ADAPTIVE VIDEO COMPRESSION FOR VIDEO SURVEILLANCE APPLICATIONS

Andrew D. Bagdanov, Marco Bertini, Alberto Del Bimbo, Lorenzo Seidenari

Media Integration and Communication Center
University of Florence
Florence, Italy
Email: bagdanov@gmail.com, bertini|delbimbo|seidenari@dsi.unifi.it

ABSTRACT

This article describes an approach to adaptive video coding for video surveillance applications. Using a combination of low-level features with low computational cost, we show how it is possible to control the quality of video compression so that semantically meaningful elements of the scene are encoded with higher fidelity, while background elements are allocated fewer bits in the transmitted representation. Our approach is based on adaptive smoothing of individual video frames so that image features highly correlated to semantically interesting objects are preserved. Using only low-level image features on individual frames, this adaptive smoothing can be seamlessly inserted into a video coding pipeline as a pre-processing state. Experiments show that our technique is efficient, outperforms standard H.264 encoding at comparable bitrates, and preserves features critical for downstream detection and recognition.

***Keywords*-Video coding; video analysis; video-surveillance; H.264.**

I. INTRODUCTION

Many critical video streaming applications require transmission of many streams over limited bandwidth. Two such examples are video surveillance networks and local UHF video streaming networks like those based on the ETSI TETRA standard used in emergency and security services [1]. These two example applications have several things in common, among them the need to deliver reasonably high-quality video from multiple cameras spread over large areas and to accomplish this using limited bandwidth [2]. One way to optimize such systems is to control the amount of redundant or irrelevant information transmitted by each camera. In this article we describe a system of adaptive video compression that automatically adjusts the amount of information transmitted according to how semantically "interesting" a part of a video is likely to be.

Consider the example of a video surveillance application, such as in a hospital or airport, where hundreds of cameras might be deployed to monitor tens of thousands of square meters. Systems of this type typically stream raw video feeds from all cameras to a central server for observation, analysis and possibly further processing. This creates a bottleneck at the central server, and bandwidth limitations become a critical issue in overall system efficiency. This bandwidth problem becomes even more acute when wireless IP cameras are deployed – an option that is becoming increasingly popular due to their rapid re-configurability and lack of infrastructure requirements such as cabling. Note also that a large percentage of bandwidth is expended transmitting scenes of little or no interest because they do not contain objects of semantic interest (e.g., people, cars or airplanes). In such application scenarios selectively compressing video streams depending on the semantic content of each frame can result in significant bandwidth savings.

Another video streaming application that can benefit from this type of semantic adaptation are the UHF networks commonly used to stream video from dash-cams installed in state and local police cars. Many police departments require that dash-cams be used to record incidents and that they stream video back to a central headquarters for monitoring and archiving. At any one time, tens or even hundreds of cameras might be streaming video and in this application bandwidth is severely limited by the limitations imposed by using UHF radio frequencies for transmission. Again, significant amounts of bandwidth can be wasted transmitting irrelevant portions of the video frame that contain no semantically relevant information in the form of faces, people, licence plates, etc.

In both of these examples, bandwidth is squandered by transmitting entire video frames at high bitrates. That is, the same number of bits is dedicated to encoding irrelevant portions of the frame, portions that have no intrinsic value to either application because they contain static and uninteresting objects, as is used to encode truly interesting parts of the frame that contain people, identifying details of cars or faces. Our approach to this problem is to detect interesting portions of video frames and allocate more bits in the compressed representation to them. The rest of the frame is allocated fewer bits in the compressed stream by smoothing away details that would otherwise be unnecessarily encoded in the transmitted video.

Robust and accurate object detectors have almost become

978-0-7695-4589-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ISM.2011.38

a commodity technology in computer vision applications. Reliable, pre-trained detectors exist for pedestrians [3], for text [4], and for a broad variety of general object categories [5], [6]. Despite recent advances in efficient object detection [7], even single object detection still requires a significant amount of computational resources. Application of multiple detectors in order to detect semantically interesting scene elements (e.g., for cars, faces, people, text and licence plates) would require massive computational resources for each individual stream. As such, the detector approach is not feasible for our application scenarios. Note also that new detectors would have to be trained for each potentially interesting scene object, which limits the generality of the detector approach as well.

Most modern detectors are based on high-frequency image features in the form of edges, corner points or other salient image features. The two most popular features are the Histogram of Oriented Gradients (HOG) [3], [5], [7], which is based on a local histogram of image gradient directions, and SIFT descriptors calculated at interesting points in the image [6], [8]. Both of these descriptors are based on image derivatives calculated across a range of scales in the image. As such, in order to preserve such features in a compressed version of the video it is essential to preserve high-frequencies in each frame and transmit them with reasonable fidelity. If we selectively smooth a video frame, preserving regions containing many high-frequency features, we are more likely to preserve recognizability, or "detectability" using modern object detectors in the transmitted representation. At the same time, if we smooth regions that do not contain dense, high-frequency features we will can reduce the amount of information that must be encoded and thus transmitted.

This paper is organized as follows: in the next section we discuss some of the related work on adaptive video encoding; the visual features used are described in section III; a description of our approach to adaptive video compression is provided in section IV. Experimental results and a comparison of adaptive video coding with respect to standard H.264 coding are reported in section V; and conclusions are drawn in section VI.

II. RELATED WORK

Traditional adaptive video compression approaches do not consider the semantic content of video and instead adapt compression depending on the requirements of the network or device used to deliver video to the end user [9]. Semantic video compression, instead, alters the video by taking into account objects [10]–[13] or a combination of objects and events [14], using pattern recognition techniques. Kim and Hwang proposed using video object planes (VOP) coding of MPEG-4 to encode differently the interesting objects in the scene [10]. However in [14] it was shown that this type of compression is less efficient than directly performing re-

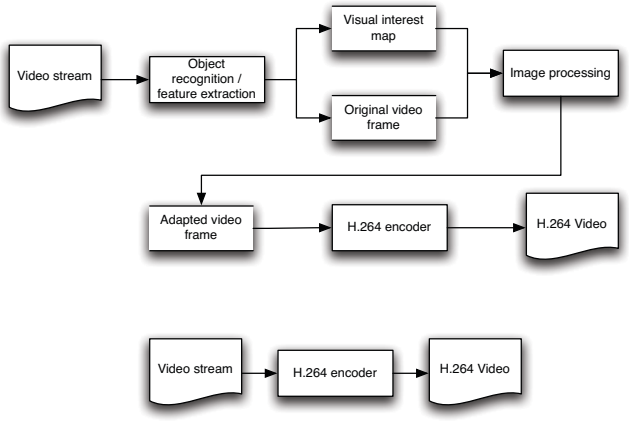


Fig. 1. Our approach to adaptive video coding. (top) The schema for semantic video adaptation. (bottom) standard video coding.

quantization blocks containing an object that is relevant to users.

Adaptation in the compressed domain has been performed through re-quantization [15], spatial resolution downscaling [16], temporal resolution downscaling [17], and by a combination of them [18].

Huang et al. [13] use smoothing to control the amount of bits allocated locally to encode each video frame. The more smoothing applied to a portion of a video frame, the fewer bits will be used to encode that region. Their approach is based on motion segmentation, however, and as such is highly sensitive to scene and camera motion. As such it is not directly applicable in cases where streams mobile or active cameras are used, or to detect static objects, like the licence plates of parked cars. Our approach is directly based on image features correlated with downstream detector features. As such, it is applicable to any type of stream, independent of motion characteristics.

Our approach to adaptive video encoding is based on the observation that the most useful image features for downstream object detection are based on edges and salient interest points. As such, by preserving these features we maximize the ability to detect semantically interesting scene objects after transmission. At the same time, by smoothing features that are unlikely to contribute to positive detections we reduce the amount of irrelevant information transmitted. Fig. 1 illustrates our system for adaptive video coding. The bottom diagram in Fig. 1 illustrates the standard H.264 video coding pipeline. At the top of Fig. 1 is shown our pipeline: before encoding each frame is passed through a sequence of low-level feature extraction (Visual Interest Map), followed by selective smoothing (Image Processing) which smoothes details in uninteresting regions of the images before H.264 encoding.

III. VISUAL FEATURES FOR ADAPTIVE VIDEO COMPRESSION

In most surveillance applications the most interesting objects are faces, people and cars. Face and people detectors are both often trained on features based on gradients [3], [19]. Other, more general object detectors are also based on similar features [5]. Moreover edge features are often exploited to estimate crowd density [20], [21] without resorting to object detection and tracking.

Since all MPEG coding standards perform an initial step of spatial color subsampling, as a form of lossy compression, the visual features we use for this work are based on the luminance of pixels. Another advantage of this is that the color space used in MPEG and M-JPEG standards is YCbCr, so it is possible to extract directly the luminance information from the Y channel, without requiring any conversion.

As mentioned above, the features we use have been selected to be highly correlated with those used for object detection. In particular, corner points can be used to effectively detect text (useful in the case of license plates or identifying text on clothing) in video [4], [22], and edge features in the form of image gradients are used in many state-of-the-art object detectors [5]–[7]. Since our application scenario requires onboard adaptive encoding, we selected the features used in order to minimize the computational resources required.

For detecting corner features we use the FAST detector [23]. This detector has recently been used on mobile phones to augment reality using the limited computational resources of the mobile device [24]. The FAST detector is an approximation of the SUSAN detector in which a circular region around a candidate corner is analyzed to determine if differences between the central point and a pre-defined sequence of pixels in the region satisfy a learned contrast threshold. This detector has been shown to produce very stable features and is the most efficient and robust corner detection algorithm available.

Edge features are characterized by high image gradient values perpendicular to the edge itself. We use the Sobel gradient operator to detect pixels in the image corresponding to edges. The Sobel operator is very efficient, involving only two 3×3 convolutions of the image:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \quad (1)$$

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I, \quad (2)$$

where I is the image to be processed (the video frame) and $*$ represents the 2D convolution operator. The Sobel edge response G, an estimate of the gradient magnitude at each point in the image, is then computed as:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}. \quad (3)$$

The FAST and Sobel edge features will be used in the next section to drive adaptive image compression. Essentially, regions of the image containing a high density of FAST corner responses, or a high density of Sobel edge responses, should be preserved. Other regions can be smoothed in order to reduce detail encoded in transmission. See Fig. 2 for an example of the extracted features on a typical video frame.

IV. ADAPTIVE VIDEO COMPRESSION

MPEG video coding is based on two basic techniques [25], [26]: transform domain-based compression (intra-coding), where blocks of 8×8 pixels are processed to compute discrete cosine transform (DCT) of each, representing it as a matrix of coefficients; and block-based motion compensation performed on macroblocks, i.e. groups of 2×2 blocks (16×16 pixels), coding them with motion vectors and with the DCT coefficients of the "residual block" obtained from motion estimation. In both cases the DCT coefficients are quantized, as a lossy compression step, so that the high frequency coefficients go to zero in order to represent them with efficient Run Length Encoding (RLE) encoding and Variable Length Codes (VLC). The residual block typically contains high frequency components that have to be quantized differently from the intra-coded blocks.

In our approach we reduce the bandwidth needed for video streaming by selectively smoothing parts of each frame. We do not directly exploit the temporal structure of videos in order to reduce the need of buffering and to allow visual feature extraction even on moving cameras. This approach helps the encoder to more efficiently compress the DCT coefficients of both intra-coded and residual blocks since they will contain fewer high frequency components. The smoothing is defined by a set of semantic binary masks which are generated by collecting statistics of low-level visual features in a video frame. These masks could also be defined by a set of detectors for objects of interest. The result would be a binary mask defined by the bounding boxes of objects detected in each frame as shown in Fig. 4. This approach performs extremely well (see Tab. II in Sec. V) but does not allow a sufficient frame-rate on low-end computational architectures and as discussed above does not generalize well when the number of objects of interest increases.

We instead design our masks by splitting each frame into square pixel regions. Region size is selected in order to optimally fit the DCT encoded pixel macroblocks used in H.264 video encoding. We tested 8×8, 16×16 and 32×32 regions. Smoothed regions will be assigned fewer bits by the encoding algorithm, allowing more bits to be assigned to non-smoothed ones. This approach will therefore decrease

(a) (b) (c)

Fig. 2. Examples of the features used for adaptive image encoding. (a) The original video frame. (b) Corners points detected using the FAST detector. (c) Edges detected using the Sobel gradient operator. Note how both the corner and edge features tend to be concentrated on and around the semantically relevant objects in the scene: people, cars, license plates, etc.

the bandwidth needed for streaming while maintaining a high quality of interesting frame regions.

The amount of smoothing applied to each region is determined by the density of Sobel and FAST features contained therein. Let I denote the current image to be encoded, and $F(\mathbf{x})$ and $S(\mathbf{x})$ the FAST and Sobel feature responses at pixel $\mathbf{x}$, respectively. Also, let B_i denote the i-th block of the image and let the function $B(\mathbf{x})$ map pixel $\mathbf{x}$ to the block containing it. We define the following feature threshold functions on local image blocks:

$$F(B_i) = \{x | x \in B_i \mid F(\mathbf{x}) > T_F\} \tag{4}$$

$$S(B_i) = \{x | x \in B_i \mid S(\mathbf{x}) > T_S\}, \tag{5}$$

where T_F and T_S are empirically determined thresholds on the FAST and Sobel feature responses.

Assuming there are n levels of smoothing, we will now define smoothing masks that are based on feature densities in each image block. The masks correspond to increasing feature densities. The i-th level mask corresponding to FAST feature density is defined as:

$$M_i^F(\mathbf{x}) = \begin{cases} 1 & \text{if } \tau_{i-1}^F \leq |F(B(\mathbf{x}))| < \tau_i^F \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where τ_i^F for $i \in \{0, 1, \ldots, n\}$ is a strictly increasing series of thresholds used to determine which feature densities correspond to which mask. We require that $\tau_0^F = 0$ and $\tau_n^S = w \times h$, where w and h are the width and height of the image blocks used for encoding. These restrictions ensure that the sequence of image masks M_i completely partitions the image:

$$\bigcap_i \bigcup_{\mathbf{x}} M_i^F(\mathbf{x}) = \emptyset \tag{7}$$

$$\bigcup_i \bigcup_{\mathbf{x}} M_i^F(\mathbf{x}) \circ I = I \tag{8}$$

.

The i-th level mask corresponding to Sobel feature density is similarly defined:

$$M_i^S(\mathbf{x}) = \begin{cases} 1 & \text{if } \tau_{i-1}^S \leq |F(B(\mathbf{x}))| < \tau_i^S \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

with identical conditions on τ_i^S as for FAST features above.

The final smoothed image can now be written as:

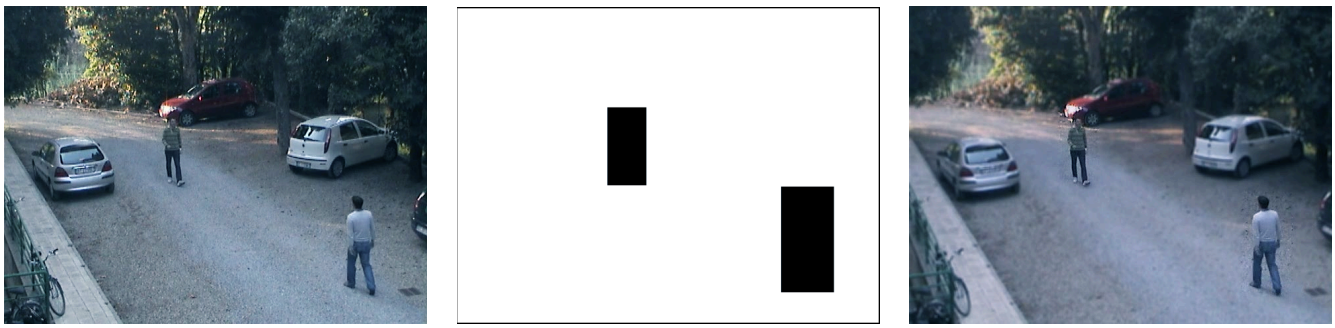
$$I_s = \sum_{i=1}^{n} (M_i^S \circ M_i^F \circ I) * G_{\sigma_i}, \tag{10}$$

where $M_i^S \circ M_i^F \circ I$ represents the Hadamard (element-by-element) multiplication of the feature masks and image I, and G_{σ_i} is a Gaussian function with variance σ_i^2. I_s is an adaptively smoothed version of the original image I. Based on the density of FAST and Sobel features in each $w \times h$ block of the image, a variable amount of smoothing will be applied. The amount of smoothing applied is controlled by the sequence σ_i, while the density thresholds τ_i^S and τ_i^F are used to determine how interesting each block is in terms of each feature.

V. EXPERIMENTAL RESULTS

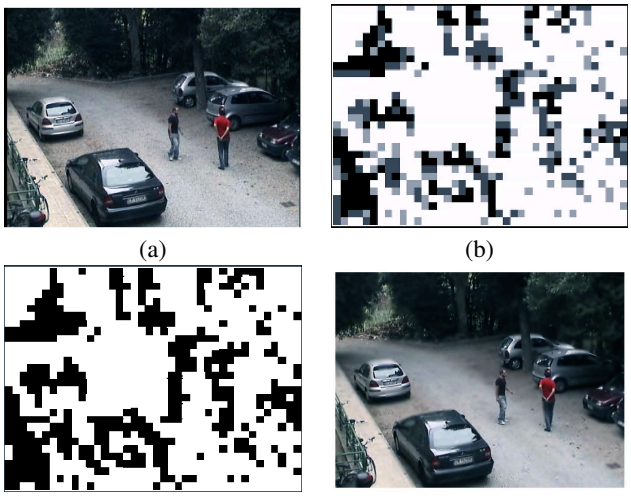
In order to evaluate the performance of our approach to adaptive video compression, we acquired a set of three test videos using a real-world surveillance setup. Two Sony SNC RZ30 cameras recorded videos of a parking lot at 640×480 pixels and 25 FPS for a total of 2327 frames. The videos were been recorded in MPEG-4 format using the H.264 codec provided by the open source library x264. Each video is encoded with an average bitrate of 3532,44 Kbit/s.

We evaluate the performance of our algorithm by measuring the structural similarity index (SSIM) [27] and comparing the non-H.264 compressed videos and videos compressed with our approach, computing SSIM on the computed masks. SSIM is a visual quality assessment metric that models the perception of compression artifacts by the human visual system better than other standard quality measures based on peak signal-to-noise ratio (PSNR) or



(a) (b) (c)

Fig. 4. Adaptive compression driven by pedestrian detector. A frame with two people (a), the masks built with the pedestrian detector (b) and the final adaptively encoded frame (c). All the scene but the two pedestrian is smoothed.



(a) (b) (c) (d)

Fig. 3. An example of feature-preserving adaptive compression. (a) Original video frame. (b) The multi-level mask computed from FAST and Sobel responses. (c) A single level mask from FAST and Sobel features. (d) The frame compressed using the single-level mask. Note how persons and license plates are encoded with high enough quality to preserve recognizability, while background details are smoothed away.

mean squared error (MSE). In fact MSE, and consequently PSNR, perform badly in predicting human perception of image fidelity and quality: MSE values of distorted images that present dramatically and visibly different visual qualities can be nearly identical [28]. For this reason its use has been proposed to drive the motion compensation coding of H.264 [29] and some encoders, like x264[1], have started to use it to drive the adaptation of the quantization coefficients during compression. SSIM is defined as:

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{(\mu_X^2 + \mu_Y^2 + C_1)(\sigma_X^2 + \sigma_Y^2 + C_2)}, \quad (11)$$

where X and Y are images, μ_X is the average of the luminance of X, μ_Y is the average of the luminance of Y, σ_{XY} is the covariance of the contrast of X and Y, σ_X^2 is the variance of the contrast of X, σ_Y^2 is the variance of the contrast of Y, $C_i = (K_i L)^2$ are constants used to avoid instability when $\mu_X^2 + \mu_Y^2$ is near 0, where $K_i \ll 1$ and L is the amplitude of the range of values that a pixel can have. SSIM is typically computed on 8×8 pixel windows, and can assume values between $[-1, 1]$, where 1 means that two images are identical. SSIM is measured in the non-smoothed regions only.

After an initial evaluation we found that the best performance is obtained by exploiting both low-level features (FAST and Sobel), using blocks of 16×16 and smoothing all regions without corners and with less than 3 non-zero pixels. Smoothing is performed using block filters approximating a Gaussian filter of the correct σ. In a preliminary set of experiments we also explored the possibility of using multiple levels of smoothing, in particular we used three levels of smoothing selected with three thresholds for both features. The performance of this approach is less appealing, see Tab. I, with respect to plain binary mask driven smoothing. Fig. 3 illustrates the performance of our approach, along with example masks derived from low-level features, on a typical video frame. Note how semantically meaningful features like the license plate and persons are preserved in the compressed representation.

V-A. Feature and Compression Evaluation

We evaluated low-level features, mask size and type for a set of sensible configurations. In particular, we tested the FAST and Sobel features separately and together. Results are reported for each feature or combination for the best window size. The average SSIM gain (ΔSSIM) is obtained with the

[1] http://www.x264.org

Method	ΔSSIM (binary masks)	ΔSSIM multi-level masks
Sobel 8	0.013	-0.002
FAST 32	0.024	0.009
Sobel + Fast 16	0.027	0.01

Table I. Comparison of different low-level features and mask generation strategies. Binary masks correspond to the situation where only two levels of smoothing are used ($n = 2$), whereas for multi-level masking three levels were used ($n = 3$). Average ΔSSIM is reported with respect to identical bitrate video encoded with standard H.264 compression. CRF is varied in order to obtain files of the same bitrate.

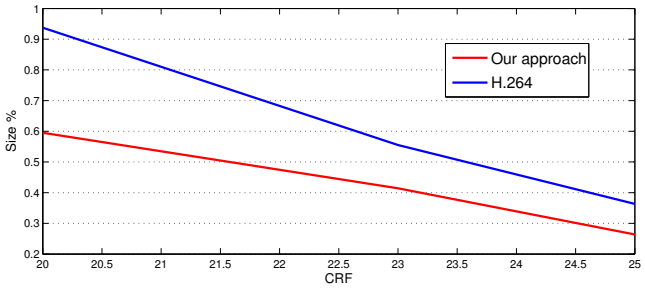


Fig. 5. Average file size obtained by varying the CRF. File size is normalized with respect to the original (high quality) video.

following procedure: first videos are compressed with H.264 with Constant Rate Factor (CRF) in the range 25-20; for each of these files we compress the original video (V_o) with our adaptive technique tuning the CRF (lowering) in order to obtain approximately the same bitrate. We finally compute ΔSSIM as SSIM(V_{ac},V_o) - SSIM(V_c,V_o), where V_{ac} and V_c are the adaptive and H.264 coded videos, respectively. A negative value means that our technique degrades the video more than standard H.264 encoding, note that this happens only for Sobel features alone and with multi-level masks. For all other combinations we improve the SSIM without increasing the bitrate.

Apart from the increase of quality in regions of interest we are mainly interested in the reduction of bandwidth. To this end we measure how the file size decrease with the increase of CRF for adaptive encoded videos and H.264 encoded videos. As shown in Fig. 5 we are able to spare between 40% to 10% of the bandwidth depending on the quality of the final encoding. It is also interesting to see how our encoder is able to retain the appearance of the original video; Fig. 5 shows the average behavior of our algorithm.

V-B. Efficiency of Our Approach

We report in Tab. II the frame rate of our approach compared with the frame rate obtained using the pedestrian detector-based approach on the same machine. Low-level features allow our system to run at a very high frame rate,

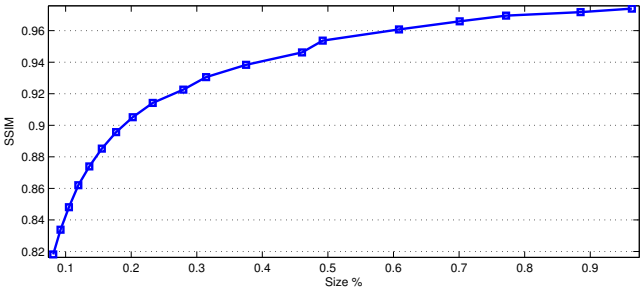


Fig. 6. SSIM versus file size (normalized with respect to the original video size). For a video size of about 20% of the original, the SSIM computed on the interesting areas is above 0.9 and approaches 0.95. Even when video size decreases rapidly, below 20% of the original size, SSIM still remains above 0.8.

Video	Method	Frames/sec	Video Size (MB)
V1	Pedestrian Detector	1.3	2.2
V1	Sobel + Fast16	71.8	4.1
V2	Pedestrian Detector	1.6	3.0
V2	Sobel + Fast16	87.0	7.2
V3	Pedestrian Detector	1.9	15.5
V3	Sobel + Fast16	71.5	18.0

Table II. Frame rate comparison for the two feature extraction approaches. File size is also reported for the same CRF (17).

permitting our system to stream video in real-time. Though pedestrian detection allows us to reduce the video size by a higher factor, it should be noted that this is mainly due to the fact that masks generated by the pedestrian detector drive the algorithm to smooth most of the frame, reducing the bandwidth needed. Even if this seems a desirable property, objects other than pedestrians are encoded with a lower quality. As an example, license plates are unreadable and other car details are consistently degraded.

V-C. Semantic Cue Preservation

Videos compressed and transmitted with our adaptive encoding will be subsequently inspected either by personnel or machines. In the following experiment we measure how encoded videos preserve the image features that allow high level object detectors to extract semantic information from videos. In particular, we processed a video with the Dalal&Triggs [3] pedestrian detector before and after the adaptive encoding for three levels of adaptive compression. From Tab. III it appears that the performance of the pedestrian detector is not substantially affected by our adaptive compression. In particular, the precision on the compressed video is reduced but the overall recall is improved. We also report the increase in true positives and false positives for compressed videos, which also explains the increase in recall.

Video (size)	Precision	Recall	ΔTP/P	ΔFP/P
Original (7.9 MB)	.92	.67	-	-
Compressed (1.8 MB)	.89	.81	.11	.09
Compressed (4.4 MB)	.93	.84	.13	.05
Compressed (7.2 MB)	.89	.84	.11	.09

Table III. Performance of a pedestrian detector on the original and adaptively encoded frames. Precision is slightly reduced but recall is increased.

VI. CONCLUSIONS

In this paper we have presented a novel method for semantic video coding based on low-level features that require a very limited computational resources. The technique can be used as a pre-processing state before DCT encoding, and is able to reduce the size of videos down to half the original size, while maintaining the perceptive quality of the areas considered of interest. The approach has been shown to have also a beneficial effect on automatic analysis of the compressed video, improving the performance of the person detector based on the histogram of gradients.

ACKNOWLEDGEMENTS

This work is partially supported by the EU EraSME ORUSSI Project and by SELEX Communications.

VII. REFERENCES

[1] N. Qadri, M. Altaf, M. Fleury, and M. Ghanbari, "Robust video communication over an urban VANET," *Mobile Information Systems*, vol. 6, no. 3, pp. 259–280, 2010.

[2] S. Z. Hussain, "Performance evaluation of H.264/AVC encoded video over TETRA enhanced data service (TEDS)," Master's thesis, Helsinki University of Technology, 2009.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of CVPR*, 2005.

[4] L. Sun, G. Liu, X. Qian, and D. Guo, "A novel text detection and localization method based on corner response," in *Proc. of the IEEE international conference on Multimedia and Expo (ICME)*. IEEE Press, 2009, pp. 390–393.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[7] M. Pedersoli, J. Gonzàlez, A. D. Bagdanov, and X. Roca, "Efficient discriminative multiresolution cascade for real-time human detection applications," *Pattern Recognition Letters*, vol. 32, no. 13, pp. 1581–1587, October 2011.

[8] C. H. Lampert, "Detecting objects in large image collections and videos by efficient subimage retrieval," in *Proc. of ICCV*, 2009.

[9] B. Tseng, C.-Y. Lin, and J. Smith, "Using MPEG-7 and MPEG-21 for personalizing video," *IEEE Multimedia*, vol. 11, no. 1, pp. 42–52, 2004.

[10] C. Kim and J.-N. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 2, pp. 122–129, 2002.

[11] A. Cavallaro, O. Steiger, and T. Ebrahimi, "Semantic segmentation and description for video transcoding," in *Proc. of the IEEE international conference on Multimedia and Expo (ICME)*, 2003.

[12] T. Nishi and H. Fujiyoshi, "Object-based video coding using pixel state analysis," in *Proc. of IEEE ICPR*, 2004.

[13] H.-J. Huang, X.-M. Zhang, and Z.-W. Xu, "Semantic video adaptation using a preprocessing method for mobile environment," in *Proc. of IEEE CIT*, 2010.

[14] M. Bertini, A. Del Bimbo, A. Prati, and R. Cucchiara, "Semantic adaptation of sport videos with user-centred performance analysis," *IEEE Transactions on Multimedia*, vol. 8, no. 3, pp. 433–443, Jun 2006.

[15] O. Werner, "Requantization for transcoding of MPEG-2 bit streams," *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 179–191, 1999.

[16] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolution and different encoding formats," *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 101–110, 2000.

[17] Z. Lei and N. Georganas, "H.263 video transcoding fo spatial resolution downscaling," in *Proc. of Conference Information Technology: Coding and Computing*, 2002.

[18] Y. Liang and Y.-P. Tan, "A new content-based hybrid video transcoding method," in *Proc. of ICIP*, 2001.

[19] P. A. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of CVPR*, 2001, pp. 511–518.

[20] D. Kong, D. Gray, and H. Tao, "Counting pedestrians in crowds using viewpoint invariant training," in *Proc. of the IEEE British Machine Vision Conference (BMVC)*, 2005.

[21] A. Chan, M. Morrow, and N. Vasconcelos, "Analysis of crowded scenes using holistic properties," in *Proc. of IEEE Intl. Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2009)*, 2009.

[22] M. Bertini, C. Colombo, and A. Del Bimbo, "Automatic caption localization in videos using salient points," in *Proc. of the IEEE international conference on Multimedia and Expo (ICME)*, 2001, pp. 68–71.

[23] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. of ECCV*, 2006.

[24] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, 2007.

[25] K. Jack, *Video Demystified*. LLH Publishing, 2001.

[26] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560 –576, july 2003.

[27] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, 2004.

[28] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, pp. 98–117, 2009.

[29] C.-l. Yang, H.-x. Wang, and L.-M. Po, "A novel fast motion estimation algorithm based on ssim for h.264 video coding," in *Advances in Multimedia Information Processing – PCM 2007*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, vol. 4810, pp. 168–176.