A fully automatic one-scan adaptive zooming algorithm for color images
Carlo Arcelli, Nadia Brancati, Maria Frucci, Giuliana Ramella, Gabriella Sanniti di Baja*
Institute of Cybernetics “E. Caianiello”, CNR, Pozzuoli (Naples), Italy

Article info
Article history:
Received 29 July 2009
Received in revised form 30 November 2009
Accepted 6 June 2010
Available online 10 June 2010

Keywords:
Image processing
Color images
Adaptive zooming
Interpolation

Abstract
We present an interpolation algorithm for adaptive color image zooming. The algorithm produces the magnified image in one scan of the input image, and is fully automatic since does not involve any a priori fixed threshold. Given any integer zooming factor $n$, each pixel of the input image generates an $n \times n$ block of pixels in the zoomed image. For the currently visited pixel of the input image, the pixels of its associated block are first assigned tentative values, which are then adaptively updated before building the next block. The method is suggested for RGB images, but can equally be employed in other color spaces. Peak signal to noise ratio (PSNR) and Structural SIMilarity (SSIM) are used to evaluate the performance of the algorithm.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Image zooming is a topic that has received noticeable attention, due to the increasing number of applications where it is involved, e.g., for matching images captured by sensors with different capturing resolution, satellite image analysis, medical image display, entertainment, and image compression and transmission. Zooming is a process that, starting from an input image at a given resolution, virtually generates a version of that image at higher resolution. Several gray-level image zooming methods are available in the literature, some of which also include a generalization to color images, see e.g., [1–9]. In turn, a few papers dealing specifically with color image zooming have been published, e.g., [10,11].

Common approaches to image magnification are based on nearest-neighbor, bilinear, and bicubic interpolation, e.g., [12–15], and their implementation can be easily accomplished by using OpenCV libraries or Matlab. Unfortunately, the nearest-neighbor interpolation method produces high resolution images whose appearance is quite blocky and, hence, visually unpleasant. Bilinear and bicubic interpolations are less affected by blocky appearance, but tend to blur the resulting image. This drawback can be alleviated by using non-linear interpolation to better preserve edges [16–18], or by resorting to more sophisticated methods (e.g., Bayesian maximum a posteriori approach [19], wavelet approach [20], fractal-based approach [21–23], and PDEs-based approach [24,25]). Staircase artifacts, which are likely to bias the generated high resolution image especially for zooming factors larger than 4, can be reduced by using adaptive methods, which are, however, characterized by a significantly larger computational cost [3,26–30].

We present a new discrete adaptive algorithm in the framework of interpolation methods, to magnify a digital color image by any integer zooming factor. The suggested algorithm accomplishes the goal in a single inspection of the image to be magnified, does not require interaction with the user or a fine tuning of parameters depending on image domain because it does not involve any a priori fixed threshold, and produces visually appealing magnified images since it does not create noticeable artifacts even for large zooming factors.

The proposed algorithm is an improved generalization to color images of an algorithm that we have recently
suggested for gray-level images [8]. It has been devised to work with RGB images, but can be applied also to images in a different color space, e.g., YUV.

For any zooming factor $n$, a window of size $m \times m$, where $m$ is the smallest odd integer larger than $n$, is used in this paper to decide which neighbors of any pixel $p$ of the input image $P$ have to be used together with $p$, in order to compute the values of the pixels $q_k$, $(k=1, 2, \ldots, n \times n)$ in the block of size $n \times n$ associated to $p$ in the output image $Q$. The $m \times m$ window is also used to determine suitable multiplicative factors, so as to compute the values of the $n \times n$ pixels $q_k$ as the weighted average of $p$ and the proper neighbors. Then, $P$ is inspected and for each pixel $p$ the values for the $n \times n$ pixels $q_k$ are computed, by using only $3 \times 3$ operations.

The neighbors of any $p$ in $P$ that are involved for the computation of each $q_k$ in the corresponding $n \times n$ block of $Q$ are the pixels in $P$ whose associated $n \times n$ blocks in $Q$ (partially) overlap the $m \times m$ window centered on $q_k$. The weights for $p$ and for the selected neighbors are given by the number of pixels in the intersections in $Q$ between the $n \times n$ blocks, corresponding to $p$ and to the selected neighbors, and the $m \times m$ window centered on each $q_k$. A tentative value of each $q_k$ is computed as the weighted average of $p$ and of the proper neighbors of $p$. Once all the pixels in a block have received a tentative value, the final values are computed before inspecting the next pixel in $P$, by an adaptive process conditioned by a threshold that is different for each block and is automatically computed. The magnified image $Q$ is obtained after all pixels in $P$ have been inspected. In the following, the algorithm, which involves multiplicative weights and an adaptive procedure, will be called weighted adaptive zooming (WAZ for short).

To evaluate the performance of WAZ, we use the peak signal to noise ratio (PSNR) and the Structural SIMilarity (SSIM) [31,32]. To this aim, we first build the lower resolution source images, which will be used as input to the magnification algorithm, starting from high resolution target images, which are reduced by a factor $n$. Then, we compare the $n$-zoomed images obtained by WAZ to the target images.

We compare the performance of WAZ with that of other algorithms in the same category of interpolation methods. In particular, we consider standard nearest-neighbor, bilinear and bicubic interpolation algorithms, as well as a recent adaptive learning zooming algorithm, ALZ [10]. The obtained results show that WAZ has a better performance, in terms of PSNR, SSIM and visual appearance, with respect to standard interpolation algorithms, and is significantly less computationally expensive of ALZ, though the performance is comparable.

Note that also when the method is used for images in a color space different from RGB, for example YUV, we apply the same one-raster scan procedure to all channel components. This is generally not the case for other methods, based on a more sophisticated and expensive procedure. Such a procedure is generally applied only to the luminance channel $Y$, while simpler magnification schemes are used for the chrominance channels, so as to limit the computational effort. To take into account that the most relevant information is carried on by the luminance channel, we resort to the Weighted Peak Signal to Noise Ratio (WPSNR) and Weighted Structural SIMilarity (WSSIM) [32–34], to evaluate the performance of the zooming method for images in the YUV color space.

## 2. Zooming method

The proposed zooming method follows a scheme similar to that of bilinear interpolation. In bilinear interpolation, each value in the magnified image is computed by suitably adding the contributions given by four points. In detail, given the intensity values at four points $p_{1,1}=(x_1,y_1), p_{1,2}=(x_1,y_2), p_{2,1}=(x_2,y_1)$, and $p_{2,2}=(x_2,y_2)$ of the image to be magnified, the value at point $q_k=(x, y)$ in the magnified image is obtained by combining the intensity values of $p_{1,1}, p_{1,2}, p_{2,1},$ and $p_{2,2}$, with multiplicative factors computed in terms of the coordinates $x, y, x_1, y_1, x_2,$ and $y_2$.

Also in our method, when the zooming factor $n$ is an even number, exactly four pixels are used (namely, the generating pixel $p$ and three properly selected neighbors in the image to be magnified) and suitable multiplicative weights are employed to obtain the tentative values of the $n \times n$ pixels $q_k$ in the block associated to $p$ in the magnified image.

A difference of our method with respect to bilinear interpolation is that our multiplicative weights are computed in terms of the overlapping between an $m \times m$ window and the $n \times n$ blocks associated to $p$ and to the involved neighbors.

Another difference is that for an odd zooming factor, the number of pixels contributing to the tentative values is likely to be larger than four (at most the generating pixel $p$ and all its eight neighbors).

Finally, another difference is that once tentative values of the pixels $q_k$ in the $n \times n$ block associated to a pixel $p$ have been computed, the values of the pixels $q_k$ are updated to their final values before inspecting the next pixel in the image to be magnified. The updating process reduces the differences between the value of the generating pixel $p$ and the final values of the generated pixels $q_k$ in the block, while better delineating edges. The effect of updating is an improvement of the performance of the algorithm in terms of PSNR and SSIM.

The algorithm is implemented by using only $3 \times 3$ operations and is rather fast. For each zooming factor, the neighbors of $p$ and the multiplicative weights are determined in a straightforward manner and the final result is achieved in a single inspection of the image to be magnified.

### 2.1. The zooming algorithm WAZ

Let $P$ be an input image consisting of $row \times column$ pixels, and let $n$, with $n$ integer, be the selected zooming factor. The $n$-zoomed image $Q$ will consist of $(n \times row) \times (n \times column)$ pixels. Let $P^R$ and $Q^H, H\in\{R,G,B\}$, denote any of the three channels into which $P$ and $Q$ can be seen as decomposed.
The eight neighbors of a pixel $p$ of $P^H$ are denoted as $tl$ (top left), $t$ (top), $tr$ (top right), $r$ (right), $br$ (bottom right), $b$ (bottom), $bl$ (bottom left) and $l$ (left), by taking into account their positions with respect to $p$, as shown in Fig. 1.

Each pixel $p$ of $P^H$ is associated with a block of $n \times n$ pixels, $q_1$, $q_2$, ..., $q_n \times n$, in the $n$-zoomed image $Q^H$. The tentative value of any pixel $q_k$ in the $n \times n$ block is computed in terms of the values of $p$ and of proper neighbors of $p$ in $P^H$, by using suitable multiplicative weights.

To determine the proper neighbors of $p$ and the corresponding weights, a window of size $m \times m$ is centered on each $q_k$, where $m$ is the smallest odd integer larger than $n$.

For simplicity, let us consider $n=4$ as zooming factor; hence, for the $m \times m$ window it is $m=5$. In Fig. 2, the nine $4 \times 4$ blocks that are associated to $p$ and to its eight neighbors in $P^H$ are framed by thick gray lines, and the $5 \times 5$ window centered on the pixel $q_k$ (depicted as a gray square in the central block) is framed by thick black lines. Depending on the position of $q_k$ within the block, the $5 \times 5$ window partially overlaps the four blocks in $Q^H$ that are associated to $p$ and to three of its neighbors.

For the pixel $q_k$ denoted by a gray square in Fig. 2, the three neighbors of $p$ whose associated $4 \times 4$ blocks intersect the $5 \times 5$ window are $t$, $tr$, and $r$. The intersection of the blocks with the window regards six pixels for the block associated to the neighbor $t$, four pixels for the block associated to $tr$, six pixels for the block associated to $r$, and nine pixels for the block associated to $p$. The four numbers 6, 4, 6 and 9 are the multiplicative weights to be used for the intensity values of $t$, $tr$, $r$ and $p$, respectively. The weighted average is then assigned to $q_k$ as its tentative value.

By shifting the $m \times m$ window on all the $q_k$ in the $n \times n$ block associated to $p$, we identify for each of them which neighbors of $p$ to select and which are the proper weights. Fig. 3 summarizes which pixels to use and which weights to consider for $n=4$ for each $q_k$.

We point out that for any even zooming factor always three neighbors of $p$ are involved together with $p$. In turn, when $n$ is odd (then, $m=n+2$) the number of involved neighbors ranges from three to eight, depending on the position of $q_k$ within the block.

The inspection of $P^H$ starts when, for the selected zooming factor $n$, the neighbors of $p$ and the weights have been determined for the $n \times n$ pixels $q_k$ in a block.

Once the tentative values of the $n \times n$ pixels $q_k$ associated to the current $p$ have been determined, the final values of the pixels $q_k$ are adaptively computed, before continuing the scan of $P^H$. Let $qmin$ and $qmax$, respectively, denote the minimum and the maximum value of the $q_k$ in the block. Moreover, let

$$A = \min_{q_k \neq p} |q_k - p|$$

$$T = \begin{cases} qmax + qmin - p & \text{if } (qmax + qmin)/2 < p \\ p & \text{if } (qmax + qmin)/2 > p \\ \end{cases}$$

Then, the final values assigned to the pixels $q_k$ are computed as follows:

1. If $p < qmin$, then $q_k = (q_k - A)$, for any $q_k > p$.
2. If $p \geq qmax$, then $q_k = (q_k + A)$, for any $q_k < p$.

![Fig. 1. The eight neighbors of a pixel $p$.](image)

![Fig. 2. The nine $4 \times 4$ blocks associated to a pixel $p$ and to its neighbors are framed by thick gray lines. The $5 \times 5$ window, centered on a pixel $q_k$ (gray square) belonging to the central $4 \times 4$ block, is framed by thick black lines.](image)

![Fig. 3. Each of the 16 cells shows the four pixels of $P^H$ and their corresponding weights, in parentheses, to be used for the computation of each of the 16 pixels $q_k$ associated to $p$.](image)
(3) if $q_{\min} < p < q_{\max}$, then

$$q_k = \min(255, q_k + D), \text{ for any } q_k \geq T,$$

$$q_k = \max(0, q_k - A), \text{ for any } q_k < T.$$

The rationale of conditions (1) and (2) is to reduce the differences between the value of $p$ and the final values of the pixels in the block, while preserving the relative differences among the $q_k$. Thus, if $q_{\min}$ for a block is larger than or equal to the value of the pixel $p$ generating that block, the final values of the pixels $q_k$ are lowered with respect to their tentative values. In turn, if $q_{\max}$ for the block is smaller than or equal to the value of the pixel $p$, the tentative values are increased. In any case, the amount for lowering/increasing is given by the minimal discrepancy in value with respect to $p$ within the block. As for condition (3), its aim is to sharpen edges. This is achieved by increasing the tentative values larger than or equal to $T$ by the amount $A$, and by decreasing tentative values smaller than $T$ by the same amount. We note that $T$ assumes different values for different blocks and has not to be fixed a priori by the user, so making the zooming method adaptive and fully automatic.

We have experimentally verified that WAZ outperforms our previous zooming algorithm for gray-level images [8] in terms of PSNR and SSIM, and improves the visual aspect of the zoomed image, since it allows a better edge delineation.

3. Experimental results

We have tested WAZ by using different zooming factors on a large number of RGB images with different resolutions, taken from public databases [35,36]. A subset including 50 target images, all with resolution $512 \times 512$, used to evaluate the performance of WAZ with different zooming factors, is shown in Fig. 4. Here, the results for $n=2$ and $4$ are reported.

Two sets of source images, respectively, of size $256 \times 256$ and $128 \times 128$, have been built. It is well known that the quality of the source images influences the quality of the zoomed images. In this respect, to limit the presence of Moiré patterns in the source images, we have chosen a decimation process that assigns to each pixel of the source image a value equal to the average of the values of the pixels in the $n \times n$ block of the target image corresponding to that pixel.

For each source image, we have computed the images zoomed by WAZ, as well as by nearest-neighbor, bilinear and bicubic interpolation. For the latter three methods, we resorted to standard OpenCV libraries.

For all pairs consisting of a target image and the zoomed image obtained starting from the corresponding source image, we use both PSNR and SSIM to evaluate our method. Since the visual aspect of the zoomed images is obviously important, we regard SSIM as a better measure with respect to PSNR, which is often inconsistent with human eye perception.

For gray-level images, PSNR is computed as follows:

$$PSNR = 20 \times \log_{10} \left( \frac{255}{\sqrt{MSE}} \right)$$

where $MSE = \frac{1}{H \times K} \sum_{i=1}^{H} \sum_{j=1}^{K} (v_{ij} - w_{ij})^2$ and $v_{ij}$ and $w_{ij}$, respectively, belong to the target image and to the zoomed image of size $H \times K$.

For RGB images, where three are the values per pixel, the definition of PSNR is still the same, but MSE is the sum over all squared value differences divided by image size and by three.

Fig. 4. Test images of size $512 \times 512$. 
The SSIM index for two gray-level images \( v \) and \( w \) is computed as follows:

\[
\text{SSIM}(v, w) = \frac{(2\mu_v\mu_w + c_1)(2\sigma_{vw} + c_2)}{\mu_v^2 + \mu_w^2 + \sigma_{vw}^2 + c_2}
\]

where \( \mu_v \) is the average of \( v \), \( \mu_w \) is the average of \( w \), \( \sigma_v^2 \) is the variance of \( v \), \( \sigma_w^2 \) is the variance of \( w \), \( \sigma_{vw} \) is the covariance of \( v \) and \( w \); \( c_1 = (k_1L)^2 \), \( c_2 = (k_2L)^2 \) are two variables to stabilize the division with weak denominator; \( L \) is the dynamic range of the pixel values (255 for 8-bit images); \( k_1 = 0.01 \) and \( k_2 = 0.03 \) are default values.

The SSIM index to evaluate image quality is computed by using a sliding window approach. The window size suggested by Wang et al. [33] is fixed to \( 8 \times 8 \). The sliding window moves pixel-by-pixel from the top-left to the bottom-right corner of the image, and the SSIM index is calculated within the sliding window. As a result, an SSIM index map of the image is obtained, and the overall similarity is computed within the sliding window. As a result, an SSIM index map of the image is obtained, and the overall similarity is computed within the sliding window. As a result, an SSIM index map of the image is obtained, and the overall similarity is computed within the sliding window.

The computation of SSIM has been done in this paper by using the code developed for OpenCV in C++ by Rabah Mehdii [37], where an \( 11 \times 11 \) Gaussian weighting function is used to compute average, variance and covariance of the images.

PSNR and SSIM, respectively, computed in correspondence with the zoomed images obtained by WAZ, bilinear and bicubic interpolation, are given in Table 1 for the 50 test images of Fig. 4 for \( n = 2 \) and 4. The highest values of PSNR and SSIM for each test image are in bold.

The average values of PSNR and SSIM on the 50 test images in the case \( n = 2 \) are, respectively, 28.530 and 0.8025 for WAZ, 28.016 and 0.7789 for bilinear interpolation, and 27.011 and 0.7692 for bicubic interpolation. In turn, for \( n = 4 \), the average PSNR and SSIM are, respectively, 24.814 and 0.6174 for WAZ, 24.678 and 0.6098 for bilinear interpolation, and 22.929 and 0.5642 for bicubic interpolation. We note that in the average the PSNR and SSIM relative to WAZ are higher than those for bilinear and bicubic interpolation, for both zooming factors.

In Table 1, we have not included the PSNR and SSIM results for the nearest-neighbor interpolation, since they are generally remarkably worse than those obtained by the other three methods and the visual aspect of the zoomed images is often significantly affected by the presence of artifacts. As an example, refer to Fig. 5, where the target image pelican, the source and the four-zoomed version obtained by nearest-neighbor interpolation are shown.

As concerns the visual aspect of WAZ zoomed images, refer to Fig. 6, where three examples are shown for both \( n = 2 \) and 4. It can be observed that blurring is limited even for \( n = 4 \).

To show the improvements of WAZ with respect to standard interpolation methods as concerns edge delineation, we use two simple synthetic images, respectively, consisting of two uniform regions (with gray-levels 25 and 225) separated by a horizontal edge and a diagonal edge. The synthetic images and the images magnified for \( n = 4 \) by WAZ, bilinear interpolation and bicubic interpolation are shown from left to right in Fig. 7. The performance diagrams are also given in Fig. 7. It can be observed that bicubic interpolation has the worst performance out of the three methods. The performance of bilinear interpolation and WAZ is similar. We note that the behavior of bilinear interpolation is symmetric, while WAZ has some asymmetrical behavior for the diagonal edge image. However, we remark that the gap between the two uniform regions in the zoomed images is narrower with WAZ than with bilinear interpolation. In fact, only two intermediate values (185 and 65) separate the two regions in the WAZ 4-zoomed horizontal edge image, while four values (200, 150, 100 and 50) are found by bilinear interpolation. Moreover, only six intermediate values (49, 121, 145, 185, 201 and 217) instead of eight values (41, 72, 103, 134, 159, 178, 197 and 216), respectively, separate the uniform regions in the WAZ and bilinear 4-zoomed diagonal edge image.

Experiments have been carried on with different integer values for the zooming factor \( n \). While, as expected, PSNR, SSIM and the visual quality of the zoomed images diminish when the zooming factor increases, we remark that the aspect of the zoomed images still remains satisfactory. As an example, in Fig. 8, a \( 64 \times 64 \) detail of the target image frog, centered on one of the eyes of the frog, is shown together with the \( 7 \)-zoomed \( 448 \times 448 \) image and the \( 11 \)-zoomed \( 704 \times 704 \) image obtained by WAZ.

WAZ has also been compared to the interpolation algorithm ALZ [10], based on adaptive learning zooming and dealing with YUV color images. ALZ is a local gradient analysis algorithm that takes into account information about discontinuities or sharp luminance variations, while magnifying the input picture. For the local gradient analysis, two thresholds \( T_1 \) and \( T_2 \) were experimentally set by the authors to 66 and 22. Fifteen source, target and reconstructed (2-zoomed) images used in [10] to show the performance of ALZ have been made available by the authors as RGB images [38]. Thus, to compare the performances of WAZ and ALZ, our zooming algorithm has been applied to the RGB source images in the dataset [38] with a zooming factor \( n = 2 \), and the PSNR has been computed by using our 2-zoomed images and the RGB target images in the dataset. As for the PSNR characterizing the performance of ALZ, we have not referred to the values reported in [10], but have computed the PSNR values by comparing the ALZ reconstructed RGB images and the target RGB images in the dataset. We have done this for two reasons: (1) the examples shown in [10] deal only with six images, i.e., a subset of the 15 samples in the dataset, (2) the PSNR has been computed in [10] by taking into account only the Y channel, as we have learned after private communication by the authors [39]. Table 2 compares the PSNR of ALZ and WAZ.

Table 2 shows that the two methods have practically the same performance. As an example, in Fig. 9 we show the two images of the dataset for which ALZ (WAZ) has
Table 1

<table>
<thead>
<tr>
<th>n=2</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WAZ</td>
<td>Lin</td>
</tr>
<tr>
<td>House</td>
<td>27.824</td>
<td>27.657</td>
</tr>
<tr>
<td>Airplane</td>
<td>30.002</td>
<td>29.747</td>
</tr>
<tr>
<td>Baboon</td>
<td>22.572</td>
<td>22.220</td>
</tr>
<tr>
<td>Boat</td>
<td>27.206</td>
<td>27.075</td>
</tr>
<tr>
<td>Girl</td>
<td>30.168</td>
<td>29.983</td>
</tr>
<tr>
<td>Lena</td>
<td>31.689</td>
<td>31.534</td>
</tr>
<tr>
<td>Peppers</td>
<td>29.396</td>
<td>29.385</td>
</tr>
<tr>
<td>Burfish</td>
<td>25.873</td>
<td>25.594</td>
</tr>
<tr>
<td>Daisyleaf</td>
<td>29.158</td>
<td>28.576</td>
</tr>
<tr>
<td>Elephant</td>
<td>31.299</td>
<td>31.503</td>
</tr>
<tr>
<td>Jeruslem</td>
<td>31.249</td>
<td>30.873</td>
</tr>
<tr>
<td>Malignt</td>
<td>29.367</td>
<td>29.210</td>
</tr>
<tr>
<td>Monolake</td>
<td>25.138</td>
<td>24.805</td>
</tr>
<tr>
<td>Pelican</td>
<td>27.058</td>
<td>26.709</td>
</tr>
<tr>
<td>Seaport</td>
<td>29.535</td>
<td>29.124</td>
</tr>
<tr>
<td>Stacchoch</td>
<td>27.767</td>
<td>27.297</td>
</tr>
<tr>
<td>Toucan</td>
<td>36.679</td>
<td>37.320</td>
</tr>
<tr>
<td>Splash</td>
<td>32.374</td>
<td>32.398</td>
</tr>
<tr>
<td>Frog</td>
<td>31.801</td>
<td>31.501</td>
</tr>
<tr>
<td>Fiore</td>
<td>28.193</td>
<td>28.219</td>
</tr>
</tbody>
</table>
| Bicubic interpolation is applied to the U and V components of the pair of images. Standard bicubic interpolation is applied to the U and V components of the pair of images.
**Fig. 5.** Source (128 × 128), 4-zoomed image (512 × 512) using nearest-neighbor interpolation, and target (512 × 512).

**Fig. 6.** From top to bottom: girl, peppers and Lena. From left to right: the 256 × 256 source, the 2-zoomed 512 × 512 image, the 128 × 128 source, and the 4-zoomed 512 × 512 image.
Fig. 7. Two synthetic images magnified by WAZ, bilinear interpolation and bicubic interpolation.
differently from ALZ, where the values for the two employed thresholds $T_1$ and $T_2$ were determined experimentally by the authors, the unique threshold $T$ used in condition (3) of WAZ is automatically computed.

Since in [10] a comparison has been done with respect to the algorithm LAZA, suggested by the same authors in a previous paper [3], and it has been shown that ALZ outperforms LAZA, we can argue that also WAZ outperforms LAZA.

Our method has been described as designed for RGB images, but can be used also in a different color space. For completeness, we briefly discuss the performance of WAZ in the YUV color space. We point out that, due to the limited cost of our process, we can still apply the same process to $Y$, $U$ and $V$, differently from [10], where the suggested adaptive learning zooming is applied only to $Y$, while standard bicubic interpolation is used for $U$ and $V$ to save computation time.

When WAZ is applied to color images in YUV, to compute PSNR and SSIM we should take into account that the $Y$ channel carries on the most relevant information contents with respect to $U$ and $V$ channels. To this aim, we compute PSNR and SSIM before converting back the YUV image to RGB, and hence resort to the weighted peak signal to noise ratio (WPSNR) and to the Weighted Structural SIMilarity (WSSIM) computed as follows:

$$WPSNR = w_Y PSNR_Y + w_U PSNR_U + w_V PSNR_V,$$

$$WSSIM = w_Y SSIM_Y + w_U SSIM_U + w_V SSIM_V,$$

where the three weights $w_Y=0.8$, $w_U=0.1$ and $w_V=0.1$ have been adopted, as suggested in [33].

As an example in the YUV color space, the WPSNR and WSSIM for the images shown in Fig. 6 are given in Table 3.

By comparing Table 3 with the values given in Table 1 for the images girl, peppers and lena in the RGB color space, we note that the performance of WAZ is even better, in terms of PSNR and SSIM, in the YUV space. In turn, no noticeable differences can be appreciated by looking at the images zoomed in RGB and YUV color spaces.

4. Concluding remarks

We have presented the adaptive algorithm WAZ for color image zooming, which is simple to implement and has a limited computational cost since it requires $3 \times 3$ operations accomplished during only one scan of the source image. The algorithm has been implemented on a Pentium IV, 3.2 GHZ, personal computer. The computational complexity is $O(N)$, where $N$ is the size of the image to be magnified. The execution time is in the average less than 0.2 s when we have applied WAZ to the 50 images in the dataset used in the paper with a zooming factor $n=4$.

WAZ can be seen as framed in the same category of interpolation methods and for an even zooming factor the scheme is similar to that of bilinear interpolation. A relevant feature is that the algorithm is fully automatic, so that no interaction with the user is necessary. Any zooming factor can be used, provided that it is an integer number. Each pixel in the source image is associated a block of $n \times n$ pixels in the $n$-zoomed image. Tentative values are ascribed to the pixels in each block, depending on the position of the pixels within the block. The

<table>
<thead>
<tr>
<th></th>
<th>PSNR ALZ</th>
<th>PSNR WAZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane 256</td>
<td>28.259</td>
<td>28.120</td>
</tr>
<tr>
<td>Airplane 512</td>
<td>30.486</td>
<td>30.002</td>
</tr>
<tr>
<td>Arial 366</td>
<td>24.033</td>
<td>24.934</td>
</tr>
<tr>
<td>Arial 732</td>
<td>26.468</td>
<td>26.859</td>
</tr>
<tr>
<td>Baboon 256</td>
<td>23.758</td>
<td>23.859</td>
</tr>
<tr>
<td>Baboon 512</td>
<td>22.430</td>
<td>22.573</td>
</tr>
<tr>
<td>Bike</td>
<td>27.967</td>
<td>28.137</td>
</tr>
<tr>
<td>Boat</td>
<td>27.113</td>
<td>27.206</td>
</tr>
<tr>
<td>Girl</td>
<td>30.252</td>
<td>30.167</td>
</tr>
<tr>
<td>House</td>
<td>29.754</td>
<td>29.439</td>
</tr>
<tr>
<td>Lena 256</td>
<td>30.004</td>
<td>29.894</td>
</tr>
<tr>
<td>Lena 512</td>
<td>31.829</td>
<td>31.690</td>
</tr>
<tr>
<td>Monarch</td>
<td>30.483</td>
<td>29.905</td>
</tr>
<tr>
<td>Peppers</td>
<td>29.486</td>
<td>29.396</td>
</tr>
<tr>
<td>Tulips</td>
<td>30.094</td>
<td>29.921</td>
</tr>
</tbody>
</table>

Fig. 8. A $64 \times 64$ source, left, the 7-zoomed $448 \times 448$ image, middle, and the 11-zoomed $704 \times 704$ image, right.
tentative values are then adaptively modified by conditions (1)–(3), which involve a threshold \(T\), whose value is different for each block and is automatically computed by exploiting the information carried on by the tentative values of the pixels in the block as well as the value of the pixel generating the block.

The algorithm has been suggested for RGB images, but can equally be employed in other color spaces. The same procedure is always used for the three channel-components, whichever color space is selected. This is not always the case for other methods, which use a simplified procedure for the chrominance channels and resort to a more sophisticated procedure for the luminance channel.

The visual aspect of the zoomed images is generally appealing, does not show evident staircase artifacts and blocky configurations, and is affected by only limited blurring. These properties still hold for images magnified by large zooming factors.

PSNR and SSIM have been used to evaluate the performance of WAZ and to compare it to the performance of other algorithms. The experiments done on a large set of images have shown that WAZ is generally better than standard interpolation methods, like nearest-neighbor, bilinear and bicubic interpolations, and shows a performance comparable to that of the algorithm ALZ [10]. However, WAZ is computationally more convenient than ALZ.

### References


[38] http://www.dmi.unict.it/~iplab/zooming/.