FAST CONVERSION OF H.264/AVC INTEGER TRANSFORM COEFFICIENTS INTO DCT COEFFICIENTS

R. Marques¹, V. Silva¹,², S. Faria¹,³, A. Navarro¹,⁴, P. Assuncao¹,³

¹Instituto de Telecomunicações, ²Universidade de Coimbra - DEEC, 3030-290 Coimbra, Portugal
³Instituto Politécnico de Leiria - ESTG, Apt 4163, 2411-901 Leiria, Portugal
⁴Universidade de Aveiro – DET, 3810-193 Aveiro, Portugal

{ricky, vitor, sergio, assuncao}@co.it.pt, navarro@av.it.pt

Keywords: Transform conversion, video transcoding.

Abstract: In this paper we propose a fast method to convert H.264/AVC Integer Transform (IT) coefficients to Discrete Cosine Transform (DCT) coefficients for applications in video transcoding. We derive the transform kernel matrix for converting, in the transform domain, four IT coefficient blocks into one 8×8 DCT block of coefficients. By exploiting the symmetry of this matrix, we show that the proposed conversion method requires fewer computations than its equivalent in the pixel domain. An integer approximation for the kernel matrix is also proposed. The experimental results show that a negligible error is introduced, while the computational complexity can be significantly reduced.

1. INTRODUCTION

The H.264 is a new video coding standard, recently approved by ITU-T and ISO/IEC as International Standard. When compared to earlier video coding standards like H.263, the H.264 video format can provide enhanced compression efficiency. Experimental results show that about 50% of the bitrate can be saved by using H.264 (Sullivan et al. 2004). Given this coding efficiency, H.264 has been adopted by Digital Multimedia Broadcasting (DBM) as the standard video codec, and is expected to be extended to many other areas of application, such as mobile communications and video conferencing.

Whenever a new standard is adopted, this always gives rise to interoperability problems with legacy systems. In the case of H.264, interoperability with MPEG-2 systems is of particular importance. In general, this is achieved through video transcoding for conversion between different standards (Chuang et al. 2005). However, there are significant differences between the H.264 and other video coding standards, which difficult the transcoding process, e.g., while most MPEG video codecs use the 8×8 DCT to produce transform coefficients, H.264 use the 4×4 IT.

This paper addresses the problem of converting H.264/AVC Integer Transform (IT) coefficients to Discrete Cosine Transform (DCT) coefficients for video transcoding applications. We derive the kernel matrix for conversion in the transform domain and along with a fast algorithm to reduce the number of operations. Then we introduce an integer approximation to increase computing performance using fixed-point arithmetic.

The organization of this paper is as follows. In section 2, we describe the proposed transform domain IT-to-DCT conversion. In sections 3 and 4 the fast conversion algorithm and its integer approximation are respectively described. The experimental results are presented in section 5 and finally section 6 concludes the paper.

2. IT-TO-DCT CONVERSION

Figure 1 shows the pixel domain implementation of the IT-to-DCT conversion. The input is comprised of four 4×4 blocks \((x_1, x_2, x_3, x_4)\) of IT coefficients. The inverse IT is applied to each of the four blocks in order to obtain the pixel domain blocks \((x_1, x_2, x_3, x_4)\). Then the four pixel domain blocks are combined to form a single 8×8 block \((x)\) to which the DCT is applied, such that an 8×8
block of transform coefficients ($y$) is obtained. However, transform domain conversion is more efficient because complete decoding up to the pixel domain is not required.

The proposed transform domain IT-to-DCT conversion is based on the so-called $S$-matrix (Xin et al. 2004). It is applied to an $8 \times 8$ block ($x$) comprised of four $4 \times 4$ blocks ($x_1, x_2, x_3, x_4$) of IT coefficients to produce the corresponding $8 \times 8$ block ($y$) of DCT coefficients. The conversion is given by the following operation,

$$ Y = S \times X \times S^T, \quad (1) $$

where $S$ is the transform kernel matrix and $S^T$ is its transpose. In order to derive $S$, we have to consider the inverse IT of blocks $x_1, x_2, x_3, x_4$, which results in pixel blocks $x_1, x_2, x_3, x_4$, each one given by $x_j = J \times x_j \times J^T$, where $J$ is the following matrix (Malvar et al. 2003).

$$ J = \begin{pmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & 1 & 1 & \frac{1}{2} \end{pmatrix} \quad (2) $$

If we consider $K = \begin{pmatrix} J \\ 0 \end{pmatrix}$, then we can compute $y$ in a single step as given by,

$$ y = K \times X \times K^T. \quad (3) $$

Since the DCT of an $8 \times 8$ block can be defined as

$$ Y = T \times y \times T^T, \quad (4) $$

where $T$ is the DCT kernel matrix, then it follows that,

$$ Y = T \times K \times X \times K^T \times T^T \quad (5) $$

From the above equation we can define the transform kernel matrix $S$ as

$$ S = T \times K \quad (6) $$

and each value of the $S$ matrix is rounded to four decimal places as follows,

\[
\begin{align*}
    a &= 1.4142 & b &= 1.2815 & c &= 0.4618 & d &= -0.1065 \\
    e &= 0.0585 & f &= 1.1152 & g &= 0.0793 & h &= -0.45 \\
    i &= 0.8399 & j &= 0.7259 & k &= -0.0461 & l &= 0.3007 \\
    m &= -0.4319 & n &= 1.0864 & o &= 0.5190 & p &= -0.2549 \\
    q &= 0.2412 & r &= -0.5308 & s &= 0.9875
\end{align*}
\]

### 3. FAST ALGORITHM

The proposed fast IT-to-DCT conversion algorithm is based on the symmetry characteristics of the $S$ matrix. As it shall be explained, this characteristic of the $S$ matrix is exploited for achieving fast computation of the transform conversion.

Since the transform defined by (1) is separable, it can be computed by column transforms followed by row transforms. If we define $z$ as an 8 point column vector and $Z$ its 1D transform, then by using the horizontal symmetry of the $S$ matrix, we can map the combinations defined in each line of $S$ to 8 variables. Then the 1D transform $Z$ is obtained from linear operations of these variables. We have derived the following algorithm in order to achieve an efficient method to determine $Z$.

![Diagram of Pixel domain IT-to-DCT conversion](image)

**Figure 1: Pixel domain IT-to-DCT conversion**
2004) architecture flexible DSPs faster than floating point
for the kernel matrix
In order to achieve
4.
conversion operations when compared to the pixel domain
significantly
proposed (operations) which yields a total of
inverse IT (320 operations)
operations. fast conversion 1D transforms (352 operations)
1D transform (operations, i.e.,
This algorithm needs 22 multiplications and 22 additions, i.e., a total of 44 operations to perform the 1D transform. The 2D S-transform needs 8 columns 1D transform \(8 \times 44 = 352 \) operations) and 8 rows 1D transforms (352 operations). Thus the proposed fast conversion algorithm needs a total of 704 operations. The pixel domain approach needs four inverse IT (320 operations) and one DCT (672 operations) which yields a total of 992 operations. (Xin et al. 2004, Lee et al. 2005). Therefore, the proposed transform domain fast algorithm significantly reduces (30%) the number of operations when compared to the pixel domain conversion.

4. INTEGER APPROXIMATION

In order to obtain integer arithmetic we scale the \( S \)-matrix by multiplying it by an integer that is a power of 2. To represent each pixel residual value, we need 9 bits and to perform the IT, we need 11 bits to represent the coefficients. The maximum gain of the 2D \( S \)-transform matrix is \( 4.67^2 \), which implies that more 5 bits are needed to represent the result of the conversion. Therefore, the scaling factor must be smaller or equal than the square root of \( \{2^{12}-2^{16}\} \). The integer transform matrix is given by \( \text{Si} = \text{round}(256 \times S) \), yielding

\[
\begin{pmatrix}
a' & 0 & 0 & 0 & a' & 0 & 0 & 0 \\
b' & c' & d' & e' & -b' & c' & -d' & e' \\
0 & f' & 0 & g' & 0 & -f' & 0 & -g' \\
h' & i' & j' & k' & -h' & i' & -j' & k' \\
0 & 0 & a' & 0 & 0 & 0 & a' & 0 \\
l' & m' & n' & o' & -l' & m' & -n' & o' \\
0 & -g' & 0 & f' & 0 & g' & 0 & -f' \\
p' & q' & r' & s' & -p' & q' & -r' & s'
\end{pmatrix}
\]

The integer values of the Si-matrix are:

\[
\begin{align*}
a' &= 362 & b' &= 328 & c' &= 118 & d' &= -27 \\
e' &= 14 & f' &= 285 & g' &= 20 & h' &= -115 \\
i' &= 227 & j' &= 185 & k' &= -11 & l' &= 75 \\
m' &= -110 & n' &= 278 & o' &= 132 & p' &= -65 \\
q' &= 61 & r' &= -136 & s' &= 352
\end{align*}
\]

As described in section 3, we can apply this algorithm to compute fast integer conversion. In order to reduce the computational complexity, we apply multiplier blocks to use parallel multiple constants multiplications. Then only low complexity operations are performed, i.e., additions, subtractions and shifts (Puschel et al. 2004). The multiplier blocks for the Si matrix are the following,

\[
\begin{align*}
b_1 &= m_1 \times \begin{bmatrix} b \\ h \\ l \\ p \\ e \end{bmatrix},
& b_2 = m_2 \times \begin{bmatrix} c \\ i \\ m \\ q \\ k \end{bmatrix},
& b_3 = m_3 \times \begin{bmatrix} d \\ j \\ n \\ r \end{bmatrix},
& b_4 = m_4 \times \begin{bmatrix} f \\ g \end{bmatrix},
& b_5 = m_5 \times \begin{bmatrix} g \\ f \end{bmatrix}
\end{align*}
\]
The number of low complexity operations required to compute each block is shown in Table 1.

Table 1 Number of operations per multiplier block

<table>
<thead>
<tr>
<th>Block</th>
<th>Add/Sub</th>
<th>Shift</th>
<th>Neg</th>
<th>Mb</th>
<th>Mu</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>31.5</td>
<td>56</td>
</tr>
<tr>
<td>b₂</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>31.0</td>
<td>56</td>
</tr>
<tr>
<td>b₃</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>31.5</td>
<td>56</td>
</tr>
<tr>
<td>b₄</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>30.5</td>
<td>56</td>
</tr>
<tr>
<td>b₅</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>18.0</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 2 shows the number of clock cycles required by a general purpose processor (Intel, 2001) to compute each multiplier block, (column Mb) as well as the conventional multiplier method (column Mu). As it can be seen, the number of clock cycles required by the integer fast approximation based on multiplier blocks is about 55% of those required by the conventional method.

Table 2 Number of clock cycles per block operations

<table>
<thead>
<tr>
<th>Block</th>
<th>Add/Sub</th>
<th>Shift</th>
<th>Neg</th>
<th>Mb</th>
<th>Mu</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>2.5</td>
<td>28.0</td>
<td>1.0</td>
<td>31.5</td>
<td>56</td>
</tr>
<tr>
<td>b₂</td>
<td>2.5</td>
<td>28.0</td>
<td>0.5</td>
<td>31.0</td>
<td>56</td>
</tr>
<tr>
<td>b₃</td>
<td>3.0</td>
<td>28.0</td>
<td>0.5</td>
<td>31.5</td>
<td>56</td>
</tr>
<tr>
<td>b₄</td>
<td>2.0</td>
<td>28.0</td>
<td>0.5</td>
<td>30.5</td>
<td>56</td>
</tr>
<tr>
<td>b₅</td>
<td>1.5</td>
<td>16.0</td>
<td>0.5</td>
<td>18.0</td>
<td>28</td>
</tr>
</tbody>
</table>

5. EXPERIMENTAL RESULTS

We have evaluated the error introduced by integer approximation of the S matrix by comparing both methods described in previous sections. A set of 3 different black&white images was used. For each one, the whole image was transformed into 4×4 IT coefficient blocks. Then each group of four adjacent 4×4 IT coefficient blocks, i.e., each 8×8 IT coefficient block, was converted into DCT coefficient blocks by means of two different methods: i) the fast algorithm described in section 3; ii) the integer approximation described in section 4. The mean squared error (MSE), between the resulting images, was used for evaluating the error introduced by the integer approximation method.

The results are shown in Table 3, where it can be seen that the error due to arithmetic precision reduction of the integer approximation in the conversion process is actually very small. In fact the resulting MSE is negligible in practical terms, which proves the usefulness of the proposed method for fast transcoding implementations.

Table 3 MSE of the integer approximation

<table>
<thead>
<tr>
<th>Image</th>
<th>Einstein</th>
<th>Smandril</th>
<th>Cameraman</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>0.337</td>
<td>0.339</td>
<td>0.340</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

In this paper, we proposed a transform domain approach for fast conversion of IT coefficients to DCT coefficients. We derived the transform kernel matrix and an efficient algorithm for computing the transform, as well as a low complexity integer approximation method. The presented results show that the proposed methods are much faster than the pixel domain approach. These methods prove to be suitable for video transcoding applications where fast processing is required.

7. REFERENCES


Texas Instruments, 2004. TMS 320C600 CPU and Instruction Set Reference Guide. Literature Number SPRU189F.