FAST GAUSSIAN FILTER WITH SECOND-ORDER SHIFT PROPERTY OF DCT-5

Kenjiro Sugimoto and Sei-ichiro Kamata

Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu City, Fukuoka, 808-0135 Japan

ABSTRACT

This paper presents an efficient constant-time Gaussian filter which provides a high accuracy at a low cost over a wide range of scale σ. It requires only 14 multiplications per pixel in image filtering regardless of σ, which is fewer than state-of-the-art constant-time Gaussian filters. Main ideas of the paper are as follows: 1) introducing a second-order shift property of the discrete cosine transform type-5 (DCT-5) to convolve cosines faster, and 2) suppressing error propagation caused by the shift property. Experiments in image processing show that the proposed algorithm is $3.7\times$ faster than a state-of-the-art recursive Gaussian filter and comparable to that of $\pm 3\sigma$-supported Gaussian convolution with $\sigma = 2.33$. The output accuracy is stable at around 80 [dB] all over $\sigma \in [1, 128]$.

***Index Terms*—** Gaussian filter, discrete cosine transform, sliding DCT, scale-space theory

1. INTRODUCTION

Gaussian filter is a fundamental tool in image processing and computer vision. Based on the success of scale-space theory [1, 2], it is widely used in a variety of tasks including object recognition [3], visual saliency [4] and edge detection [5], and high dynamic range imaging [6]. A principal problem in this scenarios is that the computational cost of Gaussian convolution proportional to scale parameter σ because the tasks require many Gaussian-filtered images over a wide range of σ. Generally, this problem has been managed by reduction of image resolution, oversimplification of Gaussian kernel, or hardware acceleration; however, they requires an additional device or causes a non-negligible loss of accuracy. A solid algorithmic solution is the use of a constant-time Gaussian filter, which has a σ-independent cost.

We summarize constant-time Gaussian filters as follows:

Iterated box filter [7, 8]**:** This has the simplest implementation in them. This approach is based on the central limit theorem, which guarantees that iteratively convolving a box kernel converges a Gaussian kernel with a certain σ at the limit. The computational cost is acceptable for all σ; however, the accuracy is comparatively insufficient due to a difficulty in discrete system that we have to control the convergence toward a target σ by combining integer-length box kernels only.

Recursive Gaussian filters [9–11]**:** This is the state-of-the-art style in constant-time Gaussian filters. It mimics a Gaussian filter as a low-order feedback system with two-pass processing. The computational cost and accuracy are basically sufficient and acceptable for some applications. However, as observed in [12], the accuracy drops drastically for $\sigma > 30$. Thus, this style is unstable in accuracy for a large σ.

Kernel decomposition [13–17]**:** This is also an effective approach to achieve constant-time filtering. It decomposes a Gaussian kernel into a sum of basis kernels which can be convolved at a σ-independent cost by using integral image [18, 19]. Basis kernels commonly-used are splines [13], polynomials [14], and cosines [15–17]. Particularly, cosine-based kernel decomposition has recently achieved the highest accuracy in them at a reasonable cost over a wide range of σ. We discuss an improvement of the cosine-based algorithms.

The cosine-based algorithms are developed based on the fact that a Gaussian kernel can be approximated by a weighted sum of cosines since it is an even function which contains low-frequency components only. Elboher and Werman presented Cosine Integral Image (CII) [15], which employed the Discrete Cosine Transform type-1 (DCT-1) for kernel decomposition and utilizes integral images for cosine convolution. Sugimoto and Kamata [16] showed how to perform cosine convolution without integral images, contributing to memory access saving. After that, they eliminated the offset distortion and reduced the computational cost by employing DCT-5 [17]. A problematic point of these algorithms is that it still requires a higher computational cost than that of state-of-the-art recursive Gaussian filters. Although these researches demonstrated that their practical filtering time outperform that of recursive Gaussian filters in their experiments, it might be environment-dependent. We therefore reveal that cosine-based algorithms achieve a lower computational cost than the recursive ones by utilizing a significant property of DCT-5.

The key technique to reduce the computational cost is sliding DCT [20–24], which is a recursive way to compute short-time DCT coefficients and has been studied in signal processing since 1980s. Because cosine convolution performed in the existing algorithms is equivalent to computation of short-time DCT coefficients, it can be replaced to sliding DCT completely. A problem for applying it to Gaussian filter is that an efficient sliding DCT-5 has not been explicitly derived yet.

The previous researches have traditionally focused only on DCT-$1, 2, 3, 4$ (with an even-length of period) but have less focused on DCT-$5, 6, 7, 8$ (with an odd-length of period) [25] except Wu [21], probably due to their compatibility to butterfly computation. Sliding DCT handles relations between two or three adjacent short-time coefficients, called a first-order or second-order shift properties of DCT, respectively. Generally, second-order one outperforms first-order one in computational cost. Wu's work focused only on a first-order one only, not a second-order one. Hence, we derive a second-order shift property of DCT-5 and then develop a lower-cost filtering algorithm based on it.

This paper presents an efficient constant-time Gaussian filter which provides a high accuracy at a low cost over a wide range of σ. It requires only 14 multiplications per pixel in image filtering regardless of σ, which is fewer than state-of-the-art constant-time Gaussian filters. Our key idea is to derive a second-order shift property of DCT-5 to compute short-time DCT coefficients in order to achieve a lower computational cost than existing algorithms. Suppression of error propagation caused by the shift property is also discussed in the paper to achieve stably higher accuracy. Experiments in image filtering demonstrate two significant improvements: the filtering speed of our algorithm is $3.7\times$ faster than state-of-the-art recursive Gaussian filters and comparable to that of Gaussian convolution with $\sigma = 2.33$. Moreover, it can stably provide an accuracy of around 80 [dB] all over $\sigma \in [1, 128]$.

2. FAST GAUSSIAN FILTERING

This paper discusses one-dimensional Gaussian filter only because of its *separability*, i.e., a multi-dimensional Gaussian kernel can be decomposed into a product of multiple one-dimensional Gaussian kernels. Our algorithm is therefore applicable to arbitrary-dimensional data. Let σ be a scale parameter and R be a truncation location ($R = \lceil 3\sigma \rceil$ in general). A truncated Gaussian kernel g_u ($u = -R, \ldots, +R$) and its decomposition via DCT-5 can be defined as

$$g_u = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{u^2}{2\sigma^2}} = \sum_{k=0}^{R} G_k \cos(\phi k u), \tag{1}$$

where $\phi = \frac{2\pi}{2R+1}$ is introduced for simplicity and G_k indicates the k-th DCT coefficient of g_u. An approximate G_k can be obtained from a closed-form expression,

$$G_k \simeq \frac{c_k}{2R+1} e^{-\frac{1}{2}\sigma^2\phi^2k^2}, \quad c_k = \begin{cases} 1 & \text{if } k = 0, \\ 2 & \text{otherwise.} \end{cases} \tag{2}$$

Since G_k exponentially decreases, g_u can be approximated by few low-frequency components. We therefore truncate frequency components higher than K (i.e., $G_k \leftarrow 0$ if $K < k$). Convolution with an input sequence f_x ($x = 0, 1, \ldots, N-1$) and an approximate kernel $\tilde{g}_u$ can be described as

$$(f * g)_x \simeq (f * \tilde{g})_x = \sum_{u=-R}^{R} f_{x+u}\tilde{g}_u = \sum_{k=0}^{K} G_k F_k^{(x)}, \tag{3}$$

where $F_k^{(x)}$ is the k-th short-time DCT coefficient of the input sequence f_x at location x, given in

$$F_k^{(x)} = \sum_{u=-R}^{R} f_{x+u} \cos(\phi k u). \tag{4}$$

Since G_k is precomputable prior to filtering process, it is important to obtain $F_k^{(x)}$ at an R-independent and lower cost.

2.1. Second-order shift property of DCT-5

Our algorithm computes $F_k^{(x)}$ recursively by managing a second-order shift property of DCT-5, which is a relationship among three consecutive short-time coefficients. Consider

$$F_k^{(x-1)} = \sum_{u=-R}^{R} f_{x+u}C_{u-1} + f_{x+R+1}C_R - f_{x-R}C_{-R-1}$$

$$F_k^{(x+1)} = \sum_{u=-R}^{R} f_{x+u}C_{u+1} - f_{x+R}C_{R+1} + f_{x-R-1}C_{-R},$$

where $C_u = \cos(\phi k u)$ for simplicity. Since $C_R = C_{-R} = C_{R+1} = C_{-R-1}$, $F_k^{(x+1)} + F_k^{(x-1)}$ can be rewritten as

$$F_k^{(x+1)} = 2C_1F_k^{(x)} - F_k^{(x-1)} + C_R\delta^{(x)}, \tag{5}$$

where $\delta^{(x)} = f_{x+R+1} - f_{x+R} - f_{x-R} + f_{x-R-1}$. The coefficients are precomputable as look-up tables. The first and second terms $F_k^{(0)}$ and $F_k^{(1)}$ are explicitly calculated from (4). The case of $k = 0$ should be treated for cost saving as

$$F_0^{(x+1)} = F_0^{(x)} + f_{x+R+1} - f_{x-R}. \tag{6}$$

The second-order shift property (5) enables us to recursively compute $F_k^{(x)}$ at an R-independent cost.

2.2. Suppression of error propagation

Consider reducing more multiplications by utilizing look-up tables. The recurrence relations (5) and (6) can absorb G_k contained in (3) into themselves. A concern is that naively moving G_k into them may result in propagating rounding-off error due to their recursive ways. We understand f_x to originally contain no rounding-off error. This is natural in most real applications because many images consist of integer pixels. A rounding-off error would occur in multiplying f_x and a real number such as G_k or $\cos(\phi k)$. The dominant component of error propagation is the one having the largest weight in all the components, i.e., $k = 0$. Hence, we move G_k into

Algorithm 1 Proposed constant-time Gaussian filter

1: ▷ Calculating first and second terms
2: $F_0 \leftarrow \sum_{u=-R}^{+R} f_u$
3: **for** $k \leftarrow 1$ to K **do**
4: $\quad Z_k^- \leftarrow \sum_{u=-R}^{+R} \{\gamma_k \cos(\phi k u)\} f_u$
5: $\quad Z_k \leftarrow \sum_{u=-R}^{+R} \{\gamma_k \cos(\phi k u)\} f_{u+1}$
6: **end for**
7:
8: ▷ Convoluting cosine terms slidingly
9: $(f * \tilde{g})_0 \leftarrow \{G_0\}(F_0 + \sum_{k=1}^{K} Z_k^-)$
10: $F_0 \leftarrow F_0 - f_{-R} + f_{R+1}$
11: **for** $x \leftarrow 1$ to $N-1$ **do**
12: $\quad$ ▷ Calculating the output value for location x
13: $\quad (f * \tilde{g})_x \leftarrow \{G_0\}(F_0 + \sum_{k=1}^{K} Z_k)$
14: $\quad$ ▷ Updating short-time DCT coefficients for the next
15: $\quad F_0 \leftarrow F_0 - f_{x-R} + f_{x+R+1}$
16: $\quad \delta \leftarrow f_{x-R-1} - f_{x-R} - f_{x+R} + f_{x+R+1}$
17: $\quad$ **for** $k \leftarrow 1$ to K **do**
18: $\quad\quad \zeta \leftarrow \{2\cos(\phi k)\} Z_k - Z_k^- + \{\gamma^{(k)} \cos(\phi k R)\}\delta$
19: $\quad\quad Z_k^- \leftarrow Z_k$, $Z_k \leftarrow \zeta$
20: $\quad$ **end for**
21: **end for**

Fig. 1. A main procedure of our algorithm.

(5) only for $k \neq 0$ in order to avoid real multiplications in (6). Specifically, (3) can be deformed as

$$(f * \tilde{g})_x = G_0 \left(F_0^{(x)} + \sum_{k=1}^{K} Z_k^{(x)} \right), \quad Z_k^{(x)} = \gamma_k F_k^{(x)},$$

where $\gamma_k = G_k / G_0 = c_k e^{-\frac{1}{2}\sigma^2\phi^2 k^2}$. A recurrence relation for $Z_k^{(x)}$ is deformed from (5) as

$$Z_k^{(x+1)} = 2C_1 Z_k^{(x)} - Z_k^{(x-1)} + C_R \gamma_k \delta^{(x)}.$$

This way can suppress error propagation because the recurrence relation for the dominant component given in (6) still contains no real multiplications. Incidentally, if error propagation is still non-negligible, the most secure solution is to directly refresh $Z_k^{(x)}$ at a regular interval via (4).

2.3. Algorithm flow and its analysis

Figure 1 shows a pseudo code of our algorithm where $\{\cdot\}$ denotes precomputed values stored in look-up tables. We count the number of arithmetic operations of our algorithm, targeting only the core filtering routine in line 11–21 because the precomputing routines has a negligible cost as compared with the the core one. Our algorithm, for each output, requires $2K + 1$ multiplications (1 in line 13, and $2K$ in line 18) and $3K + 5$ additions/subtractions (K in line 13, 2 in line 15, 3 in line 16, and $2K$ in line 18). We introduce two effective tips for its efficient implementation: loop unrolling for small-sized loops and ring buffers for updating short-time DCT coefficients. Unrolling loops with respect to k such as line 17–20 eliminates loop counting routine and condition check. Ring buffers with a length of two facilitate to remove the cyclic substitutions in line 19. Both of them are simple to implement due to a small K, surely saving the computational cost without loss of accuracy.

Table 1. Number of operations per element (1D filtering).

Method	Mul/Div*	Add/Sub*
Convolution	$R+1$	$2R+1$
Deriche [9]	$4M$	$4M-2$
Farneback and Westin [11]	$4M$	$4M-2$
van Vliet *et al*. [10]	$2M$	$2M+2$
Cosine integral image [15]	$4K$	$6K+2$
Sugimoto and Kamata [17]	$4K+2$	$4K+1$
Ours	$2K+1$	$3K+5$

*Common parameters: $R = \lceil 3\sigma \rceil$, $M = 4$, and $K = 3$.

Table 1 lists the number of arithmetic operations of various Gaussian filters. Under the common parameter setting noted at the bottom of the table, our algorithm achieves the fewest multiplications in the constant-time Gaussian filters followed by the recursive Gaussian filter proposed by van Vliet *et al*. [10]. In the case of two-dimensional filtering, they requires 14 multiplications per pixel and 16 multiplications per pixel, respectively. In comparison with other cosine-based Gaussian filters, the cost is reduce by nearly half. Another advantage of our algorithm over recursive Gaussian filters is one-pass filtering per dimension, saving memory access and simplifying the implementation. Existing recursive Gaussian filters consist of a two-pass feedback system and CII requires to construct integral images in advance, causing extra memory access. Memory access cost, as compared with arithmetic operations, has been a more principal bottleneck for modern computers recent years. Hence, our algorithm has a low computational cost in terms of both arithmetic operation and memory access costs.

3. EXPERIMENTS AND DISCUSSION

This section examines the practical computational time, accuracy, and characteristics of our algorithm as compared with existing algorithms through experiments in image filtering. The competitors are $\pm 3\sigma$-supported Gaussian convolution, recursive Gaussian filters by van Vliet *et al*. [10] and by Farneback and Westin [11], and cosine-based algorithms including CII [15], Sugimoto and Kamata [17], and our algorithm under the common parameters as noted in Table 1. The test environment mounts on Intel Core i5 2.67GHz CPU and 8GB main memory. Test images are "lenna" with 512×512 pixels, "baboon" with 512×512 pixels, and "N2"

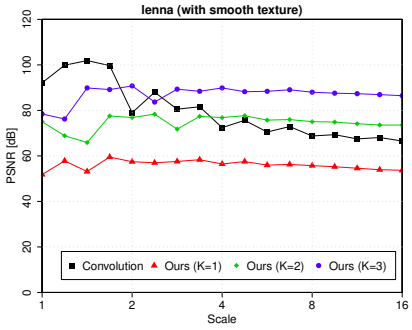

Fig. 2. Scale vs PSNR (lenna).

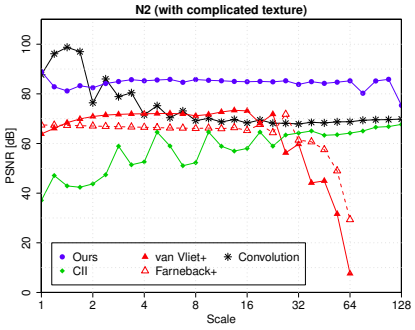

Fig. 3. Scale vs PSNR (N2).

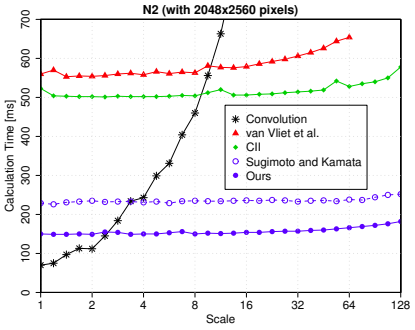

Fig. 4. Scale vs computational time (N2).

Convolution (79.0 dB, 78.3 dB) van Vliet *et al.* [10] (75.6 dB, 72.2 dB) **Ours (88.6 dB, 83.3 dB)**

Fig. 5. $500\times$ amplified error images of various Gaussian filters with $\sigma = 2$.

with 2048×2560 pixels from the Standard Image Data-Base (SIDBA) and the Standard Colour Image Data (ISO/JIS-SCID) [26], converted from 24-bits RGB color to 32bit-float grayscale with a dynamic range of $[0, 1]$. The implementations used are OpenCV 2.4.5 [27] (cv::GaussianBlur function) for Gaussian convolution and self-produced ones for the others where all the implementations were written in C++.

We find K to provide a sufficient accuracy. Figure 2 shows the Peak-Signal-To-Noise Ratio (PSNR) of Gaussian convolution and our algorithm ($K = 1, 2, 3$) where the ideal output is assumed to be the output of $\pm5\sigma$-supported Gaussian convolution. Evidently, $K = 3$ produces an accuracy comparable or superior to that of Gaussian convolution.

Figure 3 plots the PSNR of the competitors over a wider range of σ. The accuracy of our algorithm is the highest in them and stable at around 80 [dB] all over $\sigma \in [1, 128]$. By contrast, the accuracy of the two recursive filters drastically declined around $30 < \sigma$, as also reported in [12]. Thus, our algorithm can provide a more reliable quality than the others over a wide range of scale.

Figure 4 plots the computational time of the competitors. Our algorithm is the fastest in them, achieving about $3.7\times$ faster than the recursive Gaussian filter and CII, $1.6\times$ faster than Sugimoto and Kamata. The computational time is comparable to that of $\pm3\sigma$-supported Gaussian convolution with $\sigma = 2.33$ (or $R = 7$). The results of convolution and our algorithm mostly coincide with expectations from the number of arithmetic operations shown in Table 1; however, those of the recursive Gaussian filters do not. This is probably attributed to the practical cost caused by its two-pass filtering per dimension, unlike one-pass filtering per dimension such as convolution and our algorithm. Thus, our algorithm stably provides the state-of-the-art filtering speed regardless of σ.

Last, we perform visual assessment of error characteristics of our algorithm. Figure 5 lists $500\times$ amplified error images for $\sigma = 2$. The outputs of our algorithm obviously contain less error than the others. Convolution and our algorithm show substantially-difference characteristics: noise in convolution occurs around edges as ripple phenomenon; noise in our method consists of higher frequency as clearly seen in the result of "baboon". This is due to the high frequency truncation with K. Actually, this difference is trivial for many applications because of the $500\times$ amplification to facilitate visuality. Thus, our algorithm can be an alternative of $\pm3\sigma$-supported Gaussian convolution.

4. CONCLUSIONS

This paper presented an efficient constant-time Gaussian filter with a second-order shift property of DCT-5 which outperforms existing algorithms over a wide range of scale in terms of accuracy, its stability, and computational cost. Experiments in image filtering demonstrated the superiority of utilizing a second-order shift property of DCT-5. Our algorithm is an algorithmic solution for multiscale image analysis to naively overcome the cost increase problem of Gaussian convolution. We believe that our research outcome contributes to a variety of applications in image processing and computer vision.

5. REFERENCES

[1] T. Lindeberg, "Scale-space for discrete signals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 3, pp. 234–254, Mar. 1990.

[2] J. Weickert, S. Ishikawa, and A. Imiya, "Linear scale-space has first been proposed in Japan," *J. Math. Imaging and Vision*, vol. 10, no. 3, pp. 237–252, 1999.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[4] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, 1998.

[5] M. Basu, "Gaussian-based edge-detection methods — a survey," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 32, no. 3, pp. 252–260, Aug. 2002.

[6] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda, "Photographic tone reproduction for digital images," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 267–276, July 2002.

[7] W. M. Wells, "Efficient synthesis of Gaussian filters by cascaded uniform filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 2, pp. 234–239, Mar. 1986.

[8] J. A. Robinson, "Efficient gaussian filtering using cascaded prefix sums," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2012, vol. 2, pp. 117–120.

[9] R. Deriche, "Fast algorithms for low-level vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 78–87, 1990.

[10] L.J. van Vliet, I.T. Young, and P.W. Verbeek, "Recursive Gaussian derivative filters," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, 1998, vol. 1, pp. 509–514.

[11] G. Farnebäck and C. Westin, "Improving Deriche-style recursive Gaussian filters," *J. Math. Imaging and Vision*, vol. 26, no. 3, pp. 293–299, Nov. 2006.

[12] A. Bernardino and J. Santos-Victor, "Fast IIR isotropic 2-D complex Gabor filters with boundary initialization," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3338–3348, Nov. 2006.

[13] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing. II. efficiency design and applications," *IEEE Trans. Signal Process.*, vol. 41, no. 2, pp. 834–848, 1993.

[14] M. Hussein, F. Porikli, and L. Davis, "Kernel integral images: A framework for fast non-uniform filtering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, June 2008, pp. 1–8.

[15] E. Elboher and M. Werman, "Cosine integral images for fast spatial and range filtering," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sept. 2011, pp. 89–92.

[16] K. Sugimoto and S. Kamata, "Fast image filtering by DCT-based kernel decomposition and sequential sum update," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2012, pp. 125–128.

[17] K. Sugimoto and S. Kamata, "Fast Gaussian filter using DCT-V," in *Int. Workshop on Advanced Image Technology (IWAIT)*, Jan. 2013, pp. 338–343.

[18] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. ACM SIGGRAPH*. 1984, vol. 18, pp. 207–212, ACM Press.

[19] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2001, vol. 1, pp. 511–518.

[20] P. Yip and K. Rao, "On the shift property of DCT's and DST's," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 35, no. 3, pp. 404–406, Mar. 1987.

[21] L.-N. Wu, "Comments on "On the shift property of DCT's and DST's"," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 1, pp. 186–188, Jan. 1990.

[22] N.R. Murthy and M.N.S. Swamy, "On the computation of running discrete cosine and sine transform," *IEEE Trans. Signal Process.*, vol. 40, no. 6, pp. 1430–1437, June 1992.

[23] J. Xi and J.F. Chicharo, "Computing running DCT's and DST's based on their second-order shift properties," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 47, no. 5, pp. 779–783, May 2000.

[24] V. Kober, "Fast algorithms for the computation of sliding discrete sinusoidal transforms," *IEEE Trans. Signal Process.*, vol. 52, no. 6, pp. 1704–1710, June 2004.

[25] G. Strang, "The discrete cosine transform," *SIAM Review*, vol. 41, no. 1, pp. 135–147, Jan. 1999.

[26] ISO, JSA, and M. Kaji, "ISO/JIS-SCID: graphic technology —prepress digital data exchange— standard colour image data," 1995.

[27] "Open Source Computer Vision Library (OpenCV)," `http://opencv.org/`.