Scale-space Processing Using Polynomial Representations

Gou Koutaki
Kumamoto University
2-39-1 Kurokami Kumamoto, Japan
koutaki@cs.kumamoto-u.ac.jp

Keiichi Uchimura
Kumamoto University
2-39-1 Kurokami Kumamoto, Japan
uchimura@cs.kumamoto-u.ac.jp

Abstract

In this study, we propose the application of principal components analysis (PCA) to scale-spaces. PCA is a standard method used in computer vision. The translation of an input image into scale-space is a continuous operation, which requires the extension of conventional finite matrix-based PCA to an infinite number of dimensions. In this study, we use spectral decomposition to resolve this infinite eigenproblem by integration and we propose an approximate solution based on polynomial equations. To clarify its eigensolutions, we apply spectral decomposition to the Gaussian scale-space and scale-normalized Laplacian of Gaussian (LoG) space. As an application of this proposed method, we introduce a method for generating Gaussian blur images and scale-normalized LoG images, where we demonstrate that the accuracy of these images can be very high when calculating an arbitrary scale using a simple linear combination. We also propose a new Scale Invariant Feature Transform (SIFT) detector as a more practical example.

1. Introduction

Scale-space image processing is a basic technique used for object recognition and low-level feature extraction in computer vision [1][2][3][4][5]. Scale-space image processing generates a series of blurred images using a Gaussian filter with set scale parameters. The scale resolution improves as more images are generated, but increasing the number of images also increases the computational time. For example, the Scale Invariant Feature Transform (SIFT) [6], generates six Gaussian blurred images per octave.

Principal components analysis (PCA) is another method used for face recognition [7] and in other applications. PCA can compress multiple images (N-images) into a few component images, thus it may be beneficial to consider its use in the context of scale-space image processing. However, it is difficult to apply PCA to scale-spaces with a continuous scale parameter because they will comprise an infinite number of images. To overcome this problem, we propose the application spectral theory to solve continuous PCA. This allows the transformation of a matrix-based PCA problem into an integral equation-based problem, thereby reducing an infinite-dimensional processing problem to a finite-dimensional problem.

The main contributions of this study are as follows.

1. We propose and demonstrate a method for compressing scale-space images using continuous PCA (by spectral decomposition) to obtain numerical solutions.
2. We clarify the eigensolutions of the Gaussian scale-space and scale-normalized Laplacian of Gaussian (sLoG) space.

Our experimental results show that the proposed method can generate Gaussian blurred images and sLoG images at arbitrary scales with high accuracy. As a more practical example, we also introduce a new SIFT detector with high scale-resolution that uses spectral decomposition.

2. Related work

There have been many studies of scale-space filtering since the 1990s. Earlier research in the 1980s was based on discretization of the scale-space with scale and orientation dimensions, before Freedman and Adelson proposed a linear representation of a rotational scale-space filter [8][9][10]. Perona proposed the approximation of a scale-space filter with an orthonormal basis by using the singular value decomposition on Hilbert space [11]. A steerable-scalable filter was then proposed and used for edge detection [12], where the kernels were obtained by nonlinear least squares optimization [13]. The eigensolutions were discretized and solved using an iterative method based on the initially provided estimated solution. This method focused on multi-scale edge detection and orientation. In another approach, multi-scale images were approximated by polynomials[14][15][16].

In the 2000s, SIFT [17] was proposed and scale-space processing received greater attention. SIFT discretizes the scale-space using a coarse interval with a Difference of Gaussian (DoG) operator. SIFT was improved subsequently and many different types of detectors and descriptors have been proposed, such as Speeded Up Robust Features (SURF) [18], Affine Invariant SIFT (ASIFT) [19], and PCA-SIFT [20]. Recently, a new detector and descriptor were proposed on nonlinear scale-space [21].

However, the SIFT family use the traditional scale-space discretization of the 1980s. Thus, the present study refocuses the extensive work performed in the 1990s and scale-space filtering is applied to the SIFT. Furthermore, we propose a method for solving the eigensolutions of a scale-space filter by polynomial approximation. Our method does not require a particular initial estimate of the solution and it does not require iterative operations to obtain optimum solutions. We provide an analytical representation and closed form of scale-space filtering.

3. Scale space analysis

In this section, we analyze two types of scale-space: Gaussian scale-space and sLoG space.

3.1. Gaussian scale-space

For a given input image $f(x,y)$, its corresponding scale-space $r(x,y,s)$ image with scale parameter $s(s_1 \le s \le s_2)$ can be defined by convolution with a Gaussian kernel $g(x,y,s)$:

$$r(x',y',s) = \iint g(x,y,s) f(x-x',y-y') dxdy. \quad (1)$$

The two-dimensional (2D) Gaussian kernel $g(x,y,s)$ is defined by:

$$g(x,y,s) = \frac{1}{2\pi s^2} \exp\left(-\frac{x^2+y^2}{2s^2}\right), s_1 \le s \le s_2.$$

This can be expanded with a series of eigenfunctions $\varphi_i(s)$ using the scale parameter s:

$$g(x,y,s) = \sum_{i=0}^{\infty} \left(\int_{s_1}^{s_2} g(x,y,t) \varphi_i(t)\, dt \right) \varphi_i(s)$$

The series in the equation above can be approximated by truncating them to N terms:

$$g(x,y,s) \approx \sum_{i=0}^{N} \left(\int_{s_1}^{s_2} g(x,y,t) \varphi_i(t)\, dt \right) \varphi_i(s) \quad (2)$$

Substituting this into Eq.(1), we obtain:

$$\begin{aligned} r(x',y',s) \quad &\approx \quad \iint \sum_{i=0}^{N} \left(\int_{s_1}^{s_2} g(x,y,t) \varphi_i(t)\, dt \right) \varphi_i(s) \\ &\cdot f(x-x',y-y') dxdy \end{aligned}$$

Then, by changing the order of integration of $dxdy$ and dt, we obtain:

$$\begin{aligned} r(x',y',s) \quad &\approx \quad \sum_{i=0}^{N} \left\{ \iint \left(\int_{s_1}^{s_2} g(x,y,t) \varphi_i(t)\, dt \right) \right. \\ &\left. \cdot\ f(x-x',y-y') dxdy \right\} \varphi_i(s) \\ &= \quad \sum_{i=0}^{N} \varphi_i(s) \cdot \\ &\left\{ \iint F_i(x,y) f(x-x',y-y') dxdy \right\} \\ &\equiv \quad \sum_{i=0}^{N} \varphi_i(s)\, q_i(x',y'). \end{aligned} \quad (3)$$

where $F_i(x,y)$ is defined as:

$$F_i(x,y) = \int_{s_1}^{s_2} g(x,y,t) \varphi_i(t)\, dt. \quad (4)$$

In this case, $F_i(x,y)$, which can be considered as a 2D image, is called an *eigenimage*. Eq. (3) can be interpreted to represent a Gaussian blurred image of scale s, which is obtained by a linear combination of q_i and $\varphi_i(s)$, where q_i are obtained by convolving the input image f and N eigenimages $F_i(x,y)$.

To calculate the eigenfunctions, we can apply PCA to the Gaussian kernel. In the field of computer vision, PCA is generally understood to be a standard method for compressing data, which is used in processes such as the eigenface method or the subspace method. In the subspace method, for example, the eigenfunctions are obtained by solving the following $N \times N$ matrix eigenvalue problem:

$$\mathbf{C}\varphi = \lambda\varphi \quad (5)$$

The factor $\mathbf{C}$ above represents a covariance matrix defined by N images $g_1, g_2, ..., g_N$:

$$\mathbf{C} = \begin{bmatrix} \langle g_1, g_1 \rangle & \langle g_1, g_2 \rangle & \cdots & \langle g_1, g_N \rangle \\ \langle g_2, g_1 \rangle & \langle g_2, g_2 \rangle & \cdots & \langle g_2, g_N \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle g_N, g_1 \rangle & \langle g_N, g_2 \rangle & \cdots & \langle g_N, g_N \rangle \end{bmatrix} \quad (6)$$

where $\langle g_i, g_j \rangle$ is the inner product of g_i and g_j.

However, because the scale parameter s is continuous, it is difficult to apply this matrix-based PCA to scale-space compression. In the case where $N \to \infty$, it is necessary to expand the eigenproblem and this approach is known as spectral theory [22] in the functional analysis of mathematics. By applying spectral decomposition to Eq. (5), the matrix eigenproblem can be transformed into the following Fredholm integral equation:

$$\int_{s_1}^{s_2} K(t,s) \varphi(t)\, dt = \lambda \varphi(s), \quad (7)$$

where $K(t,s)$ is the integral kernel that is defined as:

$$\begin{aligned} K(t,s) &= \iint g(x,y,s)\, g(x,y,t)\, dxdy \\ &= \frac{1}{2\pi (s^2+t^2)}. \end{aligned} \tag{8}$$

If the integral kernel is non-zero, symmetric, and finite, Eq. (7) has a unique solution, but the integral equation is still difficult to solve exactly, except with a set of specific integral kernels. Therefore, we propose a solution by using a polynomial approximation:

$$\begin{aligned} \varphi_i(s) &= a_i^0 + s a_{i,1} + s^2 a_{i,2} + \cdots + s^N a_{i,N} \\ &= \left(1, s, s^2, \cdots, s^N\right) \cdot \boldsymbol{a_i}. \end{aligned} \tag{9}$$

By multiplying both sides of Eq. (7) by the polynomials $1, s, s^2, \cdots, s^N$ and then integrating, Eq. (7) is transformed into the following generalized eigenproblem of an $(N+1) \times (N+1)$ matrix:

$$\mathbf{K}\boldsymbol{a} = \lambda \mathbf{S}\boldsymbol{a}. \tag{10}$$

We define the elements of $\mathbf{K}, \mathbf{S}$ as:

$$K_{i+1j+1} = \frac{1}{2\pi} \iint \frac{s^j t^i}{s^2+t^2} dsdt, \tag{11}$$

$$S_{i+1j+1} = \int s^{i+j} ds = \frac{s^{1+i+j}}{1+i+j}. \tag{12}$$

The eigenfunctions $\varphi_i(s)$ in Eq. (9) can be obtained by solving for the $N+1$ eigenvalues λ_i and the eigenvector $\boldsymbol{a}_i$ in Eq. (10). To facilitate the orthonormalization of eigenfunctions, the following normalization is applied.

$$\boldsymbol{a} \leftarrow \frac{\boldsymbol{a}}{\left(\boldsymbol{a}^T S \boldsymbol{a}\right)^{1/2}} \tag{13}$$

To calculate the eigenimage F_i, the following equation can be obtained by substituting Eq.(4) into Eq.(9):

$$\begin{aligned} F_i(x,y) &= \int_{s1}^{s2} g(x,y,s)\varphi_i(s)\, ds \\ &= -\sum_{n=0}^{N} \frac{a_{i,n}}{2^{3/2}\pi r} \left(\frac{r}{2^{1/2}}\right)^n \Gamma\left(\frac{1-n}{2}, \frac{r^2}{2s_1^2}, \frac{r^2}{2s_2^2}\right) \end{aligned} \tag{14}$$

where $r = \sqrt{x^2+y^2}$ and Γ is a generalized incomplete gamma function that is defined as:

$$\Gamma(a, t_1, t_2) = \int_{t_1}^{t_2} t^{a-1} \exp(-t)\, dt \tag{15}$$

, which can be calculated accurately using a continued fraction expansion [23].

3.2. Scale-normalized LoG space

In the same manner as section 2.1, we present the eigensolutions of sLoG space. sLoG is used for scale invariant edge detection and SIFT, thus it is important in computer vision applications.

The sLoG space is defined by the following equation, which is a second-order differentiation, and the normalization constant s^2 for the Gaussian kernel.

$$r^s(x', y', s) = \iint s^2 \nabla^2 g(x,y,s) f(x-x', y-y') dxdy. \tag{16}$$

In this case, $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. Then, using the relationship of the diffusion equation,

$$s\nabla^2 g(x,y,s) = \frac{\partial}{\partial s} g(x,y,s),$$

Eq.(16) is transformed into the following equation.

$$r^s(x', y', s) = \iint s\frac{\partial g(x,y,s)}{\partial s} f(x-x', y-y') dxdy. \tag{17}$$

In the same manner as Eq.(1)$\sim$ Eq.(4), Eq.(17) can be expanded by eigenfunctions. The integral kernel of sLog (equivalent to Eq.(8)) is defined as:

$$\begin{aligned} K_s(s,t) &= \iint st \frac{\partial g(x,y,s)}{\partial s} \frac{\partial g(x,y,t)}{\partial t} dxdy \\ &= \frac{4s^2t^2}{\pi (s^2+t^2)^3}. \end{aligned} \tag{18}$$

To solve the integral equation above, we transform the integral equation into the matrix-based generalized eigenproblem by the polynomial approximation. Then, the elements of the matrix are obtained as:

$$K^s_{i+1j+1} = \frac{4}{\pi} \iint \frac{s^{j+2} t^{i+2}}{(s^2+t^2)^3} dsdt, \tag{19}$$

$$S^s_{i+1j+1} = \int s^{i+j} ds = \frac{s^{1+i+j}}{1+i+j}. \tag{20}$$

Thus, the eigenimage of sLoG F_i^s is defined as follows.

$$\begin{aligned} F_i^s(x,y) &= \int_{s1}^{s2} s\frac{\partial g(x,y,s)}{\partial s} \varphi_i^s(s)\, ds \\ &= -\sum_{n=0}^{N} \frac{a_{i,n}^s}{2^{1/2}\pi r} \left(\frac{r}{2^{1/2}}\right)^n \times \\ &\left[-\Gamma\left(\frac{1-n}{2}, \frac{r^2}{2s_1^2}, \frac{r^2}{2s_2^2}\right) + \Gamma\left(\frac{3-n}{2}, \frac{r^2}{2s_1^2}, \frac{r^2}{2s_2^2}\right)\right] \end{aligned} \tag{21}$$

In this case, $a_{i,n}^s$ is the coefficient of the polynomial of the eigensolution $\varphi_i^s(s)$, which is obtained by solving the generalized eigenproblem.

4. Numerical examples

In this section, we present numerical examples of eigen-solutions of Eq.(7) and demonstrate the linear generation of Gaussian and sLoG images.

4.1. Gaussian scale-space

To approximate the eigenfunction of Eq.(9), we use second- or third-order polynomials ($N = 2$ or $N = 3$) and set the integral range of the scale parameter s to $s_1 = 1.0, s_2 = 5.0$. We then solve the 3×3 or 4×4 matrix generalized eigenproblem of Eq.(10) The solutions $a_{i,j}$ and eigenvalues $\boldsymbol{a_i}, \lambda_i (0 \leq i \leq N)$ are shown in the tables at the bottom left of Figure 1.

The table shows that $\lambda_2 \approx 0.001$ is only $1[\%]$ of $\lambda_0 = 0.070$. This rapid decrease suggests that the original Gaussian function can be approximated using a low-order series expansion. The eigenimages for $N = 2$ are shown at the top left of Figure 1. The top part of the figure shows the eigenimages on the xy-plane and the middle part of the figure shows a graph of the eigenimages on $r = \sqrt{x^2 + y^2}$, which depend only on r, thus these eigenimages are isotropic functions. The lower part of the figure shows the eigenfunctions. The first-order eigenimage resembles a Gaussian and the second- and third-order eigenimages resemble a Laplacian, but there are slight differences.

4.2. sLoG space

The table at the bottom right of Figure 1 shows an example of the solution (coefficient of polynomial and eigenvalue) of sLoG for $N = 2$ and $N = 3, s_1 = 1.0, s_2 = 5.0$. The top right of Figure 1 shows the eigenimages and eigenfunctions of sLoG space.

4.3. Linear generation of Gaussian images

In this section, we introduce a method for Gaussian blur image generation with an arbitrary scale, as an application of scale-space compression.

A Gaussian blur image of scale s can be defined as:

$$r(x', y', s) = \sum_{i,j=0}^{N} q_i(x', y') s^j a_{i,j}. \tag{22}$$

In this case, $q_i \equiv f * F_i$. This equation can be interpreted as meaning that a scale s Gaussian blur image can be obtained by a linear combination of q_i and $a_{i,j}$. The factors q_i can be obtained by convolving the eigenimage F_i into an input image f.

Figure 2 shows a flowchart that illustrate the steps of image generation at scale $s = 1.2$. The blue window on the left shows the step where q_i is calculated, which indicates that a Gaussian blur image with an arbitrary scale s can be obtained immediately by linear combination after q_i has been calculated.

To evaluate the proposed method, we compared the blur images generated in the range $1 \leq s \leq 5$ with references that were generated by convolving the Gaussian kernel $g(x, y, s)$.

The top left of Figure 3 shows the images generated for the 128×128 Fruit image. The figure shows the references, the blur images generated by the proposed method for $N = 2$ and $N = 3$, and the difference images between the generated images and the references for $s = 1.2, 2.4, 3.6$, and 4.8. The figure shows that the results have few errors for $N = 2$ and $N = 3$.

The bottom left of Figure 3 shows the peak signal to noise ratio (PSNR) between the generated and reference images at scales ranging from $N = 1, 2$, and $3, s = 1.0$ to $s = 5.0$ for the images Lena and Fruit. The graph shows the PSNR with a scalable filter (five kernels used) [13]. The average PSNR error with our method was 68[dB] for $N = 3$, which shows that the proposed method can generate accurate Gaussian blur images using simple linear operations.

4.4. Linear generation of sLoG images

For $N = 3$, the sLoG images $r^s(x, y, s)$ can be obtained as follows:

$$r^s(x', y', s) = \sum_{i=0}^{3} q_i^s \left(a_{i,0}^s + s a_{i,1}^s + s^2 a_{i,2}^s + s^3 a_{i,3}^s \right). \tag{23}$$

In this case, $q_i^s \equiv f * F_i^s$. The top right of Figure 3 shows the sLoG images generated by the proposed method and conventional (scale-normalized) DoG images. The DoG image was obtained by follows:

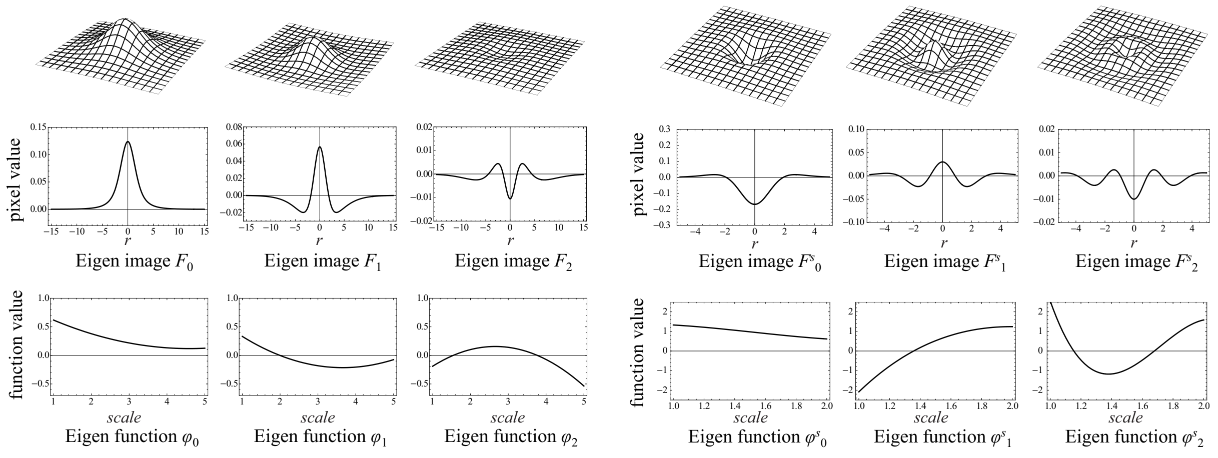
$$\mathrm{DoG}(x, y) \equiv \frac{g(x, y, k\sigma) - g(x, y, \sigma)}{1 - k}. \tag{24}$$

In this case, $k = 1.2$ was used. The pixel values are enhanced in the figures to make them visible. The two rows on the right of the figure show the difference between the reference $s\nabla^2 g * f$ (reference of the figure) and the generated image. The figure shows that the DoG image contains more errors because of the backward difference approximation Eq.(24). By contrast, the proposed method can approximate the sLoG images at various scales.

The bottom right of Figure 3 shows the numerical accuracy of the above approximation for the images Lena and Fruit. A scalable sLoG filter was implemented in an analogous manner to the scalable filter and five kernels were used. The average PSNR error with our method was 56[dB] for $N = 3$. The proposed method can approximate the sLoG accurately with a arbitrary scale using a linear combination of only four images, q_i^s.

5. Application: Spectral SIFT

We propose a new SIFT detector (spectral SIFT) that uses the sLoG space compression. The SIFT keypoints are



Coefficients Obtained for $N = 2$

i	$a_{i,0}$	$a_{i,1}$	$a_{i,2}$	λ_i
0	-1.51664	0.63295	-0.07352	0.07028
1	-1.98457	1.51593	-0.21841	0.01003
2	1.41248	-1.44794	0.29090	0.00077

Coefficients Obtained for $N = 3$

i	$a_{i,0}$	$a_{i,1}$	$a_{i,2}$	$a_{i,3}$	λ_i
0	-1.96331	1.47595	-0.40397	0.03729	0.07055
1	-2.52465	3.31488	-1.09824	0.11097	0.01088
2	2.20392	-3.77154	1.58800	-0.18386	0.00140
3	1.04991	-2.15040	1.14305	-0.16560	0.00008

(a) Eigen solutions of Gaussian space

Coefficients Obtained for $N = 2$

i	$a^s_{i,0}$	$a^s_{i,1}$	$a^s_{i,2}$	λ^s_i
0	-1.66680	0.66306	-0.07074	0.09065
1	-2.45391	1.77823	-0.25326	0.02621
2	1.86269	-1.70701	0.32655	0.00354

Coefficients Obtained for $N = 3$

i	$a^s_{i,0}$	$a^s_{i,1}$	$a^s_{i,2}$	$a^s_{i,3}$	λ^s_i
0	-1.78134	0.80365	-0.12157	0.00560	0.09067
1	-4.48103	4.32614	-1.19007	0.10394	0.02773
2	6.27885	-7.62290	2.65264	-0.27408	0.00624
3	4.07331	-5.69794	2.35606	-0.29145	0.00054

(b) Eigen solutions of sLoG space

Figure 1. Top left: Eigenimages and eigenfunctions of Gaussian scale-space. Bottom left: Numerical solutions obtained for Gaussian scale-space. Top right: Eigenimages and eigenfunctions of sLoG space. Bottom right: Numerical solutions obtained for sLoG space.

detected by finding the local extremum of sLoG. It is sufficient to find the zero position of the partial differential of sLoG. In the proposed model, the sLoG image is represented by the polynomial of s, thus it is easy to find the exact local extremum of sLoG by solving the following quadric equation.

$$\begin{aligned} &\partial r^s(x', y', s)/\partial s \\ &= \sum_{i=0}^{3} q_i^s \left(a_{i,1}^s + 2sa_{i,2}^s + 3s^2 a_{i,3}^s\right) \qquad (25) \\ &\equiv as^2 + bs + c = 0. \end{aligned}$$

Then, the optimal scales can be detected at $s = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$. The keypoint with $2as + b > 0$ is a bright keypoint and $2as + b < 0$ is a dark keypoint. After detecting the scale, 27 neighboring pixels of sLoG are checked to determine the XY-scale extremum.

Conventional SIFT requires that the 27 neighboring pixels are checked in all the scale layers (Figure 4(a)). This is a time-consuming step in SIFT, especially if the number of scale layers L increase. By contrast the proposed method can detect the optimal scale using a simple algebraic operation (Figure 4(b)). This approach is fast and accurate because it does not include discretization errors in the scale layer or interpolation artifacts.

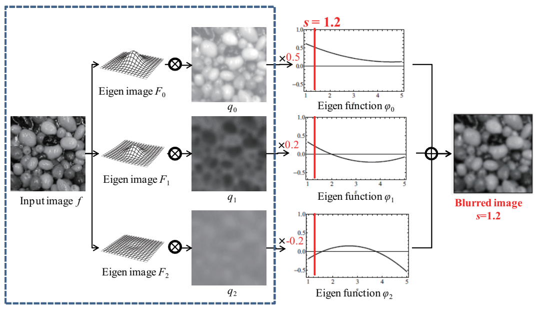


Figure 2. Flowchart illustrating Gaussian blurred image generation. A Gaussian blurred image with an arbitrary scale can be obtained by simple linear combinations of q_i.

5.1. Simple pattern testing

We evaluated our method using a simple test pattern, as shown in Figure 5. The 640 × 480 input image contained black circular patterns where the radius ranged from small

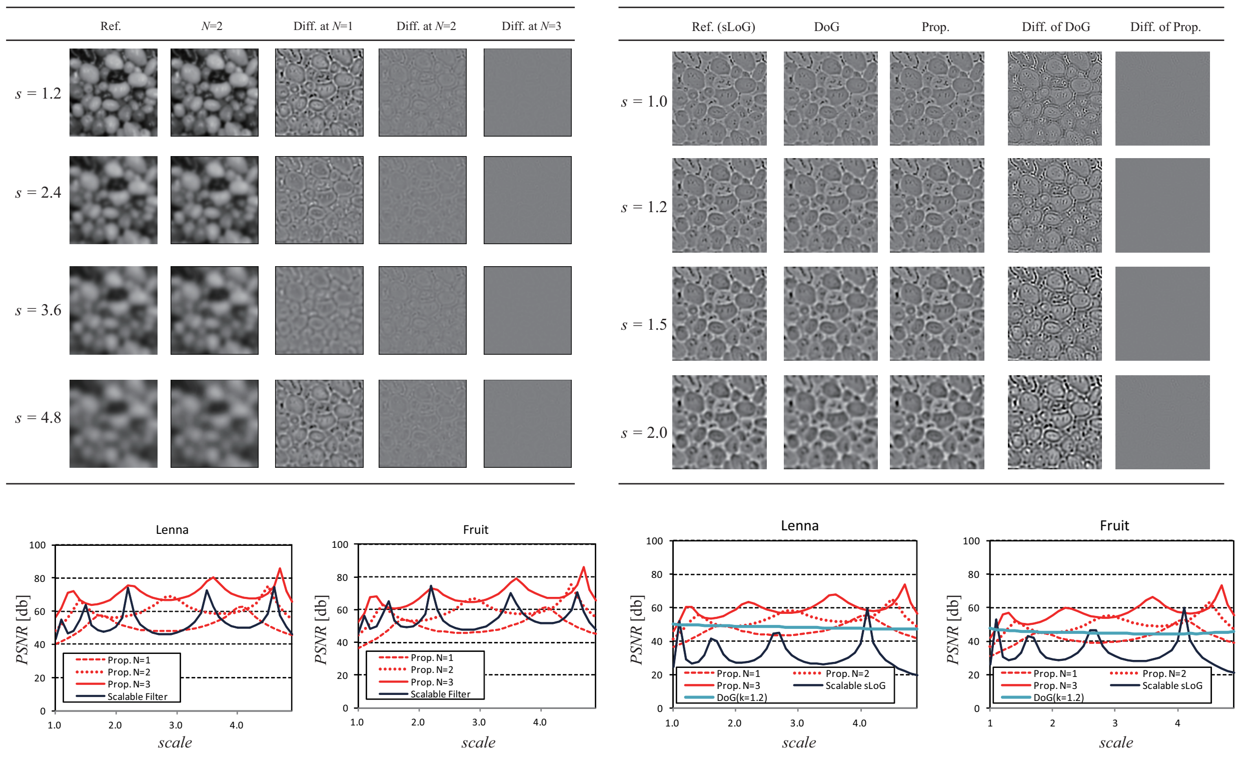


Figure 3. Top left: Gausian blurred images generated for various scales. Bottom left: PSNR evaluation of a Gausian blurred image. Top right: sLoG images generated for various scales. Bottom right: PSNR evaluation of a sLoG image.

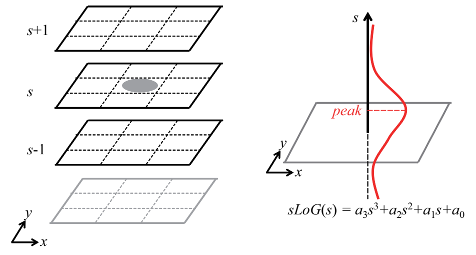
to large (about 2 $\sim$ 15[pix]). The figure shows the results obtained with three conventional SIFTs and the proposed method. We compared the results with different numbers of Gaussian images, i.e., $L = 6, 11$, and 16. In general, $L = 6$ is used for one octave. The same detection parameters were used, such as a threshold. The SIFT at $L = 6$ could not detect the small radius circles in the left of the image and it could not detect some of the large radius circles. By increasing the number of scale layers L, the scale resolution was improved and more of the small circles were detected. However, more of the large circles were missed due to scale discretization artifacts.

In traditional DoG approximation, the cascade Gaussian filtering approach is used to construct the scale-space [24]. This method generates a large-scale Gaussian image by repeated Gaussian filtering with small filter. This method is efficient but it leads to the propagation of errors.

By contrast, the proposed method can detect all of the circles correctly.

5.2. Evaluation of Detector Repeatablity

We evaluated the detector repeatability between two images (IM1 and IM2), which was defined by an ellipse in overlapping areas [25]. After detecting the keypoints in the



(a) 27 neighbor pixels extrema (b) Polynomial representation

Figure 4. Scale detection. In conventional SIFT, the scale-space is discretized. In our method, the images are represented by polynomials in the scale-space with the scale parameter s.

two images, IM2 was transformed and overlapped with IM1 using the given reference of the homography H and the keypoints in the non-overlapping region were removed. The overlapping regions between the two keypoints (a radius of $3s$ was used) were calculated. The overlap error was defined as $\epsilon = 1 - \frac{a \cap A^t bA}{a \cup A^t bA}$, where a and b are the regions defined by the ellipse parameters of the two keypoints, and A is the linearization of the homography H. The keypoint pairs where $\epsilon < 0.5$ were counted as correspondences. The

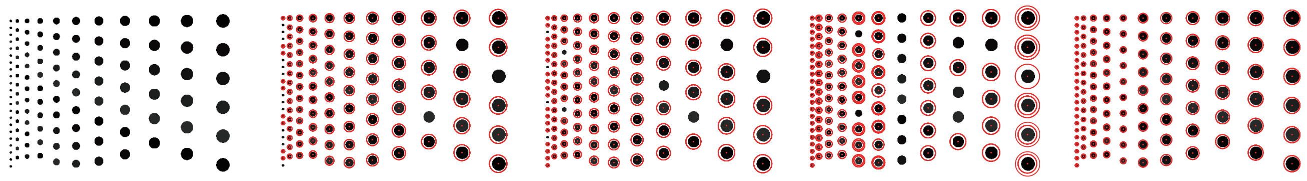


Figure 5. Results obtained with a simple test pattern. The red circles are the detected keypoints and the radius represents the scale. Conventional SIFT could not detect the circles correctly even in this simple case.

repeatability R is defined as follows:

$$R = \frac{\#correspondences}{\min(\#\text{keypoint of IM1}, \#\text{keypoint of IM2})}.$$

Figure 6 shows a comparison of the results using the Oxford dataset[1]. We compared the proposed method with the conventional SIFT using different numbers of scale layers L. The "Original" in the figure represent the results with the original Lowe's SIFT binary[2]. "OSL6," "OSL8," and "OSL11" are the results obtained using the Open SIFT Library[3] with scale layers $L = 6$, $L = 8$, and $L = 11$, respectively. To ensure that the conditions were the same, the keypoints were sorted by the Hessian response and the top 500 keypoints were used for matching. Our implementation[4] was based on the Open SIFT Library and the same detection parameters were used. The proposed method had a higher repeatability than conventional SIFT because our method could detect the keypoints at a small scale and there were few missing important keypoints, as shown in Figure 5.

5.3. Computational time

Table 1 (a) and (b) show the computational time comparisons for the **wall 1** and **boat 1** datasets. The comparisons were performed with a CPU with an Intel Core i7-4770 at 3.4 GHz and our code was implemented in C++. The detection step is time-consuming in SIFT because conventional SIFT searches 27 neighboring pixels for all the scale layers, which increases the time in proportion to the number of scale layers L. By contrast, the proposed method reduces the detection time costs because it does not search the 27 neighbors.

6. Conclusions

In this study, we proposed a method for applying PCA to scale-spaces. PCA is the standard method used for tasks in computer vision applications. However, to apply the PCA method to scale-spaces, it is necessary to extend conventional square matrix-based finite PCA to an infinite number of dimensions. To resolve this infinite eigenproblem, we used spectral decomposition to develop integral equations where approximate solutions could be developed using polynomial equations.

[1]http://www.robots.ox.ac.uk/ vgg/data/data-aff.html
[2]http://www.cs.ubc.ca/ lowe/keypoints/
[3]http://blogs.oregonstate.edu/hess/code/sift/
[4]http://navi.cs.kumamoto-u.ac.jp/ koutaki/

	Conventional SIFT			Proposed
	$L = 6$	$L = 8$	$L = 11$	$N = 3$
Filtering time	28	35	44	33
Detection time	26	42	73	32
Total time	54	77	117	65
#keypoints	2167	3049	3748	4776
Total time / #keypoints	0.025	0.025	0.031	0.013

(a) **wall 1** (1000×700)

	Conventional SIFT			Proposed
	$L = 6$	$L = 8$	$L = 11$	$N = 3$
Filtering time	24	28	34	42
Detection time	23	33	59	24
Total time	47	61	93	66
#keypoints	1731	2315	2803	4136
Total time / #keypoints	0.027	0.026	0.033	0.016

(b) **boat 1** (850×680)

Table 1. Computational time [ms]

As an application of this proposed method, we developed a method for generating Gaussian blur images and sLoG images with an arbitrary scale, which can be calculated by simple linear combination. As a practical example, we proposed a spectral SIFT detector that uses spectral decomposition.

This scale-space processing is a basic technique, thus our method can be applied to many existing scale-space processing problems. In the future, we plan to apply spectral decomposition to various scale-spaces including a scale-space with multiple parameters such as a Gabor scale-space or affine Gaussian scale-space [26] [27].

References

[1] Andrew P. Witkin. Scale-space filtering. In *IJCAI*, pages 1019–1022, 1983. 1

[2] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *IJCV*, 30(2):117–156, 1998. 1

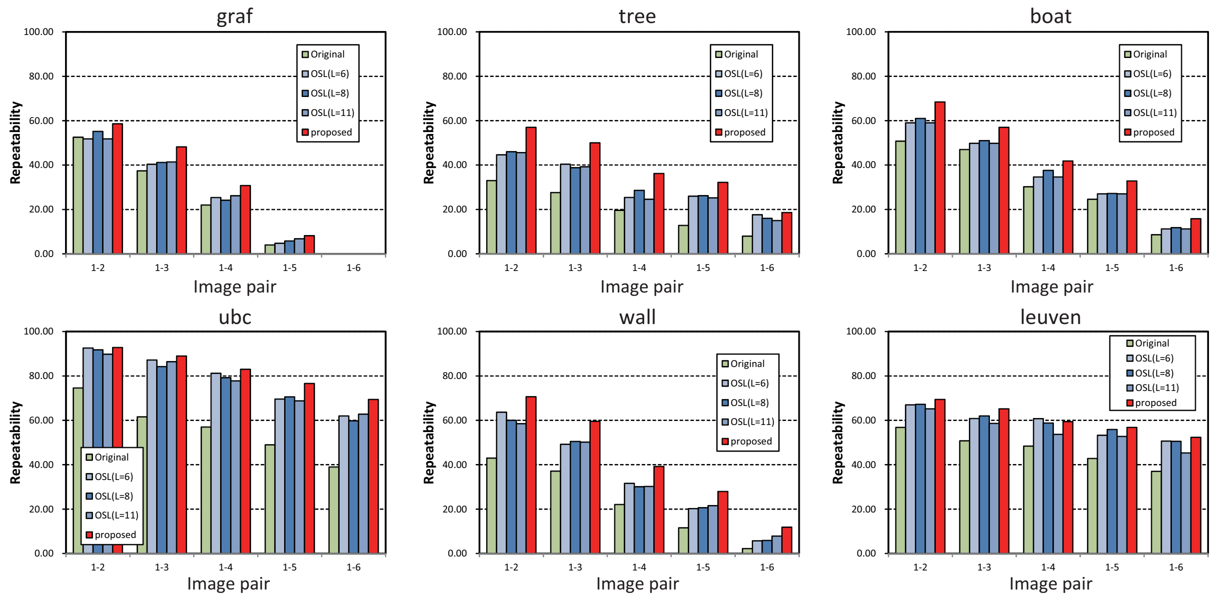


Figure 6. Detector repeatability with six scenes from the Oxford dataset.

[3] A L Yuille and T A Poggio. Scaling theorems for zero crossings. *TPAMI*, 8:15–25, 1986. 1

[4] J Babaud, A P Witkin, M Baudin, and R O Duda. Uniqueness of the gaussian kernel for scale-space filtering. *TPAMI*, 8:26–33, 1986. 1

[5] Surendra Ranganath. Image filtering using multiresolution representations. *TPAMI*, 13(5):426–440, 1991. 1

[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1

[7] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, 1991. 1

[8] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger. Shiftable multi-scale transforms. Technical Report 161, 1991. 1

[9] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *TPAMI*, 13(9):891–906, 1991. 1

[10] Anil A. Bharath. Steerable filters from erlang functions. In *BMVC*, pages 144–153, 1998. 1

[11] P. Perona. Deformable kernels for early vision. *CVPR*, pages 222–227, 1991. 1

[12] P. Perona. Steerable-scalable kernels for edge detection and junction analysis. In *ECCV*, pages 3–18, 1992. 1

[13] D. Shy and P. Perona. X-y separable pyramid steerable scalable filters. In *CVPR*, pages 237–244, 1994. 1, 4

[14] R. van den Boomgaard and J. van De Weijer. Least squares and robust estimation of local image structure. In *Proceedings of Scale-Space*, pages 237–254, Berlin Heidelberg, 2003. 1

[15] L. Florack, B. M. ter Haar Romeny, M. Viergever, and J.Koenderink. The gaussian scale-space paradigm and the multiscale local jet. *IJCV*, 18:61–75, 1996. 1

[16] S. Omachi and M. Omachi. Fast template matching with polynomials. *TIP*, 16:2139–2149, 2007. 1

[17] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–, 1999. 1

[18] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *CVIU*, 110(3):346–359, 2008. 2

[19] Jean-Michel Morel and Guoshen Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci.*, 2(2):438–469, April 2009. 2

[20] Yan Ke and Rahul Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *CVPR*, pages 506–513, 2004. 2

[21] Pablo Fernandez Alcantarilla, Adrien Bartoli, and Andrew J. Davison. Kaze features. In *ECCV*, pages 214–227, 2012. 2

[22] Shigeru Mizohata. *Introduction to integral equations*. Asakura Press, 1968. 2

[23] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition*. 3 edition, 2007. 3

[24] W. M. Wells. Efficient synthesis of gaussian filters by cascaded uniform filters. *TPAMI*, pages 234–239, 1986. 6

[25] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *TPAMI*, 27(10):1615–1630, 2005. 6

[26] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *TPAMI*, 20(11):1254–1259. 7

[27] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, pages 128–142, 2002. 7