Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 197 (2022) 105–126
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original articles

Lagrange–Chebyshev Interpolation for image resizing☆

Donatella Occorsio[a,b], Giuliana Ramella[b], Woula Themistoclakis[b,*]

[a] Department of Mathematics and Computer Science, University of Basilicata, Viale dell'Ateneo Lucano 10, 85100 Potenza, Italy
[b] C.N.R. National Research Council of Italy, Institute for Applied Computing "Mauro Picone", Via P. Castellino, 111, 80131 Naples, Italy

Received 7 September 2021; received in revised form 12 December 2021; accepted 19 January 2022
Available online 4 February 2022

Abstract

Image resizing is a basic tool in image processing, and in literature, we have many methods based on different approaches, which are often specialized in only upscaling or downscaling. In this paper, independently of the (reduced or enlarged) size we aim to get, we approach the problem at a continuous scale where the underlying function representing the image is globally approximated by its Lagrange–Chebyshev I kind interpolation polynomial corresponding to suitable (tensor product) grids of first kind Chebyshev zeros. This is a well-known approximation tool widely used in many applicative fields due to the optimal behavior of the related Lebesgue constants. Here we aim to show how Lagrange–Chebyshev interpolation can be fruitfully applied also for resizing any digital image in both downscaling and upscaling at any desired size. The performance of the proposed method has been tested in terms of the standard SSIM (Structured Similarity Index Measurement) and PSNR (Peak Signal to Noise Ratio) metrics. The results indicate that, in upscaling, it is almost comparable with the classical Bicubic resizing method with slightly better metrics, but in downscaling a much higher performance has been observed in comparison with Bicubic and other recent methods too. Moreover, for all downscaling cases with an odd scale factor, we give a theoretical estimate of the MSE (Mean Squared Error) of the output image produced by our method, stating that it is certainly null (hence PSNR equals infinite and SSIM equals one) if the input image's MSE is null.

Keywords: Image resizing; Image downscaling; Image upscaling; Lagrange interpolation; Chebyshev nodes

1. Introduction

In this paper, we deal with the problem of Image Resizing (IR). This has been widely investigated over the past decades and is still an active research area characterized by many applications in different domains, including image transmission, satellite image analysis, gaming, remote sensing, etc. (see, e.g., [9,22,24,26,43,44]).

In literature, IR methods based on different approaches have been developed. They can be broadly classified based on their underlying algorithmic approach, such as Fourier transform [18], wavelet transform [8], and neural

☆ The research has been accomplished within RITA (Research ITalian network on Approximation) and UMI (Unione Matematica Italiana) research groups: TAA-UMI (Approximation Theory and Applications) and AI&ML&MAT-UMI (Mathematics for Artificial Intelligence and Machine Learning). It has been partially supported by GNCS-INdAM and University of Basilicata (local funds).
* Corresponding author.
E-mail addresses: donatella.occorsio@unibas.it (D. Occorsio), giuliana.ramella@cnr.it (G. Ramella), woula.themistoclakis@cnr.it (W. Themistoclakis).

https://doi.org/10.1016/j.matcom.2022.01.017

network [42,45]. The main advantages and limits of these methods have been discussed in many papers; see, e.g., [28,43] and the references therein.

According to another possible classification, the IR methods can be distinguished into non-adaptive and adaptive. The methods belonging to the non-adaptive category treat all pixels equally. This category includes many of the most commonly used algorithms such as the nearest neighbor, bilinear, bicubic and B-splines interpolation, Lanczos method [28]. Usually, non-adaptive scaling methods have problems of blurring or artifacts around edges and only retain the low-frequency components of the original image. The methods belonging to the adaptive category try to maximize the quality of the resized image using adaptive local techniques. This category includes many different approaches, often used in other image processing tasks [8,18,45]. Adaptive scaling methods generally provide better image visual quality and preserve high-frequency components but they take more computational time than non-adaptive ones.

Often downscaling and upscaling are considered separate problems (see, e.g., [28,43]), and most of the existing methods are specialized in only one direction, sometimes for a limited range of scaling factors (see, e.g., [13,39]).

The method we are going to introduce works in both down and up scaling directions and for "large" scaling factors as well. It is a non-adaptive method and falls into the class of the interpolation methods. The latter are typically based on uniform grids of nodes and generally consider a local interpolation of the initial pixels rather than a global one. The novelty we are proposing begins with a non-standard modeling of the image scaling problem, where we consider sampling grids that are not uniform but are made of Chebyshev zeros of I kind for each spatial coordinate.

Discrete polynomial approximation based on Chebyshev zeros is a pillar in approximation theory and practices [38]. It has been widely studied and applied in several fields and also recently, new Chebyshev-like grids such as Xu points and Padua points have been introduced to get optimal interpolation processes on the square (see, e.g., [10,29,41]). Nevertheless, to our knowledge, its usage in image processing has been mainly limited to particular cases, such as Magnetic Particle Imaging that is strictly related to Lissajous curves generating the Padua points (see, e.g., [14,17,20]).

Here we aim to show how Lagrange interpolation at Chebyshev zeros of I kind can be fruitfully applied to resize any (color or gray-level) image to any size. The resulting method is denoted by LCI method or simply LCI. Its main idea is based on the previous non-standard model that allows to approximate globally any input image by its associated Lagrange–Chebyshev I kind bivariate polynomial, which can then be sampled to the desired size.

Due to the relevant theoretical background, LCI provides precise error estimates that are generally lacking in the other scaling methods. In particular, for all downscaling with an odd scale factor s, we state that the output image made by LCI has a Mean Squared Error (MSE) not greater than s^2 times the MSE of the input image.

To test the performance of LCI we have implemented it in Matlab, computing by Matlab the usual SSIM (Structured Similarity Index Measurement) and PSNR (Peak Signal to Noise Ratio) metrics. We have carried out the experimentation on more than one thousand of images collected in six datasets, having different characteristics on the size and the quality of the images, the variety of subjects, etc.

For an initial comparison, we focused on interpolation methods working as well in both down and up scaling directions. This is the case of the classical bicubic interpolation method (shortly BIC) [19] implemented by the Matlab built-in function `imresize` that we have used both in upscaling and downscaling to get a first comparison.

By the extensive experimentation we carried out, we assess that the performance of the LCI method is superior in downscaling (d-LCI method), where we arrive to get a null MSE (hence PSNR equals infinite and SSIM equals one) according to the theoretical estimate. On the contrary, in upscaling (u-LCI method), we find only slight differences with BIC, and the two methods seem almost comparable.

Thus it seemed relevant to us to further investigate only the downscaling case by testing d-LCI method in comparison with two other recent downscaling methods, here denoted by DPID [40] and L_0 [21], both of them not belonging to the family of interpolation methods for downscaling. The effective better performance in downscaling has been confirmed also in these cases, with an increasing gap as the downscaling factor increases. Moreover, by running the public available Matlab code at [1] (DPID) and [2] (L_0), we have observed that DPID and L_0 methods work only for integer scale factors and they are not always implementable for images with large sizes due to too long running times or too much memory. On the contrary, LCI method keeps the runtimes contained and offers high flexibility since it works in both downscaling and upscaling with non integer scale factors too, being possible to specify the desired final size or the scale factor as input parameters.

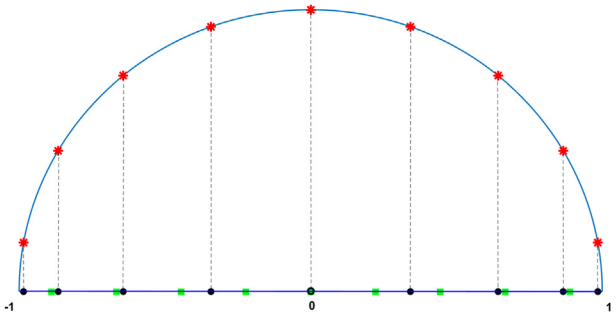

Fig. 1. For $\mu = 9$ the first kind Chebyshev zeros (black circles), the associated arc cosine (red stars) and the equidistant nodes (green squares). In this case μ is odd and the origin is both an equidistant node and a Chebyshev zero. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We remark that similar nice features are shared by the Matlab function `imresize`, BIC, that is also a little faster than our Matlab code (available on GitHub), but it has lower performances, especially for downscaling, in terms of PSNR and SSIM quality measures.

The paper is organized as follows. In Section 2, we introduce the mathematical model and the key idea of LCI. For a simpler exposition, we first describe LCI method in Section 3 for a gray-level image, and then, in Section 4, for RGB color images. Both grayscale and color cases are considered in Section 5, where some features of our approach are discussed. Finally, Section 6 consists of all the numerical experiments.

2. The mathematical model

In this section, we introduce a non-standard model for the image scaling problem, that will lead us to the key idea of LCI method. To this aim, we premise that, from the mathematical viewpoint, in the continuum, an image $\mathcal{I}$ is a function $f(x, y)$ of the spatial coordinates which, without losing the generality, we can assume belonging to the open square $A =]-1, 1[^2$. Hence, its digital version I of $n \times m$ pixels is supposed to be made of the values that f takes on a discrete grid of nodes $X_{n\times m} \subset A$. Regarding such grid, it is generally supposed that $X_{n\times m} = X_n \times X_m$ where we set

$$X_\mu := \{x_k^\mu \ : \ k = 1, \ldots, \mu\} \subset]-1, 1[, \qquad \forall \mu \in \mathbb{N}. \tag{1}$$

A typical and natural choice of the (univariate) system of nodes X_μ is to take the μ internal equidistant nodes

$$X_\mu^{equ} = \left\{ -1 + \frac{2k-1}{\mu} \ : \ k = 1, \ldots, \mu \right\}. \tag{2}$$

However, it is well known that equally spaced nodes are not the best choice for Lagrange interpolation since they lead to exponentially growing Lebesgue constants, whereas optimal Lebesgue constants (growing at the minimal projection rate, see, e.g., [38]) are provided by the Chebyshev nodes of the first kind

$$x_k^\mu = \cos\left[\frac{(2k-1)\pi}{2\mu}\right], \qquad k = 1 : \mu, \qquad \forall \mu \in \mathbb{N}, \tag{3}$$

where, throughout the paper, the notation $k = 1 : \mu$ means that $k \in \{1, 2, \ldots, \mu\}$.

Indeed, at a first look, Chebyshev zeros may seem unsuitable for sampling an arbitrary image since they are not equidistant on the segment $]-1, 1[$ of each spatial coordinate, but they are *arc sine* distributed, becoming denser near the endpoints ± 1 (see Fig. 1). Nevertheless, in our model, we transfer the sampling question from the segment $]-1, 1[$ to the semicircle of radius 1 centered at the axes origin. Thus, using the standard setting $t = \arccos x$, instead of the usual equidistant nodes on the segment, our method takes the following equidistant nodes on the semicircle

$$t_k^\mu := \frac{(2k-1)\pi}{2\mu}, \qquad k = 1 : \mu, \qquad \mu \in \mathbb{N}. \tag{4}$$

These nodes define the Chebyshev zeros $x_k^\mu = \cos t_k^\mu$ in (3) and the grid

$$X_{n\times m} = \left\{(x_i^n, x_j^m) : \quad i = 1:n,\ \ j = 1:m\right\}, \qquad n, m \in \mathbb{N}, \tag{5}$$

which we actually choose to sample arbitrary (continuous) images $\mathcal{I}$ and to obtain digital images of $n \times m$ pixels, for arbitrary $n, m \in \mathbb{N}$.

According to this model, let $I = f(X_{n\times m})$ be the digital image of $n \times m$ pixels, i.e.

$$I_{i,j} := f(x_i^n, x_j^m), \qquad i = 1:n,\ \ j = 1:m, \tag{6}$$

and let I^{in} be its more or less corrupted version really available as input data, say

$$I_{i,j}^{in} := \tilde{f}(x_i^n, x_j^m), \qquad i = 1:n,\ \ j = 1:m, \tag{7}$$

where $\tilde{f}$ denotes a corrupted version of the continuous image.

Starting from the data I^{in}, the resizing problem aims to find a good approximation of the resampled image I^{res} consisting of the samples of f at the different (more or less dense) grid $X_{N\times M}$, i.e.

$$I_{k,h}^{res} := f(x_k^N, x_h^M), \qquad k = 1:N,\ \ h = 1:M. \tag{8}$$

Hence, under our modeling, the resizing problem raises the following approximation question: starting from the approximate values of f at the Chebyshev grid $X_{n\times m}$, how to approximate f at another (coarser in downscaling or finer in upscaling) Chebyshev grid $X_{N\times M}$?

The solution we are going to propose relies on the global approximation of a function by its bivariate Lagrange polynomial interpolating at the Chebyshev grid $X_{n\times m}$. It is such kind of polynomial that we are going to sample at the new grid $X_{N\times M}$ in order to get the resized image.

We recall that Lagrange–Chebyshev I kind interpolation polynomial can be easily deduced via tensor product from the univariate case (see for instance [30,31]) and fast algorithms can be implemented for its computation (see e.g. [29]). Moreover, we recall that wavelets techniques could be applied to such kind of interpolation (see, e.g., [15] and the references therein).

In the next two sections we give the precise formulas to compute the pixel values of the output image from those of the input image. We point out that, even if we made a Matlab code for testing LCI, the given formulas are easily translatable in any programming language.

3. Re-sampling gray-level images

In the case of a grayscale image, $\mathcal{I}$ is represented at a continuous scale by a scalar function $f(x, y)$ whose values are the gray levels at the spatial coordinates $(x, y) \in A$. Moreover, at a discrete scale, digital images of $n \times m$ pixels are matrices $I \in \mathbb{R}^{n\times m}$ whose elements are the samples of f at the discrete nodes set $X_{n\times m}$ defined in our model by (5) and (3). Then, according to the notation introduced in the previous section, the resizing problem of $\mathcal{I}$ (in upscaling or in downscaling, respectively) consists in finding a good approximation of the matrix $I^{res} \in \mathbb{R}^{N\times M}$ given by (8) starting from the matrix (having a reduced or enlarged size, respectively) $I^{in} \in \mathbb{R}^{n\times m}$ defined as in (7) being $\tilde{f}$ a more or less corrupted version of f.

We approach such an approximation problem by using the following bivariate (tensor product) Lagrange–Chebyshev polynomial based on the available data I^{in}

$$L_{n,m}\tilde{f}(x, y) := \sum_{i=1}^{n}\sum_{j=1}^{m} \tilde{f}(x_i^n, x_j^m)\ell_i^n(x)\ell_j^m(y), \qquad (x, y) \in A, \tag{9}$$

where, for all $\mu \in \mathbb{N}$, ℓ_k^μ denotes the $k-$th fundamental Lagrange polynomial related to the nodes system X_μ, namely

$$\ell_k^\mu(\xi) := \prod_{\substack{s=1\\ s\neq k}}^{\mu} \frac{\xi - x_s^\mu}{x_k^\mu - x_s^\mu}, \qquad \xi \in [-1, 1], \qquad k = 1:\mu. \tag{10}$$

It is well-known that the Lagrange–Chebyshev polynomial in (9) interpolates $\tilde{f}$ at the grid $X_{n\times m}$, i.e.

$$L_{n,m}\tilde{f}(x_i^n, x_j^m) = \tilde{f}(x_i^n, x_j^m) = I_{i,j}^{in}, \qquad i = 1:n,\ \ j = 1:m. \tag{11}$$

The samples of such polynomial at the re-scaled grid $X_{N\times M}$ constitute the approximate resized image provided by the LCI method. Hence, the LCI method computes the matrix $I^{out} \in \mathbb{R}^{N\times M}$ whose entries are the following

$$I^{out}_{k,h} := L_{n,m}\tilde{f}(x^N_k, x^M_h), \qquad k = 1:N, \quad h = 1:M, \tag{12}$$

i.e., more explicitly, we have

$$I^{out}_{k,h} := \sum_{i=1}^{n}\sum_{j=1}^{m} I^{in}_{i,j}\ \ell^n_i(x^N_k)\ell^m_j(x^M_h), \qquad k = 1:N, \quad h = 1:M. \tag{13}$$

Starting from this formula, several numerical procedures can be followed. In particular, introducing the Vandermonde-like matrices

$$V_1 := \left[\ell^n_i(x^N_k)\right]_{i,k} \in \mathbb{R}^{n\times N}, \qquad V_2 := \left[\ell^m_j(x^M_h)\right]_{j,h} \in \mathbb{R}^{m\times M}, \tag{14}$$

by (13), the output matrix I^{out} can be computed from the input matrix I^{in} according to the following matrices identity

$$I^{out} = V_1^T I^{in} V_2. \tag{15}$$

Note that formula (15) can turn useful in case we have to resize, at the same scale, a set of images having the same size, e.g., acquired by a device at a fixed size. In this case, the matrices V_i can be pre-computed once, reducing the CPU time.

Moreover, it is well known that the fast computation of the matrices V_i can be achieved by using, instead of (10), the following more convenient form of the fundamental Lagrange polynomial

$$\ell^\mu_k(\cos t) = \frac{2}{\mu}{\sum_{r=0}^{\mu-1}}' \cos\left[\frac{(2k-1)r\pi}{2\mu}\right]\cos[rt], \qquad k = 1:\mu, \quad t \in [0,\pi], \tag{16}$$

where, as usual, the prime on the summation symbol means that the first addendum is halved. Hence, by (16) we get that the matrices V_1 and V_2 can be computed by using fast cosine transform algorithms (see e.g. [33]) being

$$(V_1)_{i,k} = \ell^n_i(x^N_k) = \frac{2}{n}{\sum_{r=0}^{n-1}}' \cos\left[\frac{(2i-1)r\pi}{2n}\right]\cos\left[\frac{(2k-1)r\pi}{2N}\right], \qquad i = 1:n,\ k = 1:N, \tag{17}$$

$$(V_2)_{j,h} = \ell^m_j(x^M_h) = \frac{2}{m}{\sum_{s=0}^{m-1}}' \cos\left[\frac{(2j-1)s\pi}{2m}\right]\cos\left[\frac{(2h-1)s\pi}{2M}\right], \qquad j = 1:m,\ h = 1:M. \tag{18}$$

4. Re-sampling RGB color images

Now we are going to introduce the method for a general color image $\mathcal{I}$. In this case, behind $\mathcal{I}$ there is a vector function $\mathbf{f} : A \to \mathbb{R}^3$ whose components $\mathbf{f} = (f_R, f_G, f_B)$ represent $\mathcal{I}$ in the RGB color space. According to our model, the input image I^{in} and the target resized image I^{res} are represented in the RGB space as the follows

$$\begin{aligned} &I^{in} \in \mathbb{R}^{n\times m\times 3}, \quad && I^{in} \equiv (I^{in}_R, I^{in}_G, I^{in}_B),\\ &I^{res} \in \mathbb{R}^{N\times M\times 3}, \quad && I^{res} \equiv (I^{res}_R, I^{res}_G, I^{res}_B), \end{aligned}$$

where we assume that for all $i = 1:n$, $j = 1:m$ and $k = 1:N$, $h = 1:M$ it is

$$\begin{aligned} &\left(I^{in}_R\right)_{i,j} = \tilde{f}_R(x^n_i, x^m_j), && \left(I^{res}_R\right)_{k,h} = f_R(x^N_k, x^M_h),\\ &\left(I^{in}_G\right)_{i,j} = \tilde{f}_G(x^n_i, x^m_j), && \left(I^{res}_G\right)_{k,h} = f_G(x^N_k, x^M_h),\\ &\left(I^{in}_B\right)_{i,j} = \tilde{f}_B(x^n_i, x^m_j), && \left(I^{in}_B\right)_{k,h} = f_B(x^N_k, x^M_h). \end{aligned} \tag{19}$$

Thus, similarly to the gray-level case and with obvious meaning of the notation, we start from

$$I^{in}_{i,j} = \tilde{\mathbf{f}}(x^n_i, x^m_j), \qquad i = 1:n, \quad j = 1:m, \tag{20}$$

and approximate I^{res} by

$$I^{out}_{k,h} = L_{n,m}\tilde{\mathbf{f}}(x^N_k, x^M_h), \qquad k = 1:N, \quad h = 1:M, \tag{21}$$

Table 1
Datasets list.

Dataset	n. of Images	Sizes
BSDS500	500	481 × 321 or 321 × 481
NY17	17	From 1200 × 1600 to 5430 × 3520
NY96	96	From 500 × 334 to 6394 × 3456
13US	13	From 241 × 400 to 400 × 310
URBAN100	100	From 1024 × 564 to 1024 × 1024
PEXELS300	300	1800 × 1800

i.e., for $k = 1 : N$ and $h = 1 : M$, we define

$$I^{out}_{k,h} = \left(L_{n,m}\tilde{f}_R(x^N_k, x^M_h),\ L_{n,m}\tilde{f}_G(x^N_k, x^M_h),\ L_{n,m}\tilde{f}_B(x^N_k, x^M_h) \right). \tag{22}$$

Hence, by applying the same argument of the gray-level case, we get that the RGB components of the output image $I^{out} \equiv (I^{out}_R, I^{out}_G, I^{out}_B)$ are given by

$$I^{out} = (V_1^T I^{in}_R V_2,\ V_1^T I^{in}_G V_2,\ V_1^T I^{in}_B V_2), \tag{23}$$

with V_1, V_2 defined in (17)–(18).

Finally, if we denote the entries of the input and output image as follows

$$I^{in} = \left[I^{in}(i, j, c) \right]_{i=1:n,\ j=1:m,\ c=1:3}, \qquad I^{out} = \left[I^{out}(k, h, c) \right]_{k=1:N,\ h=1:M,\ c=1:3},$$

then we have the following explicit formula

$$I^{out}(k, h, c) = \sum_{i=1}^{n} \sum_{j=1}^{m} I^{in}(i, j, c)\ \ell^n_i(x^N_k)\ell^m_j(x^M_h), \qquad k = 1 : N,\ h = 1 : M,\ c = 1 : 3. \tag{24}$$

5. Model analysis

For a general analysis of the error we get in approximating the target resized image $I^{res} = [\mathbf{f}(x^n_i, x^m_j)]_{i,j}$ with the output image $I^{out} = [L_{n,m}\tilde{\mathbf{f}}(x^N_h, x^M_k)]_{h,k}$ produced by LCI method, we refer the reader to the wide existing literature on the Lagrange interpolation error estimates (see e.g. [38] and the references therein). In this paper, we are interested in measuring the performance of LCI method through some standard metrics usually used in Image Processing (see e.g. [34] and the references therein). In particular, we focus on the Mean Squared Error (MSE) defined as follows

$$\mathrm{MSE}(I^{res}, I^{out}) = \begin{cases} \dfrac{1}{MN}\|I^{res} - I^{out}\|^2_F, & \text{gray-level images,} \\ \dfrac{1}{3NM}\left\{\|I^{res}_R - I^{out}_R\|^2_F + \|I^{res}_G - I^{out}_G\|^2_F + \|I^{res}_B - I^{out}_B\|^2_F\right\}, & \text{RGB color images,} \end{cases}$$

being $\|\cdot\|_F$ the Frobenius norm

$$\|A\|_F := \left(\sum_{k=1}^{N} \sum_{h=1}^{M} A^2_{k,h} \right)^{\frac{1}{2}}, \qquad A = (A_{k,h}) \in \mathbb{R}^{N\times M}.$$

Moreover, we consider the Peak Signal to Noise Ratio (PSNR) defined by the previous MSE as follows

$$\mathrm{PSNR}(I^{res}, I^{out}) = 20 \log_{10}\left(\frac{\max_f}{\sqrt{\mathrm{MSE}(I^{res}, I^{out})}} \right), \tag{25}$$

with $\max_f = 255$, since we use the representation by 8 bits per sample.

Finally, in our experiments we evaluate the Structured Similarity Index Measurement (SSIM). For gray-level images it is defined as

$$\mathrm{SSIM}(I^{res}, I^{out}) = \frac{\left[2\mu(I^{res})\mu(I^{out}) + c_1\right]\left[2\mathrm{cov}(I^{res},\ I^{out}) + c_2\right]}{\left[\mu^2(I^{res}) + \mu^2(I^{out}) + c_1\right]\left[\sigma^2(I^{res}) + \sigma^2(I^{out}) + c_2\right]}, \tag{26}$$

Table 2
Average results in upscaling.

	x2			x3			x4		
	PSNR	SSIM	T	PSNR	SSIM	T	PSNR	SSIM	T
BSDS500									
BIC	26,341	0,886	**0,003**	24,821	0,837	**0,002**	22,251	0,701	**0,003**
u-LCI	**26,381**	**0,888**	0,014	**24,868**	**0,839**	0,010	**22,446**	**0,769**	0,008
NY17									
BIC	34,806	0,958	**0,091**	31,205	0,924	**0,074**	29,808	0,907	**0,071**
u-LCI	**35,412**	**0,960**	1,357	**31,542**	**0,924**	0,839	**30,183**	**0,908**	0,634
NY96									
BIC	33,391	0,953	**0,056**	29,754	0,913	**0,055**	28,741	0,891	**0,051**
u-LCI	**33,900**	**0,955**	0,812	**29,956**	**0,914**	0,567	**28,986**	**0,891**	0,459
URBAN100									
BIC	25,433	0,882	**0,009**	21,306	0,755	**0,008**	21,703	0,741	**0,008**
u-LCI	**25,896**	**0,886**	0,064	**21,325**	**0,754**	0,051	**21,919**	**0,793**	0,059
13US									
BIC	24,116	0,861	**0,002**	20,754	0,734	**0,002**	20,545	0,710	**0,002**
u-LCI	**24,486**	**0,868**	0,012	**20,780**	**0,738**	0,010	**20,656**	**0,713**	0,009
PEXELS300									
BIC	34,885	0,962	**0,041**	31,727	0,932	**0,037**	29,895	0,908	**0,035**
u-LCI	**35,709**	**0,996**	0,300	**32,227**	**0,935**	0,216	**30,289**	**0,910**	0,185

where $\mu(A), \sigma(A)$ and $\mathrm{cov}(A, B)$ indicate the average, variance and covariance, respectively, of the matrices A, B, and c_1, c_2 are constants usually fixed as $c_1 = (0.01 \times L), c_2 = (0.03 \times L)$ with the dynamic range of the pixel values $L = 255$ for 8-bit images.

In the case of RGB color images, making the conversion to the color space YCbCr, the SSIM is computed by the above definition applied to the intensity Y (luma) channel.

Based on the previous metrics, the results of wide experimentation will be reported in the next section. In the sequel, we are going to underline two particular features of LCI method.

A first aspect is related to the model choice which, for odd downscaling factors $s := n/N = m/M > 1$, leads to state the following

Proposition 5.1. *Under the previously introduced notation, if there exists $\ell \in \mathbb{N}$ such that $n = (2\ell - 1)N$ and $m = (2\ell - 1)M$ hold, then we have*

$$\mathrm{MSE}(I^{res}, I^{out}) \le s^2 MSE(I, I^{in}), \qquad s := (2\ell - 1). \tag{27}$$

Moreover, if $I = I^{in}$ then we get the best results

$$\mathrm{MSE}(I^{res}, I^{out}) = 0, \qquad \textit{and} \qquad \mathrm{SSIM}(I^{res}, I^{out}) = 1. \tag{28}$$

Proof. Let us give the proof in the case of a gray-level image since for RGB color images the statement will easily follow by considering the single RGB components.

As a keynote of the proof, we observe that for any $\ell \in \mathbb{N}$, we have

$$n = (2\ell - 1)N \qquad \Longrightarrow \qquad X_N \subset X_n.$$

More precisely, setting $s := (2\ell - 1)$, it is easy to check that

$$n = sN \qquad \Longrightarrow \qquad x_k^N = x_i^n \quad \text{with} \quad i = \frac{s(2k-1)+1}{2}, \quad k = 1 : N, \tag{29}$$

where we remark that for all $k = 1 : N$, the numerator $s(2k - 1) + 1$ is certainly even since s is odd.

Table 3
Average results in downscaling.

	:2			:3			:4		
	PSNR	SSIM	T	PSNR	SSIM	T	PSNR	SSIM	T
BSDS500									
BIC	38,872	0,993	**0,006**	39,253	0,993	**0,009**	39,183	0,993	**0,017**
DPID	41,745	0,996	7,696	41,827	0,996	12,246	41,206	0,996	18,615
L_0	29,317	0,961	3,647	32,873	0,971	8,020	34,174	0,971	14,295
d-LCI	**53,732**	**1,000**	0,057	**Inf**	**1,000**	0,091	**55,889**	**1,000**	0,137
NY17									
BIC	45,969	0,995	**0,229**	47,114	0,996	**0,325**	46,973	0,996	**0,639**
DPID	47,675	0,998	448,023	47,657	0,998	731,498	47,112	0,997	1.098,113
L_0	34,754	0,972	208,574	37,285	0,979	617,386	oom	oom	oom
d-LCI	**54,265**	**0,999**	6,614	**Inf**	**1,000**	13,317	**57,491**	**1,000**	22,859
NY96									
BIC	44,757	0,996	**0,155**	45,858	0,997	**0,219**	45,682	0,996	**0,422**
DPID	46,743	0,998	291,685	46,948	0,998	479,638	46,421	0,998	714,908
L_0	33,913	0,974	133,190	oom	oom	oom	oom	oom	oom
d-LCI	**54,067**	**0,999**	4,698	**Inf**	**1,000**	8,279	**57,223**	**1,000**	13,898
URBAN00									
BIC	35,661	0,989	**0,027**	36,010	0,990	**0,041**	35,940	0,989	**0,068**
DPID	39,178	0,996	37,592	39,178	0,996	60,834	38,702	0,995	93,996
L_0	26,718	0,951	13,267	31,061	0,969	28,845	33,571	0,973	50,312
d-LCI	**52,613**	**0,999**	0,234	**Inf**	**1,000**	0,416	**55,452**	**1,000**	0,618
13US									
BIC	35,129	0,990	**0,005**	35,469	0,991	**0,009**	35,397	0,991	**0,013**
DPID	38,061	0,996	5,593	38,326	0,996	8,905	37,707	0,995	13,819
L_0	25,521	0,949	2,572	30,231	0,972	5,706	32,929	0,979	9,939
d-LCI	**52,813**	**1,000**	0,042	**Inf**	**1,000**	0,073	**55,374**	**1,000**	0,108
PEXELS300									
BIC	45,605	0,997	**0,088**	46,661	0,997	**0,120**	46,502	0,997	**0,225**
DPID	46,934	0,998	151,800	47,134	0,998	246,573	46,650	0,998	374,155
L_0	33,712	0,976	56,454	36,081	0,980	125,413	37,237	0,981	228,683
d-LCI	**54,667**	**1,000**	1,560	**Inf**	**1,000**	3,333	**57,115**	**1,000**	4,977

Consequently, we deduce (27) from (11) and (29) as follows

$$\begin{aligned}
&\mathrm{MSE}(I^{res}, I^{out}) = \frac{1}{NM}\sum_{k=1}^{N}\sum_{h=1}^{M}\left[f\left(x_k^N, x_h^M\right) - L_{n,m}\tilde{f}\left(x_k^N, x_h^M\right)\right]^2 \\
&= \frac{1}{NM}\sum_{k=1}^{N}\sum_{h=1}^{M}\left[f\left(x^n_{\frac{s(2k-1)+1}{2}}, x^m_{\frac{s(2h-1)+1}{2}}\right) - \tilde{f}\left(x^n_{\frac{s(2k-1)+1}{2}}, x^m_{\frac{s(2h-1)+1}{2}}\right)\right]^2 \\
&\le \frac{1}{NM}\sum_{i=1}^{n}\sum_{j=1}^{m}\left[f\left(x_i^n, x_j^m\right) - \tilde{f}\left(x_i^n, x_j^m\right)\right]^2 = s^2 MSE(I, I^{in}).
\end{aligned}$$

Moreover, in the case that $I = I^{in}$, we have that $L_{n,m}f = L_{n,m}\tilde{f}$ and, due to the nesting property $X_{N\times M} \subset X_{n\times m}$, by (11) we get

$$I^{out}_{h,k} = L_{n,m}\tilde{f}(x_h^N, x_k^M) = L_{n,m}f(x_h^N, x_k^M) = f(x_h^N, x_k^M) = I^{res}_{h,k}, \qquad h = 1:N \;\; k = 1:M, \tag{30}$$

that implies the first equation in (28). Finally, (30) also yields

$$2\mathrm{cov}(I^{res},\ I^{out}) = \sigma(I^{res})^2 + \sigma(I^{out})^2, \qquad \text{and} \qquad 2\mu(I^{res})\mu(I^{out}) = \mu(I^{res})^2 + \mu(I^{out})^2,$$

which conclude the proof of (28). ◇

Fig. 2. The well-known image *Flowers* taken from USC-SIPI [3] (left) upsampled at the double scale by Lagrange interpolation at Chebyshev nodes (middle) and at equally spaced nodes (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Remark 5.2. We note that Eq. (29) provides the precise position of the pixel values that coincide in the input and output image. Due to (11), such a knowledge can be used to get optimized algorithms that speed up the computation in all downscaling with an odd scale factor. Moreover, we remark that (28) holds more generally if only the previous special input pixels are exact.

We point out that the previous proposition is a direct consequence of our choice of the sampling model based on Chebyshev zeros (3), instead of the usual equally spaced nodes (2).

Such choice (and more generally the choice of "good" interpolation knots) allows to use global interpolation processes that are not possible in the case of equally spaced nodes. In fact, as second aspect, we aim to highlight the relevance of the choice of good interpolation knots by showing the effects we have in the performance of global Lagrange interpolation methods associated with the univariate nodes set X_μ^{equ} in (2). The disastrous effects of the exponential growth of Lebesgue constants are visible in the next experiment concerning an image zooming $\times 2$ from the size $n \times n$ with $n = 256$ to the size $N \times N$ with $N = 512$. Fig. 2 displays the images obtained starting from the same input image (Fig. 2, left) and using the Lagrange interpolation polynomial based on equally spaced nodes $X_n^{equ} \times X_n^{equ}$ (Fig. 2, right) and on the Chebyshev grid $X_{n\times n}$ (Fig. 2, middle). The images we see are the samples of the previous Lagrange polynomials at the respective $(N \times N)$-grid and we can check how the equally-spaced sampling model (2) yields wrong results outside of a small region of the output image.

6. Experimental results

In this section, we propose a selection of the extensive experimentation we carried on to test the benefits of our procedure in downscaling (d-LCI method) and in upscaling (u-LCI method), comparing our results with other resizing techniques. To this aim we implement the LCI method by the Matlab code we made available on GitHub (https://github.com/ImgScaling/LCIscaling). For each input image, it first computes the matrices in (17)–(18) and then yields the output color image, according to (23). Even if we are mainly interested in the quality of resizing (in terms of PSNR and SSIM metrics) we also compare the required CPU time, shortly denoted by T. Nevertheless, we point out that we have run an "educational" code, that is not optimized for speed and, in particular, it does not take into account (29). Hence, we remark that for the LCI method the displayed values of T could be improved by using more optimized codes.

6.1. Comparison methods

As already mentioned, we compare LCI with BIC provided by Matlab function `imresize` with `'bicubic'` option where we recall that a new pixel is determined from a weighted average of the 16 closest pixels using an interpolating cubic convolution function satisfying prescribed assumptions of smoothness on f to gain at least a cubic order of convergence [19]. The comparison BIC-LCI regards either down and up resizing cases, giving as input either the scale factor or the final size of the desired image. Note that we have also tested the other different options of `imresize` (namely `'linear'` and `'nearest neighborhood'`), but for brevity, we do not report the results since they give no new insight.

Moreover, to further investigate the performance in downscaling case, we compare d-LCI with other two recent downscaling methods not belonging to the interpolation method class. These methods are briefly indicated as DPID

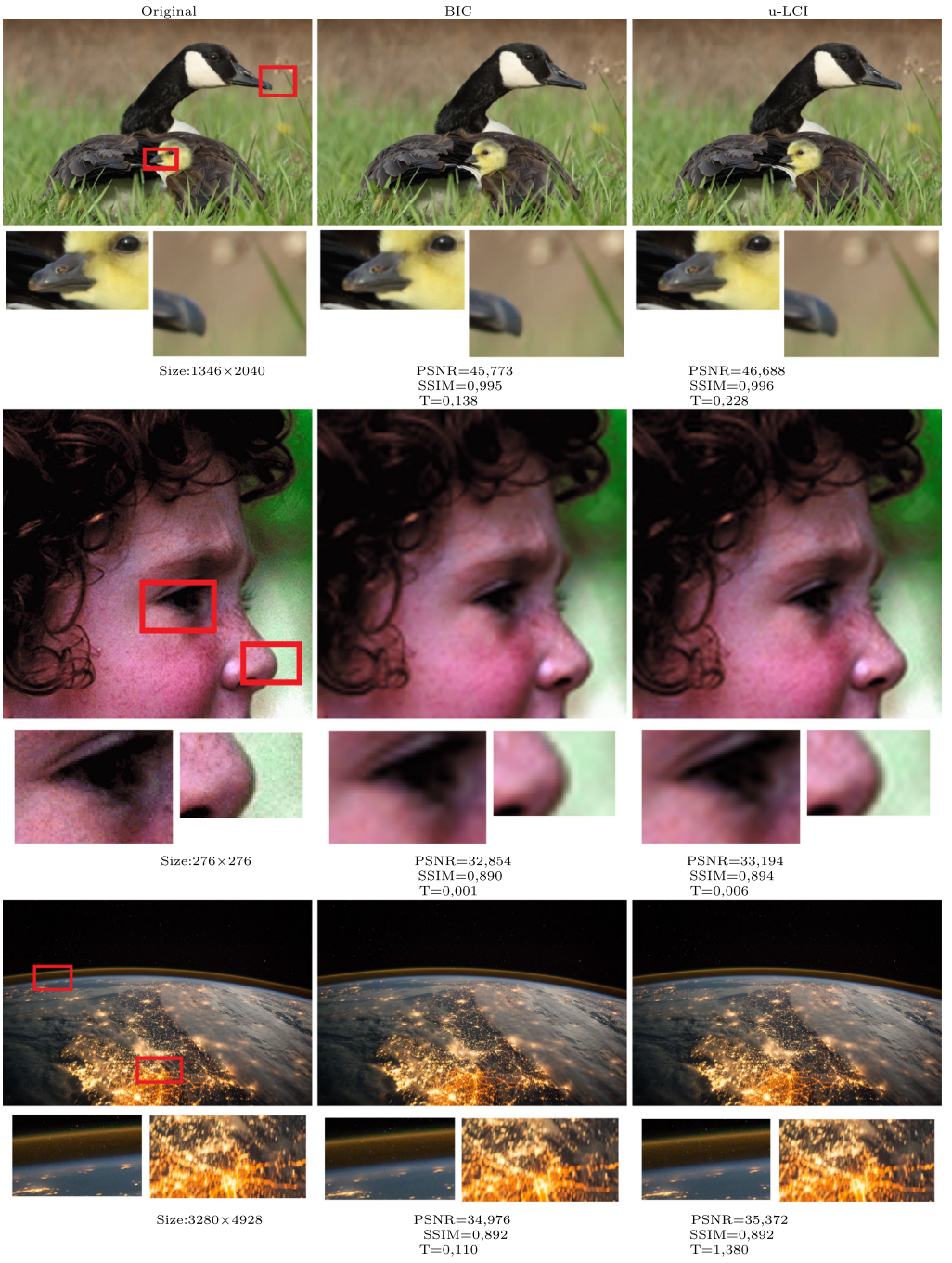

Fig. 3. Examples of upscaling performance results at the scale factor 2 (top), at the scale factor 3 (middle), at the scale factor 4 (bottom). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

method (described in [40] with the code available at [1]) and L_0 method (described in [21], with the code available at [2]).

To be fair, we remark that we forced DPID and L_0 also in upscaling direction for several scaling factors (the choice of the desired size is not allowed by DPID and L_0), but the results were very poor w.r.t. BIC and LCI and they have been not reported here.

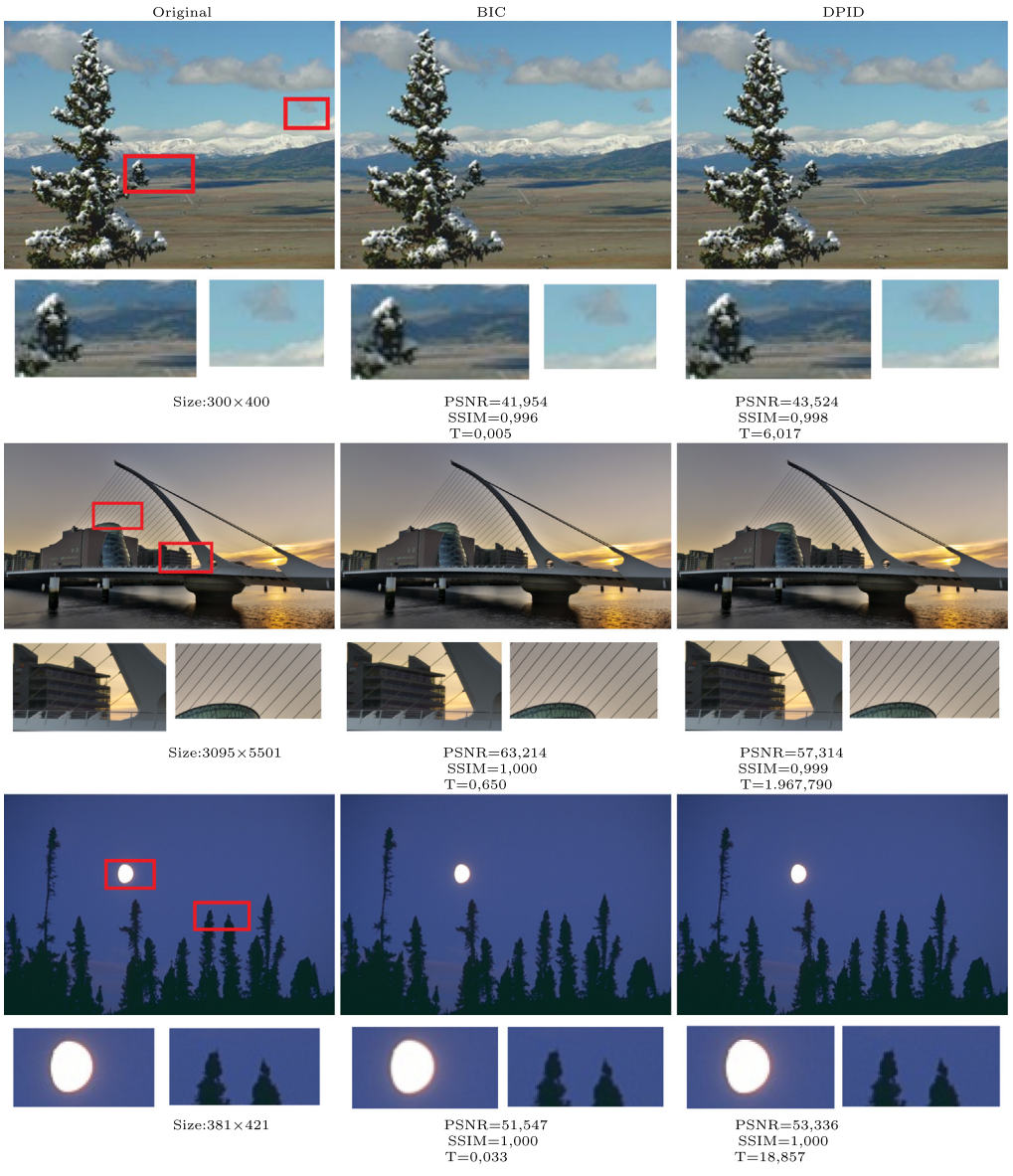

Fig. 4. Examples of downscaling performance results for BIC (2nd column) and DPID (3rd column), at the scale factor 2 (top), at the scale factor 3 (middle), at the scale factor 4 (bottom). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Finally, we point out that all the previous methods have run on the same PC with Intel Core i7 3770K CPU @350 GHz configuration.

6.2. Datasets

Since the results of any method depend on the type of input image, we have conducted the experimentation on different kinds of 8-bits images collected in six publicly available datasets, whose characteristics (acronym, number of images, range of their sizes) are synthesized in Table 1.

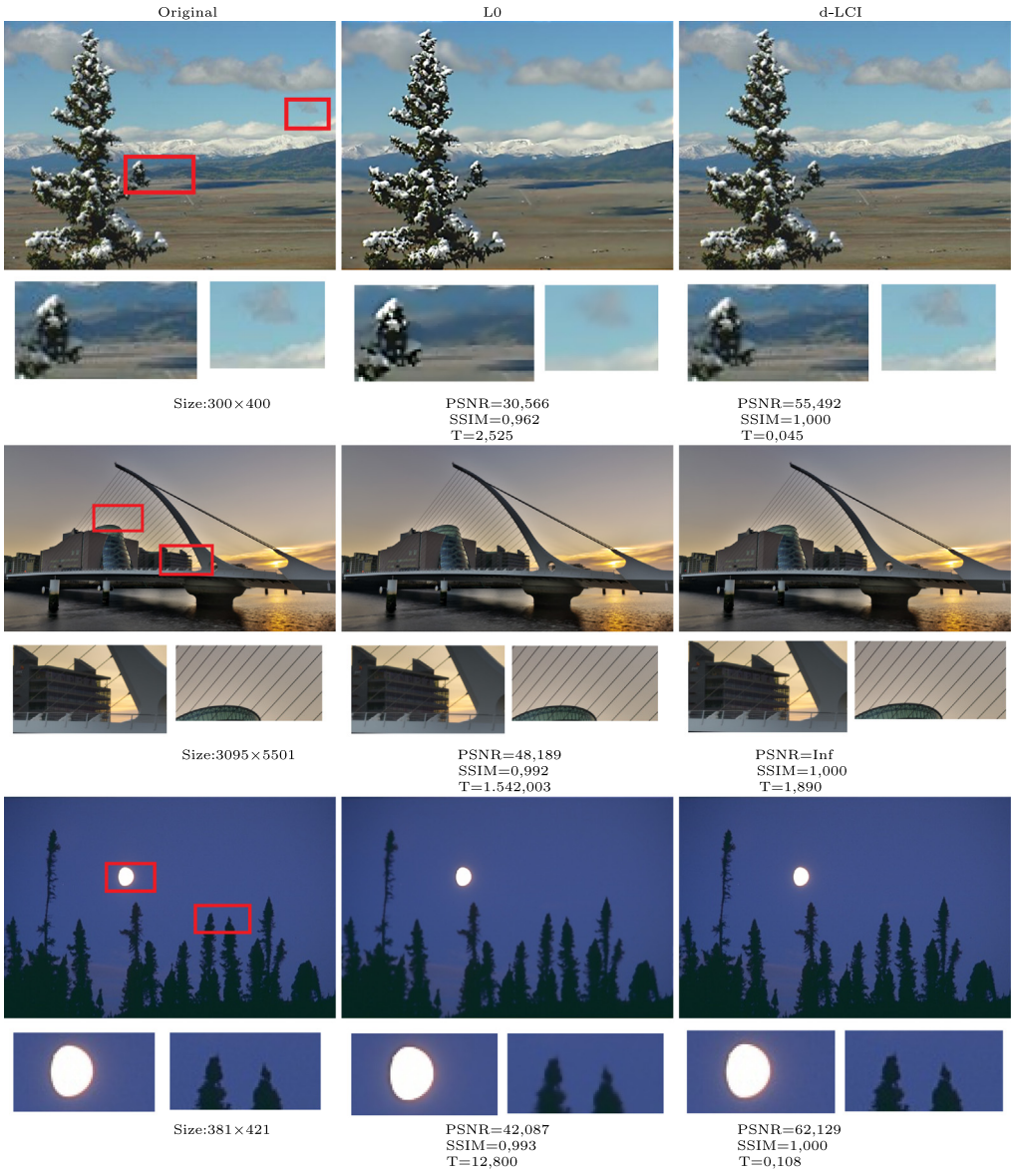

Fig. 5. Examples of downscaling performance results for L_0 (2nd column) and d-LCI (3rd column) at the scale factor 2 (top), at the scale factor 3 (middle), at the scale factor 4 (bottom). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We point out all datasets in Table 1, except PEXELS300, have been chosen since employed in [21,40] for testing DPID and L_0 methods. PEXELS300 has been generated by ourselves from a publicly available dataset in order to test images having the same large size. More specifically:

- BSDS500 dataset [25], available at [4], has been used for testing L_0 in [21]. It is sufficiently general and provides a large variety of images often employed for testing many other methods with different image analysis tasks such as image segmentation [27,35]), color quantization [11,12,36], etc.
- NY17 and NY96 datasets include those color images considered for DPID and selected in [40] from NASA Image Gallery [5] and YFCC100M (Yahoo Flickr Creative Commons 100 Million) [37] datasets.

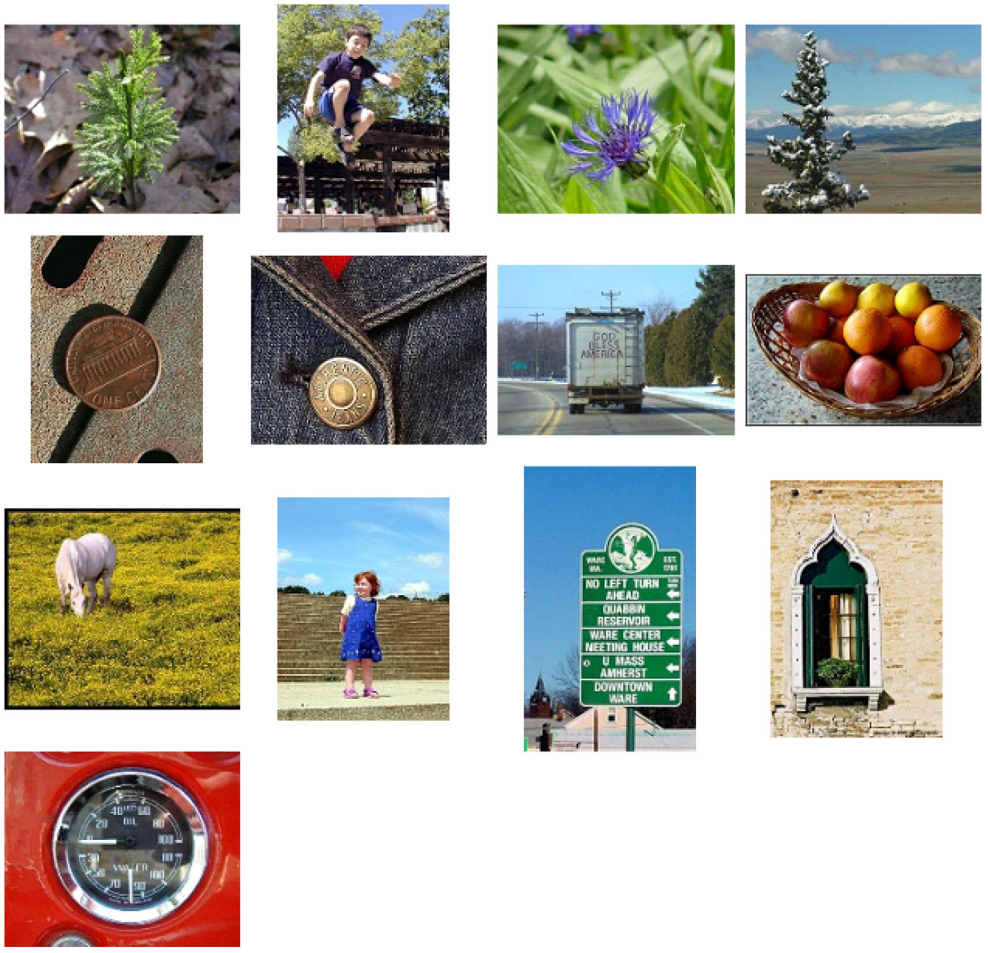

Fig. 6. 13US dataset image: [U1–U13] from left to right and from top to bottom.

- 13US [32] contains natural images, available at [6] and originally taken from the MSRA Salient Object Database [23]. It is another dataset used in [40].
- URBAN100 dataset [16], concerning urban scenes with images having one dimension equal to 1024, is commonly used to evaluate the performance of super-resolution models. It is used as test images for L_0 in [21].
- PEXELS300 has been obtained by randomly selecting from [7] 300 color images (having originally a different large size) we centrally cropped by 1800 × 1800 pixels. It is available at https://github.com/ImgScaling/LCI scaling.

All the previous datasets have been considered for either up and down scaling.

We remark that in all the performed experiments, the input images are not available from the datasets. Hence, following a typical approach in image quantitative evaluation, we fix all the images from the datasets as target images (i.e., I^{res} in our notation) to which we apply BIC in order to generate the input images (namely I^{in}) common to all methods. More precisely, to run $[\times s]$ upscaling methods ($[: s]$ downscaling methods, resp.), we generate the input image I^{in} by applying BIC to I^{res} in the opposite $[: s]$ ($[\times s]$, resp.) scaling direction.

Note that, according to such procedure, in testing $[\times s]$ upscaling methods we may find that I^{res} does not have both the dimensions N and M that are divisible for s. In this case, in order to generate I^{in} we use `imresize` with the size option, requiring $n = \lfloor \frac{N}{s} \rfloor$ and $m = \lfloor \frac{M}{s} \rfloor$. Moreover, once obtained I^{in}, both BIC and u-LCI run by specifying again the size $N \times M$ instead of the scaling factor.

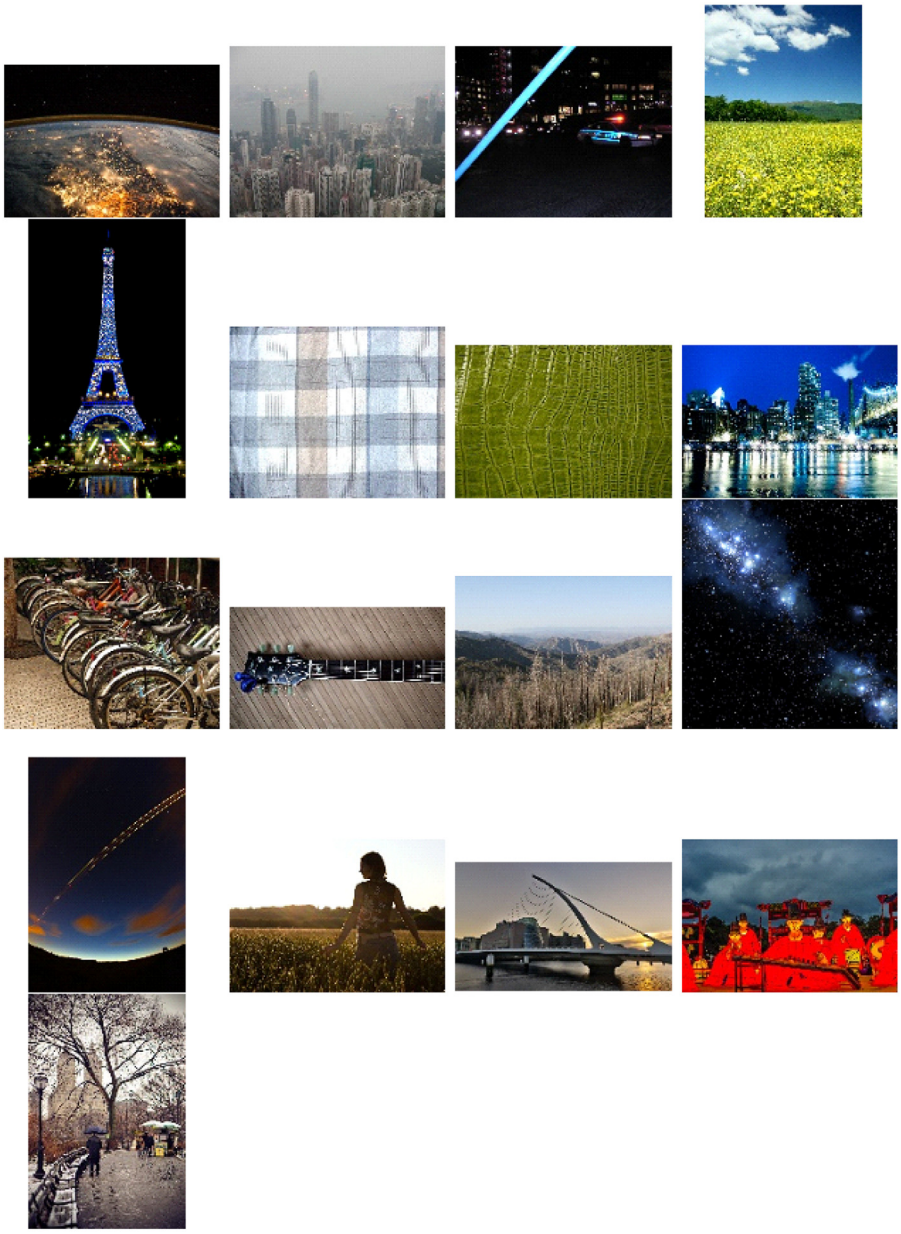

Fig. 7. NY17 dataset images: [N1, N2, N3, N5, N6, N7, N8, N9, N10, N11, N12, N13, N14, N15, N16, N17, N19] from left to right and from top to bottom.

6.3. Visual comparison

As first experiment, we focus on the visual comparison of some original images chosen in the previous datasets with the output images produced by the methods in Section 6.1. Such images are given in Fig. 3 for upscaling and in Figs. 4–5 for downscaling.

In the first column of these figures we show the target images while the successive columns display the resized images obtained as output of the considered methods. Moreover, the first, second, and third row of the figures correspond to the scaling factors 2,3 and 4, respectively. In all the cases, the images have been reported with evidence of some magnified regions of interest (ROI).

Table 4
Pointwise results on NY17 dataset in upscaling.

	x2		x4		x8		x16	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
N1 4928 × 3280								
BIC	37,472	0,945	33,453	0,892	29,828	**0,858**	26,795	**0,833**
u-LCI	**37,903**	**0,950**	**33,856**	0,892	**30,133**	0,855	**26,999**	0,831
N2 3072 × 2304								
BIC	44,235	0,986	37,820	0,952	32,417	0,888	29,065	0,840
u-LCI	**45,203**	**0,988**	**38,621**	**0,957**	**32,943**	**0,894**	**29,183**	0,840
N3 2048 × 1536								
BIC	35,708	0,963	30,845	**0,919**	27,474	**0,874**	24,593	**0,830**
u-LCI	**36,552**	**0,965**	**31,136**	0,916	**27,731**	0,865	**24,850**	0,808
N5 2701 × 3665								
BIC	31,743	0,987	25,444	0,960	22,262	0,932	20,388	0,913
u-LCI	**33,077**	**0,989**	**25,792**	**0,962**	**22,426**	**0,933**	**20,487**	0,913
N6 1836 × 3264								
BIC	33,011	0,986	28,291	**0,965**	24,250	**0,927**	21,677	**0,886**
u-LCI	**33,443**	0,986	**28,779**	0,961	**24,614**	0,908	**21,784**	0,850
N7 1600 × 1200								
BIC	29,939	0,886	25,087	0,670	23,364	0,567	22,627	0,545
u-LCI	**30,981**	**0,905**	**25,409**	**0,684**	**23,542**	**0,569**	**22,683**	**0,546**
N8 3008 × 2000								
BIC	32,298	0,987	29,295	0,974	26,346	0,956	23,345	0,932
u-LCI	**32,577**	**0,988**	**29,594**	**0,976**	**26,619**	**0,958**	**23,508**	**0,933**
N9 5430 × 3520								
BIC	35,440	0,985	28,707	0.945	24,336	0,883	22,494	**0,846**
u-LCI	**36,116**	**0,986**	**30,735**	**0,955**	**26,121**	**0,899**	**22,712**	0,845
N10 3264 × 2448								
BIC	35,758	0,973	28,875	0,916	24,213	0,837	20,914	0,767
u-LCI	**37,047**	**0,977**	**29,531**	**0,920**	**24,529**	**0,838**	**21,187**	**0,768**
N11 4368 × 2326								
BIC	36,525	**0,982**	30,149	**0,955**	24,649	**0,906**	20,966	**0,849**
u-LCI	**37,461**	0,981	**30,986**	0,951	**24,918**	0,897	**21,193**	0,846
N12 3504 × 2336								
BIC	40,471	0,989	32,786	0,951	27,777	0,888	25,116	0,850
u-LCI	**41,889**	**0,991**	**35,581**	**0,956**	**27,982**	**0,890**	**25,191**	**0,851**
N13 1200 × 1600								
BIC	24,145	**0,746**	22,370	**0,603**	21,838	**0,530**	21,555	0,483
u-LCI	**24,316**	0,730	**22,390**	0,592	**21,852**	0,529	**21,579**	**0,486**
N14 3456 × 5184								
BIC	45,041	0,992	40,137	0,986	35,808	0,978	31,982	**0,967**
u-LCI	**45,539**	0,992	**40,907**	0,986	**36,531**	0,978	**32,407**	0,966
N15 3072 × 2048								
BIC	34,457	0,990	29,574	0,967	26,731	**0,941**	24,886	**0,920**
u-LCI	**35,427**	**0,991**	**29,872**	**0,968**	**26,883**	0,940	**24,976**	0,919
N16 5501 × 3095								
BIC	47,929	0,993	40,846	0,982	34,304	0,961	29,760	**0,940**
u-LCI	**48,691**	**0,994**	**42,202**	**0,983**	**35,029**	0,961	**30,027**	0,939

(*continued on next page*)

From the qualitative point of view, by inspecting Figs. 3–5, we can deduce that in terms of visual quality, LCI has a good performance, also with respect to the chosen comparing methods. It preserves the visual structure of

Table 4 (*continued*).

	x2		x4		x8		x16	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
N17 2048 × 1363								
BIC	30,892	0,980	27,028	0,954	24,688	0,927	**22,895**	**0,899**
u-LCI	**31,380**	**0,981**	**27,219**	0,954	**24,834**	0,927	21,044	0,877
N19 3039 × 4559								
BIC	34,581	0,977	27,048	0,884	21,873	0,718	18,910	0,615
u-LCI	**35,844**	**0,981**	**27,881**	**0,896**	**22,178**	**0,723**	**19,077**	**0,617**

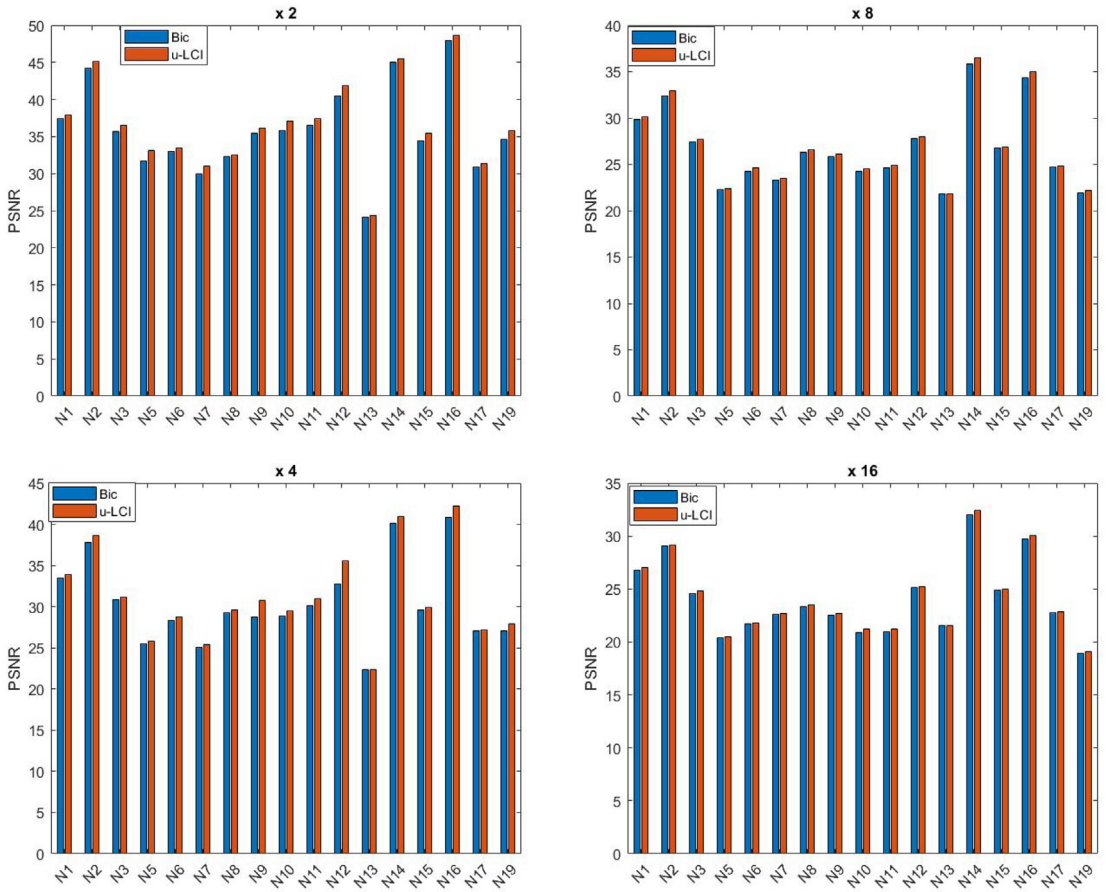

Fig. 8. Pointwise PSNR for NY17 dataset in upscaling with scale factors s = 2, 4, 8, 16.

the object without losing image details, the local contrast, and the luminance of the input image, generating resized images close to the target ones. Ringing and over smoothing artifacts are limited, and the resized images seem fair, sufficiently not blurred, and with well-balanced colors.

6.4. Quantitative comparison: average results

For each dataset selected in Section 6.2, we computed the averages of the PSNR and SSIM values achieved by LCI and the comparison methods described in Section 6.1 for the scaling factors 2, 3, 4. The results are displayed in Table 2 for upscaling and Table 3 for downscaling, reporting in both the tables also the averages of the required CPU time, shortly denoted by T.

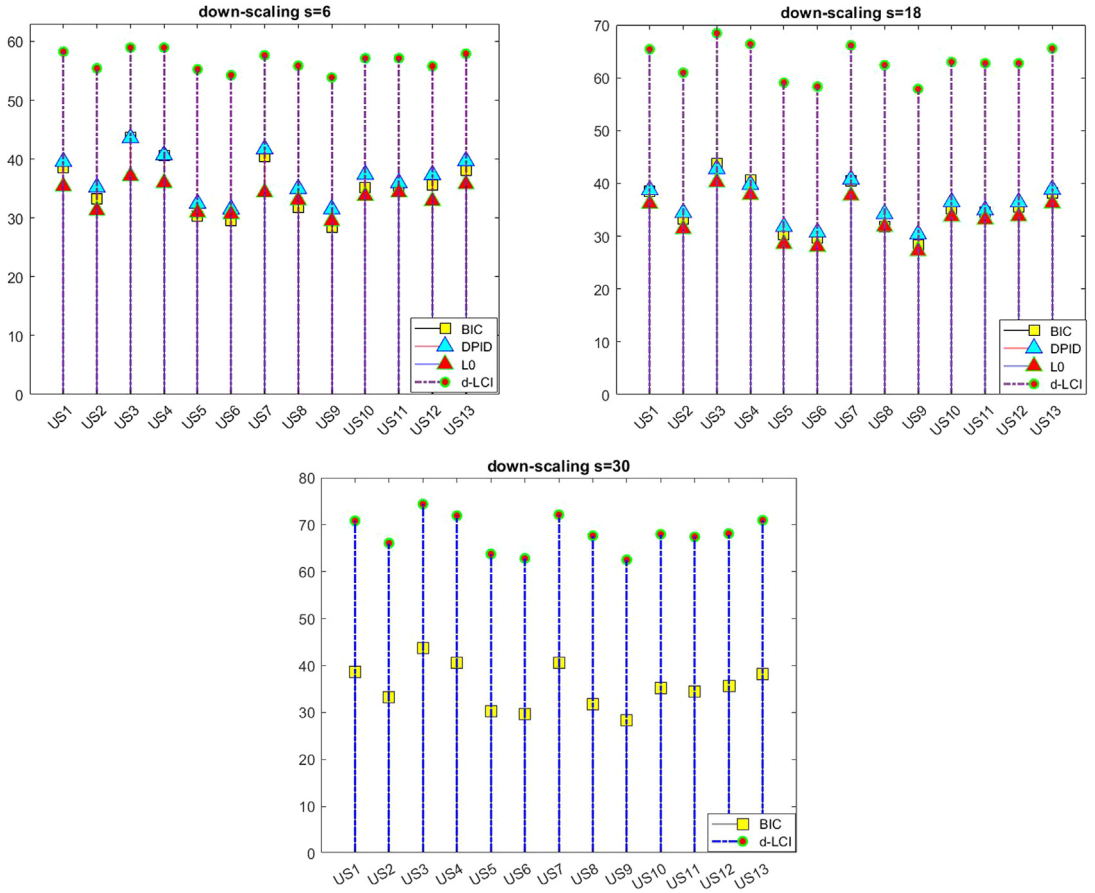

Fig. 9. Pointwise PSNR values for 13US dataset in downscaling with scale factors s = 6, 18, 30. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As announced in the introduction, from Table 2, we see that the values attained by u-LCI are only slightly better than those achieved by BIC, and the execution time of u-LCI is a little bit greater than BIC. Of course, as the scale factor increases, both PSNR and SSIM decrease in all upscaling methods. Moreover, the gaps between BIC and u-LCI in both the metrics are maintained also for large size images.

On the contrary, in downscaling, by inspecting Table 3 we observe that the performance provided by d-LCI is much better than the other methods for both the metrics. In particular, when the downscaling factor is 3 the theoretical results claimed in Proposition 5.1 are confirmed. Moreover, looking at the even downscaling factors 2,4 the improvement by d-LCI seems to increase as the scale factor increases while BIC seems to be affected by saturation, with an increasing gap between d-LCI and the remaining methods.

About the CPU time, except for BIC, d-LCI is faster than the other methods that need very long computational time in the case of very large images (see the results for NY17 and NY96 datasets that include target images of size 6394×3456, i.e., input images with size 25576×13824 in case of a downscaling factor equal 4). Moreover, in some cases, wherever we see oom (which means out of memory), the available code of L_0 does not arrive to produce any result.

6.5. Quantitative comparison: some pointwise results

Here we focus on each single image from smaller datasets of Table 1, namely we consider 13US dataset in downscaling, with target images of small size displayed in Fig. 6, and NY17 dataset in upscaling, with target images of large size displayed in Fig. 7.

Table 5
Pointwise results on 13US dataset in downscaling.

	:3		:6		:9		:18	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
US1 300 × 400								
BIC	38,625	0,996	38,573	0,996	38,583	0,996	38,584	0,996
DPID	40,846	0,997	39,580	0,997	39,160	0,996	38,764	0,996
L_0	27,994	0,949	35,405	0,983	35,138	0,986	36,183	0,991
d-LCI	**Inf**	**1,000**	**58,265**	**1,000**	**Inf**	**1,000**	**65,394**	**1,000**
US2 400 × 300								
BIC	33,348	0,987	33,298	0,987	33,303	0,987	33,302	0,987
DPID	36,428	0,994	35,217	0,992	34,823	0,992	34,450	0,991
L_0	29,184	0,963	31,289	0,967	30,647	0,968	31,394	0,977
d-LCI	**Inf**	**1,000**	**55,418**	**1,000**	**Inf**	**1,000**	**60,956**	**1,000**
US3 300 × 400								
BIC	43,820	0,999	43,754	0,999	43,774	0,999	43,786	0,999
DPID	44,853	0,999	43,589	0,999	43,162	0,999	42,751	0,999
L_0	34,756	0,992	37,140	0,995	38,087	0,996	40,237	0,998
d-LCI	**Inf**	**1,000**	**58,958**	**1,000**	**Inf**	**1,000**	**68,436**	**1,000**
US4 300 × 400								
BIC	40,673	0,996	40,623	0,996	40,631	0,996	40,633	0,996
DPID	42,066	0,998	40,699	0,997	40,236	0,997	39,794	0,996
L_0	32,944	0,973	35,973	0,981	36,453	0,986	37,836	0,992
d-LCI	**Inf**	**1,000**	**58,958**	**1,000**	**Inf**	**1,000**	**66,386**	**1,000**
US5 400 × 300								
BIC	30,357	0,979	30,307	0,979	30,311	0,979	30,311	0,979
DPID	33,492	0,991	32,471	0,989	32,133	0,988	31,813	0,987
L_0	26,807	0,962	30,944	0,973	28,911	0,963	28,551	0,965
d-LCI	**Inf**	**1,000**	**55,257**	**1,000**	**Inf**	**1,000**	**59,065**	**1,000**
US6 300 × 400								
BIC	29,718	0,978	29,659	0,978	29,633	0,978	29,663	0,978
DPID	32,578	0,990	31,486	0,988	31,111	0,987	30,752	0,985
L_0	25,322	0,959	30,666	0,966	28,724	0,958	28,022	0,962
d-LCI	**Inf**	**1,000**	**54,273**	**1,000**	**Inf**	**1,000**	**58,321**	**1,000**
US7 273 × 400								
BIC	40,559	0,995	40,513	0,995	40,521	0,995	40,522	0,995
DPID	43,135	0,997	41,717	0,996	41,232	0,996	40,771	0,996
L_0	31,832	0,949	34,364	0,964	35,604	0,975	37,697	0,987
d-LCI	**Inf**	**1,000**	**57,639**	**1,000**	**Inf**	**1,000**	**66,113**	**1,000**
US8 322 × 400								
BIC	31,857	0,989	31,802	0,989	31,808	0,989	31,808	0,989
DPID	36,382	0,996	34,979	0,994	34,601	0,994	34,239	0,993
L_0	29,562	0,974	33,051	0,977	32,378	0,980	31,828	0,985
d-LCI	**Inf**	**1,000**	**55,880**	**1,000**	**Inf**	**1,000**	**62,394**	**1,000**
US9 322 × 400								
BIC	28,407	0,984	28,350	0,984	28,536	0,984	28,354	0,984
DPID	32,542	0,994	31,504	0,993	34,601	0,992	30,393	0,992
L_0	29,562	0,974	29,504	0,985	27,361	0,980	27,214	0,979
d-LCI	**Inf**	**1,000**	**53,879**	**1,000**	**Inf**	**1,000**	**57,877**	**1,000**

(*continued on next page*)

The PSNR values achieved for every single image have been plotted in Fig. 8 for NY17 dataset with upscaling factor 2, 4, 8, 16, and in Fig. 9 for 13US dataset with downscaling factor 6, 18, 30.

On the same datasets, more detailed results are given in Tables 4–6 reporting in the first columns the name and the size of all images from 13US and NY17. In particular, Table 4 contains the detailed upscaling results on NY17

Table 5 (*continued*).

	:3		:6		:9		:18	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
US10 400 × 307								
BIC	35,220	0,992	35,168	0,992	35,172	0,992	35,173	0,992
DPID	38,947	0,997	37,406	0,996	36,929	0,995	36,494	0,995
L_0	31,031	0,984	33,760	0,986	33,227	0,985	33,744	0,988
d-LCI	**Inf**	**1,000**	**57,131**	**1,000**	**Inf**	**1,000**	**62,986**	**1,000**
US11 400 × 241								
BIC	34,599	0,995	34,528	0,995	34,536	0,995	34,535	0,995
DPID	37,345	0,998	35,950	0,997	35,432	0,997	34,937	0,996
L_0	27,631	0,976	34,346	0,986	34,143	0,989	33,189	0,992
d-LCI	**Inf**	**1,000**	**57,147**	**1,000**	**Inf**	**1,000**	**62,744**	**61,000**
US12 400 × 266								
BIC	35,698	0,993	35,639	0,993	035,645	0,993	35,647	0,993
DPID	38,626	0,997	37,345	0,996	36,898	0,995	36,471	0,995
L_0	29,883	0,972	32,902	0,978	33,091	0,982	33,776	0,987
d-LCI	**Inf**	**1,000**	**55,801**	**1,000**	**Inf**	**1,000**	**62,744**	**1,000**
US13 310 × 400								
BIC	38,220	0,996	38,166	0,996	38,173	0,996	38,177	0,996
DPID	41,001	0,998	39,707	0,997	39,276	0,997	38,861	0,997
L_0	31,593	0,982	35,761	0,989	35,898	0,991	36,263	0,993
d-LCI	**Inf**	**1,000**	**57,905**	**1,000**	**Inf**	**1,000**	**65,535**	**1,000**

dataset, for the scaling factors $s \in \{2, 4, 8, 16\}$ while Tables 5–6 concern the downscaling results obtained on 13US dataset for the scaling factors $s \in \{3, 6, 9, 18\}$ (Table 5) and for the larger factors $s \in \{20, 21, 27, 30, 33\}$ (Table 6).

According to the average results, the pointwise ones corroborate the global trend. In particular, Table 5 confirms that in downscaling d-LCI is the method allowing the largest PSNR and SSIM. Also, note that the scaling factors chosen in Table 5 are all multiple of three but, according to Proposition 5.1 we can see the difference between the odd and even downscaling factors. Moreover, we can affirm that d-LCI works fine also for large downscaling factors as reported in Table 6 where the target images in 13US have been zoomed in up to 33 times, getting input images whose size goes up to 13200 × 10230. We point out that in Table 6 the results for DPID and L_0 methods are missing because the publicly available Matlab codes do not work for so large scale factors. Moreover, if $s \in \{21, 27, 33\}$ then, in accordance with Proposition 5.1, PSNR and SSIM again reach the limit values (infinite and 1, respectively) by d-LCI while the same does not hold for BIC applied to the same input images.

7. Conclusions

We present a novel approach to the image resizing problem, based on a non standard mathematical modeling. According to such a model, we are led to approximate the image at the desired size by means of global interpolation processes based on (tensor product) Chebyshev grids of I kind. In particular, we present LCI method that takes the pixel values of any (color or grayscale) input image to built up the corresponding bivariate Lagrange–Chebyshev (I kind) interpolation polynomial, whose sampling gives the output resized image. The explicit formulas for the pixels values of the image resized by LCI are given by (13) for grayscale images and by (24) for color images.

One of the main advantages of LCI method is its high flexibility of working in both the scaling directions, either setting a scale factor or giving a particular final size.

Moreover, odd scale factors are special cases for LCI: some pixels values of the input and output image coincide and by (29) we give the precise position of such equal pixels. We underline that this feature can be an advantage when the input image is "close" to the original, but it is a disadvantage when the image has a "large" MSE. In particular, for all image downscaling with an odd scale factor s, we prove that LCI produces an output image whose MSE is not greater than s^2 times the MSE of the input image (cf. Proposition 5.1 and the successive remark).

We evaluate the LCI performance in terms of the PSNR and SSIM quality metrics, measuring the CPU time too. Comparisons with other resizing procedures have been reported on 6 different (commonly available) datasets composed of 1026 images in total.

Table 6
Performance results on 13US dataset for high downscaling.

	:20		:21		:27		:30		:33	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
US1 300 × 400										
BIC	38,583	0,996	38,585	0,996	38,586	0,996	38,585	0,996	38,584	0,996
d-LCI	**66,485**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**70,845**	**1,000**	**Inf**	**1,000**
US2 400 × 300										
BIC	33,302	0,987	33,302	0,987	33,302	0,987	33,303	0,987	33,303	0,987
d-LCI	**66,347**	**1,000**	**Inf**	**1,000**	**Inf**	**66,092**	**1,000**	**1,000**	**Inf**	**1,000**
US3 300 × 400										
BIC	43,786	0,999	43,787	0,999	43,786	0,999	43,788	0,999	43,784	0,999
d-LCI	**69,721**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**74,415**	**1,000**	**Inf**	**1,000**
US4 300 × 400										
BIC	40,633	0,996	40,634	0,996	40,636	0,996	49,636	0,996	49,635	0,996
d-LCI	**68,054**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**71,936**	**1,000**	**Inf**	**1,000**
US5 400 × 300										
BIC	30,311	0,979	30,311	0,979	30,311	0,979	30,311	0,979	30,311	0,979
d-LCI	**59,853**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**63,768**	**1,000**	**Inf**	**1,000**
US6 300 × 400										
BIC	30,311	0,979	29,663	0,978	29,664	0,978	29,663	0,978	29,663	0,978
d-LCI	**59,135**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**62,833**	**1,000**	**Inf**	**1,000**
US7 273 × 400										
BIC	40,524	0,995	40,521	0,995	40,524	0,995	40,536	0,995	40,520	0,995
d-LCI	**67,318**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**72,151**	**1,000**	**Inf**	**1,000**
US8 322 × 400										
BIC	31,808	0,989	31,807	0,989	31,807	0,989	31,807	0,995	31,806	0,989
d-LCI	**63,539**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**67,653**	**1,000**	**Inf**	**1,000**
US9 322 × 400										
BIC	28,354	0,984	28,355	0,984	28,354	0,984	28,355	0,984	28,355	0,984
d-LCI	**58,651**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**62,536**	**1,000**	**Inf**	**1,000**
US10 400 × 307										
BIC	35,174	0,992	35,172	0,992	35,174	0,992	35,175	0,992	35,175	0,992
d-LCI	**63,924**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**67,994**	**1,000**	**Inf**	**1,000**
US11 400 × 241										
BIC	34,535	0,995	34,535	0,995	34,534	0,995	34,534	0,995	34,535	0,995
d-LCI	**63,445**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**67,442**	**1,000**	**Inf**	**1,000**
US12 400 × 266										
BIC	35,647	0,993	35,646	0,993	35,645	0,993	35,646	0,993	35,648	0,993
d-LCI	**63,760**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**67,127**	**1,000**	**Inf**	**1,000**
US13 310 × 400										
BIC	38,175	0,996	38,175	0,996	38,174	0,996	38,175	0,996	38,175	0,996
d-LCI	**66,347**	**1,000**	**Inf**	**1,000**	**Inf**	**1,000**	**70,960**	**1,000**	**Inf**	**1,000**

The numerical experience in upscaling shows a performance comparable with the bicubic interpolation method, while in downscaling a much better performance is obtained with respect to all the benchmarking methods. Moreover, in all downscaling with odd scale factors, d-LCI method is the unique to reach the best possible PSNR and SSIM measures, confirming the theoretical results.

One limitation of LCI concerns the CPU time since our Matlab (non optimized) code is a little bit slower than `imresize`, i.e., the Matlab optimized version of bicubic interpolation method (BIC). Nevertheless, it is definitely faster than the other benchmarking downscaling methods (DPID and L_0) and we believe that there is room to improve this aspect by adopting an optimized algorithm scheme. In particular, we wonder whether further improvements can be achieved by employing wavelets technique too.

Finally, we remark that the proposed mathematical approach leaves the way open to future developments and possible improvements by using finer and more recent global approximation tools based on Chebyshev grids.

References

[1] https://www.gcc.tu-darmstadt.de/home/proj/dpid/index.en.jsp.
[2] http://www.shengfenghe.com/publications/.
[3] http://sipi.usc.edu/database/.
[4] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/.
[5] https://www.nasa.gov/multimedia/imagegallery/index.html.
[6] https://www.cl.cam.ac.uk/~aco41/Files/Sig15UserStudyImages.html.
[7] https://www.pexels.com/search/color/.
[8] R.S. Asamwar, K.M. Bhurchandi, A.S. Gandhi, Interpolation of images using discrete wavelet transform to simulate image resizing as in human vision, Int. J. Autom. Comput. 7 (1) (2010) 9–16.
[9] P.M. Atkinson, Downscaling in remote sensing, Int. J. Appl. Earth Obs. Geoinf. 22 (2013) 106–114.
[10] L. Bos, M. Caliari, S. De Marchi, M. Vianello, Y. Xu, Bivariate Lagrange interpolation at the padua points: the generating curve approach, J. Approx. Theory 143 (2006) 15–25.
[11] V. Bruni, G. Ramella, D. Vitulano, Automatic perceptual color quantization of dermoscopic images, in: J. Braz, et al. (Eds.), VISAPP 2015, Vol. 1, Scitepress Science and Technology Publications, 2015, pp. 323–330.
[12] J. Chaki, N. Dey, Introduction to image color feature, in: Springer-Briefs in Applied Sciences and Technology, Singapor, 2021.
[13] G. Chen, H. Zhao, C.K. Pang, T. Li, C. Pang, Image scaling: How hard can it be? IEEE Access 7 (2019).
[14] S. De Marchi, W. Erb, F. Marchetti, Spectral filtering for the reduction of the gibbs phenomenon for polynomial approximation methods on lissajous curves with applications in MPI, Dolom. Res. Notes Approx. 10 (2017) 128–137.
[15] B. Fischer, W. Themistoclakis, Orthogonal polynomial wavelets, Numer. Algorithms 30 (2002) 37–58.
[16] J.-B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transforme self-exemplars, in: Proc. CVPR, 2015, pp. 5197–5206.
[17] C. Kaethner, W. Erb, M. Ahlborg, P. Szwargulski, T. Knopp, T.M. Buzug, Non-equispaced system matrix acquisition for magnetic particle imaging based on lissajous node points, IEEE Trans. Med. Imaging 35 (11) (2016) 2476–2485.
[18] H.B. Kekre, T. Sarode, S. Natu, Image zooming using sinusoidal transforms like hartley, DFT, DCT, DST and real Fourier transform, Int. J. Comput. Sci. Inf. Secur. (2014) 11–16.
[19] R. Keys, Cubic convolution interpolation for digital image processing, IEEE Trans. Acoust. Speech Signal Process. 29 (6) (1981) 1153–1160.
[20] T. Knopp, T.M. Buzug, Magnetic Particle Imaging, Springer-Verlag Berlin Heidelberg, Berlin, 2012.
[21] J. Liu, S. He, R.W.H. Lau, L_0-regularized image downscaling, IEEE Trans. Image Process. 27 (3) (2018) 1076–1085.
[22] H. Liu, X. Xie, W.Y. Ma, H.J. Zhang, Automatic browsing of large pictures on mobile devices, in: 11th ACM International Conference on Multimedia, Berkeley, CA, USA, 2003.
[23] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, H.-Y. Shum, Learning to detect a salient object, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 353–367.
[24] A. Lookingbill, J. Rogers, D. Lieb, J. Curry, S. Thrun, Reverse optical flow for self-supervised adaptive autonomous robot navigation, Int. J. Comput. Vis. 74 (3) (2007) 287–302.
[25] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proc. 8th Int. Conf. Computer Vision, Vol. 2, 2001, pp. 416–423.
[26] E.H.W. Meijering, W.J. Niessen, M.A. Viergever, Quantitative evaluation of convolution-based methods for medical image interpolation, Med. Image Anal. 5 (2001) 111–126.
[27] H. Mittal, A.C. Pandey, M. Saraswat, et al., A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets, Multimedia Tools Appl. (2021).
[28] C.H. Neetha, C.J. Moses, D. Selvathi, Image interpolation using non-adaptive scaling algorithms for multimedia applications. a survey, in: V. Komanapalli, N. Sivakumaran, S. Hampannavar (Eds.), Advances in Automation, Signal Processing, Instrumentation, and Control, Vol. 700, 2021, pp. 1509–1516.
[29] D. Occorsio, W. Themistoclakis, Uniform weighted approximation on the square by polynomial interpolation at Chebyshev nodes, Appl. Math. Comput. 385 (125457) (2020).
[30] D. Occorsio, W. Themistoclakis, On the filtered polynomial interpolation at Chebyshev nodes, Appl. Numer. Math. 166 (2021) 272–287.
[31] D. Occorsio, W. Themistoclakis, Some remarks on filtered polynomial interpolation at Chebyshev nodes, Dolom. Research. Notes on Approx. 14 (2021) 68–84.
[32] A.C. Oztireli, M. Gross, Perceptually based downscaling of images, ACM Trans. Graph. 34 (4) (2015) 1–10.
[33] G. Plonka, D. Potts, G. Steidl, M. Tasche, Numerical Fourier Analysis, in: Applied and Numerical Harmonic Analysis, Birkhüser Springer Nature Switzerland AG, Berlin.
[34] G. Ramella, Evaluation of quality measures for color quantization, Multimedia Tools Appl. 80 (2021) 32975–33009, http://dx.doi.org/10.1007/s11042-021-11385-y.
[35] G. Ramella, G. Sanniti di Baja, From color quantization to image segmentation, in: Signal Imag. Techn. Internet-Based Syst. - SITIS 2016, 2016, pp. 798–804.
[36] G. Ramella, G. Sanniti di Baja, A new method for color quantization, in: Signal Imag. Techn. Internet-Based Syst. - SITIS 2016, 2016, pp. 1–6.

[37] B. Thomee, D.A. Shamma, G. Friedland, B. Elizalde, D. Poland, K. Ni, D. Borth, L.-J. Li, Yfcc100m: The new data in multimedia research, Commun. ACM 59 (2) (2016) 64–73.
[38] L. Trefethen, Approximation Theory and Approximation Practices, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2013.
[39] T. Vlašic, I. Ralašic, A. Tafro, D. Seršic, Spline-like Chebyshev polynomial model for compressive imaging, J. Vis. Commun. Image 66 (102731) (2020) 370–378.
[40] N. Weber, M. Waechter, S.C. Amend, S. Guthe, M. Goesele, Rapid detail-preserving image downscaling, ACM Trans. Graph. 35 (6) (2016) 1–6.
[41] Y. Xu, Lagrange interpolation on Chebyshev points of two variables, J. Approx. Theory 87 (2) (1996) 220–238.
[42] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, Q. Liao, Deep learning for single image super-resolution: A brief review, IEEE Trans. Multimedia 21 (12) (2019).
[43] T. Yao, Y. Luo, Y. Chen, D. Yang, L. Zhao, Single-image super-resolution: A survey, in: L. Q., L. X., N. Z., W. W., M. J., Z. B. (Eds.), CSPS 2018, Vol. 516, 2020, pp. 119–125.
[44] M. Zhang, L. Zhang, Y. Sun, L. Feng, W. Ma, Auto cropping for digital photographs, in: K. Yetongnon, et al. (Eds.), Signal Imag. Techn. Internet-Based Syst. - SITIS 2016, 2005.
[45] D.-X. Zhou, Theory of deep convolutional neural networks: Downsampling, Neural Netw. 124 (2020) 319–327.