Multimed Tools Appl (2017) 76:8471–8496
DOI 10.1007/s11042-016-3479-y

Boundary image matching supporting partial denoising using time-series matching techniques

Bum-Soo Kim[1] · Yang-Sae Moon[2] · Jae-Gil Lee[1]

Received: 17 September 2015 / Revised: 1 February 2016 / Accepted: 18 March 2016 /
Published online: 5 April 2016

Abstract In this paper, we deal with the problem of boundary image matching which finds similar boundary images regardless of partial noise exploiting time-series matching techniques. Time-seris matching techniques make it easier to compute distances for similarity identification, and therefore it is feasible to perform boundary image matching even on a large image database. To solve this problem, we first convert all boundary images into times-series and derive *partial denoising time-series*. The partial denoising time-series is generated from an original time-series by removing partial noise; that is, it is obtained by changing a position of *partial denoising* from original time-series. We then introduce the *partial denoising distance*, which is the minimum distance from a *query* time-series to all possible partial denoising time-series generated from a *data* time-series, and propose *partial denoising boundary image matching* using the partial denoising distance as a similarity measure. Computing the partial denoising distance, however, incurs a severe computational overhead since there are a large number of partial denoising time-series to be considered. Thus, in order to improve its performance, we present a tight lower bound of the partial denoising distance and also optimize the computation of the partial denoising distance. We finally propose range and k-NN query algorithms according to a query processing method for partial denoising boundary image matching. Through extensive experiments, we show

The preliminary version of this paper was published in *Proc. of the Int'l Conf. on Big Data and Smart Computing* (*BigComp 2015*), Jeju, Korea, pp. 136–141, Feb. 2015.

✉ Jae-Gil Lee
jaegil@kaist.ac.kr

Bum-Soo Kim
bumsoo.kim@kaist.ac.kr

Yang-Sae Moon
ysmoon@kangwon.ac.kr

[1] Department of Knowledge Service Engineering, KAIST, Daejeon, Korea

[2] Department of Computer Science, Kangwon National University, Chuncheon, Korea

that our lower bound-based approach and the optimization method of the partial denoising distance improve search performance by up to an order of magnitude.

Keywords Time-series databases · Data mining · Boundary image matching · Time-series matching · Moving average transform · Partial denoising

1 Introduction

There have been a number of efforts to utilize the large number of time-series data, and accordingly, time-series matching has become an important research topic in data mining [1, 6, 7, 9, 18]. In addition, there have been several recent attempts to apply these time-series matching techniques to practical applications such as high dimensional indexing [18], image matching [11, 17], and biological sequence matching [13]. Among these applications, we focus on boundary image matching for a large image database. *Boundary image matching* is a problem of finding the boundary images similar to a given boundary image. In this paper, we deal with boundary image matching considering the *partial noise*, which is a limited amount of noise embedded in a boundary image. In real applications, there are many examples of the partial noise. Figure 1 shows various examples of boundary images containing the partial noise. We note that the partial noise is regarded as not only white noise but also distortion of boundary. In the case of these examples, the matching results may be misrepresented if we perform boundary image matching without considering the partial noise.

For the matching on a large image database, in particular, we remove the partial noise in the time-series domain instead of the image domain [11, 17]. In this paper, we use the moving average transform [16] to remove the partial noise in the time-series domain. In more detail, we apply it to the *subsequence* of the time-series for partial denoising while

(a) A boundary image of a torn leaf.

(b) A boundary image of a rose with lens flare.

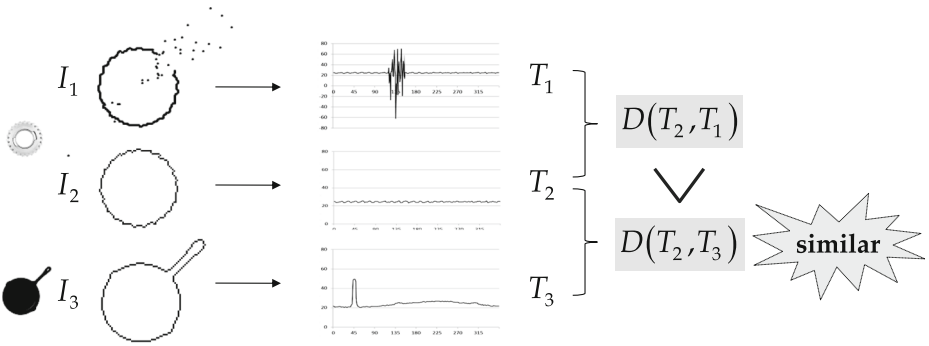
(c) A boundary image of a hand drum with a mallet.

Fig. 1 Various examples of boundary images containing the partial noise

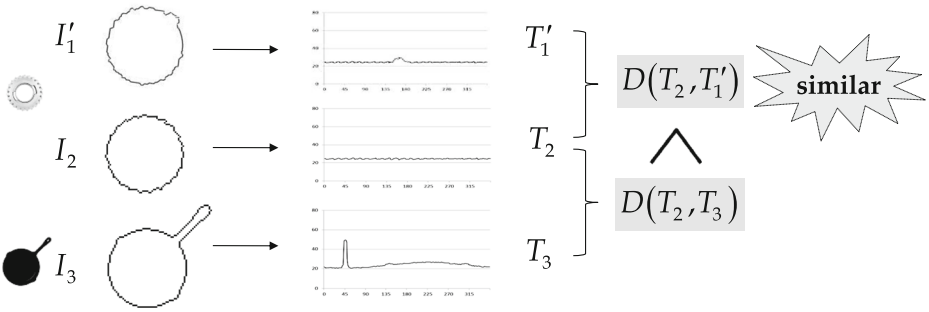
the previous work [16] applies it to the *whole* time-series. We first convert boundary images to time-series and then perform boundary image matching by removing the partial noise from these time-series in the time-series domain. We then call this matching *partial denoising boundary image matching* or, simply, *partial denoising boundary matching*. Figure 2 shows our motivating example that explains in more detail why partial denoising boundary matching is necessary.

Motivating example In Fig. 2a, images I_1 and I_2 are originally the same, but the image I_1 contains the partial noise in the top right corner of the boundary. On the other hand, the image I_3 is originally different from images I_1 and I_2. For boundary image matching, as shown in Fig. 2a, the three boundary images I_1, I_2, and I_3 are converted to their corresponding time-series, T_1, T_2, and T_3, respectively. We then compute the Euclidean distances, $D(T_2, T_1)$ and $D(T_2, T_3)$. Based on these distances, we identify I_3, not I_1, as the similar image of I_2 since $D(T_2, T_3) < D(T_2, T_1)$, while in Fig. 2b we can get the more intuitive result by partial denoising. In Fig. 2b, we first perform partial denoising on I_1 and get the denoising image I_1' and its time-series T_1'. We then identify I_1' rather than I_3 as the similar image of I_2 since $D(T_2, T_1') < D(T_2, T_3)$.

As shown in the motivating example, partial denoising boundary matching can provide more intuitive matching results than the previous boundary image matching without partial denoising.



(a) Comparing time-series of *original* boundary images.



(b) Comparing time-series of *partial denoising* boundary images.

Fig. 2 A motivating example of partial denoising boundary matching

Performing partial denoising boundary matching is not trivial since the partial noise varies as a level, a position, and a length; that is, we have to consider all possible partial noises for partial denoising boundary matching. Figure 3 shows an example of various partial noises represented by changing a level, a position, and a length. Figure 3a, b, and c represent examples of various partial noises by changing a level, a position, and a length, respectively. To consider partial denoising in boundary image matching, we first define the *partial denoising time-series*, which is a time-series obtained by removing the subsequence of the partial noise from an original time-series according to the given level, position, and length. Thus, by definition, a lot of the partial denoising time-series are generated as all possible levels, positions, and lengths. Thus, we need an efficient solution for partial denoising boundary matching since there are many partial denoising time-series compared with a query time-series.

We define the *partial denoising distance* for comparing many partial denoising time-series and propose a similarity measure based on this distance. In order to simplify the problem, we assume that the level and the length for partial denoising are given by a user, and we present an *interactive* approach to get the pseudo-optimal result by querying several times [11]; that is, a user can get his/her own best matching result by changing the amount and length of denoising. Then, the partial denoising distance is defined as the minimum distance from the query time-series to the partial denoising time-series generated by all possible positions of partial denoising; that is, if the partial noise is removed from data time-series by using the level and the length, the partial denoising distance is the minimum value among the distances from the query time-series to all partial denoising time-series considering the position of the partial noise. We then formally define partial denoising boundary matching and propose the range and k-NN query algorithms of partial denoising boundary matching. The partial denoising distance, however, requires a high computational complexity since the partial denoising time-series are generated by all possible positions of partial denoising even if the level and the length of partial denoising are given by a user. To address this high complexity, we propose a method of computing the lower bound instead of the partial denoising distance and also optimize the computation of the partial denoising distance.

Through extensive experiments, we show that the proposed method provides more intuitive and exact matching results than the boundary image matching without supporting the partial denoising. We also confirm that the matching algorithms using the lower bound and the optimized partial denoising distance outperform the naive matching algorithms only

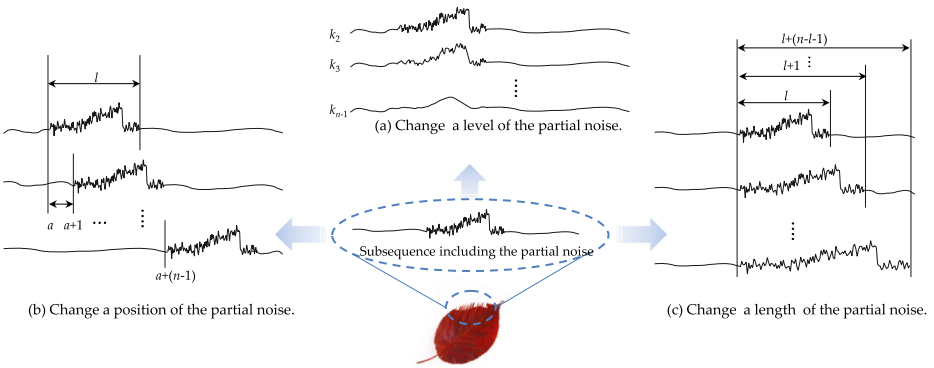


Fig. 3 Examples of various partial noises represented by changing a level, a position, and a length

using the partial denoising distance. According to these results, we believe that our method is the superior approach solving partial denoising in boundary image matching.

The rest of this paper is organized as follows. Section 2 explains background and related work on time-series matching and image matching. Section 3 presents the concept of partial denoising boundary matching and its solution. Section 4 explains experimental results on partial denoising boundary matching. We finally summarize and conclude the paper in Section 5.

2 Related work

2.1 Time-series matching

A *time-series* is a sequence of real numbers representing values at specific time points. It has been found in various types of data such as stock prices, medical data, and temperature data. Finding similar time-series compared with a given time-series in a time-series database is called *time-series matching* [1, 6, 15, 18]. In time-series matching, there have been many research efforts on similarity model. In this paper, we use the Euclidean distance-based similarity model [1, 6, 17]. Given two time-series X and Y of the same length n, the Euclidean distance $D(X, Y)$ is defined as the following (1).

$$D(X, Y) \equiv \sqrt{\sum_{i=0}^{n-1}(x_i - y_i)^2} \tag{1}$$

The Euclidean distance-based similarity model is applied to a range or a k-NN query search using $D(X, Y)$, which is the distance between two time-series. Besides the Euclidean distance-based model, there are several similarity models such as DTW(dynamic time warping) [9] and LCSS(longest common subspaces) [24]. In addition, similarity models supporting preprocessing transformations, such as linear detrending [7, 23], shifting and scaling [3, 21], normalization [14, 16], moving average transform [16, 21], were proposed for time-series matching. These studies focused on preprocessing of time-series, but did not handle image domain problems unlike this paper. These similarity models and transformation techniques are orthogonal to our approach; that is, the problem of removing the partial noise from time-series data is orthogonal to the problem of identifying the similar time-series even whether those time-series are partially (or totally) denoised or not. It means that the partial denoising technique used in the paper is orthogonal to the similarity measure. Thus, we exploit time-series matching using the moving average transform and the Euclidean distance for boundary image matching.[1]

[1] According to the report of [5], the accuracy differences among different similarity measures are not significant. Thus, even though we use other similarity measures including the dynamic time warping distance, the matching result will be very similar with the case of using the Euclidean distance in this paper. Also, in this paper we focus on the performance improvement, i.e., the reduction of execution times, rather than the accuracy improvement of removing the partial noise. To confirm the performance improvement, we adopt and optimize the Euclidean distance, which is simple and one of the most widely used distance measures. For the other similarity measures, we need to develop different optimization techniques, which are out of scope of this paper.

2.2 Image matching

Image matching is the problem of finding data images similar to a given query image using features of an image. It is one of the most important research topics in image processing [20]. In image matching there have been many research attempts to use various features of images. For example, colors [12], textures [4], and shapes [26] were used as major features for segmentation techniques in image matching. Image matching can be used together with different features since these features are orthogonal to each other, and in this paper we focus on the image matching based on shape features. In general, this shape-based image matching is useful when the images contain boundary objects, and their color features or texture features have similar values [22].

2.2.1 Shape-based image matching

The shape-based image matching is classified according to shape features. In this paper, we use object boundaries as a shape feature and exploit the centroid contour distance (*CCD*) [8, 11, 17], which is the simplest method that uses the boundary feature of an image. CCD maps boundary features to a time-series of length n (or n-dimensional) as follows: it first evenly divides 360° into n angles of the same size ($\Delta\theta = 2\pi/n$), where the direction is from the centroid to the boundary; it then obtains n boundary features; and it finally computes the distance of each boundary feature from the centroid. Figure 4 shows an example of converting a boundary image to a point of the n-dimensional space, i.e., a time-series of length 360, by CCD. Likewise, we can map boundary images to time-series and exploit time-series matching techniques for boundary image matching by using CCD [11, 17, 25]. Hereafter, we use "boundary image" and "boundary time-series" interchangeably unless confusion occurs.

Recently, a few studies that are using boundary time-series in boundary image matching were reported [11, 17, 25]. Vlachos et al. [25] first presented the rotation-invariant property of DFT magnitudes and proposed a novel solution to rotation-invariant image matching by indexing the boundary time-series. Moon et al. [17] dealt with the scaling-invariant problem of considering all possible scaling factors for boundary image matching. To solve the scaling-invariant problem, they first converted scaling of a boundary image into scaling of a time-series by using interpolation and proposed the notion of scaling-invariant distance

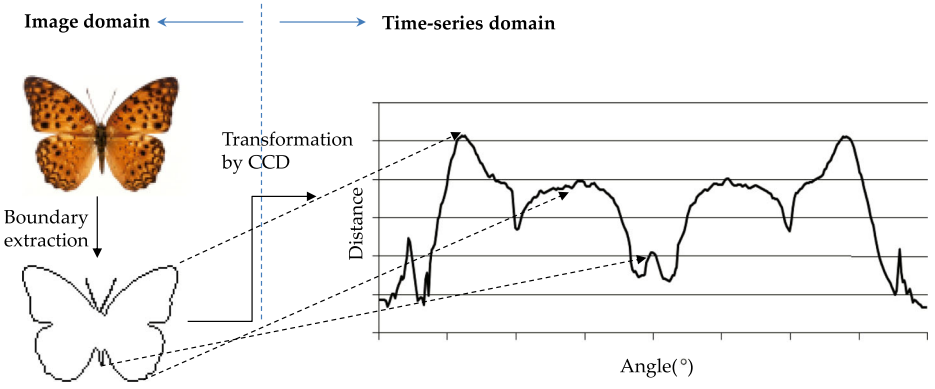


Fig. 4 An example of converting an image to a time-series by CCD

between the scaled boundary time-series. These studies solved the rotation-invariant and the scaling-invariant problems in the time-series domain, but they did not consider the partial denoising problem to be solved in this paper.

There have been many research efforts on shape-based image matching using the shape context in the image domain [2, 10, 19]. The shape context is represented as the histogram between the distance and the angle from a selected point to all other points on the contours of a shape [2]. This feature is invariant to image scaling, translation, and rotation [10]. Thus, the previous studies deal with various matching problems using the shape context [2, 10, 19]. However, the previous studies do not consider partial denoising. In the paper, we propose a fast solution which performs interactive boundary image matching supporting partial denoising in the time-series domain, and compare with shape-based image matching using the shape context and the proposed method in Section 4.2.

2.2.2 Image denoising

To remove noise in the image domain, image matching uses the spatial filtering method and the frequency filtering method [8]. These filtering methods can quickly and exactly remove noise. However, the shape-based image matching considering the partial noise incurs heavy computation overhead because of finding and removing the partial noise in a boundary image [2]. Thus, in this paper we present the partial denoising problem, and then propose the efficient solution for partial denoising boundary matching by converting the image domain to the time-series domain.

The recent work [11] in the time-series domain presents the solution for removing noise on boundary image matching. This work removes noise by applying the moving average transform to the whole length of the boundary time-series, and it is called *whole denoising*. It also focuses on supporting the moving average transform of arbitrary order. Meanwhile, this paper deals with partial denoising instead of whole denoising on time-series matching. In particular, this paper focuses on solving the problem of partial denoising that is anywhere in boundary time-series. Furthermore, extending partial denoising to whole denoising includes the result of the previous work [11]. Thus, we can say that our method is more general than the previous method.

3 Partial denoising boundary image matching

3.1 Problem definition and naive solution

In this paper, we deal with the partial denoising boundary matching problem that considers all partial noises of data (boundary) images stored in an image database. In the case of a query (boundary) image, partial denoising is simple since it is performed once by preprocessing. On the other hand, in that of data images, it is a challenge problem because of considering all possible partial noises from data images. Thus, we focus on partial denoising in data images.

To perform partial denoising for data images, we can consider two different approaches in the image domain and the time-series domain, respectively. In the image domain, we first obtain the boundary image by applying the "appropriate" partial denoising from the original image itself, i.e., partial denoising is done in the image domain, and it is then converted to the corresponding time-series. However, if we perform partial denoising boundary matching on a large image database, this approach may cause a heavy computational overhead

due to time-consuming operations of image processing. In addition, the "appropriate" partial denoising to each data image is difficult. The various factors for partial denoising are dynamically and frequently changed in our proposed solution, so this complicated image domain approach cannot be used as a practical solution. On the other hand, to avoid the complicated partial denoising in the image domain, we obtain the partial denoising time-series in the time-series domain; that is, we first get the boundary time-series from an original boundary image and then can directly obtain the partial denoising time-series after removing the partial noise from the boundary time-series itself. This approach can easily remove various partial noises and quickly compute the distances. Thus, in this paper, we use the time-series domain approach for partial denoising in boundary image matching.

To exploit the time-series domain approach in computing the distance between the query and the data images, we formally define the notion of the partial denoising time-series. We first assume that we find the partial noise from the boundary time-series and then define the denoising subsequence of the boundary time-series as follows.

Definition 1 Let a boundary time-series X $(= \{x_0, \ldots, x_{n-1}\})$ of the length n be converted from a boundary image, and its subsequence $X[i : (i+l-1)\%n]$ of the length l be removed by using the moving average transform of the moving average order d.[2] After which, the length l, the moving average order d, and the starting position i are called *denoising length*, *denoising level*, and *denoising position*, respectively. The subsequence $X_i^{d,l}$ without noise is also called *denoising subsequence*, and then $X_i^{d,l}$ is defined as (2):

$$X_i^{d,l} = \left\{ x_{i\%n}^{d,l}, x_{i+1\%n}^{d,l}, \ldots, x_{(i+l-1)\%n}^{d,l} \right\}, \tag{2}$$

where $x_j^{d,l} = \frac{1}{d} \sum_{k=j}^{j+d-1} x_{k\%n},\ 0 \le i \le n-1,\ i \le j \le (i+l-1),\ 1 < d \le n-1,$

and where '%' is a modular operator.

Therefore, using Definition 1 we formally define the boundary time-series including the denoising subsequence as follows.

Definition 2 Given a boundary time-series X of length n, the denoising level d, the denoising length l, and the denoising position i, the time-series $\widetilde{X}_i^{d,l}$, called *partial denoising time-series*, is replaced by the denoising subsequence $X_i^{d,l}$ instead of its subsequence $X[i : i+l-1]$ and then is defined as (3):

$$\widetilde{X}_i^{d,l} = \left\{ \widetilde{x}_{i,0}^{d,l}, \widetilde{x}_{i,1}^{d,l} \ldots, \widetilde{x}_{i,n-1}^{d,l} \right\}, \tag{3}$$

where $0 \le i \le n-1$ and $\widetilde{x}_{i,j}^{d,l} = \begin{cases} x_j^{d,l} & \text{if } j \in \{i\%n, (i+1)\%n, \ldots, (i+l-1)\%n\}; \\ x_j & \text{otherwise.} \end{cases}$

[2]The entry values used by the moving average transform may be included in the other values besides the values corresponding to the partial noise; that is, the moving average transform may interchangeably use not only the partial noise values but also the other values of the boundary time-series for removing noise. Although the partial denoising is affected by the entry values, we especially do not consider the effect since it is insignificant.

Figure 5 shows examples of the denoising subsequence and the partial denoising time-series. Various partial denoising time-series are generated by changing the denoising level d, the denoising length l, and the denoising position i. For example, let length of a boundary time-series X, the denoising level, and the denoising length be 8, 4, and 4, respectively. Given denoising subsequences $X_1^{4,4}$ and $X_5^{4,4}$, where their denoising positions are 1 and 5, their partial denoising time-series are computed as $\widetilde{X}_1^{4,4} = \left\{x_0, x_1^{4,4}, x_2^{4,4} x_3^{4,4}, x_4^{4,4}, x_5, x_6, x_7\right\}$ and $\widetilde{X}_5^{4,4} = \left\{x_0^{4,4}, x_1, x_2, x_3, x_4, x_5^{4,4}, x_6^{4,4}, x_7^{4,4}\right\}$, respectively.

In this paper we propose the similarity measure, which is the minimum distance from a query time-series to all possible partial denoising time-series; that is, boundary image matching is the problem of finding partial denoising time-series similar to a query time-series. In order to simplify this problem, we assume that the denoising level d and the denoising length l are given by a user, and this minimum distance is formally defined as the following Definition 3.

Definition 3 Let X and Y be two boundary time-series, and d and l be the denoising level and the denoising length, respectively. The distance $PDD(X, Y, d, l)$ of X and Y, called *partial denoising distance*, is defined as the minimum distance from X to all possible partial denoising time-series of Y; that is, $PDD(X, Y, d, l)$ is computed as (4):

$$PDD(X, Y, d, l) = \min_{i=0}^{n-1} D\left(X, \widetilde{Y}_i^{d,l}\right) = \min_{i=0}^{n-1} \sqrt{\sum_{j=0}^{n-1} \left|x_j - \widetilde{y}_{i,j}^{d,l}\right|^2}. \tag{4}$$

Here, $D(X, Y)$ is the Euclidean distance between X and Y; i.e., $D(X, Y) = \sqrt{\sum_{j=0}^{n-1} |x_j - y_j|^2}$.

Using the notion of the partial denoising distance, we now formally redefine the problem of partial denoising boundary matching as follows.

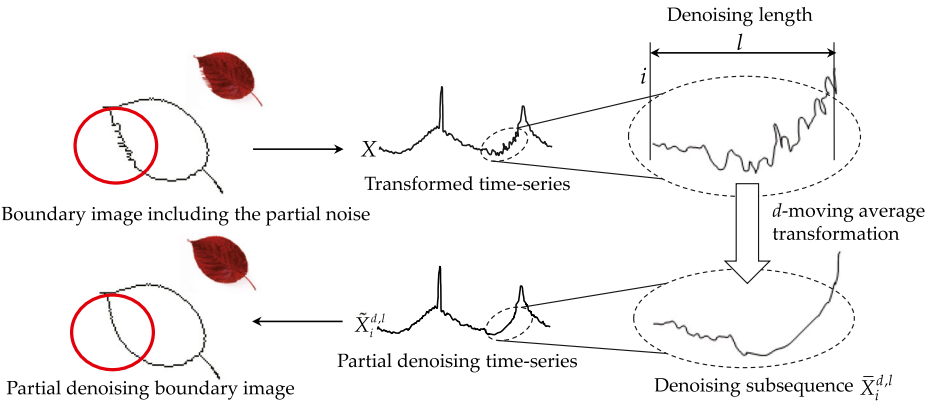


Fig. 5 Examples of the denoising subsequence and the partial denoising time-series

Definition 4 Given a query (boundary) time-series Q, a tolerance ϵ, a denoising level d, and a denoising length l, if a data (boundary) time-series T of a data image whose the partial denoising distance $PDD(Q, T, d, l)$ is less than or equal to ϵ; i.e., $PDD(Q, T, d, l) \leq \epsilon$, we say that T is *similar* to Q. Also, we call finding all such similar images from the image database *partial denoising boundary (image) matching*.

We propose naive algorithms for performing partial denoising boundary matching. We first present a range query algorithm that finds data time-series whose the distance from a query time-series is less than or equal to ϵ. Also, we propose a k-NN(nearest neighbor) query algorithm that finds the k nearest data time-series from a query time-series. These algorithms use the partial denoising distance in Definition 3. This range query algorithm is fundamental to partial denoising boundary matching, and we can extend to k-NN query algorithm as the query method. Algorithm 1 shows the range query algorithm. The inputs to the algorithm are a time-series database, a query time-series Q, a tolerance ϵ, the denoising level d, the denoising length l; the outputs are similar data time-series. As shown in the algorithm, in Lines 2 to 5 we access each data time-series stored in a time-series database. Then, in Line 3 we compute the partial denoising distance between a query time-series and each data time-series and finally identify the similar data time-series if this distance satisfies the tolerance.

Algorithm 1 *Naive-range* (Q, $\mathbb{T}$, ϵ, d, l)

// Query time-series Q, Time-series database $\mathbb{T}$, Tolerance ϵ, Denoising level d,
// Denoising length l

```
1: ℝ := ∅   // ℝ = the result set
2: for each data time-series T ∈ 𝕋 do
3:       if PDD(Q, T, d, l) ≤ ε then
4:             ℝ := ∪ {T};
5:       end-if
6: end-for
7: return ℝ;
```

Algorithm 2 shows the k-NN query algorithm. The inputs are the same as the range query algorithm except the number of results k instead of the tolerance. As shown in the matching operations, in Line 1 we first generate the priority queue of k entries. Next, in Lines 3 to 9 we access each data time-series stored in a time-series database and compare with a query time-series and the data time-series. In Lines 4 to 6 if the partial denoising distance between those time-series is less than or equal to the maximum value of the priority queue, we insert such data time-series into that queue. If the priority queue is full, in Line 5 we delete the entry of it corresponding to the maximum value from it. Then, in Line 6 we insert the data-time-series and the value of the partial denoising distance into the priority queue. In Line 7 we also update the maximum value of the priority queue. In Line 10 if all comparisons between the query time-series and each data time-series are finished, we finally obtain the k nearest data time-series from it by computing the partial denoising distance.

Algorithm 2 *Naive-kNN* (Q, $\mathbb{T}$, k, d, l)

// Query time-series Q, Time-series database $\mathbb{T}$, Number of results k,
// Denoising level d, Denoising length l

```
R := priority queue having k entries;   // entry = <time-series, distance>
maxdist = ∞;
for each data time-series T ∈ 𝕋 do
    if dist := PDD(Q, T, d, l) ≤ maxdist then
        if R is full then R.dequeue( );   // dequeue a time-series having the maximum distance
        R.enqueue(< T, dist >);   // enqueue the current entry of < T, dist >
        maxdist := R[k].distance;   // the maximum distance in R
    end-if
end-for
return R;
```

In order to facilitate the understanding of these algorithms, Fig. 6 describes the overall framework of the partial denoising boundary image matching system. In the preprocessing step, we convert images to boundary time-series by CCD and generate a query time-series as well as a time-series database. In the matching step, we compute the partial denoising distance between a query time-series and each data time-series stored in the time-series database with the denoising length and the denoising level given by a user. That is, we compute the *minimum* distance from the query time-series to the partial denoising time-series generated by all possible positions of partial denoising on each data time-series. We

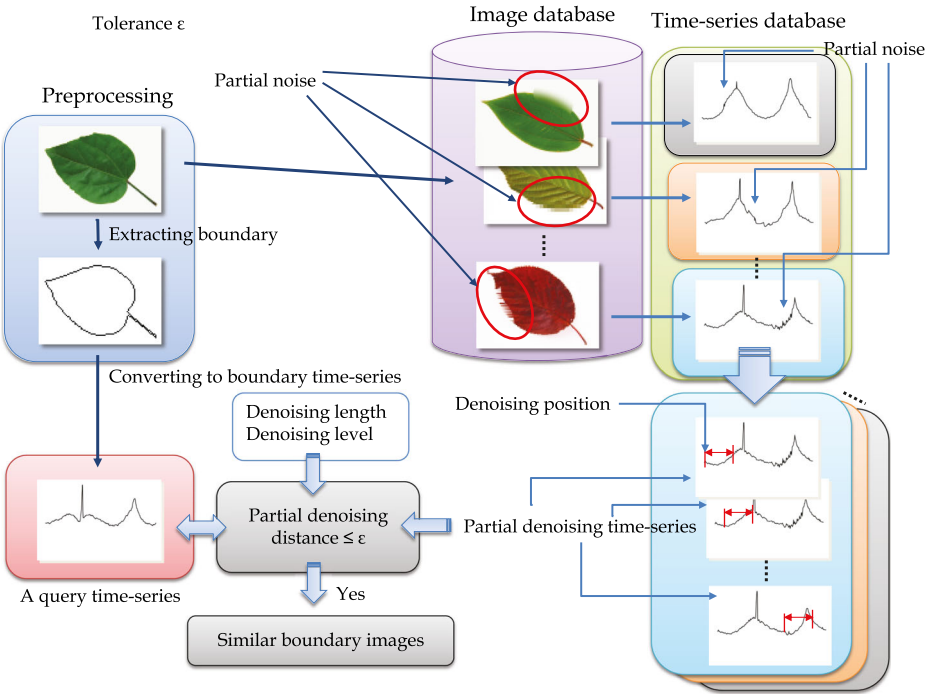


Fig. 6 The overall framework of the partial denoising boundary image matching system

then compare this minimum distance with a given tolerance (or a k-th value) to identify similar boundary images.

As shown in Algorithms 1 and 2, these algorithms are simple, but incur high CPU overhead. That is, these algorithms incur a heavy computational overhead since the partial denoising distance is computed on all possible data time-series. To solve this overhead, Section 3.2 proposes the lower bound of the partial denoising distance and describes the matching algorithms based on the lower bound, and Section 3.3 presents the process optimizing the computation of the partial denoising distance.

3.2 Lower bound of partial denoising distance

The most important operation in partial denoising boundary matching is the partial denoising distance. As shown in Algorithms 1 and 2, the computation of the partial denoising distances from the query time-series to each data time-series occurs frequently. For more accurate analysis, we analyze the computational complexity of the partial denoising distance $PDD(Q, T, d, l)$ as follows. First, the moving average transform for partial denoising incurs $\Theta(dl)$ [16], where the denoising level and the denoising length are d and l, respectively. The Euclidean distance also incurs $\Theta(n)$. Thus, the Euclidean distance between a query time-series and a data time-series, such as $D\left(Q, \widetilde{T}_i^{d,l}\right)$, incurs $\Theta(ndl)$. However, in Definition 3, $PDD(X, Y, d, l)$ incurs $\Theta\left(n^2 dl\right)$ since the computation of the partial denoising distance is repeated n times for the minimum distance. We can then say that this complexity is significantly very high because it exploits a sizable database that is a large number of data time-series to be compared.

To solve this high computational complexity of the partial denoising distance, in this paper, we present its lower bound and exploit this lower bound to the matching. The complexity of the proposed lower bound, of course, incurs less than the partial denoising distance. Thus, we can improve the matching performance by pruning a lot of data time-series after computing that lower bound. In Theorem 1 we propose the lower bound of the partial denoising distance as follows.

Theorem 1 *Let two boundary time-series of the length n be X and Y, respectively. The following $PDD_{LB}(X, Y, d)$ in (5) is the lower bound of $PDD(X, Y, d, l)$, where the denoising level d and the denoising length l.*

$$PDD_{LB}(X, Y, d) = \sqrt{\sum_{i=0}^{n-1} \begin{cases} (x_i - u_i)^2 & \text{if } x_i > u_i; \\ (x_i - l_i)^2 & \text{if } x_i < l_i; \\ 0 & \text{otherwise;} \end{cases}} \tag{5}$$

$$\text{where } L = \{l_0, l_1, \ldots, l_{n-1}\}, l_i = \min\left\{y_i, y_i^{d,n}\right\},$$

$$\text{and } U = \{u_0, u_1, \ldots, u_{n-1}\}, u_i = \max\left\{y_i, y_i^{d,n}\right\}.$$

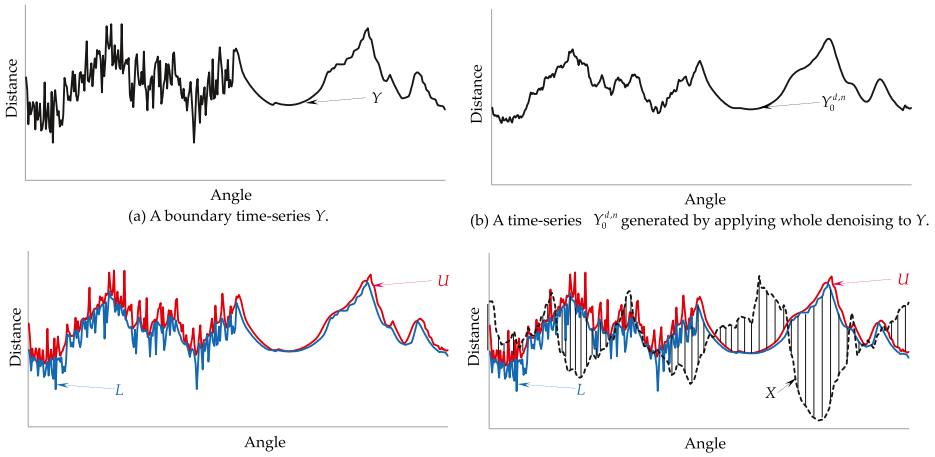
Proof Assume that Z is the partial denoising time-series of T whose partial denoising distance with the query time-series Q is the minimum distance. The partial denoising distance is computed by $D(Q, Z) = \sqrt{\sum_{i=0}^{n-1} |q_i - z_i|^2}$. All possible partial denoising time-series $\widetilde{T}_i^{d,l}$ including Z is contained between L and U. We here note that L is time-series constructing the minimum value as compared with T and $\widetilde{T}_i^{d,l}$ every same position and U also is

time-series constructing the maximum value as the same method above; that is, $l_i \leq z_i \leq u_i$ holds. Here, we know that if $q_i > u_i$, $|q_i - z_i| \geq |q_i - u_i|$ holds by $z_i \leq u_i$ and if $q_i < l_i$, $|q_i - z_i| \geq |q_i - l_i|$ holds by $l_i \leq z_i$. Otherwise, $(l_i \leq q_i \leq u_i)$ and $|q_i - z_i| \geq 0$ trivially holds. Thus, $PDD_{LB}(Q, T, d)$ obtained by summing $(q_i - u_i)^2$, $(q_i - l_i)^2$, and 0 should be less than or equal to $D(Q, Z)$ obtained by summing $(q_i - z_i)^2$. Therefore, $PDD_{LB}(Q, T, d)$ is a lower bound of $D(Q, Z)$. □

Figure 7 shows a graphical representation of the lower bound of the partial denoising distance on boundary time-series X and Y. In Fig. 7a, a boundary time-series Y contains partial noise. Figure 7b represents a time-series $Y_0^{d,n}$ generated by applying whole denoising to Y. We then can generate time-series L and U by using Y and $Y_0^{d,n}$. As shown in Fig. 7c, a time-series L is constructed by minimum values between Y and $Y_0^{d,n}$, and a time-series U is constructed by maximum values between Y and $Y_0^{d,n}$. Thus, we can obtain its lower bound distance from another boundary time-series X by summing up the shaded area of Fig. 7d.

In Theorem 1 the lower bound $PDD_{LB}(X, Y, d)$ incurs $\Theta(nd)$; that is, this complexity only needs $\Theta(d)$ and $\Theta(n)$, where d is the denoising level for the moving average transform, and n is the number of computing the distance and constructing U and L from Y. In conclusion, the lower bound $PDD_{LB}(X, Y, d)$ is improved $\Theta(nl)$ times than the partial denoising distance $PDD(X, Y, d, l)$. Thus, we improve the matching performance of $PDD(X, Y, d, l)$ by using its lower bound $PDD_{LB}(X, Y, d)$.

We improve the naive matching algorithms of Algorithms 1 and 2 by exploiting the lower bound of the partial denoising distance in Theorem 1. The matching results are exactly same in the naive and lower bound-based matching algorithms since the pruning approach by lower bounds is proven to incur no false dismissal in Theorem 1. First, Algorithm 3 shows that the range query algorithm is improved by using its lower bound $PDD_{LB}(Q, T, d)$. This algorithm compared with that of Algorithm 1 uses two more additional operations. First, in Line 3 we construct two time-series L and U for computing its lower bound from the time-series T. Second, in Line 4 we compute the lower bound $PDD_{LB}(Q, T, d)$. Then, if it is



(a) A boundary time-series Y.

(b) A time-series $Y_0^{d,n}$ generated by applying whole denoising to Y.

(c) A time-series L representing a lower bound and a time-series U representing a upper bound.

(d) A lower bound of the partial denoising distance between boundary time-series X and Y.

Fig. 7 An example of the lower bound of the partial denoising distance between boundary time-series X and Y

more than the tolerance ϵ, we discard the corresponding data time-series. Otherwise, in Line 5 we compute the partial denoising distance $PDD(Q, T, d, l)$. Thus, we can improve the performance of the range query algorithm by as many as possible pruning with the lower bound.

Algorithm 3 *LB-range* $(Q, \mathbb{T}, \epsilon, d, l)$

```
// Query time-series Q, Time-series database 𝕋, Tolerance ε, Denoising level d,
// Denoising length l
 1: ℝ := ∅   // ℝ = the result set
 2: for each data time-series T ∈ 𝕋 do
 3:       Construct L and U from T using d and l;
 4:       if PDD_LB(Q, T, d) ≤ ε then
 5:            if PDD(Q, T, d, l) ≤ ε then
 6:                 ℝ := ∪{T};
 7:            end-if
 8:       end-if
 9: end-for
10: return ℝ;
```

Algorithm 4 shows that the k-NN query algorithm of Algorithm 2 is improved by using the lower bound $PDD_{LB}(Q, T, d)$. As the same of Algorithm 3, this algorithm compared with that of Algorithm 2 uses two more additional operations by its lower bound. First, in Line 4 we also construct two time-series L and U for computing its lower bound from the time-series T. Second, in Line 5 we prune discarded data time-series before the partial denoising distance is computed. Thus, as the data time-series pruned by its lower bound increases, partial denoising boundary matching provides the higher performance.

Algorithm 4 *LB-kNN* $(Q, \mathbb{T}, k, d, l)$

```
// Query time-series Q, Time-series database 𝕋, Number of results k,
// Denoising level d, Denoising length l
 1: R := priority queue having k entries;   // entry = <time-series, distance>
 2: maxdist = ∞;
 3: for each data time-series T ∈ 𝕋 do
 4:       Construct L and U from T using d and l;
 5:       if PDD_LB(Q, T, d) ≤ maxdist then
 6:            if dist := PDD(Q, T, d, l) ≤ maxdist then
 7:                 if R is full then R.dequeue( );   // dequeue a time-series having the maximum distance
 8:                 R.enqueue(< T, dist >);   // enqueue the current entry of < T, dist >
 9:                 maxdist := R[k].distance;   // the maximum distance in R
10:            end-if
11:       end-if
12: end-for
13: return R;
```

3.3 Optimization of partial denoising distance

In the paper, we present an optimization technique for improving the performance of the partial denoising distance. We note that, in computing the partial denoising distance, the same squared sum of the Euclidean distance is repeated for each partial denoising time-series. To eliminate this repetition, we use a dynamic programming technique which stores cumulative squared sums of the Euclidean distance in memory and reuses it in the next

same operation. For this, given two boundary time-series $X = \{x_0, x_1, \ldots, x_{n-1}\}$ and $Y = \{y_0, y_1, \ldots, y_{n-1}\}$, we first define a list of cumulative squared sums $\mathbb{S}_{X,Y}$ between X and Y as the following (6).

$$\mathbb{S}_{X,Y} = \left\{ S_0 = (x_0 - y_0)^2, S_1 = S_0 + (x_1 - y_1)^2, \ldots, S_{n-1} = S_{n-2} + (x_{n-1} - y_{n-1})^2 \right\} \tag{6}$$

Next, given the *whole denoising time-series* $Y^{d,n} = \left\{ y_0^{d,n}, y_1^{d,n}, \ldots, y_{n-1}^{d,n} \right\}$ instead of Y, we also define $\mathbb{S}_{X,Y^{d,n}}$ as the following (7).

$$\mathbb{S}_{X,Y^{d,n}} = \left\{ \widetilde{S}_0 = \left(x_0 - y_0^{d,n}\right)^2, \widetilde{S}_1 = \widetilde{S}_0 + \left(x_1 - y_1^{d,n}\right)^2, \ldots, \widetilde{S}_{n-1} = \widetilde{S}_{n-2} + \left(x_{n-1} - y_{n-1}^{d,n}\right)^2 \right\} \tag{7}$$

Thus, we can rewrite $D\left(X, \widetilde{Y}_i^{d,l}\right)$ by using (6) and (7) in (4) of $PDD(X, Y, d, l)$. For example, given a query time-series X and a partial denoising time-series $\widetilde{Y}_1^{4,4}$ of length 8, $D\left(X, \widetilde{Y}_1^{4,4}\right)$ is computed by using (6) and (7) as follows.

$$\begin{aligned} D\left(X, \widetilde{Y}_1^{4,4}\right) &= \sqrt{\begin{array}{l}(x_0 - y_0)^2 + \left(x_1 - y_1^{4,4}\right)^2 + \left(x_2 - y_2^{4,4}\right)^2 + \left(x_3 - y_3^{4,4}\right)^2 \\ + \left(x_4 - y_4^{4,4}\right)^2 + (x_5 - y_5)^2 + (x_6 - y_6)^2 + (x_7 - y_7)^2\end{array}} \\ &= \sqrt{S_0 + (\widetilde{S}_4 - \widetilde{S}_0) + (S_7 - S_4)} \\ &= \sqrt{(S_7 - S_4 + S_0) + (\widetilde{S}_4 - \widetilde{S}_0)} \end{aligned}$$

That is, we can compute the Euclidean distance from X to all possible partial denoising time-series of Y as three cases of the denoising position: (1) $i = 0$, (2) $0 < i \leq n - l$, and (3) $n - l < i \leq n - 1$.

In more detail, if $i = 0$, which is the first point on the partial denoising time-series $\widetilde{Y}_0^{d,l}$, we can represent the Euclidean distance between X and $\widetilde{Y}_0^{d,l}$ as the following equations in $PDD(X, Y, d, l)$.

$$\begin{aligned} D\left(X, \widetilde{Y}_0^{d,l}\right) &= \sqrt{\sum_{j=0}^{n-1} \left| x_j - \widetilde{y}_{0,j}^{d,l} \right|^2} \\ &= \sqrt{\begin{array}{l}\underbrace{(x_0 - y_0^{d,l})^2 + (x_1 - y_1^{d,l})^2 + \ldots + (x_{l-1} - y_{l-1}^{d,l})^2}_{=\widetilde{S}_{l-1}} \\ + \underbrace{(x_l - y_l)^2 + (x_{l+1} - y_{l+1})^2 + \ldots + (x_{n-1} - y_{n-1})^2}_{=S_{n-1} - S_{l-1}}\end{array}} \\ &= \sqrt{\widetilde{S}_{l-1} + S_{n-1} - S_{l-1}} \\ &= \sqrt{S_{n-1} - S_{l-1} + \widetilde{S}_{l-1}} \end{aligned}$$

Next, if $0 < i \leq n-l$, which means the consecutive denoising subsequence until the last point of the partial denoising time-series, the Euclidean distance between X and $\widetilde{Y}_i^{d,l}$ is rewritten as follows.

$$
\begin{aligned}
D\left(X, \widetilde{Y}_i^{d,l}\right) &= \sqrt{\sum_{j=0}^{n-1} \left| x_j - \widetilde{y}_{i,j}^{d,l} \right|^2}, \text{ where } 0 < i \leq n-l \\
&= \sqrt{\begin{array}{l} \underbrace{(x_0 - y_0)^2 + (x_1 - y_1)^2 + \ldots + (x_{i-1} - y_{i-1})^2}_{=S_{i-1}} \\ + \underbrace{(x_i - y_i^{d,l})^2 + (x_{i+1} - y_{i+1}^{d,l})^2 + \ldots + (x_{i+l-1} - y_{i+l-1}^{d,l})^2}_{=\widetilde{S}_{i+l-1} - \widetilde{S}_{i-1}} \\ + \underbrace{(x_{i+l} - y_{i+l})^2 + (x_{i+l+1} - y_{i+l+1})^2 + \ldots + (x_{n-1} - y_{n-1})^2}_{=S_{n-1} - S_{i+l-1}} \end{array}} \\
&= \sqrt{S_{n-1} - S_{i+l-1} + S_{i-1} + \widetilde{S}_{i+l-1} - \widetilde{S}_{i-1}}
\end{aligned}
$$

Likewise, if $n-l < i \leq n-1$, i.e., the subsequent point of the last point on the partial denoising time-series is the first point in the consecutive denoising subsequence, we also rewrite this Euclidean distance as follows.

$$
\begin{aligned}
D\left(X, \widetilde{Y}_i^{d,l}\right) &= \sqrt{\sum_{j=0}^{n-1} \left| x_j - \widetilde{y}_{i,j}^{d,l} \right|^2}, \text{ where } n-l < i \leq n-1 \\
&= \sqrt{\begin{array}{l} \underbrace{(x_0 - y_0^{d,l})^2 + (x_1 - y_1^{d,l})^2 + \ldots + (x_{(i+l-1)\%n} - y_{(i+l-1)\%n}^{d,l})^2}_{=\widetilde{S}_{(i+l-1)\%n}} \\ + \underbrace{(x_{((i+l-1)\%n)+1} - y_{((i+l-1)\%n)+1})^2 + \ldots + (x_{i-1} - y_{i-1})^2}_{=S_{i-1} - S_{(i+l-1)\%n}} \\ + \underbrace{(x_i - y_i^{d,l})^2 + (x_{i+1} - y_{i+1}^{d,l})^2 + \ldots + (x_{n-1} - y_{n-1}^{d,l})^2}_{=\widetilde{S}_{n-1} - \widetilde{S}_{i-1}} \end{array}} \\
&= \sqrt{\widetilde{S}_{(i+l-1)\%n} + S_{i-1} - S_{(i+l-1)\%n} + \widetilde{S}_{n-1} - \widetilde{S}_{i-1}} \\
&= \sqrt{S_{i-1} - S_{(i+l-1)\%n} + \widetilde{S}_{n-1} - \widetilde{S}_{i-1} + \widetilde{S}_{(i+l-1)\%n}}
\end{aligned}
$$

Finally, we rewrite (4) as (8) using (6) and (7). Here, we call (8) the *optimized partial denoising distance*.

$$
PDD_{opt}(X, Y, d, l) = \min_{i=0}^{n-1} \sqrt{\begin{array}{ll} S_{n-1} - S_{l-1} + \widetilde{S}_{l-1} & \text{if } i = 0; \\ S_{n-1} - S_{i+l-1} + S_{i-1} + \widetilde{S}_{i+l-1} - \widetilde{S}_{i-1} & \text{if } 0 < i \leq n-l; \\ S_{i-1} - S_{(i+l-1)\%n} + \widetilde{S}_{n-1} - \widetilde{S}_{i-1} + \widetilde{S}_{(i+l-1)\%n} & \text{if } n-l < i \leq n-1. \end{array}} \tag{8}
$$

These $\mathbb{S}_{X,Y}$ and $\mathbb{S}_{X,Y^{d,n}}$ are computed by scanning a data time-series only once. Afterwards, we store these sums into arrays and repeatedly reuse them in computing the partial denoising distance. In the optimization precess, the computational complexity only incurs

$\Theta(n)$, and the space complexity just increases to $\mathcal{O}(2n)$, which is negligible in real implementation, more than that of the partial denoising distance. Although the complexity of the optimized partial denoising distance is almost same as that of the proposed lower bound, the performance of this optimization-based matching algorithm is the superior to that of the lower bound-based matching algorithm regardless of the number of similar images since computing PDD is unnecessary in the optimization-based matching algorithm. Thus, the optimization is practically applicable to large boundary image databases.

4 Experimental evaluation

4.1 Experimental data and environment

In the experiments, we constructed an image database consisting of a total of one hundred thousand images. For this, we first collected 10 thousand original images from the Web. Afterwards, we generated boundary images including nine different partial noises by changing the length and the position from each original image. We used the Gaussian noise model [8] and then applied it to the noise generation of boundary images. Figure 8 showed how we generated boundary images including these nine different partial noises. As shown in the figure, we generated boundary images including nine different partial noises from the original i mage by changing each of the length $l = \{36, 72, 108\}$ and the position $i = \{0, 120, 240\}$.

A total of 102,590 boundary time-series were generated from the original images and boundary images including different partial noises by using the CCD method, and those time-series were stored in the time-series database. Even though we used one hundred

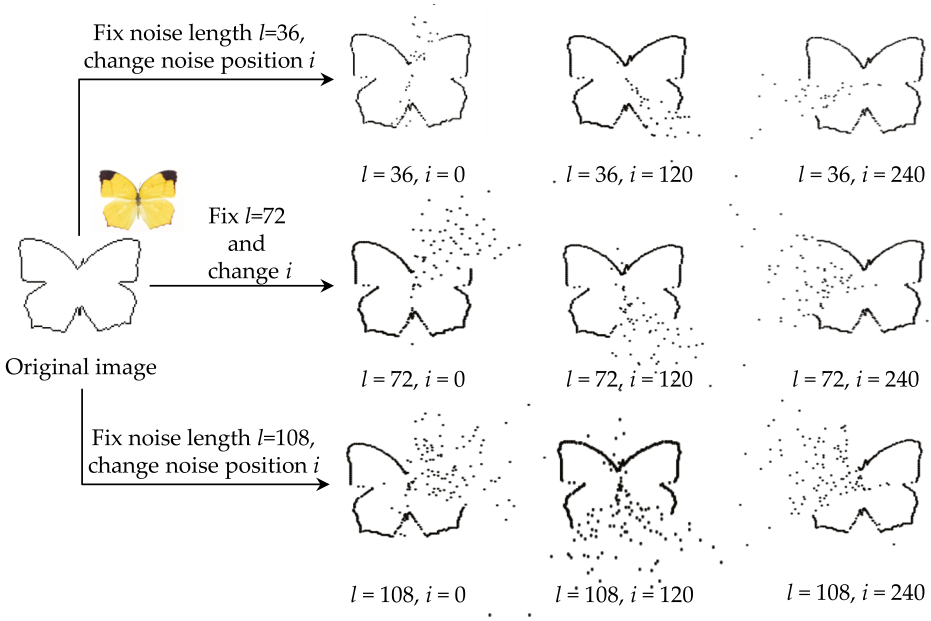


Fig. 8 Examples of nine different partial noise images generated from an original image

thousand images, we extracted more than the number of those images since one image might contain multiple boundary objects. Thus, a boundary time-series included the ID number of the original image and its real values of the length 360 as a text file format.

We performed the experiments in the following environment. The hardware platform was an IBM compatible PC equipped with a 2.0GHz Intel Core 2 Duo CPU, 2.0GB RAM, and 500GB hard disk. Its software platform was the CentOS 6.3 Linux operating system. We used GNU C language for the implementation. Converting an image to a time-series can be done by two steps: (1) the binary transformation of a gray-scaled image and (2) the boundary tracing of a binary image. For the binary transformation, we performed repetitive evaluations and chose 240 as the binary threshold. Also, we used a well-known tracing algorithm exploiting 8-neighborhood connectivity [8] for the boundary tracing of a binary image.

4.2 Experimental results

We performed extensive experiments using a variety of the denoising level and the denoising length to confirm the effectiveness of partial denoising boundary matching. The denoising levels used for those experiments were 6, 12, 24, 32, and 48; likewise, the denoising lengths were 36, 72, 108, 144, and 180. That is, we confirmed the change of the matching results by a variety of denoising levels and denoising lengths. In the experiments, we set the query tolerance ϵ to the maximum among the tolerance values that return one image; that is, another similar data image, except a query image itself, as the boundary image matching result without considering the partial noise. If the denoising level was 1, we used the tolerance that returns one image except a query image itself as the partial denoising boundary matching result.

Figure 9 shows the experimental result when a leaf image is used as a query. In Fig. 9, we fix the denoising length l=108 and the tolerance ϵ=33.0, but increase the denoising level d from 6 to 48. As shown in the figure, when the denoising level is the smallest value (i.e., when d=6), all three boundary images including the partial noise are retrieved as similar

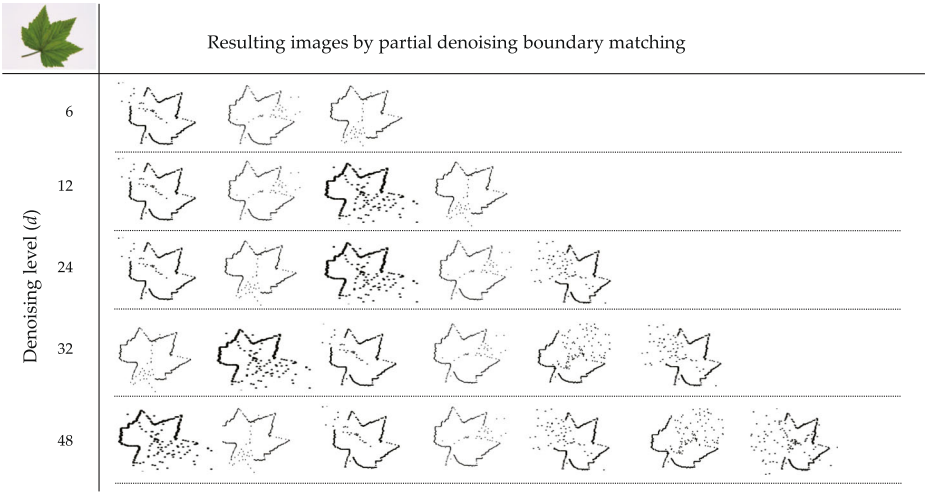


Fig. 9 Partial denoising boundary matching results on different denoising levels (l=108)

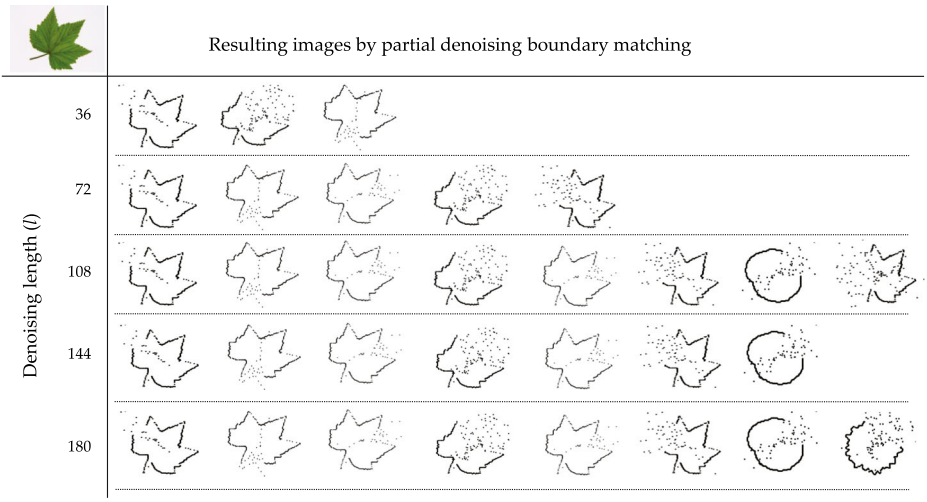


Fig. 10 Partial denoising boundary matching results on different denoising lengths (d=24)

images. These images are the same as the original image, and the length of the partial noise, which is 36, is the same. However, the positions of each partial noise are different. We note that as the denoising level increases, the more boundary images are returned as similar ones, and then both the length and the position of the partial noise are varied. This is because as the denoising level increases, the matching result is largely affected by the partial denoising using the moving average transform. In particular, when d=48, the partial denoising boundary matching returns a total of seven boundary images among nine boundary images, which are generated from the same original image, as similar ones. Thus, the proposed method can find similar images by changing the denoising level.

Figure 10 shows, when d=24 and ϵ=33.0, those matching results varying as the denoising length using the same leaf image used in Fig. 9. Like Fig. 9, as the denoising length increases, the more boundary images, which include various partial noises, are returned as similar ones. Unlike Fig. 9, however, as l increases, some different boundary images, which are not generated from the same original image, are also returned as similar ones. That is, in Fig. 10, our matching method returns boundary images of the stone and the toothed wheel, which are similar to the boundary of the leaf image. This means that, the larger l exploits the larger partial denoising, but at the same time it may return the more wrong images as similar ones. In more detail, as the denoising length increases, the more boundary characteristics may distort by the partial denoising, and then the more boundary images that are different from the original image are returned as similar ones. Even though the result set contains one or two wrong images, we still find most leaf boundary images in Fig. 10. As a result, we conclude that our partial denoising boundary matching retrieves similar images almost correctly, and the denoising length and the denoising level can be used as measures of controlling the degree of the partial denoising.

Table 1 shows the experimental result that compares the proposed partial denoising boundary matching with the previous shape context matching [2]. We have obtained the source code of the shape context matching from the Web site[3] and slightly modified it for

[3] http://www.umiacs.umd.edu/~zhengyf/PointMatching.htm

Table 1 Comparison of the partial denoising boundary matching and the shape context matching

	Partial denoising boundary matching (tp)					Shape context matching (tp)				
	Denoising level					Regularization parameter				
k-NN	6	12	24	32	48	1	1.5	2	2.5	3
k=1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
k=2	1.25	1.25	1.44	1.56	1.81	1.38	1.38	1.38	1.38	1.38
k=3	1.56	1.62	1.62	1.88	2.06	1.81	1.81	1.81	1.81	1.75
k=4	2.00	1.94	1.88	2.06	2.44	2.06	2.06	2.00	2.00	2.00
k=5	2.06	2.06	2.12	2.38	2.81	2.19	2.19	2.19	2.19	2.19

our experimental environment. We repeat the same experiment for another data set, MixedBag [25], which was used in [17, 25]. In this experiment, we generate 1,440 partial noise images from 160 original images of MixedBag by using the partial noise generation method of this section, which generates nine different partial noise images from one original image, and use total 1,600 images containing 160 original images. As the experimental results, in the k-nearest neighbor (k-NN) search we count true positives (i.e., similar images) by varying the denoising level d and holding the denoising length 108 for our approach and by varying the *regularization* parameter, which controls the sensitivity of boundaries [2] for the shape context matching. For each pair of (k of k-NN, denoising level or regularization parameter), we run 16 different query images, which are 10 % of 160 original images, and use their average as the result. As shown in Table 1, the matching accuracy of the proposed partial denoising boundary matching is in general higher than that of the shape context matching. That is, we can say that our partial denoising boundary matching correctly retrieves similar images, and the denoising level d can be used as a measure of controlling the degree of partial noise reduction. Thus, we also can say that, as the denoising level increases, we can provide the more accurate matching results. However, although the regularization parameter increases, the results of the shape context matching show negligible differences. The shape context matching also takes much more time than partial denoising boundary matching since it requires the high computational complexity. For example, the shape context matching takes average 676 seconds for processing a query image while our partial denoising boundary matching takes only average 1.3 seconds. Thus, it is very difficult to use the shape context matching for the large-scale boundary image database while the partial denoising boundary matching is very suitable for it.

4.3 Comparison of naive and advanced matching algorithms

In this subsection we compare the query response times of the naive and the advanced matching algorithms for partial denoising boundary matching. As we explained in Section 3, while the naive matching algorithms compute the partial denoising distances to all data time-series, the advanced matching algorithms compute the partial denoising distance considering other data time-series only except the dissimilar ones that are pruned by its lower bound and the optimized partial denoising distance, respectively. In this subsection, we confirm the performance improvement by this pruning and the optimization of the partial denoising distance with the experimental results. In the experiments, we measure the query response times of those matching methods by changing the denoising level, the denoising length, and the number of similar images, respectively. As query images, we use

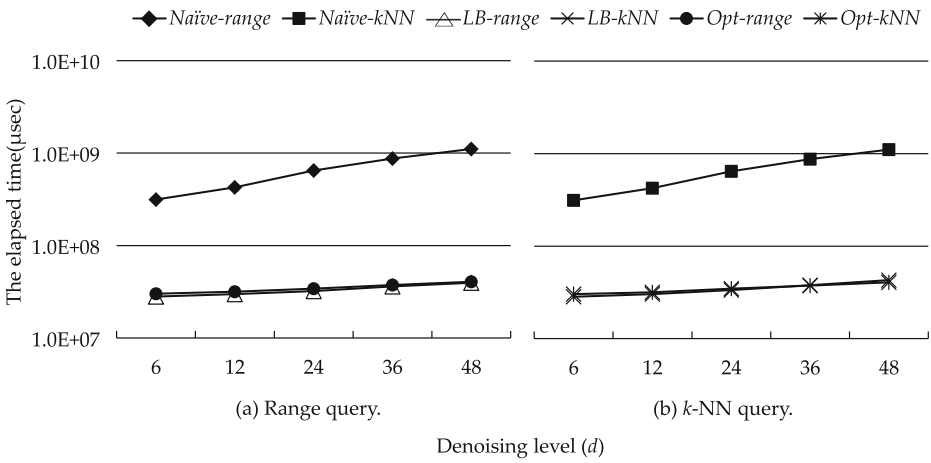


Fig. 11 The query response time on different denoising levels (l =72)

one hundred images that are randomly selected from 10 thousand original images. Afterwards, we measure the elapsed times of one hundred query images and use their average as the experimental result.

First, Fig. 11 shows the elapsed times of the range and k-NN query algorithms by changing the denoising level on the fixed denoising length and tolerance corresponding to k=10. As shown in the figure, the lower bound-based (matching) algorithms (*LB-range* and *LB-kNN*) and the algorithms based on the optimized partial denoising distance (*Opt-range* and *Opt-kNN*) significantly reduce the elapsed times compared with the naive (matching) algorithms (*Naïve-range* and *Naïve-kNN*). (Note that Y-axis is a log-scale.) Using the optimized partial denoising distance and pruning dissimilar images by the proposed lower bound cause this performance improvement, respectively. Meanwhile, as shown in the detailed figure, as the denoising level increases, the elapsed times of all algorithms show the slightly increasing trend. This is because the partial denoising needs the higher computational overhead as the denoising level increases. In summary of the results in Fig. 11, the advanced algorithms improve the performance by 11 to 28 times compared with the naive algorithms.

Next, Fig. 12 shows the elapsed times of the range and k-NN query algorithms by changing the denoising length on the fixed denoising level and tolerance corresponding to k=10. As shown in the figure, the advanced algorithms still outperform the naive algorithms in all denoising levels. We note that, as the denoising length increases, the elapsed time of the advanced algorithms shows little change while that of the naive algorithms increases. This is because, while the computation time the advanced algorithms is independent on the denoising length, the naive algorithms incur a lot of the elapsed time for generating the partial denoising time-series as the denoising length increases; that is, in the case of the lower bound-based algorithms, they shows no change since the pruning effect is greater than that of the lower bound computation. On the other hand, the elapsed time of the naive algorithms increases since they need more computations as the denoising length increases. In more detail, they access all data time-series and compute the partial denoising distance to each data time-series. In the experimental results in Fig. 12, the advanced algorithms improve the performance by 12 to 45 times compared with the naive algorithms.

Finally, Fig. 13 shows the elapsed times of the range and k-NN query algorithms by changing the tolerance ϵ or the number of images retrieved k on the fixed denoising level

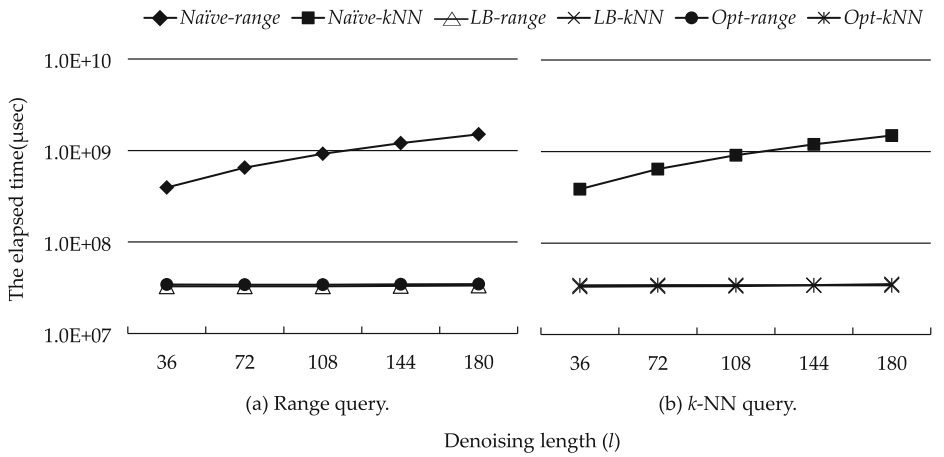


Fig. 12 The query response time on different denoising lengths (d =24)

and denoising length. This number of results can be controlled by changing the tolerance for range queries and k for k-NN queries. Like the experimental results in Figs. 11 and 12, the advanced algorithms also outperform the naive algorithms in all cases. We note that, as the tolerance or k increases, lower bound-based algorithms incur the more performance degradation. In the naive algorithms, the performance differences by changing the tolerance or k show no change since we compute the partial denoising distances to all possible data time-series. Thus, in lower bound-based algorithms, this degradation is because the number of pruning by the lower bound decreases as the tolerance or k increases. Also, the elapsed time of the k-NN query takes more than that of the range query since using the priority queue

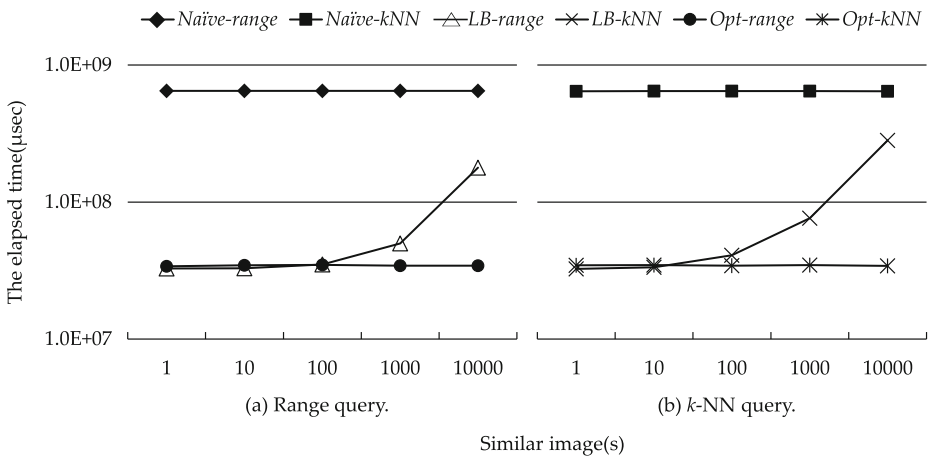


Fig. 13 The query response time on different tolerances or numbers of k (d =24, l =72)

incurs the overhead. In the case of the optimization-based algorithms, they outperform the lower bound-based algorithms regardless of the number of similar images in all case. In the experimental results in Fig. 13, the advanced algorithms improve the performance by 19 times compared with the naive algorithms.

5 Conclusions

In this paper we solved the partial denoising problem of boundary image matching using time-series matching techniques. The contributions of the paper can be summarized as follows. First, we defined the *partial denoising time-series* and proposed a method to efficiently construct this partial denoising time-series in the time-series domain. Second, we presented a notion of *partial denoising distance* as a similarity measure of boundary images. Third, we proposed the lower bound of the partial denoising distance between two boundary time-series and proved its correctness. Fourth, we optimized the computation of the partial denoising distance for improving performance. Fifth, we presented the matching algorithms of *range* and *k-NN* query, respectively. Sixth, through the extensive experiments, we showed that the partial denoising boundary matching was intuitively and correctly performed and the superiority of the advanced matching algorithms over the naive matching algorithms was validated. Experimental results indicated that our solution provided similar boundary images including the partial noise, which were not found by the simple boundary image matching, as the matching results. Also, the advanced matching algorithms of the lower bound and the optimized partial denoising distance outperformed the naive matching algorithms by one or two orders of magnitude.

As the future work, we plan to redefine the partial denoising distance by the denosing length and the denoising level of arbitrary values and present its solution. We also may use a lower dimensional transformation and a multidimensional index. We also may propose a partial denoising boundary matching using this multidimensional index for efficient matching.

Acknowledgments This research, "Geospatial Big Data Management, Analysis and Service Platform Technology Development," was supported by the MOLIT(The Ministry of Land, Infrastructure and Transport), Korea, under the national spatial information research program supervised by the KAIA(Korea Agency for Infrastructure Technology Advancement) (16NSIP-B081011-03).

References

1. Agrawal R, Faloutsos C, Swami A (1993) Efficient similarity search in sequence databases. In: Proceedings of the 4th Int'l Conf. on Foundations of Data Organization and Algorithms, Chicago, Illinois, pp 69–84
2. Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. IEEE Trans. on Pattern Analysis and Machine Intelligence 24(4):509–522
3. Chu KW, Wong MH (1999) Fast time-series searching with scaling and shifting. In: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Philadelphia, Pennsylvania, pp 237–248
4. Do MN (2002) Wavelet-based texture retrieval using generalized gaussian density and Kullback-Leibler distance. IEEE Trans Image Processing 11(2):146–158

5. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. In: Proceedings of the 34th Int'l Conf. on Very Large Data Bases, Auckland, New Zealand, pp 1542–1552
6. Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases. In: Proceedings of the Int'l Conf. on Management of Data, ACM SIGMOD, Minneapolis, Minnesota, pp 419–429
7. Gil M-S, Moon Y-S, Kim B-S (2011) Linear detrending subsequence matching in time-series databases. IEICE Trans Information and Systems E94-D(4):917–920
8. Gonzalez RC, Woods RE (2008) Digital image processing, 3rd Edn. Prentice Hall, New Jersey
9. Han W-S, Lee J, Moon Y-S, Hwang S-W, Yu H (2011) A new approach for processing ranked subsequence matching based on ranked union. In: Proceedings of Int'l Conf. on Management of Data, ACM SIGMOD, Athens, Greece, pp 457–468
10. Holte MB, Moeslund TB, Fihl P (2010) View-invariant gesture recognition using 3D optical flow and harmonic motion context. Comput Vis Image Underst 114(12):1353–1361
11. Kim B-S, Moon Y-S, Choi M-J, Kim J (2014) Interactive noise-controlled boundary image matching using the time-series moving average transform. Multimedia Tools and Applications 72(3):2543–2571
12. Lin C-H, Lin W-C (2011) Image retrieval system based on adaptive color histogram and texture features. Comput J 54(7):1136–1147
13. Liu H, Frishkoff G, Frank R, Dou D (2012) Sharing and integration of cognitive neuroscience data: Metric and pattern matching across heterogeneous ERP datasets. Neurocomputing 92:156–169
14. Loh W-K, Kim S-W, Whang K-Y (2004) A subsequence matching algorithm that supports normalization transform in time-series databases. Data Min Knowl Disc 9(1):5–28
15. Moon Y-S, Whang K-Y, Han W-S (2002) General match: a subsequence matching method in time-series databases based on generalized windows. In: proceedings of the Int'l Conf. on Management of Data, ACM SIGMOD, Madison, Wisconsin, pp 382–393
16. Moon Y-S, Kim J (2007) Efficient moving average transform-based subsequence matching algorithms in time-series databases. Inf Sci 177(23):5415–5431
17. Moon Y-S, Kim B-S, Kim MS, Whang K-Y (2010) Scaling-invariant boundary image matching using time-series matching techniques. Data Knowl Eng 69(10):1022–1042
18. Moon Y-S, Lee BS (2014) Safe MBR-transformation in similar sequence matching. Inf Sci 270(2):28–40
19. Mori G, Malik J (2002) Estimating human body configuration using shape context matching. In: Proceedings of the 7th European Conf. on Computer Vision, Copenhagen, Denmark, pp 666–680
20. Pratt WK (2007) Digital image processing, 4th Edn. Eastman Kodak Company, Rochester
21. Rafiei D, Mendelzon AO (2000) Querying time series data based on similarity. IEEE Trans Knowl Data Eng 12(5):675–693
22. Suetens P, Fua P, Hanson AJ (1992) Computational strategies for object recognition. ACM Comput Surv 24(1):5–62
23. Shumway RH, Stoffer DS (2006) Time series analysis and its applications: with r examples(Ed. 2) springer texts in statistics
24. Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Proceedings of the 18th IEEE Int'l Conf. on Data Engineering(ICDE), San Jose, California, pp 673–684
25. Vlachos M, Vagena Z, Yu PS, Athitsos V (2005) Rotation invariant indexing of shapes and line drawings. In: Proceedings of ACM Conf. on Information and Knowledge Management, Bremen, Germany, pp 131–138
26. Wang X-Y, Yu Y-J, Yang H-Y (2011) An effective image retrieval scheme using color, texture and shape features. Computer Standards & Interfaces 33(1):59–68

Bum-Soo Kim received his B.S. (2006) degree in computer engineering from Halla University and M.S. (2008) and Ph. D. (2013) degrees in computer science from Kangwon National University. From 2013 to 2014, he was a postdoctoral researcher in Advanced Information Technology Research Center (AITrc) from Korea Advanced Institute of Science and Technology (KAIST). From 2014 to 2015, he was a research engineer in Institute of Telecommunication and Information from Kangwon National University. He is currently a postdoctoral researcher in Industry Management Research Center from Korea Advanced Institute of Science and Technology (KAIST). His research interests include time-series data mining, social-network and graph data mining, and data mining applications.

Yang-Sae Moon received B.S. (1991), M.S. (1993) and Ph.D. (2001) degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST). From 1993 to 1997, he was a research engineer in Hyundai Syscomm, Inc., where he participated in developing 2G and 3G mobile communication systems. From 2002 to 2005, he was a technical director in Infravalley, Inc., where he participated in planning, designing, and developing CDMA and W-CDMA mobile network services and systems. He is currently a professor at Kangwon National University. He was a visiting scholar at Purdue University in 2008?-2009. His research interests include data mining, knowledge discovery, storage systems, access methods, multimedia information retrieval, mobile/wireless communication systems, and network communication systems. He is a member of the IEEE and of the ACM.

Jae-Gil Lee is an Associate Professor in the Department of Knowledge Service Engineering, KAIST. He worked at IBM Almaden Research Center before joining KAIST. He earned the Ph.D. degree from KAIST in 2005. His research interests encompass spatio-temporal data mining, social-network and graph data mining, and big data analysis with Hadoop/MapReduce.