Fast Template Matching With Polynomials

Shinichiro Omachi, *Member, IEEE*, and Masako Omachi

***Abstract*—Template matching is widely used for many applications in image and signal processing. This paper proposes a novel template matching algorithm, called *algebraic template matching*. Given a template and an input image, algebraic template matching efficiently calculates similarities between the template and the partial images of the input image, for various widths and heights. The partial image most similar to the template image is detected from the input image for any location, width, and height. In the proposed algorithm, a polynomial that approximates the template image is used to match the input image instead of the template image. The proposed algorithm is effective especially when the width and height of the template image differ from the partial image to be matched. An algorithm using the Legendre polynomial is proposed for efficient approximation of the template image. This algorithm not only reduces computational costs, but also improves the quality of the approximated image. It is shown theoretically and experimentally that the computational cost of the proposed algorithm is much smaller than the existing methods.**

***Index Terms*—Algebraic template matching, image processing, object detection, polynomial, template matching.**

I. Introduction

TEMPLATE matching is one of the most common techniques used in signal and image processing. Template matching applications include image retrieval [3], image recognition [13], image mosaicing or registration [2], object detection [7], and stereo matching [17]. Given an input and a template image, the matching algorithm finds the partial image that most closely matches the template image in terms of specific criterion, such as the Euclidean distance or cross correlation [5]. However, conventional template matching methods using a template image consume a large amount of computational time. A number of techniques have been investigated with the intent of speeding up the template matching [8]–[10], [15]–[18], [20], [23], [24].

The coarse-to-fine strategy, proposed by Rosenfeld and VanderBrug [15], [24], is a well-known approach to reduce the computational cost of template matching. This strategy uses a low-resolution template and an input image for initial coarse matching. Matching between high-resolution template and input images is applied for fine matching only when there is high similarity in the coarse matching. Tanimoto proposed a hierarchical search algorithm and introduced pyramid data structures [20]. However, this strategy cannot always find the most similar part in the fine matching since the other part may be detected in coarse matching. This strategy can be easily combined with other speed-up techniques.

The fast Fourier transform (FFT) helps calculate cross correlations efficiently. The Fourier transform of the correlation of two functions is the product of their Fourier transforms [14]. The cross correlation of the template image and every part of the input image can be calculated simultaneously. The calculated non-normalized cross correlation can be efficiently normalized by the algorithm provided by Lewis [10]. Uenohara and Kanade proposed a method that combines the FFT and the Karhunen–Loeve transform for matching images with a large set of template images [23].

Another strategy to accelerate template matching is estimating the upper boundary of the similarity and eliminating unnecessary calculations. In this strategy, the reduction rate of the computational cost depends on the template and input images. Gharavi–Alkhansari combined this strategy with the coarse-to-fine strategy, and proposed a method for estimating a threshold in the coarse search and pruning the candidates in the fine search [9]. Stefano and Mattoccia proposed an efficient template matching algorithm by defining an upper-boundary function for normalized cross correlation, using the inequality of arithmetic and geometric means [18].

In the field of stereo matching, given a left and a right image, their corresponding points can be found. The windows in both images move together in the same way, and similarities between these windows are calculated. In this situation, an incremental calculation scheme is effective [8], [17]. That is, if the similarity of a region is already calculated, the similarity of an overlapped region can be calculated using only the differences in these regions. However, this situation is different from template matching where the template is fixed and the location of the partial image in the input image varies.

Approximation of the image by a function may reduce the computational cost. Schweitzer *et al.* proposed an efficient template matching algorithm that introduced integral images and approximated the input image with second- or third-order polynomials [16].

All of these methods handle a fixed-size template image. Some methods (e.g., those using the FFT or incremental algorithms mentioned above) efficiently calculate similarities between a template and the fixed-size partial images at various *locations*. However, if the width and height of the partial image to be detected within the input image are unknown, we must change them and then repeat the procedure.

This paper proposes a novel template matching method, called *algebraic template matching*. Algebraic template matching (ATM) efficiently calculates similarities between the template and partial images of the input image of various

Manuscript received September 28, 2006; revised February 25, 2007. This work was supported in part by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research, 18500125, 2006. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eli Saber.

S. Omachi is with the Graduate School of Engineering, Tohoku University, Sendai-shi 980-8579, Japan (e-mail: machi@ecei.tohoku.ac.jp).

M. Omachi is with the Faculty of Science and Technology, Tohoku Bunka Gakuen University, Sendai-shi 981-8551, Japan (e-mail: fan@ait.tbgu.ac.jp).

Digital Object Identifier 10.1109/TIP.2007.901243

widths and heights. First, the template image is approximated by a high-order polynomial. Then, this polynomial is used to match an input image instead of the template image. We propose an efficient incremental algorithm for calculating the similarities using this polynomial. According to the authors, there is no algorithm that calculates similarities incrementally like the proposed algorithm by changing the widths and heights of the partial image.

The polynomial has been studied as a tool for representing shapes [21], [22]. However, in many cases, the approximation by the polynomial is used for feature extraction. That is, the coefficients of the polynomial that represent the shape are regarded as the features of the shape. Our approach approximates the image with a high-order polynomial used as a template. This approach has not been previously considered for the following reasons.

1) Approximation by a high-order polynomial using the conventional least-square method requires large amounts of computational time, and often fails due to truncation error.
2) There is no use in representing the image with a polynomial, as an algorithm to efficiently use the polynomial had not been found.

This paper uses an algorithm based on the Legendre polynomial to solve the first problem. The latter problem is addressed with an efficient incremental calculation algorithm.

The organization of the rest of the paper is as follows. First, in Section II, we present an algorithm for approximating an image with a polynomial. In Section III, we propose our template matching algorithm, called algebraic template matching, which uses the polynomial. In Section IV, we describe the processing time and matching accuracy results to show the effectiveness of the proposed algorithm.

II. Image Representation by Polynomials

According to the Stone-Weierstrass theorem [6], any 2-D continuous function defined in closed intervals can be approximated by a polynomial with two variables. However, the approximation of an image by a polynomial using the classical least-square method requires a great deal of computation and often fails due to truncation error. Thus, we propose an approximation algorithm that uses the Legendre polynomial.

A. Approximation by the Least-Square Method

Given an $M_t \times N_t$ gray-scale image, a simple way of finding a polynomial that approximates the image is to use the least-square method. Let $z(i,j)$ be the intensity of the ith column and the jth row of the image $(0 \le i \le M_t - 1, 0 \le j \le N_t - 1)$, and $f(x,y)$ be the polynomial of degree d that is defined in $0 \le x \le 1, 0 \le y \le 1$ as

$$f(x,y) = \sum_{0 \le k+l \le d} a_{kl} x^k y^l. \tag{1}$$

The least-square method finds the coefficients a_{kl} of $f(x,y)$ that minimizes

$$J = \sum_{i=0}^{M_t-1} \sum_{j=0}^{N_t-1} \left(z(i,j) - f\left(\frac{i}{M_t}, \frac{j}{N_t} \right) \right)^2 \tag{2}$$

by solving the following L linear equations:

$$\frac{\partial J}{\partial a_{kl}} = 0 \quad (0 \le k+l \le d) \tag{3}$$

where $L = (d+1)(d+2)/2$ is the number of coefficients a_{kl}.

The computational costs of obtaining these equations and solving them are $O(M_t N_t L^2)$ and $O(L^3)$, respectively. Therefore, the total computational cost is $O(M_t N_t d^4 + d^6)$. As the value of d increases, the approximation requires more computation. Moreover, in order to solve the linear equations, inverse matrix calculation or a similar approach is needed. The truncation error of the inverse matrix calculation may degrade the quality of the approximation.

B. Legendre Polynomials

Legendre polynomials [1] are orthogonal functions that are defined in $-1 \le x \le 1$. The Legendre polynomial of degree $n, P_n(x)$, is written as

$$P_n(x) = \sum_{k=0}^{[n/2]} (-1)^k \frac{(2n-2k)!}{2^n k!(n-k)!(n-2k)!} x^{n-2k}. \tag{4}$$

Equation (4) shows that $P_n(x)$ has only even powers of x for even n, and odd powers for odd n. It is known that the following formulae are satisfied:

$$\begin{cases} P_0(x) = 1 \\ P_1(x) = x \\ P_{n+1}(x) = \dfrac{2n+1}{n+1} x P_n(x) - \dfrac{n}{n+1} P_{n-1}(x) \quad (n \ge 1) \end{cases}. \tag{5}$$

The set $\{P_n(x)\}$ is orthogonal because the following equation is satisfied:

$$\int_{-1}^{1} P_m(x) P_n(x) dx = \begin{cases} \dfrac{2}{2n+1}, & (m = n) \\ 0, & (\text{otherwise}) \end{cases}. \tag{6}$$

Since the set is also a complete system, any function $h(x)$ defined in $-1 \le x \le 1$ can be expanded by the Legendre polynomials as

$$h(x) = \sum_{n=0}^{\infty} c_n P_n(x) \tag{7}$$

where

$$c_n = \frac{2n+1}{2} \int_{-1}^{1} h(x) P_n(x)\, dx. \tag{8}$$

The multivariate Legendre polynomials [4] with q variables are defined in $-1 \le x_1 \le 1, \ldots, -1 \le x_q \le 1$ as

$$P_{n_1 \cdots n_q}(x_1, \ldots, x_q) = P_{n_1}(x_1) \cdots P_{n_q}(x_q). \tag{9}$$

For example, the multivariate Legendre polynomials with two variables x and y up to second degree are

$$\begin{aligned} P_{00}(x,y) &= 1 \\ P_{10}(x,y) &= x \\ P_{01}(x,y) &= y \\ P_{20}(x,y) &= \frac{3}{2}x^2 - \frac{1}{2} \\ P_{11}(x,y) &= xy \\ P_{02}(x,y) &= \frac{3}{2}y^2 - \frac{1}{2}. \end{aligned}$$

Since the set of the multivariate Legendre polynomials is orthogonal, a 2-D continuous function $h(x,y)$ defined in $-1 \le x \le 1, -1 \le y \le 1$ can be expanded as

$$h(x,y) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} c_{mn} P_m(x) P_n(y) \tag{10}$$

where

$$c_{mn} = \frac{(2m+1)(2n+1)}{4} \times \int_{-1}^{1} \int_{-1}^{1} h(x,y) P_m(x) P_n(y)\, dx\, dy. \tag{11}$$

C. Approximation Using Legendre Polynomials

We propose an algorithm to approximate a 2-D image using the completeness of the Legendre polynomials. If we rewrite (4) as

$$P_n(x) = \sum_{k=0}^{n} p_{nk} x^k \tag{12}$$

we can calculate the coefficients p_{nk} $(0 \le n, 0 \le k \le n)$ with the following recurrent formulae:

$$\begin{cases} p_{00} = 1 \\ p_{10} = 0, \quad p_{11} = 1 \\ p_{(n+1)k} = \dfrac{2n+1}{n+1} p_{n(k-1)} - \dfrac{n}{n+1} p_{(n-1)k} \quad (n \ge 1) \end{cases} \tag{13}$$

where

$$p_{nk} = 0 \quad (k < 0, n < k).$$

Given an image $z(i,j)(0 \le i \le M_t - 1, 0 \le j \le N_t - 1)$, first, $z(i,j)$ is approximated with the polynomial $h(x,y)$ of degree d, defined in $-1 \le x \le 1, -1 \le y \le 1$, as

$$h(x,y) = \sum_{0 \le m+n \le d} c_{mn} P_m(x) P_n(y) \tag{14}$$

where

$$c_{mn} = \frac{(2m+1)(2n+1)}{M_t N_t} \times \sum_{i=0}^{M_t-1} \sum_{j=0}^{N_t-1} z(i,j)\, P_m\left(\frac{2i}{M_t} - 1\right) P_n\left(\frac{2j}{N_t} - 1\right). \tag{15}$$

Equatoin (15) shows that the coefficients can be obtained by calculating the inner products instead of calculating the inverse matrix. Note that since we replaced the integral of (11) with the summation of (15), in order to achieve a good approximation, the values of M_t and N_t should be large.

Now we can approximate the image $z(i,j)$ by the polynomial $f(x,y)$ defined in $0 \le x \le 1, 0 \le y \le 1$ as

$$\begin{aligned} &f(x,y) \\ &\quad = h(2x-1, 2y-1) \\ &\quad = \sum_{0 \le m+n \le d} c_{mn} P_m(2x-1) P_n(2y-1) \\ &\quad = \sum_{0 \le m+n \le d} c_{mn} \sum_{k'=0}^{m} \sum_{l'=0}^{n} p_{mk'} (2x-1)^{k'} p_{nl'} (2y-1)^{l'}. \end{aligned} \tag{16}$$

Comparing the coefficients of $x^k y^l$ of (1) and (16), we have

$$a_{kl} = \sum_{m \ge k, n \ge l, m+n \le d} \sum_{k'=k}^{m} \sum_{l'=l}^{n} w_{klk'l'} c_{mn} p_{mk'} p_{nl'} \tag{17}$$

where

$$w_{klk'l'} = (-1)^{k'+l'-k-l} 2^{k+l} \binom{k'}{k} \binom{l'}{l}. \tag{18}$$

In the proposed method, the coefficients of (1) are obtained by the linear transformation of (17). In this process, numerical errors are introduced. However, the errors are much smaller than those introduced by calculating the inverse matrix in the least-square method. This is confirmed using a real image in Section II-D (see Figs. 2 and 4).

The computational costs for calculating (15) and calculating a_{kl} for all k and l in (17) are $O(M_t N_t d^3)$ and $O(d^4)$, respectively. Therefore, the total computational cost is $O(M_t N_t d^3 + d^4)$, which is much smaller than that of the least-square method.

Note that there are orthogonal polynomials other than the Legendre polynomials that can be used for representing an image. These include the Chebyshev polynomials, the Laguerre polynomials, and the Hermite polynomials [1]. However, all these polynomials have a *weighting function* and, therefore, cannot be converted into the form of (1). Since the proposed algorithm described in Section III assumes that the approximated image is represented in the form of (1), these polynomials cannot be used directly for the proposed algorithm.

D. Example of Image Representation

In order to show the effectiveness of representing an image by the polynomial and the presented algorithm for calculating

Fig. 1. Test image (128 × 128).

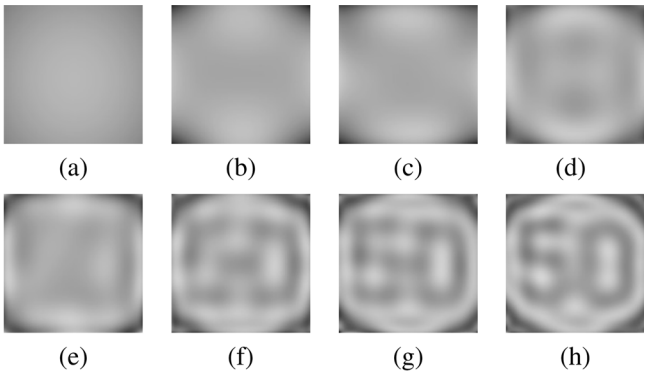

Fig. 2. Images generated by the polynomials calculated using the Legendre polynomials. (a) $d = 2$. (b) $d = 4$. (c) $d = 6$. (d) $d = 8$. (e) $d = 10$. (f)$d = 12$. (g) $d = 14$. (h) $d = 16$.

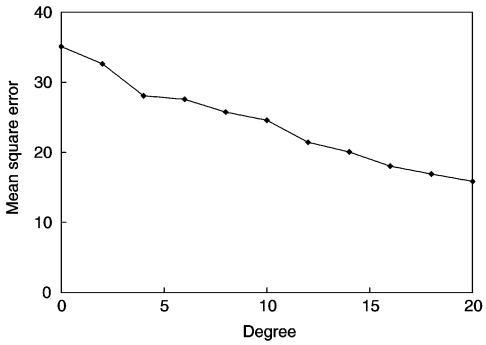

Fig. 3. Mean square errors.

the polynomial, an example for approximation of an image by the polynomial is provided. A gray-scale image of 128×128 pixels is shown in Fig. 1. The image is a part of a road sign. The proposed method and an algorithm using the least-square method were implemented in C language, using 64-bit floating point.

Fig. 2 shows images generated by the polynomials of different degrees d, applying the algorithm described in Section II-C to the image. It is clarified that the generated image becomes more similar to the original image as the value of d increases. The mean square errors between the original and the generated images are shown in Fig. 3. The horizontal axis is the degree d. When $d = 0$, the image is represented by a single gray level. The mean square error for $d = 0$ is the standard deviation of the intensities of the pixels in the original image, and can be used as a criterion to evaluate the quality of the mean square errors when $d > 0$. If d becomes large, the mean square error becomes sufficiently small compared to that of $d = 0$. This means an image can be approximated by the polynomial to certain extent.

Fig. 4 shows the images generated by the polynomials calculated by the least-square method. The images obtained for

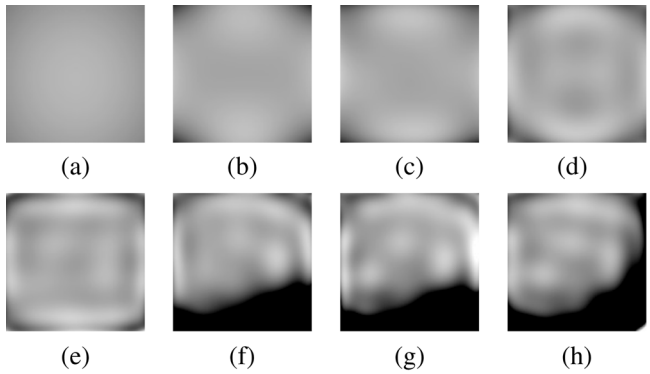

Fig. 4. Images generated by the polynomials calculated by the least-square method. (a) $d = 2$. (b) $d = 4$. (c) $d = 6$. (d) $d = 8$. (e) $d = 10$. (f) $d = 12$. (g) $d = 14$. (h) $d = 16$.

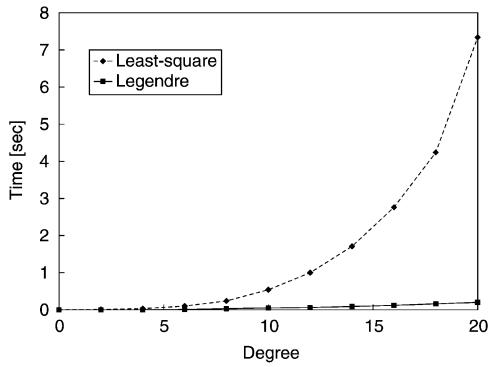

Fig. 5. Processing time for finding a polynomial that approximates an image.

$d \leq 10$ in Fig. 4 are similar to the ones in Fig. 2. However, the images are degraded for $d \geq 12$, because of the occurrence of truncation error when solving (3).

The processing times of a Xeon 2.4-GHz computer for the two methods are shown in Fig. 5. As the degree d increases, the computational cost of the least-square method increases rapidly. However, when using the Legendre polynomials, the increase in processing time is not large. Obtaining a polynomial of degree $d = 10$ [Fig. 2(e)] requires only 0.04 s. These results show the effectiveness of the proposed algorithm using the Legendre polynomials.

III. Algebraic Template Matching

We propose an algorithm that efficiently calculates similarities between the template and the partial images of various widths and heights in the input image. First, the definition of the problem of template matching is clarified. Given an $M_t \times N_t$ template image and an $M \times N$ input image, the partial image in the input image most similar to the template, regardless of location, width, and height, is detected as shown in Fig. 6. To generalize the problem, we limit the maximum size of the partial image to $M_1 \times N_1$.

Various criteria, such as the Euclidean distance or correlations, can be used for template matching. This paper uses normalized cross correlation (NCC). By using polynomials, we can rapidly calculate NCCs between the template image and partial images of the input image.

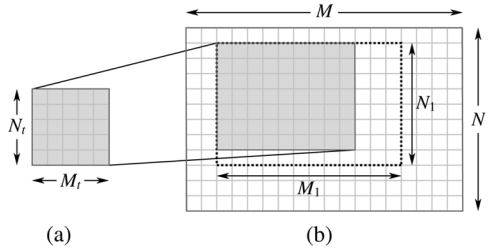

Fig. 6. Problem definition of template matching. (a) Template. (b) Input image.

In Section III-A and B, we explain the algorithms for calculating NCCs between the template and the input by fixing the origin. Since the 2-D case is too complex to explain, the principle of the proposed algorithm is described using a 1-D example. Then the algorithm for a 2-D case is described by referring to the 1-D case. In Section III-C, we describe the algorithm for algebraic template matching.

A. One-Dimensional Case

Given a template $f(x)$ $(0 \leq x \leq 1)$ and an input $g(x)(0 \leq x \leq M)$ shown in Fig. 7(a) and (b), we can calculate the normalized cross correlation $\mathrm{NCC}(\alpha)$ between $g(x)$ and $f(x/\alpha)$ to find the maximum NCC. Here, $f(x/\alpha)$ is an expanded or contracted pattern of $f(x)$, as shown in Fig. 7(c). If the origin is fixed, $\mathrm{NCC}(\alpha)$ can be written as

$$\mathrm{NCC}(\alpha) = \frac{\int_0^\alpha f(x/\alpha)g(x)dx}{\sqrt{\int_0^\alpha f^2(x/\alpha)dx \int_0^\alpha g^2(x)dx}} = \frac{\int_0^\alpha f(x/\alpha)g(x)dx}{\sqrt{\alpha} I_1 \sqrt{\int_0^\alpha g^2(x)dx}} \tag{19}$$

where

$$I_1 = \sqrt{\int_0^1 f^2(x)dx} \tag{20}$$

is a constant.

If the input has discrete values $G(i)$ $(i = 0, 1, \ldots, M-1)$, the function of NCC is written as

$$\mathrm{NCC}(\alpha) = \frac{\sum_{i=0}^{\alpha-1} f(i/\alpha)G(i)}{I_1 \sqrt{\alpha \sum_{i=0}^{\alpha-1} G^2(i)}}. \tag{21}$$

Here, the value of α is limited to the following integers:

$$\alpha = 1, 2, \ldots, M. \tag{22}$$

If $f(x)$ is also discrete, to calculate $\mathrm{NCC}(\alpha)$ for M kinds of α, (21) should be calculated for each α. The computational cost will be

$$1 + 2 + \cdots + M = \frac{M(M+1)}{2} = O(M^2). \tag{23}$$

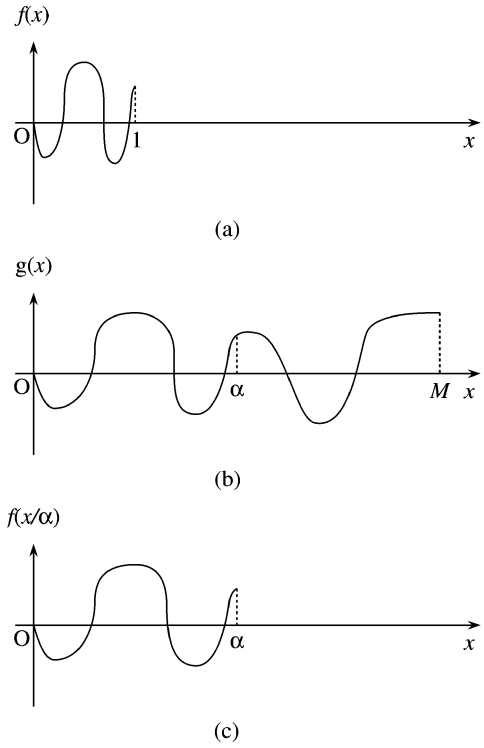

Fig. 7. Calculation of NCC between a template and an input (1-D case). (a) Template. (b) Input. (c) Expanded template.

If $f(x)$ is represented by a polynomial as

$$f(x) = \sum_{k=0}^{d} a_k x^k \tag{24}$$

(21) will be written as

$$\mathrm{NCC}(\alpha) = \frac{\sum_{i=0}^{\alpha-1} \sum_{k=0}^{d} a_k i^k G(i)/\alpha^k}{I_1 \sqrt{\alpha \sum_{i=0}^{\alpha-1} G^2(i)}}. \tag{25}$$

Let

$$b_k(\alpha) = \sum_{i=0}^{\alpha-1} i^k G(i) \tag{26}$$

$$s(\alpha) = \sum_{i=0}^{\alpha-1} G^2(i). \tag{27}$$

The values of $b_k(\alpha)$ $(k = 0, 1, \ldots, d)$ and $s(\alpha)$ for all α can be calculated successively by the following recurrent formulae:

$$\begin{cases} b_k(1) = 0^k G(0) \\ b_k(\alpha) = b_k(\alpha-1) + (\alpha-1)^k G(\alpha-1) \quad (\alpha \geq 2) \end{cases} \tag{28}$$

$$\begin{cases} s(1) = G^2(0) \\ s(\alpha) = s(\alpha-1) + G^2(\alpha-1) \quad (\alpha \geq 2) \end{cases}. \tag{29}$$

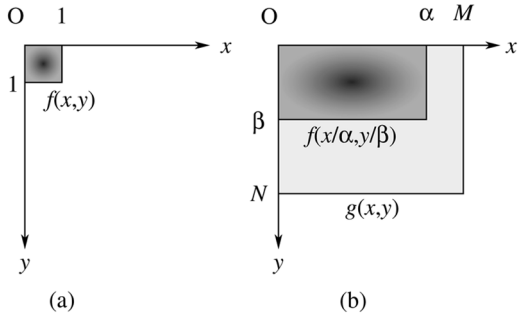

Fig. 8. Calculation of NCC between a template and an input (2-D case). (a) Template. (b) Input.

Using these values, $\mathrm{NCC}(\alpha)(\alpha = 1, 2, \ldots, M)$ is calculated as

$$\mathrm{NCC}(\alpha) = \frac{\sum_{k=0}^{d} a_k b_k(\alpha)/\alpha^k}{I_1\sqrt{\alpha s(\alpha)}}. \tag{30}$$

Since the computational cost for each α is $O(d)$, the total computational cost is $O(dM)$.

B. Two-Dimensional Case

Given a template $f(x, y)$ and an input $g(x, y)$, $\mathrm{NCC}(\alpha, \beta)$ is defined as the normalized cross correlation between $g(x, y)$ and $f(x/\alpha, y/\beta)$ as shown in Fig. 8.

Let the intensity of the ith column and the jth row of the input image be $G(i, j)$ $(i = 0, 1, \ldots, M-1,\ j = 0, 1, \ldots, N-1)$, and the template be

$$f(x, y) = \sum_{0 \le k+l \le d} a_{kl} x^k y^l \quad (0 \le x \le 1, 0 \le y \le 1) \tag{31}$$

$\mathrm{NCC}(\alpha, \beta)$ is written as

$$\mathrm{NCC}(\alpha, \beta) = \frac{\sum_{i=0}^{\alpha-1}\sum_{j=0}^{\beta-1}\sum_{0 \le k+l \le d} a_{kl} i^k j^l G(i, j)/\alpha^k \beta^l}{I_2\sqrt{\alpha\beta\sum_{i=0}^{\alpha-1}\sum_{j=0}^{\beta-1} G^2(i, j)}} \tag{32}$$

where

$$I_2 = \sqrt{\int_0^1 \int_0^1 f^2(x, y)\, dx\, dy} \tag{33}$$

is a constant that can be calculated analytically, and

$$\begin{cases} \alpha = 1, 2, \ldots, M \\ \beta = 1, 2, \ldots, N. \end{cases} \tag{34}$$

Let

$$\tilde{b}_{kl}(\alpha, \beta) = \sum_{i=0}^{\alpha-1} i^k (\beta - 1)^l G(i, \beta - 1) \tag{35}$$

$$b_{kl}(\alpha, \beta) = \sum_{i=0}^{\alpha-1}\sum_{j=0}^{\beta-1} i^k j^l G(i, j) \tag{36}$$

$$\tilde{s}(\alpha, \beta) = \sum_{i=0}^{\alpha-1} G^2(i, \beta - 1) \tag{37}$$

$$s(\alpha, \beta) = \sum_{i=0}^{\alpha-1}\sum_{j=0}^{\beta-1} G^2(i, j). \tag{38}$$

Here, $\tilde{b}_{kl}(\alpha, \beta)$ and $\tilde{s}(\alpha, \beta)$ are the sums of the βth row, and $b_{kl}(\alpha, \beta)$ and $s(\alpha, \beta)$ are the sums from the first row to the βth row.

The values of $\mathrm{NCC}(\alpha, \beta)$ can be calculated successively by the following recurrent formulae:

$$\begin{cases} \tilde{b}_{kl}(1, \beta) = 0^k(\beta - 1)^l G(0, \beta - 1) \\ \tilde{b}_{kl}(\alpha, \beta) = \tilde{b}_{kl}(\alpha - 1, \beta) \\ \quad +(\alpha - 1)^k(\beta - 1)^l G(\alpha - 1, \beta - 1) \quad (\alpha \ge 2) \end{cases} \tag{39}$$

$$\begin{cases} b_{kl}(\alpha, 1) = \tilde{b}_{kl}(\alpha, 1) \\ b_{kl}(\alpha, \beta) = b_{kl}(\alpha, \beta - 1) + \tilde{b}_{kl}(\alpha, \beta) \quad (\beta \ge 2) \end{cases} \tag{40}$$

$$\begin{cases} \tilde{s}(1, \beta) = G^2(0, \beta - 1) \\ \tilde{s}(\alpha, \beta) = \tilde{s}(\alpha - 1, \beta) \\ \quad +G^2(\alpha - 1, \beta - 1) \quad (\alpha \ge 2) \end{cases} \tag{41}$$

$$\begin{cases} s(\alpha, 1) = \tilde{s}(\alpha, 1) \\ s(\alpha, \beta) = s(\alpha, \beta - 1) + \tilde{s}(\alpha, \beta) \quad (\beta \ge 2) \end{cases} \tag{42}$$

$$\mathrm{NCC}(\alpha, \beta) = \frac{\sum_{0 \le k+l \le d} a_{kl} b_{kl}(\alpha, \beta)/\alpha^k \beta^l}{I_2\sqrt{\alpha\beta s(\alpha, \beta)}}. \tag{43}$$

Fig. 9 displays the details of the calculation. In Fig. 9, the black rectangle represents the region in which the similarity to the template is calculated. Here, only the underlined parts of each expression need to be calculated each time. The pixel corresponding to the underlined parts is shown as a gray pixel.

First, $\mathrm{NCC}(1, 1)$, which is the NCC between the 1×1 region in Fig. 9(a) and the template, is calculated. From (43), only $b_{kl}(1, 1)$ and $s(1, 1)$ are necessary for calculating $\mathrm{NCC}(1, 1)$. These values can be calculated by using the value of $G(0, 0)$. Next, $\mathrm{NCC}(2, 1)$, which is the NCC between the template and the 2×1 region in Fig. 9(b) is calculated. In this case, $b_{kl}(2, 1)$ and $s(2, 1)$ are necessary for calculation. Each value can be calculated by using the values of $G(1, 0)$ and $b_{kl}(1, 1)$ or $s(1, 1)$. Accordingly, the NCC between the template and the 2×1 region can be calculated using the value of $G(1, 0)$.

Fig. 9(c) and (d) shows the 1×2 and 2×2 regions, respectively. In these cases, NCC can be calculated by adding $\tilde{b}_{kl}(\alpha, 2)$ to $b_{kl}(\alpha, 1)$ and $\tilde{s}(\alpha, 2)$ to $s(\alpha, 1)$. In this way, each NCC can be calculated using the value of one pixel.

There are MN ways of selecting α and β, therefore, the computational cost for calculating MN NCCs is $O(d^2MN)$. On the other hand, for conventional template matching, which calculates NCC for every width and height separately (hereafter called simple matching), the computational cost is $O(M^2N^2)$.

C. Algorithm for Algebraic Template Matching

Next, we show the algorithm for algebraic template matching (ATM). Given an $M_t \times N_t$ template image and an $M \times N$ input image, we can find the partial image in the input image that is most similar to the template image, regardless of its location, width, or height.

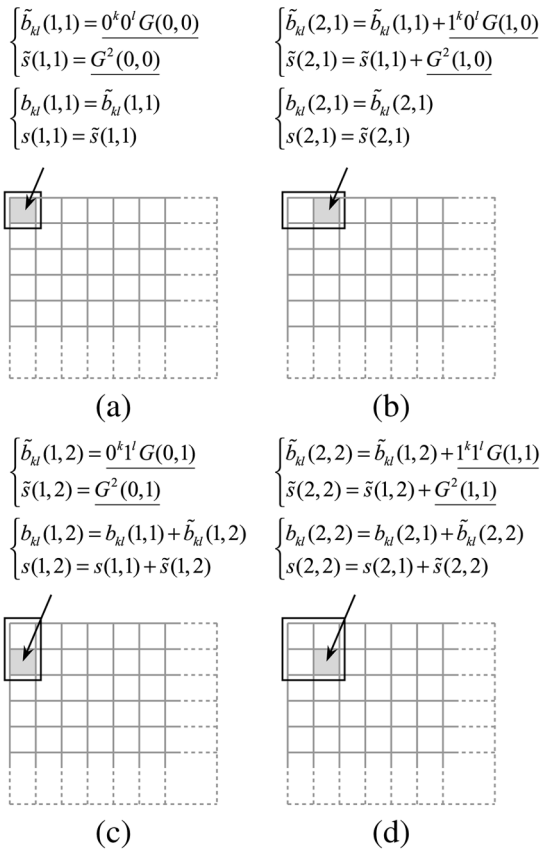

Fig. 9. Calculation of NCCs for 2-D case.

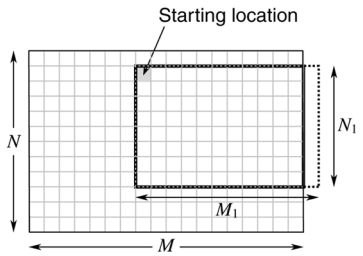

Fig. 10. Search for partial images.

First, the algorithm described in Section II-C finds the polynomial that approximates the template image. Then, we calculate the NCC between the template and each partial image smaller than, or equal to, $M_1 \times N_1$. As shown in Fig. 10, we select a starting location in the input image and consider an $M_1 \times N_1$ rectangle whose top left corner is located at the starting location. Here, the part that is not included in the input image need not to be considered. The algorithm described in Section III-B is applied to this rectangle. Since there are MN different ways of selecting the starting location, and the computational cost for searching within the rectangle is $O(d^2 M_1 N_1)$, the total computational cost of the algorithm is $O(d^2 MNM_1N_1)$. As described in Section III-B, the cost of simple matching within a rectangle is $O(M_1^2 N_1^2)$, and the total cost is $O(MNM_1^2N_1^2)$.

Fig. 11. Original input image.

Fig. 12. Example of the matching result. Input image size is 80×60 and degree of the polynomial is eight.

IV. Experiments

In order to illustrate the effectiveness of ATM, we carried out experiments to examine the processing time and the matching accuracy. A few examples of applying ATM to object detection in scene images are provided.

A. Processing Time

The processing time for ATM was compared to *very fast template matching* [16] (VFTM), a method that use FFT, and simple matching. VFTM seems to be the algorithm most closely related to the proposed algorithm, in the sense that, in both methods, a polynomial approximates an image. The VFTM algorithm is summarized as follows.

1) Compute the moments of the template.
2) Compute the integral images.
3) Compute the moments of the input image for each location in the input image.
4) Compute NCC at each location.

The number of moments is given by $L = (d+1)(d+2)/2$, and the calculation of each moment requires summation up to L terms, where d is the degree of the polynomial. Therefore, the computational cost of step 3) for each location of the input image is $O(L^2) = O(d^4)$.

In the implementation of VFTM and FFT, the width and height of the template image were changed and the procedure was repeated. Since the calculation of moments requires $O(d^4)$ for each location, the total computational cost of VFTM is $O(d^4 MNM_1N_1)$. According to the authors, VFTM is the only existing algorithm that has a computational cost of $O(MNM_1N_1)$, except for the coarse-to-fine strategy, when d is regarded as a constant. The computational cost of FFT is $O(M_1N_1\mathcal{M}^2 \log(\mathcal{M}))$, where $\mathcal{M} = 2^{\max(\lceil \log_2 M \rceil, \lceil \log_2 N \rceil)}$.

Various image sizes were generated by reducing the 320×240 image shown in Fig. 11. These images were used as input images and the image shown in Fig. 1 was used

TABLE I
PROCESSING TIME [SECONDS]

Size	ATM (Proposed method)						VFTM				FFT	Simple
	$d=2$	$d=4$	$d=6$	$d=8$	$d=10$	$d=12$	$d=2$	$d=4$	$d=6$	$d=8$		
40×30	0.04	0.07	0.14	0.21	0.29	0.42	0.35	1.28	4.05	11.1	2.01	1.95
60×45	0.12	0.22	0.36	0.56	0.81	1.17	0.98	3.65	11.2	31.4	2.36	7.86
80×60	0.26	0.46	0.76	1.11	1.61	2.33	2.01	7.38	22.5	64.4	10.8	18.1
100×75	0.42	0.78	1.29	1.90	2.72	3.86	3.47	12.4	38.2	110	10.9	32.6
120×90	0.66	1.17	1.96	2.88	4.07	5.77	5.43	18.8	58.0	169	11.0	51.4
160×120	1.23	2.24	3.72	5.46	7.82	11.1	10.5	35.7	111	328	49.2	102

as a template image, and $M_1 = N_1 = 30$. Fig. 12 displays an example of the matching result.

The processing times for a Xeon 2.4-GHz computer using these methods with images of various sizes are shown in Table I. Since processing times for ATM and VFTM depend on the degree of the polynomial, various degrees are tested. The values of d in the table are the degrees of the polynomial. The processing time for ATM includes the processing time for approximation described in Section II-C.

From the table, it is clarified that the processing time for ATM is much less than that for VFTM, FFT, and simple matching. For example, using the 80×60 image and polynomial of degree eight (the matching result of Fig. 12 was obtained), the processing time for ATM was 1.11 s, however, FFT and simple matching required 10.8 and 18.1 s, respectively. ATM is more effective than FFT and simple matching as the values of M_1 and N_1 increase. Note that VFTM was much faster than FFT and simple matching when the degree of the polynomial d was small. However, VFTM was not fast when d was large, since its processing time is proportional to d^4. For any degree d, and any input-image size, ATM was faster than FFT.

Fig. 13(a) shows the relationship between the processing time and MNM_1N_1 for various degrees, d. Fig. 13(b) shows the relationship between the processing time and d^2 for various input-image sizes. Experimentation verified that the processing time of ATM is $O(d^2MNM_1N_1)$.

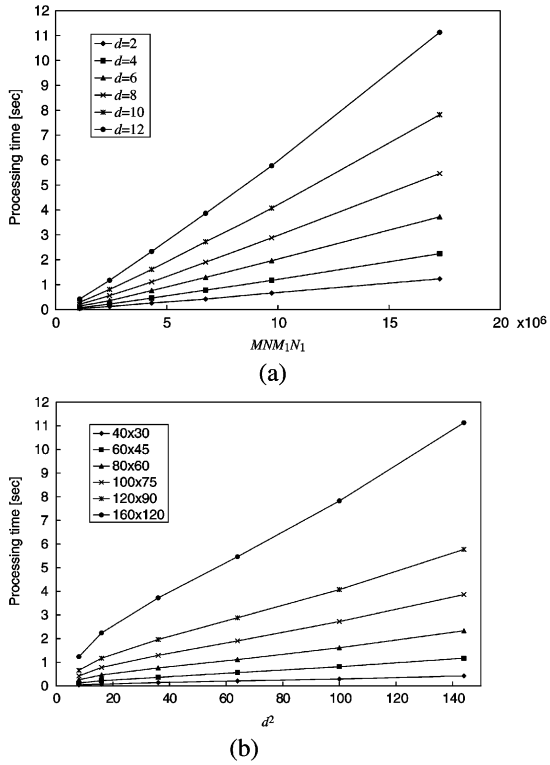

Fig. 13. Relationship between the processing time and (a) MNM_1N_1 and (b) d^2.

B. Matching Accuracy

The matching accuracy for ATM was tested quantitatively using 200 images. The size of each input image was 320×240. An 80×80 partial image of each input image was used as the template image.

Note that Table I shows that it is necessary to reduce the input-image size to some extent for practical use. Therefore, each input image was reduced to 80×60. ATM was then used to detect the location, width, and height of the partial image most similar to the template image. The degree d of the polynomial was changed from 1 to 15.

The template image was considered correctly matched if the errors in the row, column, width, and height were within two dots. The results are shown in Fig. 14. The figure shows that the matching accuracy increases as the value of d becomes large. The results of simple matching and VFTM are shown in the same figure. Since the input images were reduced, the accuracy of simple matching was not very high. When $d \geq 10$, the accuracy of ATM is better than that of simple matching.

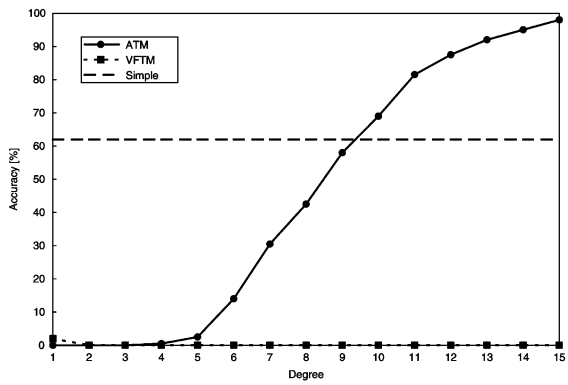

Fig. 14. Matching accuracy.

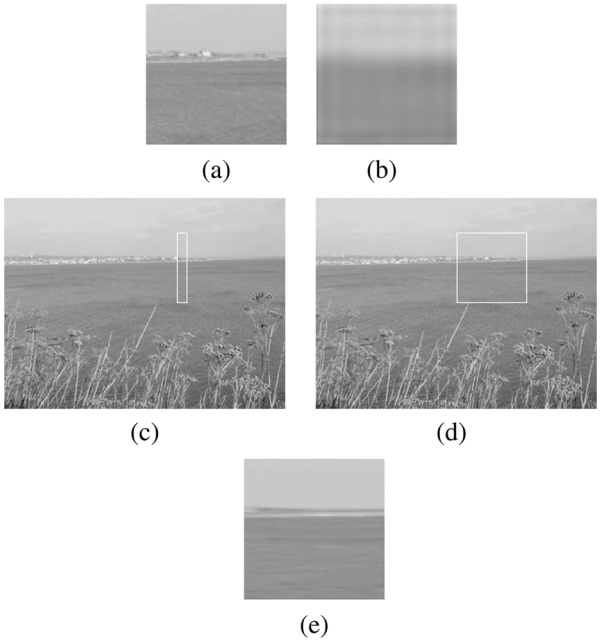

Fig. 15. Example of incorrect detection by the proposed method. (a) Template. (b) Image generated by the polynomial. (c) Incorrect detection result by the proposed method. (d) Correct detection result by simple matching. (e) Horizontally enlarged image of the region detected by the proposed method.

The matching accuracy for FFT is theoretically the same as that of simple matching. The matching accuracy for VFTM was low for any degree d. When the template image is exactly the same, or almost the same, as a part of the input image, VFTM can detect the part very quickly using a low-order polynomial. However, when the template image differs from the input image by only a little, the accuracy decreases rapidly. This is because the low-order polynomial cannot represent an image faithfully as shown in Fig. 4. Moreover, since the approximation process of VFTM is based on the least-square method, degradation of the quality of the approximation may occur as described in Section II-A. Therefore, it is difficult to use high-order polynomial in the framework of VFTM.

However, the proposed method may fail when an image has a high frequency content. Fig. 15 shows an example of incorrect matching by the proposed method when $d = 15$. Fig. 15(a) and (b) shows the original template image and the image generated by the polynomial. Fig. 15(c) and (d) shows the matching results for the proposed method and simple matching. In order to give an obvious image of the detected region by the proposed method, Fig. 15(e) shows a horizontally enlarged image of the detected region in Fig. 15(c). In this case, besides the detected region is similar to the template image, the template image has a high frequency content [white small buildings in Fig. 15(a)] that cannot be represented by the proposed method. Simple matching provides correct detection because these contents can be used as an important feature of the template image.

C. Application to Object Detection in Scene Images

ATM was applied for the object detection in the scene images to show the practical application of the proposed algorithm. Although the processing time of ATM is much smaller than that of the other template matching techniques, the processing time is still not small enough when the input image is large. For the construction of a practical object detection system, ATM was combined with the coarse-to-fine strategy, and the effectiveness of the combined approach was tested with the real scene images.

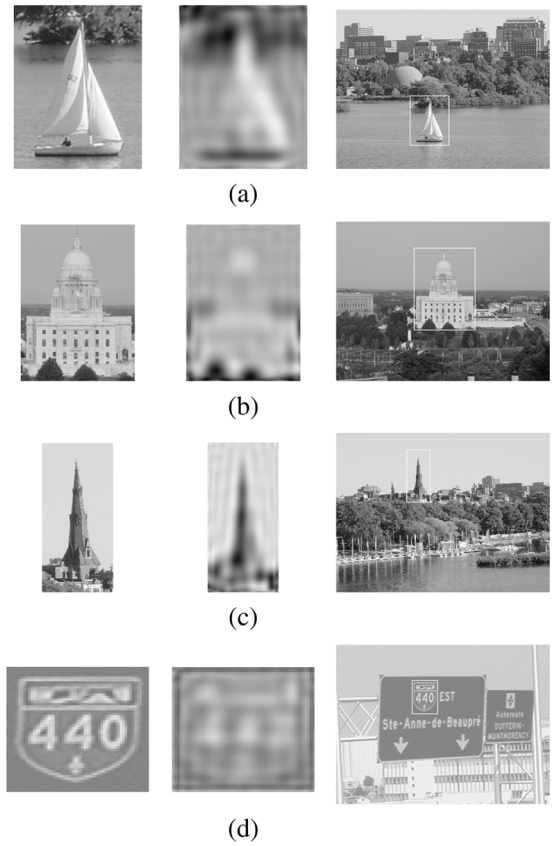

Fig. 16. Examples of object detection by the proposed method: (left) template images, (center) images generated by the polynomial, and (right) the detection results of the proposed method.

Given a template and an input image, first the input image was reduced to one-fourth of its original size by averaging the values of 4×4 pixels. The ATM performed coarse matching using the reduced image as the input image. Fine matching was performed using the original input image, only when there was a significant similarity in the coarse matching. The same polynomial that approximates the template image was used for both coarse and fine matching.

The experiment used 320×240 input images. A part of the input image that included an object was extracted manually, and the extracted or enlarged image was used as the template. Examples of the object detection are shown in Fig. 16. In each row, the left image is the template image, the center image is the image generated by the polynomial $(d = 20)$, and the right image is the detection result. For the images of Fig. 16(a)–(c), part of the input image is used as the template. For the image of Fig. 16(d), a horizontally enlarged portion of the input image is used as the template.

The object in the template image is detected correctly in all cases. Fig. 16(d) displays a template with an aspect ratio different from that of the partial image in the input image. This kind of template image, having a changed aspect ratio, is often obtained by taking a picture of a road sign from a moving vehicle. The figures show that the location, width, and height of the partial image most similar to the template image were correctly detected. The sizes of the template image and the detected partial image are 59×52 and 39×51, respectively.

V. Conclusion

In this paper, we have proposed a novel framework for template matching. The proposed algebraic template matching (ATM) uses a polynomial, which approximates a template image, to match an input image instead of using the template image. ATM efficiently calculates normalized cross correlations between the template and partial images of varying *widths and heights* by using recurrent formulae. The proposed algorithm is effective especially when the width and height of the template image are different from those of the input image. According to the authors, no other algorithm has been proposed to calculate similarities incrementally by changing the width and height of the partial image.

For efficient approximation of the template image by the polynomial, an algorithm using the Legendre polynomial was proposed. The computational cost of the proposed method is much less than that of the conventional least-square method. Moreover, the proposed algorithm only calculates the inner product and does not require the inverse matrix calculation thus reduces the effects of the truncation error.

It has been shown theoretically and experimentally that the computational cost of the proposed algorithm is $O(d^2MNM_1N_1)$, where the size of the input image is $M \times N$, the maximum size of the partial image is $M_1 \times N_1$, and the degree of the polynomial is d. The proposed algorithm resolves one of the shortcomings of the image-based approach where the computational cost is huge, and expands the usefulness of template matching.

In this paper, we focused only on image-based template matching. For practical object detection, feature-based approaches are also effective, e.g., using color histogram [19], [25] or detecting local feature points [11], [12]. It will be important, in future work, to combine the proposed method with the feature-based object detection algorithms to achieve a practical object detection system.

The proposed algorithm only treats expansion, contraction and translation of the partial image. To expand the ability of the proposed framework, other geometrical transformations such as rotation, affine transformation, and projective transformation must be considered. Developing efficient template matching algorithms for these kinds of transformations will be addressed in a future work.

Acknowledgment

The authors would like to thank Prof. H. Aso from Tohoku University and Dr. H. Sakano from NTT Data Corporation for their fruitful comments. They would also like to thank the anonymous reviewers for helpful comments.

References

[1] G. B. Arfken and H. J. Weber, *Mathematical Methods for Physicists*, 6th ed. New York: Academic, 2005.

[2] Y. Bentoutou, N. Taleb, K. Kpalma, and J. Ronsin, "An automatic image registration for applications in remote sensing," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 9, pp. 2127–2137, Sep. 2005.

[3] A. D. Bimbo and P. Pala, "Visual image retrieval by elastic matching of user sketches," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 121–132, Feb. 1997.

[4] J. Deutscher, "Nonlinear model simplification using L_2-optimal bilinearization," *Math. Comput. Model. Dyn. Syst.*, vol. 11, no. 1, pp. 1–19, 2005.

[5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[6] R. M. Dudley, *Real Analysis and Probability*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[7] R. M. Dufour, E. L. Miller, and N. P. Galatsanos, "Template matching based object recognition with unknown geometric parameters," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1385–1396, Dec. 2002.

[8] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Berlin, and C. Proy, "Real time correlation based stereo: algorithm, implementations and applications," INRIA Tech. Rep. RR-2013, 1993.

[9] M. Gharavi-Alkhansari, "A fast globally optimal algorithm for template matching using low-resolution pruning," *IEEE Trans. Image Process.*, vol. 10, no. 4, pp. 526–533, Apr. 2001.

[10] J. P. Lewis, "Fast template matching," in *Proc. Vision Interface*, 1995, pp. 120–123.

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

[12] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, 2004.

[13] H. Peng, F. Long, and Z. Chi, "Document image recognition based on template matching of component block projections," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1188–1192, Sep. 2003.

[14] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[15] A. Rosenfeld and G. J. VanderBrug, "Coarse-fine template matching," *IEEE Trans. Syst., Man. Cybern.*, vol. SMC-7, no. 2, pp. 104–107, Feb. 1977.

[16] H. Schweitzer, J. W. Bell, and F. Wu, "Very fast template matching," in *Proc. 7th Eur. Conf. Computer Vision IV*, 2002, pp. 358–372.

[17] L. D. Stefano, M. Marchionni, and S. Mattoccia, "A fast area-based stereo matching algorithm," *Image Vis. Comput.*, vol. 22, no. 12, pp. 983–1005, 2004.

[18] L. D. Stefano and S. Mattoccia, "Fast template matching using bounded partial correlation," *Mach. Vis. Appl.*, vol. 13, no. 4, pp. 213–221, 2003.

[19] M. Swain and D. Ballard, "Colour indexing," *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, 1991.

[20] S. L. Tanimoto, "Template matching in pyramids," *Comput. Graph. Image Process.*, vol. 16, no. 4, pp. 356–369, 1981.

[21] T. Tasdizen, J.-P. Tarel, and D. B. Cooper, "Improving the stability of algebraic curves for applications," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 405–416, Mar. 2000.

[22] G. Taubin, "Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 11, pp. 1115–1137, Nov. 1991.

[23] M. Uenohara and T. Kanade, "Use of Fourier and Karhunen-Loeve decomposition for fast pattern matching with a large set of templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 8, pp. 891–898, Aug. 1997.

[24] G. J. VanderBrug and A. Rosenfeld, "Two-stage template matching," *IEEE Trans. Comput.*, vol. C-26, no. 4, pp. 384–393, Apr. 1977.

[25] V. V. Vinod and H. Murase, "Focused color intersection with efficient searching for object extraction," *Pattern Recognit.*, vol. 30, no. 10, pp. 1787–1797, 1997.

Shinichiro Omachi (M'96) received the B.E., M.E., and Doctor of Engineering degrees in information engineering from Tohoku University, Japan, in 1988, 1990, and 1993, respectively.

He worked as a Research Associate in the Education Center for Information Processing, Tohoku University, from 1993 to 1996. He is now an Associate Professor at the Graduate School of Engineering, Tohoku University. He was a Visiting Associate Professor at Brown University, Providence, RI, from 2000 to 2001. His research interests include image processing, image retrieval, pattern recognition, and data mining.

Dr. Omachi is a member of the Institute of Electronics, Information, and Communication Engineers, the Information Processing Society of Japan, the Pattern Recognition Society, and the Japanese Society of Artificial Intelligence.

Masako Omachi received the B.E., Master of Information Sciences, and Doctor of Engineering degrees from Tohoku University, Japan, in 1994, 1996, and 1999, respectively.

She joined the Faculty of Science and Technology, Tohoku Bunka Gakuen University, Japan, in 1999, where she is now a Lecturer. Her research interest includes image processing, image retrieval, and image recognition.

Dr. Omachi is a member of the Institute of Electronics, Information, and Communication Engineers.