IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 24, 2019, accepted September 6, 2019, date of publication September 11, 2019, date of current version September 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2940353

Image Scaling: How Hard Can It Be?

GENLANG CHEN[1,2], HUANYU ZHAO[3,4], CHRISTOPHER KUO PANG[5], TONGLIANG LI[3,4], AND CHAOYI PANG[1,2]

[1]Ningbo Institute of Technology, Zhejiang University, Ningbo 310058, China
[2]Ningbo Research Institute, Zhejiang University, Ningbo 310058, China
[3]Institute of Applied Mathematics, Hebei Academy of Sciences, Shijiazhuang 050051, China
[4]Hebei Authentication Technology Engineering Research Center, Shijiazhuang 050051, China
[5]Faculty of Engineering, Architecture and Information Technology, The University of Queensland, Brisbane, QLD 4072, Australia

Corresponding authors: Huanyu Zhao (huanyuzhao@qq.com) and Chaoyi Pang (chaoyi.pang@qq.com)

This work was supported in part by the Natural Science Foundation of China under Grant 61572022, in part by the Hebei Academy of Sciences Project under Grant 161303, in part by the Ningbo eHealth Project under Grant 2016C11024, and in part by the Hebei "One Hundred Plan" Project under Grant E2012100006).

ABSTRACT For centuries, human artists have been scaling images by hand, sketching with varying levels of detail depending on the demands. Image scaling is a fundamental image operation and exists for various devices including mobile phones and computers. It has great commercial values and has attracted great efforts in research. In this article, we suggest a stretch-shrink based framework for image scaling which imitates the work of a human artist. This can be done by firstly representing the image with line segments (sketches) and then stretching or shrinking the length of the representing segments under proper ratios to achieve the desired scaling. Through extensive experiments, this framework exhibits the prominent ability to achieve high quality scaled images efficiently with less storage consumption.

INDEX TERMS Image compression, image scaling (resizing), maximum-error bound, piecewise linear approximation (PLA), sample-rate interpolation.

I. INTRODUCTION

Image scaling refers to representing and resizing an original image with the use of a higher or lower number of pixels. Resizing an image under guaranteed quality has many applications ranging from daily activities to advanced sciences including astronomy, biology and medical sciences. To support qualified arbitrary scaling of images, many advanced techniques have been developed, such as sample-rate interpolation based techniques (i.e., bilinear, bicubic, box sampling etc.) which are designed to prevent aliasing artifacts.

The most common existing linear methods are limited in reconstructing complex structures, often resulting in aliasing artifacts, over-smoothed regions, and reduced sharp edges [9]. Traditional image up-scaling methods suffer from either a loss in high-frequency texture details or a high cost in execution.

A. IMAGE SCALING

In a 2013 analysis from [23], Directional Cubic Convolution Interpolation (DCCI) [27] had the best scores in PSNR and SSIM on a series of test images. Kong *et al.* [6] method is similar to [27] but is much more time efficient. Kopf *et al.* [7] proposed a filtering approach that adjusts the filter kernels on the image content, whereas Weber *et al.* [20] used convolutional filters to down scale images with hardly any temporal artifacts. Oztireli and Gross [12] use the structural similarity index to optimize the down-scaled image. Sun *et al.* [19] have proposed a novel spatial interpolation method that reconstructs a high-resolution (HR) image by collaging the patches of its low-resolution (LR) observation. Compared to those structured sparse representation-based methods such as [3], [18], Sun's method is very efficient and does not require to solve complex optimization equations. Recently, there have been developments of powerful example-based or learning-based methods that can reproduce fine texture details in images. The methods of [1], [2], [17], [26] utilize an external database of images to learn a mapping from LR patches to their HR counterparts.

B. IMAGE COMPRESSION

Data compression techniques have been widely used to reduce the costs on storage and transmission. There are a large number of methods to compress images, including Discrete Wavelet Transform [13], Symbolic Mapping [14],

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Asikuzzaman.

Histograms [15], Piecewise Linear Approximation (PLA) [8], [10], [24] etc. Due to its simplicity and visual intuitiveness, PLA is still regarded as one of the most widely used methods [21]. Its basic idea is to represent a series of pixels with a number of straight line segments so that the original pixels can be satisfactorily approximated.

In literatures, most of the early research[1] focus on constructing a pre-defined (or minimized) number of line segments under minimized (or pre-defined) holistic error bound between the original pixels and its representations respectively. However, these holistic error bound (i.e., L_2 norm) approaches are incapable of generating error-guaranteed representations on each individual pixel and can be inefficient in obtaining error-guaranteed analytical results for many applications [4], [13]. In regard to this, many of the recent research focus on constructing a minimum number of line segments under a pre-defined maximum-error between the original pixels and its representations, which is denoted as L_∞-bound PLA. For instance, Liu *et al.* [10] proposed an FSW (Feasible Space Window) algorithm to construct segments from a fixed initial point. Qi *et al.* [16], [22] extended FSW to the polynomial functions in the processing of multidimensional data. Xie *et al.* [21] gave an optimal linear-time algorithm that constructs minimum number of disconnected line segments through maximally extending each ''local'' segment forward. Zhao *et al.* [24], [25] proposed new algorithms that construct connected and optimal semi-connected line segments by optimizing two adjacent ''local'' segments only.

In this article, different from the most popular interpolation based methods on image scaling nowadays, we study how to use the line-segmentation and PLA techniques for image scaling which is yet unknown in current literature. In our proposed framework, the pixels in an image are linearized and represented by straight line segments either precisely or under maximum-error bound (L_∞ norm) on a small error tolerance (for example, $\delta = 0.3$). This guaranteed quality on errors would subsequently pass to the up-scaled image without requiring any manual adjustments such as interpolation, denoising and deinterlacing. For small error like $\delta < 0.5$, this framework losslessly compresses images. This is because that each pixel is an integer and can be decoded exactly. As such, the constructed scaled images from this framework possess desired properties such as *up-stability* and *topological down-consistency*. A pair of up-scaling and down-scaling algorithms are called up-scaling stable if, for any input image, after up-scaling and then down-scaling to its original size, the output image is exactly the same as the input image. Furthermore, a scaling algorithm is called topological down-consistency if it transforms each *representing segment* of the input image into a *representing segment* of the down-scaled output image. These two properties either directly or indirectly imply that the scaled images are very similar to the original ones. This has been confirmed in our experiments in terms of high peak signal-to-noise ratio (PSNR) and high structural similarity (SSIM).

To the best of our knowledge, no other linear-time image scaling algorithms with these two properties have yet to be reported. Our proposed image scaling algorithms have linear-time complexity with extremely low memory requirements. The effectiveness of the algorithms has been demonstrated on several test cases yielding results that, in practice, possess a low appearance of artifacts.

Along with the high efficiency in both execution and quality, the proposed methods also revealed that scaling operations can be realized directly from compressed data (i.e., segments), which can be significantly smaller than the total number of original data points and thus can be more efficiently processed. This indicates that data compression technique can efficiently support data analysis (operations) in addition to the data storage reduction and data transmission acceleration.

The rest of this article is organized as follows. Section 2 introduces our methods and the proposed algorithms. Section 3 describes the properties of the scaled images from the proposed framework. Section 4 reports our experiment results. Section 5 concludes this article.

II. ALGORITHMS

In this section, we first explain two fundamental algorithms which take pixels of a row (or column) and represent them in line segments in Section II-A. This is followed by a discussion on the general idea of image scaling in Section II-C.

A. SEGMENTATION

The two segmentation algorithms, Linearization (Function 1) and OptimalPLR (Function 2) are displayed in Figure 1(I) where OptimalPLR is adopted from [21]. As depicted in Figure 1(III), Linearization and OptimalPLR are called by NaiveScale and PLAscale respectively for image scaling.

The difference between Linearization and OptimalPLR is their methods in generating line segments: Linearization can quickly construct segments by linearizing pixels while OptimalPLR builds line segments under L_∞ norm tolerant. Mathematically, OptimalPLR and Linearization can be defined formally [21] as follows.

Given an array of pixels $P = (p_1, \ldots, p_n)$ and an error bound δ, OptimalPLR is to divide P and build k segments $S_1, S_2, \ldots, S_k$; where each segment $S_i = (p_{s_i}, p_{s_i+1}, \ldots, p_{e_i})$ ($i \in \{1, 2, \ldots, k\}$, $p_{s_1} = p_1$, $p_{e_i+1} = p_{s_{i+1}}$ and $p_{e_k} = p_n$) can be approximated by a linear function $f_i(t)$ satisfying the pre-defined error bound of

$$|f_i(t) - p_t| \leq \delta \tag{1}$$

for $t \in \{s_i, s_i + 1, \ldots, e_i\}$ with minimized k value. That is, OptimalPLR constructs the minimum number of segments for P where each segments satisfies Equation 1. Interestingly enough, it comes to Linearization when $\delta = 0$.

OptimalPLR constructs desired segments through incrementally maintaining two minimized convex hulls when

[1] including the most existing commercial compression software such as JPEG, JPEG2000, etc.

Function 1：Linearization(P)

Input: A time series P = (p_1,p_2,⋯,p_n)
Output: Segmentation points S
1: Push p_1 into S
2: Let *Lcurrent* be the line of p_1 and p_2
3: **while** (not finished P) **do**
4: **if** (check whether *Lcurrent passes the* coming point p_i) **then**
5: Continue
6: **else**
7: Push p_{i-1} into S
8: Reset *Lcurrent* created by p_{i-1} and p_i
9: Continue
10: **end if**
11: **end while**

Function 2：OptimalPLR(P)

Input: A time series P = (p_1,p_2,⋯,p_n) and a fixed error bound δ
Output: Segmentation points S
1: Let *Ulcurrent* be the line of (1,$p_{1.y}$- δ) and (2,$p_{2.y}$+ δ),*Llcurrent* be the line of (1,$p_{1.y}$+ δ) and (2,$p_{2.y}$- δ)
2: Set t_{start}=$p_{1.x}$
3: **while** (not finished P) **do**
4: **if** (check whether *Ulcurrent and Llcurrent approximate the* coming point p_i with error bound δ) **Then**
5: Update *Ulcurrent* and *Llcurrent* by ‘Convex-hulls’ method[Xie]
7: Continue
7: **else**
8: Denote the points of *Ulcurrent* at t_{start} and $p_{i.x}$ as s_i and $s_{i+1.}$
Push s_i and s_{i+1} into S
9: Reset *Ulcurrent, Llcurrent* created by ($p_{i.x}$,$p_{i.y}$- δ) and ($p_{i+1.x}$,$p_{i+1.y}$+ δ), ($p_{i.x}$,$p_{i.y}$+ δ) and ($p_{i+1.x}$,$p_{i+1.y}$- δ)
10: Update t_{start}=$p_{i.x}$
11: Continue
12: **end if**
13: **end while**

(I) Two Functions on compression and segmentation

Policy I（stretch segments）

Input: Segmentation points S of P = (p_1,p_2,⋯,p_n) and ratio m
Output: the extended series P´= (p'_1,p'_2,⋯,p'_{n*m})
1: Push out the first point $s_{current}$ from S
2: **while** (not finished S) **do**
3: Push out the next point s_{next} from S
4: **if** ($s_{current.x}$=1) **then**
5: Set the value of points from time tag 1 to m as p_1
6: **end if**
7: **if** ($s_{next.x}$-$s_{current.x}$=1) **then**
8: Set the value of points from time tag $s_{current.x}$*m+1 to $s_{next.x}$*m as p_{next}
9: **else**
10: Construct a line *Lcurrent* by ($s_{current.x}$*m,$s_{current.y}$) and ($s_{next.x}$*m,$s_{next.y}$)
11: Intersect the value of points from time tag $s_{current.x}$*m+1 to $s_{next.x}$*m by *Lcurrent*
12: **end if**
13: Reset $s_{current}$=s_{next}
14: **end while**

Policy II（stretch segments）

Input: Segmentation points S of P = (p_1,p_2,⋯,p_n) and ratio m
Output: the extended series P´= (p'_1,p'_2,⋯,p'_{n*m})
1: Push out the first point $s_{current}$ from S
2: **while** (not finished S) **do**
3: Push out the next point s_{next} from S
4: **if** ($s_{current.x}$=1) **then**
5: Set the value of points from time tag 1 to m as p_1
6: **end if**
7: Construct a line *Lcurrent* by ($s_{current.x}$*m,$s_{current.y}$) and ($s_{next.x}$*m,$s_{next.y}$)
8: Intersect the value of points from time tag $s_{current.x}$*m+1 to $s_{next.x}$*m by *Lcurrent*
9: Reset $s_{current}$=s_{next}
10: **end while**

Policy I or II（shrink segments）

Input: Segmentation points S of P = (p_1,p_2,⋯,p_n) and ratio m
Output: the shrunk series P´= (p'_1,p'_2,⋯,$p'_{n/m}$)
Description:
Resampling from P to construct P´, i.e, p'_1=p_1, $p'_{n/m}$=p_n and p'_i=p_{i*m} (1<i<n/m)

（II）Two Policies on stretching or shrinking segments (in ratio m)

Alg 1.1：NaiveScale(Policy I)

INPUT： image I
OUTPUT: image O

Up-scaling:
1. Use Function 1 to construct segments on each row of image I
2. Use Policy I to stretch each segment and get image I’
3. Use Function 1 to construct segments on each column of image I’
4. Use Policy I to stretch each segment;
5. Output image O

Down-scaling:
1. Use Function 1 to construct segments on each column of image I
2. Use Policy I to shrink each segment and get image I’
3. Use Function 1 to construct segments on each row of image I’
4. Use Policy I to shrink each segment;
5. Output image O

Alg 1.2：NaiveScale(Policy II)
By replacing Policy I with Policy II in Alg 1.1

Alg 2.1：PLAscale(Policy I)

INPUT： image I
OUTPUT: image O

Up-scaling:
1. Use Function 2 to construct segments on each row of image I
2. Use Policy I to stretch each segment and get image I’
3. Use Function 2 to construct segments on each column of image I’
4. Use Policy I to stretch each segment
5. Output image O

Down-scaling:
1. Use Function 2 to construct segments on each column of image I
2. Use Policy I to shrink each segment and get image I’
3. Use Function 2 to construct segments on each row of image I’
4. Use Policy I to shrink each segment
5. Output image O

Alg 2.2：PLAscale(Policy II)
By replacing Policy I with Policy II in Alg 2.1

(III) Scaling algorithms

FIGURE 1. The proposed algorithms.

a new point is added in. To avoid extensively searching throughout the entire convex hull, this algorithm only requires checking two points to decide whether or not the current convex hull needs to be updated. OptimalPLR achieves the minimum number of line segments (optimal output) through maximally extending each ‘‘local’’ segment forwardly.

In order to adjust a line segment to approximate the maximum number of pixel points, the algorithm determine the range of all feasible line segments, which is incrementally maintained during the processing of consecutive sequence points. Whenever the current point cannot be approximated within the error bound, the algorithm begins to construct a new segment.

As declared in [21], OptimalPLR is a linear-time online algorithm and has demonstrated better performances than many other similar algorithms in terms of time and space costs on the extensively tested data sets.

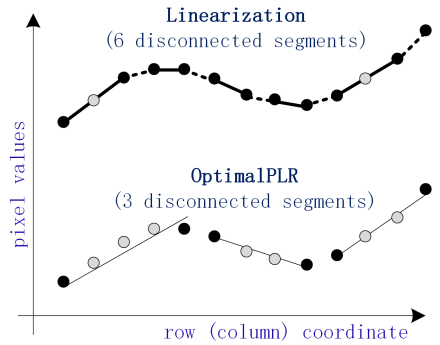


FIGURE 2. Segments from Linearization and OptimalPLR.

Example 1: Figure 2 displayed the segments built from Linearization and OptimalPLR respectively. Linearization requires that each constructed segment should go through every pixel point located in the region while OptimalPLR allows a δ tolerance on the distances between pixel points and segments. As such, OptimalPLR generates less number of segments than that of Linearization.

In terms of functionality, OptimalPLR is more flexible on error tolerance and Linearization is more efficient in execution. In reality, both NaiveScale and PLAscale achieve very similar high quality image scaling for $\delta < 0.5$. However, PLAscale does offer better memory efficiency.

B. METHOD

In comparison to the kernel based image scaling methods where new pixels are constructed from adjacent pixels on up-scaling, NaiveScale and PLAscale interpolate by stretching representing segments. Our intuition for the proposed techniques follows: under a small maximum-error bound, a line segment representing the pixels in the original image can be stretched or shrunk to represent the pixels in the scaled images. Through this, we can represent significantly larger patterns using simple structures with high accuracy and reduced storage simultaneously. It should be noted that the line segments constructed via L_2-bound (on holistic error) PLA algorithms may not work well as it does not guarantee the approximate error at each individual data point [5], [11], [24]. Furthermore, for simplicity, only scaled imaged of integer multiples are considered within this article.

Scaling an image to a plausible larger or smaller image needs to stretch and shrink segments in a proper way. For the visual effects of the scaled images, two policies for the stretch and shrink operations are proposed. As depicted in Figure 1(II) and Figure 3, the major difference between Policy I and Policy II is that Policy I does not (overly) smooth the pixels of two adjacent segments like that of Policy II. Policy I and II exemplify the two most common encountered interpolation situations in image scaling. In applications, Policy I is preferred for binary images and Policy II is much more suitable for grey scaled and other images.

C. SCALING

For NaiveScale and PLAscale, the general steps for up-scale and down-scale operations can be illustrated in Figure 1(III) and Figure 4. In the process of up-scaling, both proposed algorithms first segment the pixels for each row of the input image with the respective functions of Linearization and OptimalPLR (Figure 1(I)). The algorithms then stretch the formed segments under the desired policy (Figure 1(II)) and ratio to obtain a row-enlarged image. In this new image, the above segmentation and stretching processes are then performed for the columns of the row-enlarged image to eventually obtain the enlarged image. In the process of down-scaling, we conventionally suggest performing the segmentation and shrinking processes on columns first and then on rows for the consistency of consequently applying up-scaling and then down-scaling (or down-scaling and then up-scaling) operations.

Under this framework, we only need to store and transmit the set of segments on rows (or columns) when transmitting images for up-scaling (or down-scaling) respectively. As such, the data shipping cost is reduced.

III. PROPERTIES

The core procedures in the proposed scaling algorithms are Linearization and OptimalPLR which allow an input image to be partially loaded into memory and processed progressively on row (or column) segmentations. As a result, the proposed scaling algorithms can potentially manipulate very large images with low requirements on memory size. These algorithms can be used to scale video, 3D object data and be efficiently parallel-processed. In terms of time costs, since each algorithms of Figure 1(I-II) has a linear time complexity, the proposed scaling algorithms of Figure 1(III) have linear time costs on the number of input and output pixels.

In the following, we use two concepts to characterize the quality of scaling algorithms and their practical advantages: *Stability* and *Topological Consistency*. We indicate that the proposed algorithms are *up-stable* and *down-consistent*.

Let I_{input} and I_{output} be an input image and the manipulated image from the input respectively. Let $z = f(x, y)$ be the input image I_{input} where $x \in [0, m]$ and $y \in [0, n]$. Suppose we want to scale it $w \times h$ (or $1/w \times 1/h$) times in width and height. For simplicity, we only consider that w and h are integer values in this article. Mathematically, the scaled image $z = g(x, y)$ can be expressed as $g(x, y) = f(x/w, y/h)$ with $x \in [0, w * m]$ and $y \in [0, h * n]$ for $w \times h$ times scaling.

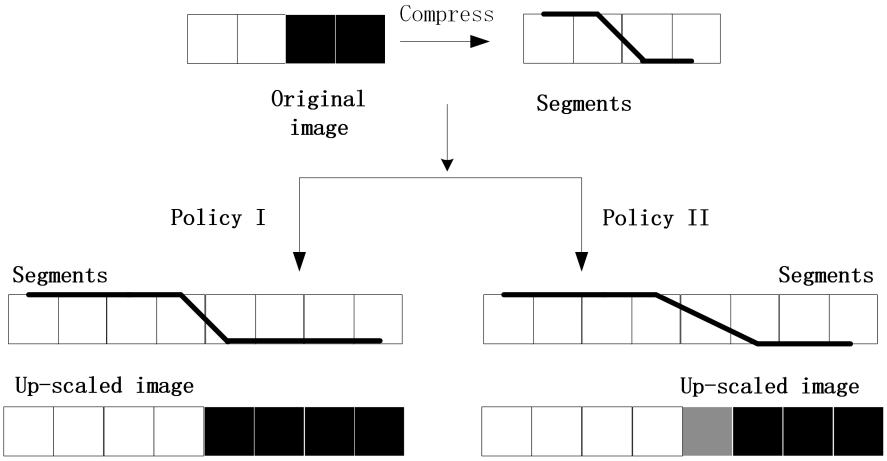


FIGURE 3. Two policies on stretching segments (in ratio 2).

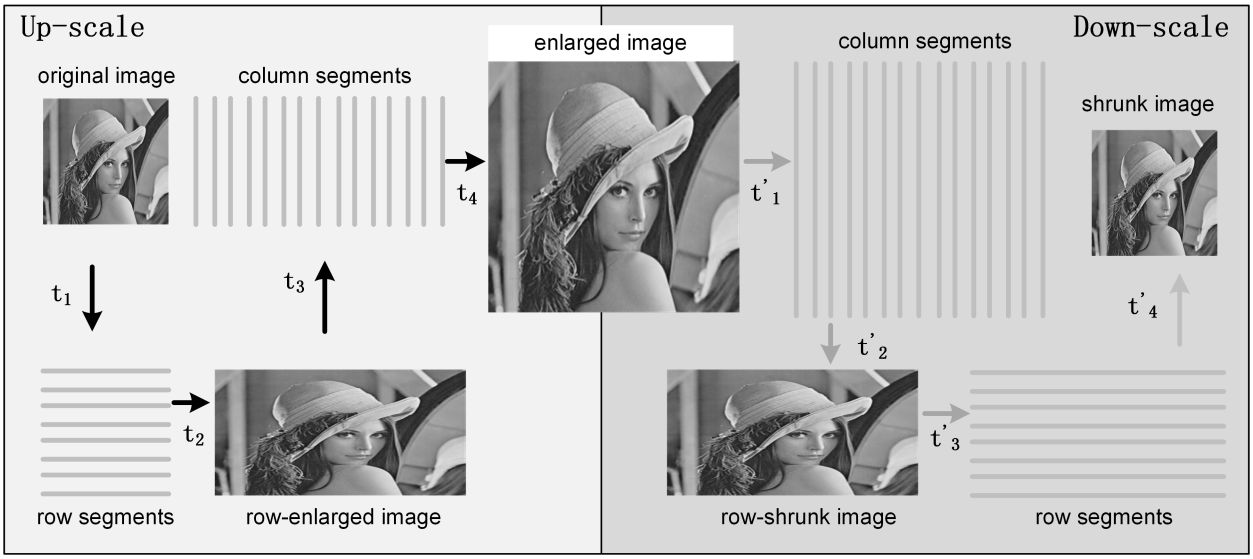


FIGURE 4. Up-scaling and down-scaling processes.

A. STABILITY ON SCALING

Let $I_{up-down}$ be the output image which is constructed by first up-scaling I_{input} into I' and then down-scaling I' into the same size of I_{input}.

A pair of up-scaling and down-scaling algorithms is called *up-stable* if $I_{up-down} = I_{input}$ holds for any I_{input} under any integer zooming ratio.Similarly, $I_{down-up}$ and *down-stable* can be defined.

In the up-scaling process, all the proposed algorithms first stretch the row segments and then stretch the column segments. Alternatively, in the down-scaling process, all the proposed algorithms shrink in the reverse order; shrinking the column segments first and then shrinking the row segments. Therefore, $p_{x,y} = p'_{x',y'}$ holds for any pixel $p_{x,y} \in I_{input}$ and its corresponding mapped pixel $p'_{x',y'} \in I_{up-down}$ as each pixel $p_{x,y} \in I_{input}$ would not vanish through the scaling process, implying that this pair of proposed up-scaling and down-scaling algorithms is up-stable.

It should be noted that our proposed algorithms may not preserve down-stability as some segments may be completely removed in the down-scaling stage. These removed segments may not be recoverable in the afterwards up-scaling stage.

B. TOPOLOGICAL CONSISTENCY

This property measures if any "representing" line segments in the original image can be preserved in the scaled image. Let $z = sg(q_1, q_2)$ or $z = sg(x, y)$ be a straight line segment with q_1 and q_2 as the two end points. Let $D_{XY}(sg)$ be the domain of $sg(x, y)$ in the XY-plane. Segment $sg_f(x, y)$ of I_{input} is called a δ-representing segment (or simply, δ-segment) if $|q_{x,y} - p_{x,y}| < \delta$ holds for any $q_{x,y}$ of $sg_f(x, y)$ and $p_{x,y}$ of I_{input} where $(x, y) \in D_{XY}(sg_f)$. It should be noted that

the pixels of q_1 and q_2 on the δ-representing segment ($z = sg_f(q_1, q_2)$) may not be pixels of I_{input}. The definition implies that $|q_{x_1,y_1} - p_{x_1,y_1}| < \delta$ and $|q_{x_2,y_2} - p_{x_2,y_2}| < \delta$ where $q_1 = q_{x_1,y_1}$, $q_2 = q_{x_2,y_2}$ and $p_{x_1,y_1}, p_{x_2,y_2} \in I_{input}$.

Suppose that a scaling algorithm transforms a δ-segment $z = sg_f(q_{x_1,y_1}, q_{x_2,y_2})$ of I_{input} into a segment $z' = sg_g(q_{x'_1,y'_1}, q_{x'_2,y'_2})$ of I_{output} where $(x'_1, y'_1) = (w*x_1, h*y_1)$ and $(x'_2, y'_2) = (w * x_2, h * y_2)$. A scaling algorithm is called *topological consistent* (or simply, *consistent*) if each δ-segment $z = sg_f(q_{x_1,y_1}, q_{x_2,y_2})$ of I_{input} is transformed into a δ-segment of I_{output}. Specifically, it is called *up-consistent* (or *down-consistent*) if each δ-segment of I_{input} is transformed into a δ-segment of any up-scaled (or down-scaled) image I_{output}. It can be seen that our proposed scaling algorithms are down-consistent as $q_{x',y'} = q_{x,y}$ holds for each point $q_{x',y'}$ of $z' = sg_g(q_{x'_1,y'_1}, q_{x'_2,y'_2})$ and point $q_{x,y}$ of $z = sg_f(q_{x_1,y_1}, q_{x_2,y_2})$ where $x' = x/w$ and $y' = y/h$.

IV. EXPERIMENTS

In the following, we demonstrate the advantages of our proposed algorithms against four common interpolation kernel based methods: NearestNeighbor, Bilinear, Bicubic and Lanczos methods. All the tested algorithms are implemented in Eclipse with C++ and use the Open Source Computer Vision Library (OpenCV). All the experiments are performed on a laptop with CPU of Intel Core i7-5500U 2.40GHz and 12G memory. The error bound of PLAscale method is set to $\delta = 0.4$.

We have tested more than 50 images for the proposed algorithms in terms of compressed storage, time efficiency and visual quality. Because of the space limitations and the similarities on the tested results, we only exemplified a few in this article.

TABLE 1. The compressed storage.

Image name	Original storage	Compressed storage ($\delta = 0$)	Compressed storage ($\delta = 0.4$)
Lenna	65536	61194	58594
Boat	32000	29424	27509
Butterfly	21600	19570	18105
Character	40000	**4814**	**3789**
Man	40000	38917	38287
Monkey	38400	37757	37403
QRCode	16384	9688	8767
River	40000	38189	37014
Vegetable	40000	37728	36420
Bird	2073600	**1125513**	**790280**
City	921600	613097	525526
Mountain	3686400	3119437	2859261

A. COMPRESSED STORAGE

In Figure 1(I), OptimalPLR constructs line segments in linear time and low memory costs. It can compress images losslessly when the error bound is set to a value less than 0.5. The Linearization is a special case of OptimalPLR on $\delta = 0$. The storage costs on different images are list in Table 1. From this table, we can see:

- In general, both OptimalPLR and Linearlization can effectively reduce the storages of original images, about 10% and 5% repsectively.
- This reduction is more effective for binary or larger images. For instance, the compressed storages of "Bird" by OptimalPLR are less 50% than the original storage.

In addition, the compression storages of OptimalPLR will be decreased as the error bounds increase. For example, the compressed storages on "Lenna" from PLAscale are 58594, 47104, 36574, 23665 and 16022 under the error bound of 0.4, 1, 2, 5, and 10 respectively.

B. TIME COSTS

Figure 5 lists the time costs of various methods on up-scaling image "Lenna" of 256*256 pixels to different sizes. We have the following observations on up-scaling:

- In PLAscale and NaiveScale, the time costs on compression have little impact on the total time consumption. Furthermore, these two proposed algorithms have a very similar performance on both Policy I and II.
- NearestNeighbor, Bilinear, Bicubic and Lanczos are less efficient than that of PLAscale (I and II) and NaiveScale (I and II), especially on a larger scale. The time costs of interpolation methods increase much more than that of PLAsacle and NaiveScale. For example, the time costs for 15 times up-scaling are 0.655s, 0.965s, 1.729s, 6.288s, 0.497s, 0.472s, 0.469s and 0.469s respectively. The time differences are due to their different indwelling interpolation mechanisms. Bilinear, Bicubic and Lanczos reconstruct each new pixel by considering its 4,16 and 64 neighbor pixels respectively, while the proposed methods only use two points to generate the representation lines for interpolating the new pixels.

For an original image "Lenna", the compression time costs of NaiveScale and PLAscale are about 3ms and 4ms, which are independent on scaled sizes. In fact, the time cost of OptimalPLR is very low, with about 1 microsecond for each data point [21]. The cost of Linearlization is much lower than that of OptimalPLR since it does not need to compute the "convex hull" required for the implementation of OptimalPLR.

Unfortunately, the down-scaling performance of our methods is more expensive than that of interpolation methods Our intuitive explanations are as follows.

Let t be the time cost of scaling a $w*h$ image into $m*n$ times magnification and t' be the time cost of scaling it back into the original size. In a interpolation method, t is about $m * n$ times of t'. Therefore, the time cost on up-scaling is much higher than that of down-scaling in a interpolation method.

For our proposed methods, as depicted in Figure 4, the time cost consists of four parts: the cost of column (or row) compression, the cost of column (or row) scaling, the cost of row (or column) compression and the cost of row (or column) scaling, which are briefly denoted as t_1, t_2, t_3, t_4 for up-scaling and t'_1, t'_2, t'_3, t'_4 for down-scaling. It can see that t'_1 is about $m*n$ times of t_1. All the others, t'_2, t'_3 and t'_4 are quite similar to t_2, t_3 and t_4 respectively. As a result, the time cost for down-scaling is more expensive than that of up-scaling for our methods.

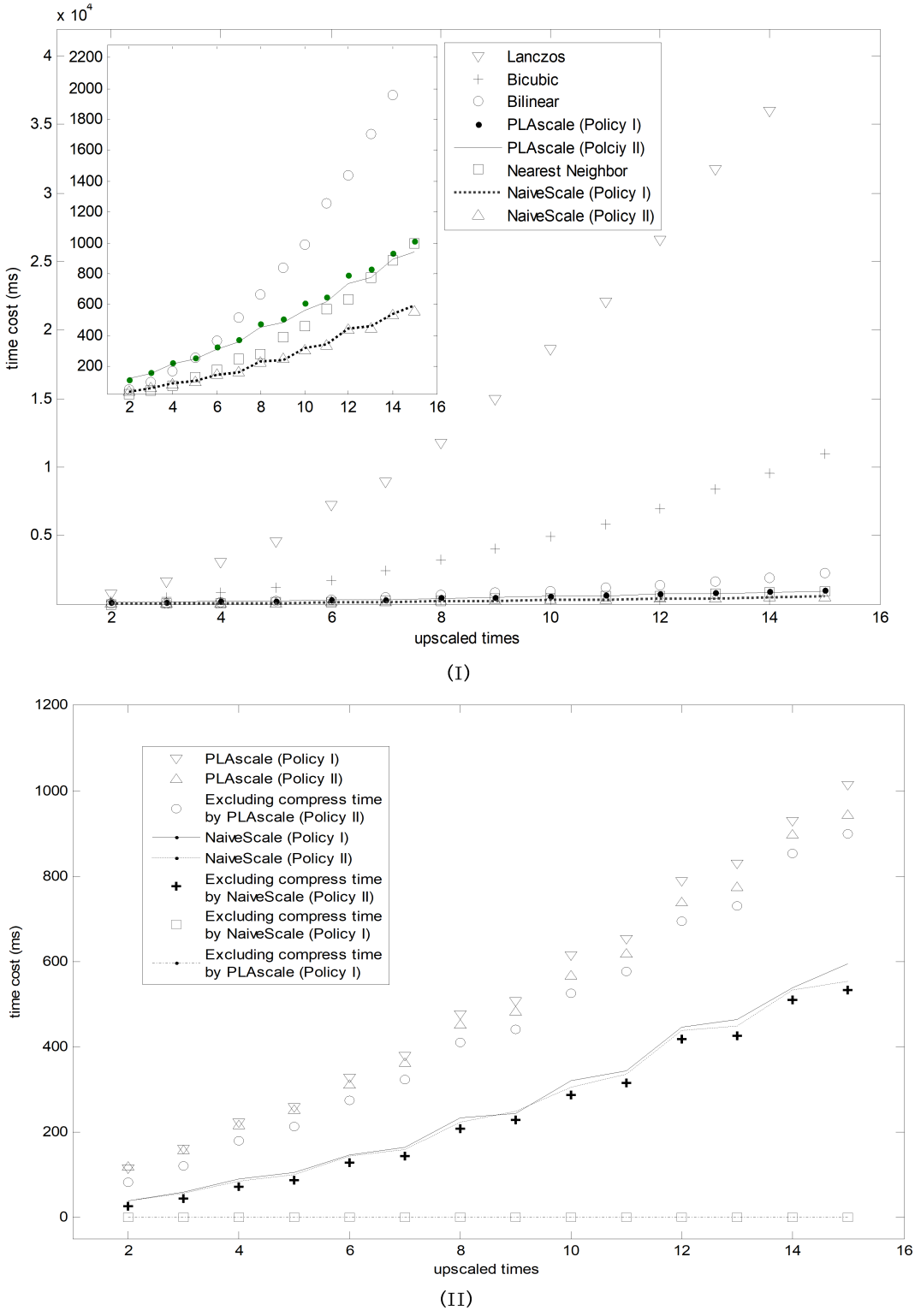


FIGURE 5. Time costs including compression costs.

C. VISUAL QUALITY

We would compare the scaled visual qualities for binary and non-binary images seperately.

For binary images, the quality comparisons on up-scaling are listed in Figure 6. Figure 6(I) illustrates the results of up-scaling of an original image (a) 10 times from 8 different

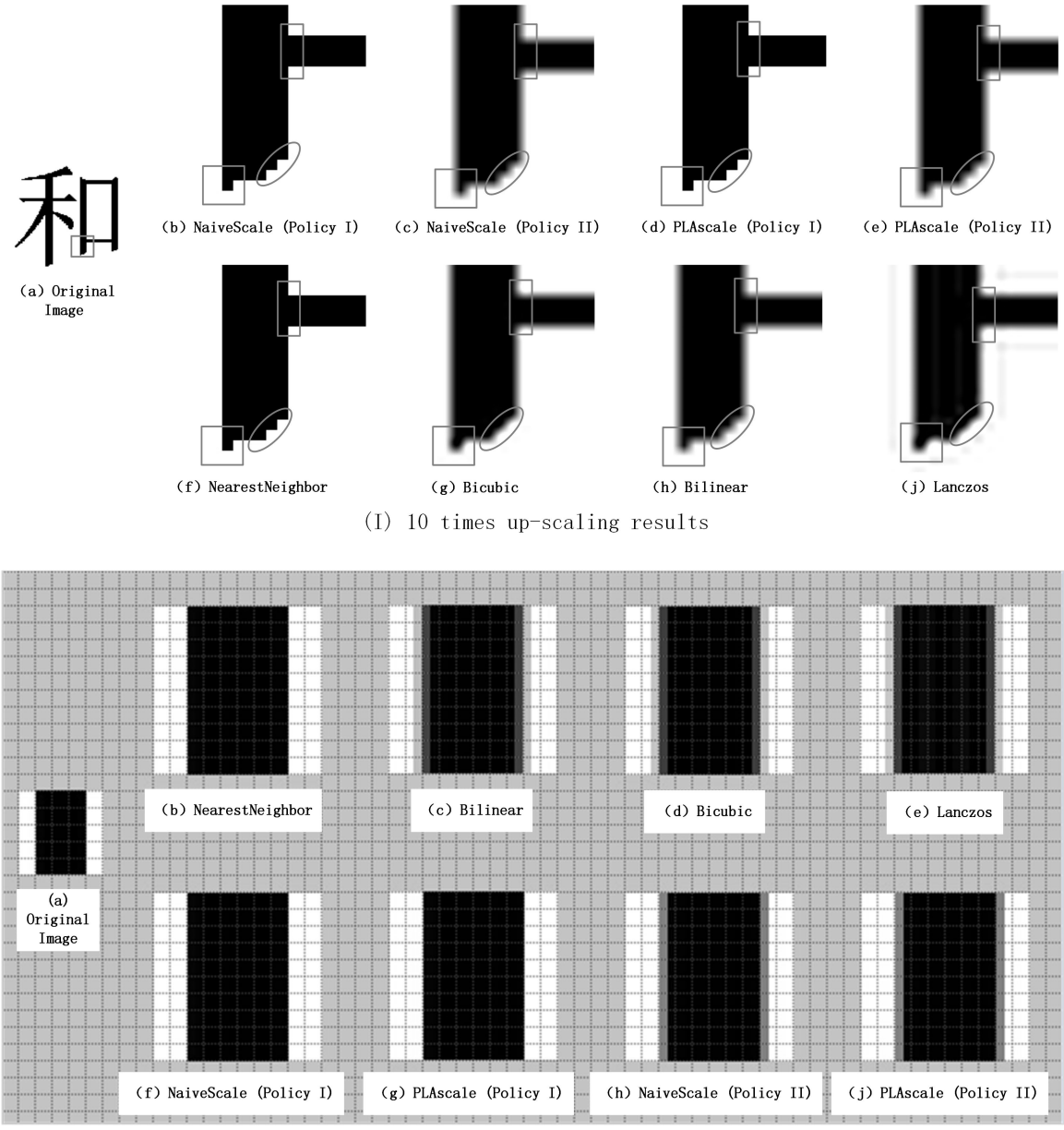


FIGURE 6. **Binary Image (a) and its up-scaling results.**

methods. Intuitively, both NaiveScale (b) and PLAscale (d), using Policy I, can successfully upscale the binary image and achieve a similar up-scaling quality as that of Nearest Neighbor (f). All others, (c), (e), (g), (h) and (j), cannot maintain the original brightness level. As seen in Figure 6(I), Bicubic (g) and Lanczos (j) have changed the structural features of the original image, particularly at the corners and the spike areas but Lanczos (j) generates an additional grey shape around the image. Furthermore,

- As shown in Figure 6(I)(g) marked by rectangles, the black area is invaded by white and grey at the corner;
- As shown in Figure 6(I)(g) and (j) marked by ellipses, the black jagged edge is smoothed;
- As shown in Figure 6(I)(g) and (j) marked by squares, the black spike area blurred and rounded.

In Figure 6(II), the impacts of various methods upon the Nearest Neighbor method were examined to indirectly compare the visual quality. For the original image (a) of 10*10 pixels, we construct the images of Figure 5(b to j) by first up-scaling (a) two times with the listed methods and then using Nearest Neighbor to upscale each of them five times again. We have the following observations:

TABLE 2. Measurements on down-up scaling.

PSNR SSIM	Bilinear	Bicubic	Lanczos	Nearest Neighbor	NaiveScale (Policy I)	NaiveScale (Policy II)	PLAscale (Policy I)	PLAscale (Policy II)
Bilinear	34.647 0.876	35.420 0.901	35.479 0.902	34.538 0.868	34.537 0.871	33.771 0.841	34.584 0.871	33.887 0.841
Bicubic	35.248 0.895	**35.623** **0.908**	35.427 0.901	34.508 0.868	34.576 0.868	33.886 0.846	34.574 0.872	33.992 0.846
Lanczos	35.182 0.889	35.558 0.900	35.302 0.892	34.588 0.863	34.665 0.868	33.929 0.840	34.663 0.867	34.029 0.840
Nearest Neighbor	34.110 0.839	33.960 0.838	33.745 0.830	34.247 0.793	34.280 0.796	32.481 0.716	34.271 0.795	32.557 0.716
NaiveScale (Policy I)								
NaiveScale (Policy II)	33.877	33.805	33.636	34.134	34.243	**35.834**	34.271	**35.988**
PLAscale (Policy I)	0.841	0.844	0.838	0.801	0.806	**0.898**	0.807	**0.898**
PLAscale (Policy II)								

TABLE 3. Measurements on up-down scaling.

PSNR SSIM	Bilinear	Bicubic	Lanczos	Nearest Neighbor	NaiveScale (Policy I)	NaiveScale (Policy II)	PLAscale (Policy I)	PLAscale (Policy II)
Bilinear	39.274 0.969	41.983 0.978	42.085 0.978	37.320 0.954	37.637 0.963	37.637 0.963	37.637 0.963	37.637 0.963 0.963
Bicubic	43.628 0.984	46.567 0.987	47.556 0.986	36.996 0.956	37.259 0.966	37.259 0.966	37.259 0.966	37.259 0.966
Lanczos	44.114 0.984	46.771 0.987	47.665 0.987	37.120 0.955	37.407 0.964	37.407 0.964	37.407 0.964	37.407 0.964
Nearest Neighbor	46.603 0.987	40.064 0.977	38.628 0.969	45.665 0.984	50.241 0.995	50.241 0.995	50.241 0.995	50.241 0.995
NaiveScale (Policy I)	46.666 0.987	39.915 0.975	38.212 0.967	45.197 0.984	**INF** **1**	**INF** **1**	**INF** **1**	**INF** **1**
NaiveScale (Policy II)	37.034 0.955	37.708 0.965	37.944 0.968	34.291 0.879	**INF** **1**	**INF** **1**	**INF** **1**	**INF** **1**
PLAscale (Policy I)	46.442 0.988	39.924 0.975	38.225 0.967	44.643 0.983	**INF** **1**	**INF** **1**	**INF** **1**	**INF** **1**
PLAscale (Policy II)	37.278 0.955	37.924 0.965	38.183 0.968	34.545 0.878	**INF** **1**	**INF** **1**	**INF** **1**	**INF** **1**

- All of the three methods, NaiveScale (Policy I) (f), PLAscale (Policy I) (g) and NearestNeighbor (b), are very similar to the original image (a): Both the white and black areas are stretched 2 times in both horizontally and vertically;
- The other methods can overly amplify the original image size than the proposed methods. For example, the width of NaiveScale(Policy II) (h) and PLAscale(Policy II) (j) is 6.5 cells, while Bilinear (b), Bicubic (c) and Lanczos (d) reach 7 cells. NaiveScale and PLAscale under Policy II possess lower appearances of artifacts than that of Bilinear, Bicubic and Lanczos. Additionally, it turns out that our algorithms can be more accurate and have fewer artifacts than that of Bilinear (b), Bicubic (c) and Lanczos (d).

For non-binary images, we show visual results of 2 times up-scaling "Lenna" in Figure 7, 8 and 9 on an error bound of 0.4. As shown in Figure 8 and 9, the up-scaled image in Policy II has a higher overall visual quality than that of in Policy I. In our view, Figure 8(II) and Figure 9(II) are very similar. It is suggested that (1) uses a smaller max-error bound (<0.5) for high quality image; and (2) uses NaiveScale for a faster process but with some increased storage.

D. QUALITY MEASUREMENT

For further illustration of the differences between the kernel based interpolations and the proposed methods, we use higher peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) to indirectly measure the performance qualities and confirm our above observations. Our experiments are conducted on eight methods (i.e., Bilinear, Bicubic, Lanczos, NearestNeighbor, NaiveScale and PLAscale (in both Policy I and Policy II)) under the downscale/upscale operations. We adopt the common used image "Lenna" of 256*256 pixels as the original image. The referee images are the images that has been downscaled/upscaled from the original 2 times through one scaling algorithm then scaled back through the same (or another) algorithm. Table 1 and Table 2 are the measurements on scaling-stability. For example, the PSNR and SSIM at cell (Lanczos, Bicubic) of Table 2 (Table 3) are derived from the original image against the image that was first down-scaled (up-scaled) by Lanczos and

(I) The original image with 256*256 pixels

(II) 2 times upscaling Image by Bicubic

FIGURE 7. The original image (I) and its two times up-scaling results (II) by Bicubic.

then up-scaled (down-scaled) by Bicubic to its original size respectively.

From these two tables, the following observations were made:

1) In these two tables, the better results (marked in bold) depend on which methodology is used. The best results in these two tables are generated from our proposed algorithms. This is due to the differing ideologies

(I) 2 times up-scaling by NaïveScale (Policy I)

Storage: 61194.

(II) 2 times up-Scaling by NaïveScale (Policy II)

FIGURE 8. Two times up-scaling results by NaiveScale.

behind the two types of techniques. The pixels of scaled images from convoluted interpolations are generated from its neighboring pixels, while our methods reconstruct the scaling pixels from the compressed representation lines.

2) Either way, results of the up-down scaling process (Table 3) are better than the ones of the down-up (Table 2). The down sampling process may lose some information that cannot be recoverable, due to its reduction of the original.

(I) 2 times up-scaling by PLAscale (Policy I) under max-error 0.4

Storage: 58594.

(II) 2 times up-scaling by PLAscale (Policy II) under max-error 0.4

FIGURE 9. Two times up-scaling results by PLAscale.

3) In the above "down-up" process, the quality measurements at each cell would drop when the scaling increases. The drop trends of our methods are a little more serious than that of interpolation methods. For example, bicubic performed relatively well. However, the results on our algorithms could be further improved

by adding some heuristic strategies into the down-scaling process. For example, in the down-scaling process, we may not discard the representation lines when the number of its covering pixels is smaller than the down-scaling factors. Those lines may contain the jumping grey information which may be characteristics of the original image.

4) As shown in Table 3, when the error bound is smaller than 0.5, our algorithms is up-stable. This property is derived from the fact that the compressed results by PLA with smaller error bound (less than 0.5) is a lossless compression for images.

Through a large number of experiments, our general observations of our methods can be summarized as follows:

- On compressed storage, the OptimalPLR is an effective and efficient algorithm for compressing binary or non-binary images. Especially, when the max error bound is less than 0.5, this compression is lossless.
- On time cost, the performance of our proposed algorithms outperform that of interpolation methods on up-scaling.
- On visual quality, both NaiveScale and PLAscale achieve very similar high quality results efficiently: NaiveScale (Policy I) is preferred for binary images and PLAscale (Policy II) is much more suitable for grey scaled images. Compared with interpolation methods, the results of Policy I (with NaiveScale or PLAscale) is similar to that of Nearest Neighborhood, and NaiveScale (or PLAscale) with Policy II is not obviously different from Lanczos.
- On the quality measurements of PSNR and SSIM, the up-stable property of our algorithms guarantees the high quality up-scaled output image.

V. CONCLUSION

In this article, we have presented two new image scaling algorithms. These two algorithms have linear-time complexity with extremely low memory requirements, and can construct quality-guaranteed scaled images. The effectiveness of the algorithms has been demonstrated on several test cases yielding results that, in practice, possess a low appearance of artifacts. Two policies on segment stretching and shrinking operations have been proposed. In terms of applications, a better perceived visual result could be achieved through using both policies in a combined manner for image scaling. This would need to be studied in further detail. Our other future work would consider: (a) if the proposed algorithms satisfy the up-consistent property; and (b) how to design more efficient image scaling algorithms, specifically in terms of high image quality and compression rate. We will investigate the use of other max-error bound PLA algorithms such as the connected or semi-connected PLA of [24], [25] in the image processing for further space reductions.

ACKNOWLEDGMENTS

The pending patent based on this work: "Quality Assured Image Scaling Method: Linear Fitting Compression Algorithm Based on Maximum Error" (Patent Application No. 2017111237368).

The authors would like to thank the authors of [21] for their generously giving us their source code for the comparison tests in this article.

REFERENCES

[1] C. Dong, C. C. Loy, K. He, and X. Tang, *Learning a Deep Convolutional Network for Image Super-Resolution*. Cham, Switzerland: Springer, 2014, pp. 184–199.

[2] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[3] W. Dong, N. Zhou, J.-C. Paul, and X. Zhang, "Optimized image resizing using seam carving and scaling," *ACM Trans. Graph.*, vol. 28, no. 5, p. 125, Dec. 2009.

[4] M. Garofalakis and P. B. Gibbons, "Probabilistic wavelet synopses," *ACM Trans. Database Syst.*, vol. 29, no. 1, pp. 43–90, 2004.

[5] M. Garofalakis and A. Kumar, "Deterministic wavelet thresholding for maximum-error metrics," in *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, New York, NY, USA, Jun. 2004, pp. 166–176.

[6] W. Kang, J. Jeon, S. Yu, and J. Paik, "Fast digital zooming system using directionally adaptive image interpolation and restoration," *SpringerPlus*, vol. 3, no. 1, p. 713, 2014.

[7] J. Kopf, A. Shamir, and P. Peers, "Content-adaptive image downscaling," *ACM Trans. Graph.*, vol. 32, no. 6, p. 173, Nov. 2013.

[8] A. Koski, M. Juhola, and M. Meriste, "Syntactic recognition of ECG signals by attributed finite automata," *Pattern Recognit.*, vol. 28, no. 12, pp. 1927–1940, Dec. 1995.

[9] J.-G. Leu, "Sharpness preserving image enlargement based on a ramp edge model," *Pattern Recognit.*, vol. 34, no. 10, pp. 1927–1938, Oct. 2001.

[10] X. Liu, Z. Lin, and H. Wang, "Novel online methods for time series segmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1616–1626, Dec. 2008.

[11] G. Luo, K. Yi, S.-W. Cheng, Z. Li, W. Fan, C. He, and Y. Mu, "Piecewise linear approximation of streaming time series data with max-error guarantees," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 173–184.

[12] A. C. Öztireli and M. Gross, "Perceptually based downscaling of images," *ACM Trans. Graph.*, vol. 34, no. 4, p. 77, Aug. 2015.

[13] C. Pang, Q. Zhang, X. Zhou, D. Hansen, S. Wang, and A. Maeder, "Computing unrestricted synopses under maximum error bound," *Algorithmica*, vol. 65, no. 1, pp. 1–42, Jan. 2013.

[14] C.-S. Perng, H. Wang, S. R. Zhang, and D. S. Parker, "Landmarks: A new model for similarity-based pattern querying in time series databases," in *Proc. 16th Int. Conf. Data Eng.*, Mar. 2000, pp. 33–42.

[15] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 1996, pp. 294–305.

[16] J. Qi, R. Zhang, K. Ramamohanarao, H. Wang, Z. Wen, and D. Wu, "Indexable online time series segmentation with error bound guarantee," *World Wide Web*, vol. 18, no. 2, pp. 359–401, Mar. 2015.

[17] Y. Romano, J. Isidoro, and P. Milanfar, "RAISR: Rapid and accurate image super resolution," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 110–125, Mar. 2017.

[18] Y. Romano, M. Protter, and M. Elad, "Single image interpolation via adaptive nonlocal sparsity-based modeling," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 3085–3098, Jul. 2014.

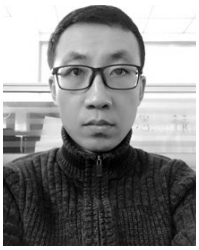
[19] D. Sun, Q. Gao, and Y. Lu, "Image interpolation via collaging its non-local patches," *Digital Signal Process.*, vol. 49, p. 33–43, Feb. 2016.

[20] N. Weber, M. Waechter, S. C. Amend, S. Guthe, and M. Goesele, "Rapid, detail-preserving image downscaling," *ACM Trans. Graph.*, vol. 35, no. 6, p. 205, Nov. 2016.

[21] Q. Xie, C. Pang, X. Zhou, X. Zhang, and K. Deng, "Maximum error-bounded piecewise linear representation for online stream approximation," *VLDB J.—Int. J. Very Large Data Bases*, vol. 23, no. 6, pp. 915–937, Dec. 2014.

[22] Z. Xu, R. Zhang, K. Ramamohanarao, and U. Parampalli, "An adaptive algorithm for online time series segmentation with error bound guarantee," in *Proc. 15th Int. Conf. Extending Database Technol.*, Mar. 2012, pp. 192–203.
[23] S. Yu, R. Li, R. Zhang, M. An, S. Wu, and Y. Xie, "Performance evaluation of edge-directed interpolation methods for noise-free images," in *Proc. 5th Int. Conf. Internet Multimedia Comput. Service*, New York, NY, USA, Aug. 2013, pp. 268–272.
[24] H. Zhao, Z. Dong, T. Li, X. Wang, and C. Pang, "Segmenting time series with connected lines under maximum error bound," *Inf. Sci.*, vol. 345, pp. 1–8, Jun. 2016.
[25] H. Zhao, C. Pang, R. R. Kotagiri, C. K. Pang, and T. Li, "An optimal piecewise linear approximation algorithm on semi-connected segmentation under maximum error bound," Zhejiang Univ., Hangzhou, China, Tech. Rep. 20170920, 2017.
[26] Y. Zhao, R.-G. Wang, W. Jia, W.-M. Wang, and W. Gao, "Iterative projection reconstruction for fast and efficient image upsampling," *Neurocomputing*, vol. 226, pp. 200–211, Feb. 2017.
[27] D. Zhou, X. Shen, and W. Dong, "Image zooming using directional cubic convolution interpolation," *IET Image Process.*, vol. 6, no. 6, pp. 627–634, Aug. 2012.

GENLANG CHEN received the Ph.D. degree in computer science from Zhejiang University, in 2012. He was a Visiting Scholar with the University of Arkansas. He is currently an Associate Professor with the Ningbo Institute of Technology, Zhejiang University. His research interests include big data processing and analysis, the Internet of Things technologies and applications, eHealth services, and parallel computing applications.



HUANYU ZHAO received the M.Sc. degree in applied mathematics from Hebei University, China, in 2009. He is currently an Associate Professor with the Institute of Applied Mathematics, Hebei Academy of Sciences, and also with the Hebei Authentication Technology Engineering Research Center, Shijiazhuang, China. His current research interests include stream data compression and processing, machine learning, evolutionary computation, and computing algorithm.

CHRISTOPHER KUO PANG is currently pursuing the degree with The University of Queensland. He is also pursuing the B.E. and B.Sc. degrees in electronics and computer sciences.

TONGLIANG LI received the Ph.D. degree in computer science from Tianjin University, China, in 2014. He is currently a Researcher with the Institute of Applied Mathematics, Hebei Academy of Sciences, and also with the Hebei Authentication Technology Engineering Research Center, Shijiazhuang, China. His current research interests include stream data compression and processing, information privacy, and data mining.

CHAOYI PANG received the Ph.D. degree from the University of Melbourne, in 1999, under the supervision of Prof. G. Dong and Prof. R. Kotagiri. From 1999 to 2002, he was an IT industrial as an IT Engineer and a Consultant, and from 2002 to 2014, he was a Research Scientist in with CSIRO, Australia. He is currently the Dean of the School of Computer and Data Engineering, Ningbo Institute of Technology (NIT), Zhejiang University. He is also a Distinguished Professor with the NIT, Zhejiang University, Hebei Academy of Sciences, and Hebei University of Economics and Business. His research interests lie in algorithm, stream data compression and processing, data warehousing, data integration, database theory, graph theory, and e-health. He is a Senior Member of ACM.