

Reducing computational complexity of three-dimensional discrete cosine transform in video coding process

Jari J. Koivusaari^a, Jarmo H. Takala^b, and Moncef Gabbouj^a

^aInstitute of Signal Processing, ^bInstitute of Digital and Computer Systems,
Tampere University of Technology, P.O.Box 553, FIN-33101 Tampere, Finland

ABSTRACT

Low complexity video coding schemes are aimed to provide video encoding services also for devices with restricted computational power. Video coding process based on the three-dimensional discrete cosine transform (3D DCT) can offer a low complexity video encoder by omitting the computationally demanding motion estimation operation. In this coding scheme, extended fast transform is also used, instead of the motion estimation, to decorrelate the temporal dimension of video data. Typically, the most complex part of the 3D DCT based coding process is the three-dimensional transform. In this paper, we demonstrate methods that can be used in lossy coding process to reduce the number of one-dimensional transforms required to complete the full 3D DCT or its inverse operation. Because unnecessary computations can be omitted, fewer operations are required to complete the transform. Results include the obtained computational savings for standard video test sequences. The savings are reported in terms of computational operations. Generally, the reduced number of computational operations also implies longer battery lifetime for portable devices.

Keywords: video coding, 3D DCT, discrete cosine transform, low complexity

1. INTRODUCTION

Increasing number of mobile devices is equipped with camera devices that are capable of capturing sequential frames to produce a video sequence. However, standard video encoding approaches are still too complex to be efficiently realized on a mobile device. To realize a video encoding process on a mobile device, approximation and optimization of standard video coding algorithms is required, or totally different video encoding scheme need to be used. Low complexity video coding schemes are targeted to offer video encoding also for devices with restricted computational power. Video coding process based on the three-dimensional discrete cosine transform (3D DCT) can offer a low complexity video encoder by omitting the computationally demanding motion estimation operation. In this coding scheme, extended fast transform is used, instead of the motion estimation, to decorrelate the temporal redundancy of a video sequence.

The most complex part of the 3D DCT based coding process is the three-dimensional transform. Separability property of the DCT allows the 3D DCT to be computed by means of one-dimensional transforms. The resulting 3D DCT can be obtained by taking one-dimensional transforms consecutively in each of the three dimensions, i.e., vertical, horizontal, and temporal dimensions. To complete the full 3D DCT operation, totally 192 one-dimensional DCTs are required if the video cube of size $8 \times 8 \times 8$ is used during the processing of the video sequence.

In this paper, we demonstrate methods that can be used in lossy coding process to reduce the required number of one-dimensional transforms to complete the 3D DCT operation or its inverse. In the decoder, the coefficient information can be used to determine which inverse transforms need to be computed to obtain correct reconstruction. For video sequences with low motion, typically half of the one-dimensional transforms can be omitted without reducing the reconstruction quality. In the encoder, computation of the transform and quantization can be reordered so that many of the forward transforms can be omitted to obtain equal coefficient representation.

Correspondence: Email: jari.koivusaari@tut.fi, Tel: +358 3 3115 11, Fax: +358 3 3115 3095

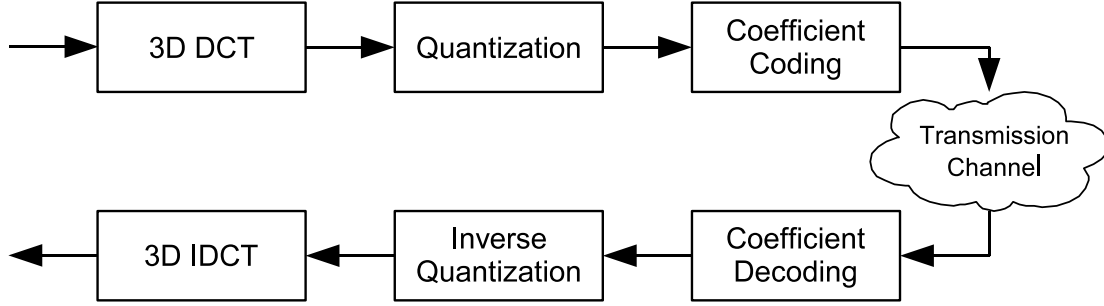


Figure 1. Block diagram of video coding process based on the three-dimensional discrete cosine transform (3D DCT).

Section 2 introduces briefly a video coding process that is based on the three-dimensional discrete cosine transform. Section 3 proposes an algorithm to reduce the number of forward one-dimensional transforms while computing the 3D DCT on encoder. Section 4 proposes a method that can be used to reduce number of inverse transforms on the 3D DCT based decoder. Section 5 represents the obtained results, and Section 6 concludes the paper.

2. VIDEO CODING PROCESS BASED ON 3D DCT

The video coding process based on the 3D DCT can be considered as a logical extension to the 2D DCT based still image codec¹ like the baseline sequential codec of the Joint Photographic Expert Group (JPEG).² In the 3D DCT based video codec, the discrete cosine transform is used not only to eliminate the spatial redundancy but also to eliminate the temporal redundancy of video sequences as well. Furthermore, the processing of image pixel data with 8×8 blocks is extended to the processing of video voxel data with $8 \times 8 \times 8$ cubes.

A block diagram of the 3D DCT based video coding process is shown in Figure 1. The encoding process begins with a partitioning operation. The downsampled YCbCr (4:2:0) video data is partitioned into smaller non-overlapping $8 \times 8 \times 8$ cubes containing a portion of pixel data from 8 sequential source frames. Then, the 3D DCT, quantization, and coefficient coding is applied to each individual $8 \times 8 \times 8$ cube to obtain the final compressed binary representation.

2.1. Three-Dimensional Discrete Cosine Transform

For pixel values $\mathbf{s}(x, y, z)$ of three-dimensional input data cube of size $L \times M \times N$, the three-dimensional discrete cosine transform can be defined as

$$\mathbf{S}(u, v, w) = \sqrt{\frac{8}{LMN}} \alpha_u \alpha_v \alpha_w \sum_{x=0}^{L-1} \sum_{y=0}^{M-1} \sum_{z=0}^{N-1} \mathbf{s}(x, y, z) \cos \frac{\pi(2x+1)u}{2L} \cos \frac{\pi(2y+1)v}{2M} \cos \frac{\pi(2z+1)w}{2N}, \quad (1)$$

and equally its inverse as

$$\mathbf{s}(x, y, z) = \sqrt{\frac{8}{LMN}} \sum_{u=0}^{L-1} \sum_{v=0}^{M-1} \sum_{w=0}^{N-1} \alpha_u \alpha_v \alpha_w \mathbf{S}(u, v, w) \cos \frac{\pi(2x+1)u}{2L} \cos \frac{\pi(2y+1)v}{2M} \cos \frac{\pi(2z+1)w}{2N}, \quad (2)$$

where

$$\alpha_u, \alpha_v, \alpha_w = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v, w = 0 \\ 1 & \text{for } u, v, w \neq 0 \end{cases} \quad (3)$$

and $\mathbf{s}(x, y, z)$ is a pixel value in spatial domain and $\mathbf{S}(u, v, w)$ is a resulting transform coefficient in frequency domain. The three-dimensional transform aims to reduce the correlation of pixel values within an $8 \times 8 \times 8$ cube and to represent the information with a small number of visually significant transform coefficients. The 3D DCT is a separable transforms, thus one-dimensional discrete cosine transforms can be used to compute the full 3D

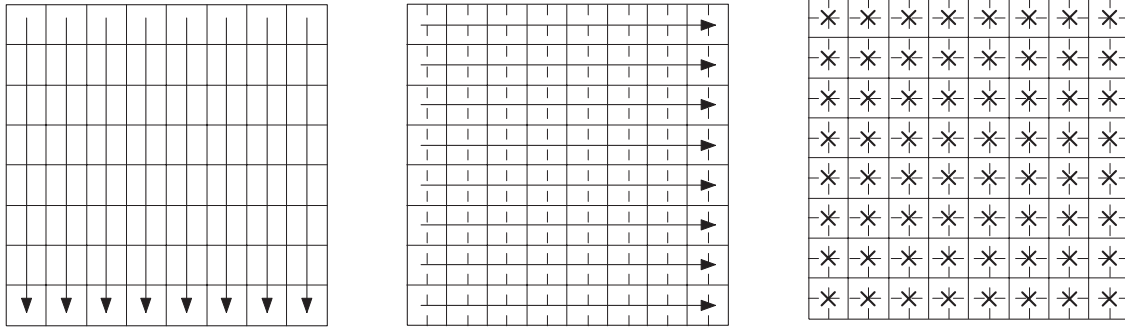


Figure 2. Separable operations: (1) to columns (2) to rows (3) to the time dimension

DCT as depicted in Figure 2. First, we can apply one-dimensional transforms to columns, then to rows, and finally to the time dimension to obtain the full 3D DCT representation. Naturally, any selected order can be used to obtain equivalent result by applying one-dimensional transforms sequentially to each of the three dimensions. For an $8 \times 8 \times 8$ cube, totally 192 one-dimensional transforms are required to complete the 3D DCT operation.

If one-dimensional N -point DCT is computed using direct matrix operations, $N(N - 1)$ additions and N^2 multiplications are required. However, many sophisticated DCT algorithms have been introduced to reduce computational complexity. Table 1³ shows the number of computational operations for fast 8-point DCT algorithms that has been introduced previously.

Table 1. Number of computational operations for one-dimensional 8-point DCT.³

Author	Chen	Wang	Lee	Vetterli	Suehiro	Hou	Loeffler
Multiplication	16 (13)	13	12	12	12	12	11
Addition	26 (29)	29	29	29	29	29	29

2.2. Quantization

Quantization is a many-to-one mapping. The idea is to exclude visually unimportant transform coefficients and to reduce, as much as reasonable, dynamic range of remaining coefficients, thus increasing compression. Previously, some quantization strategies have been proposed for 3D DCT based video coding. These strategies utilize the following ideas. First, a quantization table, described for example by JPEG,² can be used individually for each time-dependent 8×8 block⁴ or can be used to construct a whole $8 \times 8 \times 8$ quantization volume.⁵ Second, the unique quantization volume can be constructed, for example, based on human visual acuity⁶ or based on the statistical properties of coefficients,^{7,8} like variance distribution. In addition, adaptive methods minimizing the quantization error⁶ have been proposed to construct a quantization volume.

Unfortunately, the interaction between the quantized 3D DCT coefficients and the perceived quality of the reconstructed video sequence is still hardly known. Therefore, it is quite an elaborate task to construct an efficient quantization scheme for the $8 \times 8 \times 8$ cube of DCT coefficients. None of the reported quantization methods was superior for our purposes. Therefore, we decided to utilize a simple quantization scheme in our coding process. The uniform quantization is used for AC coefficients and the DC coefficient is always quantized with the constant value of 20.

2.3. Coefficient Coding

After the quantization operation, coefficients need to be reordered to achieve an efficient run-length representation by minimizing intervening zeros within coefficients. Some scanning strategies for 3D DCT coefficients were

introduced in prior experiments as follows. The coefficients can be arranged according to the magnitudes of variances,⁷ so that coefficients with higher magnitudes are chosen first. Equally, the well-known 2D zigzag scan can be extended into the 3D version by scanning coefficients with isometric planes⁵ defined as $x + y + z = c$ where c is varying from 0 to 21, that is, from DC to higher frequencies. Further, the method referred to as *shifted complement hyperboloid*⁸ rearranges coefficients based on the exponential function formulated by scrutinizing the distribution and the dynamic range of coefficients.

The method based on isometric planes was selected to our baseline codec, because none of the abovementioned methods was distinctly superior for every video test sequence we used. After the selection, variable-length codes for run-length value pairs was constructed to obtain the final compressed binary representation. Due to its implementational simplicity, Huffman coding was used.

Because in the $8 \times 8 \times 8$ DCT cube, the dynamic range of coefficients can be almost three times wider and the number of consecutive zeros can be considerably higher than in the 8×8 DCT block, a new set of Huffman tables to map run-length value pairs into variable-length codes was constructed. This was carried out by using various video test sequences to determine the frequency of occurrence of each value pair. Based on the resulting statistics, three static Huffman code tables were generated: one for luminance channel Y and two for chrominance channels Cb and Cr.

3. REDUCING FORWARD TRANSFORMS

Previously, various methods have been introduced for standard coding processes, such as H.263⁹ and MPEG-2,¹⁰ and mainly to reduce computations of inter-coded frames. The computational complexity of the coding processes can be reduced by using approximation, optimization, and prediction algorithms. For example, by successfully predicting of all-zero blocks before the transform and quantization, both of these operations can be avoided and computations can be saved.

To reduce computational complexity of forward transform, optimization and approximation can be used. However, in case of lossy coding process, such as a video coding process, we have many possibilities to truncate unnecessary information. Because quantization operation is typically truncating a lot of transform coefficient, we have a possibility to avoid unnecessary computations during the transform. Computations can be saved especially if we can compute the transform step by step. In this case, we can predict or even determine which computations can be excluded to obtain equivalent result. For example, in video coding process based on multidimensional transform, there is no need to compute some of the transforms if those coefficients are going to be lost anyway.

Only few transform coefficients survive through the quantization operation. We can take advantage of this fact to minimize the computations of the transform. In case of 3D DCT, we can operate in row-column-depth manner and after each one-dimensional transform step we can truncate some part of the $8 \times 8 \times 8$ cube that is not required for the further processing. The further processing is applied only for the remaining sub-cube. Thus, the size of the sub-cube is reduced after each step of operation, as well as, the number of transforms that are required to achieve the final outcome.

First, we can apply, e.g., 64 one-dimensional horizontal transforms to an $8 \times 8 \times 8$ cube. After transforms, a threshold value can be used to truncate some low energy coefficients. In the same time, we store the maximum width α for the cube, i.e., the plane that is dividing the cube into two sub-cubes. Now, one sub-cube contains coefficients that are going to be used for further processing and the other sub-cube contains unnecessary coefficients that are truncated. Second, we apply 8α one-dimensional vertical transforms to the remaining sub-cube. After the transforms of this second stage, we store the maximum height β for the sub-cube same way as in the first stage. Finally, we apply $\alpha\beta$ one-one dimensional temporal transforms to complete the full three-dimensional transform.

Figure 3 demonstrates the aforementioned operation for the cube of pixel data where the three-dimensional transform is applied. First, we can choose one of the three dimensions to be processed. This can be done, e.g., by using some estimation scheme or just by randomly selecting one of the three. If the correct operation order for each dimension can be determined, the more transforms can be reduced while processing the next dimension. Second, we apply one-dimensional transforms along to selected dimension. After each transform step some of the high frequency coefficients are truncated. In Figure 3, planes are demonstrating how the sub-cubes are divided

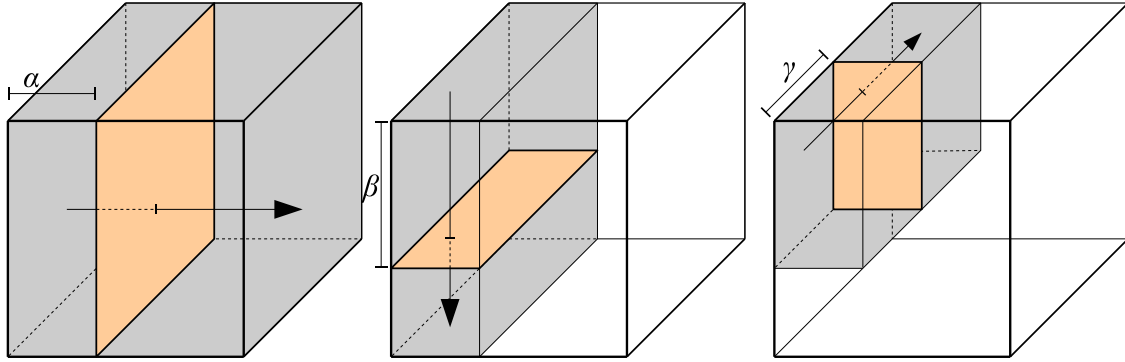


Figure 3. After a set of one-dimensional transforms, the resulting coefficients are divided into two sub-cubes. One is truncated and the other is used for further processing.

into two smaller sub-cubes during each processing steps. One sub-cube is truncated, containing only coefficients that are below predefined threshold. The other is going to be used when next dimension is processed. The arrows are pointing out the direction of the transform during each step.

4. REDUCING INVERSE TRANSFORMS

On the decoder side, the operation is easier to carry out, because we are able to obtain the width, height, and depth for the volume, i.e. sub-cube, that is containing all transform coefficients that need to be mapped back to the pixel values by using one-dimensional inverse discrete cosine transforms. After zero run-length decoding operation, inverse scanning order is applied to coefficients. During the inverse scanning operation we can store the maximum distance from DC coefficient in terms of width, height, and depth for the sub-cube within all of the non-zero coefficients are located. By applying one-dimensional transforms first towards the dimension obtaining the largest distance we can save the number of required transforms quite dramatically.

During the inverse coefficient scanning operation, we store dimensions of the sub-cube, let's say α , β , and γ , each describing the width, height, and depth of the sub-cube, correspondingly. All non-zero coefficients are located within the sub-cube. Now, we are taking three steps to complete the inverse transform. During each step, we apply one-dimensional inverse transforms along one of the three dimensions, and at the same time, we are expanding the corresponding dimension of the sub-cube to its full length.

Inverse transforms should be applied in such order that total number of transforms is minimized. So let us assume that we are processing $8 \times 8 \times 8$ cubes and let us assume that we have ordering, e.g., $\alpha \leq \beta \leq \gamma$. So the γ is the largest dimension and α is the smallest. To minimize the number of required one-dimensional inverse transforms we should operate in each step towards the largest dimension. With the abovementioned ordering, the following three steps are applied:

1. Apply one-dimensional transforms along largest dimension, i.e. γ
(Requires $\alpha\beta$ one-dimensional transforms)
2. Apply one-dimensional transforms along second largest dimension, i.e., β
(Requires 8α one-dimensional transforms in case of $8 \times 8 \times 8$ cube)
3. Apply one-dimensional transforms along the last dimension, i.e., α
(Requires 64 one-dimensional transforms in case of $8 \times 8 \times 8$ cube)

Thus, the total number of transforms in case of $8 \times 8 \times 8$ cube size is $8^2 + 8\alpha + \alpha\beta$ that is typically much smaller than the total 192 transforms as the results in the following section will show.

5. EXPERIMENTAL RESULTS

The algorithms represented in Section 3 and Section 4 were implemented to a three-dimensional discrete cosine transform based video coding process by using C language and Gnu Compiler Collection (GCC) tools. Standard YCbCr video test sequences were encoded and during the encoding process the number of one-dimensional discrete cosine transforms was computed. The compressed bitstreams of each sequences were decoded with 3D DCT based decoder with and without the algorithm introduced in Section 4 and the corresponding number of require one-dimensional inverse transforms was computed. The objective and subjective quality was maintained while the algorithm was applied. The reason is that the algorithm does not compute the transforms for the values that are going to be truncated by the quantization operation anyway.

Table 2 shows the number of forward transforms that was used to encode each sequence with 3D DCT based video encoder. We can see that less than half of the transforms are actually needed for video sequences with low motion activity to obtain equivalent result with the original coding process that does not take advantage of the proposed algorithm. Table 3 contains the corresponding numbers of inverse transforms that was used to decode video sequences. We can see that the savings obtained on decoder side are somewhat larger than on the encoder side. The reason for this is that quantization was truncating more coefficients than was estimated on the encoder side. Furthermore, we actually know the best order to apply inverse transforms on decoder side so the total number of required transforms can be minimized.

Table 2. Number of one-dimensional forward transforms used to encode video sequences and the obtained saving ratios for each sequence.

Sequence name	Original	Proposed	Proposed/Original	Size
Akiyo	4333824	1814769	0.41	QCIF
Coastguard	4333824	2760242	0.64	QCIF
Glasgow	10720512	7361849	0.69	QCIF
Miss America	2166912	938572	0.43	QCIF
Tempete	15054336	10315344	0.68	CIF

Table 3. Number of one-dimensional inverse transforms used to decode video sequences and the obtained saving ratios for each sequence.

Sequence name	Original	Proposed	Proposed/Original	Size
Akiyo	4333824	1715939	0.40	QCIF
Coastguard	4333824	2124726	0.49	QCIF
Glasgow	10720512	5772444	0.54	QCIF
Miss America	2166912	862859	0.40	QCIF
Tempete	15054336	7863666	0.52	CIF

Table 4 demonstrates how much faster encoding and decoding processes were obtained by utilizing the proposed methods to reduce number of transforms. GNU Time tool was used to measure encoding and decoding times. The obtained results are normalized against the original 3D DCT based video coding processes. Based on the result, one can see that the proposed methods can offer significant gain in coding speed. In the decoder, the unnecessary computations of transforms can be more efficiently avoided than in the encoder.

Table 4. Coding speed improvement for the encoder and the decoder utilizing the proposed methods to reduce the number of transforms.

Sequence name	Original	Encoder	Decoder	Size
Akiyo	1	1.64	2.05	QCIF
Coastguard	1	1.27	1.67	QCIF
Glasgow	1	1.22	1.55	QCIF
Miss America	1	1.62	2.02	QCIF
Tempete	1	1.23	1.59	CIF

6. CONCLUSIONS

Two algorithms were proposed to reduce the number of one-dimensional transforms, i.e., computational complexity of three-dimensional discrete cosine transform in a video coding process. Based on the experiments that were carried out with several standard video test sequences, only from 40% to 70% of one-dimensional DCTs are required to obtain equivalent result with the original. Furthermore, encoding speed can be increased by a factor from 1.2 to 1.6. Correspondingly, the obtained decoding speed was 1.5 to 2.0 times faster than the original. The proposed methods can be applied with any kind of DCT computation schemes from the basic definition to fast DCT algorithms. Furthermore, similar approach can be utilized with other separable transforms too in multidimensional transform based coding processes. Best savings can be expected in lossy coding processes, such as video coding processes.

REFERENCES

1. R. Westwater and B. Furht, "The XYZ algorithm for real-time compression of full-motion video," *Real-Time Imaging Journal, Special Issue on Image and Video Processing in Multimedia Systems* **2**, pp. 19–34, Feb. 1996.
2. ISO/IEC 10918-1 | ITU-T Recommendation T.81, *Information Technology - Digital Compression and Coding of Continuous-tone Still Images - Requirements and Guidelines*, Sept. 1992.
3. C. Loeffler, A. Ligtenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-89)* **2**, pp. 988–991, May 1989.
4. M. Bakr and A. E. Salama, "Implementation of 3D-DCT based video encoder/decoder system," *The 45th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS-2002)* **2**, pp. II–13–16, Aug. 2002.
5. Y. Boon-Lock and L. Bebe, "Volume rendering of DCT-based compressed 3D scalar data," *IEEE Transactions on Visualization and Computer Graphics* **1**, pp. 29–43, Mar. 1995.
6. R. Westwater and B. Furht, "Three-dimensional DCT video compression technique based on adaptive quantizers," *Second IEEE International Conference on Engineering of Complex Computer Systems*, pp. 189–198, Oct. 1996.
7. J. Roese, W. Pratt, and G. Robinson, "Interframe cosine transform image coding," *IEEE Transactions on Communications* **COM-25**, pp. 1329–1339, Nov. 1977.
8. M. C. Lee, R. K. W. Chan, and D. A. Adjeroh, "Quantization of 3D-DCT coefficients and scan order for video compression," *Journal of Visual Communications and Image Representation* **8**, pp. 405–422, Dec. 1997.
9. ITU-T Recommendation H.263, *Video coding for low bit rate communication*, Feb. 1998.
10. ISO/IEC 13818-2 | ITU-T Recommendation H.262, *Information technology - Generic coding of moving pictures and associated audio information: Video*, July 1995 (E).