Resizing of Images by Arbitrary Factors in the Spatial Domain and Implementation on Blackfin BF533 Processor

Ajay A.V [*], Deepti S.M [*], Rajendra C.Y [*], Venkatesan N.E[*] and Dr Sumam David [*], *Senior Member IEEE*

[*] Department of Electronics & Communication Engg.,
National Institute of Technology Karnataka, Surathkal, INDIA
email: sumam@ieee.org

Keywords: Image resizing, Linear transformation, DCT, ADSP BF533.

Abstract

A novel approach for resizing of images in the spatial domain is presented. Image transformations are seen from a Euclidean space perspective and expressions for image resizing are derived. The proposed method can be used for resizing by any arbitrary rational ratio L/M. The algorithm has been tested for resizing between standard formats like NTSC, QVGA, QCIF and CIF and the performance is compared with other algorithms in terms of PSNR and computational efficiency. Real time implementation on Analog Devices Blackfin-BF533 DSP processor shows promise.

1 Introduction

Image resizing finds application mainly in digital displays. With the advent in digital display technology, there have been a variety of display devices like digital TV, digital cameras, handy cams, mobile phones etc. having variable screen sizes [3]. Even among the same family of devices, the size of the display unit differs. When the same data needs to be displayed in different devices, it needs to be resized. The constraint is more so in the case of video applications as the processing time available is very less. Most of the present algorithms [2, 6], deal with resizing by factors that are powers of two or integral factors. However, certain applications demand resizing by rational factors. Most of the present algorithms for resizing, are implemented in the compressed domain [2, 4, 5, 6]. This is advantageous while processing stored images and video as they are in the compressed form. However, in real time applications like a handy cam, the video frames generated will be in the uncompressed format and these algorithms are not efficient. Also, most of the algorithms are based on resizing both the rows and columns by equal factors. However, in certain video format conversions it maybe required to resize the columns and rows by different factors.

Various techniques like fast computation technique for image halving and doubling in the frequency domain [2], L/M-Fold Image Resizing in Block-DCT Domain Using Symmetric Convolution [6], and arbitrary resizing using DCT sub-band approximation [4, 5] are currently available for resizing of images in spatial and DCT domains.

In this paper, we present a computationally efficient approach for resizing of images by rational factors in the spatial domain based on the results of Dugad and Ahuja [2]. Our method eliminates the need for a low pass filter during up sampling and down sampling, thereby reducing the number of computations. The resizing is accomplished in a single step without passing through the steps of up sampling and down sampling. We look at the images from an Euclidean space perspective [7] and derive transformations for image resizing. The proposed algorithm supports resizing by different scaling factors along rows and columns.

Section 2 explains how images can be seen from an Euclidean space perspective and the linear transformation required for moving from the spatial domain to the transformed domain. Section 3 gives a brief overview of 2-D DCT and its matrix representation. The proposed algorithm for resizing is presented in Sections 4 and 5. Section 6 compares the performance of the proposed algorithm with standard resizing techniques and implementation issues are discussed in Section 7.

2 Images seen as a vector in Euclidean Space

A vector by definition is an ordered set of numbers. The numbers could be either complex or real. Depending on this we define whether the vector is defined over a real field $\mathcal{R}$ or a complex field $\mathcal{C}$. A collection of such vectors closed under addition and scalar multiplication constitutes a vector space. A vector space closed under the operation of dot product is a Euclidean space. A gray scale image of size $m \times n$ can be viewed as a mn-D vector defined over $\mathcal{R}$. We consider Gray scale images as the results derived hold good for color images when extended to each of the colour planes.

A vector space of dimension mn can be described as the set of all $m \times n$ images in this context. Let us denote this vector space as I_{MN}. The pixel values are constrained to lie between 0 and 255. Image resizing can be thought of as a linear transformation from one vector space (I_{MN}) to another vector space (I_{PQ}) where M, N, P, Q are arbitrary integers. mn is the dimension of the original vector space and pq is the dimension of the vector space containing the resized images.

Linear transformation can be viewed as a matrix multiplication. Therefore, if X_{MN} is a vector, i.e. an image in this case, in I_{MN} and Y_{PQ} is a vector in I_{PQ} then we have

$$Y_{PQ} = A^{pre}_{PM} X_{MN} A^{post}_{NQ} \tag{1}$$

where A^{pre}_{PM} and A^{post}_{NQ} are the *pre* and *post* transformation matrices.

Hence, if we know the transformation matrices we can convert any vector in I_{MN} space to the corresponding vector in I_{PQ} space. Since the transformation matrices are common for all the vectors, they can be pre-computed and stored in memory for conversions between standard formats.

3 Discrete Cosine Transform of Images

The DCT of a 2-D image X(m,n); 0≤m≤M-1, 0≤n≤N-1, is given by

$$\begin{aligned} C_{MN}(k,l) &= \frac{2}{\sqrt{MN}}\alpha(k)\alpha(l)\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X(m,n) \\ &\quad cos\left(\frac{(2m+1)\pi k}{2M}\right) cos\left(\frac{(2n+1)\pi l}{2N}\right) \\ &\quad 0 \le l \le M-1, 0 \le k \le N-1 \end{aligned} \tag{2}$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{2}} & \text{for u = 0} \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

Thus for a image block of size $M \times N$, we get $M \times N$ DCT coefficients. We exploit the energy compaction property of DCT transformation for resizing. Consider that we have a $M \times N$ DCT coefficient block. Suppose that we remove the last *i* rows and last *j* columns. We then get a DCT matrix of size $P \times Q$ where $P = M - i$ and $Q = N - j$. If we apply IDCT to this coefficient block, we will get an image of size $P \times Q$. This image is an approximation of the original image. Similarly if we had appended *i* null rows and *j* null columns, we would have got an image of size $P \times Q$, where $P = M + i$ and $Q = N + j$. In this case, we can view the original image to have been an approximation of the resized larger image. However, since we are not adding more information to the image, both the images essentially contain the same information. We shall use these concepts as a basis for our resizing algorithm.

4 Determination of Transformation matrices

The 2-D DCT of an image can be obtained by multiplying the image with a pre and post matrix which are denoted by D^{pre}_{MM} and D^{post}_{NN} respectively. The matrices can be calculated as follows.

$$D^{pre}_{MM}(i,m) = \frac{2}{\sqrt{M}}\alpha(i) cos\left(\frac{(2m+1)\pi i}{2M}\right) \tag{4}$$

$$D^{post}_{NN}(n,j) = \frac{2}{\sqrt{N}}\alpha(j) cos\left(\frac{(2n+1)\pi j}{2N}\right) \tag{5}$$

Thus the DCT of an image X_{MN} is given by

$$C_{MN} = D^{pre}_{MM} \times X_{MN} \times D^{post}_{NN} \tag{6}$$

The resizing in the DCT domain is done by multiplying C_{MN} by a pre and post matrix respectively denoted by R^{pre}_{PM} and R^{post}_{NQ} .

$$G_{PQ} = R^{pre}_{PM} \times C_{MN} \times R^{post}_{NQ} \tag{7}$$

These matrices for various cases can be computed as follows.

$$R_{IJ} = \begin{cases} [I_{II} \mid Z_{I(J-I)}] & \text{if } I < J \\ I_{II} & \text{if } I = J \\ \left[\frac{I_{JJ}}{Z_{(I-J)J}}\right] & \text{if } I > J \end{cases} \tag{8}$$

where I is the identity matrix and Z is the matrix of zeros.

Once resizing is done, we can obtain the resized image by taking the IDCT. The matrices for obtaining the IDCT are given by

$$ID^{pre}_{PP}(i,p) = 2\sqrt{P}\alpha(i) cos\left(\frac{(2p+1)\pi i}{2P}\right) \tag{9}$$

$$ID^{post}_{QQ}(q,j) = 2\sqrt{Q}\alpha(j) cos\left(\frac{(2q+1)\pi j}{2Q}\right) \tag{10}$$

Thus the resized image Y_{PQ} is obtained by multiplying the resized DCT coefficient matrix with the IDCT matrices.

$$Y_{PQ} = ID^{pre}_{PP} \times G_{PQ} \times ID^{post}_{QQ} \tag{11}$$

Combining the above equations we get

$$Y_{PQ} = ID^{pre}_{PP} \times R^{pre}_{PM} \times D^{pre}_{MM} \times X_{MN} \times D^{post}_{NN} \times R^{post}_{NQ} \times ID^{post}_{QQ} \tag{12}$$

Comparing this equation with Equation (1) we get

$$A^{pre}_{PM} = ID^{pre}_{PP} \times R^{pre}_{PM} \times D^{pre}_{MM} \tag{13}$$

$$A^{post}_{NQ} = D^{post}_{NN} \times R^{post}_{NQ} \times ID^{post}_{QQ} \tag{14}$$

Once the required transformation is known we can calculate the A^{pre}_{PM} and A^{post}_{NQ} matrices and apply them to the image. The equations required for computation of A^{pre}_{PM} and A^{post}_{NQ} can be calculated from the above set of equations. They are given as follows

$$A^{pre}_{PM}(i,j) = \sum_{k=0}^{R-1} cos\left(\frac{(2i+1)\pi k}{2P}\right) cos\left(\frac{(2j+1)\pi k}{2M}\right) \alpha^2(k) \tag{15}$$

$$A^{post}_{NQ}(i,j) = \sum_{k=0}^{C-1} cos\left(\frac{(2i+1)\pi k}{2Q}\right) cos\left(\frac{(2j+1)\pi k}{2N}\right) \alpha^2(k) \tag{16}$$

where $R = min(M,P)$ and $C = min(N,Q)$.

The equation for the intermediate matrix T_{PN} obtained after multiplying X_{MN} with the pre matrix is given by

$$T_{PN}(i,j) = \sum_{i=0}^{P-1} \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} A_{PM}^{pre}(i,k) X_{MN}(k,j) \qquad (17)$$

The final resized image matrix is given by

$$Y_{PQ}(i,j) = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} \sum_{k=0}^{N-1} T_{PN}(i,k) A_{NQ}^{post}(k,j) \qquad (18)$$

It should be noted that the above transformation is not always invertible. Invertibility exists only when $P \geq M$ and $Q \geq N$ where the resized image is an interpolated version of the original image. As there is no loss of information and thus we can exactly reconstruct the original image by decimation using suitable factors. However, when either $P < M$ or $Q < N$ there is a loss of information and the original image cannot be exactly reconstructed. From the point of view of linear transformations, it can be said that the transformation is not orthogonal in nature as the size of the vector is changing during the transformation.

5 Decomposition and recomposition of images into spatial blocks

The spatial correlation reduces when a image of large block size is taken. Hence, we split the image into smaller blocks and then apply the algorithm to each block as shown in Fig. 1.

Pseudo code
Input: M × N image
Output: P × Q image

- ▷ Compute $R = HCF(M,P)$ and $C = HCF(N,Q)$
- ▷ Compute input block size (R_i, C_i)
  $R_i = M/R \ \ and \ \ C_i = N/C$
- ▷ Compute output block size (R_o, C_o)
  $R_o = P/R \ \ and \ \ C_o = Q/C$
- ▷ Compute the pre matrix $A_{R_o R_i}^{pre}$ and post matrix $A_{C_i C_o}^{post}$ for conversion of input block to output block
- ▷ Multiply each input block with the obtained matrices and arrange output blocks to obtain the resized image.

The resizing results obtained for different output sizes are as shown in Fig. 2.

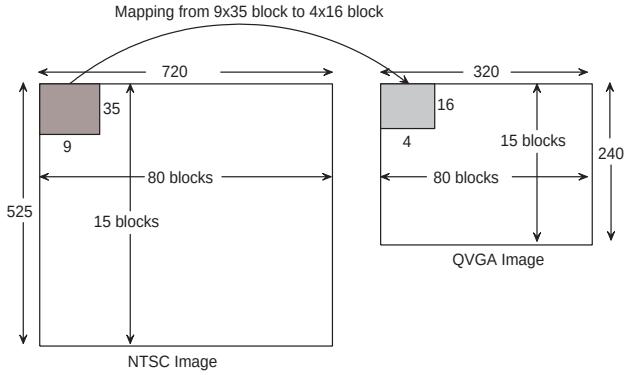

Figure 1: Block processing of image

a

b *c* *d*

Figure 2: Image resizing *(All images are scaled to 30%)*
(a) Original NTSC (525 × 720) image
(b) CIF (352 × 288) image from QCIF
(c) QVGA (240 × 320) image from NTSC
(d) QCIF (176 × 144) image from QVGA

Algorithm/ Images	Flowers	Cameraman	Lena	Peppers
Nearest	29.88	29.01	32.48	34.01
Bilinear	35.71	32.13	32.62	37.69
Bicubic	38.57	34.05	33.36	39.41
Proposed Method	42.31	39.11	43.38	41.47

Table 1: PSNR values (dB) for conversion of images from NTSC to QVGA

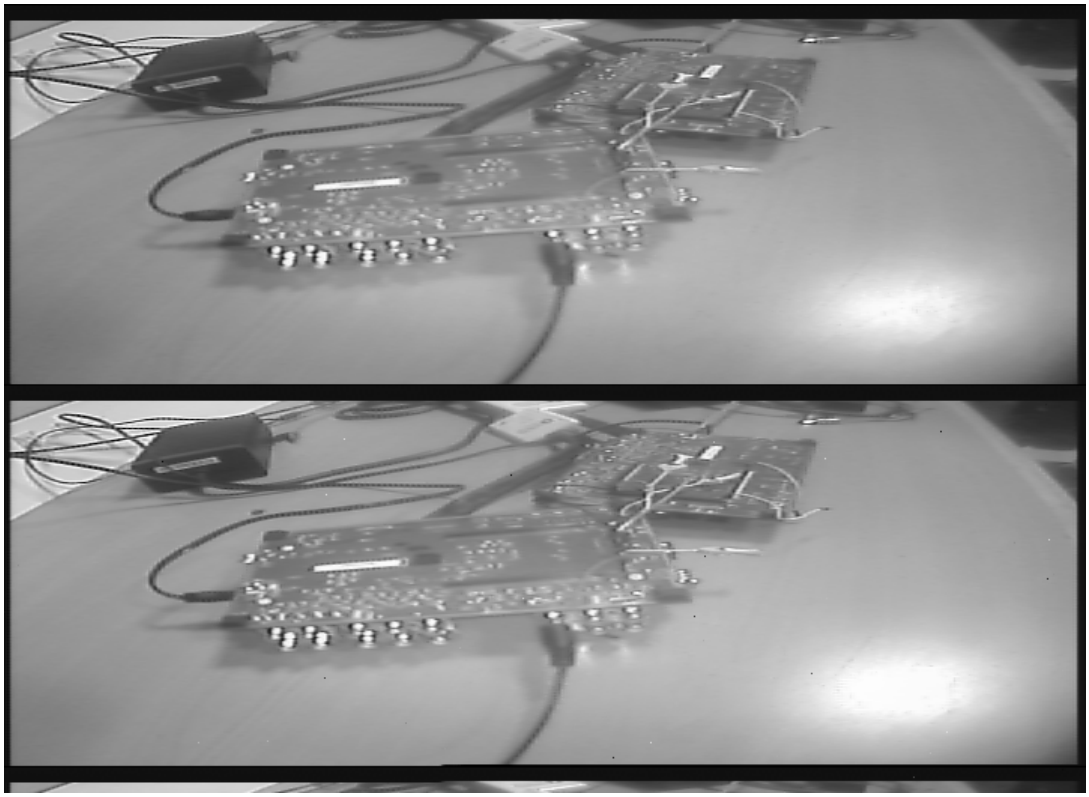

Figure 3: Original NTSC image showing SPORT transfer between two ADSP BF533 EZ kits scaled to 75 %

6 Comparison with Standard Algorithms

The performance of the proposed resizing algorithm was compared with some of the standard algorithms through simulation.

6.1 PSNR Calculation

The PSNR was computed similar to the down sample-up sample PSNR method used by Mukherjee and Mitra [4] for resizing with arbitrary factors. We first resized the image to the required format (Y_{PQ}) and then reconverted it back to the original image size (X'_{MN}). The mean square error (MSE) between the original image and the image obtained after the two resizing operations is found and the PSNR is calculated as

$$PSNR = 10log\left(\frac{[255]^2}{MSE}\right) dB \tag{19}$$

The PSNR values obtained for NTSC to QVGA conversion using our algorithm and standard MATLAB image resizing function is given in Table 1. It can be observed that the proposed method gives higher PSNR for all the images. The subjective quality of all the images was better. The above method of PSNR calculation is not suitable for the case $P \geq M$ and $Q \geq N$ as MSE is zero.

6.2 Computational Requirements

The resizing operation involves matrix multiplications. Let the cost of a multiplication be ρ and that of an addition be μ. The computational cost Γ for resizing a MN image to a PQ image is given by

$$\begin{aligned}\Gamma &= [(R_oR_iC_i + R_oC_iC_o)\rho + (R_oC_i(R_i - 1) + \\ &\quad R_oC_o(C_i - 1))\mu]RC\end{aligned} \tag{20}$$

The computational cost per pixel Γ_n is given by

$$\Gamma_n = \frac{\Gamma}{R_iC_iRC} \tag{21}$$

7 Implementation on ADSP Blackfin-533 EZ kit

ADSP BF-533 is an enhanced DSP processor from the Blackfin family from Analog Devices [1]. The Blackfin processor core architecture combines a dual MAC signal processing engine, an orthogonal instruction set, flexible SIMD capabilities, and multimedia features into a single instruction set architecture. Blackfin features dynamic power management. The ability to vary both the voltage and frequency of operation optimises the power consumption profile to the specific task. The ADSP BF-533 EZ-kit has one Video-In and one Video-Out port, but, at a

time, only one of these can be used. Hence, for real time video processing two EZ-kits are required.

The NTSC image captured from the digital camera connected to the Video-In port of EZ-kit is stored in the external memory (SDRAM). The internal memory (32 KB) is not sufficient to store the entire image. For real time operation, as it is not possible to access the image pixel values at a fast data-rate from the external memory we use the *Ping-Pong* DMA approach. The internal memory is split into two banks. DMA is initialised to load a block of the image into one bank while processing is done on the other bank. The processed image is then sent to the SPORT (also a DMA transfer) for serial transfer between the two kits and image is displayed through the video port of the second kit.

Pseudo Code

- ▷ Load image frame into external memory
- ▷ Initialize DMA transfer for Block A
- ▷ While (all the blocks are processed)
  - – Wait until DMA transfer is complete
  - – *If (odd cycle)*
    Initialize DMA for Block B, Process on Block A
    Else
    Initialize DMA for Block A, Process on Block B
  - – Initialize DMA for transfer through SPORT
- ▷ End loop

The algorithm was tested for a single image frame grabbed from the digital camera interfaced to the EZ-kit. The original NTSC image showing port transfer between the EZ-kits is given in Fig. 3 and Fig. 4 shows the resized image for NTSC to QCIF conversion. The resizing algorithm had a code size of 4464 bytes and was executed in 457867 cycles. The pre and post matrices for NTSC to QCIF conversion occupied 948 bytes. The memory requirement and execution time can be improved by assembly language code optimisations.

8 Conclusions

In this paper, we have discussed the resizing of images by arbitrary factors in the spatial domain. The algorithm has been implemented on Analog Devices BlackFin BF533 processor and has been applied for standard video format conversions such as NTSC, QVGA, QCIF and CIF. For RGB images it needs to be applied to all the three colour planes. The algorithm finds promise in real time image format conversions especially in the case of video streaming as the conversion can be done in spatial domain.

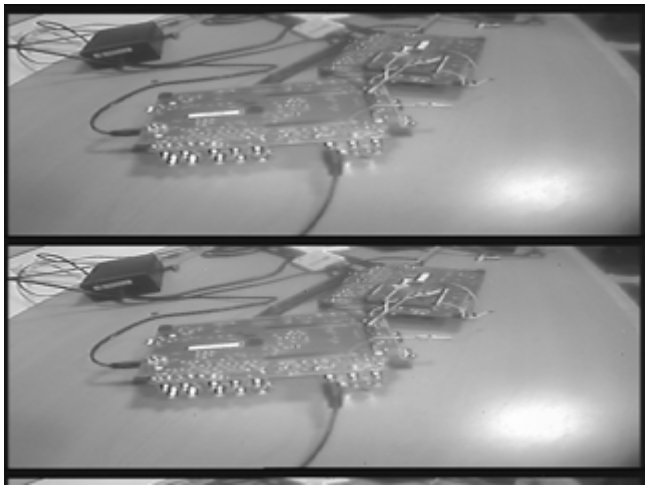

Figure 4: QCIF image obtained from NTSC image

Acknowledgments

The authors gratefully acknowledge Analog Devices University program and Mr Kunal Singh, Mr Gurudath N.V., and Mr Rupjyoti Sarmah of ADI Bangalore for their support to this technical work.

References

[1] ADSP-BF533 Blackfin Processor Hardware Reference, *Analog Devices Inc.* (2005)

[2] R. Dugad, N. Ahuja. “A Fast Scheme for Image Size Change in the Compressed Domain”, *IEEE Trans. on Circuits and Sys. for Video Tech.*, **11**, pp.461–474, (2001)

[3] K. Jack. “Video Demystified”, *LLH Tech. Pub.* (2001)

[4] J. Mukherjee, S. K. Mitra. “Arbitrary Resizing of Images in DCT space”, *IEE Proc.-Vis. Image Signal Process.*, **152**, pp.157–164, (2005).

[5] J. Mukhopadhyay, S.K.Mitra. “Resizing Of Images in the DCT space by Arbitrary factors”, *Intl. Conf. on Image Processing ICIP*,pp.2801–2804, (2004).

[6] H. W. Park, Y. S. Park, S. K. Oh. “L/M-Fold Image Resizing in Block-DCT Domain Using Symmetric Convolution”, *IEEE Trans. on Image Processing*, **12**, pp.1016–1034 (2003).

[7] G. Strang. “Linear Algebra and Its Applications”, *Thompson Books/Cole*, (2005)

[8] C. Wang, H. B. Yu, M. Zheng. “A Fast Scheme for Arbitrarily Resizing of Digital Image in the Compressed Domain” *IEEE Trans. on Consumer Electronics*, **49**, pp.466–471 (2003)