How fast can one arbitrarily and precisely scale images?

Leonid Bilevich*[a] and Leonid Yaroslavsky**[a]
[a]Department of Physical Electronics, Faculty of Engineering,
Tel Aviv University, 69978, Tel Aviv, Israel[1]

ABSTRACT

Image scaling is a frequent operation in video processing for optical metrology. In the paper, results of comparative study of computational complexity of different algorithms for scaling digital images with arbitrary scaling factors are presented and discussed. The following algorithms were compared: different types of spatial domain processing algorithms (linear, cubic, cubic spline interpolation) and a new DCT-based algorithm, which implements perfect (interpolation error free) scaling through discrete sinc-interpolation and is virtually free of boundary effects (characteristic for the DFT-based scaling algorithms). The comparison results enable evaluation of the feasibility of real-time implementation of the algorithms for arbitrary image scaling.

Keywords: scaling, DCT, convolution, fast algorithm

1 INTRODUCTION

Scaling is one of basic image processing tasks with a wide range of applications. Conventional approaches to scaling utilize low order spline interpolation (linear/cubic/spline), which does not secure interpolation error free scaling. Discrete sinc interpolation methods in DFT domain[1] do secure interpolation error free scaling but suffer for boundary effects associated with the cyclicity property of the discrete sinc-function. The DCT-based scaling algorithm introduced in this paper also secures perfect interpolation error free image arbitrary scaling and is, in addition, virtually free of boundary effects.

In video processing, a crucial issue is its computational complexity. Generally, there is a trade-off between the accuracy and computational complexity. In this paper, we, along with introducing a new improved DCT-based method of image arbitrary scaling, compare its computational complexity with the complexity of spatial domain interpolation methods.

2 SIGNAL DOMAIN CONVOLUTION BASED INTERPOLATION METHODS

The most commonly used signal domain convolution based interpolation methods are described by the equation:

$$\tilde{a}_k = \sum_{n=0}^{N-1} a_n h\left(\frac{k}{\sigma} - n\right), \tag{1}$$

where a_n are signal samples, $\tilde{a}_k$ is interpolated signal and $h(n)$ is interpolation kernel. One of the simplest methods is linear interpolation, for which

$$h(n) = \begin{cases} 1-|n| & \text{if} \quad 0 \le |n| < 1 \\ 0 & \text{if} \quad 1 \le |n| \end{cases}. \tag{2}$$

[1] *bilevich@eng.tau.ac.il; http://www.eng.tau.ac.il/~bilevich/
**yaro@eng.tau.ac.il; http://www.eng.tau.ac.il/~yaro /

The more accurate interpolation method is cubic interpolation[2], for which

$$h(n)=\begin{cases}(q+2)|n|^3-(q+3)|n|^2+1 & \text{if} \quad 0\le|n|<1\\ q|n|^3-5q|n|^2+8q|n|-4q & \text{if} \quad 1\le|n|<2\\ 0 & \text{if} \quad 2\le|n|\end{cases} \tag{3}$$

and $q=-0.5$.

Even more accurate interpolation method is cubic spline interpolation[3-5] defined as:

$$\tilde{a}_k=\sum_{n=0}^{N-1}\sum_{m=0}^{3}b_n^{(m)}h^{(m)}\left(\frac{k}{\sigma}-n\right). \tag{4}$$

where

$$h^{(m)}(n)=|n|^m\cdot\begin{cases}1 & \text{if} \quad 0\le|n|<1\\ 0 & \text{if} \quad 1\le|n|\end{cases}. \tag{5}$$

The coefficients $b_n^{(m)}$ are determined as follows:

$$\begin{pmatrix}b_n^{(0)}\\ b_n^{(1)}\\ b_n^{(2)}\\ b_n^{(3)}\end{pmatrix}=\begin{pmatrix}a_n\\ a_n'\\ 3\Delta a_n-2a_n'-a_{n+1}'\\ -2\Delta a_n+a_n'+a_{n+1}'\end{pmatrix}, \tag{6}$$

where $\Delta a_n=a_{n+1}-a_n$ and $\{a_n'\}$ are found by solving the following tridiagonal system of linear equations:

$$\begin{pmatrix}2 & 4 & & & & \\ 1 & 4 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & 4 & 1\\ & & & & 4 & 2\end{pmatrix}\begin{pmatrix}a_0'\\ \\ \vdots\\ \\ a_{N-1}'\end{pmatrix}=\begin{pmatrix}-5a_0+4a_1+a_2\\ 3(a_2-a_0)\\ \vdots\\ 3(a_{N-1}-a_{N-3})\\ -a_{N-3}-4a_{N-2}+5a_{N-1}\end{pmatrix}. \tag{7}$$

3 DCT-BASED SCALING METHOD

The DFT-based image scaling[6,7] suffers from border effects caused by the periodic nature of the DFT. In order to eliminate these border effects, we suggest replacing DFT by DCT. Signal scaling by means of the Inverse Scaled DCT is given by the following formula:

$$\tilde{a}_k=\sqrt{\frac{2}{N}}\sum_{r=0}^{\lfloor\sigma N\rfloor-1}\alpha_r^{C(1/2),ZP}\cos\left\{\pi\frac{\left[k+1/2-\left(\lfloor\sigma N\rfloor-\sigma N\right)/2\right]r}{\sigma N}\right\}, \tag{8}$$

where $\tilde{a}_k$ is a σ-times scaled signal, $\lfloor\bullet\rfloor$ is a rounding-off operator:

$\sigma\ge1$	$\sigma<1$
$\lfloor x\rfloor=\mathrm{CEIL}(x)$	$\lfloor x\rfloor=\mathrm{FLOOR}(x)$

, (9)

$\alpha_r^{C(1/2),ZP}$ is the zero-padded or truncated DCT spectrum α_r^C:

$\sigma \geq 1$	$\sigma < 1$
$\alpha_r^{C(1/2),ZP} = \begin{cases} \alpha_r^C/2 & \text{if} \quad r = 0, \\ \alpha_r^C & \text{if} \quad r = 1,\ldots,N-2, \\ \alpha_r^C/2 & \text{if} \quad r = N-1, \\ 0 & \text{if} \quad r = N,\ldots,\lfloor \sigma N \rfloor - 1 \end{cases}$	$\alpha_r^{C(1/2),ZP} = \begin{cases} \alpha_r^C/2 & \text{if} \quad r = 0, \\ \alpha_r^C & \text{if} \quad r = 1,\ldots,\lfloor \sigma N \rfloor - 2, \\ \alpha_r^C/2 & \text{if} \quad r = \lfloor \sigma N \rfloor - 1 \end{cases}$

(10)

and α_r^C is the DCT spectrum of the input signal a_n:

$$\alpha_r^C = \mathrm{DCT}(a_n) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} a_n \cos\left[\pi \frac{(n+1/2)r}{N}\right]. \tag{11}$$

For efficient computation, Eq. (8) can be converted into four convolutions:

$$\begin{aligned} \tilde{a}_k = {} & \sqrt{\frac{2}{N}} \cos\left[\frac{\pi}{2\sigma N} k^2\right] \left\{ \begin{array}{l} \sum_{r=0}^{\lfloor \sigma N \rfloor - 1} \left\{ \alpha_r^{C(1/2),ZP} \cos\left[\frac{\pi}{2\sigma N}\left(r^2 + (1 - \lfloor \sigma N \rfloor + \sigma N) r\right)\right] \right\} \cos\left[\frac{\pi}{2\sigma N}(k-r)^2\right] \\ + \sum_{r=0}^{\lfloor \sigma N \rfloor - 1} \left\{ \alpha_r^{C(1/2),ZP} \sin\left[\frac{\pi}{2\sigma N}\left(r^2 + (1 - \lfloor \sigma N \rfloor + \sigma N) r\right)\right] \right\} \sin\left[\frac{\pi}{2\sigma N}(k-r)^2\right] \end{array} \right\} \\ & + \sqrt{\frac{2}{N}} \sin\left[\frac{\pi}{2\sigma N} k^2\right] \left\{ \begin{array}{l} \sum_{r=0}^{\lfloor \sigma N \rfloor - 1} \left\{ \alpha_r^{C(1/2),ZP} \cos\left[\frac{\pi}{2\sigma N}\left(r^2 + (1 - \lfloor \sigma N \rfloor + \sigma N) r\right)\right] \right\} \sin\left[\frac{\pi}{2\sigma N}(k-r)^2\right] \\ - \sum_{r=0}^{\lfloor \sigma N \rfloor - 1} \left\{ \alpha_r^{C(1/2),ZP} \sin\left[\frac{\pi}{2\sigma N}\left(r^2 + (1 - \lfloor \sigma N \rfloor + \sigma N) r\right)\right] \right\} \cos\left[\frac{\pi}{2\sigma N}(k-r)^2\right] \end{array} \right\}. \end{aligned} \tag{12}$$

In order to avoid boundary effects which are characteristic for cyclic convolution performed using FFT, one can perform convolution in DCT domain (rather than in DFT domain)[8]. The convolutions in Eq. (12) can be computed in DCT domain using the formula:

$$b_k = \sqrt{\frac{\lfloor \sigma N \rfloor}{2}} \left\{ \mathrm{IDCT}\left[\xi_p^C \eta_p^{CI}\right] + \mathrm{IDcST}\left[\xi_p^S \eta_p^{CI}\right] \right\}, \tag{13}$$

where ξ_p^C is the DCT (Discrete Cosine Transform) of α_r:

$$\xi_p^C = \mathrm{DCT}(\alpha_r) = \sqrt{\frac{2}{\lfloor \sigma N \rfloor}} \sum_{r=0}^{\lfloor \sigma N \rfloor - 1} \alpha_r \cos\left[\pi \frac{(r+1/2)p}{\lfloor \sigma N \rfloor}\right], \tag{14}$$

ξ_p^S is the DcST (Discrete Sine Transform) of α_r:

$$\xi_p^S = \mathrm{DcST}(\alpha_r) = \sqrt{\frac{2}{\lfloor \sigma N \rfloor}} \sum_{r=0}^{\lfloor \sigma N \rfloor - 1} \alpha_r \sin\left[\pi \frac{(r+1/2)p}{\lfloor \sigma N \rfloor}\right], \tag{15}$$

η_p^{CI} is the so called Discrete Cosine Transform – type I ($\mathrm{DCT}^{\mathrm{I}}$) of h_r (assuming that $h_{\lfloor \sigma N \rfloor} = 0$):

$$\eta_p^{CI} = \mathrm{DCT}^I(h_r) = \frac{1}{2\sqrt{\lfloor \sigma N \rfloor}} \left[h_0 + (-1)^p h_{\lfloor \sigma N \rfloor} + 2 \sum_{r=1}^{\lfloor \sigma N \rfloor - 1} h_r \cos\left(\pi \frac{r\,p}{\lfloor \sigma N \rfloor}\right) \right], \tag{16}$$

IDCT is the Inverse Discrete Cosine Transform:

$$a_k = \mathbf{IDCT}(\alpha_p) = \frac{1}{\sqrt{2\lfloor \sigma N \rfloor}} \left\{ \alpha_0 + 2 \sum_{p=1}^{\lfloor \sigma N \rfloor - 1} \alpha_p \cos\left[\pi \frac{(k+1/2)p}{\lfloor \sigma N \rfloor} \right] \right\} \tag{17}$$

and IDcST is the Inverse Discrete Sine Transform:

$$a_k = \mathbf{IDcST}(\alpha_p) = \frac{1}{\sqrt{2\lfloor \sigma N \rfloor}} \left\{ (-1)^k \alpha_{\lfloor \sigma N \rfloor} + 2 \sum_{p=1}^{\lfloor \sigma N \rfloor - 1} \alpha_p \sin\left[\pi \frac{(k+1/2)p}{\lfloor \sigma N \rfloor} \right] \right\}. \tag{18}$$

This convolution (Eq. (13)) can be converted into the "all-DCT" form containing only – DCT (Eq. (14)) and IDCT (Eq. (17)):

$$b_k = \sqrt{\frac{\lfloor \sigma N \rfloor}{2}} \left\{ \mathbf{IDCT}\left[\xi_p^C \eta_p^{CI} \right] + (-1)^k \, \mathbf{IDCT}\left[\left\{ \xi_p^S \eta_p^{CI} \right\}_{\lfloor \sigma N \rfloor - p} \right] \right\}, \tag{19}$$

where

$$\xi_p^C = \mathbf{DCT}(\alpha_r), \tag{20}$$

$$\xi_p^S = \left\{ \mathbf{DCT}\left[(-1)^r \alpha_r \right] \right\}_{\lfloor \sigma N \rfloor - p} \tag{21}$$

and

$$\eta_p^{CI} = -\frac{1}{\sqrt{2\lfloor \sigma N \rfloor}} h_0 + \cos\left(\frac{\pi}{2\lfloor \sigma N \rfloor} p \right) \mathbf{DCT}(h_r) + \sin\left(\frac{\pi}{2\lfloor \sigma N \rfloor} p \right) \left\{ \mathbf{DCT}\left[(-1)^r h_r \right] \right\}_{\lfloor \sigma N \rfloor - p}. \tag{22}$$

The suggested DCT-based scaling method produces perfectly sharp images and is virtually free of boundary effects, demonstrating perfect accuracy of resampling.

4 COMPUTATIONAL COMPLEXITY

We assume that both the input signal a_n and the output signal $\tilde{a}_k$ are 1D real vectors. In order to compute a scaled signal by the DCT-based scaling method, one needs to generate the **cos** / **sin** factors, perform multiplications and compute DCT transforms.

The "numerically stable" DCT algorithm needs $2\frac{1}{3} N \log_2 N - 2\frac{2}{9} N$ floating point operations (flops) and the "numerically stable" DCT type-I algorithm needs $2\frac{1}{3} N \log_2 N - 2\frac{8}{9} N$ flops[9].

Computational complexity of discussed image scaling methods is listed in Table 1.

Table 1. Comparison of computational algorithms of scaling methods.

Type of the scaling method	**Number of flops per one output sample**
Linear interpolation	**3**
Cubic interpolation	**17**
Cubic spline interpolation	$\mathbf{14+11}\frac{N}{\lfloor \sigma N \rfloor}$
Inverse Scaled Discrete Cosine Transform with DCT convolution	$\mathbf{32}\frac{\mathbf{2}}{\mathbf{3}}\mathbf{log}_2\lfloor \sigma N \rfloor + \mathbf{2}\frac{\mathbf{1}}{\mathbf{3}}\frac{N}{\lfloor \sigma N \rfloor}\mathbf{log}_2 N$
Inverse Scaled Discrete Cosine Transform with "all-DCT" convolution	$\mathbf{37}\frac{\mathbf{1}}{\mathbf{3}}\mathbf{log}_2\lfloor \sigma N \rfloor + \mathbf{2}\frac{\mathbf{1}}{\mathbf{3}}\frac{N}{\lfloor \sigma N \rfloor}\mathbf{log}_2 N$

From Table 1 it is clear that the computational complexity of the DCT-based scaling method is higher than that of the low-order splines. From the other hand, the DCT-based scaling method is equivalent to very high-order spline and provides very accurate scaling results that outperform those of low-order splines. In order to provide very high quality scaling using spline interpolation, one has to use a high-order spline, e.g., B-spline of order $\boldsymbol{p}$ with computational complexity $\mathbf{4}\boldsymbol{p}+\mathbf{2}$ flops per one output sample[10]. In this case the complexity increases with the order of spline and becomes comparable and even higher that the complexity of the DCT-based scaling. So the DCT-based scaling method is a natural choice for very accurate image scaling with reasonable computational complexity.

5 CONCLUSION

Novel boundary effect free scaling method of perfect image scaling is introduced. Computational complexity of different known and new image scaling algorithms is compared. It is shown that DCT-domain scaling algorithm presents a valuable alternative to other image scaling methods, certainly in terms of the accuracy but also in terms of the computational complexity. This makes it an attractive solution in real-time image and video processing applications.

REFERENCES

[1] Yaroslavsky, L., "Efficient algorithm for discrete sinc interpolation," *Appl. Optics* **36**, 460–463 (Jan. 10, 1997).

[2] Keys, R. G., "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoust., Speech, Signal Process.* **29**, 1153–1160 (Dec. 1981).

[3] Moler, C. B., [*Numerical Computing with MATLAB*], Society for Industrial and Applied Mathematics (2004).

[4] Wolberg, G., [*Digital Image Warping*], IEEE Computer Society Press (1990).

[5] de Boor, C., [*A Practical Guide to Splines*], Springer-Verlag (1978).

[6] Yaroslavsky, L., "Discrete transforms, fast algorithms, and point spread functions of numerical reconstruction of digitally recorded holograms," in [*Advances in Signal Transforms: Theory and Applications*], Astola, J. and Yaroslavsky, L., eds., *EURASIP Book Series on Signal Processing and Communications* **7**, ch. 3, 93–141, Hindawi (2007).

[7] Yaroslavsky, L., "Fast discrete sinc-interpolation: a gold standard for image resampling," in [*Advances in Signal Transforms: Theory and Applications*], Astola, J. and Yaroslavsky, L., eds., *EURASIP Book Series on Signal Processing and Communications* **7**, ch. 8, 337–405, Hindawi (2007).

[8] Yaroslavsky, L., "Boundary effect free and adaptive discrete signal sinc-interpolation algorithms for signal and image resampling," *Appl. Optics* **42**, 4166–4175 (July 10, 2003).

[9] Plonka, G. and Tasche, M., "Fast and numerically stable algorithms for discrete cosine transforms," *Linear Algebra Appl.* **394**, 309–345 (Jan. 1, 2005).

[10] Unser, M., Aldroubi, A., and Eden, M., "Fast B-spline transforms for continuous image representation and interpolation," *IEEE Trans. Pattern Anal. Mach. Intell.* **13**, 277–285 (Mar. 1991).