Efficient Halving and Doubling 4×4 DCT Resizing Algorithm

James McAvoy, C.D., B.Sc., M.C.S.
ThetaStream Consulting
Ottawa, Canada, K2S 1N5
Email: jimcavoy@thetastream.com

Chris Joslin, Ph.D.
School of Information Technology
Carleton University
Ottawa, Canada, K1S 5B6
Email: chris.joslin@carleton.ca

***Abstract*—We proposed an efficient halving and doubling algorithm in the DCT domain that re-sizes images based on a 4×4 DCT block framework. Our approach improved an existing resizing algorithms through fixed-point integer transforms to reduce computational cost by more than 60% with negligible dB loss. When comparing images that were halved or doubled through bilinear interpolation, our algorithm produced images with similar or higher PSNR values at significantly lower computational cost.**

***Index Terms*—Image Halving and Doubling, DCT Domain resizing.**

I. Introduction

Resizing video or images for storage or transmission is often required because of imposed constraints dictated by network bandwidth or device capabilities. Usually, systems resize video or images in the spatial domain; however, it would be more attractive to resize the them directly in the compressed format, which would avoid high computational overhead associated with decompression and compression operations. Recently, video data is often stored in a compress format based on blocks of 4×4 discrete cosine transform (DCT) coefficients. Video compression standards such as H.264/AVC [1] [2] and H.265/HEVC employ 4×4 fixed-point approximation of the popular DCT to transform frames from the spatial to the frequency domain.

In the early days of developing resizing algorithms in the Discrete Cosine Transform (DCT) domain, researchers devoted effort to image[1] halving and doubling problem. Some of the early contributors in this field were Chang and Messerchmitt [3], and Merhav and Bhaskaran [4] who developed resizing algorithms that exploit linear, distributive and unitary transform properties of DCT. Although image quality was similar and often times superior than resizing in the spatial domain, the computational complexity was almost same as spatial domain resizing techniques.

Dugad and Ahuja [5] proposed a simple fast computation algorithm for image halving and doubling by exploiting the low frequency DCT coefficients. Later, Mukherjee and Mitra [6] proposed modifications to Dugad and Ahuja [5] schema. These modifications improved image quality but increased computational cost. Mukherjee and Mitra [6] observed both of these techniques use subband approximation of DCT coefficients while performing image resizing operations in the frequency domain.

[1]Image and video frames will be used interchangeably in this paper.

Jiang and Feng [7] formulated spatial relationships of the DCT coefficients between a block and sub-blocks. Using their approach one can decompose and recompose blocks of DCT coefficients. Mukherjee and Mitra [8] created several resizing algorithms based on decomposition and re-composition combined with subband approximation. Depending on the ordering of these operations, one can vary the computational cost and the final image quality for a resizing algorithm.

Two of these algorithms were of interest to us in our research. The first was an image halving algorithm, *Image Halving through Approximation followed by Composition* (IHAC) [8] and its inverse doubling algorithm, *Image Doubling through Decomposition followed by Approximation* [8]. Both of these algorithms included conversion matrix to compose and decompose blocks of DCT coefficients when resizing images in the DCT domain. These matrices consist of entries that are approximation of irrational numbers. Although conversion matrices contained several zero entries, matrix multiplication with blocks of DCT coefficients contributed most to the overall computational cost. However, both these algorithms have been shown to provide greater efficiency than resizing in the spatial domain [9] [8].

We proposed to enhance Mukherjee and Mitra image and doubling algorithms by deriving a fix-point approximation of these conversion matrices. We followed a similar approach that the H.264/AVC standard body developed to define the default inverse transform process. H.264/AVC deviated from previous video compression standards by employing a 4×4 fix-point integer transform instead of the popular 8×8 DCT. Malvar et al [10] [11] reported a reduction in complexity with negligible impact on image quality.

This paper will first describe how we enhanced Mukherjee and Mitra IHAC and IDDA algorithms [8] by deriving a fix-point conversion matrix to replace the floating-point conversion matrix in the block composition and decomposition step. Then a section will cover the image quality performance followed by a section on computational cost of our proposed algorithms. The last section contains closing remarks and future directions.

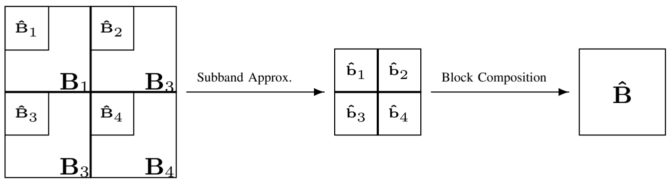


Fig. 1. IHAC [8]. Four 2×2 approximated DCT coefficients of adjacent blocks are composed into one 4×4 DCT block.

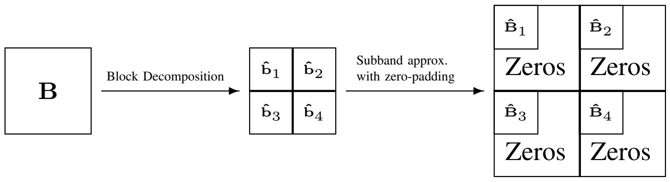


Fig. 2. IDDA [8]. An 4×4 DCT block is decomposed into four $\frac{4}{2} \times \frac{4}{2}$ blocks where each is approximated to an 4×4 DCT block with zero-padding.

II. Improving Image Halving and Doubling Algorithms through Fix-Point Conversion Matrix

Image halving is an operation that takes an image of size $N \times N$ and outputs an image of $N/2 \times N/2$, where image doubling is the inverse operation to resize an image to a resolution of $2N \times 2N$.

Let b an 4×4 block in the spatial domain whose DCT coefficients are encoded as 4×4 block $\mathbf{B}$ in the compressed domain, with elements $B(k,l), k,l = 0,1,2,3$. Generally, to half an image one needs to convert four adjacent DCT blocks, $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ and $\mathbf{B}_4$, to a single 4×4 DCT block, $\hat{\mathbf{B}}$. In Mukherjee and Mitra [8] IHAC algorithm, four 2×2 adjacent blocks are derived from the corresponding 4×4 DCT blocks using subband approximation. Then the 2×2 blocks are recomposed to form an single 4×4 block using an conversion matrix as illustrated in Fig. 1.

To double an image Mukherjee and Mitra [8] employed DCT block decomposition. In the IDDA algorithm, an 4×4 DCT block is first decomposed using conversion matrix to four 2×2 DCT blocks. Then each of these blocks are transformed into an 4×4 DCT block by using subband approximation and zero-padding. Fig. 2 provides a schematic representation of this algorithm.

Let a compressed image be based on a 4×4 DCT block framework. To halve this image the IHAC resizing algorithm would contain the following composition step.

$$\begin{aligned} \mathbf{B}_d &= \mathbf{A}_{(2,2)} \begin{bmatrix} \hat{\mathbf{B}}_{00}^{(2\times2)} & \hat{\mathbf{B}}_{01}^{(2\times2)} \\ \hat{\mathbf{B}}_{10}^{(2\times2)} & \hat{\mathbf{B}}_{11}^{(2\times2)} \end{bmatrix} \mathbf{A}_{(2,2)}^T \\ \mathbf{B}_d &= \mathbf{A}_{(2,2)} \cdot \hat{\mathbf{B}} \cdot \mathbf{A}_{(2,2)}^T \end{aligned} \tag{1}$$

Where $\cdot$ denotes matrix multiplication and the conversion matrix:

$$\mathbf{A}_{(2,2)} = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0.6533 & 0.2706 & -0.6533 & 0.2706 \\ 0 & 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ -0.2706 & 0.6533 & 0.2706 & 0.6533 \end{bmatrix} \tag{2}$$

Note that the rows of $\mathbf{A}_{(2,2)}$ are orthogonal and unit norms, which is a necessary condition for an orthogonal block transform. All the entries in $\mathbf{A}_{(2,2)}$ require processors to approximate irrational numbers. A fixed-point approximation is equivalent to scaling each row of $\mathbf{A}_{(2,2)}$ and rounding to the nearest integer. We chose to multiple $\mathbf{A}_{(2,2)}$ by 2.5 and round, gives $\mathbf{C}_{(\mathbf{2},\mathbf{2})}$:

$$\mathbf{C}_{(2,2)} = \text{round}\left(2.5 \cdot \mathbf{A}_{(2,2)}\right) \tag{3}$$

$$\mathbf{C}_{(2,2)} = \begin{bmatrix} 2 & 0 & 2 & 0 \\ 2 & 1 & -2 & 1 \\ 0 & 2 & 0 & -2 \\ -1 & 2 & 1 & 2 \end{bmatrix} \tag{4}$$

We selected the scaling constant of 2.5 because it was same one that the H.264/AVC designers used to develop their fix-point approximation of 4-point DCT [10] [11].

The row norms of $\mathbf{C}_{(\mathbf{2},\mathbf{2})}$ are $\neq 1$. To restore the orthonormal property of the original matrix of $\mathbf{A}_{(2,2)}$, we multiply all the values of c_{ij} in row r by $\frac{1}{\sqrt{\sum_j c_{rj}^2}}$:

$$\mathbf{A}_{(2,2)} = \mathbf{C}_{(2,2)} \bullet \mathbf{R}_{(2,2)} \tag{5}$$

where

$$\mathbf{R}_{(2,2)} = \begin{bmatrix} 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \\ 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \end{bmatrix} \tag{6}$$

The operator $\bullet$ denotes an element-by-element multiplication. The two-dimensional transform in Equation (1) becomes:

$$\begin{aligned} \mathbf{B}_d &= \mathbf{A}_{(2,2)} \cdot \hat{\mathbf{B}} \cdot \mathbf{A}_{(2,2)}^T && (7) \\ &= [\mathbf{C}_{(2,2)} \bullet \mathbf{R}_{(2,2)}] \cdot \hat{\mathbf{B}} \cdot [\mathbf{C}_{(2,2)}^T \bullet \mathbf{R}_{(2,2)}^T] && (8) \end{aligned}$$

Rearranging to extract the scaling arrays $\mathbf{R}_{(2,2)}$:

$$\begin{aligned} \mathbf{B}_d &= [\mathbf{C}_{(2,2)} \cdot \hat{\mathbf{B}} \cdot \mathbf{C}_{(2,2)}^T] \bullet [\mathbf{R}_{(2,2)} \bullet \mathbf{R}_{(2,2)}^T] && (9) \\ &= [\mathbf{C}_{(\mathbf{2},\mathbf{2})} \cdot \hat{\mathbf{B}} \cdot \mathbf{C}_{(2,2)}^T] \bullet \mathbf{S}_{(2,2)} && (10) \end{aligned}$$

Where

$$\begin{aligned} \mathbf{S}_{(2,2)} &= \mathbf{R}_{(2,2)} \bullet \mathbf{R}_{(2,2)}^T && (11) \\ &= \begin{bmatrix} 1/8 & 1/\sqrt{80} & 1/8 & 1/\sqrt{80} \\ 1/\sqrt{80} & 1/10 & 1/\sqrt{80} & 1/10 \\ 1/8 & 1/\sqrt{80} & 1/8 & 1/\sqrt{80} \\ 1/\sqrt{80} & 1/10 & 1/\sqrt{80} & 1/10 \end{bmatrix} && (12) \end{aligned}$$

Using the new fixed-point approximation of the conversion matrix $\mathbf{C}_{(2,2)}$ with its scaling matrix $\mathbf{S}_{(2,2)}$, we modified the IHAC algorithm. Fig. 3 outlines our new proposed image halving algorithm that takes advantage of these matrices in the block composition step.

```
Input: $4 \times 4$ block based DCT encoded image.
Output: $4 \times 4$ block based DCT encoded downsampled image.
  for every four adjacent $4 \times 4$, $\mathbf{B}_{00}, \mathbf{B}_{01}, \mathbf{B}_{10}, \mathbf{B}_{11}$ of the input image do
    1. Subband approximation: Get corresponding $2 \times 2$ 2-point DCT blocks $\{\hat{\mathbf{B}}_{ij}^{(2\times2)}\}$, $i = 0,1$ and $j = 0,1$ using low-pass truncated approximation, as follows.
       $$\hat{\mathbf{B}}_{ij}^{(2\times2)} = \frac{1}{2}[\mathbf{B}_{ij}(k,l)]_{0\leq k,l\leq 1}$$
    2. Block composition: Convert four $2 \times 2$ DCT blocks to a $4 \times 4$ DCT block $\mathbf{B}_d$ as follows.
       $$\mathbf{B}_d = \left(\mathbf{C}_{(2,2)} \begin{bmatrix} \hat{\mathbf{B}}_{00}^{(2\times2)} & \hat{\mathbf{B}}_{01}^{(2\times2)} \\ \hat{\mathbf{B}}_{10}^{(2\times2)} & \hat{\mathbf{B}}_{11}^{(2\times2)} \end{bmatrix} \mathbf{C}_{(2,2)}^T\right) \bullet \mathbf{S}_{(2,2)}$$
  end for
```

Fig. 3. Our proposed resizing algorithm, IHAC$_{fpa}$, for halving an image based on a 4×4 DCT block framework.

```
Input: $4 \times 4$ block based DCT encoded image.
Output: Upsampled image in the compressed domain.
  for each $4 \times 4$ blocks $\mathbf{B}$ do the following: do
    1. Block decomposition: Convert the block to four $2\times2$ DCT blocks as follows.
       $$\begin{bmatrix} \mathbf{B}_{00}^{(2\times2)} & \mathbf{B}_{01}^{(2\times2)} \\ \mathbf{B}_{10}^{(2\times2)} & \mathbf{B}_{11}^{(2\times2)} \end{bmatrix} = \left(\mathbf{C}_{(2,2)}^T \mathbf{B} \mathbf{C}_{(2,2)}\right) \bullet \mathbf{S}_{(2,2)}$$
    2. Subband approximation and zero padding: Compute the approximate $4 \times 4$-point DCT coefficients from each of $\mathbf{B}_{ij}^{(2\times2)}, i = 0,1$ and $j = 0,1$, low-pass truncated approximation. Form four $4 \times 4$ DCT blocks by zero padding each of them (the high frequency components are assigned to zero $0_2$).
       $$\mathbf{B}_{ij}^{(4\times4)} = 2\begin{bmatrix} \mathbf{B}_{ij}^{(2\times2)} & 0_2 \\ 0_2 & 0_2 \end{bmatrix}$$
  end for
```

Fig. 4. Our resizing algorithm IDDA$_{fpa}$ for doubling images based on a 4×4 DCT block framework.

Similarly, we modified the IDDA algorithm in the decomposition step to leverage the fix-point approximation of the conversion matrix with its scaling matrix. Fig. 4 displays the outline of our proposed fix-point approximation of the IDDA algorithm (IDDA$_{fpa}$).

III. Image Quality Experiments and Observations

We developed three experiments to evaluate image quality performance of our proposed fixed-point approximation of the IHAC and IDDA algorithms. In the first experiment, for each image in the sample set I_{orig} of size $N \times N$ is first spatially downsampled using MATLAB bicubic interpolation with anti-aliasing to an image I_d of size $N/2 \times N/2$. These halved images provided a reference so PSNR can be computed when comparing images produced from various halving algorithms.

TABLE I
Experiment 1. PSNR values from image halving experiment.

Image	IHS	IHAC	IHAC$_{fpa}$
Fishing Boat	39.95	41.98	41.65
Cameraman	40.48	46.09	45.47
Elaine	45.14	45.86	45.97
Goldhill	42.39	43.62	43.36
House	46.50	50.47	50.92
Jetplane	40.36	43.49	43.22
Lake	39.13	42.66	42.39
Lena	42.00	45.62	45.45
Livingroom	40.06	42.00	41.69
Mandril	36.60	36.38	36.20
Peppers	42.72	44.25	44.27
Pirate	40.84	43.23	43.06
Walking Bridge	37.90	39.70	39.39
Watch	36.70	41.03	40.76
Woman Blonde	41.74	42.33	42.21
Woman Darkhair	48.76	51.15	51.61
mean(PSNR)	41.33	43.74	43.60

The original images I_{orig} in the sample set are raw grayscale (8 bits/pixel) images with a resolution of 512×512 pixels. In all the experiments, we applied MATLAB `dct2` function to transform all the images I_{orig} from the spatial to the DCT domain to represent compressed images formatted as blocks of 4×4 DCT coefficients. We applied the DCT resizing algorithms on the compressed I_{orig} and outputted a compressed halved image. Using MATLAB `idct2`, we transformed these newly compressed halved images back into the spatial domain where we computed an PSNR with the reference halved images. We also compared our proposed halving algorithm with halved images that were resized in the spatial domain by bilinear interpolation, which will be referred as IHS. Table I shows the PSNR values computed from comparing the halved images generated from our proposed halving algorithm with IHAC and IHS.

For the second experiment to evaluate image quality performance of our IDDA$_{fpa}$ algorithm, we first spatially downsampled image I_{orig} to create an image I_d that are compressed then upsampled in the DCT domain. We used MATLAB bicubic interpolation with anti-aliasing to halve images in the spatial domain to produce image I_d. Using MATLAB `idct2` function, we transformed the spatially halved image I_d to the DCT domain to represent a compressed image formatted as blocks of 4×4 DCT coefficients. Then we applied the doubling algorithms to resize and output compressed image to the original resolution. We transformed the compressed doubled image back into the spatial domain using MATLAB `idct2` function so a PSNR value can be computed between the upsampled image with the original image I_{orig}. Similar in the first experiment, we doubled the halved image I_d in the spatial domain using bilinear interpolation for comparison, which will be referred as IDS in the paper. Table II displays the result of this experiment.

In the third experiment, an image is first halved and then doubled. The resulting upsampled image is compared with the original image I_{orig}. We compared the PSNR value

TABLE II
EXPERIMENT 2. PSNR VALUES FROM IMAGE DOUBLING EXPERIMENT.

Image	IDS	IDDA	IDDA$_{fpa}$
Fishing Boat	28.94	29.40	29.11
Cameraman	33.21	33.68	32.91
Elaine	32.53	32.75	32.50
Goldhill	30.62	43.62	43.36
House	41.72	41.18	39.66
Jetplane	30.12	30.64	30.25
Lake	29.34	29.77	29.36
Lena	32.69	33.17	32.72
Livingroom	28.58	28.93	28.69
Mandril	23.05	23.48	23.37
Peppers	31.18	31.59	31.28
Pirate	29.99	30.43	30.14
Walking Bridge	26.23	26.70	26.47
Watch	27.11	27.55	27.14
Woman Blonde	28.96	29.37	29.18
Woman Darkhair	39.83	40.04	39.44
mean(PSNR)	30.88	31.23	30.81

TABLE III
EXPERIMENT 3. PSNR VALUES FROM IMAGE HALVING FOLLOWED BY IMAGE DOUBLING.

Image	IHS IDS	IHAC IDDA	IHAC$_{fpa}$ IDDA$_{fpa}$
Fishing Boat	27.99	28.64	29.37
Cameraman	31.24	33.94	33.16
Elaine	31.89	32.96	32.64
Goldhill	29.81	31.27	31.02
House	38.65	41.73	39.68
Jetplane	29.02	30.88	30.47
Lake	28.15	30.00	29.57
Lena	31.41	33.43	32.92
Livingroom	27.71	29.15	28.93
Mandril	22.52	23.71	23.62
Peppers	30.35	31.83	31.44
Pirate	29.04	30.66	30.36
Walking Bridge	25.43	26.92	26.71
Watch	25.88	27.75	27.33
Woman Blonde	28.34	29.59	29.40
Woman Darkhair	38.43	40.39	39.48
mean(PSNR)	29.74	31.49	31.01

computed from images generated from resizing algorithms used in tandem: spatial resizing using IHS-IDS, IHAC-IDDA and our proposed halving and doubling algorithm. The results of this experiment are shown in Table III.

When comparing image quality, our proposed halving and doubling algorithms produced images with slightly lower PSNR values compared to its floating-point implementation (IHAC), which was expected because of rounding errors associated with integer transforms with scaling. For the first experiment, our IHAC$_{fpa}$ algorithm generated images that were 0.14 dB lower than ones generated from IHAC algorithm. However, images generated from our halving algorithm were 2.27 dB higher than IHS images over our sample set.

For image doubling experiment, our IDDA$_{fpa}$ algorithm produced images with a PSNR value of 0.42 dB lower on average than images doubled from IDDA algorithm. However, on average our doubling approach created images only 0.07 dB lower than images that were spatially resized. In the third experiment, when using our proposed algorithms in tandem, the PSNR values was 0.48 dB lower than the IHAC-IDDA pairing but we observed an 1.27 dB gain over the spatial approach implemented using IHS-IDS pairing. These limited experiments demonstrated to us that our algorithms could produced images equal or better than resizing images in the spatial domain using bilinear interpolation while benefiting from the computational savings derived from our approach. The next section will discuss how computationally efficient our algorithms are.

TABLE IV
COMPUTATIONAL COMPLEXITY OF HALVING ALGORITHMS.

ops	IHS	IHAC	IHAC$_{fpa}$
n_m	1.25	0.5	0.1875
n_a	8.25	1	0.25

IV. COMPUTATIONAL COST

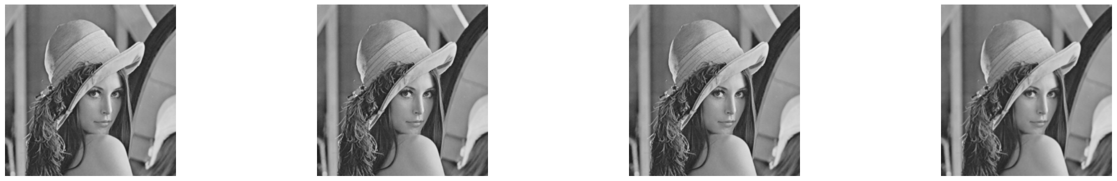
This section outlines the computational cost of our proposed resizing algorithms. By using a fix-point approximation of a conversion matrix in our proposed algorithms, we hoped to decrease computational cost. Leveraging fixed-point arithmetic, multiplying or dividing values that are a power of two can be accomplished by binary shift operations. You will observe that the conversion matrix $\mathbf{C}_{(2,2)}$ entries are either 0, 1's or 2's. Therefore, matrix multiplications can be carried out *multiplier-free*.

Let n_m be number of multiplications and n_a be number of additions. The IHAC$_{fpa}$ algorithm first does subband approximation and multiplies each element in the input 4×4 DCT coefficients by half, which can be implemented as a right shift operation; thus no cost. The composition step contains two matrix multiplications by applying the conversion matrix $\mathbf{C}_{(2,2)}$ on input DCT coefficients twice. Since matrix multiplication with $\mathbf{C}_{(2,2)}$ can be carried out multiplier-free, only additions count, which there are $n_a = 8$. An element-by-element multiplication is applied with scaling matrix $\mathbf{S}_{(2,2)}$. The scaling matrix $\mathbf{S}_{(2,2)}$ contains elements equal to $1/8$, which can be implemented as a right shift operation; thus $n_m = 12$ and $n_a = 0$. Finally, the four 4×4 input DCT blocks represent 64 pixels of the input image. Therefore, on average our method will consume $n_m = 0.1875$ and $n_a = 0.25$ per pixel of the original image.

Similarly, the our IDDA$_{fpa}$ algorithm contains two matrix multiplication using the conversion matrix $\mathbf{C}_{(2,2)}$. Also, scaling matrix and subband approximation are similar. Hence, our proposed doubling algorithm requires $n_m = 0.1875$ and $n_a = 0.25$ per pixel of the upsampled image.

Table IV compares computational complexity for the IHAC$_{fpa}$ algorithm with other image halving algorithms. Table V provides the complexity of IDDA$_{fpa}$ algorithm with other image doubling algorithms.

Both of our proposed algorithms is more efficient than their floating-point implementations. When comparing our proposed halving algorithm, IHAC$_{fpa}$, with its floating-point implementation, it was 63% more efficient in n_m and 75% in



(a) I_d (b) IHAC$_{fpa}$ (c) IHAC (d) IHS

Fig. 5. Resultant images downsampled in experiment 1. Figure 5a is an image halved through bicubic interpolation with anti-aliasing, which is the reference image so PSNR can be computed. Figure 5b is an image halved by our proposed algorithm, IHAC$_{fpa}$, with PSNR value of 45.45 dB. Figure 5c was downsampled through floating-point implementation of the IHAC algorithm, which provide a PSNR value of 45.62 dB. Figure 5d was a spatial downsampled image using bilinear interpolation that produced a PSNR value of 42.00 dB.

TABLE V
COMPUTATIONAL COMPLEXITY OF DOUBLING ALGORITHMS.

ops	IDS	IDDA	IDDA$_{fpa}$
n_m	1.25	2	0.1875
n_a	8.25	1.5	0.25

n_a. As for our proposed doubling algorithm, IDDA$_{fpa}$, it was 91% more efficient in n_m and 83% in n_a when comparing with its floating-point version, IDDA. When comparing both of our proposed algorithms with spatial resizing, the computational saving was 85% in n_m and 97% in n_a.

V. Conclusion

Several resize algorithms employed conversion matrices to compose or decompose blocks of DCT coefficients in their implementation. We developed an fixed-point approximation of a conversion matrix that is included in IHAC and IDDA resizing algorithms. Matrix multiplications with fixed-point approximation of conversion matrices are *multiplier-free*, which substantial reduced computational cost in our proposed resizing algorithms. Images generated from our fix-point resizing algorithms provided PSNR values negligibly lower than their floating-point implementations. Systems utilizing our proposed low-complexity algorithms will produce images with good quality at a greatly reduced computational cost compare to resizing images in the spatial domain.

Presently, our proposed low-complexity resizing algorithms only does image halving and doubling, and only on conversion matrices of size $\mathbf{A}_{(2,2)}$. The approach of creating fix-point approximation of these conversion matrices can be extended so one can resize images by integral or arbitrary factors. Mukherjee and Mitra [8] developed several resizing algorithms that use conversion matrices for composition and decomposition such as IHCA, IDAD, LMDS and LMUS, which would be good candidates for this approach. This would be a logical avenue for future research.

References

[1] *Advanced video coding for generic audiovisual services*, International Telecommunication Union Std. H.264, Rev. 16, January 2012. [Online]. Available: http://www.itu.int/rec/T-REC-H.264

[2] *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*, ISO Std. ISO/IEC 14 496-10:2012, January 2012. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=61490

[3] S.-F. Chang and D. G. Messerchmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, January 1995.

[4] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image down-sampling and for inverse motion compensation," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 486–476, June 1997.

[5] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, April 2001.

[6] J. Mukherjee and S. K. Mitra, "Image resizing in the compressed domain using subband DCT," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 620–627, July 2002.

[7] J. Jiang and G. Feng, "The spatial relationship of DCT coefficients between a block and its sub-blocks," *IEEE Transaction on Signal Processing*, vol. 50, no. 5, pp. 1160–1169, May 2002.

[8] J. Mukherjee and S. K. Mitra, "Arbitrary resizing of images in the DCT space," in *IEE Proceeding Vision, Image and Signal Processing*, vol. 152, 2005, pp. 155–164.

[9] J. Mukhopadhyay and S. Mitra, "Resizing of images in the DCT space by arbitrary factors," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 4, October 2004, pp. 2801–2804.

[10] A. Hallapuro, M. Karczewicz, and H. S. Malvar, "Low-complexity transform and quantization," in *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-VCEG*, January 2002.

[11] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Transaction Circuits and Systems Video Technology*, vol. 13, pp. 598–603, July 2003.