Chapter 7

A Real-Time Video Encoding Scheme Based on the Contourlet Transform

Stamos Katsigiannis, Georgios Papaioannou and Dimitris Maroulis

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51735

1. Introduction

Real-time video communication over the internet and other heterogeneous IP networks has become a significant part of modern communications, underlining the need for highly efficient video coding algorithms. The most desirable characteristic of such an algorithm would be the ability to maintain satisfactory visual quality while achieving good compression. Additional advantageous characteristics would be low computational complexity and real-time performance, allowing the algorithm to be used in a wide variety of less powerful computers. Transmission of video over the network would benefit by the ability to adapt to the network's end-to-end bandwidth and transmitter/receiver resources, as well as by resistance to packet losses that might occur. Additionally, scalability and resistance to noise would be highly advantageous characteristics for a modern video compression algorithm. Most state of the art video compression techniques like the H.264, DivX/Xvid, MPEG2 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 they depend on multipass statistical and structural analysis of the whole video content, which cannot happen in cases of live video stream generation as in the case of video-conferencing.

In this chapter, a more elaborate analysis of a novel algorithm for high-quality real-time video encoding, originally proposed in [1], is presented. The algorithm is designed for content obtained from low resolution sources like web cameras, surveillance cameras, etc. Critical to the efficiency of video encoding algorithm design is the selection of a suitable image representation method. Texture representation methods proposed in the literature that utilize the Fourier transform, the Discrete Cosine transform, the Wavelet transform as well as other frequency domain methods have been extensively used for image and video encoding. Never-

theless, these methods have some limitations that have been partially addressed by the Contourlet Transform (CT) [2], which our video encoding algorithm is based on. The Contourlet Transform offers multiscale and directional decomposition, providing anisotropy and directionality, features missing from traditional transforms like the Discrete Wavelet Transform [2]. In recent years, the Contourlet Transform has been successfully utilised in a variety of texture analysis applications, including synthetic aperture radar (SAR) [3], medical and natural image classification [4], image denoising [5], despeckling of images, image compression, etc. By harnessing the computational power offered by modern graphics processing units (GPUs), a gpu-based contourlet transform is able to provide an image representation method with advantageous characteristics, while maintaining a fast performance.

The rest of this chapter is organised in four sections. First, some background knowledge and information needed for better understanding the algorithm is presented in section 2. Then, the aforementioned video encoding algorithm is presented in section 3, whereas an experimental study for the evaluation of the algorithm is provided in section 4. Conclusions and future perspectives of this work are presented in section 5.

2. Background

2.1. The Contourlet Transform

The Contourlet Transform (CT) is a directional multiscale image representation scheme proposed by Do and Vetterli, which is effective in representing smooth contours in different directions of an image, thus providing directionality and anisotropy [2]. The method utilizes a double filter bank in which, first the Laplacian Pyramid (LP) [6] detects the point discontinuities of the image and then the Directional Filter Bank (DFB) [7] links those point discontinuities into linear structures. The LP provides a way to obtain multiscale decomposition. In each LP level, a downsampled lowpass version of the original image and a more detailed image with the supplementary high frequencies containing the point discontinuities are obtained. This scheme can be iterated continuously in the lowpass image and is restricted only by the size of the original image due to the downsampling. The DFB is a 2D directional filter bank that can achieve perfect reconstruction, which is an important characteristic for image and video encoding applications. The simplified DFB used for the contourlet transform consists of two stages and leads to 2^l subbands with wedge-shaped frequency partitioning [8], with l being the level of decomposition. The first stage of the DFB is a two-channel quincunx filter bank [9] with fan filters that divides the 2D spectrum into vertical and horizontal directions, while the second stage is a shearing operator that just reorders the samples. By adding a 45 degrees shearing operator and its inverse before and after a two-channel filter bank, a different directional frequency partition is obtained (diagonal directions), while maintaining the ability to perfectly reconstruct the original image, since the sampling locations coincide with the (integer) pixel grid.

The combination of the LP and the DFB is a double filter bank named Pyramidal Directional Filter Bank (PDFB). In order to capture the directional information, bandpass images from the LP decomposition are fed into a DFB. This scheme can be repeated on the coarser image levels. The

http://dx.doi.org/10.5772/51735

combined result is the contourlet filter bank, which is a double iterated filter bank that decomposes images into directional subbands at multiple scales. The contourlet coefficients have a similarity with wavelet coefficients since most of them are almost zero and only few of them, located near the edge of the objects, have large magnitudes [10]. In the presented algorithm, the Cohen and Daubechies 9-7 filters [11] have been utilized for the Laplacian Pyramid. For the Directional Filter Bank, these filters were mapped into their corresponding 2D filters using the McClellan transform as proposed by Do and Vetterli in [2]. It must be noted that these filters are not considered as optimal. The creation of optimal filters for the contourlet filter bank remains an open research topic. An outline of the Contourlet Transform is presented on Figure 1, while an example of decomposition is shown on Figure 2.

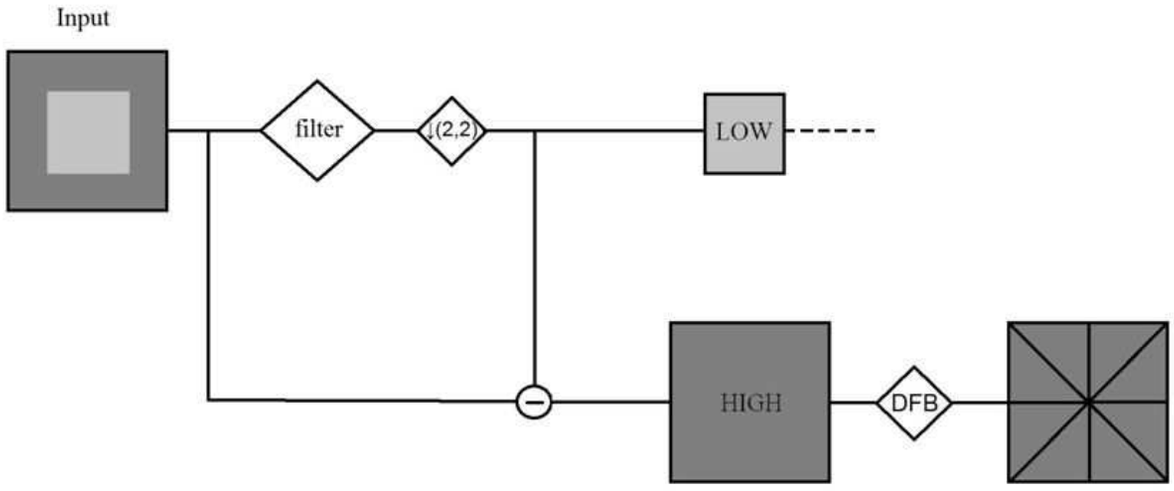

Figure 1. The Contourlet Filter Bank.

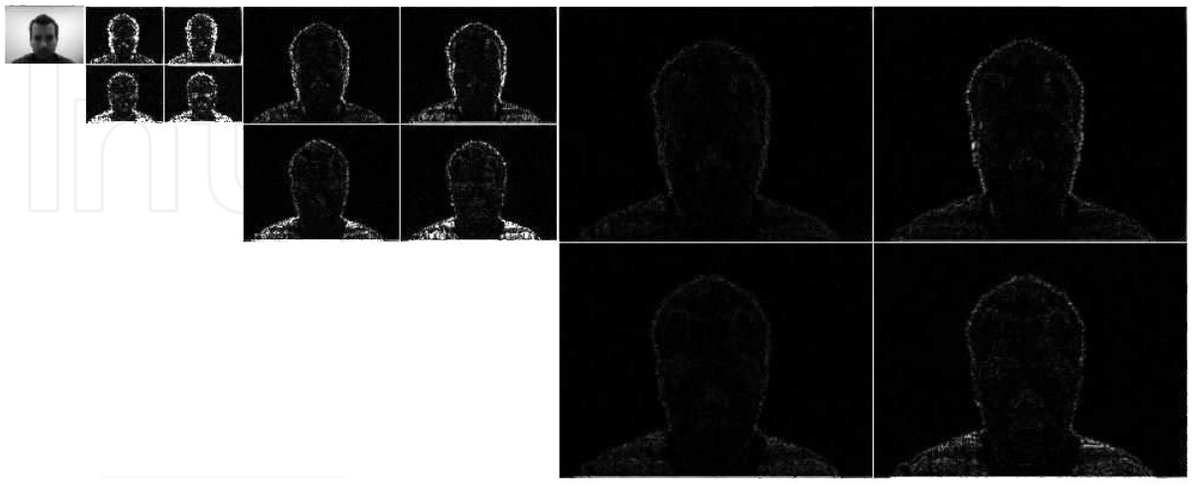

Figure 2. Example of contourlet transform decomposition of a greyscale image. Three levels of decomposition with the Laplacian Pyramid were applied, each then decomposed into four directional subbands using the Directional Filter Bank.

2.2. GPU-based contourlet transform

By analysing the structure of the contourlet transform, it is evident that its most computationally intensive part is the calculation of all the 2D convolutions needed for complete decomposition or reconstruction. Calculating the convolutions on the CPU using the 2D convolution definition is not feasible for real-time applications since performance suffers significantly due to the computational complexity. Utilizing the DFT or the FFT in order to achieve better performance provides significantly faster implementations but still fails to achieve satisfactory real-time performance, especially in mobile platforms such as laptops and tablet PCs. The benefits of the FFT for the calculation of 2D convolution can only be fully exploited by an architecture supporting parallel computations. Modern personal computers are commonly equipped with powerful graphics processors (GPUs), which in the case of live video capture from web or surveillance cameras are underutilized. Intensive, repetitive computations that can be computed in parallel can be accelerated by harnessing this "dormant" computational power. General purpose computing on graphics processing units (GPGPU) is the set of techniques that use a GPU, which is otherwise specialized in handling computations for the display of computer graphics, in order to perform computations traditionally handled by a CPU. The highly parallel structure of GPUs makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data can be done in parallel.

For the GPU implementation of the contourlet transform, the NVIDIA Compute Unified Device Architecture (CUDA) has been selected due to the extensive capabilities and specialized API it offers. CUDA is a general purpose parallel computing architecture that allows the parallel compute engine in NVIDIA GPUs to be used in order to solve complex computational problems that are outside the scope of graphics algorithms. In order to compute the contourlet transform, first the image and the filters are transferred from the main memory to the GPU dedicated memory. Then, the contourlet transform of the image is calculated by migrating all the calculations on the GPU in order to reduce the unnecessary transfers to and from the main memory that introduce delay to the computations. The 2D convolutions required are calculated by means of the FFT. After calculating the contourlet transform of the image, the output is transferred back to the main memory and the GPU memory is freed. Considering that this implementation will be used for video encoding, the filters are loaded once at the GPU memory since they will not change from frame to frame. In order to evaluate the performance of this approach, various implementations of the contourlet transform were developed, both for the CPU and the GPU. These implementations were based on the FFT (frequency domain) and the 2D convolution definition (spatial domain). Except for the basic GPU implementation using spatial domain convolution, other out-of-core implementations were developed, based on the 2D convolution definition and utilizing memory management schemes in order to support larger frames when the GPU memory is not sufficient [12]. The GPU implementation based on the FFT outperformed all the aforementioned implementations in our tests and was therefore the method of choice for our video encoder.

http://dx.doi.org/10.5772/51735

2.3. The YCoCg colour space

Inspired by recent work on real-time RGB frame buffer compression using chrominance subsampling based on the YCoCg colour transform [13], we investigated the use of these techniques in conjunction with the contourlet transform to efficiently encode colour video frames.

The human visual system is significantly more sensitive to variations of luminance compared to variations of chrominance. Encoding the luminance channel of an image with higher accuracy than the chrominance channels provides a simple low complexity compression scheme, while maintaining satisfactory visual quality. Various image and video compression algorithms take advantage of this fact in order to achieve increased efficiency. First introduced in H.264 compression, the RGB to YCoCg transform decomposes a colour image into luminance (Y), orange chrominance (Co) and green chrominance (Cg) components and has been shown to exhibit better decorrelation properties than YCbCr and similar transforms [14]. It was developed primarily to address some limitations of the different YCbCr colour spaces [15]. The transform and its reverse are calculated by the following equations:

$$Y = R/4 + G/2 + B/4 \tag{1}$$

$$Co = R/2 - B/2 \tag{2}$$

$$Cg = -R/4 + G/2 - B/4 \tag{3}$$

$$R = Y + Co - Cg \tag{4}$$

$$G = Y + Cg \tag{5}$$

$$B = Y - Co - Cg \tag{6}$$

Image set	Number of images	Average PSNR (dB)
Kodak	23	59.27
Canon	18	59.05
Outdoor scene images	963	58.87

Table 1. Average PSNR obtained for each image set after transforming from RGB to YCoCg and back using the same precision for the RGB and YCoCg components.

In order for the reverse transform to be perfect and to avoid rounding errors, the Co and Cg components should be stored with higher precision than the RGB components. Experiments using 23 images from the Kodak image set and 18 images from the Canon image set, all obtained from [16], as well as 963 outdoor scene images obtained from [17], 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 for all the image sets, as shown in Table 1. This

loss of quality cannot be perceived by the human visual system, resulting to no visible alteration of the image. Nevertheless, it indicates the highest quality possible when used for image compression.

3. The presented algorithm

Listings 1 and 2 depict the presented algorithm for encoding and decoding respectively. Input frames are considered to be in the RGB format. The first step of the algorithm is the conversion from RGB to YCoCg colour space for further manipulation of the luminance and chrominance channels. The luminance channel is decomposed using the contourlet transform, while chrominance channels are subsampled by a user-defined factor *N*. The levels and filters for contourlet transform decomposition are also defined by the user. From the contourlet coefficients obtained by decomposing the luminance channel, only a user-specified percentage of the most significant ones are retained. Then, the precision allocated for storing the contourlet coefficients is reduced. All computations up to this stage are performed on the GPU, avoiding unnecessary memory transfers from the main memory to the GPU memory and vice versa. After reducing the precision of the retained contourlet coefficients of the luminance channel, the directional subbands are encoded using a run length encoding scheme that encodes only zero valued elements. The large sequences of zero-valued contourlet coefficients that occur after the insignificant coefficient truncation make run length encoding ideal for their encoding.

The encoding algorithm
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: Keep the M% most significant CT coefficients
7: Round the CT coefficients to the *n*-th decimal
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: Adjust precision of all components
14: IF frame is NOT the last frame
15: GOTO Start
16: END IF
17: Finish

Listing 1. Steps of the encoding algorithm. Highlighted steps refer to calculations performed on the GPU, while the other steps refer to calculations performed on the CPU.

The algorithm divides the video frames into two categories; keyframes and internal frames. Keyframes are frames that are encoded using the steps described in the previous paragraph and internal frames are the frames between two key frames. The interval between two keyframes is a user defined parameter. At the step before the run-length encoding, 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 processed on the GPU while all the remaining steps of the algorithm are performed on the CPU unless otherwise stated. Then, run length encoding is applied to the chromatic channels, the low frequency contourlet component of the luminance channel, as well as the directional subbands of the luminance channel. It must be noted that steps that are executed on the CPU are inherently serial and cannot be efficiently mapped to a GPU.

The last stage of the algorithm consists of the selection of the optimal precision for each video component. The user can select between lossless or lossy change of precision, directly affecting the output's visual quality.

The decoding algorithm
1: Start
2: Input encoded frame
3: IF the frame is a keyframe
4: Decode the run-length encoded directional subbands of Y
5: Keep keyframe in memory and discard old keyframe
6: ELSE IF the frame is an internal frame
7: Decode the run-length encoded Co, Cg, lowpass CT component of Y
8: Decode the run-length encoded directional subbands of Y
9: Calculate the frame as the sum of the frame and the previous keyframe
10: END IF
11: Upsample Co and Cg by N
12: Reconstruct Y
13: Convert to RGB
14: IF frame is NOT the last frame
15: GOTO Start
16: END IF
17: Finish

Listing 2. Steps of the decoding algorithm. Highlighted steps refer to calculations performed on the GPU, while the other steps refer to calculations performed on the CPU.

3.1. Chrominance channel subsampling

Exploiting the fact that the human visual system is relatively insensitive to chrominance variations, in order to achieve compression, the chrominance channels Co and Cg are subsampled by a user-defined factor N that directly affects the output's visual quality and the compression achieved. The chrominance channels are stored in lower resolution,

thus providing compression. For the reconstruction of the chrominance channels at the decoding stage, the missing chrominance values are replaced with the nearest available subsampled chrominance values. This approach is simple and naïve but has been selected due to the significantly smaller number of (costly) memory fetches and minimal computation cost, compared to other methods like bilinear interpolation. Utilizing the nearest neighbour reconstruction approach can introduce artifacts in the form of mosaic patterns in regions with strong chrominance transitions depending on the subsampling factor. In order to address this problem, given adequate computational resources, the receiver can choose to use the bilinear interpolation approach. Figure 3 shows an example of subsampling the Co and Cg chrominance channels by various factors, while using the nearest neighbour and the bilinear interpolation approach for reconstruction. Only a small, magnified part of the "baboon" image used is shown for clarity. As demonstrated in Figure 3, subsampling by a factor of 2 or 4 does not have a drastic effect on visual quality. Further subsampling leads to visible artifacts indicating the need for an optimal trade-off between quality and compression.

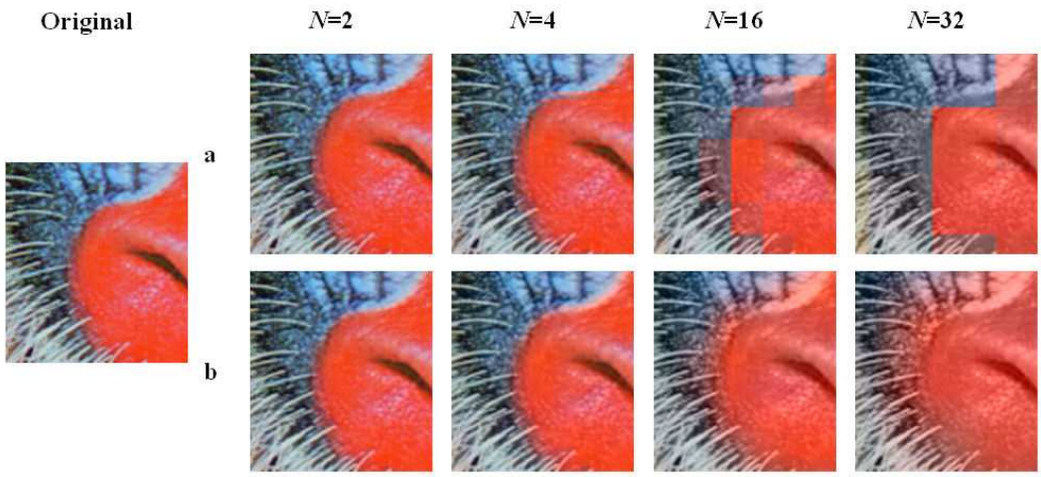

Figure 3. Example of chroma subsampling by factor *N* of the Co and Cg channels of the "baboon" image. Row (a) depicts images reconstructed using the nearest neighbour method, while (b) those reconstructed using bilinear interpolation.

3.2. Contourlet Transform decomposition of luminance channel and quality selection

The luminance channel of the frame is decomposed using the contourlet transform. Decomposition levels, as well as the filters used are user-defined and directly affect the quality of the output. Decomposition at multiple scales offers better compression while providing scalability, i.e. multiple resolutions inside the same video stream. This characteristic 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 without re-encoding the video frames at the source, by just dropping the encoded information referring to higher resolution than needed.

In order to achieve compression, after the decomposition of the luminance channel with the contourlet transform, a user-defined amount of the contourlet coefficients from the direc-

http://dx.doi.org/10.5772/51735

tional subbands are dropped by means of keeping only the most significant coefficients. The amount of coefficients dropped drastically affects the output's visual quality as well as the compression ratio. Contourlet coefficients with large magnitudes are considered more significant than coefficients with smaller magnitudes. Exploiting this fact, a common method for selecting the most significant contourlet coefficients is to keep the M most significant coefficients, or respective percentage, while dropping all the others [2] (coefficient truncation). This procedure leads to a large number of zero-valued sequences inside the elements of the directional subbands, a fact exploited by using run length encoding in order to achieve even higher compression. Considering the values and the distribution of contourlet coefficients at the directional subbands, only the zero-valued coefficients are run length encoded along the horizontal direction. Compression gained by run length encoding of all the different values is minimum and does not justify the increased computational cost. It is worth mentioning that dropping all the contourlet coefficients is similar to lowering the luminance channel's resolution while applying a lowpass filter and then, at the decoding stage, upscaling it without reincorporating the high frequency content.

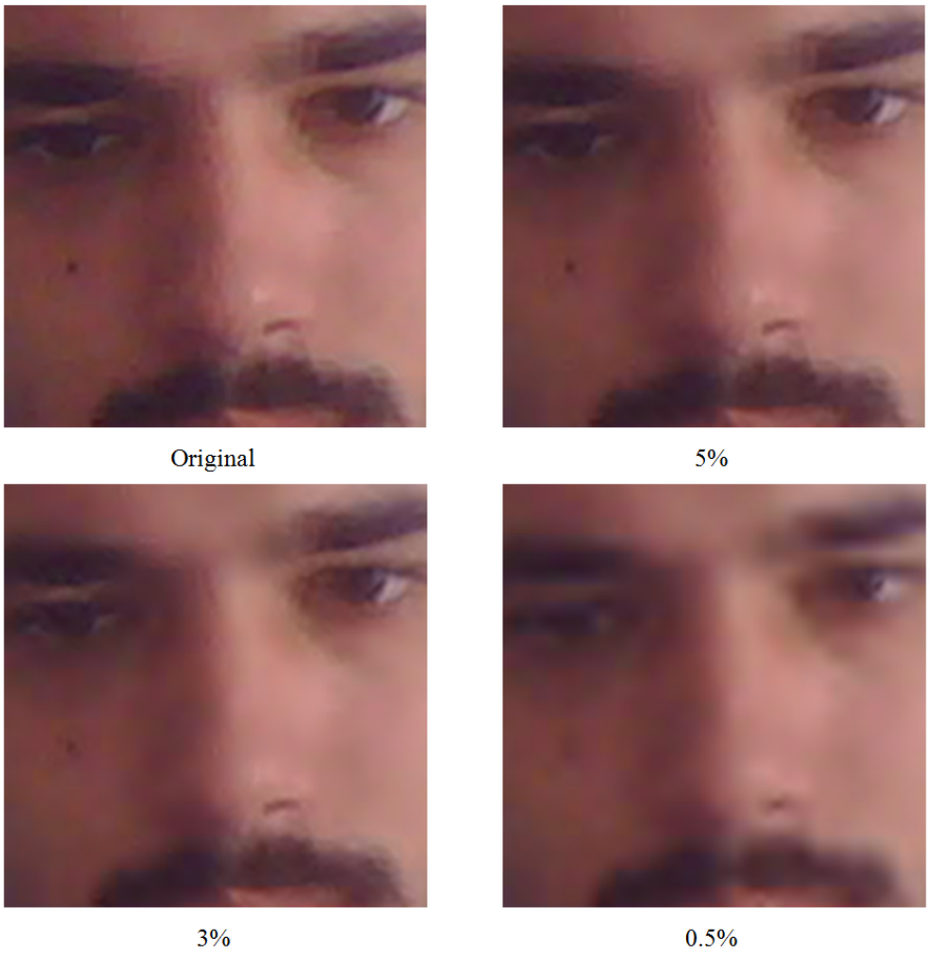

Figure 4. Example of smoothing due to the dropping of contourlet coefficients. The caption indicates the percentage of the contourlet coefficients retained. Images are cropped and scaled to 200% of their original size.

Keeping only the most significant contourlet coefficients also provides a means to suppress the noise induced by low-quality sensors usually encountered in web-cameras. Random noise is largely unstructured and therefore not likely to generate significant contourlet coefficients [2]. As a result, keeping only the most significant contourlet coefficients provides enhanced visual quality, which is a highly desirable characteristic since no additional filtering of the video stream is required in order to reduce the noise level. On Figure 4, an example of smoothing due to the dropping of contourlet coefficients is shown. Mosaicing artifacts and noise introduced due to the low quality of the web camera's sensor are suppressed and replaced by a fuzzier texture, resulting in a smoother and more perceptually acceptable image.

At the contourlet transform decomposition stage, 32 bit single precision floating point elements are used in order to avoid rounding errors and precision loss. Experiments with the precision allocated for the contourlet coefficients showed that the contourlet transform exhibits resistance to quality loss due to arithmetic precision reduction. This fact is exploited in order to achieve better compression by reducing the precision of the contourlet coefficients through rounding to a specific decimal point. Visual quality is not affected at all when only one decimal or more are kept. Rounding to the integer provides a PSNR of more than 60 dB when only the directional subbands' coefficients are rounded. Additionally, also rounding the low pass content provides a PSNR of more than 55 dB. In both cases, the loss of quality cannot be perceived by the human visual system and is considered as insignificant. For these experiments, the images were first transformed into the YCoCg colour space. Then the luminance channel was decomposed using the contourlet transform and the contourlet coefficients were rounded. No alteration was done to the chrominance channels. After the manipulation of the contourlet coefficients, the luminance channel was reconstructed and the image was transformed back into the RGB colour space.

3.3. Frame types

As mentioned before, frames are divided into keyframes and internal frames, with an internal frame being the difference between the current frame and the respective keyframe. Consecutive frames tend to have small variations, with many identical regions. This fact can be exploited by calculating the difference between a frame and the keyframe. This procedure provides components with large sequences of zero values leading to improved compression through the run length encoding stage. Especially in the case of video-conferencing or surveillance video, the background tends to be static, with slight or no variations at all. The occurrence of static background leads to many parts of the consecutive frames to be identical. As a result, calculating the difference of each frame from its respective keyframe provides large sequences of zero values leading to improved compression when run length encoding is applied. Run length encoding of the difference of contourlet-transformed images is even more efficient, since static noise is drastically suppressed by the coefficient truncation. Experiments showed that the optimal compression is achieved for a relatively small interval between keyframes, in the region of 5-7 internal frames, providing small groups of pictures (GOP) that depend to a keyframe. This characteristic makes the algorithm more resistant to packet loses when transmitting over a network. In the case of a scene change, consecutive frames drastically differ from each other and

http://dx.doi.org/10.5772/51735

the compression achieved for the internal frames until the next keyframe is similar to that of a keyframe. If this scenario occurs, having small intervals between consecutive keyframes reduces the number of non optimally encoded frames. Nevertheless, in cases where the video is expected to be mostly static, like surveillance video for example, a larger interval between keyframes will provide considerably better compression.

3.4. Other supported colour spaces

Except for the YCoCg colour space, the algorithm supports the YCbCr and Greyscale colour spaces without the need to alter its core functionality. The process of encoding greyscale videos consists of handling the video as a colour video with only the luminance channel. All the steps of the algorithm are calculated except for those referring to the chromatic channels. On the other hand, due to the similarity of the YCbCr colour space with the YCoCg colour space the algorithm remains the same. The only difference is the RGB-to-YCbCr conversion at the encoder and the YCbCr-to-RGB conversion at the decoder. The luminance channel is identically handled as in the YCoCg-based algorithm, and the same holds for the replacement of CoCg channels with the CbCr ones. Nevertheless, the CbCr channels have a different range of values compared to CoCg channels. As a consequence, the optimal precision for the CbCr channels differs from that of the CoCg channels and has to be taken into consideration.

4. Quality and performance analysis

For evaluating the presented algorithm, two 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 a typical video-conference sequence with static background showing the upper part of the human body and containing some motion, and a surveillance video with almost no motion depicting the entrance of a building.

The captured videos were encoded using the YCoCg, YCbCr and Greyscale colour spaces. The chrominance channels of the colour videos were subsampled by a factor of 4 and the video stream contained two resolutions: the original VGA (640x480) as well as the lower QVGA (320x240). The method utilized for the reconstruction of the chrominance channels was the nearest neighbour method. The percentage of the most significant contourlet coefficients of the luminance channel retained was adjusted for each encoded video, providing results of various quality and compression levels. Furthermore, at each scale, the luminance channel's high frequency content was decomposed into four directional subbands. In order to test the algorithm using the YCbCr colour space, the RGB to YCbCr conversion formula for SDTV found in [18] was utilised. For the Greyscale colour space, the aforementioned videos were converted from RGB to greyscale using the standard NTSC conversion formula [18] that is used for calculating the effective luminance of a pixel:

$$Y(i,\ j) = 0.2989 \cdot R(i,\ j) + 0.5870 \cdot G(i,\ j) + 0.1140 \cdot B(i,\ j) \tag{7}$$

The sample videos were encoded using a variety of parameters. The mean PSNR value for each video was calculated based on a set of different percentages of contourlet coefficients to be retained. The compression ratios achieved when using the scheme that incorporates both key frames and internal frames and when compressing all the frames as keyframes were also calculated. The interval between the key frames was set to five frames for the video-conference sample video and to twenty frames for the surveillance video. Detailed results are shown on Tables 2, 3 and 4 while sample frames of the encoded videos utilizing the YCoCg, YCbCr and Greyscale colour space for a set of settings are shown on Figures 5-10.

Examining the compression ratios achieved, it is shown that utilizing the keyframe and internal frame scheme outperforms the naive method of encoding all the frames the same way, as expected. However, the selection of an efficient entropy encoding algorithm that will further enhance the compression ability of our algorithm is still an open issue. Another interesting observation is that the contourlet transform exhibits substantial resistance to the loss of contourlet coefficients. Even when only 5% of its original coefficients are retained, the visual quality of the image is not seriously affected. This fact underlines the efficiency of the contourlet transform in approximating natural images using a small number of descriptors and justifies its utilization in this algorithm. The slightly lower PSNR achieved for the surveillance video sample can be explained due to the higher complexity of the scene compared to the video conference sample. More complex scenes contain higher frequency content, a portion of which is then discarded by dropping the contourlet coefficients.

(a) Video conference sample				**(b) Video surveillance sample**			
	PSNR (dB)				PSNR (dB)		
Contourlet coefficients retained (%)	YCoCg	YCbCr	Greyscale	Contourlet coefficients retained (%)	YCoCg	YCbCr	Greyscale
10	45.11	44.77	52.04	10	44.18	44.03	50.11
5	44.53	44.29	49.71	5	43.54	43.45	47.88
3	43.88	43.70	47.72	3	42.96	42.89	46.33
1	42.30	42.23	44.28	1	41.57	41.50	43.45
0.5	41.62	41.56	43.10	0.5	40.80	40.76	42.17
0.2	41.30	41.25	42.60	0.2	40.17	40.14	41.21
0	39.15	39.13	39.82	0	39.59	39.55	40.39

Table 2. PSNRs achieved for the (a) video conference and (b) video surveillance samples, retaining various percentages of contourlet coefficients and utilizing the YCoCg. YCbCr and Greyscale colour spaces.

http://dx.doi.org/10.5772/51735

Video conference sample						
Contourlet coefficients retained (%)	Compression ratio					
	YCoCg		YCbCr		Greyscale	
	Only keyframes	Keyframes & Internal frames	Only keyframes	Keyframes & Internal frames	Only keyframes	Keyframes & Internal frames
10	4.96:1	11.06:1	4.93:1	12.05:1	2.09:1	3.49:1
5	6.44:1	14.39:1	6.44:1	16.31:1	2.94:1	4.44:1
3	7.36:1	16.39:1	7.37:1	18.98:1	3.56:1	5.02:1
1	8.71:1	19.46:1	8.70:1	23.15:1	4.57:1	5.85:1
0.5	9.07:1	20.24:1	9.06:1	24.33:1	4.87:1	6.07:1
0.2	9.22:1	20.62:1	9.21:1	24.81:1	5.00:1	6.17:1
0	11.71:1	25.84:1	11.71:1	32.89:1	7.65:1	7.53:1

Table 3. Compression ratios achieved for the video conference sample, retaining various percentages of contourlet coefficients and utilizing the YCoCg. YCbCr and Greyscale colour spaces.

Video surveillance sample						
Contourlet coefficients retained (%)	Compression ratio					
	YCoCg		YCbCr		Greyscale	
	Only keyframes	Keyframes & Internal frames	Only keyframes	Keyframes & Internal frames	Only keyframes	Keyframes & Internal frames
10	4.35:1	21.55:1	4.35:1	22.73:1	1.78:1	7.26:1
5	5.89:1	28.74:1	5.92:1	31.06:1	2.63:1	9.68:1
3	6.85:1	32.89:1	6.89:1	35.97:1	3.23:1	11.09:1
1	8.14:1	38.02:1	8.18:1	42.19:1	4.15:1	12.79:1
0.5	8.58:1	39.53:1	8.61:1	44.05:1	4.48:1	13.30:1
0.2	8.89:1	40.65:1	8.92:1	45.45:1	4.74:1	13.70:1
0	11.71:1	49.26:1	11.71:1	56.50:1	7.65:1	16.58:1

Table 4. Compression ratios achieved for the video surveillance sample, retaining various percentages of contourlet coefficients and utilizing the YCoCg. YCbCr and Greyscale colour spaces.

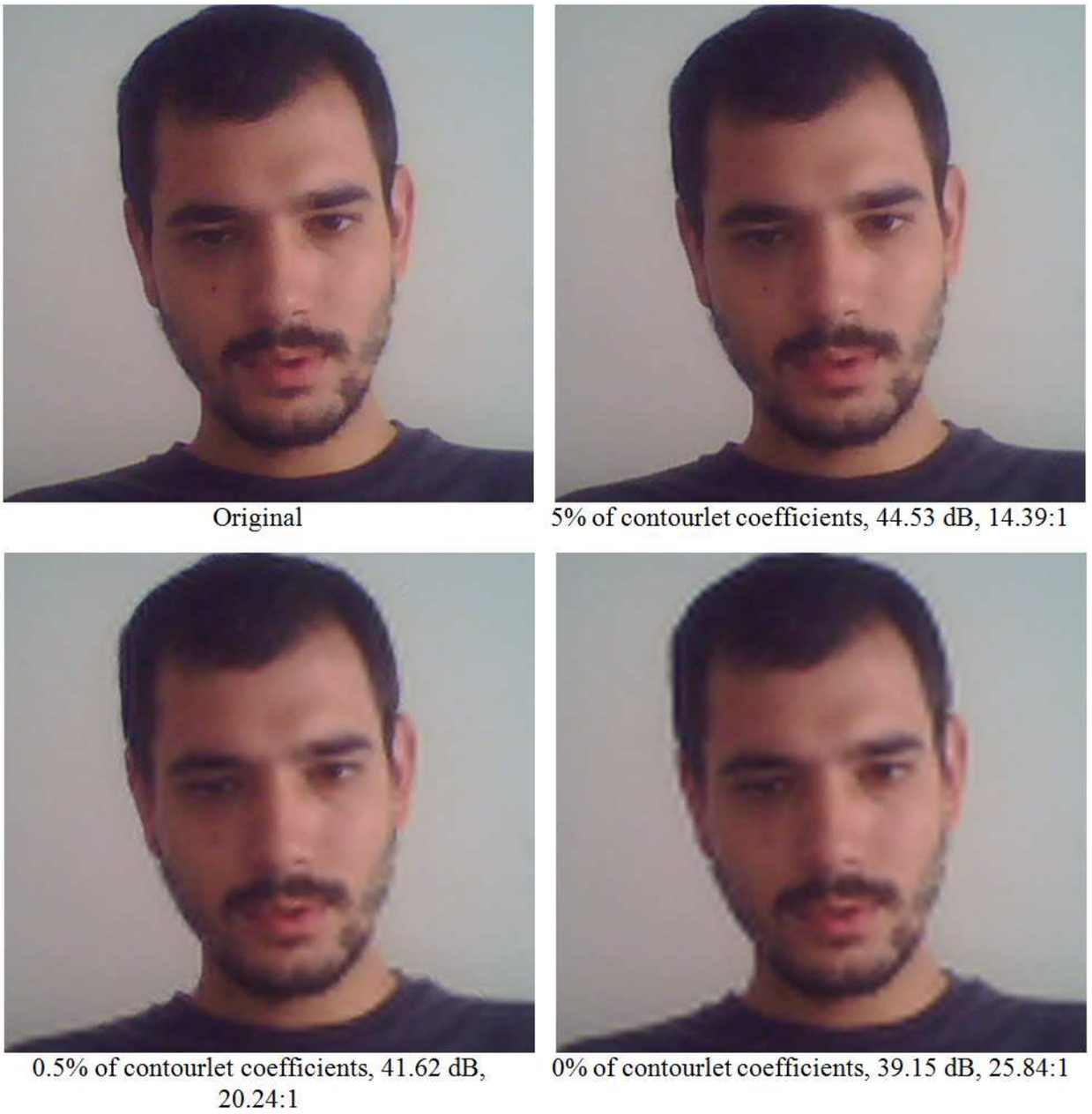

Figure 5. Sample frame of the encoded video-conference video for each setting using the *YCoCg* colour space. The frame has been resized and cropped to fit the figure.

http://dx.doi.org/10.5772/51735

Figure 6. Sample frame of the encoded video-conference video for each setting using the *YCbCr* colour space. The frame has been resized and cropped to fit the figure.

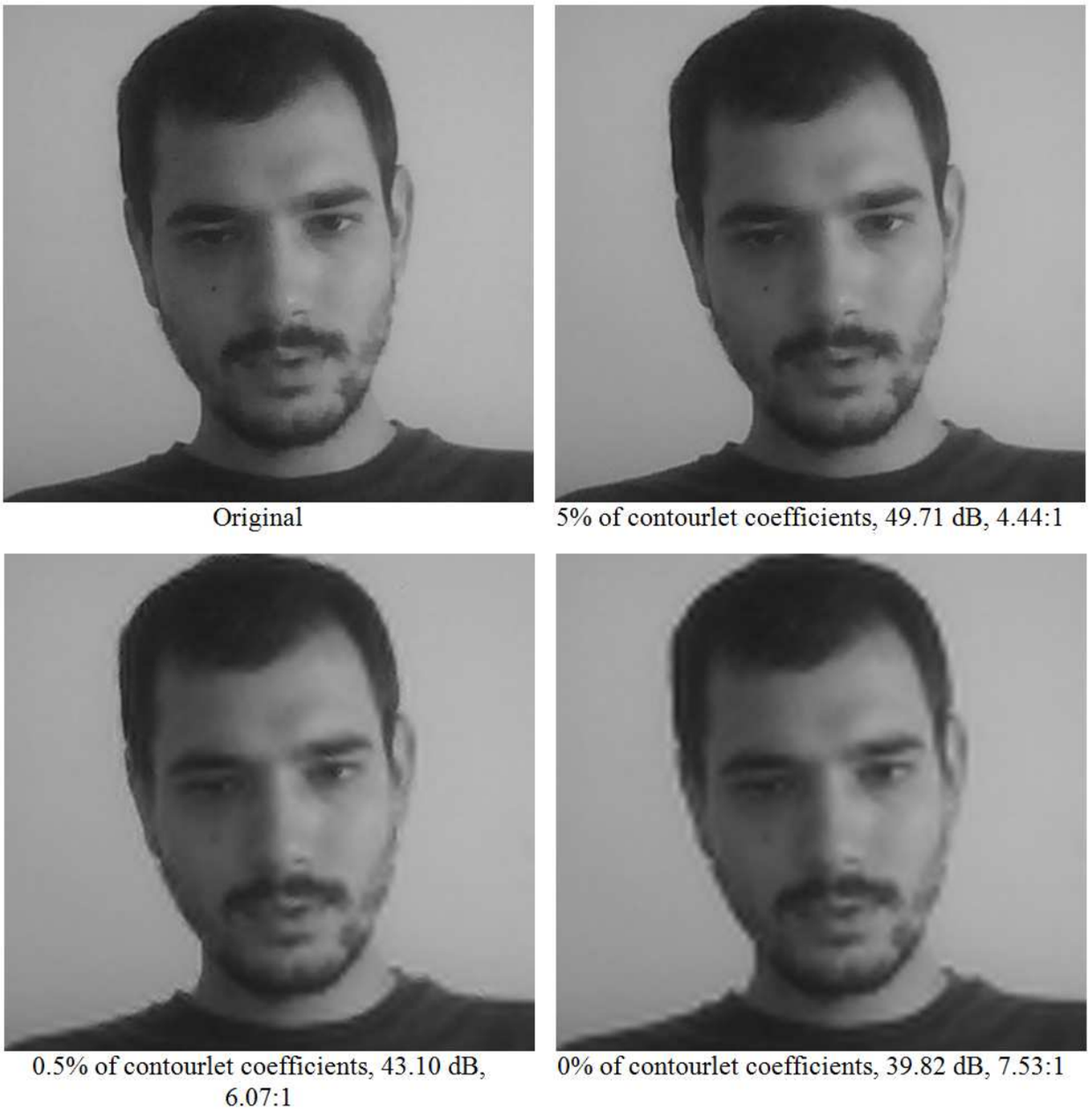

Figure 7. Sample frame of the encoded video-conference video for each setting using the *Greyscale* colour space. The frame has been resized and cropped to fit the figure.

http://dx.doi.org/10.5772/51735

Figure 8. Sample frame of the encoded video surveillance video for each setting using the *YCoCg* colour space. The frame has been resized and cropped to fit the figure.

Figure 9. Sample frame of the encoded video surveillance video for each setting using the *YCbCr* colour space. The frame has been resized and cropped to fit the figure.

http://dx.doi.org/10.5772/51735

Figure 10. Sample frame of the encoded video surveillance video for each setting using the *Greyscale* colour space. The frame has been resized and cropped to fit the figure.

Considering the YCoCg and the YCbCr colour spaces, for the two video samples tested, it is shown on Tables 2-4 and Figures 11 and 12 that the YCoCg colour space achieves slightly better visual quality (higher PSNR), while the YCbCr colour space provides better compression (higher compression ratio). The Greyscale examples cannot be directly compared to the colour samples since the calculated PSNR characterizes the original and encoded greyscale samples. Nevertheless, it is clear that in the case of Greyscale colour space, compression suffers greatly compared to the other colour spaces.

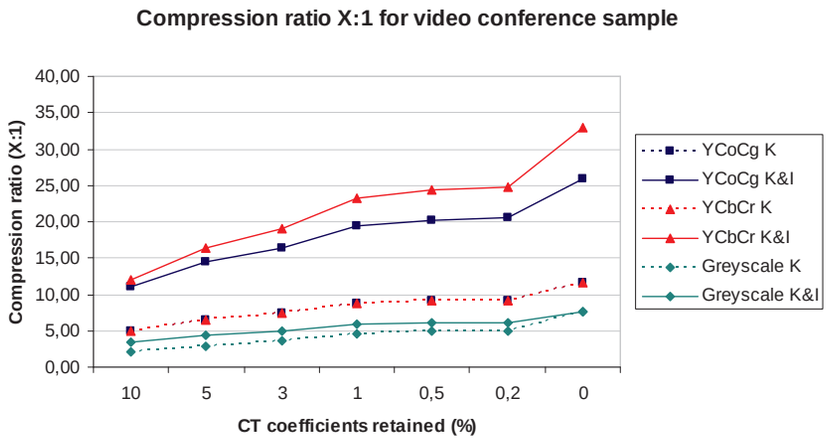

Figure 11. Compression ratio vs percentage of contourlet coefficients retained diagram, for the video conference sample, utilizing the YCoCg. YCbCr and Greyscale colour spaces. K refers to using only keyframes, while K&I refers to using both keyframes and internal frames.

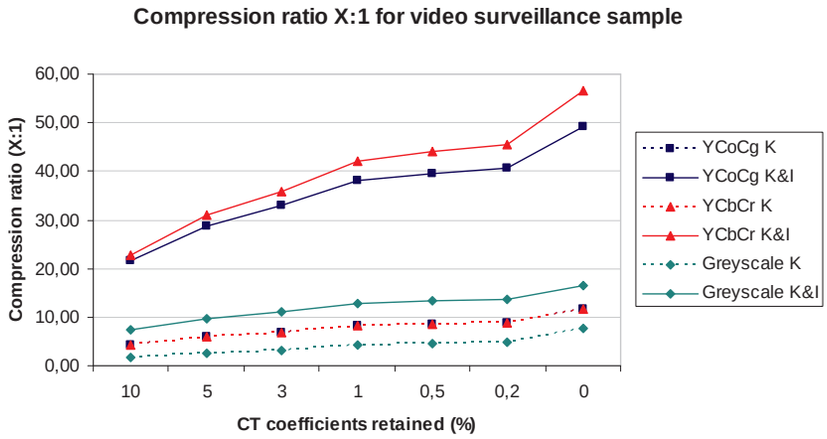

Figure 12. Compression ratio vs percentage of contourlet coefficients retained diagram, for the video surveillance sample, utilizing the YCoCg. YCbCr and Greyscale colour spaces. K refers to using only keyframes, while K&I refers to using both keyframes and internal frames.

http://dx.doi.org/10.5772/51735

Average execution times for the basic operations of the encoding and decoding algorithm for a frame of the video conference sample are presented on Table 5. Parameters were kept the same as in the previous examples and the computer utilised for the performance tests was equipped with an Intel Core i3 CPU, 4 GB of memory and a NVIDIA GeForce 430 graphics card with 1 GB of memory.

Operation	Time (ms)
Transfer of RGB frame to GPU memory	1.385
Transfer of encoded frame to main memory	1.050
Conversion from RGB to YCoCg	1.067
Conversion from YCoCg to RGB	0.402
Contourlet transform decomposition	59.040
Contourlet transform reconstruction	57.102
Run-length encoding of directional subbands	2.424
Run-length decoding of directional subbands	7.008
Contourlet coefficients dropping	0.492

Table 5. Average execution times (in milliseconds) for the basic operations of the algorithm for a 640x480 video frame. The chrominance channels were subsampled by a factor of 4 and the video stream contained the original VGA (640x480) as well as the lower QVGA (320x240) resolution.

5. Conclusions

In this chapter, a low complexity algorithm for real-time video encoding based on the contourlet transform and optimized for video conferencing applications and surveillance cameras has been presented and evaluated. The algorithm provides a scalable video compression scheme ideal for video conferencing content as 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. A video stream can contain various resolutions avoiding the need for reencoding at the source. The receiver can select the desired quality by dropping the components referring to higher quality than needed. Furthermore, the algorithm has the inherent ability to suppress the noise induced by low-quality sensors, without the need of an extra denoising or image enhancement stage, due to the manipulation of the structural characteristics of the video through the rejection of insignificant contourlet transform coefficients. In the case of long recordings for surveillance systems, where higher compression is needed, the visual quality degradation is much more eye-friendly than with other well established video compression methods, as it introduces fuzziness and blurring instead of artificial block artifacts, providing smoother images and facilitating image rectification/recognition procedures. Additionally, due to the relatively small GOPs, the algorithm is more resistant to frame losses that can occur during transmis-

sion over IP networks. Another advantageous characteristic of the presented algorithm is that its most computationally intensive parts are calculated on the GPU. The utilization of the usually "dormant" GPU computational power lets the CPU to be utilized for other tasks, further enhancing the multitasking capacity of the system and enabling the users to take full advantage of their computational capabilities. The experimental evaluation of the presented algorithm provided promising results. Nevertheless, in order to compete for compression efficiency with state of the art video compression algorithms, a highly efficient entropy encoding scheme has to be incorporated to the algorithm. Modern entropy encoding methods tend to be complex and computationally intensive. As a result, the optimal trade-off between compression rates and complexity has to be decided in order to retain the low complexity and real time characteristics of our algorithm.

Acknowledgements

The authors would like to thank Pavlos Mavridis for his fruitful advice concerning the YCoCg colour space.

Author details

Stamos Katsigiannis[1*], Georgios Papaioannou[2] and Dimitris Maroulis[1]

*Address all correspondence to: stamos@di.uoa.gr

1 Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece

2 Department of Informatics, Athens University of Economics and Business, Athens, Greece

References

[1] Katsigiannis, S., Papaioannou, G., & Maroulis, D. (2012). A contourlet transform based algorithm for real-time video encoding. *Proceedings of SPIE*, 8437, 843704, doi: 10.1117/12.924327.

[2] , M. N., & Vetterli, M. (2005). The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14(12), 2091-2106, doi: 10.1109/TIP.2005.859376.

[3] Liu, Z. (2008). Minimum Distance Texture Classification of SAR Images in Contourlet Domain. *Proceedings of the 2008 International Conference on Computer Science and Software Engineering, CSSE*, doi: 10.1109/IASP.2010.5476106.

[4] Katsigiannis, S., Keramidas, E., & Maroulis, D. (2010). A Contourlet Transform Feature Extraction Scheme for Ultrasound Thyroid Texture Classification. *Engineering Intelligent Systems*, 18, 3/4.

[5] Liu, Z., & Xu, H. (2010, 9-11 April 2010). Image denoising using Contourlet and two-dimensional Principle Component Analysis. Xi'an, China. *Proceedings of 2010 International Conference on Image Analysis and Signal Processing, IASP*, 309-313, doi: 10.1109/IASP.2010.5476106.

[6] Burt, P. J., & Adelson, E. H. (1983). The Laplacian Pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4), 532-540, doi: 10.1109/TCOM.1983.1095851.

[7] Bamberger, R. H., & Smith, M. J. T. (1992). A filter bank for the directional decomposition of images: Theory and design. *IEEE Transactions on Signal Processing*, 40(4), 882-893, doi: 10.1109/78.258085.

[8] Shapiro, J. M. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12), 3445-3462, doi: 10.1109/78.258085.

[9] Vetterli, M. (1984). Multidimensional subband coding: Some theory and algorithms. *Signal Processing 1984*, 6(2), 97-112, doi: 10.1016/0165-1684(84)90012-4.

[10] Yifan, Z., & Liangzheng, X. (2008). Contourlet-based feature extraction on texture images. *Proceedings of the 2008 International Conference on Computer Science and Software Engineering, CSSE*, 221-224, doi: 10.1109/CSSE.

[11] Cohen, A., Daubechies, I., & Feauveau, J. C. (1992). Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45(5), 485-560, doi: 10.1002/cpa.3160450502.

[12] Katsigiannis, S. (2011). Acceleration of the Contourlet Transform. *M.Sc. thesis*, Athens University of Economics and Business.

[13] Mavridis, P., & Papaioannou, G. (2013). The Compact YCoCg Frame Buffer. *GPU Pro 4*, CRC Press.

[14] Malvar, H., & Sullivan, G. (2003). 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.

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

[16] *Center for Image Processing Research (CIPR), Rensselaer Polytechnic Institute*, http://www.cipr.rpi.edu/resource/stills/index.html, accessed 1 July 2012.

[17] *Object and Concept Recognition for Content-Based Image Retrieval Groundtruth Database*, University of Washington, http://www.cs.washington.edu/research/imagedatabase/groundtruth/, accessed 1 July 2012.

[18] International Telecommunication Union. (2011). Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. *Recommendation BT. 601-7 (03/11)*.

IntechOpen