An Automatic Image Scaling Up Algorithm

Maria Frucci, Carlo Arcelli, and Gabriella Sanniti di Baja

Istituto di Cibernetica “E.Caianiello”, CNR
Via Campi Flegrei 34, 80078 Pozzuoli, Naples, Italy
{m.frucci,c.arcelli,g.sannitidibaja}@cib.na.cnr.it

Abstract. A fully automatic scaling up algorithm is presented in the framework of interpolation methods. For any integer zooming factor n , the algorithm generates a magnified version of an input color image in one scan of the image. The computational complexity of the algorithm is $O(N)$, where N is the size of the input image. The visual aspect of the magnified images is generally appealing also when considering large zooming factors. Peak Signal to Noise Ratio and Structural SIMilarity are used to evaluate the performance of the algorithm and to compare it with other scaling up algorithms.

Keywords: digital images, color images, zooming, interpolation.

1 Introduction

Scaling up addresses the problem of generating a high-resolution image HR, of which a low-resolution version LR is available. The goal is to provide a reasonably natural and artifact free magnified image with a low computational complexity. This aim can generally be achieved for small magnification rates, while some perturbation becomes unavoidable when greatly enlarging the image.

One of the reasons for which scaling up receives much attention is that nowadays many digital images are transferred via Internet to different media. Often these images have small size, but it may be necessary to increase their size to better fit the layout of the documents where the images are inserted. Another reason is the increasing number of applications where scaling up is useful, e.g., medical imaging, astronomical imaging, digital photographs, remote sensing and entertainment.

Interpolation methods are commonly adopted scaling up techniques [1, 2]. Linear techniques (e.g., bilinear and bicubic interpolation) cause a blocky appearance of HR or produce blurred edges, but are still often employed for their computational simplicity. Other methods include non-linear interpolation techniques [3,4], adaptive interpolation techniques [5-8], Bayesian maximum a posteriori approach [9], wavelet approach [10], fractal-based approach [11], and PDEs-based approach [12].

In this paper, we describe a simple discrete algorithm in the framework of interpolation methods, to magnify color images for any integer zooming factor n . The suggested Fully Automatic Scaling up algorithm, called FAS in the following, builds HR in a single inspection of LR, during which tentative values for the $n \times n$ pixels of HR associated with each single pixel p of LR are computed and suitably updated to

their final values. This algorithm is an improvement of our previous algorithm WAZ suggested in [13]. While in this paper tentative values are computed as in [13], final values are obtained by a new updating criterion producing significantly better results.

FAS works in any color space, with any integer zooming factor, does not involve any threshold, produces visually appealing magnified images, and requires a reduced computational effort. The performance of FAS and of other scaling up algorithms has been evaluated on 50 images, taken from publicly available repositories, in terms of Peak Signal to Noise Ratio (PSNR) [14] and Structural SIMilarity (SSIM) [15].

2 The Scaling Up Algorithm FAS

FAS can be applied to images in any color space, by using exactly the same one-raster scan procedure to all channel components. In the following we will refer to the RGB color space. The RGB input image LR is split into the three channel components that are processed as gray-level images. The HR color image is obtained by composing the three magnified gray-level images. Since the same procedure is applied to each channel component, we limit the description to the case of a single gray-level image.

Once the zooming factor n has been chosen, each pixel p of LR is associated with $n \times n$ pixels q_k , $k = 1, \dots, n \times n$, in the scaled up image HR. The value of each q_k depends on the values of the generating pixel p and of suitable neighbors of p in LR. In the following, we denote the neighbors of p as tl (top left), t (top), tr (top right), r (right), br (bottom right), b (bottom), bl (bottom left) and l (left). Moreover, let m be the smallest odd integer number larger than n (i.e., $m = n + 1$ for n even, $m = n + 2$ for n odd). We will use an $m \times m$ sliding window centered on each pixel q_k of the $n \times n$ block to record the appropriate multiplicative weights to be used for the computation of the value of q_k in terms of the values of p and of its eight neighbors. Thus, for each q_k a 3×3 mask of nine weights has to be computed. Since $n \times n$ pixels q_k form the block, in total $n \times n$ masks of nine weights have to be computed.

Fig. 1. The nine $n \times n$ blocks ($n=3$ left, $n=4$ right) associated in HR to a pixel p and to its eight neighbors in LR, are framed by thick black lines. The 5×5 sliding window, centered on pixel q_5 , is shown with a gray background.

For any p in LR we consider the nine $n \times n$ blocks in HR associated to p and to its eight neighbors. Then, we move the $m \times m$ sliding window in forward fashion from top left (pixel q_1) to bottom right (pixel $q_{n \times n}$) and count the number of pixels in the intersection of each of the nine blocks with the sliding window centered on the current q_k . The so obtained numbers are the multiplicative weights for p and for its eight neighbors for the computation of the current q_k . While the intersection between the sliding window and the block corresponding to p is always non-empty, whichever is the current q_k , an empty intersection may occur, for some q_k , when considering the blocks corresponding to certain neighbors of p . For such neighbors of p , the weights result to be equal to zero. In other words, only the neighbors of p for which a non-empty intersection exists actually contribute together with p to the computation of the values of the $n \times n$ pixels q_k .

Fig. 1 shows two examples in the two cases $n=3$ and $n=4$ (for both of which it is $m=5$). The nine $n \times n$ blocks are framed by thick black lines and the $m \times m$ sliding window, centered in both cases on q_5 , is shown with a gray background.

It can be verified that when n is even, only three neighbors of p contribute to the value of each q_k . If n is odd, the number of contributing neighbors ranges from three to eight, depending on the position of q_k . For example, with reference to the case $n=3$ in Fig. 1 left, we note that the nine blocks associated to p and to all its neighbors overlap the $m \times m$ window centered on q_5 . The weights for computing the value of q_5 are: 9 for p ; 3 for t , l , b , and r ; and 1 for tr , br , bl and tl . This is the information on the weights to be recorded for q_5 for $n=3$. Analogously, with reference to the case $n=4$ in Fig. 1 right, the only three neighbors of the generating p having their blocks overlapping the $m \times m$ window centered on q_5 are tl , t , and l . The weights for computing the value of q_5 are: 12 for p ; 2 for tl ; 3 for t ; 8 for l ; and 0 for all the other neighbors of p . This is the information to be recorded for q_5 for $n=4$.

For $n=3$ ($n=4$), Fig. 2 shows the nine (sixteen) masks of nine weights, to be used for the nine (sixteen) q_k associated in HR to any generating pixel p . Each mask gives the multiplicative weights for p and for each of its eight neighbors in LR. Scaling up actually starts once all $n \times n$ masks have been built.

4	6	0	2	6	2	0	6	4
6	9	0	3	9	3	0	9	6
0	0	0	0	0	0	0	0	0
2	3	0	1	3	1	0	3	2
6	9	0	3	9	3	0	9	6
2	3	0	1	3	1	0	3	2
0	0	0	0	0	0	0	0	0
6	9	0	3	9	3	0	9	6
4	6	0	2	6	2	0	6	4

4	6	0	2	8	0	0	8	2	0	6	4
6	9	0	3	12	0	0	12	3	0	9	6
0	0	0	0	0	0	0	0	0	0	0	0
2	3	0	1	4	0	0	4	1	0	3	2
8	12	0	4	16	0	0	16	4	0	12	8
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
8	12	0	4	16	0	0	16	4	0	12	8
2	3	0	1	4	0	0	4	1	0	3	2
0	0	0	0	0	0	0	0	0	0	0	0
6	9	0	3	12	0	0	12	3	0	9	6
4	6	0	2	8	0	0	8	2	0	6	4

Fig. 2. Masks of weights for the computation of the pixels q_k for $n=3$, left, and $n=4$, right

FAS generates HR in a single inspection of LR. Two processes are accomplished for each currently inspected pixel p of LR, during the same scan of LR. The first process is devoted to the computation of tentative values for the pixels q_k . The tentative value of each q_k is computed as the weighted arithmetic mean of the generating pixel p and its neighbors in LR by using the weights of the k -th mask. The second process (updating) is devoted to the computation of the final value for each q_k . We aim at final values of the $n \times n$ pixels q_k such that their arithmetic mean is as close as possible to the value of the generating pixel p . To this purpose, updating is done in two steps, as described below.

During the first step, the arithmetic mean μ_1 of the tentative values in the $n \times n$ block is computed as well as the absolute value of the difference Δ_1 between p and μ_1 . If $p = \mu_1$, the tentative values are not modified. Otherwise, depending on whether $p < \mu_1$ or $p > \mu_1$, Δ_1 is subtracted from or is added to the value of each q_k . If the result of this subtraction/addition is in the range [0-255], the result is taken as updated value of q_k . If the result is outside the range [0-255], the value of q_k is not modified.

During the second updating step, the arithmetic mean μ_2 of the values in the $n \times n$ block is computed as well as the absolute value of the difference Δ_2 between p and μ_2 , and Δ_2 is subtracted/added as before. The result of this subtraction/addition is taken as final value of q_k , if is in the range [0-255]. When the resulting value is less than 0 (larger than 255), such a value is updated to the final value 0 (255).

The rationale of the first updating step is that when $p < \mu_1$ the tentative values result in the average larger than desirable and, hence, the tentative values are decreased. In turn, when $p > \mu_1$ the tentative values are generally smaller than desired and, hence, the tentative values are increased. If all pixels in a block are modified during the first step, their arithmetic mean almost coincides with the value of p . In turn, if some tentative values are not modified, the arithmetic mean within the block may differ significantly from the value of p . Thus, the second updating step is profitable with the purpose of making the resulting arithmetic mean closer to p . One could argue that only one updating step could be performed, where values falling outside the range [0-255] are directly set to 0 and 255. However, we have experimentally verified that the refinement produced by using the updating process in two steps improves the performance of FAS in terms of PSNR and SSIM.

3 Experimental Results

FAS has been tested by using different magnification rates on a dataset of 50 RGB images with different resolutions, taken from public databases [16,17]. As an example, eight target images with resolution 512×512 are shown in Fig. 3.

To evaluate the performance of FAS in terms of PSNR and SSIM, we build LR images (sources) from the images of the dataset (targets). Different sets of sources are considered, depending on the magnification rate. To build the sources, a decimation process is used to assign to each pixel of the source image a value equal to the arithmetic mean in the $n \times n$ block of the target image corresponding to that pixel.

We compare the performance of FAS with that of other methods available on a commercial software, where several interpolation algorithms are implemented on the SAR Image Processor package [18]. The methods taken from SAR are: Bilinear

Interpolation (BL), Bicubic Interpolation (BC), Lanczos (L), and non linear methods such as Pseudo Inverse (PI), Pseudo LAD (LAD), New Data Dependent Lanczos (DDL).

Fig. 3. Target images

As concerns the images in Fig.3, PSNR (expressed in decibel) and SSIM (evaluated in the range [0,100], where 100 means excellent), computed when using FAS, BL, BC, L, PI, LAD and DDL for $n=2$, are given in Table 1 and Table 2, respectively. The largest values are in boldface. It can be seen that in the majority of cases both PSNR and SSIM are larger for FAS.

Table 1. PSNR for $n=2$

	FAS	BL	BC	L	PI	LAD	DDL
baboon	22.967	22.220	22.749	22.806	22.525	22.786	22.464
butfish	26.804	25.594	26.464	26.671	26.394	26.550	26.105
desert	21.188	20.476	20.977	21.053	20.803	20.915	20.713
frog	32.794	31.501	32.453	32.709	32.362	32.691	32.335
girl	30.629	29.883	30.347	30.475	30.125	29.865	30.337
lena	32.907	31.534	32.638	32.979	32.59	33.044	32.503
st_elmo	21.740	20.763	21.432	21.557	21.423	21.655	21.352
toucan	39.762	37.329	37.409	37.672	36.804	37.616	37.392

Table 2. SSIM for $n=2$

	FAS	BL	BC	L	PI	LAD	DDL
baboon	71.150	61.546	67.362	68.488	69.451	69.635	68.460
butfish	80.586	73.832	78.052	78.782	76.982	78.782	77.286
desert	64.600	54.164	60.220	61.507	63.182	63.030	62.609
frog	92.342	90.618	91.831	91.976	86.819	90.397	88.772
girl	87.543	84.783	86.527	86.758	83.782	86.231	84.982
lena	87.171	84.140	86.249	86.590	84.083	86.286	85.390
st_elmo	70.328	59.931	66.078	67.132	68.621	68.943	67.973
toucan	96.890	96.071	94.767	94.860	90.165	94.332	93.665

We have also used the L_I norm criterion to compare the results obtained by FAS, BL, BC, L, PI, LAD and DDL to the targets. In Fig. 4, the average values on the $n=2$ zoomed images are given, clearly showing that FAS has a better performance.

Fig. 4. Average values of the comparisons between the results obtained by FAS, BL, BC, L, PI, LAD and DDL and the targets for $n=2$ with the L_I norm criterion

We also compare the performance of FAS, BL, BC, L, PI, LAD and DDL when the sources are built by applying to the targets different decimation processes. Generally, the performance of the various methods only very slightly changes when a different decimation process is used to build the LR images, and in all cases the results produced by FAS are almost always better. As an example, see Table 3 and Table 4, where PSNR and SSIM are shown in the case $n=2$ and for LR images generated by applying to the target images bilinear interpolation with a reduction factor 0.5.

Table 3. PSNR for $n=2$ (sources obtained by means of bilinear interpolation)

	FAS	BL	BC	L	PI	LAD	DDL
baboon	22.966	22.220	22.749	22.806	22.526	22.786	22.464
butfish	26.802	25.593	26.464	26.669	26.393	26.550	26.104
desert	21.188	20.475	20.976	21.052	20.803	20.915	20.713
frog	32.788	31.500	32.454	32.710	32.361	32.691	32.337
girl	30.626	29.881	30.346	30.474	30.127	29.865	30.338
lena	32.898	31.533	32.638	32.979	32.589	33.045	32.500
st_elmo	21.739	20.763	21.432	21.556	21.422	21.656	21.351
toucan	39.726	37.925	39.533	40.026	38.721	40.045	39.568

Table 4. SSIM for $n=2$ (sources obtained by means of bilinear interpolation)

	FAS	BL	BC	L	PI	LAD	DDL
baboon	71.153	61.547	67.365	68.487	69.454	69.638	68.422
butfish	80.575	73.816	78.042	78.771	76.960	78.771	76.262
desert	64.602	54.162	60.224	61.511	63.179	63.073	62.600
frog	92.339	90.608	91.834	91.978	86.803	90.403	88.793
girl	87.556	84.786	86.531	86.763	83.795	86.245	84.990
lena	87.176	84.142	86.250	86.598	84.084	86.302	85.392
st_elmo	70.323	59.923	66.073	67.131	68.618	68.938	67.966
toucan	96.848	96.046	96.881	97.041	92.139	96.554	95.917

The average values of PSNR and SSIM on the whole dataset for FAS, BL, BC, L, PI, LAD, DDL, and WAZ are given in Table 5, for $n=2$ and $n=4$.

Table 5. Average values of PSNR and SSIM

	FAS	BL	BC	L	PI	LAD	DDL	WAZ
PSNR, $n=2$	29.025	27.791	28.667	28.962	28.743	28.805	28.625	28.53
SSIM, $n=2$	82.648	77.196	80.185	81.494	80.053	81.621	80.760	80.25
PSNR, $n=4$	25.023	24.546	24.859	24.967	24.885	25.258	24.936	24.81
SSIM, $n=4$	63.888	60.538	62.302	62.655	61.271	64.657	62.953	61.74

For $n>2$, FAS still favorably compares to BL, BC, L, PI, DDL and WAZ, while LAD may perform better. This is partially due to the different behavior of LAD and FAS in correspondence of edges. In this respect, let us consider the two synthetic sources of Fig. 5, showing two uniform regions with gray-levels 25 (dark gray) and 225 (light gray) separated by a horizontal edge (Fig. 4 top) and a diagonal edge (Fig. 4 bottom), respectively. A sharp edge is desirable in the scaled up images.

By looking at the performance diagrams shown in Fig. 5 for FAS and LAD, along a vertical line of the magnified images obtained from the synthetic sources for $n=4$, we note that LAD has a better behavior as far as the sharpness of the horizontal edge is concerned, while both algorithms show a similar distortion for the diagonal edge.

Fig. 5. Synthetic images with a horizontal edge, top, and a diagonal edge, bottom, and corresponding performance diagrams for FAS, left, and LAD, right, for $n=4$

Still with reference to Table 5, we point out that the increment of PSNR for FAS with respect BL, BC, L, PI, LAD, DDL and WAZ is often less than 0.5. Since according to the MPEG committee an increase of 0.5 should characterize the PSNR to obtain a visible improvement, one could argue that FAS does not significantly

outperform the other methods. However, we think that PSNR is not the only parameter to judge a method and that also other qualitative features, e.g., the visual appearance of edges, should be taken into account. For example, compare Fig. 6 left with Fig. 6 right where the synthetic image of Fig. 5 top is shown magnified by a zooming factor $n=4$ by FAS and BC, respectively. In both cases, the gray-levels do not monotonically change from 25 to 225 in the magnified images. However, a more unnatural appearance of the edge characterizes the BC magnified image.

Fig. 6. FAS magnification, left, and BC magnification, right, of the image in Fig. 4 top

Nowadays, some authors make their software available in the web for the scientific community. For example, the Matlab code of iNEDI [19], which is a popular method derived and improved from [3], can be downloaded at <http://inedi.tecnick.com/>. The results obtained by iNEDI are definitely appreciable. However, a quantitative comparison between FAS and iNEDI cannot be accomplished since iNEDI returns images whose size is slightly smaller than the size of the target images. In any case, the time requested by iNEDI is very high, as pointed out by the authors (nearly eight minutes for iNEDI and just a few seconds for FAS, for $n=2$ and a 128×128 source image). Moreover, only even zooming factors are possible for iNEDI, both odd and even zooming factors can be used with FAS.

Table 6. Comparisons for $n=2$ and gray-level images

	PSNR			SSIM		
	FAS	KR	ARI	FAS	KR	ARI
baboon	23.343	22.286	22.324	75.663	63.281	66.044
butfish	27.137	24.923	24.857	83.996	74.261	75.772
desert	21.195	20.186	20.075	69.612	56.017	58.566
frog	33.466	31.419	31.215	91.438	88.860	88.983
girl	31.680	29.451	29.869	89.875	86.431	86.732
lena	33.885	30.431	30.351	92.371	88.482	88.715
st_elmo	21.715	20.096	19.981	73.979	59.908	61.824
toucan	39.171	35.391	35.143	96.712	95.119	95.255

Other three methods for which the software has been provided by the Authors are: 1) Sparse Mixing Estimators SME [7] (<http://www.cmap.polytechnique.fr/~mallat/SME.htm>), 2) Kernel Regression KR [5] (<http://users.soe.ucsc.edu/~htakeda/Kernel>)

ToolBox.htm), and Autoregressive Image Interpolation ARI [6] (<http://wavesharp.com/interpolation/>). The above three algorithms process gray-level images. Thus, the comparison with FAS has to be done by resorting to gray-level versions of the images in our dataset. However, FAS and SME cannot be quantitatively compared since, analogously to iNEDI, SME returns images whose size is slightly smaller than the size of the target images. The quantitative comparisons with FAS are hence done only for KR and ARI. The results obtained for the test images for $n=2$ are shown in Table 6 in terms of PSNR (columns 2-4) and SSIM (columns 5-7). The sources have been generated by applying to the gray-level target images bicubic interpolation with a reduction factor 0.5. It can be noted that in all cases FAS outperforms KR and ARI.

Finally, as concerns super-resolutions methods, we point out that some of them do produce high quality output images, at the expenses of a more computationally heavy process.

To show qualitatively the performance of FAS for different magnification rates refer to Fig. 7, where a detail of the target image “toucan” (top left) as well as the FAS scaled up images for different values of n are given. It may be observed that artifacts and blurring do not affect significantly the magnified images.

Fig. 7. A source image and the corresponding FAS scaled up images for $n=2, 3, 4, 5$

4 Conclusion

We have presented the fully automatic algorithm FAS to magnify color images. FAS is based on the use of local operations that are accomplished during only one scan of the source image, whichever is the selected integer zooming factor n . Each pixel in the source image is associated a block of $n \times n$ pixels in the scaled up image. Tentative values are ascribed to the pixels in each block, depending on the position of the pixels within the block. The tentative values are then modified in two updating steps, before inspecting the next generating pixel in the source image. The computational complexity of FAS is $O(N)$ where N is the size of the source image. From a qualitative point of view, the visual aspect of the magnified images is generally appealing, does not show evident artifacts and is affected by limited blurring. A quantitative evaluation has also been carried on by comparing the performance of FAS to the performance of other algorithms in terms of PSNR and SSIM. The method is suggested for RGB images, but can equally be employed in other color spaces.

References

1. Lehmann, T.M., Gonner, C., Spitzer, K.: Survey: interpolation methods in medical image processing. *IEEE Trans. Medical Imaging* 18, 1049–1075 (1999)
2. Amanatiadis, A., Andreadis, I.: A survey on evaluation methods for image interpolation. *Measurement Science and Technology* 20(10), 104015–104023 (2009)
3. Li, X., Orchard, M.T.: New edge-directed interpolation. *IEEE Trans. Image Processing* 10(10), 1521–1527 (2001)
4. Zhang, L., Wu, X.: An edge-guided image interpolation algorithm via directional filtering and data fusion. *IEEE Trans. Image Processing* 15(8), 2226–2238 (2006)
5. Takeda, H., Farsiu, S., Milanfar, P.: Kernel regression for image processing and reconstruction. *IEEE Trans. Image Processing* 16(2), 349–366 (2007)
6. Zhang, X., Wu, X.: Image Interpolation by Adaptive 2-D Autoregressive Modeling and Soft-Decision Estimation. *IEEE Trans. Image Processing* 17(6), 887–896 (2008)
7. Mallat, S., Yu, G.: Super-resolution with sparse mixing estimators. *IEEE Trans. Image Processing* 19(11), 2889–2900 (2010)
8. Jurio, A., Pagola, M., Mesiar, R., Beliakov, G., Bustince, H.: Image magnification using interval information. *IEEE Trans. Image Processing* 20(11), 3112–3123 (2011)
9. Schultz, R.R., Stevenson, R.L.: A Bayesian approach to image expansion for improved definition. *IEEE Trans. Image Processing* 3(3), 233–242 (1994)
10. Carey, W.K., Chuang, D.B., Hemami, S.S.: Regularity preserving image Interpolation. *IEEE Trans. Image Processing* 8(9), 1293–1297 (1999)
11. Mitra, S.K., Murthy, C.A., Kundu, M.K.: A technique for image magnification using partitioned iterative function system. *Pattern Recognition* 33(7), 1119–1133 (2000)
12. Tschumperle, D., Deriche, R.: Vector-valued image regularization with PDEs: a common framework for different applications. *IEEE Trans. Pattern Analysis and Machine Intelligence* 27(4), 506–517 (2005)
13. Arcelli, C., Brancati, N., Frucci, M., Ramella, G., Sanniti di Baja, G.: A fully automatic one-scan adaptive zooming algorithm for color images. *Signal Processing* 91, 61–71 (2010)
14. De Simone, F., Ticca, D., Dufaux, F., Ansorge, M., Ebrahimi, T.: A comparative study of color image compression standards using perceptually driven quality metrics. In: *Proc. SPIE Conf. Optics and Photonics, Applications of Digital Image Processing XXXI* (2008)
15. Wang, Z., Lu, L., Bovik, A.C.: Video quality assessment based on structural distortion measurement. *Signal Processing: Image Communication* 19(2), 121–132 (2004)
16. <http://decsai.ugr.es/cvg/dbimagenes/>
17. <http://sipi.usc.edu/database/>
18. <http://www.general-cathexis.com/>
19. Asuni, N., Giachetti, A.: Accuracy Improvements and artifacts removal in edge based image interpolation. In: *Proc. VISAPP 2008*, pp. 58–65 (2008)