# A Novel Location and Neighborhood Adaptive Method for Binary Image Interpolation

Pullat Joy Prabhakaran, Palanganda Ganapathy Poonacha
International Institute of Information Technology – Bangalore,
Electronic city, Bangalore, India 560100
Email: joy@iiitb.ac.in, 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. We have shown that 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 is a variable. Some rules can be influenced by and influence distant pixels. We present examples showing the effectiveness of our method. The results are visually appealing. 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-Interpolation; binary-image; thinness; corner; slope.***

## I. INTRODUCTION

When High Resolution (HR) images are created by interpolating Low Resolution (LR) images using popular methods like nearest neighbor, bilinear and bicubic interpolation, unpleasant artifacts are seen. Two commonly noticed artifacts are smoothing of edges and pixelation. These 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. Such methods cause more unwanted artifacts in the case of binary image zooming.

A large number of interpolation methods are available in the literature [1]–[14]. Some of these, like Nearest Neighbor, Bilinear and Bicubic [1] 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 [2], 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 [4], 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 [5], 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 [6]. 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 [6] because it also tries to find and preserves local edges.

In [7] and [8], a two-step super resolution process is studied. In the first step, the low resolution image is interpolated using Bicubic interpolation. In the second step, the interpolated image is further processed to improve the quality at the edges. In [7], the gradient profile of the low resolution image is used as a driving gradient prior to change the gradient profile of the interpolated image. This process makes the edges sharper. In [8], 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 [9] and [10], 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 training based approach is discussed in [11]. Unknown pixels in the interpolated image are generated using the training data set that best matches. 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. A training based method, to expand binary text images, with an explicit noise model is discussed in [12].

In [13], edges are found as a first step. The edges are used to compute unknown pixels using cubic spline. In [14], unknown pixels are assigned the value of the neighbor that is closest to the value got by bilinear interpolation.

In this paper, we have developed interesting rules, based on location and nature of content in the neighborhood, for the interpolation of binary images by a scale factor of 2. These rules derive inspiration from the way an artist might zoom an image. We present a method of obtaining interpolated image pixels using sixteen rules, grouped into five categories. The rules are of widely varying complexities.

 ISBN: 978-1-61208-559-3 

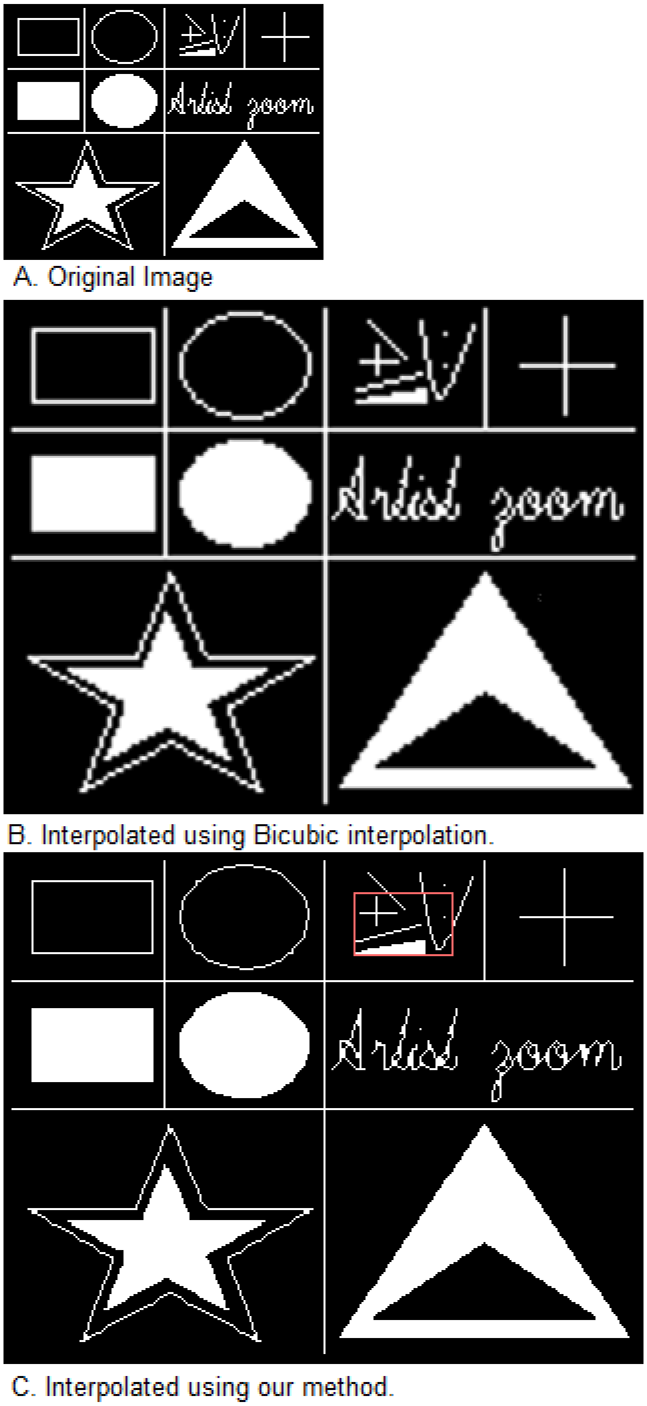


Figure 1. Comparison of our method with bicubic interpolation.

The choice of the rule to assign value to a particular 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. Similarly, some rules can influence distant pixels in the output.

If the neighborhood meets certain conditions, our method 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 compared with simple pixel replication. The deletion ensures that smoothing does not cause extra thickening.

Figure 1A shows the image we have used to explain our method. Figure 1B shows the image, interpolated using bicubic interpolation. The bicubic interpolation is done using Matlab. Figure 1C 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 1C, shown in the red box, will be used to explain our method.

The rest of this paper is organized as follows. Section 2 describes the interpolation process and five categories of rules. Sub sections A to E, in Section 2, describe the categories and associated rules. Experimental results are given in Section 3. Conclusions and suggestions for further extensions are given in Section 4.

## II. THE INTERPOLATION PROCESS

For each unknown pixel, the method does four things. Based on the location, it gets the neighbors and decides the applicable category of rules. The neighbors are from the LR image. Based on the content, it determines the neighborhood to be considered. The neighborhood can extend well beyond immediate neighbors. Based on the neighborhood, the method chooses the rule to be applied. The rule sets the unknown pixel and may also assign values to other pixels.

The interpolation process starts with an empty canvas that is double the height and width of the input image. We use blue color to represent pixels in the empty canvas. These will be assigned values by applying appropriate rules. We refer to these blue pixels as unknown pixels. At the start of the process, all the pixel values are unknown.

We assume that the row and column numbering start at the top left corner of the image. The first row and first column are referred to as row zero and column zero respectively.

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

In order to handle the boundary pixels in a uniform way, we assume a two pixel wide background region on all four boundaries.

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

The method categorizes unknown pixels in the HR canvas based on their locations. This is shown in Figure 2. 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. 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.

 ISBN: 978-1-61208-559-3 

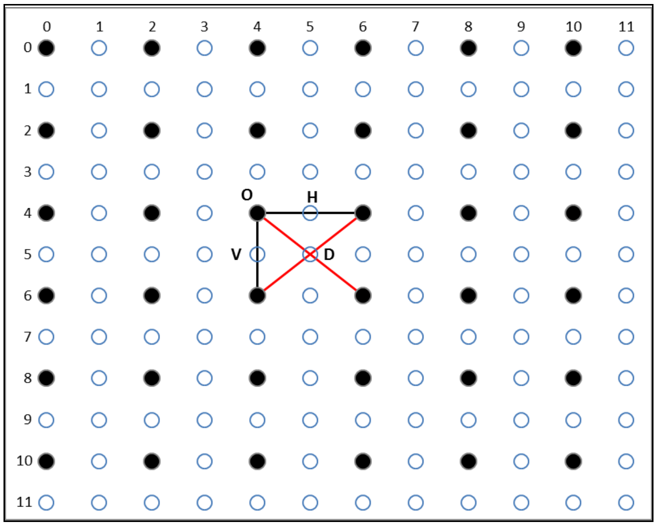


Figure 2. A representation of the HR image showing the types of pixels that need to be generated through interpolation.

The pixels of type O, H, V and D are assigned values using Category 1 to 4 rules respectively. The rules to assign value to pixels, in each category, are discussed in sub-sections A to D.

Some rules can override or pre-empt other rules, depending on the neighborhood conditions.

Different rules use data from neighborhoods of different sizes. In some cases, the size of the neighborhood is adaptive and it depends on the content in the neighborhood.

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 applied even if their invocation conditions are met. In the following sub-sections, 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.

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 3A represents the output if only Category 1 rules are applied and Figure 3B shows the output if both Category 1 and Category 2 rules are applied. The change from Figure 3A to Figure 3B 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 2, 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.

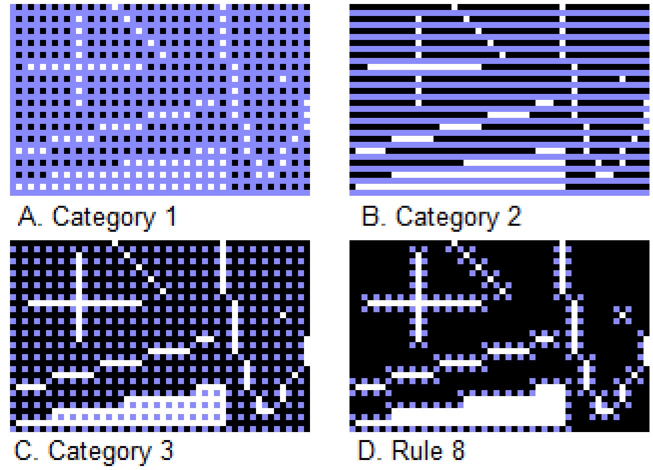


Figure 3. Impact of different rules. The captions show the additional category of rules or specific rule applied.

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 3A shows the enlarged portion of the destination canvas and it depicts how the empty destination 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 2, 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 3B is generated by applying Rules 1 to 4. The changes from Figure 3A 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 2, 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.

 ISBN: 978-1-61208-559-3 

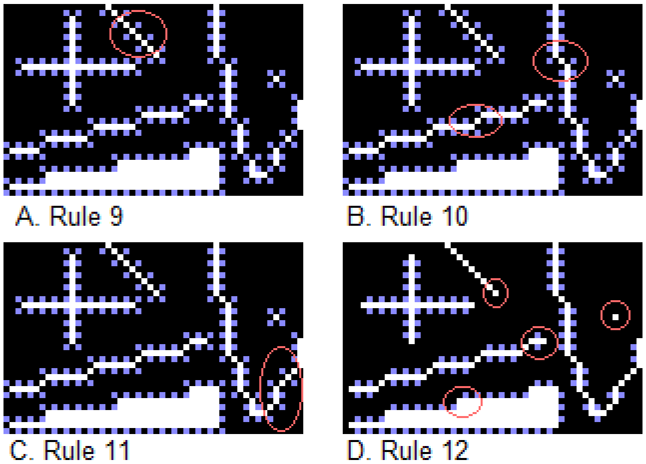


Figure 4. Impact of Rules 9-12. The captions show the additional rule and the red call outs show its impact.

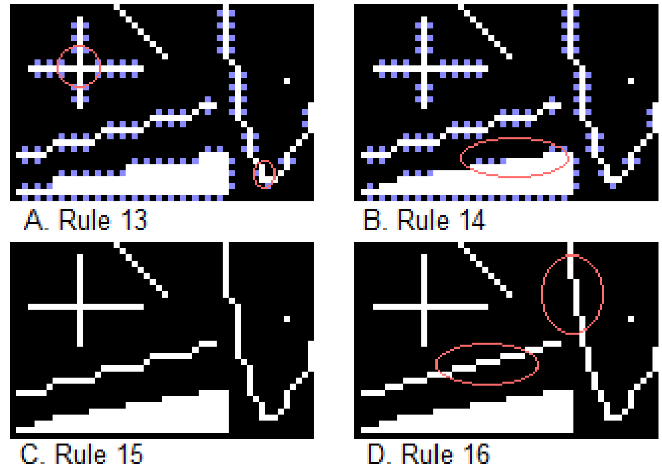


Figure 5. Impact of applying Rules 13-16. The captions show the additional rule and the red call outs show its impact.

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

Figure 3C shows the impact of these rules. The changes from Figure 3B to Figure 3C 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 2, 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 3D is generated by applying Rules 1 to 8. The change from Figure 3C to Figure 3D 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 4A 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 4B 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 4C 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 4D.

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 two pixels or is part of a longer segment. The lengths are with reference to 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 5A 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 5B 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.


ISBN: 978-1-61208-559-3

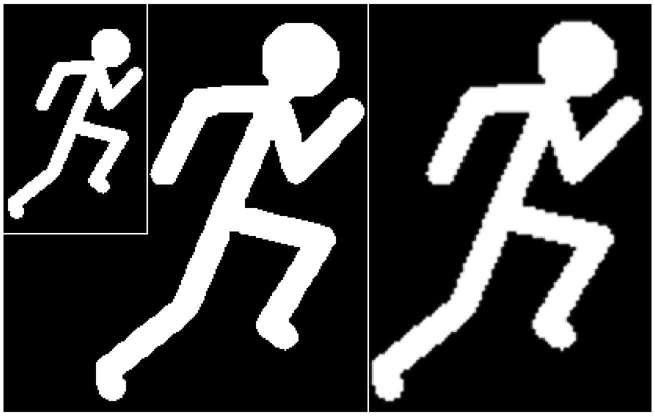

Figure 6. Comparison of zooming. The first image is the input; the second is generated by our method and the third by bicubic interpolation.

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 5C 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 5D 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 6 shows another comparison of our method with bicubic interpolation. The differences are clearly visible and the output of our method is more pleasing.

## III. EXPERIMENTAL RESULTS

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.

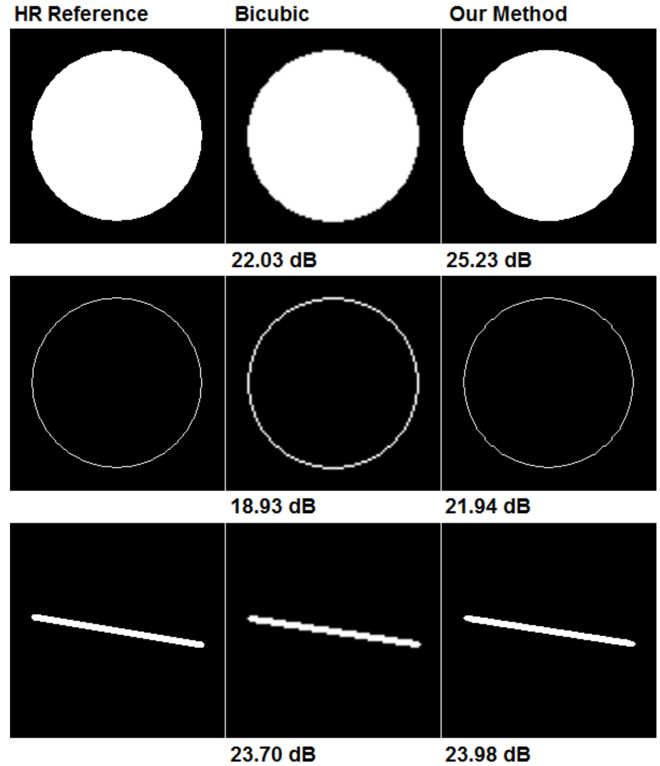


Figure 7. PSNR comparison of Bicubic interpolation and our method.

TABLE I. 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 |

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.

 ISBN: 978-1-61208-559-3 

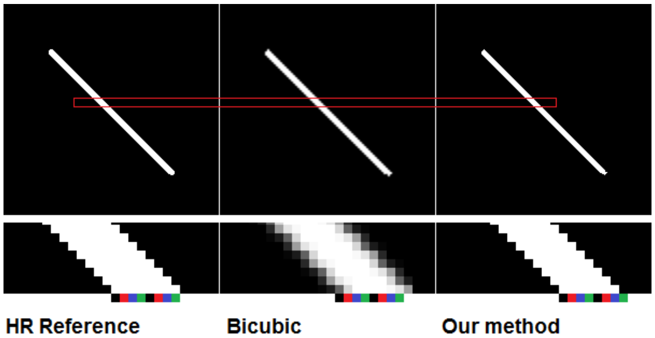


Figure 8. Magnified comparison of the outputs of interpolation.

TABLE II. COMPARISON OF OUR METHOD WITH OTHER METHODS.

| | Percentage of PSNR improvement over Bicubic | | | | |
|---|---|---|---|---|---|
| | *Our Method* | | *CIM [4]* | *Gradient Orientation [13]* | *NNV [14]* |
| | PSNR | MPSNR | | | |
| Lena | 2.59 | 12.90 | 2.35 | 0.92 | |
| Peppers | 2.50 | 14.12 | 1.09 | | 1.71 |

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 7 shows the comparison of our method 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 I shows the comparison for more images using both PSNR and Modified PSNR (MPSNR). MPSNR is generated 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. We see good PSNR improvement by both measures. Table I also shows a PSNR decrease for a three pixel thick line at 45 degrees. In Figure 8, 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.

A direct comparison of our method with results available in [1]-[14] 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.

In [12], 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.

We compared the execution times of our method with bicubic (on a computer with Intel i5-3210M CPU @ 2.50GHz, 4.00 GB RAM and running 64 bit Windows 8) by running ten iterations. The minimum time taken for bicubic interpolation of Lena and Peppers was 37.98 and 36.16 milliseconds respectively. The corresponding values for our method were 28.56 and 27.68 milliseconds.

## IV. CONCLUSIONS

We have developed a new method of zooming binary images using rules inspired to some extent by what an artist may do. All images shown in this paper are generated by a computer program that implements the rules discussed. The results from our method are visually appealing. 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. Also, corners retain sharpness.

From the results, we observe that one pixel thin lines remain thin while thick lines become thicker in our method. This is a desirable feature and one of the goals of our method. However, this might not lead to visually appealing results for fonts. This aspect requires more study.

More work is needed to extend this method to grayscale and color images. A useful solution could probably be built by working with ranges of color values and using functions to specify values for unknown pixels. This can lead to better image zooming techniques due to its location and content based adaptive nature.

## REFERENCES

[1] 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.

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

[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. 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

 ISBN: 978-1-61208-559-3 