Wiegand, 2005. "Video compression-from concepts to the H. 264/AVC standard." *Proceedings of the IEEE* 93, no. 1 (2005): 18-31.

Paul, Manoranjan, Manzur Murshed, and Laurence Dooley, 2003. "An arbitrary shaped pattern selection algorithm for very low bit-rate video coding focusing on moving regions." *Proc. of 4th IEEE Pacific-Rim Int. Con. on Multimedia (PCM-03), Singapore* (2003).

Murshed, M., Manoranjan, P., 2004. Pattern Identification VLC for Pattern-based Video Coding using Co-occurrence Matrix. In: International Conference on Computer Science, Software Engineering, Information Technology, e-Business & Applications, Egypt (2004)

Galpin, Franck, Raphaele Balter, Luce Morin, and Stéphane Pateux, 2004. "Efficient and scalable video compression by automatic 3d model building using computer vision." In *Picture Coding Symposium, PCS'2004, San Francisco, USA*. 2004.

Balter, Raphale, Patrick Gioia, and Luce Morin, 2006. "Scalable and efficient video coding using 3-d modeling." *IEEE Transactions on Multimedia* 8, no. 6 (2006): 1147-1155.

Liu, Dong, Yue Li, Jianping Lin, Houqiang Li, and Feng Wu, 2020. "Deep learning-based video coding: A review and a case study." *ACM Computing Surveys (CSUR)* 53, no. 1 (2020): 1-35.

Claude Elwood Shannon, 1948. A mathematical theory of communication. *Bell System Technical Journal* 27, 3 (1948), 379-423.

Zhibo Chen, Tianyu He, Xin Jin, and Feng Wu., 2019. Learning for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*. DOI:10.1109/TCSVT.2019.2892608

Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl., 2018. Video compression through image interpolation. In *ECCV*. 416-431.

Tong Chen, Haojie Liu, Qiu Shen, Tao Yue, Xun Cao, and Zhan Ma., 2017. DeepCoder: A deep neural network-based video compression. In *VCIP*. IEEE, 1-4.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao., 2019. DVC: An end-to-end deep video compression framework. In *CVPR*. 11006-11015.

Jiahao Li, Bin Li, Jizheng Xu, Ruiqin Xiong, and Wen Gao., 2018. Fully connected network-based intra prediction for image coding. *IEEE Transactions on Image Processing* 27, 7 (2018), 3236-3247.

Yueyu Hu, Wenhan Yang, Mading Li, and Jiaying Liu., 2019. Progressive spatial recurrent neural network for intra prediction. *IEEE Transactions on Multimedia* 21, 12 (2019), 3024 - 3037. DOI:10.1109/TMM.2019.2920603

Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu., 2018. Generative adversarial network-based frame extrapolation for video coding. In *VCIP*. 1-4.

Lei Zhao, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao., 2019. Enhanced motion-compensated video coding with deep virtual reference frame generation. *IEEE Transactions on Image Processing* 28, 10 (2019), 4832-4844.

Cucchiara, R., Grana, C., Piccardi, M., & Prati, A., 2003. Detecting moving objects, ghosts, and shadows in video streams. *IEEE transactions on pattern analysis and machine intelligence*, *25*(10), 1337-1342.

Calderara, S., Melli, R., Prati, A., & Cucchiara, R., 2006. Reliable background suppression for complex scenes. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks* (pp. 211-214). ACM.

Prati, A., Mikic, I., Grana, C., & Trivedi, M. M., 2001. Shadow detection algorithms for traffic flow analysis: a comparative study. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)* (pp. 340-345). IEEE.

Khan, S., & Shah, M., 2001. Object based segmentation of video using color, motion and spatial information. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (Vol. 2, pp. II-II). IEEE.

Chang, M. M., Tekalp, A. M., & Sezan, M. I., 1997. Simultaneous motion estimation and segmentation. *IEEE transactions on image processing*, *6*(9), 1326-1333.

Kuhn, H. W., 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83-97.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P., 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, *13*(4), 600-612.

Hore, A., & Ziou, D., 2010. Image quality metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition* (pp. 2366-2369). IEEE.

Table 1: Performance Metrics for Proposed Object Based Video Compression Technique (Highlighting algorithm's bit reduction compared to H.265).

Sample Name	Frame Width	Frame Height	Bit Rate (Kbps)	Algorithm's SSIM	Bit Reduction	H.265 SSIM
Cars Driving Under a bridge	1280	720	9768	0.96	34%	0.97
Medium Complexity Pedestrian Traffic	768	576	818	0.97	38%	0.98
Complex Pedestrian Traffic	640	480	2033	0.93	17%	0.96

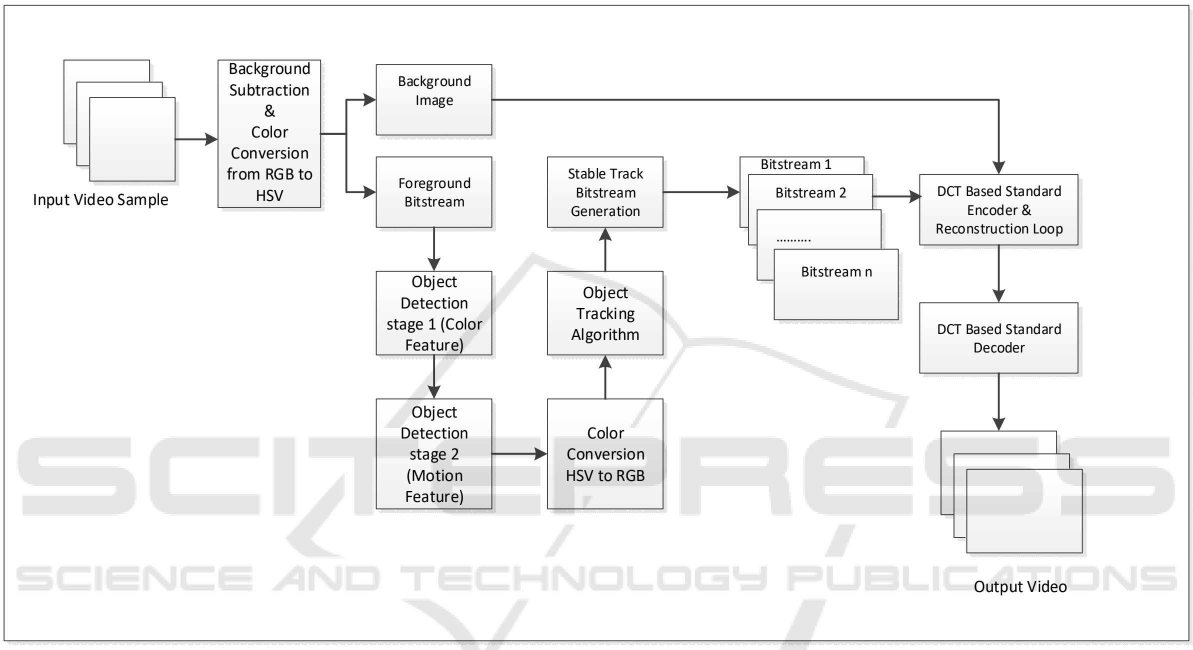

Input Video Sample
Background Subtraction & Color Conversion from RGB to HSV
Background Image
Foreground Bitstream
Object Detection stage 1 (Color Feature)
Object Detection stage 2 (Motion Feature)
Color Conversion HSV to RGB
Object Tracking Algorithm
Stable Track Bitstream Generation
Bitstream 1
Bitstream 2
....
Bitstream n
DCT Based Standard Encoder & Reconstruction Loop
DCT Based Standard Decoder
Output Video

Figure 1: An Overview of the Proposed Hybrid Object-based Algorithm.

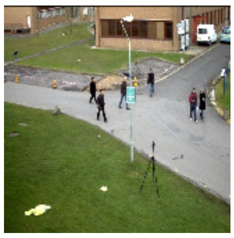
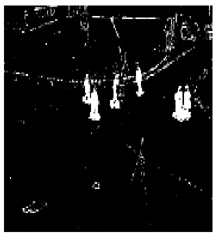
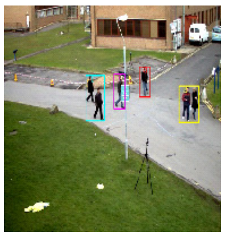
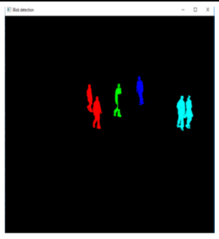
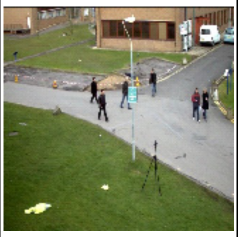
Input Video Frame	Foreground Mask	Tracking Result	Corresponding Blob Result	Final Output
				

Figure 2: Algorithm Results Across all Stages.