ONTARIO TECH UNIVERSITY
FACULTY OF SCIENCE

Project Group 26 December 8, 2023

Bicubic Interpolation for Image Resizing

Ravichandra Pogaku Gowtham Rajendra Ivan Wang

Abstract: Bicubic interpolation is similar to bilinear interpolation but with the use of cubic curves instead of linear lines in two dimensions. The two bicubic resizing methods explored are the bicubic Convolution method and the bicubic Spline method. The theory for both methods already exist so this paper will focus on practical application of these methods. The results discovered from both methods is that the Spline method is slower but yields greater accuracy in the resized image while the Convolution method is much faster while being slightly more accurate than bilinear resizing.

1 Introduction

For the course project, we decided to focus on image resizing with bicubic interpolation. We performed nearest neighbor and bilinear interpolation for the labs and this is an extension of those to compare the different resizing methods. We will use two methods for bicubic interpolation: the bicubic convolution method and the bicubic spline method. Furthermore, our application will include the other functionalities implemented in previous labs to deliver an image editing software with a graphical user interface (GUI). So, in addition to image resizing, the software will include histogram equalization, image blurring, colour transformation (saturation, contrast, colour palette) and image stylizing (giving the image a "painted look"). The software will also include simple standard features such as image saving/opening and saving settings applied to the image.

2 Bicubic Convolution Interpolation

2.1 Introduction of Algorithm

Resizing images using the convolution method requires the creation of the kernels for the x and y dimensions, as well as a matrix of 16 neighboring pixels values. Using these kernels and the values matrix, we can obtain the new pixel values for the resized image using convolution.

2.2 The Interpolation Kernel

(Keys, 1981) explains that the kernel is created based on four cubic polynomials in the interval (-2, 2). This interval consists of the subintervals (-2, -1), (-1, 0), (0, 1) and (1, 2) where points not in these intervals causes the interpolation kernel to be 0 (Keys, 1981). This also means that only four sampling points can be used to create this kernel (Keys, 1981). Although kernel was created for a one-dimensional scenario, this kernel formula can also be applied it to a two-dimensional scenario by creating a kernel in each of the two dimensions, x and y (Keys, 1981). The final version of the kernel is shown in the equation below where a is usually set to -0.5 or -0.75 and s is a sampling point.

$$u(s) = \begin{cases} (a+2)|s|^3 - (a+3)|s|^2 + 1 & 0 \leq |s| \leq 1 \\ a|s|^3 - 5a|s|^2 + 8a|s| - 4a & 1 < |s| \leq 2 \\ 0 & 2 < |s| \end{cases} \tag{1}$$

We decided to use $a = -0.5$ since (Keys, 1981) concluded that it was the most optimal value for accuracy and efficiency. We also tested the difference between $a = -0.75$ and $a = -0.5$ and found no noticeable differences in the resized image.

2.3 Applying the Algorithm

Before the algorithm can be applied, the image needs to be padded by two pixels on each side because we are convolving a 4x4 patch of pixels. The next step is to iterate through the new image's dimensions to fill it in with the interpolated pixels. We need to create kernels and the value matrix for each pixel.To create the kernels, we need four x values and four y values to be sent to the interpolation kernel function. We find these by scaling them to values between 0 and 2. These values will be used to evaluate (1) to create the following kernels.

$$U_x = \begin{bmatrix} u(x_1) & u(x_2) & u(x_3) & u(x_4) \end{bmatrix} \quad U_y = \begin{bmatrix} u(y_1) \\ u(y_2) \\ u(y_3) \\ u(y_4) \end{bmatrix} \tag{2}$$

We can then use the scaled x and y values to find the neighboring pixels in the original image to create the values matrix. Each value in the matrix are two samples from all sides of the current pixel. This is demonstrated in (3) where I is the original image and P is the points. A shortened version is shown in (4).

$$P(x,y) = \begin{bmatrix} I_{x-x_1,y-y_1} & I_{x-x_1,y-y_2} & I_{x-x_1,y+y_3} & I_{x-x_1,y+y_4} \\ I_{x-x_2,y-y_1} & I_{x-x_2,y-y_2} & I_{x-x_2,y+y_3} & I_{x-x_2,y+y_4} \\ I_{x+x_3,y-y_1} & I_{x+x_3,y-y_2} & I_{x+x_3,y+y_3} & I_{x+x_3,y+y_4} \\ I_{x+x_4,y-y_1} & I_{x+x_4,y-y_2} & I_{x+x_4,y+y_3} & I_{x+x_4,y+y_4} \end{bmatrix} \tag{3}$$

$$P(x,y) = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix} \tag{4}$$

Finally, the kernels shown in (2) and the value matrix shown in (4) can be used to find the pixel value at each position of the resized image. This operation is shown in (5) where I_{new} is the resized image.

$$I_{new(x,y)} = \begin{bmatrix} u(y_1) \\ u(y_2) \\ u(y_3) \\ u(y_4) \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix} \begin{bmatrix} u(x_1) & u(x_2) & u(x_3) & u(x_4) \end{bmatrix} \tag{5}$$

This method results in a more accurate resized image than the nearest-neighbor and bilinear method while also having a fast computation speed.

3 Bicubic Spline Interpolation

3.1 Introduction of Algorithm

The bicubic spline method is done in a similar way as bilinear interpolation. The difference between the two methods being that the bicubic spline method uses 4x4 patches of the image instead of 2x2 patches to fit a bicubic surface to the 2x2 center of that patch. This is done by taking into account not only the values of the 4 corners of the 2x2 center patch but also the x, y, and mixed partial derivatives at those 4 corners (Bicubic Interpolation).

3.2 Image Derivative

A popular kernel for calculating image derivatives is the sobel filter (Sobel Operator).

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

$$p_x(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} i a_{ij}x^{i-1} y^j$$

$$p_y(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} j a_{ij}x^i y^{j-1}$$

$$p_{xy}(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} ij a_{ij}x^{i-1} y^{j-1}$$

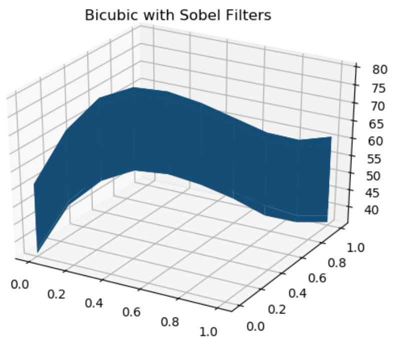
Figure 1: 2x2 Bicubic Surface Equations

$$s_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

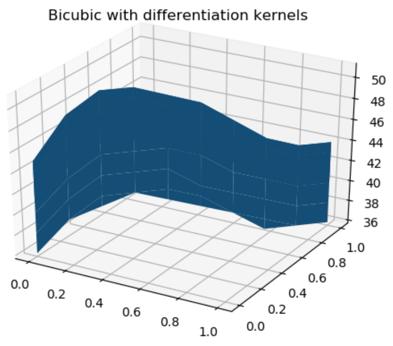
These kernels can be convolved with an image to get the x and y derivatives respectively. The sobel filter is a separable filter composed of a smoothing kernel, to reduce noise, and differentiation kernel, to calculate the derivative (Sobel Operator).

$$s_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

We tested the difference in interpolation quality between using the sobel filters and just the differentiation kernels.



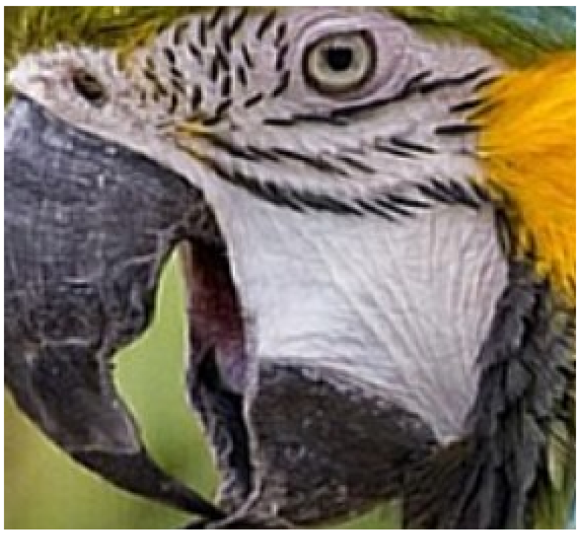
(a) Sobel Filters



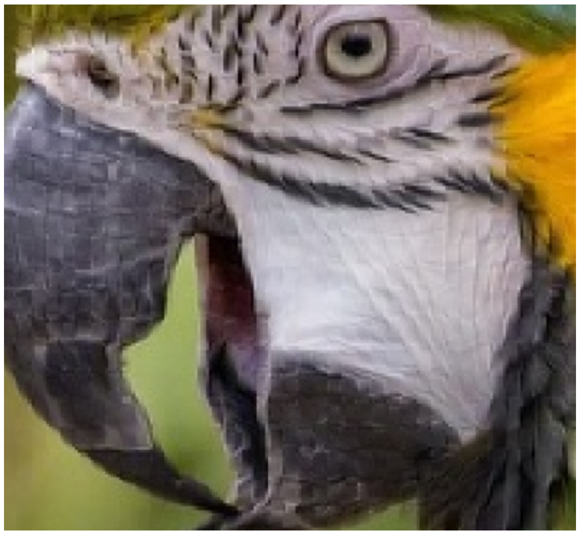
(b) Differentiation Kernels

Figure 2: Comparison of the two filters

As seen in Figure 2, convolving the patch with the sobel filters results in the interpolated bicubic surface being much smoother with a lot more overshoot/undershoot which results in a sharper image.



(a) Sobel Filters



(b) Differentiation Kernels

Figure 3: Comparison of the resulting images

We have concluded that the computational savings from using the smaller differentiation kernels does not outweigh the sharpness of the resulting image when using the sobel filters.

3.3 Solving System for Coefficients

Following the equations from Figure 1 will yield 16 equations that make up a surface on the 2x2 center patch. The coefficients a_{ij} are unknown so we separate the coefficients to create a system of linear equations in the form of $Ax = b$ (Bicubic Interpolation).

$$A = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&2&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0\\
0&1&2&3&0&1&2&3&0&1&2&3&0&1&2&3\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&2&0&0&0&3&0&0&0\\
0&0&0&0&1&1&1&1&2&2&2&2&3&3&3&3\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&2&3&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&2&0&0&0&3&0&0\\
0&0&0&0&0&1&2&3&0&2&4&6&0&3&6&9
\end{bmatrix} \tag{6}$$

$$x = \begin{bmatrix} a_{00} & a_{10} & a_{20} & a_{30} & a_{01} & a_{11} & a_{21} & a_{31} & a_{02} & a_{12} & a_{22} & a_{32} & a_{03} & a_{13} & a_{23} & a_{33} \end{bmatrix}^T$$

$$b = \begin{bmatrix} f(0,0) & f(1,0) & f(0,1) & f(1,1) & f_x(0,0) & ... & f_y(0,0) & ... & f_{xy}(0,0) & ... & \end{bmatrix}^T$$

Unable to fit the entire b vector on the page, repeat the same 4 points for f_x, f_y, and f_{xy}.

Rearrange system to $A^{-1}b = x$ to easily calculate the unknown coefficients. A is always the same so its inverse can be hard coded for efficiency. The coefficients can then be used to interpolate points $p(x, y)$ on the surface. Note that the x and y values should be scaled to range from $[0, 1]$ (Bicubic Interpolation).

$$p(x,y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix} \tag{7}$$

3.4 Applying the Algorithm

This method requires 4x4 patches of data so you will need to pad your input image with 1 pixel on each side so that points on the border of the image can be interpolated. Then you can either calculate all the sampling points beforehand or as you need them. Iterate through your padded image in 4x4 patches, calculate the bicubic coefficients for that patch by solving for x using (6) and then interpolate the values at the relevant sampling points using (7), if any. If there aren't any sampling points in the center 2x2 of a particular 4x4 patch, then you may skip it altogether to save time. Repeating this process with every 4x4 patch of the image and applying the scaled sampling points as the x and y values will result in a resized image that has higher sharpness than bilinear interpolation can produce.

4 Comparing Convolution and Spline Methods

The graph in Figure 4 shows that the bicubic spline method takes significantly more time to resize images in comparison to both bicubic convolution and bilinear interpolation. However the quality of the image resized with the spline method is much sharper than both the convolution and bilinear methods. This can be seen in Figure 5 where the details of on the beak and the cheeks of the bird are more clear when the bicubic spline method is used. While the details of the resized image using the bicubic convolution method is only somewhat more clear than bilinear interpolation.

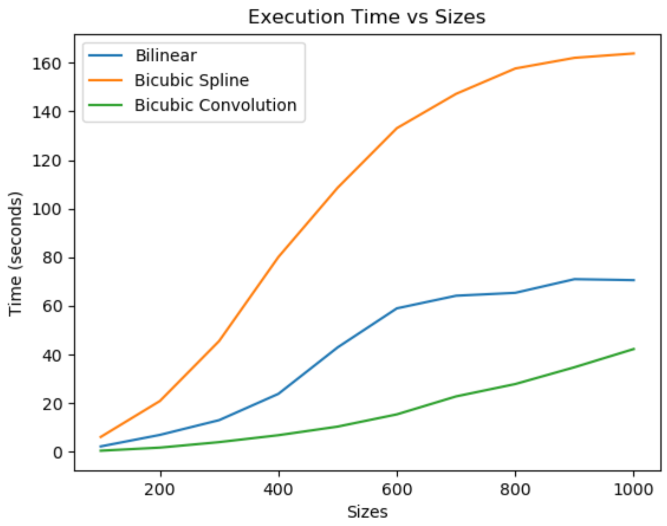
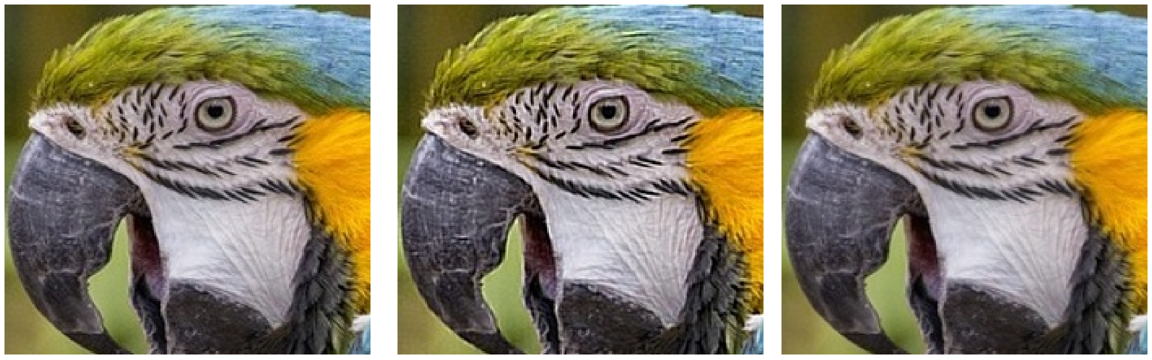


Figure 4: Resizing a 640x427 image to various sizes



(a) Bicubic Convolution (b) Bicubic Spline (c) Bilinear

Figure 5: Comparison of the three methods

5 Conclusion

The main difference between the methods discussed is that the bicubic convolution method is much faster but the accuracy of the resized image is only marginally better than resizing with bilinear interpolation. The bicubic spline method is much slower than both the convolution and bilinear methods but the resized image is far more accurate and the fine details in the original image are retained. If a fast algorithm with adequate details is required, the bicubic convolution method can be used. But if speed is not a concern and accuracy is more important, bicubic spline method can be used.

6 Acknowledgements

Special thanks to @rootpine (on GitHub) for their work on the Bicubic Interpolation Convolution formula. Your work was instrumental in the completion of this paper.

7 References

1. R Keys. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoust.*, 29(6):1153–1160, 1981.

2. Wikipedia contributors. Bicubic interpolation. https://en.wikipedia.org/wiki/Bicubic_interpolation, December 2023. Accessed: 2023-12-8.

3. Wikipedia contributors. Sobel operator. https://en.wikipedia.org/wiki/Sobel_operator, November 2023. Accessed: 2023-12-8.