

# ***BLAZE: Bilinear LAZA Algorithm for Zooming Enhancement***

S. Battiato, G. Di Blasi, G. Gallo, S. Toro  
{battiato, gdibiasi, gallo}@dmi.unict.it, salvo82@wind.it

Image Processing Laboratory – University of Catania – Italy  
<http://www.dmi.unict.it/~iplab>

## **Abstract**

In this paper we address the problem of producing an enlarged picture from a given digital image (zooming). We propose a method that takes into account information about discontinuities or sharp luminance variations while zooming the input picture. This is realized by a nonlinear iterative procedure of the zoomed image and could hence be implemented with limited computational resources. The algorithm works on monochromatic images and RGB color pictures. The basic idea of our technique, BLAZE, extends a previously proposed method: LAZA. LAZA is able to zoom only by a factor 2, BLAZE extends LAZA allowing to zoom by any factor  $f$ , with  $f$  an even number.

Experimental results show that the proposed method beat in quality and efficiency the classical zooming techniques (e.g. pixel replication, bilinear and bicubic).

*Keywords:* Image processing, Zooming, Interpolation, LAZA

## **1. Introduction**

One of the major factors impacting the perceived quality of digital images is the resolution size. The image resolution depends on the acquisition device (e.g. the number of photo-detector present on the tool): a high photo-detector density leads to images with a high resolution, while a low photo-detector density leads to images where the single pixel can be recognized. A way to increase the sampling percent is to increase the photo-detector number and decrease their dimension. However there is a physical constraint which limits the dimension decreasing. So, it is important to create new algorithmic techniques to increase the resolution of an image being able to solve the economic and technological sensor limits.

This problem arises frequently whenever a user wishes to zoom-in to get a better view of a given picture. There are several issues to take into account about zooming: unavoidable smoothing effects, reconstruction of high frequency details without the introduction of artifacts and computational efficiency both in time and in memory requirements. Several good zooming techniques are nowadays well known [5, 6, 8, 9, 10, 11, 15, 19, 21, 22].

A generic zooming algorithm takes as input an RGB picture and provides as output a picture of greater size preserving as much as possible the information content of the original image. For a large class of zooming techniques this is

achieved by interpolation: replication, bilinear and bicubic are the most popular choices and they are implemented in commercial digital image processing software.

Unfortunately, these methods, while preserving the low frequencies content of the source image, are not equally able to enhance high frequencies in order to provide a picture whose visual sharpness matches the quality of the original image. The method proposed in this paper, similar to other published algorithms [4, 7, 21], tries to take into account information about discontinuities or sharp luminance variations while increasing the input picture. A brief mention to an adaptive technique is presented in [21] where a prior classification of pixel neighborhood is used to apply the most suitable interpolation strategy. Different from [21] our technique realizes a nonlinear multiple scan of the original image and could hence be implemented with limited computational resources.

The research of new heuristic/strategies able to outperform classical image processing techniques is nowadays the key-point to produce digital consumer engine (e.g. Digital Still Camera, 3G Mobile Phone, etc.) with advanced imaging applications [3].

The basic idea of the new algorithm is to perform a gradient-controlled, weighted interpolation. The method speed up the entire adaptive process without requiring a preliminary gradient computation because the relevant information is collected during the zooming process. Although not linear, our technique is simpler and hence much faster than fractal-based zooming algorithms [15, 18].

Our experiments show that the proposed method beats, in quality, pixel replication, bilinear interpolation and bicubic interpolation.

The rest of this paper is organized as follows: in Section 2 we present our algorithm, Section 3 extends the proposed technique to RGB images. In Section 4 we show the experimental results. Finally in Section 5 we suggest directions for future work and research.

## **2. The basic algorithm**

In this section we give a detailed description of the proposed algorithm. The basic idea of our technique, BLAZE, extends a previously proposed method: LAZA [2]. LAZA is able to zoom only by a factor 2, BLAZE extends LAZA allowing to zoom by any factor  $f$ , with  $f$  an even number. The idea behind this new algorithm is based on a bilinear interpolation weighting the pixel value based on their position and an adaptive edge-sensing technique focused on the image. Differently from other adaptive techniques [3, 6, 7], BLAZE accelerates the whole process of local adaptation without requiring preliminary information about the image (for instance a preliminary calculation of the image gradient), because all the important information is collected during the zooming process. The algorithm works in four successive stages as described in the following Subsections.

## 2.1 Step 1: simple enlargement

The first stage is the simplest one and requires expanding the source  $w \cdot h$  pixels image into a regular grid of size  $(w \cdot f - (f - 1)) \cdot (h \cdot f - (f - 1))$  where  $f$  is the magnifying factor (Figure 1). More precisely if  $S(i, j)$  denotes the pixel in the  $i$ -th row and  $j$ -th column of the source image and  $Z(l, k)$  denotes the pixel in the  $l$ -th row and  $k$ -th column in the zoomed picture the expansion is described as a mapping  $E: S \rightarrow Z$  according to the equation:

$$E(S(i, j)) = Z(f \cdot i, f \cdot j) \quad \text{with } i = 0, 1, \dots, w; j = 0, 1, \dots, h$$

The mapping  $E$  leaves undefined the value of all the pixels in  $Z$  with at least a coordinate which is not a multiple of  $f$  (white dots in Figure 1).

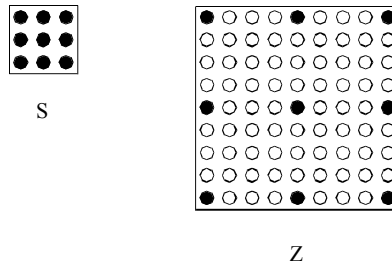


Figure 1: The first stage of zooming (simple enlargement).

## 2.2 Step 2: filling the holes (part I)

In the second stage, the algorithm scans line by line the pixels in  $Z$  whose coordinates equal to  $(f/2 + \gamma \cdot f, f/2 + \delta \cdot f)$ , with  $\gamma, \delta \in N$  (e.g. the pixels denoted by a gray dot, labeled with  $X$  in Figure 1). These pixels are elected as representatives of the dashed square in Figure 1.

For reference, in the following description we will use the capital letters  $A, B, C$  and  $D$  as in Figure 1, to denote the pixels surrounding the  $X$  and that have already been assigned a value in the previous step (black dots).

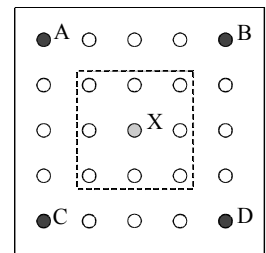


Figure 1: Uniformity.

For every pixel  $X$  one of the following mutual exclusive conditions is tested and a consequential action is taken:

- *Uniformity*: if  $\text{range}[(a, b, c, d)] < T_1$  then the values of the pixels belonging to the square centered in  $X$  and side  $f-1$  are calculated ("range" stands for the maximum value minus the minimum value). The values of these pixels (contained in the dashed square in Figure 1) are assigned according to the classical bilinear zooming algorithm ( $x$  and  $y$  represent the coordinates relative to  $X$  which occupies the position  $(0, 0)$ ):

$$val = (1 - \alpha)[(1 - \beta)A + \beta \cdot B] + \alpha[(1 - \beta)C + \beta \cdot D] \text{ with } \alpha = \frac{x + f/2}{f}, \beta = \frac{y + f/2}{f}$$

- *Edge in SW-NE direction:* if  $|a - d| > T_2 \otimes |a - d| \gg |b - c|$  then the values of the pixels belonging to the square centered in  $X$  and side  $f-1$  are calculated. The values of these pixels (contained in the dashed square in Figure 1) are assigned according to their relative position respect to  $X$  (note that this formula represents a sort of “bilinear directional zooming”):

$$val = (1 - |\beta|)[(1 - \alpha)B + \alpha \cdot C] + |\beta| \left[ \left( \frac{1}{k} - \beta \right) A + \left( \frac{1}{k} + \beta \right) D \right] \frac{k}{2}$$

$$\text{with } \alpha = \frac{f - (x - y)}{2 \cdot f}, \beta = \frac{x + y}{k \cdot f}$$

- *Edge in NW-SE direction:* if  $|b - c| > T_2 \otimes |b - c| \gg |a - d|$  then the values of the same pixels as the preceding case are calculated:

$$val = (1 - |\beta|)[(1 - \alpha)A + \alpha \cdot D] + |\beta| \left[ \left( \frac{1}{k} - \beta \right) B + \left( \frac{1}{k} + \beta \right) C \right] \frac{k}{2}$$

$$\text{with } \alpha = \frac{f + (x + y)}{2 \cdot f}, \beta = -\frac{(x - y)}{k \cdot f}$$

- *Edge in N-S direction:* if  $|a - d| > T_1 \otimes |b - c| > T_1 \otimes (a - d)(b - c) > 0$  then the values of

the pixels in the segments  $(A, B)$  and  $(C, D)$  are assigned (dashed rectangles in Figure 1). The values of the pixels around  $X$  are left indefinite. The values of the pixels to be assigned are calculated according to the classical linear zooming algorithm. In particular the values of the pixels near to  $H_1$  are calculated according to the formula:

$$val = (1 - \alpha)A + \alpha \cdot B \text{ with } \alpha = \frac{x + f/2}{f}$$

while the values of the pixels near to  $H_2$  are calculated according to the formula:

$$val = (1 - \alpha)C + \alpha \cdot D \text{ with } \alpha = \frac{x + f/2}{f}$$

- *Edge in E-W direction:* if  $|a - d| > T_1 \otimes |b - c| > T_1 \otimes (a - d)(b - c) < 0$  then the values of the pixels in the segments  $(A, C)$  and  $(B, D)$  are assigned (dashed rectangles in Figure 1).

The values of the pixels around  $X$  are left indefinite. The values of the pixels to be assigned are calculated (as in the previous case) according to the linear zooming algorithm. In particular the values of the pixels near to  $V_1$  are calculated according to the formula:

$$val = (1 - \alpha)A + \alpha \cdot C \text{ with } \alpha = \frac{y + f/2}{f}$$

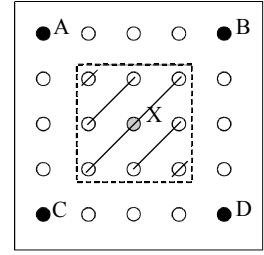


Figure 1: Edge in SW-NE direction.

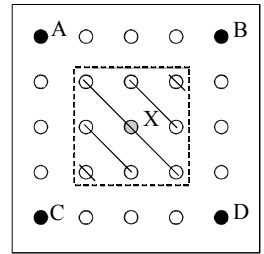


Figure 1: Edge in NW-SE direction.

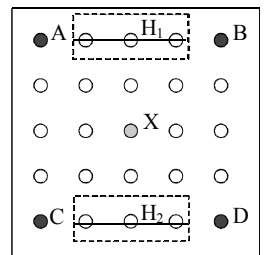


Figure 1: Edge in N-S direction.

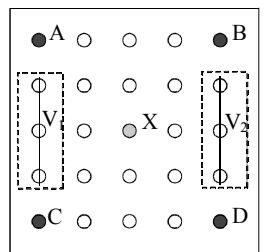


Figure 1: Edge in E-W direction.

while the values of the pixels near to  $V_2$  are calculated according to the formula:

$$val = (1 - \alpha)B + \alpha \cdot D \quad \text{with } \alpha = \frac{y + f/2}{f}$$

In the introduced formulas parameters  $T_1$  and  $T_2$  represent two threshold values for the image structure identification. They will be described in detail in Subsection 2.5.

There will be many pixels  $X$ , as like that as  $H_1, H_2, V_1$  and  $V_2$  pixels, to the end of this Step with an indefinite value. A next scanning of the enlarged image will allow the filling of these empty spaces.

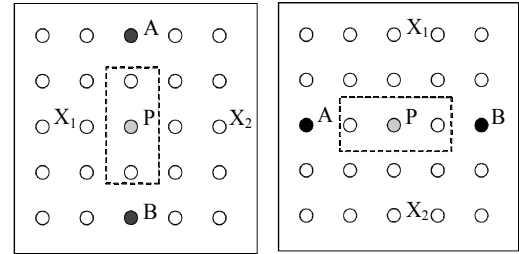
### 2.3 Step 3: filling the holes (Part II)

In the third stage, the algorithm scans “line by line” image  $Z$  looking for pixels left undefined in the previous stage and with coordinates equals to  $(\gamma \cdot f, f/2 + \delta \cdot f)$  or  $(f/2 + \gamma \cdot f, \delta \cdot f)$  with  $\gamma, \delta \in N$ .

These are the pixels denoted by the gray dot with label  $P$  in Figure 1.

They will act as representatives for the pixels in the segment  $(A, B)$ .

The two cases shown in Figure 1 will be handled in a similar way in



**Figure 1:** pixels taken into account in this Step.

the following discussion. For reference we will use the letters  $A$  and  $B$  as in Figure 1, to denote the pixels above and below (left and right, respectively) of pixel  $P$  and that have already been assigned a value in the expansion stage (black dots). Observe that the values of pixels  $X_1$  and  $X_2$  may, or may not, have been assigned in the previous stage. The algorithm considers the following two cases and takes consequential actions:

- $X_1$  or  $X_2$  have not been assigned a value. In this case the algorithm checks if  $|a - b| < T_1$  with the same threshold  $T_1$  introduced in the previous stage. If it is so, the values of the pixel  $P$  and the belonging ones in the dashed rectangle in Figure 1 are calculated according to the classical linear zooming algorithm ( $p$  represents the coordinate,  $x$  or  $y$ , relative to  $P$  which occupies the position  $(0,0)$ ):

$$val = (1 - \alpha)A + \alpha \cdot B \quad \text{with } \alpha = \frac{p + f/2}{f}$$

- $X_1$  and  $X_2$  have both been assigned a value. In this case the algorithm checks for the presence of a vertical or a horizontal edge. The following sub-cases arise:

- Presence of an edge in direction  $X_1$ - $X_2$ : if  $|a - b| > T_2 \otimes |a - b| \gg |x_1 - x_2|$  then a value is assigned to the pixels on the segment  $(A, B)$ . The value of  $P$  is calculated as an average of the pixel values  $X_1$  and  $X_2$ :

$$val = \frac{X_1 + X_2}{2}$$

the same one happens for the pixels beyond and beneath  $P$ , respectively doing the average between the pixels beyond and beneath  $X_1$  and  $X_2$ .

- *Presence of an edge in direction A-B*: if  $|x_1 - x_2| > T_2 \otimes |x_1 - x_2| \gg |a - b|$  then the values for  $P$  and the pixels in the dashed rectangle in Figure 1 are calculated. The values of these pixels are calculated (as in the “ $X_1$  or  $X_2$  not assigned” case), according to the formula:

$$val = (1 - \alpha)A + \alpha \cdot B \text{ with } \alpha = \frac{p + f/2}{f}$$

- *None of above*: No action is taken and the value of  $P$  is left undefined.

## 2.4 Final Step: rebinning

The final stage of the algorithm scans once more time picture  $Z$  looking for pixels with undefined value and fix the hole using a suitably weighted mean value as described below. Preliminarily the gray value scale (typically ranging from 0 to 255) is subdivided into a rougher scale of  $m=256/q$  bins  $B_i$  where  $q$  is a suitable, user selected integer value. The  $i$ -th bin includes the gray values ranging from  $q \cdot (i-1)$  up to  $q \cdot i-1$ . The algorithm first looks for every pixel  $X$ , with coordinates equals to  $(f/2 + \gamma \cdot f, f/2 + \delta \cdot f)$ , with  $\gamma, \delta \in N$  and with an undefined value. In this case the pixels  $A, B, C$  and  $D$  surrounding  $X$  as in Figure 1 fall within at least one and at most four bins  $B_j$ . The representative values of these bins (typically the median) are averaged to produce a value for  $X$ . For the pixels surrounding  $X$  (pixels in the dashed square in Figure 1), a value based on their relative position respect to  $X$  is calculated (This formula represents a sort of "weighted zooming" between a bilinear zooming and the  $X$  value):

$$val = \left[1 - \frac{1}{k}\right]X + \frac{1}{k} \left\{ (1 - \alpha) \left[ (1 - \beta)A + \beta \cdot B \right] + \alpha \left[ (1 - \beta)C + \beta \cdot D \right] \right\}$$

$$\text{with } \alpha = \frac{x + f/2}{f}, \beta = \frac{y + f/2}{f}$$

Observe that in this way the more frequent values are less weighted: this “trick” guarantees, as shown in the experiments, a better detail preservation in the zoomed picture. In order to preserve visual sharpness is fundamental reducing as much as possible the low-pass operation. It is also important to note that the values considered to assign a value to  $X$  are coming from the original values in the source and hence are not affected by the smoothing operations done in the three previous stages. Eventually the algorithm looks for every pixel  $P$ , with coordinate equals to  $(\gamma \cdot f, f/2 + \delta \cdot f)$  or  $(f/2 + \gamma \cdot f, \delta \cdot f)$  with  $\gamma, \delta \in N$ , whose value is still left undefined. In this case the same rebinning procedure described for  $X$  pixels is performed starting from the values  $A, B, X_1$  and  $X_2$  as in Figure 1. Observe that at this step, all four of these values have been already set. For the pixels surrounding  $X$  (pixels in the dashed

rectangle in Figure 1), a value based on their relative position respect to  $X$  is calculated ( $p$  represents the coordinate,  $x$  or  $y$ , relative to  $P$  which occupies the position  $(0,0)$ ):

$$val = \left[ 1 - \left( \frac{1}{k} \right) \right] P + \frac{1}{k} [(1 - \alpha)A + \alpha \cdot B]$$

with  $\alpha = \frac{p + f/2}{f}$

## 2.5 Parameter settings

During the algorithm explanation, it has been observed that BLAZE requires the choice of some threshold values. All tests have been made with some default fixed values, in particular  $T_1 = 4$ ,  $T_2 = 3$  and  $K = 1$ . These threshold values should be chosen in a more proper way using local adaptive measures of activity in a local neighborhood (e.g. local variance and/or local edge sensing [20]). Our heuristic values have been selected because they provided in preliminary experiments the best zooming quality. Preliminary experiments, moreover, have shown that small variations in these parameters do not produce large quality changes in the zoomed picture.

## 2.6 Complexity

The proposed algorithm requires a complexity of  $O(N)$  to enlarge a digital image of  $N$  pixels, it only requires the memory space engaged by the image enlarged without occupying other resources.

The same complexity  $O(N)$  is required by classical zooming algorithms where indeed the multiplicative constants are sensibly greater. This is particularly true, having a strong impact on the overall performance, when the dimensions of the input image are extremely large (e.g. 3/4 Mpixel acquired by a High Quality Digital Camera).

## 3. Zooming a color picture

The basic algorithm described above for gray scale pictures can be easily generalized to the case of RGB colored digital images. To do so, we take advantage of the higher sensitivity of human visual system to luminance variations with respect to chrominance values. Hence it makes sense to allocate larger computational resources to zoom luminance values, while chrominance values may be processed with a simpler and more economical approach. This simple strategy is inspired by analogues techniques used by classical lossy image compression algorithms like JPEG and/or JPEG2000 [14, 17, 24] vastly implemented in almost digital still camera engines. We propose to operate as follows:

- translate the original RGB picture into the YUV color model;
- zoom the luminance values Y according with the basic algorithm described above;
- zoom the U and V values using a simpler pixel replication algorithm;
- back translates the zoomed YUV picture into an RGB image.

The results obtained by using this basic approach are qualitatively comparable with the results obtained by using bicubic interpolation over the three-color channels. From the computational point of view, it is important to note how no significant difference in terms of timing response has been observed between the simple application of our approach to the three RGB planes and the approach described above (RGB-to-YUV conversion, Y zooming U-V replication, YUV-to-RGB conversion).

In real applications (DSC, 3G Mobile phone, etc.) the zooming process inside typical Image Generation Pipeline (if present) is realized just before compression [3]: the YUV conversion is always performed as a crucial step to achieve visual lossless compression. In this case the color conversion itself does not introduce further computational costs.

## 4. Experimental results

The validation of a zooming algorithm requires the assessment of the visual quality of the zoomed pictures. Unfortunately this qualitative judgment involves qualitative evaluation of many zoomed pictures from a large pool of human observers and it is hard to be done in a subjective and precise way. For this reason several alternative quantitative measurements related to picture quality have been proposed and widely used in the literature. To validate our algorithm we have chosen the approaches proposed in [12, 13, 16]. In particular we have used the Cross-Correlation and the PSNR between the original picture and the reconstructed picture to assess the quality of reconstruction.

In our experimental contest we have first collected a test pool of 240 gray scale pictures. For each image  $I$  in this set we have first performed the following operations:

- reduction by decimation using a factor of  $1/4$ , obtaining the image  $I_d$
- reduction by averaging using a factor of  $1/4$ , obtaining the image  $I_a$

The size reduction technique adopted may have influences on the quality of the zoomed picture. Starting from  $I_d$  and  $I_a$  we have obtained the zoomed images listed as follows:

- $I_{dr}$ , the picture  $I_d$  zoomed by a factor 4 using a simple pixel replication
- $I_{ar}$ , the picture  $I_a$  zoomed by a factor 4 using a simple pixel replication
- $I_{dl}$ , the picture  $I_d$  zoomed by a factor 4 using bilinear interpolation
- $I_{al}$ , the picture  $I_a$  zoomed by a factor 4 using bilinear interpolation
- $I_{db}$ , the picture  $I_d$  zoomed by a factor 4 using bicubic interpolation
- $I_{ab}$ , the picture  $I_a$  zoomed by a factor 4 using bicubic interpolation
- $I_{dn}$ , the picture  $I_d$  zoomed by a factor 4 using BLAZE

- $I_{an}$ , the picture  $I_a$  zoomed by a factor 4 using BLAZE

The Cross-Correlation has been calculated according to the formula:

$$C = \frac{\left| \sum_{x,y} A(x,y)B(x,y) - XYab \right|}{\sqrt{\left( \sum_{x,y} A^2(x,y) - XYa^2 \right) \left( \sum_{x,y} B^2(x,y) - XYb^2 \right)}}$$

where

- $a$  and  $b$  denote, respectively, the average value of picture  $A$  and  $B$
- $X$  and  $Y$  denote, respectively, width and height, in pixels, of images  $A$  and  $B$

Notice that Cross-Correlation coefficients are between 0 and 1. The more the coefficient approaches 1, the better the reconstruction quality.

For every pair  $(I, I_{dr}), (I, I_{dl}), (I, I_{db}), (I, I_{dn}), (I, I_{ar}), (I, I_{al}), (I, I_{ab}), (I, I_{an})$  of original and zoomed pictures, we have computed the relative Cross-Correlation coefficient. The coefficients have been averaged over the test pool. The values obtained are reported in Table 1.

$C_{dr}$	$C_{dl}$	$C_{db}$	$C_{dn}$
0.95429802	0.963681914	0.9636664	0.965549722
$C_{ar}$	$C_{al}$	$C_{ab}$	$C_{an}$
0.945977297	0.953460184	0.953325669	0.955819978

**Table 1:** Cross-Correlation values using different techniques

The table promptly shows that the Cross-Correlation coefficients obtained with the proposed technique are better than the analogous coefficients obtained using bicubic interpolation, bilinear interpolation and replication. This proves the better or equal ability of BLAZE algorithm to fill in with proper details the missing information in the larger picture.

Another classic quality measure we used in our validation experiments is the *Peak Signal Noise Ratio (PSNR)*. Table 2 shows the results of PSNR tests on replication, bilinear interpolation, bicubic interpolation and BLAZE, as from an image reduced by decimation and by averaging. In both cases the BLAZE algorithm has PSNR values higher both than bicubic and bilinear techniques.

	BLAZE	Replication	Bilinear	Bicubic
Decimation	28.07	26.56	27.72	27.71
Average	26.78	25.84	26.56	26.55

**Table 2:** PSNR values using different techniques

The same tests, described above in detail for an  $x4$  zooming factor, have been also performed for higher factors always obtaining positive results. Figure 2, Figure 3, Figure 4 and Figure 5 show the Cross-Correlation and PSNR values calculated on decimation and average for zooming factors of  $x2$ ,  $x4$ ,  $x6$  and  $x8$ . Figure 6 shows an image zoomed using the four different algorithms while Figure 7 shows two images enlarged with the BLAZE algorithm.

The algorithm has been implemented in Java2 Standard Edition 1.4.2 and all experiments have been carried out on a PC Athlon XP-M 1800+, 192MB RAM, with Windows XP Home Edition. To allow the reader to directly test the quality of our algorithm an applet is free available at the URL [1], at the same URL is also available for download a Java application.

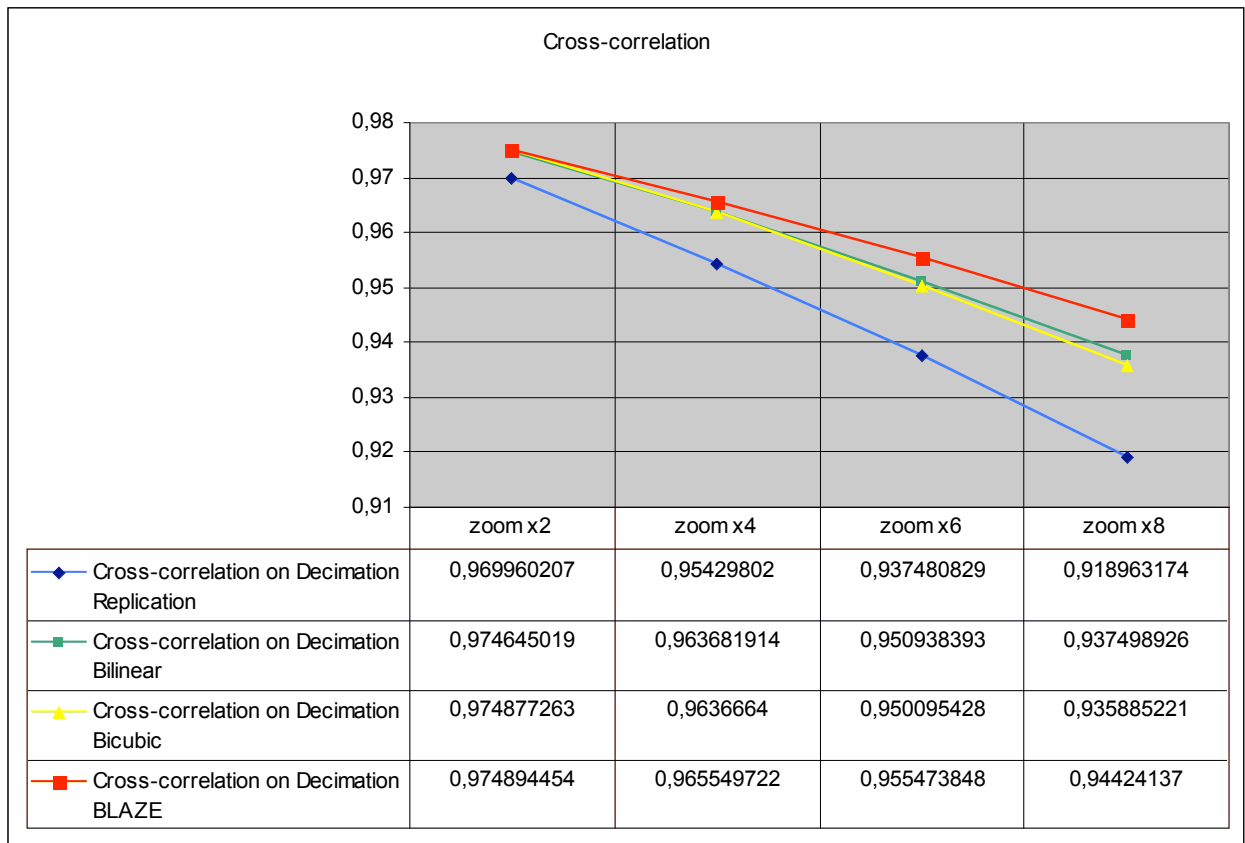
## 5. Conclusions

In this paper we have proposed a new zooming technique for digital images both in grey scale, and in RGB color format. The proposed technique has been compared according to different performance indicators to bicubic interpolation, bilinear interpolation and pixel replication algorithm. The experimental tests have shown that BLAZE produces qualitatively the best results respect to the other techniques examined, even if the algorithmic complexity is equal or even lower.

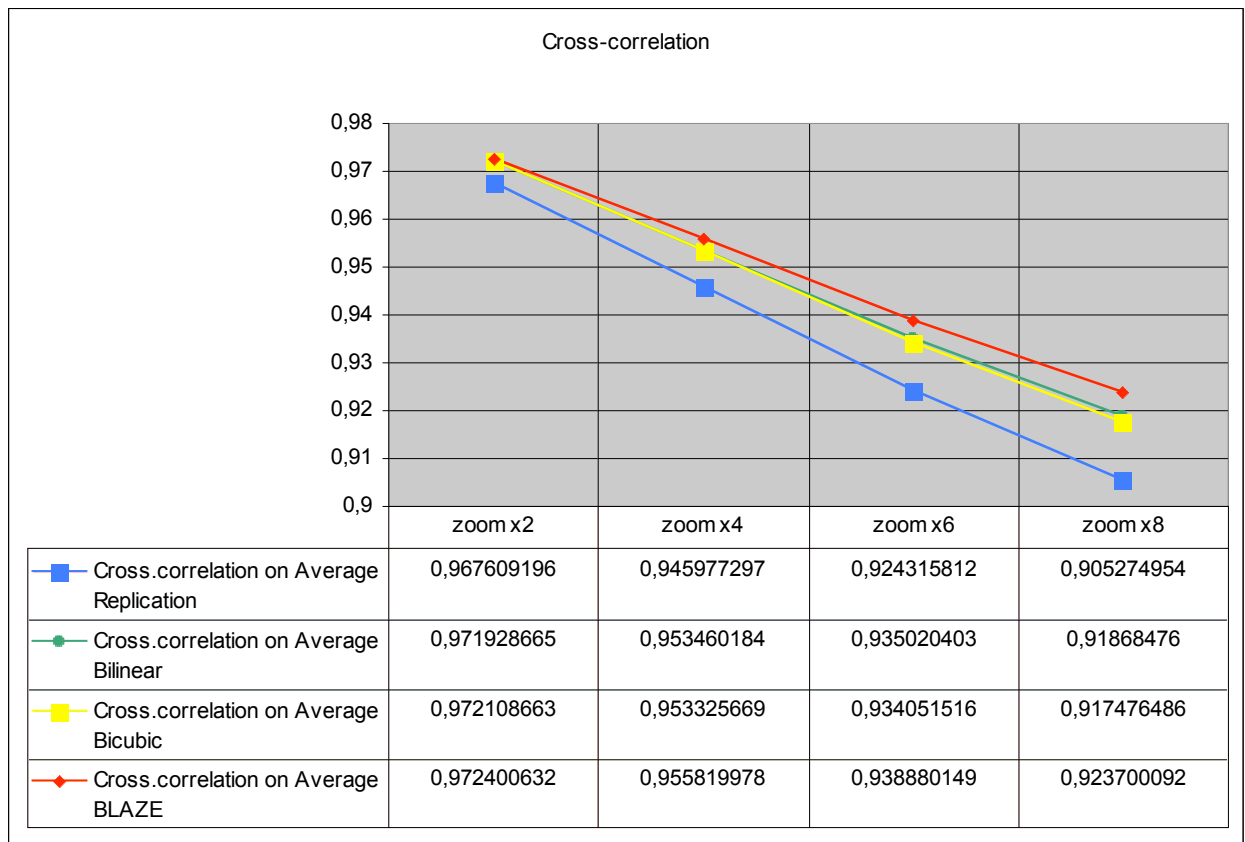
## Reference

1. S. Battiato, G. Di Blasi, G. Gallo, S. Toro, *the BLAZE Applet*, [www.dmi.unict.it/~gdibiasi/BLAZE/BLAZE.html](http://www.dmi.unict.it/~gdibiasi/BLAZE/BLAZE.html). Java application [www.dmi.unict.it/~gdibiasi/BLAZE/BLAZE.jar](http://www.dmi.unict.it/~gdibiasi/BLAZE/BLAZE.jar), 2006
2. S. Battiato, G. Gallo, F. Stanco, *A Locally-Adaptive Zooming Algorithm for Digital Images*. Elsevier Image Vision and Computing Journal 20/11, pp. 805-812, 2002
3. S. Battiato, M. Mancuso, *An introduction to the digital still camera technology*. ST Journal of System Research, Special Issue on Image Processing for Digital Still Camera 2 (2), 2001
4. S.D. Bayarakeri, R.M. Mersereau, *A new method for directional image interpolation*. In Proceedings of International Conference on Acoustics, Speech and Signal Processing, Detroit, MI 24, 1995
5. C.C. Chang, Y.C. Chou, Y.H. Yu, K.J. Shih, *An Image Zooming Technique Based on Vector Quantization Approximation*. submitted to Image and Vision Computing.
6. D.F. Florencio, R.W. Schafer, *Post-sampling aliasing control for images*. In Proceedings of International Conference on Acoustics, Speech and Signal Processing, Detroit, MI 2, pp 893–896, 1995
7. K.P. Hong, J.K. Paik, H. Ju Kim, C. Ho Lee, *An edge-preserving image interpolation system for a digital camcorder*. IEEE Transactions on Consumer Electronics 42 (3), 1996
8. H.S. Hou, H.C. Andrews, *Cubic splines for image interpolation and digital filtering*. IEEE Transactions on Acoustics, Speech, Signal Processing ASSP-26 (6), pp. 508–517, 1978
9. A.K. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989
10. R.G. Keys, *Cubic convolution interpolation for digital image processing*, IEEE Transactions. on Acoustics, Speech, Signal Processing 29 (6), pp. 1153–1160, 1981
11. S.W. Lee, J.K. Paik, *Image interpolation using adaptive fast B-spline filtering*. In Proceedings of International Conference on Acoustics, Speech, and Signal Processing 5, pp. 177–179, 1993

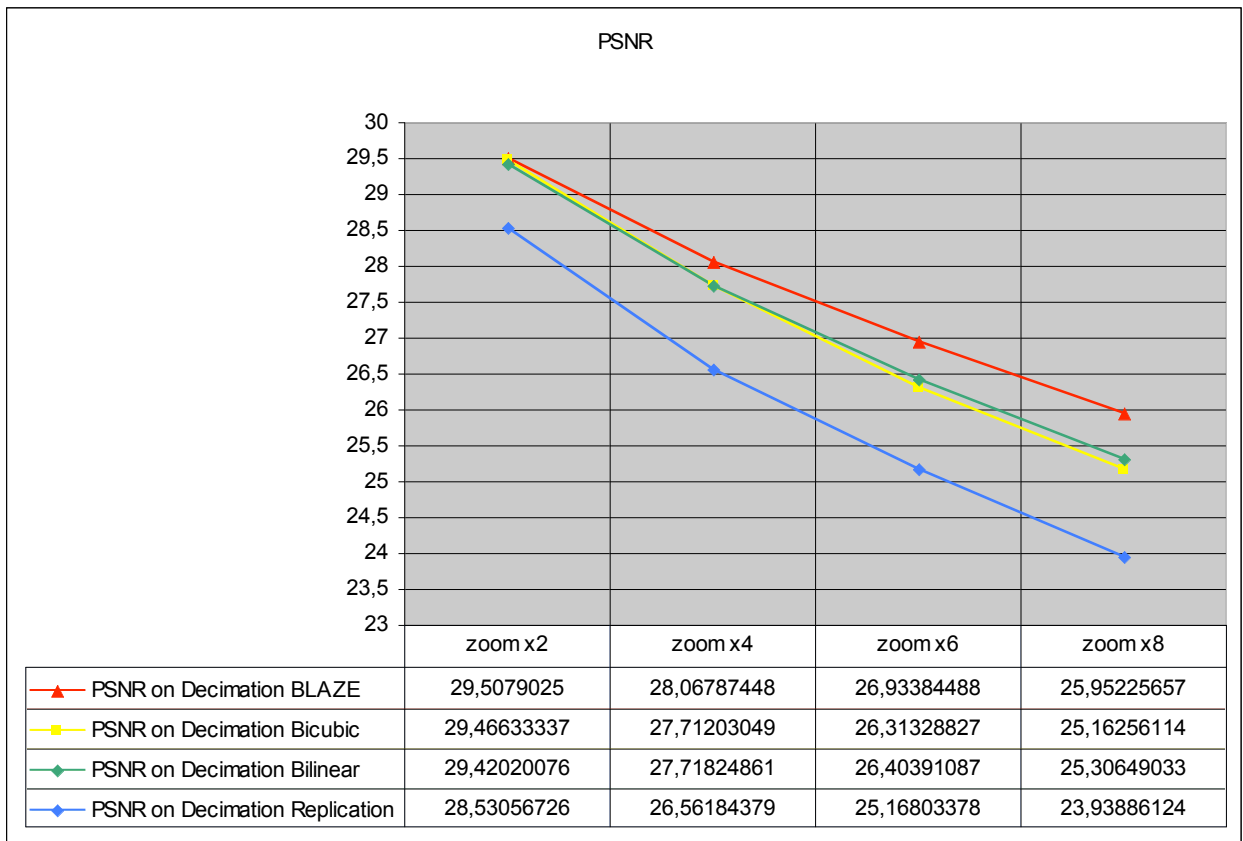
12. T.M. Lehmann, C. Gonner, K. Spitzer, Survey: *Interpolation methods in medical image processing*. IEEE Transactions on Medical Imaging 18 (11), 1999
13. E. Maeland, *On the comparison of interpolation methods*. IEEE Transactions on Medical Imaging MI-7, pp. 213–217, 1988
14. M.W. Marcellin, M.J. Gormish, A. Bilgin, M.P. Boliek, *An overview of JPEG2000*. In Proceedings of IEEE DCC, 2000
15. D.M. Monro, P.D. Wakefield, *Zooming with implicit fractals*. In Proceedings of International Conference on Image Processing ICIP97 1, pp. 913–916, 1997
16. J.A. Parker, R.V. Kenyon, D.E. Troxel, *Comparison of interpolating methods for image resampling*. IEEE Transactions on Medical Imaging MI-2, pp. 31–39, 1983
17. W.B. Pennebaker, J. Mitchell, *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, NY, 1993
18. E. Polidori, J.L. Dugelay, *Zooming using iterated function systems*. NATO ASI Conference on Fractal Image Encoding and Analysis, Trondheim, Norway, 1995
19. L. Rodrigues, D.L. Borges, L.M. Gonçalves, *A Locally Adaptive Edge Preserving Algorithm for Image Interpolation*. In Proceeding of SIBGRAPI 2002, pp 300-305, 2002
20. T. Sakamoto, C. Nakanishi, T. Hase, *Software pixel interpolation for digital still cameras suitable for a 32-bit MCU*. IEEE Transactions on Consumer Electronics 44 (4), pp. 1342–1352, 1998
21. P.V. Sankar, L.A. Ferrari, *Simple algorithms and architecture for B-spline interpolation*. IEEE Transactions on Pattern Analysis Machine Intelligence 10, pp. 271–276, 1988
22. D. Su, P. Willis, *Image Interpolation by Pixel Level Data-Dependent Triangulation*. Computer Graphics Forum Volume23, Issue 2, 2004
23. S. Thurnhofer, S. Mitra, *Edge-enhanced image zooming*, Optical Engineering 35 (7), pp. 1862–1870, 1996
24. G.K. Wallace, *The JPEG still picture compression standard*, Communications of the ACM 34 (4), 1991



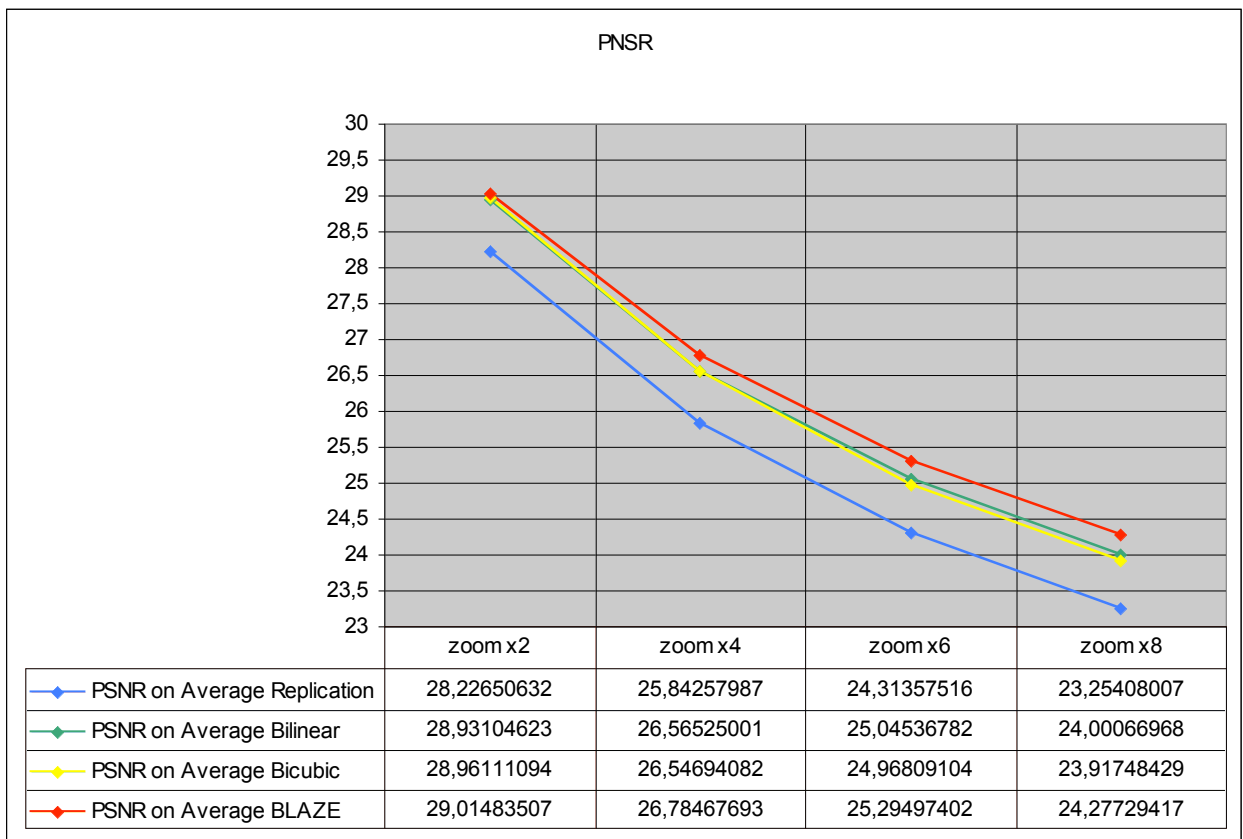
**Figure 2:** Cross-Correlation values on decimation for zooming factor of  $x_2$ ,  $x_4$ ,  $x_6$  and  $x_8$ .



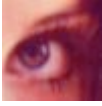
**Figure 3:** Cross-Correlation values on average for zooming factor of  $x_2$ ,  $x_4$ ,  $x_6$  and  $x_8$ .



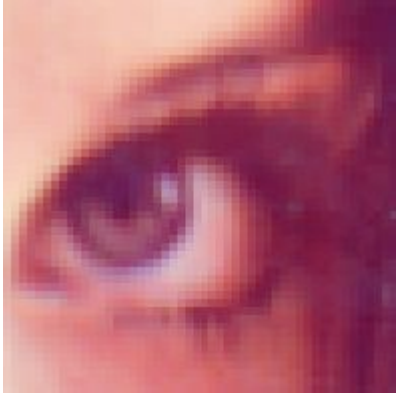
**Figure 4:** PSNR values on decimation for zooming factor of  $x2$ ,  $x4$ ,  $x6$  and  $x8$ .



**Figure 5:** PSNR values on average for zooming factor of  $x2$ ,  $x4$ ,  $x6$  and  $x8$ .



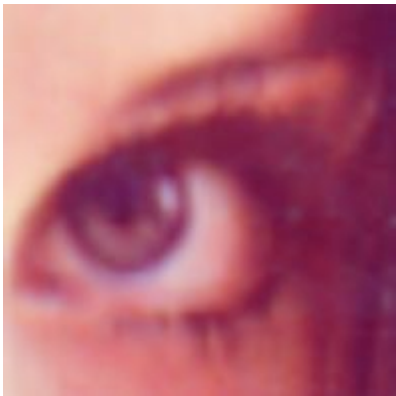
(a) Original Image



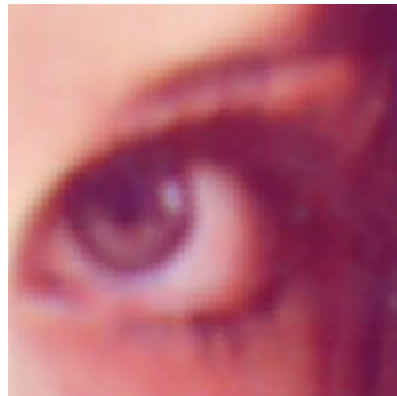
(b) Image zoomed by replication algorithm



(c) Image zoomed by bilinear algorithm



(d) Image zoomed by bicubic algorithm



(e) Image zoomed by BLAZE

**Figure 6:** Examples of zoomed images using different techniques.



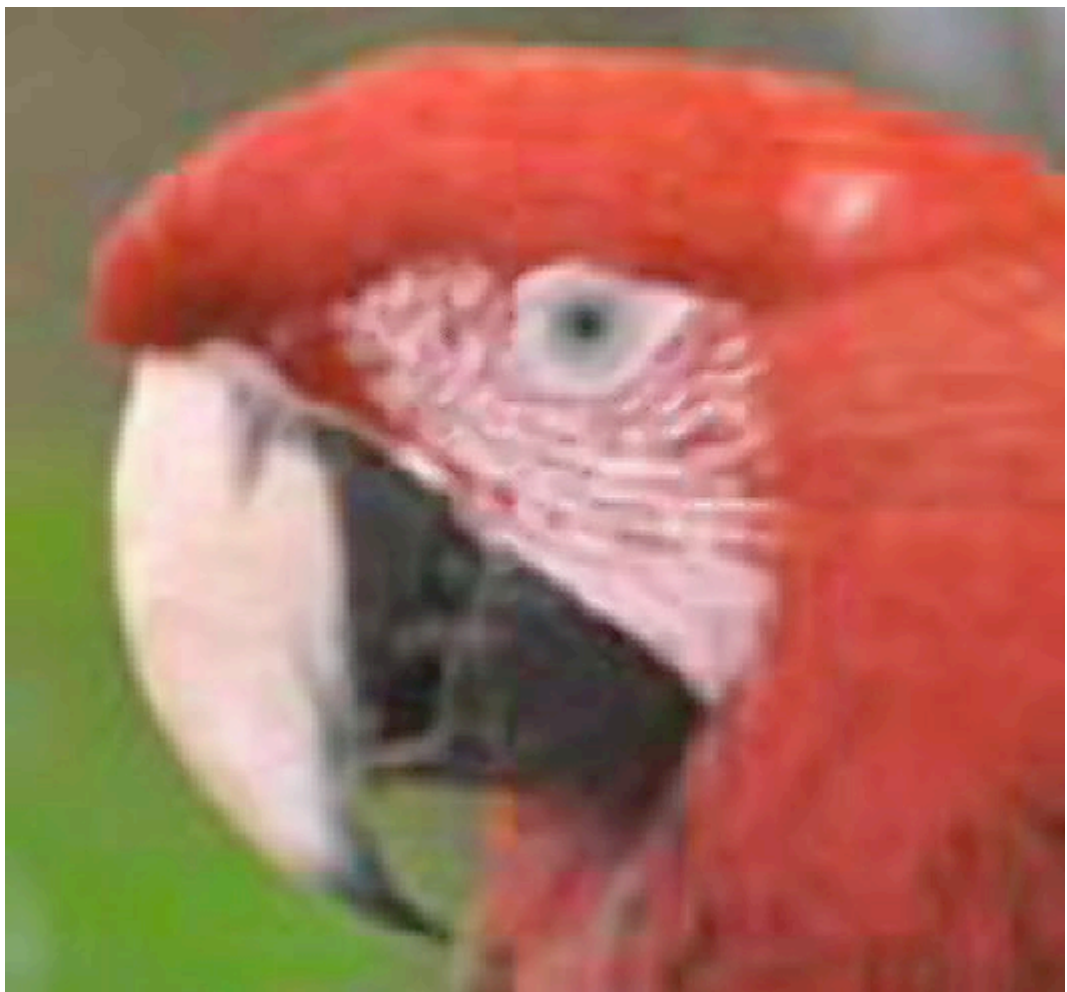
(a) Original Image



(b) Original Image



(c) Zoomed image



(d) Zoomed mage

**Figure 7:** Examples of zoomed images using BLAZE.