A Non-Linear Method to Interpolate Binary Images Using Location and Neighborhood Adaptive Rules

Pullat Joy Prabhakaran
International Institute of Information Technology – Bangalore
Electronic city, Bangalore, India 560100
e-mail: joy@iiitb.ac.in

Palanganda Ganapathy Poonacha
International Institute of Information Technology – Bangalore
Electronic city, Bangalore, India 560100
e-mail: poonacha.pg@iiitb.ac.in

Abstract— **In this paper, we propose a new zooming technique for binary images, using location and neighborhood adaptive, non-linear interpolation rules. These rules are inspired by the way an artist would draw an enlarged image. Using simple examples, we show that the output generated by popular interpolation techniques is very different from what a human does. Our rules are based on such observations for simple objects, and they try to mimic what an artist does. Using these rules, we interpolate complex images and demonstrate the impact. We compare our method with bicubic interpolation and show that our method gives better visual quality. We also show that, on an average, our method results in higher PSNR and a lower MPSNR. The SSIM of the output images are nearly the same for both methods. Our method overcomes a number of problems associated with known interpolation techniques, such as blurring and thickening of edges. Our method uses a set of sixteen rules in five categories. Each pixel in the interpolated image is computed by a chosen rule. The choice depends on the location of the pixel and the content in the neighborhood. The size of the neighborhood varies. Some rules can be influenced by distant pixels in the input and can impact distant pixels in the output. We present examples showing the effectiveness of our method. The results are visually appealing. We show that lines and dots, with single pixel thickness, retain their thickness. Inclined lines and solids don't develop as much jaggedness as happens with bicubic interpolation. Similarly, curves are also relatively smoother.**

Keywords- resolution; interpolation; binary-image; thinnes; location-adaptive; neighborhood-adaptiv; corner; slope.

I. INTRODUCTION

When High Resolution (HR) images are created by interpolating Low Resolution (LR) images, unpleasant artifacts are seen. Interpolation artifacts are errors introduced into the HR image by the interpolation process. In [1], we proposed an interpolation method for binary images that generated visually appealing images. Popular methods like nearest neighbor, bilinear and bicubic [2] interpolation, generate more unpleasant artifacts like smoothing of edges and pixelation. Figure 1 shows the artifacts that are created when a dot and a line are interpolated using bicubic interpolation and Average of Nearest Neighbors (ANN) [3] interpolation. For clarity, the image shown is magnified by a factor of 8, and some of the unchanging regions have been removed. The interpolation is by a factor of 2. Figure 1B and 1D show the outputs of bicubic and ANN interpolation, respectively. Interpolation introduces artifacts and these are the intermediate shades of gray. To bring this out clearly, the new shades have been assigned colors in Figure 1C and 1E. Different colors represent different magnitudes of gray. The specific colors have no significance here. We see that more pixels are distorted by bicubic interpolation.

It can be seen that these kinds of errors will not be introduced by an artist. Also, a human can identify and eliminate many of the errors caused by popular interpolation methods. Our method tries to encapsulate some of the things that we feel an artist does to enlarge images.

Interpolation artifacts are most likely to arise at object edges, on lines and curves that are one pixel thick, on inclined and curved solids or object intersections. Popular interpolation methods cause more unwanted artifacts in the case of binary image zooming.

A large number of interpolation methods are available in the literature [1]–[15]. Some of these, like Nearest Neighbor, Bilinear and Bicubic [2] methods, use surface fitting techniques with pre-defined constraints. These methods often create undesirable artifacts in the output. Many methods have been proposed to minimize such artifacts. In [4], an orientation constraint is computed for each pixel to be generated. The pixel value is computed as a function of this constraint and the four surrounding neighbors. In an earlier work [3], we proposed an interpolation method called Average of Nearest Neighbors (ANN). This was based on the idea that each pixel in the interpolated image should be generated by using all the available nearest neighbors in the original image and none of the other pixels.

In [5], curvature of the low resolution image is evaluated and this curvature information is interpolated using bilinear interpolation. The interpolated curvature information is used as a driving constraint to interpolate the complete image. In [6], the image is first interpolated using bilinear interpolation. As a second step, the quality is improved using a fourth order Partial Differential Equation (PDE) based method. A directional bicubic scheme is proposed in [7]. Here, the strongest edge in each 7x7 neighborhood is detected. If the edge strength is greater than a threshold, a one-dimensional bicubic interpolation is done along the edge. Our method shares some similarities with [7] because it also tries to find and preserves local edges.

In [8] and [9], a two-step super resolution process is studied. In the first step, the low resolution image is interpolated using Bicubic interpolation. Then, the HR image is further processed to improve the quality at the edges. In [8], the gradient profile of the LR image is used as a driving

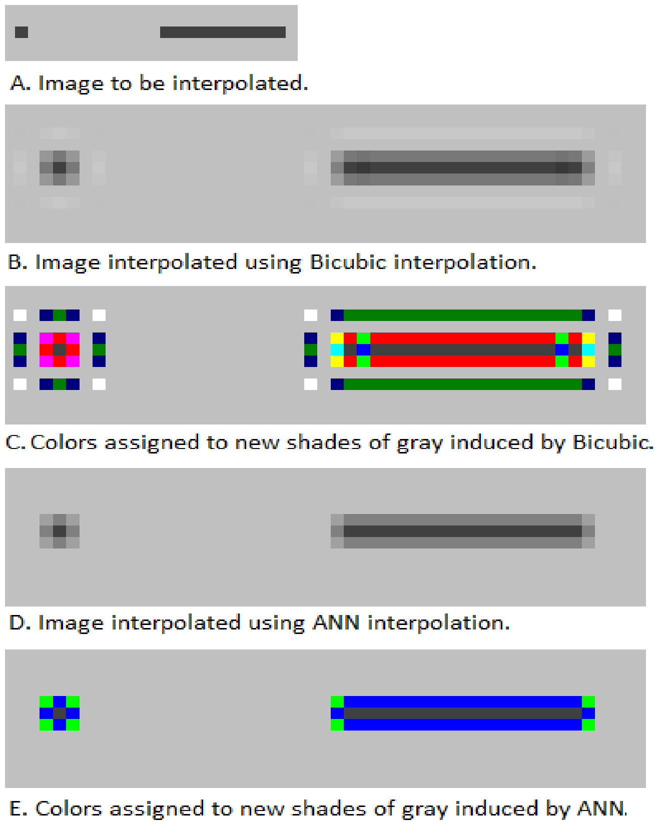
A. Image to be interpolated.
B. Image interpolated using Bicubic interpolation.
C. Colors assigned to new shades of gray induced by Bicubic.
D. Image interpolated using ANN interpolation.
E. Colors assigned to new shades of gray induced by ANN.

Figure 1. Impact of interpolation by different methods. The input dot and line are one pixel thick. The original image has two shades of gray. Interpolation introduces pixels in shades of gray that were not present in the original. In C and E, the shades introduced by interpolation are shown in different colors for better visibility.

gradient prior to change the gradient profile of the interpolated image. This process makes the edges sharper. In [9], this idea is extended by splitting the feature space into multiple subspaces and generating multiple priors.

In some scenarios, a frame from a video sequence needs to be interpolated. In [10] and [11], techniques to use information from adjacent frames to improve quality are discussed. The former uses an adaptive Wiener filter while the later uses Delaunay triangulation.

A machine learning based approach is discussed in [12]. Unknown pixels in the interpolated image are generated using the training data set that best matches the region near the pixel. The patch around the unknown pixel is matched with patches in the training set. Using the best matched stored patch, the pixel is assigned a value. The algorithm to do the matching takes into account the LR patch and the neighboring patches. A training based method that incorporates an explicit noise model is used to expand binary text images in [13].

In [14], edges are found as a first step. The edges are used to compute unknown pixels using cubic spline. In [15], unknown pixels are assigned the value of the neighbor that is closest to the value got by bilinear interpolation.

In this paper, we propose a Location and Neighborhood Adaptive (LNA) interpolation for binary images to be scaled by a factor of 2. Qualitatively, we show that the method generates visually pleasing images. For quantitative evaluation, we have compared LNA with bicubic interpolation using three standard metrics. These are Peak Signal to Noise Ratio (PSNR), Modified PSNR (MPSNR) and Structural Similarity (SSIM) index. PSNR represents a measure of the peak error while MPSNR is PSNR computed after applying a low pass filter. SSIM tries to measure the perceptual difference between two images.

PSNR is computed as:

$$\mathrm{PSNR} = 10\log_{10}\frac{\mathrm{Max}^2}{\mathrm{MSE}}$$

Where Max is the maximum possible value of each pixel and has the value 1 in our examples of binary images.

MPSNR is computed by passing the images being compared through a low pass filter, and then finding the PSNR of the filtered images. We have used a nine point, mean filter for our computations.

SSIM is computed as follows:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where x and y are two windows of equal size, μ_x and μ_y are the averages of the values in x and y, σ_x and σ_y their variances and σ_{xy} the covariance of x and y. c_1 and c_2 are constants of value $(0.01 * L)^2$ and $(0.03 * L)^2$ respectively, where L is the maximum value of a pixel magnitude. For our samples, the value of L is 1.

SSIM index can be computed for different windows. We have divided the test images into equal blocks of size 8x8 and computed SSIM index for each block. The average of all the block indices is the SSIM index of the image.

To explain the LNA method and also to demonstrate some of its features, we have used a few synthetic images that we have created. To validate the method, we have used sixteen commonly used test images. These images are from USC [16], Kodak [17] and Hlevkin [18].

The rest of this paper is organized as follows. Section II explains the LNA process and gives an overview. Section III describes the interpolation process and the five categories of rules. Subsections A to E, in Section III, describe the categories and associated rules. Experimental results are given in Section IV. Conclusions and suggestions for further extensions are given in Section V.

II. THE LNA APPROACH TO INTERPOLATION

LNA comprises of a set of rules. The rules try to mimic what an artist would do to define each pixel in the interpolated image. The outputs of a few popular interpolation techniques are shown in Figure 1. We see that the output is different from what an artist is likely to generate. Some of these differences are because, in Figure 1, the algorithms were allowed to generate pixels in grayscale, i.e., each pixel could take any one of 256 shades of gray. In Figure 2, we restrict the input and output to binary, i.e., each

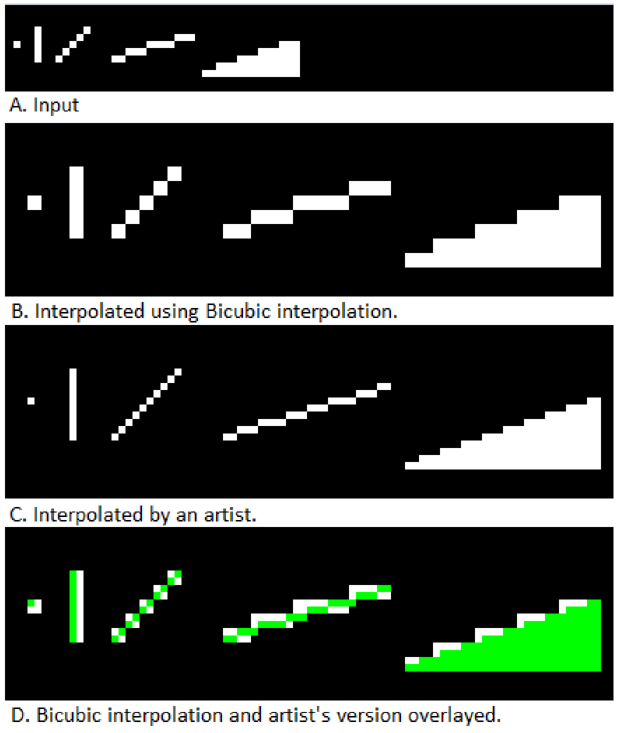
A. Input
B. Interpolated using Bicubic interpolation.
C. Interpolated by an artist.
D. Bicubic interpolation and artist's version overlayed.

Figure 2. A comparison of simple figures interpolated using bicubic interpolation and interpolated by an artist.

pixel can take one of two possible values. Figure 2 shows the difference in result when a simple binary image is interpolated using bicubic interpolation and when an artist enlarges it.

Figure 2D overlays the results of the interpolation and the exact differences are seen. The artist has retained thin features as thin. As a result, the dot has remained a single pixel in HR and the line has retained single pixel thickness. Similarly, slopes are better formed when an artist enlarges an image.

The rules of LNA try to mimic the interpolation process of the artist. The rules required to interpolate a typical color image with three color planes or a typical grayscale image with 256 gray shades are very complex. In this paper, we demonstrate the effectiveness of the LNA approach for binary images. The rules for binary images cannot be directly extended to grayscale or color images.

The method consists of a set of sixteen rules, grouped into five categories. The rules are of widely varying complexities. The choice of rule to assign value to a pixel depends on its location and the content in the neighborhood. The size of the neighborhood is dynamic and depends on the content. Some rules can be influenced by distant pixels in the input, and some rules can influence distant pixels in the output. This can be seen in Rules 14 and 16, described in the next section.

If the neighborhood meets certain conditions, our method implicitly tries to detect if an unknown pixel is part of an edge, a line or a corner. Based on this, it applies appropriate rules. To maintain smoothness of lines and edges, it both adds and deletes pixels in the foreground color when

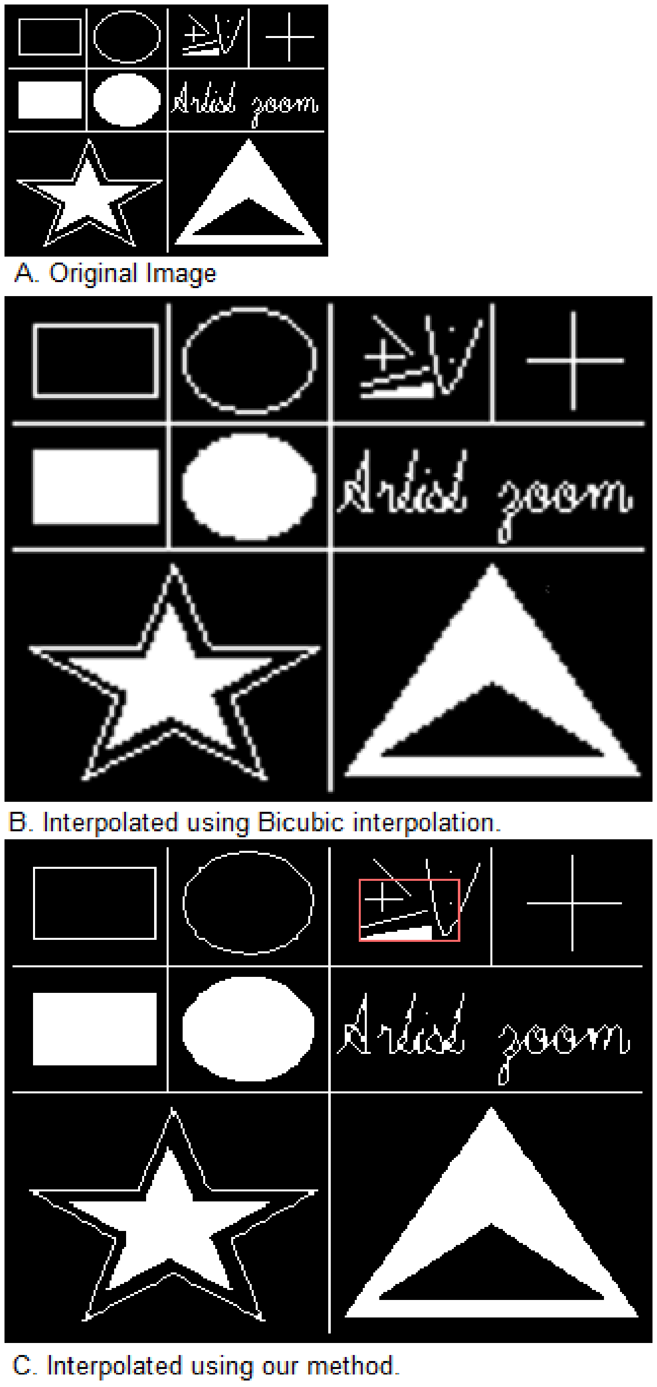
A. Original Image
B. Interpolated using Bicubic interpolation.
C. Interpolated using our method.

Figure 3. Comparison of our method with bicubic interpolation.

compared with simple pixel replication. The deletion ensures that smoothing does not cause extra thickening.

The interpolation process and individual rules are explained in Section III. The explanation is with reference to simple geometric figures. In Section IV we show that by applying the same rules to binary versions of standard test images like Lena, Baboon, Monarch and Barbara, we get good visual results.

Figure 3A shows the image used to explain our method. Figure 3B shows the image, interpolated using bicubic interpolation. The bicubic interpolation is done using Matlab. Figure 3C shows the same image, magnified using our method. As can be seen, the bicubic interpolation introduces more distortion than our method. The region in Figure 3C, shown in the red box, will be used to explain our method.

III. THE INTERPOLATION PROCESS

LNA categorizes unknown pixels in the HR canvas based on their locations. This is shown in Figure 4. The circles in the image represent individual pixels. At the start of the interpolation process, all these pixels are unknown. We categorize the pixels as O, H, V and D. Pixels on the even rows and even columns are of type O. The O pixels are what would be retained if the HR image is decimated by subsampling, using a scale factor of 2. Pixels on even rows and odd columns are of type H. Pixels on the odd rows and even columns are of type V. Pixels on odd rows and odd columns are of type D. Every pixel in the image falls into one of these categories.

Our method comprises of a set of rules and a decision tree to choose the rule for each pixel. Figure 5 shows a high level flowchart of the interpolation process. The first part depicts the initial setup. It starts with the LR image being read in. After this, a two pixel wide empty margin is added on all four sides of the LR image. This makes it possible for LNA to process the boundary pixels in the LR image without having additional logic to handle boundary conditions. After the interpolation process, this empty margin is stripped from the interpolated HR image.

After the margin is added, an empty HR canvas, with twice the height and width of the LR image (including the empty margin), is created. Then, a four pixel wide margin is created in the canvas by setting all the pixels in this margin to background color. This ends the setup process.

After setup, the HR canvas is scanned pixel by pixel. The scan starts at the top left corner (0, 0), and ends at the bottom right. We refer to the pixel at the current scan location as the current pixel. The first two decision boxes in the flowchart control the scan.

The third decision box checks if the current pixel has already been assigned a value. This can happen if it is a part of the empty margin, or if the current pixel was assigned a value when the scan was at an earlier pixel. The latter is possible because LNA can assign values to multiple pixels in a single iteration. So, if the current pixel has already been assigned a value, the scan continues to the next location.

The next three checks are to see if the current pixel is an 'O' pixel, 'H' pixel or 'V' pixel. The 'O' pixel is assigned a value using the one rule in Category 1. This does not depend on the neighborhood.

Some rules used to define ‘H’, ‘V’ and ‘D’ pixels, use multiple pixels in the neighborhood to decide the value of a single pixel. The number of pixels a rule considers and the locations of the pixels considered, depend on the values of the pixels, i.e., the image content. The logic used to choose the relevant neighborhood for a rule is explained along with the rules in the subsections III A to III E.

The 'H' pixel is assigned a value using Category 2 rules. An 'H' pixel has two immediate LR neighbors, one on the left and other to the right. Category 2 rules use the LR neighbors, and, under certain conditions, may use a bigger neighborhood. This is discussed in the next section when we discuss individual rules.

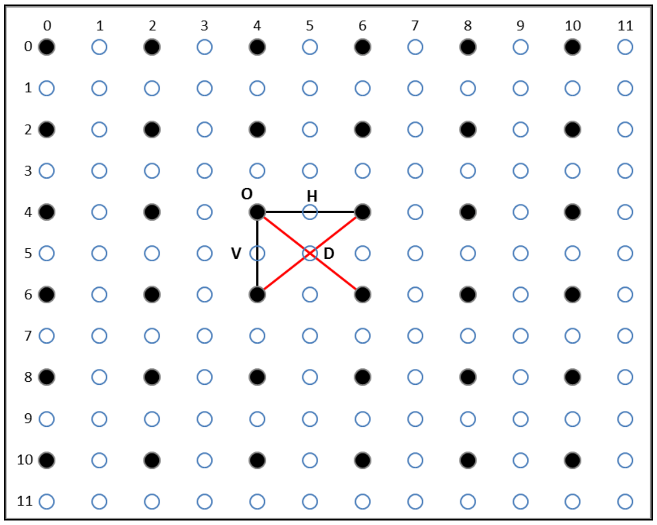

Figure 4. A representation of the HR image showing the types of pixels that need to be generated through interpolation.

The 'V' pixel is assigned a value using Category 3 rules. These are similar to Category 2 rules. The only difference is that a 'V' pixel has LR neighbors above and below.

If the current pixel is not of type 'O', 'H' or 'V', it is a 'D' pixel. These are assigned values using Category 4 rules. Some of the Category 4 rules assign values to pixels that are not in the current scan location. This is shown as the next step in the flow chart. If this path is not taken, there is a possibility that the neighborhood is such that Category 5 rules are to be applied. This is the next step shown.

After the rules pertaining to the current pixel and its neighborhood are applied, the scan moves to the next pixel and the process gets repeated. Once the scan completes, the margin that was added in the beginning is removed and the HR image is stored.

In all the examples, we have used a white foreground and black background.

In the interpolation process, we say that a pixel in the LR image is horizontally thin if its immediate horizontal neighbors, on the left and right, are of different magnitude from it. Similarly, we say that a pixel is vertically thin if the pixel’s immediate vertical neighbors, above and below, are of different magnitude from it.

Some rules can assign values to pixels ahead of the current scan location. Some rules can override or pre-empt other rules, depending on the neighborhood conditions.

Depending on the neighborhood of the pixel, one of the rules in the chosen category is invoked. The rules, in each category, have an order of precedence. If one rule is applied, the rules with lower precedence are not considered. In the following subsections, the rules are described in the order of their decreasing precedence.

Category 5 has one rule and it can change values assigned by other rules.

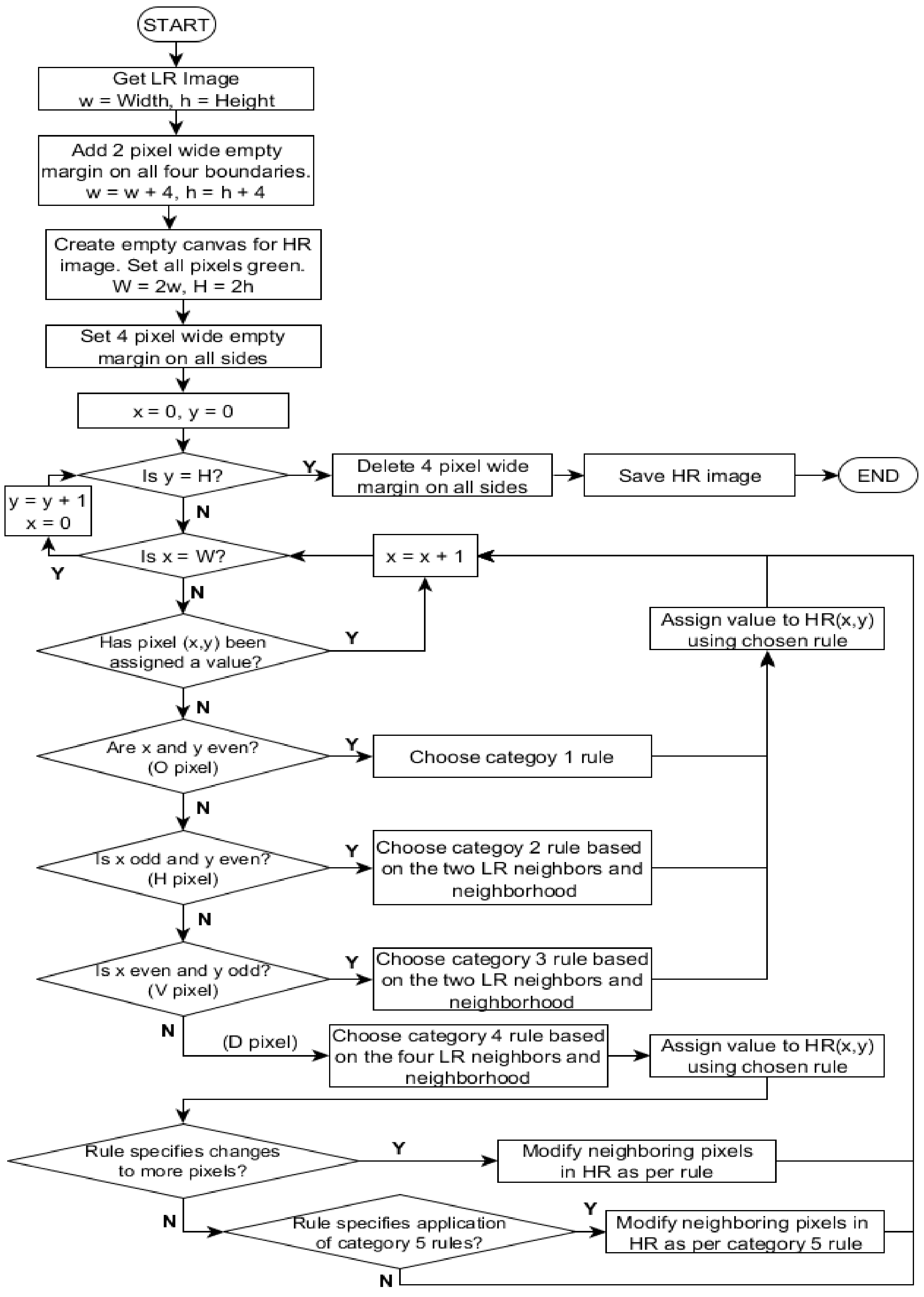
START
Get LR Image
w = Width, h = Height
Add 2 pixel wide empty margin on all four boundaries.
w = w + 4, h = h + 4
Create empty canvas for HR image. Set all pixels green.
W = 2w, H = 2h
Set 4 pixel wide empty margin on all sides
x = 0, y = 0
Is y = H?
Y
Delete 4 pixel wide margin on all sides
Save HR image
END
y = y + 1
x = 0
N
Is x = W?
Y
x = x + 1
N
Has pixel (x,y) been assigned a value?
Y
Assign value to HR(x,y) using chosen rule
N
Are x and y even? (O pixel)
Y
Choose categoy 1 rule
N
Is x odd and y even? (H pixel)
Y
Choose categoy 2 rule based on the two LR neighbors and neighborhood
N
Is x even and y odd? (V pixel)
Y
Choose category 3 rule based on the two LR neighbors and neighborhood
N
(D pixel)
Choose category 4 rule based on the four LR neighbors and neighborhood
Assign value to HR(x,y) using chosen rule
Rule specifies changes to more pixels?
Y
Modify neighboring pixels in HR as per rule
N
Rule specifies application of category 5 rules?
Y
Modify neighboring pixels in HR as per category 5 rule
N

Figure 5. High level flowchart of the LNA process flow and decision tree.

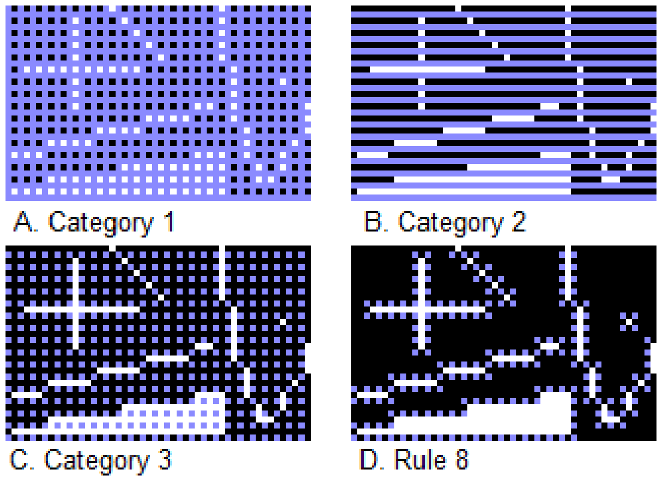

Figure 6. Impact of different rules. The captions show the additional category of rules or specific rule applied.

The method is implemented as a single pass. It starts at the top left corner and scans through the image, row by row. For each pixel, it chooses the appropriate rule and applies it.

We describe the method as a set of rules. Corresponding to each rule or a group of rules, we have a figure showing impact of the rule or group of rules. For example, Figure 6A represents the output if only Category 1 rules are applied and Figure 6B shows the output if both Category 1 and Category 2 rules are applied. The change from Figure 6A to Figure 6B is the impact of the Category 2 rules.

A. Category 1 rule

This category has one rule and applies to all pixels of the type O. In Figure 4, these pixels are shown as filled, black circles. The rule maps all pixels in the LR image to the HR image.

1) Rule 1: Assign the value of the pixel at location (x, y) in the original image (LR) to the pixel at location (2x,2y) in the interpolated (HR) image.

For example, pixels at locations (4,4) and (6,4) in the HR image are assigned values of pixels at locations (2,2) and (3,2) respectively in the LR image. Figure 6A shows the enlarged portion of the HR canvas and it depicts how the empty canvas gets partially populated.

B. Category 2 rules

The three rules in this category apply to unknown pixels of the type H. H pixels have a known horizontal neighbor each on the left and right. In Figure 4, the neighbors of pixel H are shown connected to it by black lines. The values of these neighbors are known because they are the values in the LR image.

1) Rule 2: If the neighbors on the left and right are equal, assign the value of the neighbors to the unknown pixel.

2) Rule 3: If the neighbors on the left and right differ and if only one of them is horizontally thin, assign the value of the pixel that is not thin to the unknown pixel.

3) Rule 4: If none of the preceding rules assigned a value to the unknown pixel, set it to the background color.

Figure 6B is generated by applying Rules 1 to 4. The changes from Figure 6A are caused by the category 2 rules. We see that the horizontal lines, in both colors, have become better formed. We also see that unknown pixels on either side of known pixels, in a vertical line in the foreground color, have been set to the background color.

C. Category 3 rules

These rules are similar to the category 2 rules but apply to unknown pixels of the type V. Such pixels have vertical neighbors with known magnitudes. In Figure 4, the neighbors of pixel V are shown connected to it by black lines. Here we will use the concept of vertical thinness that was defined earlier.

1) Rule 5: If the neighbors above and below are equal, assign their value to the unknown pixel.

2) Rule 6: If the neighbors above and below differ and if only one of them is vertically thin, assign the value of the pixel that is not thin to the unknown pixel.

3) Rule 7: If none of the preceding rules assigned a value to the unknown pixel, set it to the background color.

Figure 6C shows the impact of these rules. The changes from Figure 6B to Figure 6C are caused by the category 3 rules. We see that the vertical lines have become well-formed and more unknown pixels near horizontal lines have been assigned values.

D. Category 4 rules

The eight rules in this category apply to the unknown pixels of the type D. Such pixels have four diagonal neighbors whose magnitudes are known. In Figure 4, the four neighbors of pixel D are shown connected to it by red lines. Unlike the rules in the preceding categories, some of the rules here impact more than one pixel. However, they do not change any pixel that was assigned value by Rule 1.

1) Rule 8: If all four diagonal neighbors are equal, assign the value of the neighbors to the unknown pixel.

Figure 6D is generated by applying Rules 1 to 8. The change from Figure 6C to Figure 6D is caused by Rule 8. We see that most of the unknown pixels have been resolved and solids are well-formed. Most of the unknown pixels that remain are at the edges.

2) Rule 9: If all four neighbors are not equal and diagonally opposite neighbors are equal, then attempt to resolve as follows. If one and only one diagonal pair is both horizontally and vertically thin, then assign its value to the unknown pixel. Else, if all neighbors are horizontally and vertically thin, then assign it the foreground color.

Figure 7A shows the impact of this rule. We see that the diagonal lines are better formed. Unknown pixels adjacent to the diagonal line and also at its end remain unresolved.

3) Rule 10: If all four neighbors are not equal but the diagonally opposite neighbors are equal and the two diagonally opposite pixels in the foreground color are end points of two horizontal or two vertical line segments, assign the foreground color to the unknown pixel. After doing this, apply Rule 16.

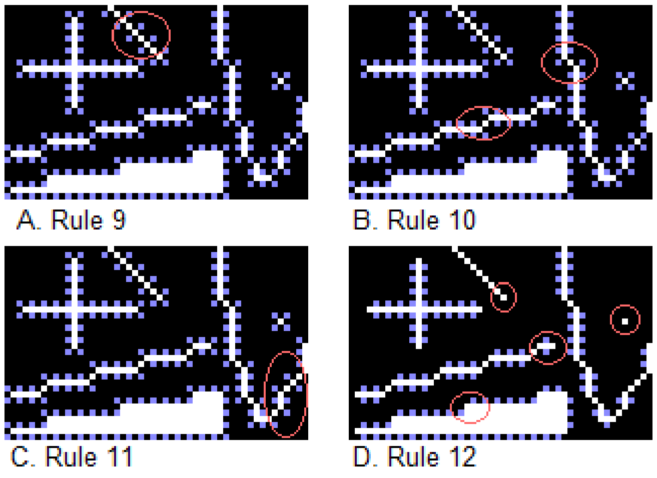

Figure 7. Impact of applying Rules 9-12. The captions show the additional rule and the red call outs show its impact.

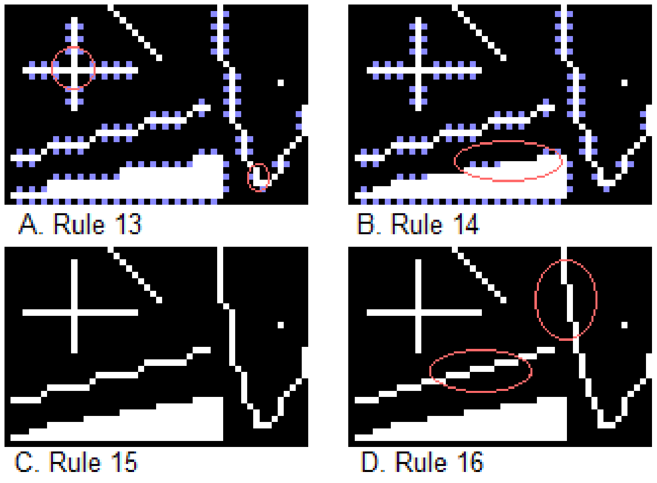

Figure 8. Impact of Rules 13-16. The captions show the additional rule and the red call outs show its impact.

Figure 7B shows the impact of this rule. This rule connects line segments forming longer lines or curves.

4) *Rule 11:* If diagonally opposite neighbors are equal and the preceding rules did not resolve the unknown pixel, assign it the foreground color.

Figure 7C shows the impact is similar to that of Rule 10.

5) *Rule 12:* If the unknown pixel has three diagonal neighbors of the background color, set it to the background color.

This rule makes corners of solids and dots better formed. The impact can be seen in Figure 7D.

The next three rules use the following definitions. These are applicable when only three neighbors are equal to the foreground color. These pixels form two perpendicular segments. Each of these has a length of two pixels or is part of a longer segment. The lengths are from the LR image.

Corner: If both the perpendicular arms have a length of two or if both of them are parts of longer segments.

Slope: If one perpendicular arm is of length two and the other is part of a longer segment.

Well-formed slope: If the longer arm of a slope does not have any adjacent pixel, on the same side as the shorter arm, having the foreground color.

6) *Rule 13:* If the unknown pixel has three neighbors that are a part of a corner, set it to the background color.

Figure 8A shows the impact of this rule. The corners formed by intersecting segments become better formed.

7) *Rule 14:* If three neighbors are part of a slope, set the unknown pixel to the foreground color. If the slope is well-formed, extend the unknown pixel in the direction of the longer arm by the length of the longer arm in the original image. Flag the extension to prevent overwriting.

Figure 8B shows the impact of the rule. Rule 14 differs from the preceding rules as it can impact pixels far removed from the unknown pixel. It makes inclines smoother, as seen on the inclined edge of the solid element in the figure.

This smoothness in the feature is achieved by converting each step like feature into two steps. This makes transitions smaller. This rule can impact pixels about half way across in the image, in either horizontal or vertical directions. The new step drawn is always on an odd numbered row or column. So it does not change any pixel that was assigned a value from the original image by Rule 1.

8) *Rule 15:* If none of the preceding rules assigned a value to the unknown pixel, set it to the background color.

Figure 8C shows the impact of this rule. After Rule 15 is applied, no pixel remains unknown.

E. Category 5 rule

Category 5 has one rule. It is categorized separately because of its unique behavior. It is invoked whenever Rule 10 is applied. If Rule 10 assigns a value to the unknown pixel, two of the diagonal neighbors of the pixel are end points of two horizontal or two vertical segments in the foreground color.

1) *Rule 16:* Draw two segments from the unknown pixel, parallel to the two segments whose endpoints are diagonal neighbors. The length of the new segments should be half the lengths of the corresponding segments in the original image. Set the pixels corresponding to the two original segments that are now adjacent to the new segments, to the background color. Flag all the impacted pixels so that they are not changed later when subsequent pixels are considered.

Figure 8D shows the impact of Rule 16. It is the only rule that changes pixels that were assigned values by Rule 1. Rule 16 helps better interpolate inclined lines where the inclination is not 45 degrees. The impact is seen on curves also because curves are formed using segments and points.

Figure 9 shows another comparison of our method with bicubic interpolation. The differences are clearly visible and the output of LNA is more pleasing.

IV. EXPERIMENTAL RESULTS

In this section we compare LNA with bicubic interpolation, with some recently reported work and with the ideal reference for simple geometric figures. We compare the computation time and interpolation quality. The comparisons are for standard test images and also simple geometric

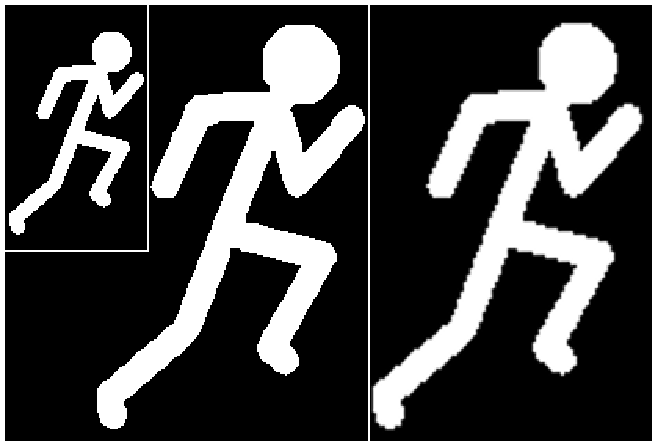
Figure 9. Comparison of zooming. The first image is the input; the second is generated by our method and the third by bicubic interpolation.

shapes. The experiments are on a general purpose computer. Since the execution time for the same process and inputs vary, the experiment is done ten times and the minimum and average time taken is shown. To evaluate the output, we examine visual quality as well as three standard metrics namely PSNR, MPSNR and SSIM. The experiments are performed on 16 different test images from standard sources.

A. *Computation time*

Table I shows the execution time for LNA and bicubic interpolation. The time shown is from ten executions. The time was measured on a computer with an Intel i5-3210M CPU @ 2.50GHz, 4.00 GB RAM and 64 bit Windows 8. The bicubic and LNA interpolations were implemented using Visual C++ in Visual Studio 2010. While measuring time, no other user processes were running. The results show that LNA is faster.

B. *Visual comparison between LNA and bicubic interpolation*

In Figures 10 and 11, we compare the outputs when standard test images are interpolated. The figures shown are Lena, Baboon, Barbara and Monarch. Only a portion of the original images are shown in the figures. This is to ensure that the artifacts are clearly visible. The figures show the portion of the original, low-resolution, color image and its binary version in the first row. The second row shows the output of bicubic interpolation and the third row the output of LNA interpolation.

The original color images were converted to grayscale using the formula:

GRAY = 0.2989 * R + 0.5870 * G + 0.1140 * B

The R (red), G (green) and B (blue) values were in the range 0 to 255 and the resultant GRAY value also is in the same range. The binary image was generated by setting all pixels with grayscale value greater than 127 to white and the rest to black.

The figures clearly show the sharp edges that LNA is able to generate. In these images, the notion of foreground color and background color is difficult to define. Because of this, in certain situations, some of the rules in LNA do not perform the intended function fully. In spite of this, the visual quality is better than bicubic interpolation. The difference between the outputs generated by bicubic and LNA interpolation can be clearly seen at the edges, inclined lines and curves. We can also see that fine features are retained by LNA. This is particularly noticeable in the Monarch and Barbara images in Figure 11.

TABLE I. COMPARISON OF EXECUTION TIME

	Time in milliseconds			
	Bicubic		***LNA***	
	Min	***Ave***	***Min***	***Ave***
lena	38.0	47.1	28.6	36.8
baboon	37.3	43.0	29.8	34.9
peppers	36.2	42.6	27.7	32.4
airplane	39.4	47.1	27.0	33.9
house	9.5	15.9	7.8	11.4
splash	35.6	42.0	30.8	41.3
jellybeans	8.8	11.7	6.9	9.5
car	36.2	42.8	29.2	39.4
sailboat	37.2	45.0	27.2	37.7
san_diego	38.0	41.3	28.7	33.8
earth	36.5	39.8	27.7	34.2
kodim23	53.1	62.7	40.3	49.0
tree	10.2	16.0	7.2	12.2
monarch	54.8	62.5	41.0	51.5
barbara	56.6	70.2	44.3	54.8
goldhill	56.2	63.9	43.7	51.7
Average	**40.3**	**43.4**	**34.0**	**35.3**

C. *PSNR and MPSNR*

When a reference image is available, PSNR and Modified PSNR (MPSNR) are often used as measures of interpolation quality. Table II shows the PSNR and MPSNR of sixteen standard test images, after they were interpolated using LNA and bicubic interpolation.

The data in Table II is for the complete test image, and not portions of the image, as shown in Figures 10 and 11.

We see that LNA gives better PSNR, while bicubic interpolation gives better MPSNR.

It should be noted that some of the LNA rules, like rule 16, reduce PSNR. However, as seen in Figures 10 and 11, this results in better visual quality.

D. *SSIM*

Table II also shows the SSIM index of sixteen standard test images after they were interpolated. We see that the SSIM index, for both LNA and bicubic interpolation, is similar.

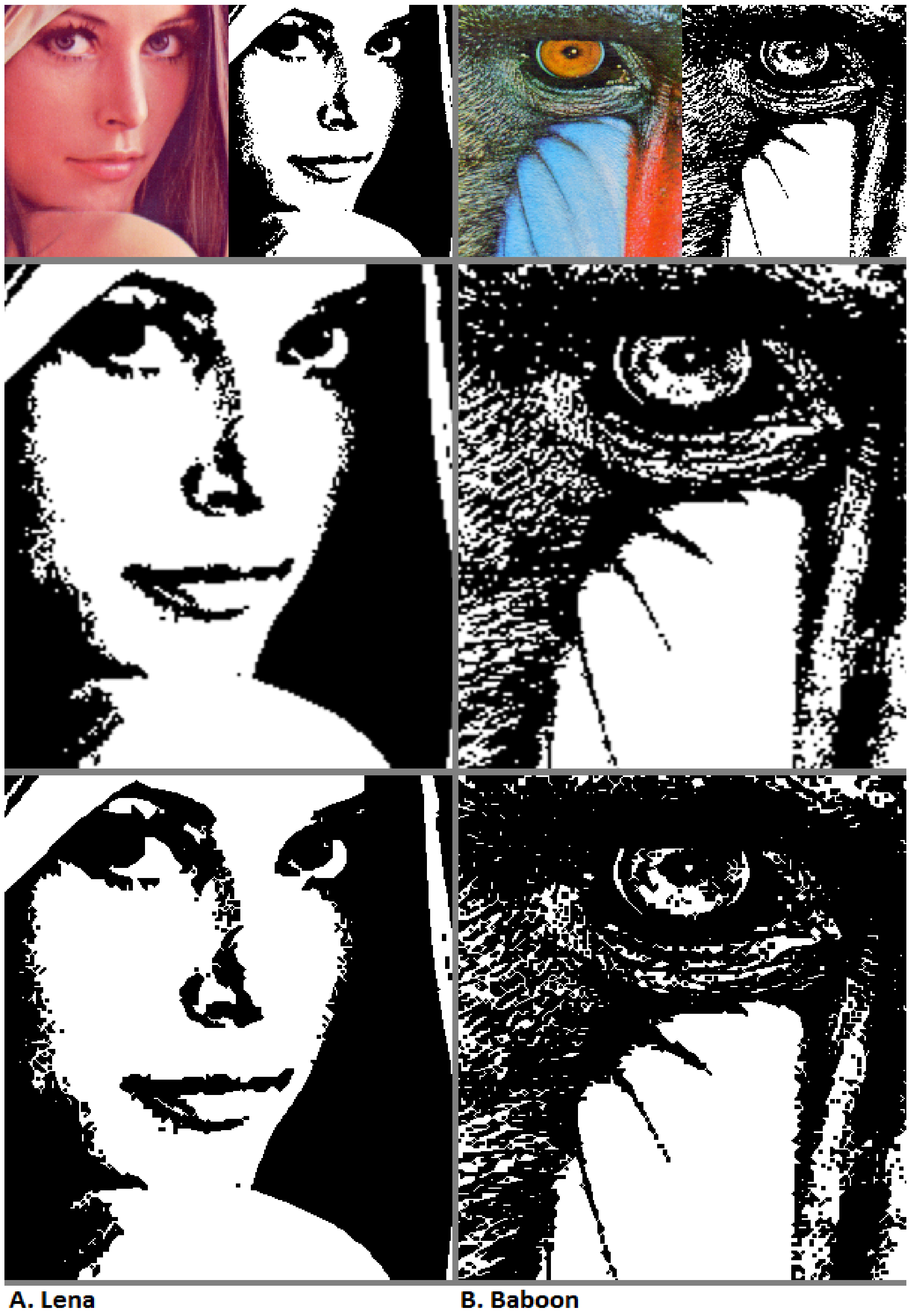

Figure 10. Comparison using Lena and Baboon. Row 1: Original and binary version. Row 2: Output of bicubic. Row 3: Output of LNA.

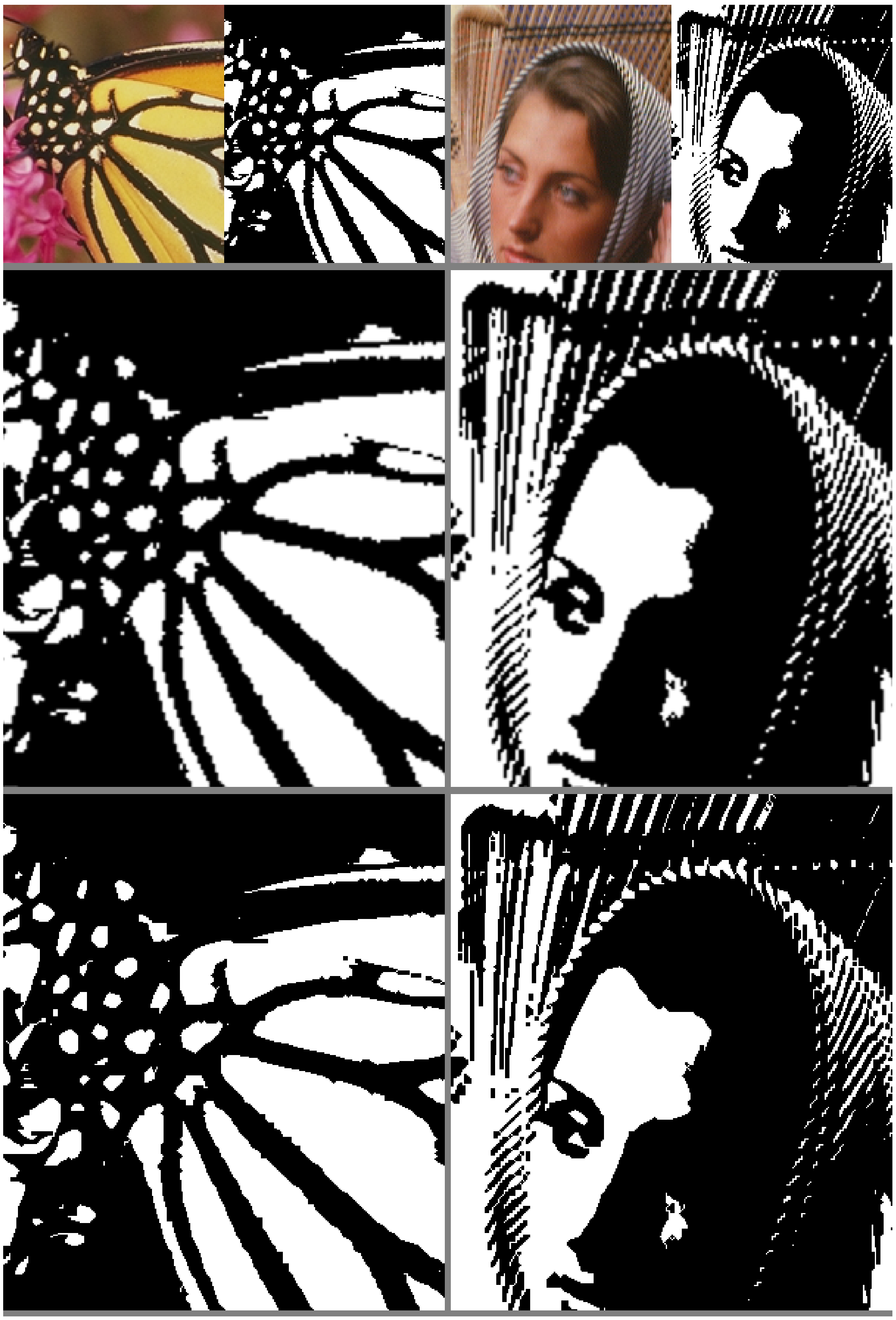

Figure 11. Comparison using Monarch and Barbara. Row 1: Original and binary version. Row 2: Output of bicubic. Row 3: Output of LNA.

TABLE II. COMPARISON OF LNA AND BICUBIC INTERPOLATION USING PSNR, MPSNR AND SSIM

			PSNR in dB		**MPSNR in dB**		**SSIM Index**	
	Height	***Width***	***Bicubic***	***LNA***	***Bicubic***	***LNA***	***Bicubic***	***LNA***
lena	512	512	14.26	14.63	21.64	21.33	0.84	0.84
baboon	512	512	8.73	8.75	16.2	15.14	0.55	0.53
peppers	512	512	15.21	15.59	22.34	22.05	0.85	0.86
airplane	512	512	15.76	15.53	22.93	21.64	0.89	0.88
house	256	256	11.22	13.07	17.8	19.62	0.71	0.74
splash	512	512	20.58	21.73	27.48	28.19	0.95	0.95
jellybeans	256	256	15.63	15.61	22.55	21.78	0.89	0.88
car	512	512	13.13	12.76	20.5	18.92	0.81	0.81
sailboat	512	512	14.79	14.84	22.15	21.24	0.82	0.82
san_diego	512	512	9.69	9.9	17.3	16.57	0.58	0.56
earth	512	512	13.67	13.34	21.34	19.73	0.79	0.77
kodim23	512	768	17.48	17.71	25.13	24.37	0.91	0.92
tree	256	256	12.84	12.97	19.94	19.2	0.79	0.78
monarch	512	768	16.28	16.64	23.7	23.34	0.89	0.89
barbara	576	720	11.61	11.77	19.16	18.64	0.77	0.77
goldhill	576	720	14.18	15.12	21.06	21.54	0.82	0.83
Average			**14.07**	**14.37**	**21.33**	**20.83**	**0.80**	**0.80**

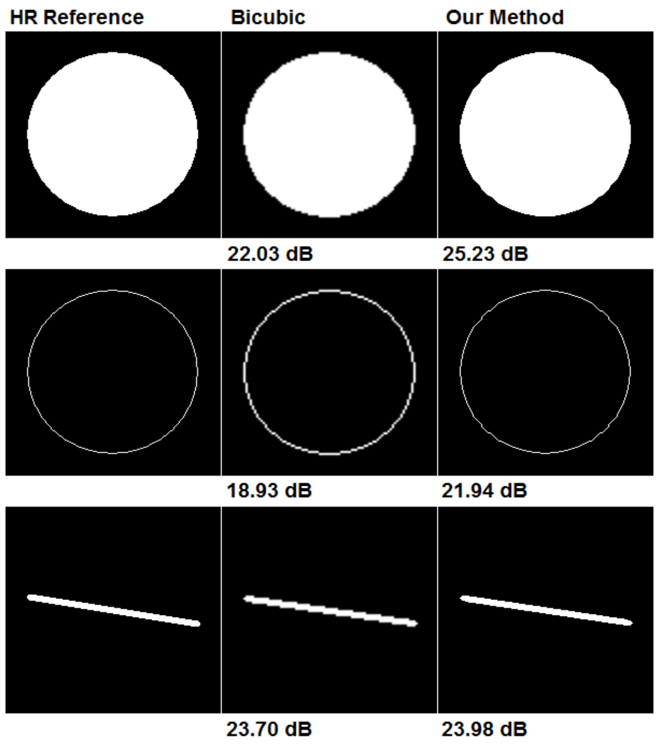

Figure 12. Comparison in Decibel (dB) of interpolation with ideal.reference.

E. *PSNR comparison using ideal reference*

In this section, we evaluate our method using geometric shapes. This allows us to specify the desired result of interpolation and generate reference images in HR for comparison.

The reference HR images, for a scale factor of two, are defined as follows. For a line thickness of one, a line of length l in LR should produce a line length 2l in HR, a circle of radius r should produce a circle of radius 2r and a rectangle of dimension h x w should produce a rectangle of size 2h x 2w. Each of the interpolated images should retain a line thickness of one pixel.

If the source image has thickness, then the thickness is also to be doubled. A line of n pixel thickness and length l, should produce a line of thickness 2n and length 2l, if n > 1.

The input images and reference images were drawn using Visual C++. Lines, rectangles and circles were drawn using the LineTo, Ellipse and Rectangle functions in the CDC class. Line thickness was set using the CreatePen function in the CPen class.

Figure 12 shows the comparison of LNA with bicubic interpolation. In the figure, the first test case is a filled circle. The reference image was drawn as a filled circle of radius 80. The input to bicubic interpolation and to our method was a filled circle of radius 40. The same approach was used to generate reference images for other shapes also. The Bicubic interpolation was done using Matlab.

TABLE III. COMPARISON OF PSNR AND MPSNR

	Thickness		PSNR in dB		MPSNR in dB	
	LR	*HR*	*Our method*	*Bicubic*	*Our method*	*Bicubic*
Rectangle	1	1	match	22.98	match	27.33
Circle	1	1	21.94	18.93	36	24.33
Line - 45 degree	1	1	46.02	26.54	56.16	31.92
Line - 10 degree	1	1	24.35	23.11	35.43	27.98
Rectangle	3	6	22.37	21.10	27.57	25.13
Circle	3	6	21.46	19.96	29.19	25.27
Line - 45 degree	3	6	25.19	27.40	32.14	34.68
Line - 10 degree	3	6	23.98	23.70	30.09	28.12
Filled Rectangle	NA	NA	match	26.49	match	30.85
Filled Circle	NA	NA	25.23	22.03	33	26.76

TABLE IV. COMPARISON OF OUR METHOD WITH OTHER METHODS

	Percentage of PSNR (dB) improvement over Bicubic				
	Our Method		*CIM [5]*	*Gradient Orientation [14]*	*NNV [15]*
	PSNR	MPSNR			
Lena	2.59	12.90	2.35	0.92	
Peppers	2.50	14.12	1.09		1.71

Table III shows the comparison for more simple images using both PSNR and MPSNR. We see good PSNR improvement by both measures. Table III also shows a PSNR decrease for a three pixel thick line at 45 degrees. In Figure 13, this image is analyzed. The figure shows a portion of the image, marked in red, magnified 8 times. In the magnified region, a set of colored squares with the same size as a pixel, have been shown just below the line. Using these pixels to help count, we see that the reference line is 8 pixels wide along the x axis, while our method has generated a line of width 7 pixels. This happens because, in many situations, our method assigns the background color when other rules don't resolve an unknown pixel. This biases images towards thinness and the bias is of one pixel. This helps the image look sharp but the difference in thickness is reflected in the lower PSNR.

F. Comparison with some similar, recent work

A direct comparison of our method with results available in [1]-[15] is difficult because our method is only formulated for binary images. To do a comparison, we converted two of the commonly used images, Lena and Peppers, to binary and used these as the reference images. We decimated these images by a factor of 2 and then interpolated them back to original size. We compared the interpolated images with the reference. The results are shown in Table II. The results have to be viewed keeping in mind the fact that the input for our experiments is binary while the input to the other methods is a grayscale image.

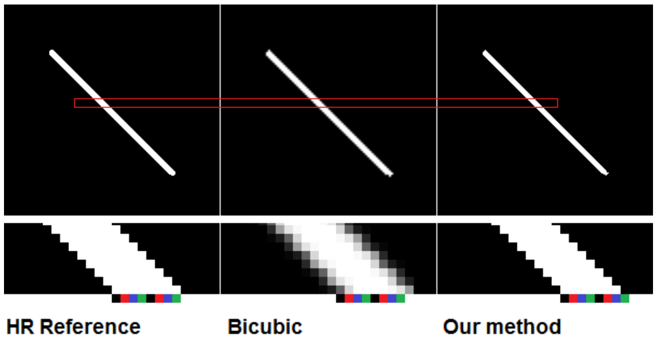

Figure 13. Magnified comparison of the outputs of interpolation.

In [13], a text super-resolution is considered. Here the input is binary. It uses text images for training. It achieves an improvement between 0% and 19% in Mean Square Error (MSE), when compared with pixel replication. The results are for different text symbols. Our method improved MSE by 5.7% for Lena and 2.1% for Peppers.

G. Analysis of the results

The results presented in this section show that LNA is computationally more efficient than bicubic interpolation. The amount of processing required in LNA depends on the image content.

Visual comparison shows that LNA generates visually pleasing output. This is true for the geometric shapes and also the standard test images. This is because LNA is designed to maintain sharpness and to generate smooth lines in the interpolated image.

Quantitative quality comparison shows that LNA is better in terms of PSNR while bicubic is better in terms of MPSNR. This is because LNA is designed to keep features sharp and so, on an average, the error in pixels is lower. This reflects in the higher PSNR. LNA is inferior in terms of MPSNR because it is computed after passing the images through a low pass filter. This blurs the image edges. LNA is designed to keep the edges sharp and so when the PSNR is computed after filtering both the reference and target, the result is inferior.

V. CONCLUSIONS

To generate visually pleasing, magnified images, we have proposed a new technique for binary images. It uses non-linear zooming rules that are Location and Neighborhood Adaptive (LNA). These rules are inspired by the way an artist would enlarge an image. The method overcomes a number of problems, such as blurring and thickening of edges that are associated with known interpolation techniques. The results are visually appealing. Lines and dots, with single pixel thickness, retain their thickness. Inclined lines, curves and solids look much better compared to other interpolation methods.

In LNA, some rules need the foreground color as an input. Some other rules assign a default color because the method could not determine a pixel's color based on the inputs it considered. Furthermore, a study is required to improve these rules. Studies are required to analyze the possibility of using larger neighborhoods as input, to further improve the quality of interpolation.

A consequence of LNA is that it may change relative sizes of some objects. Objects without thickness remain without thickness while those with thickness, increase in thickness. So, if an object is formed using both these kinds of components, the change in relative thickness becomes visible. Also, a study is required to devise mechanisms to scale all components of an object consistently.

In the future, a study is needed to extend this method to grayscale and color images. A solution that is usable could probably be built by working with ranges of color values, and using functions to specify values for unknown pixels.

References

[1] P. J. Prabhakaran, and P. G. Poonacha, “A Novel Location and Neighborhood Adaptive Method for Binary Image Interpolation,” SIGNAL 2017: The Second International Conference on Advances in Signal, Image and Video Processing, pp. 14-20, 2017.

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C, 2nd ed. Cambridge University Press, pp. 125-127, 1992.

[3] P. J. Prabhakaran and P. G. Poonacha, "A new decimation and interpolation algorithm and an efficient lossless compression technique for images," Communications (NCC), 2015 Twenty First National Conference on, pp. 1-6, 2015.

[4] H. Jiang and C. Moloney, “A new direction adaptive scheme for image interpolation,” International Conference on Image Processing, Vol. 3, pp. 369-372, 2002.

[5] H. Kim, Y. Cha, and S. Kim, “Curvature Interpolation Method for Image Zooming,” IEEE Transactions on Image Processing, Vol. 20, No. 7, pp. 1895-1903, July 2011.

[6] R. Gao, J. P. Song, and X. C. Tai, “Image zooming algorithm based on partial differential equations technique,” International Journal of Numerical Analysis and Modelling, Vol. 6, No. 2, pp. 284-292, 2009.

[7] L. Jing, Z. Gan, and X. Zhu, "Directional Bicubic Interpolation-A New Method of Image Super-Resolution," 3rd International Conference on Multimedia Technology (ICMT-13). Atlantis Press, pp. 470-477, November 2013.

[8] J. Sun, Z. Xu, and H. Y. Shum, "Image super-resolution using gradient profile prior," 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, pp. 1-8, 2008.

[9] C. Y. Yang and M. H. Yang, "Fast Direct Super-Resolution by Simple Functions," 2013 IEEE International Conference on Computer Vision, Sydney, NSW, pp. 561-568, 2013.

[10] R. Hardie, “A fast image super-resolution algorithm using an adaptive Wiener filter,” IEEE Transactions on Image Processing, Vol. 16, No. 12, pp. 2953-2964, 2007.

[11] S. Lerttrattanapanich and N. K. Bost, “High resolution image formation from low resolution frames using delaunay triangulation,” IEEE Transaction on Image Processing, Vol. 11, No. 12, pp. 1427-1441, 2002.

[12] W. T. Freeman, T. R. Jones, and E. C. Pasztor, “Example-Based Super-Resolution,” IEEE Comput. Graph. Appl. 22, 2, pp. 56-65, 2002.

[13] G. Dalley, B. Freeman, and J. Marks, "Single-frame text super-resolution: a Bayesian approach," Image Processing, 2004. ICIP '04. 2004 International Conference on, 2004, Vol. 5, pp. 3295-3298, 2004.

[14] S. Ousguine, F. Essannouni, L. Essannouni, and D. Aboutajdine, "A new image interpolation using gradient-orientation and cubic spline interpolation," ISSR-Journals, vol. 5, no. 3, 2014.

[15] O. Rukundo and C. Hanqiang, "Nearest Neighbor Value Interpolation," in 2014 International Conference on Computer Vision Theory and Applications (VISAPP), 2012.

[16] The University of Southern California – Signal and Image Processing Institute (USC-SIPI) image database. "http://sipi.usc.edu/database/database.php". [Online; accessed 31-July-2017].

[17] Kodak lossless true color image suite. http://r0k.us/graphics/kodak/". [Online; accessed 31-July-2017].

[18] Index of testimages. "http://www.hlevkin.com/TestImages/". [Online; accessed 31-July-2017].