Acceleration of Gaussian Filter with Short Window Length Using DCT-1

Takahiro Yano*, Kenjiro Sugimoto*, Yoshimitsu Kuroki†, and Sei-ichiro Kamata*
*Waseda University, Graduate School of Information, Production and Systems, JAPAN
†National Institute of Technology, Kurume College, JAPAN

***Abstract*—This paper presents an accelerated constant-time Gaussian filter ($O(1)$ GF) specialized in short window length where constant-time ($O(1)$) means that computational complexity per pixel does not depend on filter window length. Our method is extensively designed based on the idea of $O(1)$ GF based on Discrete Cosine Transform (DCT). This framework approximates a Gaussian kernel by a linear sum of cosine terms and then convolves each cosine term in $O(1)$ per pixel using sliding transform. Importantly, if window length is short, DCT-1 consists of easily-computable cosine values such as 0, $\pm\frac{1}{2}$ and ±1. This behavior is not satisfied in other DCT types. From this fact, our method accelerates the sliding transform by employing DCT-1 focusing on short window length. Experiments show that our method overcomes naive Gaussian convolution and existing $O(1)$ GF in terms of computational time. Interestingly, the results also reveal that, without truncating negligible terms, our method runs faster than convolution.**

I. Introduction

Gaussian filter (GF) is one of the fundamental tools in signal processing and computer vision. It has been widely used in a variety of tasks including object recognition [1], stereo matching [2], visual saliency [3], edge-preserving smoothing [4], [5], [6], [7], [8]. Since the GF is implemented in general as convolution, its computational complexity is proportional to the length of filter window. This behavior is a severe problem for applications that require large filter window length, e.g., scalespace analysis [9] and high-resolutional image processing.

In order to address the problem, constant-time GF ($O(1)$ GF) has been discussed in the past where $O(1)$ means that computational complexity per pixel is independent of window length. Existing $O(1)$ GF algorithms can be categorized into two representative groups: recursive filter approximation [10], [11], [12], [13] and cosine approximation [14], [15], [16], [17]. We focus on the latter one because so far it has achieved the highest performance trade-off between computational complexity and approximation accuracy. This framework first approximates Gaussian kernel by a linear sum of few cosine terms and then convolves each cosine term with an input sequence in an $O(1)$ manner. The kernel approximation is generally derived via Discrete Cosine Transform (DCT). It is well known that DCT has eight types of definition, called DCT-1, 2, ..., 8. Existing methods employed different DCT types based on different concepts. For instance, Elboher and Werman [14] selected DCT-1 and proposed an efficient filtering technique using integral images [18], [19] for convolution. Sugimoto and Kamata [8], [20] employed DCT-5 and accelerated convolution by deriving a sliding transform based on second-order shift property [21]. Charalampids [22] used DCT-3 and presented how to derive more accurate transform coefficients that maintain both unity and variance of Gaussian kernel. We mention that, although a variety of DCT types have been employed depending on the researches, they show almost the same approximate accuracy and computational complexity.

As mentioned, a major motivation for exploring $O(1)$ GF was originally for accelerating the case of long window length. However, as [16] showed, the computational time is significantly faster than convolution even for short window length. We further explore this result by focusing on DCT-1 that tends to consist of many easily-computable cosine values such as 0, $\pm\frac{1}{2}$, and ±1 when window length is short. This implies that it is possible to reduce more multiplications by replacing them to addition/subtraction/shift operations. This advantage enables us to design a faster $O(1)$ GF algorithm based on DCT.

This paper presents an accelerated $O(1)$ GF algorithm specialized in short window length using DCT-1. After summarizing Gaussian convolution, Section 2 describes $O(1)$ GF based on DCT-1 and specifies our key idea for acceleration specialized in short window length. Section 3 compares naive convolution, existing $O(1)$ GF, and our method in terms of computational time and approximate accuracy through some experiments using a natural image. These results reveal that, our method can run faster than naive convolution without truncating negligible cosine terms even if window length is short. This fact implies that our method can run faster if we deal with arbitrary even-symmetric kernels whose short window length.

II. Proposed Method

A. Gaussian convolution

Let $x_t \in \mathbb{R}$ $(t = 0, 1, \ldots)$ be a one-dimensional input sequence and $h_n \in \mathbb{R}$ $(n = -N+1, \ldots, N-1)$ be a Gaussian kernel. The Gaussian kernel is defined by

$$h_n = \eta^{-1} e^{-\frac{n^2}{2\sigma^2}}, \qquad \eta = \sum_{n=-N+1}^{N-1} e^{-\frac{n^2}{2\sigma^2}}, \tag{1}$$

where σ is the standard deviation of the kernel. This definition followed the sampled Gaussian kernel in [23]. Convolution

between x_t and h_n is described as

$$(x * h)_t = \sum_{n=-N+1}^{N-1} x_{t+n} h_n. \quad (2)$$

In general, N has to be appropriately determined from the parameter σ, e.g., $N = \lceil 3\sigma \rceil + 1$. Obviously, this operation requires $(2N-1)$ multiplications per pixel, i.e., $O(N)$ per pixel. This means that the wider window length causes the longer computational time. This behavior is a severe problem for applications that require large filter window length, e.g., scale space analysis and high-image filtering.

B. $O(1)$ Gaussian filter based on DCT-1

An efficient solution to the above problem is $O(1)$ GF where $O(1)$ means its computational complexity per pixel is independent of window length. Because of the even symmetry of (2), we decompose Gaussian kernel into

$$h_n = \sum_{k=0}^{N-1} \hat{h}^{(k)} C_n^{(k)}, \qquad C_n^{(k)} = \cos\left(\frac{\pi}{N-1} kn\right), \quad (3)$$

where $\hat{h}^{(k)}$ is a transform coefficient. Note that $C_n^{(k)}$ is related to DCT-1 and introduced for simplicity. By substituting (3) for (2), we obtain

$$(x * h)_t = \sum_{n=-N+1}^{N-1} \sum_{k=0}^{N-1} x_{t+n} \hat{h}^{(k)} C_n^{(k)} = \sum_{k=0}^{N-1} \hat{h}^{(k)} \hat{x}_t^{(k)}, \quad (4)$$

where

$$\hat{x}_t^{(k)} = \sum_{n=-N+1}^{N-1} x_{t+n} C_n^{(k)}. \quad (5)$$

We call $\hat{x}_t^{(k)}$ a short-time transform coefficient of the input sequence x_t at time t. Note that short-time transform coefficients are slightly different from short-time DCT-1 coefficients, due to a difference from the range of the summation. Since the spectrum of Gaussian kernel has also a Gaussian shape, the coefficients $\hat{h}^{(k)}$ decreases exponentially. Hence, we can well approximate (3) by truncating the terms at $K(<< N)$ where K is the number of cosine terms used for approximation.

The short-time transform coefficients are computable in $O(1)$ time per pixel as follows: Between three adjacent short-time transform coefficients $\hat{x}_{t-1}^{(k)}$, $\hat{x}^{(k)}$ and $\hat{x}_{t+1}^{(k)}$, the following relationship holds:

$$\hat{x}_{t-1}^{(k)} + \hat{x}_{t+1}^{(k)} = C_1^{(k)} \{2\hat{x}_t^{(k)} - (-1)^k \chi_{t,N-1}\} + (-1)^k \chi_{t,N}, \quad (6)$$

where $\chi_{t,N} = x_{t-N} + x_{t+N}$. By using (6), we can recursively compute $\hat{x}_{t+1}^{(k)}$ from the already-computed $\hat{x}^{(k)}$ and $\hat{x}_{t-1}^{(k)}$ in $O(1)$ time per pixel. This operation is generally called sliding transform. The relationship between the three adjacent coefficients is called second-order shift-property [21] in particular. Evidently, (6) requires one multiplication, one shift operation, and several additions/subtractions. As discussed later, we can reduce more operations using look-up table technique.

TABLE I
PARAMETERS OF ALL DCT TYPES

DCT type	T		k_0		n_0	
DCT-1 (inverse DCT-1)	$2N-2$	△	0	◊	0	◯
DCT-2 (inverse DCT-3)	$2N$	△	0	◊	1/2	
DCT-3 (inverse DCT-2)	$2N$	△	1/2		0	◯
DCT-4 (inverse DCT-4)	$2N$	△	1/2		1/2	
DCT-5 (inverse DCT-5)	$2N-1$		0	◊	0	◯
DCT-6 (inverse DCT-7)	$2N-1$		0	◊	1/2	
DCT-7 (inverse DCT-6)	$2N-1$		1/2		0	◯
DCT-8 (inverse DCT-8)	$2N+2$	△	1/2		1/2	

TABLE II
THREE SPECIAL CASES OF (6)

$C_1^{(k)}$	right hand side of (6)
0	$(-1)^k \chi_{t,N}$
$\pm\frac{1}{2}$	$\pm\hat{x}_t^{(k)} \mp \frac{1}{2}(-1)^k \chi_{t,N-1} + (-1)^k \chi_{t,N}$
± 1	$\pm 2\hat{x}_t^{(k)} \mp (-1)^k \chi_{t,N-1} + (-1)^k \chi_{t,N}$

C. Advantages of DCT-1 over the other DCT types

The use of DCT-1 provides the following advantages over the other DCT types. In order to replace multiplications to additions/subtractions/shift operations (ADD/SUB/SHIFT), cosine values and their phases to be computed require to be $\cos 0^\circ = 1$, $\cos 60^\circ = \frac{1}{2}$, $\cos 90^\circ = 0$ and so on. Let us analyze phases of each DCT using the general form

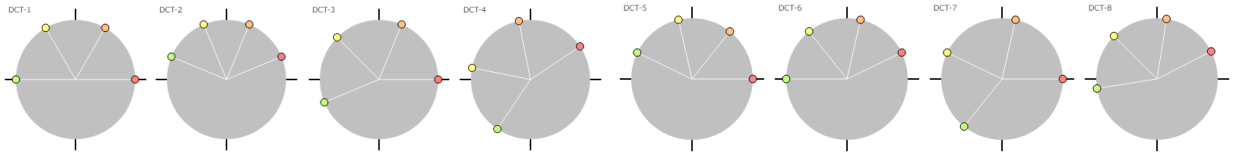
$$h_n = \sum_{k=0}^{N-1} \hat{h}^{(k)} \cos\left(\frac{2\pi}{T}(k+k_0)(n+n_0)\right) \quad (7)$$

where T, k_0, n_0 are parameters that characterize DCT type, as Table I listed. First of all, because Gaussian kernel is even-symmetric at $n = 0$, i.e., $h_n = h_{-n}$, the DCT types satisfying this condition can be applied to approximate Gaussian kernel (◯). Second, if the initial phase starts from 0°, i.e., $k_0 = 0$, the phases tend to locate on horizontal and/or vertical axis (◊). Third, each DCT divides 360° by T. If T is even, the cosine values could be symmetric at horizontal and/or vertical axes (△). In order to graphically understand these differences, we show some phases used in DCT-1, 2, ... 8 in Fig. 1 where $N = 4$ and $k = 1$. Obviously, since DCT-1 satisfies all the conditions simultaneously, it is the most appropriate for eliminating multiplications.

Let us observe some specific cases of (6) that are simplified by the fact mentioned above. Table II shows the right hand side of (6) in the case of $C_1^{(k)} = 0$, $\pm\frac{1}{2}$, and ± 1. Evidently, we reduce many multiplications in these cases. Table III shows the number of operations required in the case of $N = 2, 3, \ldots, 7$ where ACCESS means the number of pixel loading operations of input sequences. It is clear that all the operations in these cases can be implemented as ADD/SUB/SHIFT. Thus, DCT-1 provides fast computation if N is small.

D. Policy of implementation

We here describe how to determine the parameters K, N, and $\hat{h}^{(k)}$. Given σ, Gaussian kernel with any σ can be accurately-approximated using $K = 2, 3, 4$. This parameter is determined in terms of acceptable computational time. Then, as [16] did, the optimal value of N can be found via error



(a) DCT-1 (b) DCT-2 (c) DCT-3 (d) DCT-4 (e) DCT-5 (f) DCT-6 (g) DCT-7 (h) DCT-8

Fig. 1. DCT-1 tends to produce easily-computable cosine values such as $\cos 0^\circ = 1$, $\cos \pm 60^\circ = \pm\frac{1}{2}$, $\cos \pm 90^\circ = 0$ and $\cos \pm 180^\circ = -1$ while DCT-2, 3, ... 8 do not. The red, orange, yellow and green points correspond to $n = 0, 1, 2$, and 3, respectively.

TABLE III
THE NUMBER OF OPERATIONS FOR SMALL N

N	k	SHIFT	ADD/SUB	MUL	ACCESS
2	0,1	1	5	0	4
3	0,2	1	5	0	4
	1	0	2	0	2
4	0,1,2,3	1	5	1	4
5	0,4	1	5	0	4
	1,3	1	5	1	4
	2	0	2	0	2
6	0,5	1	5	0	4
	1,2,3,4	1	5	1	4
7	0,2,4,6	1	5	0	4
	1,5	1	5	1	4
	3	0	2	0	2

minimization. Using the computed values of K and N, we compute transform coefficients $\hat{h}^{(k)}$ in a least-squares manner. For achieving more accurate approximation, we adjust $\hat{h}^{(k)}$ to maintain the surface area of the kernel as one, i.e., the unity of Gaussian kernel. Note that these parameters are computable in advance and the computational time is negligible as compared with main filtering process. For faster filtering, $\hat{h}^{(k)}$ and its related constant numbers, e.g., $C_1^{(k)}$ are precomputed and then stored in look-up tables.

III. EXPERIMENTS AND DISCUSSION

This section confirms practical performance of our method through several experiments using a natural image. Throughout all the experiments, the test image is "N3" provided in ISO/JIS SCID [24], where it has 2560 $\times$ 2048 pixels and we transformed the original RGB color to 8-bits grayscale. Test environment consists of Intel Core i5 2.67GHz CPU and 8GB main memory. We implemented all the methods in C++ and do not explicitly use parallel computation architecture such as multi-core/vector computing.

A. Evaluation in truncating-frequency case

We evaluated performance of our method in truncating-frequency case where $K = 2$. Fig. 2 plots computational time of the naive Gaussian convolution, the existing $O(1)$ GF based on DCT-1 (existing method), and our method under a variety of σ in column, row, or both directions. Our method outperformed the existing method in row-direction filtering. Importantly, in image filtering, the existing method and our method can run significantly faster than naive Gaussian convolution even if window length is small. This is because both methods requires the fewer number of pixels to be loaded than Gaussian convolution.

Fig. 3 plots computational time and Peak Signal-to-Noise Ratio (PSNR) of $O(1)$ GF with DCT-1, 3, 5, 7 (DCT-1, 3, 5, 7 method) [17] where the DCT-1 means our method. PSNR indicates value between ideal and real output images where we assume as the output of $\pm 6\sigma$-supported Gaussian convolution are ideal output images. The saw-tooth-like steep changes of PSNR in Fig. 3 are caused by change in the window length. Our method is slightly slower than the DCT-3 method in term of computational time. This is because a sliding transform of the DCT-3 method references only two input pixels by some features of DCT-3, i.e., $C_{+N-1}^{(k)} = C_{-N+1}^{(k)}$, $C_{+N}^{(k)} = C_{-N}^{(k)} = 0$. The sliding transform of the DCT-3 method is given as

$$\hat{x}_{t-1}^{(k)} + \hat{x}_{t+1}^{(k)} = 2C_1^{(k)}\hat{x}_t^{(k)} + C_{N-1}^{(k)}\chi_{t,N}. \tag{8}$$

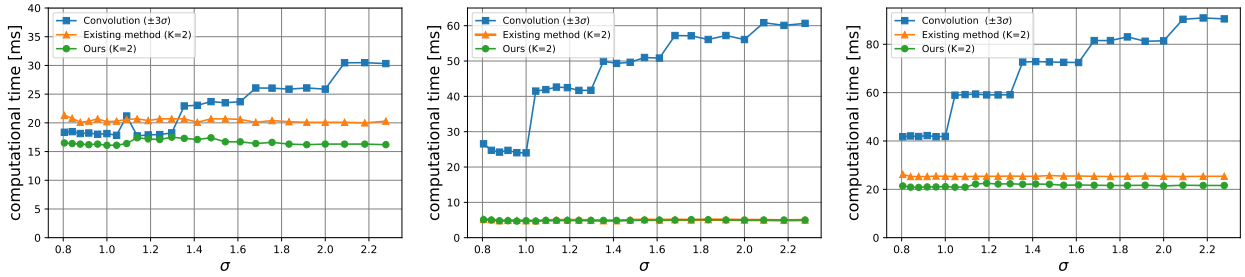
In PSNR, our method outperforms the DCT-3 method. These experimental results demonstrated that our method outperform the $O(1)$ GF of other DCT-type in term of computational time and PSNR.

B. Evaluation in full-frequency case

We evaluated computational time and PSNR in full-frequency cases, i.e., $K = N$, as Fig. 3 shows. Interestingly, our method was still faster than naive convolution. Specifically, it showed significant superiority to convolution when $\sigma < 1.6$. This performance comes from the number of reference pixels: naive convolution requires $(2N - 1)$ reference pixels but our method only four reference pixels. Since it is time-consuming for modern CPU to perform memory access, our method with less memory access runs faster than the convolution. The DCT-based $O(1)$ GF has achieved $O(1)$ filtering by truncating negligible cosine terms using the parameter K in 3. However, these results revealed that our method has high potential for efficiently filtering arbitrary even-symmetric kernels whose short window length without truncating frequency. In approximate accuracy, PSNR is higher than 60 dB for all the cases, which means it is sufficiently acceptable for visual perception.

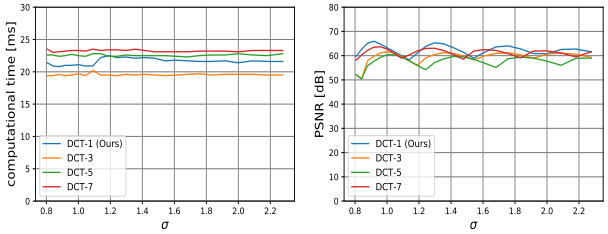
IV. CONCLUSIONS

This paper presented $O(1)$ GF with DCT-1 specialized in short window length. Our experiments revealed that our method accelerated $O(1)$ GF based on DCT-1 and outperformed the $O(1)$ GF of other DCT-type considering computational time and PSNR. Moreover, we found that, in the case of Gaussian kernel, our method in full-frequency case was able to run faster than convolution. As future tasks, we will evaluate real performance of our method for other even-symmetric kernels and volume data filtering.



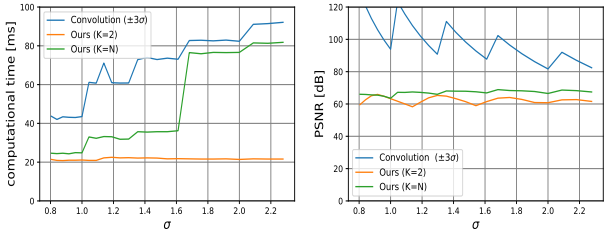
(a) Computational time in rows (b) Computational time in columns (c) Total computational time

Fig. 2. The figures show evaluation of the convolution, the existing method and our method. Our method is about 20 % faster than the existing method over $\sigma = 0.8052 \sim 2.2776$, i.e., $N = 2, 3 \ldots 7$ because MUL and ACCESS are reduced as compared with the existing method.



(a) Total computational time (b) PSNR

Fig. 3. (a) Our method is about 11 % slower than the DCT-3 method. (b) Our method showed about 13 % higher PSNR than the DCT-3 method.



(c) Total computational time (d) PSNR

Fig. 4. Our method could run faster than convolution if $N \leq 5$ and the kernel is even-symmetric. This is because full-frequency case ($K = N$) is about 50 % faster than the convolution over $\sigma = 0.8052 \sim 1.6105$, i.e., $N = 2, 3, 4, 5$

ACKNOWLEDGMENT

This work was partly supported by JSPS KAKENHI (No. JP16K16092, JP17H01764, and JP18K18076).

REFERENCES

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

[2] Q. Yang, R. Yang, J. Davis, and D. Nister, "Spatial-depth super resolution for range images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, June 2007.

[3] 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.

[4] Q. Yang, K. H. Tan, and N. Ahuja, "Real-time $O(1)$ bilateral filtering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, June 2009, number 1, pp. 557–564.

[5] K. N. Chaudhury, "Acceleration of the shiftable $O(1)$ algorithm for bilateral filtering and nonlocal means," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1291–1300, Apr. 2013.

[6] Q. Yang, N. Ahuja, and K.-H. Tan, "Constant time median and bilateral filtering," *Int. J. Comput. Vis. (IJCV)*, vol. 112, no. 3, pp. 307–318, May 2015.

[7] G. Deng, "Fast compressive bilateral filter," *Electronics Letters*, vol. 53, pp. 150–152, Feb. 2017.

[8] K. Sugimoto and S. Kamata, "Compressive bilateral filtering," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3357–3369, Nov. 2015.

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

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

[11] I. T. Young and L. J. van Vliet, "Recursive implementation of the Gaussian filter," *Signal Process.*, vol. 44, no. 2, pp. 139–151, June 1995.

[12] 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.

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

[14] 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.

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

[16] K. Sugimoto and S. Kamata, "Efficient constant-time Gaussian filtering with sliding DCT/DST-5 and dual-domain error minimization," *ITE Trans. Media Technol. Appl.*, vol. 3, no. 1, pp. 12–21, 2015.

[17] K. Sugimoto, S. Kyochi and S. Kamata, "Universal approach for DCT-based constant-time Gaussian filter with moment preservation," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP2018)*, Apr. 2018, pp. 1498–1502.

[18] F. C. Crow, "Summed-area tables for texture mapping," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 18, no. 3, pp. 207–212, July 1984.

[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)*, Dec. 2001, vol. 1, pp. 511–518.

[20] K. Sugimoto and S. Kamata, "Fast Gaussian filter with second-order shift property of DCT-5," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sept. 2013, pp. 514–518.

[21] 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.

[22] D. Charalampidis, "Recursive implementation of the Gaussian filter using truncated cosine functions," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3554–3565, 2016.

[23] P. Getreuer, "A survey of Gaussian convolution algorithms," *Image Processing On Line*, vol. 3, pp. 286–310, 2013.

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