A FAST ARBITRARY FACTOR H.264/AVC VIDEO RE-SIZING ALGORITHM

Vasant Patil
Institute for Systems Studies and Analyses
Delhi, India
vgpats@yahoo.com

Rajeev Kumar
Computer Science & Engineering Department
Indian Institute of Technology Kharagpur, India
rkumar@cse.iitkgp.ernet.in

ABSTRACT
An algorithm for re-sizing H.264/AVC video in DCT space is presented. We demonstrate that a frame re-sizing operation can be represented as multiplication by fixed matrices and propose an efficient computation scheme. The proposed approach is general enough to accommodate re-sizing operations with arbitrary factors conforming to the syntax of H.264 video. It shows a good PSNR than the spatial domain bilinear and bicubic spline interpolation at much reduced cost.

Index Terms— Video transcoding, DCT-domain processing, arbitrary re-sizing, H.264

1. INTRODUCTION
Video transcoding is emerging as a key technology for universal multimedia access (UMA) and many other applications requiring a variety of access links, devices and resources. Frame down-sizing video transcoding by an arbitrary factor provides much finer and dynamic adjustment of pre-coded video to meet various channel conditions and user requirements, and thus, has evoked greater attention in recent times. On the other hand, H.264 with superior coding efficiency achieved by employing techniques such as variable block-size motion estimation and mode decision, intra prediction, and multiple reference frames is expected to be widely used in variety of networked multimedia applications. In this paper, we consider a problem of re-sizing H.264 video by an arbitrary factor in DCT domain. A straightforward approach of decoding followed by re-encoding at desired resolution is unsuitable, due to inherent high complexity, for H.264 based real-time applications. Therefore, it is worthwhile to develop fast algorithms for re-sizing H.264 directly in the compressed domain.

In this paper, the problem of re-sizing H.264 video by an arbitrary factor is handled in a manner similar to that in [1]. However, it demands a different treatment because H.264 uses modified DCT, intra prediction and $4 \times 4$ transform block size [2]. We propose a fast algorithm to construct a target DCT frame as a whole in one go, from the anchor $4 \times 4$ modified DCT blocks in the original frame. By this the hidden shared information is exposed and the overall process of computation is significantly sped up. We demonstrate that basic operation in re-sizing can be represented as multiplication by fixed matrices and propose a faster computation scheme.

The paper is organized as follows. The proposed arbitrary down-sizing and up-sizing algorithms are presented in Section 2 and Section 3, respectively. Experimental results are presented in Section 4 before we conclude in Section 5.

2. ARBITRARY DOWN-SIZING ALGORITHM
In video frame down-sizing, the $M' \times N'$ number of $16 \times 16$ macroblocks in the down-sized frame are related to the $M \times N$ number of $16 \times 16$ macroblocks in the original frame. Here, $M$ and $N$ represent the number of $16 \times 16$ macroblocks in the original frame along vertical and horizontal directions, respectively. Similarly, $M'$ and $N'$ represent the number of $16 \times 16$ macroblocks in the down-sized frame along vertical and horizontal directions, respectively. For the horizontal down-sizing factor $R_x$ and the vertical down-sizing factor $R_y$, we have $M'=M/R_y$ and $N'=N/R_x$. Note that, the down-sizing factors, defined as the ratio of original resolution to the target resolution, are greater than one and may differ from each other. Thus, we can consider reduction in spatial resolution as well as change in aspect ratio of the original frame.

Let us select the smallest integer numbers $m$ and $n$ such that $h = 4m R_y$ and $w = 4n R_x$ are integers. A down-sized frame can be partitioned into $P \times Q = 4M'/m \times 4N'/n$ number of blocks of size $4m \times 4n$ each such that a block $B_{i,j}$ ($1 \leq i \leq P, 1 \leq j \leq Q$) is related to a larger pixel block, called mapped block, of size $h \times w$ in the original frame as shown in Fig. 1. Thus, the original frame can also be partitioned into $P \times Q$ number of mapped blocks of size $h \times w$ each. For example, when an original frame of size $352 \times 288$ is down-sized to $256 \times 256$ ($R_x = 11/8$ and $R_y = 9/8$), we have $w = 4 \cdot 2 \cdot 11/8 = 11$ yielding $n = 2$ and $h = 4 \cdot 2 \cdot 9/8$ yielding $m = 2$. Therefore, an original frame is partitioned into $P \times Q = 4 \cdot 16/2 \times 4 \cdot 16/2 = 32 \times 32$ mapped blocks of size $h \times w = 9 \times 11$ each. As we can see in Fig. 1, the mapped blocks may not align with the boundaries of the $4 \times 4$ blocks in original frame. Therefore, our approach to arbitrary down-
sizing includes two steps - 1) Extracting the $P \times Q$ number of mapped blocks from the original frame, and 2) Down-sizing each mapped block to a block of size $4m \times 4n$ with $m \times n$ number of modified DCT blocks of size $4 \times 4$ each.

Let, $a_{ij}$ and $A_{ij}$ ($1 \leq i \leq 4M$ and $1 \leq j \leq 4N$) represent the $4 \times 4$ spatial domain blocks and the $4 \times 4$ modified DCT blocks (after pre-scaling) of the original frame, respectively. It may be noted that, H.264 supports $b_d$ and $b_s$ intra predictions, which can be removed using transform domain intra prediction proposed in [3]. In addition, we also need to perform inverse DC transform for $b_s$ prediction in order to obtain the modified DCT blocks. In this section, down-sizing of the luma samples is discussed. Extension to the chroma samples is straightforward.

### 2.1. Extracting the Mapped Blocks in Original Frame

The total $P \times Q$ number of mapped blocks in DCT domain can be obtained from the original frame as follows:

$$I = \mathcal{S}_{h,p} \times \begin{pmatrix} a_{11} & \cdots & a_{1(4N)} \\ \vdots & \ddots & \vdots \\ a_{4M \times 1} & \cdots & a_{4M \times (4N)} \end{pmatrix} \times \mathcal{S}_{w,q}^T$$

where 'T' denotes matrix transposition and $\mathcal{S}_{h,p}$ ($l = h$ or $w$, and $p = P$ or $Q$) denotes a matrix of size $lp \times lp$ having $p$ number of $S_l$ (a $l$-point DCT matrix) matrices as the diagonal blocks as follows:

$$\mathcal{S}_{l,p} = \begin{pmatrix} S_l & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & S_l \end{pmatrix}$$

Note that, $\mathcal{S}_{h,p}$ and $\mathcal{S}_{w,q}$ are $16M \times 16M$ and $16N \times 16N$ matrices, respectively.

Let us denote $\mathcal{C}_{f,n}$ as a matrix of size $4n \times 4n$ with $n$ number of $C_f$ (a 4-point modified forward DCT matrix) matrices as the diagonal blocks, $\mathcal{C}_{i,n}$ as a matrix of size $4n \times 4n$ with $n$ number of $C_i$ (a 4-point modified inverse DCT matrix) matrices as the diagonal blocks and $A_{p,q}$ as a matrix of size $4p \times 4q$ as follows:

$$\mathcal{C}_{f,n} = \begin{pmatrix} C_f & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C_f \end{pmatrix} \quad \mathcal{C}_{i,n} = \begin{pmatrix} C_i & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C_i \end{pmatrix}$$

$$A_{p,q} = \begin{pmatrix} A_{11} & \cdots & A_{1q} \\ \vdots & \ddots & \vdots \\ A_{p1} & \cdots & A_{pq} \end{pmatrix}$$

Assuming that we have partially decoded the $a_{ij}$’s, our aim is to obtain a DCT domain frame using $A_{ij}$’s. By using the definitions of $\mathcal{C}_{f,n}$, $\mathcal{C}_{i,n}$ and $A_{p,q}$, (1) can be re-written as follows:

$$I = \mathcal{S}_{h,p} \times \{ \mathcal{C}_{i,4M}^T \times A_{4M,4N} \times \mathcal{C}_{i,4N} \} \times \mathcal{S}_{w,q}^T$$

A matrix multiplication inside the curly braces gives a $16M \times 16N$ matrix representing the original frame in spatial domain prior to the deblocking filter process. A pre-multiplication by $\mathcal{S}_{h,p}$ and post-multiplication by $\mathcal{S}_{w,q}^T$ give a frame that consists of the $P \times Q$ mapped blocks in DCT domain.

### 2.2. Down-sizing the Mapped Blocks into DCT Blocks

In a real image, most of the signal energy is concentrated in lower frequency range of the DCT coefficients. Hence, a reasonable down-sizing approach is to retain the lower frequency...
components and discard high frequency components [4]. A process to discard high-frequency components and extract only the low-frequency part of size \( 4n \times 4n \) from each one of the \( P \times Q \) number of DCT domain mapped blocks contained in a frame represented by (2) can be expressed by

\[
t_d = \mathcal{T}_{4m \times h, p} \times \left\{ \mathcal{C}_{i,4M}^T \times A_{4M,4N} \times \mathcal{C}_{i,4N} \right\} \times \mathcal{T}_{4n \times w, Q}^T
\]

where \( \mathcal{T}_{4m \times h, p} = \mathcal{F}_{4m \times h, p} \times \mathcal{S}_{h, p} \) is a \( 16M' \times 16M \) matrix and \( \mathcal{T}_{4n \times w, Q}^T \) is a \( 16N \times 16N' \) matrix. \( B_{q \times l, p} \) denotes a \( q \times l \) matrix with \( p \) number of elements of the original frame represented by (2). As most of the signal energy is concentrated in lower frequency range of the DCT coefficients, a reasonable approach to up-sizing is to expand the lower frequency components by zero padding. A process to expand the \( P \times Q \) number of mapped blocks of the DCT frame represented by (2) to the size of \( 4m \times 4n \) each by zero padding the lower frequency components can be expressed by (3). However, a \( q \times l \) matrix \( B \), here, is given by \( B_{q \times l} = \left[ I_q \right] \). Finally, an up-sized frame with \( 4M' \times 4N' \) number of \( 4 \times 4 \) modified DCT blocks is obtained using (4).

\[
\mathcal{T}_{4m \times h, p} = \left( \begin{array}{ccc} L & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & L \end{array} \right) \mathcal{T}_{4n \times w, Q}^T = \left( \begin{array}{ccc} L^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & L^T \end{array} \right)
\]

where \( L = B_{4m \times h} \times S_h \) is a \( 4m \times h \) matrix and \( L^T = S_w^T \times B_{4n \times w} \) is a \( w \times 4n \) matrix. That is, \( L \) is an \( h \)-point DCT matrix vertically truncated to \( 4m \) and \( L^T \) is a \( w \)-point DCT matrix horizontally truncated to \( 4n \).

A down-sized frame represented by (3) consists of \( P \times Q \) real DCT blocks of size \( 4m \times 4n \) each. However, H.264 [2] uses a low-complexity modified \( 4 \times 4 \) DCT. Therefore, we need to convert a down-sized frame as represented by (3) to one with the \( 4 \times 4 \) modified DCT blocks. A down-sized frame with \( 4M' \times 4N' \) number of \( 4 \times 4 \) modified DCT blocks (before post-scaling) can be obtained from (3) as follows:

\[
I_d = \mathcal{C}_{f,4M'}^T \times S_{4m',p}^T \times \mathcal{T}_{4m \times h, p} \times \left\{ \mathcal{C}_{i,4M}^T \times A_{4M,4N} \times \mathcal{C}_{i,4N} \right\} \times \mathcal{T}_{4n \times w, Q}^T \times S_{4n,Q} \times \mathcal{C}_{f,4N'}^T.
\]

The multiplications by \( \mathcal{C}_{i,4M}^T \), \( S_{4m',p}^T \), \( \mathcal{T}_{4m \times h, p} \), \( S_{4n,Q} \) and \( \mathcal{C}_{f,4M} \) matrices can be realized by performing multiplications by their component matrices \( C_i^T(C_i), L(L^T), S^T(S) \) and \( C_f^T(C_f^T) \), respectively.

### Table 1. Computations For Down-sizing When The Original Resolution (width × height) Is 352 × 288.

<table>
<thead>
<tr>
<th>Target Resolution (width × height)</th>
<th>( R_x )</th>
<th>( R_y )</th>
<th>Computations per pixel of the original size</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 × 256</td>
<td>11 : 8</td>
<td>9 : 8</td>
<td>( (4.24, 20.75, 1.65) \times 5.17, 9.82, 1.65 \times (21.98, 24.69, 1.65) )</td>
</tr>
<tr>
<td>256 × 192</td>
<td>11 : 8</td>
<td>3 : 2</td>
<td>( (3.03, 15.09, 1.48) \times 3.88, 8.36, 1.48 \times (16.48, 19.51, 1.48) )</td>
</tr>
<tr>
<td>128 × 192</td>
<td>11 : 4</td>
<td>3 : 2</td>
<td>( (2.36, 12.36, 1.24) \times 1.94, 6.18, 1.24 \times (8.24, 11.76, 1.24) )</td>
</tr>
<tr>
<td>128 × 128</td>
<td>11 : 4</td>
<td>9 : 4</td>
<td>( (2.02, 11.58, 1.16) \times 1.29, 5.45, 1.16 \times (5.49, 9.17, 1.16) )</td>
</tr>
</tbody>
</table>

Up-sizing is basically reverse of the down-sizing operation with a difference that a block \( B_{i,j} \) \( (1 \leq i \leq P = 4M'/m, 1 \leq j \leq Q = 4N'/n) \) of size \( 4m \times 4n \) in the up-sized frame is related to a smaller pixel block, called mapped block, of size \( h \times w \) in the original frame. Note that, the up-sizing factors \( R_x \) and \( R_y \) are less than one, and \( M' \) and \( N' \) are the number of macroblocks along vertical and horizontal axes, respectively, in the up-sized frame. Also, \( m \) and \( n \) are defined as in the case of down-sizing. Again, an original frame can be partitioned into \( P \times Q \) number of mapped blocks of size \( h \times w \) each. Like down-sizing, our approach to arbitrary up-sizing also includes two steps - 1) Extracting \( P \times Q \) number of mapped blocks from the original frame, and 2) Up-sizing each mapped block to a block of size \( 4m \times 4n \) with \( m \times n \) number of modified DCT blocks of size \( 4 \times 4 \) each.

The \( P \times Q \) number of mapped blocks can be extracted from the original frame using (2). As most of the signal energy is concentrated in lower frequency range of the DCT coefficients, a reasonable approach to up-sizing is to expand the lower frequency components by zero padding. A process to expand \( P \times Q \) number of mapped blocks of the DCT frame represented by (2) to the size of \( 4m \times 4n \) each by zero padding the lower frequency components can be expressed by (3). However, a \( q \times l \) matrix \( B \), here, is given by \( B_{q \times l} = \left[ I_q \right] \). Finally, an up-sized frame with \( 4M' \times 4N' \) number of \( 4 \times 4 \) modified DCT blocks is obtained using (4). Table 2 shows computations per pixel of the target resolution (352 × 288) for various up-sizing factors.

It may be noted that, when the horizontal re-sizing ratio \( (R_x) \) and the vertical re-sizing ratio \( (R_y) \) are greater than one then frame down-sizing algorithm in section 2 is employed.
Table 2. Computations For Up-sizing When The Target Resolution (width × height) Is 352 × 288.

<table>
<thead>
<tr>
<th>Original Resolution width × height</th>
<th>$R_x$</th>
<th>$R_y$</th>
<th>Computations per pixel of the target size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>proposed (m,a,s)</td>
</tr>
<tr>
<td>256 × 256</td>
<td>8 : 11</td>
<td>8 : 9</td>
<td>(6.43, 25.03, 1.55)</td>
</tr>
<tr>
<td>256 × 192</td>
<td>8 : 11</td>
<td>2 : 3</td>
<td>(4.71, 20.29, 1.41)</td>
</tr>
<tr>
<td>128 × 192</td>
<td>4 : 11</td>
<td>2 : 3</td>
<td>(3.52, 16.01, 0.98)</td>
</tr>
<tr>
<td>128 × 128</td>
<td>4 : 11</td>
<td>4 : 9</td>
<td>(4.00, 16.75, 0.94)</td>
</tr>
</tbody>
</table>

Fig. 2. Foreman sequence down-sized to 256×192.

Whereas, frame up-sizing algorithm in section 3 is employed when $R_x$ and $R_y$ are less than one. Thus, the frame downsizing operation in horizontal direction and up-sizing operation in vertical direction and vice-a-versa can be considered in a single re-sizing operation using the proposed approach.

4. EXPERIMENTAL RESULTS

The experimental results, are based on our transcoding implementation using JM reference software version 10.2. The Foreman and Coastguard sequences in SIF (352×288) format are encoded with baseline profile using quantization parameter $(Q_P) = 28$ and intra period of 1 $(M=N=1)$. The first 300 frames of these bitstreams are then transcoded to obtain sub-SIF resolution of $256 \times 256 \ (R_x=11/8 \text{ and } R_y=9/8)$ and $256 \times 192 \ (R_x=11/8 \text{ and } R_y=3/2)$ at $Q_P = 28$. Down-sized pictures are up-sized to the original size by using the DCT-domain zero padding, since it introduces no quality enhancement, and compared to the original picture. Table 3 shows the average PSNR (dB) obtained by the proposed algorithm and spatial-domain bilinear and bicubic spline interpolation. Fig 2 shows the PSNR (dB) for individual frames of Foreman sequence down-sized to 288 × 192. Average PSNR obtained is 1.5 dB and 1.67 dB better than bilinear and bicubic interpolation for Foreman sequence (256 × 256), respectively. Similar gain can be seen for Coastguard sequence and 256 × 256 target resolution. Thus, the proposed method presents a better quality picture by preserving the most vital information.

Table 3. Quality For Various Down-sizing Algorithms

<table>
<thead>
<tr>
<th>Method</th>
<th>Coastguard</th>
<th>Foreman</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>256 × 256</td>
<td>256 × 192\</td>
</tr>
<tr>
<td>Proposed</td>
<td>31.28</td>
<td>29.63</td>
</tr>
<tr>
<td>Bicubic</td>
<td>30.35</td>
<td>27.76</td>
</tr>
<tr>
<td>Bilinear</td>
<td>30.16</td>
<td>27.18</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

We have presented a fast algorithm to achieve arbitrary factor re-sizing of H.264 video in DCT space. The proposed algorithm operates on the entire frame as a whole and with the proposed computation scheme offers significant computational reduction. The proposed algorithm can accommodate re-sizing operations with integral as well as rational factors conforming to the syntax of H.264/AVC video.

6. REFERENCES


