A GPU based real-time video compression method for video conferencing

Stamos Katsigiannis, Dimitris Maroulis
Department of Informatics and Telecommunications
University of Athens
Athens, Greece
{stamos , dmaroulis}@di.uoa.gr

Georgios Papaioannou
Department of Informatics
Athens University of Economics and Business
Athens, Greece
gepap@aueb.gr

Abstract—**Recent years have seen a great increase in the everyday use of real-time video communication over the internet through video conferencing applications. Limitations on computational resources and network bandwidth require video encoding algorithms that provide acceptable quality on low bitrates and can support various resolutions inside the same stream. In this work, the authors present a scalable video coding algorithm based on the contourlet transform that incorporates both lossy and lossless methods, as well as variable bitrate encoding schemes in order to achieve compression. Furthermore, due to the transform utilized, it does not suffer from blocking artifacts that occur with many widely adopted compression algorithms. The proposed algorithm is designed to achieve real-time performance by utilizing the vast computational capabilities of modern GPUs, providing satisfactory encoding and decoding times at relatively low cost. These characteristics make this method suitable for applications like video conferencing that demand real-time performance. The performance and quality evaluation of the algorithm shows that the proposed algorithm achieves satisfactory quality and compression ratio.**

Keywords—real-time video encoding; GPU computing; video conferencing; surveillance video; contourlet transform;

I. Introduction

Video conferencing applications have been extensively utilized in recent years in order to provide real-time video communication over the internet and other IP networks. The cost of computational and network resources underlines the need for highly efficient video coding algorithms that could be successfully utilized in casual computational devices with limited bandwidth. As a result, the most desirable characteristic of such an algorithm would be the ability to maintain satisfactory visual quality while achieving good compression. Low computational complexity and real-time performance would also be an advantage since it would allow the algorithm to be used in a wide variety of less powerful computers. Additionally, the ability to adapt to the network's end-to-end bandwidth and transmitter/receiver resources, as well as resistance to packet loss during transmission would be much desirable attributes. Another highly advantageous characteristic for a modern video compression algorithm would be resistance to noise induced by low quality video sources like web cameras. Most state of the art video compression techniques fail to achieve real time performance without the use of dedicated hardware due to their high computational complexity. Moreover, in order to achieve optimal compression and quality, these techniques depend on multipass statistical and structural analysis of the whole video content, a procedure that cannot happen in cases of live video stream generation as in the case of video-conferencing. Another disadvantage of popular Discrete Cosine Transform (DCT)-based algorithms like H.264 [1], VP8 [2], other algorithms belonging to the MPEG family, etc. is that at low bitrates they introduce visible block artifacts that distract the viewers and degrade the communication experience. In [3], we proposed a novel algorithm for high quality real-time video encoding designed for content obtained from low resolution sources like web cameras, surveillance cameras, etc. However, as mentioned on the conclusions of the aforementioned work, although the algorithm provided satisfactory visual quality, it was still at an early version and lacked some basic elements that would provide sufficient compression efficiency. In this work, the authors expand their previous work in order to achieve increased compression efficiency while minimizing the loss of visual quality

A critical aspect of every video encoding algorithm design is the texture representation method utilized. Frequency domain methods like the Fourier transform, the Discrete Cosine transform, the Wavelet transform, etc. have been extensively studied and utilized in image and video encoding algorithms. The proposed video encoding algorithm is based on the Contourlet Transform (CT) [4], which partially addresses some of the limitations of the aforementioned methods. One of the most significant advantages of the Contourlet Transform is its multiscale and directional decomposition, providing anisotropy and directionality, features missing from traditional transforms like the Discrete Wavelet Transform [4]. In recent years, methods based on the Contourlet Transform have been proposed for a variety of texture analysis applications, including image denoising [5], medical and natural image classification [6], image fusion [7], etc. Additionally, by taking advantage of the computational power offered by modern graphics processing units (GPUs), a GPU-based contourlet transform encoder is able to provide all the CTs benefits, while achieving real-time performance in commodity hardware.

The rest of this paper is organised in four sections. Section 2 provides some background knowledge needed for better understanding the proposed algorithm, which is presented in section 3. Then, an experimental study for the evaluation of the algorithm is provided in section 4, whereas conclusions and future perspectives of this work are presented in section 5.

II. Background

A. The Contourlet Transform

The Contourlet Transform (CT) [4] is a directional multiscale image representation scheme proposed by Do and Vetterli that provides directionality and anisotropy, and is effective in representing smooth contours in different directions on an image. The method is realized as a double filter bank consisting of first the *Laplacian Pyramid* (LP) [8], which detects the point discontinuities of the image and then the *Directional Filter Bank* (DFB) [9], which links those point discontinuities into linear structures. In each LP level, the image is decomposed into a downsampled lowpass version of the original image and a more detailed image with the supplementary high frequencies. This scheme can be iterated continuously in the lowpass image and provides a way to obtain multiscale decomposition. The DFB is a 2D directional filter bank that can achieve perfect reconstruction and the simplified DFB used for the contourlet transform decomposes the image into 2^l subbands with wedge-shaped frequency partitioning [10], with l being the level of decomposition.

After the LP decomposition, bandpass images are fed into a DFB in order to capture the directional information. This scheme can be iterated on the coarser image levels, with image size being the only restriction due to the continuous downsampling. The combined result is a double iterated filter bank that decomposes images into directional subbands at multiple scales and is called the C*ontourlet Filter Bank*. The contourlet coefficients are similar to wavelet coefficients since most of them have values near zero and only those located near the edge of the objects have larger magnitudes [11].

A wide variety of filters can be used for calculating the LP and DFB. In the presented work, the Cohen and Daubechies 9-7 filters [12] have been utilized for the Laplacian Pyramid, while for the Directional Filter Bank, these filters were mapped into their corresponding 2D filters using the McClellan transform as proposed in [4].

B. The YCoCg colour space

It is well established in literature that the human visual system is significantly more sensitive to variations of luminance compared to variations of chrominance. Exploiting this fact, the encoding of the luminance channel of an image with higher accuracy than the chrominance channels provides a low complexity compression scheme that succeeds in maintaining satisfactory visual quality while offering significant compression. This technique, commonly referred to as *chroma subsampling*, is utilized by various image and video compression algorithms.

First introduced in H.264 compression and recently proposed for use on a real-time RGB frame buffer compression scheme using chrominance subsampling [13], the RGB-to-YCoCg transform decomposes a colour image into luminance (Y), orange chrominance (Co) and green chrominance (Cg) components. It was developed primarily to address some limitations of the different YCbCr colour spaces [14] and has been shown to exhibit better decorrelation properties than YCbCr and similar transforms [15].

The Co and Cg components should be stored with higher precision than the RGB components in order for the reverse transform to be perfect and avoid rounding errors. Nevertheless, experiments showed that using the same precision for the YCoCg and RGB data when transforming from RGB to YCoCg and back results in an average PSNR of more than 58.87 dB [3] for typical natural images. This loss of quality results to no visible alteration of the image since it cannot be perceived by the human visual system.

III. THE ALGORITHM

TABLE I. THE ENCODING ALGORITHM

Steps
1: Start
2: Input RGB frame
3: Convert to YCoCg
4: Downsample Co and Cg by N
5: Decompose Y with the Contourlet Transform
6: Quantize CT coefficients / Keep the M% most significant CT coefficients
7: Round all components' elements to the nearest integer
8: IF the frame is an internal frame
9:     Calculate the frame as the difference between the frame and the previous keyframe
10:     Run-length encoding of Co, Cg and the lowpass CT component of Y
11: END IF
12: Run-length encoding of the directional subbands of Y
13: DEFLATE all components seperately
14: IF frame is NOT the last frame
15:     GOTO Start
16: END IF
17: Finish

Steps 3,4,5,6,7 and 9 refer to calculations performed on the GPU, while the other steps refer to calculations performed on the CPU.

The steps of the presented algorithm for the encoding process are listed in Table I, while the decoding process can be characterized as the reverse procedure. The most computationally intensive steps of the algorithm are processed on the GPU. GPUs are ideal for performing massively parallel computations, especially when inter-thread data exchange can be brought to a minimum. They are commodity hardware and are easy to write optimized code for. Furthermore, the most computationally intensive part of the presented algorithm is the 2D convolutions needed for calculating the contourlet transform. Calculating the 2D convolutions using the FFT provides increased efficiency due to its parallel nature and also takes advantage of a very optimized implementation on the GPU in CUDA

Assuming that input frames are encoded in the RGB colour space, the first step of the algorithm is the conversion from RGB to YCoCg colour space. Then, the chrominance channels are subsampled by a user-defined factor *N*, exploiting the fact that the human visual system is relatively insensitive to chrominance variations. The user-defined factor *N* directly affects the output's visual quality and the compression achieved. For the reconstruction of the chrominance channels at the decoding stage, the missing chrominance values are replaced by using bilinear interpolation from the four neighboring pixels. Using the bilinear interpolation method provides smoother chrominance transitions, enhancing the output visual quality. A simpler method would be to replace the missing values with the nearest available subsampled chrominance values (Nearest Neighbor method). However it can introduce artifacts in the form of mosaic patterns in regions with strong chrominance transitions depending on the subsampling factor.

After manipulating the chrominance channels, the luminance channel is decomposed using the contourlet transform, with the levels and filters used for the contourlet transform decomposition being also defined by the user. Decomposing into multiple levels using the LP provides the means to separately store different scales of the input that can be in turn individually decoded, thus providing the desirable scalability feature of the algorithm, i.e. multiple resolutions inside the same video stream. This property allows video coding algorithms to adapt to the network's end-to-end bandwidth and transmitter/receiver resources. The quality for each receiver can be adjusted by just dropping the encoded information referring to higher resolution than requested, without the need to re-encode the video frames at the source. This feature can be invaluable for video broadcast systems and internet video services, as the video quality is selected at the destination, while the source streams and the network caches a single video. Moreover, due to the downsampling of the lowpass component at each level, it provides enhanced compression.

Then, the number of contourlet coefficients from the directional subbands of the luminance channel is reduced. The user has the choice between two methods for reducing the number of coefficients: 1) By retaining only a user-specified percentage of the most significant coefficients (coefficient truncation). This procedure leads to a large number of zero-valued sequences inside the elements of the directional subbands, a fact exploited by run length encoding in a later stage of the algorithm. Furthermore, it also provides a means to suppress the noise induced by low-quality sensors, usually encountered in web-cameras, since random noise is largely unstructured and therefore not likely to generate significant contourlet coefficients [4]. The amount of coefficients dropped drastically affects the output's visual quality as well as the compression ratio. 2) By quantizing the coefficients using custom quantization matrices, effectively reducing the possible values of the coefficients, as proposed in this work. After this step, the precision of all the frame's components (for both the luminance and chrominance channels) is reduced by rounding to the nearest integer, resulting in all the components fitting into 8 bit variables.

Computations up to this point are all performed on the GPU, avoiding unnecessary memory transfers from the main memory to the GPU memory and vice versa. The input frame is passed to the GPU at the beginning of the encoding process and the encoded components are returned to the main memory. After passing all the components to the main memory, the directional subbands of the luminance channel are encoded using a run length encoding scheme on the CPU. The insignificant coefficient truncation or the quantization of the directional subbands' coefficients provide large sequences of zero-valued contourlet coefficients that make run length encoding ideal for their compression. Taking into consideration the values and the distribution of contourlet coefficients at the directional subbands, the zero valued coefficients are run-length-encoded only along the horizontal direction. Run length encoding of all the other sub-band values would offer minimum compression gain and does not justify the increased computational cost.

Video frames are divided into two categories by the algorithm; keyframes and internal frames. Keyframes are frames that are encoded using the steps described before, while internal frames are the frames between two consecutive key frames. The number of internal frames between two keyframes is a user defined parameter. At the step before the run-length encoding of the directional subbands and before passing the components to the main memory, when a frame is identified as an internal frame all its components are calculated as the difference between the respective components of the frame and those of the previous key frame. This step is also processed on the GPU while all the remaining steps of the algorithm are performed on the CPU. Then, run length encoding is applied to the chrominance channels' components, the lowpass contourlet component of the luminance channel, as well as the directional subbands of the luminance channel. Consecutive frames tend to have small variations, with many identical regions, especially in video sequences with static background. This fact can be exploited by calculating the difference between a frame and the keyframe, providing components with large sequences of zero values and leading to improved compression through the run length encoding stage.

The last stage of the algorithm consists of the use of the DEFLATE algorithm [16] in order to encode each of the frame components separately, potentially in multiple threads. Separate encoding of the chrominance components, the lowpass luminance component and the directional subbands of the luminance channel serves two purposes. The computational time needed for using the DEFLATE algorithm is reduced and the scalability of the algorithm is not affected, allowing for separate decoding of each of the frame's scale. Another advantage of using the DEFLATE algorithm is that it provides a standard bit stream, making the development of decoders for our algorithm and porting them to other platforms easier. Except for its extensive use in many everyday computer applications, the DEFLATE algorithm has also been used for image processing applications, with the most notable one being the PNG image file format [17]. The reverse procedure at the decoding stage is the INFLATE algorithm.

Except for the YCoCg colour space, the algorithm can support the YCbCr and Grayscale colour spaces without the need to alter its core functionality. The process of encoding grayscale videos consists of handling the video as a colour video with only the luminance channel, while omitting the steps referring to the chrominance channels. For the YCbCr color space, due to its similarity with YCoCg, the algorithm remains the same, incorporating the RGB-to-YCbCr conversion at the encoder and the YCbCr-to-RGB conversion at the decoder. However, the difference in range between the

CbCr and CoCg channels should be taken into consideration when manipulating their precision. The luminance and chrominance channels are then identically handled as in the YCoCg-based algorithm.

It must be noted that all the steps that are executed on the CPU are inherently serial and thus cannot be efficiently mapped to a GPU. Processing those steps on the GPU would provide worse performance than on the host CPU.

IV. QUALITY AND PERFORMANCE ANALYSIS

For the evaluation of the presented algorithm, three videos were captured using a VGA web camera that supported a maximum resolution of 640x480 pixels. Low resolution web cameras are very common on everyday personal computer systems showcasing the need to design video encoding algorithms that take into consideration the problems arising due to low-quality sensors. The videos captured were two typical video conference sequences with static background showing the upper part of the human body and containing some motion, and a sequence depicting part of an office, resembling a surveillance video.

Detailed results for each video sample are shown in Tables II-IV. The mean PSNR value for each video and for each video's luminance channel are both presented, alongside the compression ratio achieved for each case. The chrominance channels of the video samples were subsampled by various factors *N* and the video stream contained two resolutions: the original VGA (640x480) as well as the lower QVGA (320x240). The contourlet coefficients of the luminance channel were quantized using three custom quantization matrices created after exhaustive experiments on various image datasets, providing results of various quality and compression levels. The three quantization matrices targeted low, medium and high quality respectively. At each scale, the luminance channel's high frequency content was decomposed into four directional subbands. At the decoding process, the bilinear interpolation method was utilized for the reconstruction of the chrominance channels. The sample videos were all encoded using the scheme that incorporates both key frames and internal frames, as preliminary experiments showed that it outperforms the simple case of compressing all the frames as keyframes. The interval between the key frames was set to five frames for the video-conference samples and to twenty frames for the surveillance video. Sample frames of the encoded videos for the medium and low quality setting are shown on Fig. 2.

TABLE II. VIDEO CONFERENCE SAMPLE 1

	***N*=4**			***N*=8**		
	PSNR ALL (dB)	***PSNR Luma (dB)***	***CR***	***PSNR ALL (dB)***	***PSNR Luma (dB)***	***CR***
HIGH	40.79	49.31	10.62:1	38.85	49.32	12.43:1
MEDIUM	39.83	44.39	27.43:1	38.21	44.39	38.51:1
LOW	37.59	39.02	45.39:1	36.55	39.02	78.67:1

CR: Compression ratio

TABLE III. VIDEO CONFERENCE SAMPLE 2

	***N*=4**			***N*=8**		
	PSNR ALL (dB)	***PSNR Luma (dB)***	***CR***	***PSNR ALL (dB)***	***PSNR Luma (dB)***	***CR***
HIGH	40.21	49.15	12.47:1	38.36	49.16	13.71:1
MEDIUM	39.31	44.28	25.89:1	37.75	44.28	36.15:1
LOW	36.46	37.56	40.28:1	35.58	37.56	78.23:1

CR: Compression ratio

TABLE IV. VIDEO SURVEILLANCE SAMPLE

	***N*=4**			***N*=8**		
	PSNR ALL (dB)	***PSNR Luma (dB)***	***CR***	***PSNR ALL (dB)***	***PSNR Luma (dB)***	***CR***
HIGH	36.69	46.89	6.81:1	34.17	46.98	7.62:1
MEDIUM	35.50	40.08	16.86:1	33.47	32.67	24.85:1
LOW	31.60	31.98	45.45:1	25.17	31.98	82.87:1

CR: Compression ratio

The lower PSNR achieved for the surveillance video sample can be explained due to the higher complexity of the scene compared to the video conference samples. More complex scenes contain higher frequency content and higher chromatic variance, a portion of which is then discarded by dropping the contourlet coefficients and downsampling the chrominance channels.

Average execution times for the basic operations of the encoding and decoding algorithm for a frame of the video conference sample 1 are presented on Fig. 1. The computer utilized for the performance tests was equipped with an Intel Core i3 CPU, 4 GB of memory and a NVIDIA GeForce GTX 570 graphics card with 1280 MB of GDDR5 memory. It must be noted that since the algorithm is still under development, the encoder/decoder implementation is not fully optimized for performance.

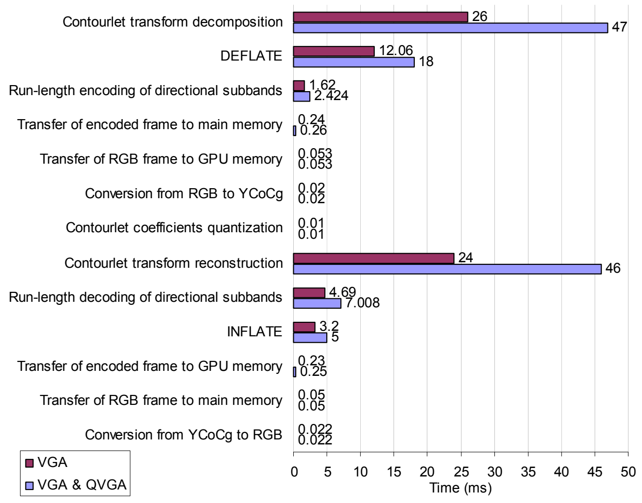

Figure 1. Average execution times for the basic operations of the encoding and decoding algorithm for a frame of the video conference sample 1

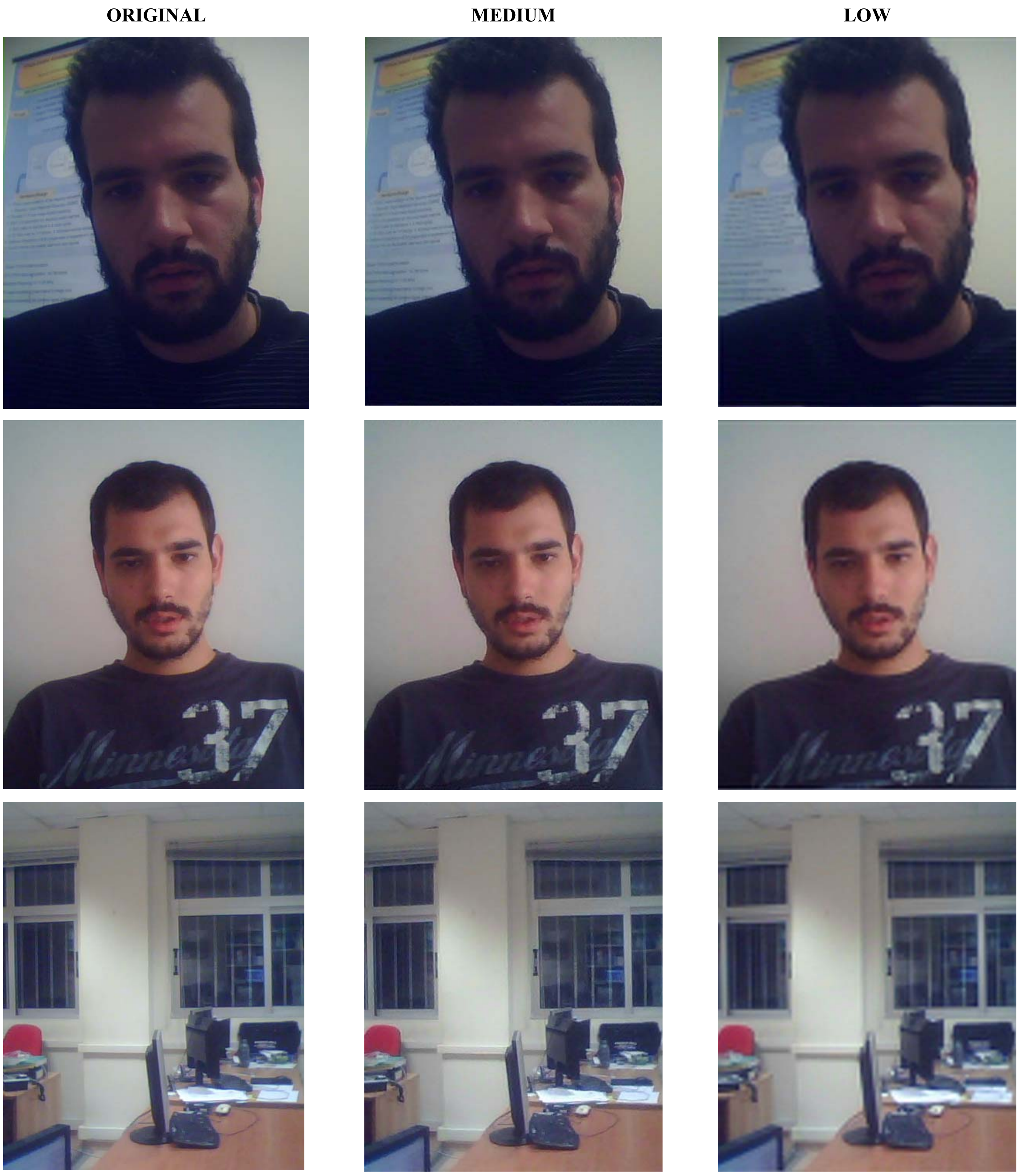

Figure 2. Sample decoded frames for medium and low quality with *N*=4. Frames have been cropped to fit the figure. The first two rows depict the two video conference samples respectively, with the third row being the video surveillance sample. PSNR and compression ratios are shown on Tables II-IV.

V. CONCLUSIONS

In this work, the authors present and evaluate an extension to their previous low complexity algorithm for real-time video encoding based on the contourlet transform and optimized for video conferencing applications and surveillance cameras. The incorporation of the DEFLATE algorithm to the method, along with the various quantization steps provided increased compression efficiency compared to the older version, while introducing minimal visual quality degradation. The scalable video compression scheme that the algorithm provides is ideal for video conferencing content. It achieves high quality encoding and increased compression efficiency for static regions of the image, while maintaining low complexity and adaptability to the receivers' resources. Allowing the video stream to contain various resolutions enables the receiver to select the desired quality by dropping the components referring to higher quality than needed. This ability avoids the need for reencoding at the source or maintaining multiple copies of cached video content in streaming video service networks. Furthermore, depending on the method used for manipulating the contourlet coefficients of the directional subbands of the luminance channel, the algorithm has the inherent ability to suppress the noise induced by low-quality sensors. This is achieved through the manipulation of the structural characteristics of the video through the rejection of insignificant coefficients. In the cases where higher compression is needed, the visual quality degradation is much more eye-friendly than with other well established video compression methods. The proposed method does not suffer from artificial block artifacts but introduces fuzziness and blurring, providing smoother images. Another advantageous characteristic of the presented algorithm is that it can exploit general purpose GPU computing techniques in order to provide enhance computational efficiency, by migrating its most computationally intensive parts on the GPU. The experimental evaluation of the presented algorithm provided promising results. Future work could include an improvement on the handling of the chrominance channels in order to reduce the difference in the PSNR values between the full video sequence and its luminance channel.

REFERENCES

[1] Recommendation ITU-T H.264, Advanced video coding for generic audiovisual services, 2012.

[2] RFC 6386, "VP8 Data format and decoding guide", 2011.

[3] S. Katsigiannis, G. Papaioannou, and D. Maroulis, "A contourlet transform based algorithm for real-time video encoding," Proceedings of SPIE 8437, 2012.

[4] M.N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," IEEE Transactions on Image Processing, vol. 14, no. 12, pp. 2091-2106, 2005.

[5] Z.-F. Zhou and P.-L. Shui, "Contourlet-based image denoising algorithm using directional windows," Electronics Letters, vol. 43, no. 2, pp. 92-93, 2007.

[6] S. Katsigiannis, E. Keramidas, and D. Maroulis, "A contourlet transform feature extraction scheme for ultrasound thyroid texture classification," Engineering Intelligent Systems, vol. 18, no. 3/4, 2010.

[7] B. Yang, S. Li, and F. Sun, "Image fusion using nonsubsampled contourlet transform," 4th International Conference on Image and Graphics (ICIG 2007), pp. 719-724, August 2007.

[8] PJ. Burt and EH Adelson, "The laplacian pyramid as a compact image code," IEEE Transactions on Communications, vol. 31, no. 4, pp. 532-540, 1983.

[9] RH Bamberger and MJT Smith, "A filter bank for the directional decomposition of images: Theory and design," IEEE Transactions on Signal Processing, vol. 40, no. 4, pp. 882-893, 1992.

[10] JM. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Transactions on Signal Processing, vol. 41, no. 12, pp. 3445-3462, 1993.

[11] Z. Yifan and X. Liangzheng, "Contourlet-based feature extraction on texture images," Proceedings of the 2008 International Conference on Computer Science and Software Engineering (CSSE '08), pp. 221-224, 2008.

[12] A. Cohen, I. Daubechies, and JC Feauveau, "Biorthogonal bases of compactly supported wavelets," Communications on Pure and Applied Mathematics, vol. 45, no. 5, pp. 485-560, 1992.

[13] P. Mavridis and G. Papaioannou, "The compact YCoCg frame Buffer," Journal of Computer Graphics Techniques, vol. 1, no. 1, pp. 19-35, 2012.

[14] D. Van Rijsselbergen, "YCoCg(-R) color space conversion on the GPU," 6th FirW PhD Symposium, Faculty of Engineering, Ghent University, paper no. 102, 2005.

[15] H. Malvar and G. Sullivan, "YCoCg-R: A Color space with RGB reversibility and low dynamic range," Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Document No. JVTI014r3, 2003.

[16] RFC 1951, DEFLATE Compressed Data Format Specification, 1996.

[17] W3C recommendation, Portable Network Graphics (PNG) Specification (Second Edition), 2003.