[[1]](#footnote-1)

Lab 2 Image Resizing

Chengpin Luo, 11510880, SUSTech

*Abstract*— Image resizing is an interesting topic in image processing. In this lab, we will propose 4 methods to resize image—pixel replication, nearest neighborhood, bilinear interpolation and bicubic interpolation. And we would use OpenCV to implement the algorithms as well as compare the differences between them.

*Index Terms*— Image resizing, pixel replication, nearest neighborhood, interpolation

# INTRODUCTION

I

MAGE resizing, a significant and interesting topic in computer vision, usually refers to reduction and enlargement of the image. Basically, there are four kinds of algorithms in image resizing, which we will discuss later. They are: pixel replication, nearest neighborhood, bilinear interpolation and bicubic interpolation. Those four methods have different complexities while some of them are more time-consuming. In this lab, we are going to implement those algorithms using OpenCV and discuss their pros and cons.

The following sections would be constructed as: (II) Principles of image resizing algorithms; (III) Implementation and Discussion; (IV) Conclusion.

# Principle of image resizing algorithms

In this part, we are going to introduce the principle of the four image resizing algorithms.

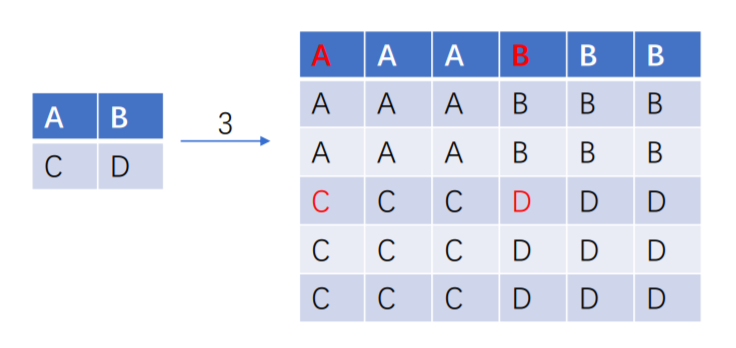
## Pixel Replication

Pixel replication is about replicating the pixel around the original pixels to the size we want. As shown in figure 1, we want to enlarge the image from 2x2 to 6x6 so we just replicate each pixel in the original image 8 times. Assume that the original size is w x h and the modified size is w’ x h’, what if w’/w is not an integer? Apparently, we need to take approximation. Assume we want the image to enlarge by 1.5 times, the pixel in the enlarged image is (x, y) and the mapped pixel in the original image is at (x’, y’). Then we have:

x’=floor(x/1.5),

y’=floor(y/1.5)

Then we map each pixel in the enlarged image to a pixel value in the original image and in this way we would get the output image.



**Fig. 1.** Pixel Replication

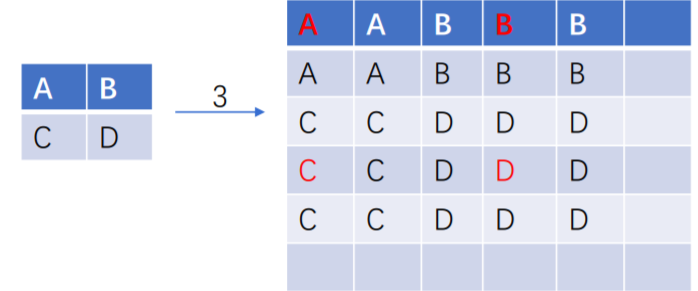
## Nearest Neighborhood

Nearest Neighborhood method is a little bit similar to pixel replication. As shown in figure 2, we map the pixel value in the resized image to the nearest pixel that is in the original image. To be more concise, we have:

x’=round(x/1.5),

y’=round(y/1.5).

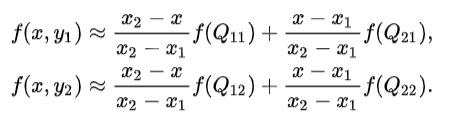
where (x, y) is the pixel value in the resized image, (x’, y’) is the mapping value in the original image and 1.5 represents the enlargement.



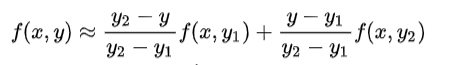
**Fig. 2.** Nearest Neighborhood

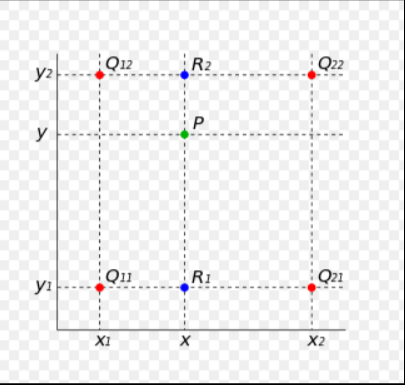
## Bilinear interpolation

Bilinear interpolation is an extension of linear interpolation. As shown in figure 3, we want to interpolate a point P surrounded by the nearset 4 pixels. Then we firstly do the linear interpolation along the columns:



Then we do the linear interpolation along the rows and get the pixel value we want to assign to the point P (f(x, y)):



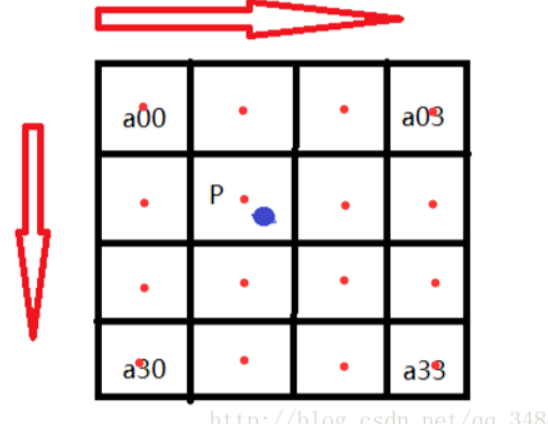


**Fig. 4.** Bilinear Interpolation

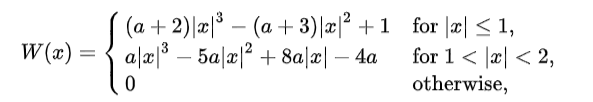
## Bicubic Interpolation

Bicubic interpolation is an extension of cubic interpolation for interpolating points in a two-dimensional grid. Different from bilinear interpolation which uses 4 pixels for interpolation, bicubic interpolation uses 16. That is, the interpolated point, f(x, y), could be represented as:

Here, we would perform the interpolation in a little bit tricky way—Bicubic Convolution Algorithm. Firstly, we map the pixels in the resized image to the original image, as shown in figure 5. In figure 5, the blue point P is the interpolated point, with 16 pixels surrounded.



**Fig. 5**. Bicubic Interpolation

And we then do the convolution using the 1-D kernel W(x) along both two directions. Kernel W(x) is represented as: 

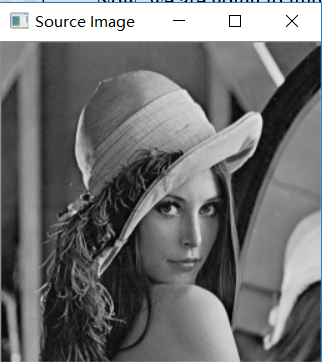
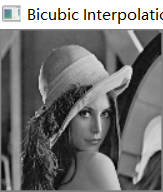
where a is set as -0.5 in this lab and x is the distance between P and the surrounded points. And W(x) represents the weights of the 16 points. For example, we suppose that P is (x+u, y+v), then for a00, the weight is (1+u,1+v). Then we can perform the convolution, that is, multiply the weight with the pixel value and add them together to get the pixel value of the interpolated point. The interpolated point could be formulated as:

where W(i) is the weight along x axis and W(j) is the weight along y axis and P(X,Y) is the interpolated value.

# Implementation and discussion

Now, we are going to implement the image resizing algorithms using OpenCV.

Firstly, we did alternative line reduction using bicubic algorithms—reduce the image to half the size, as shown in figure 6.

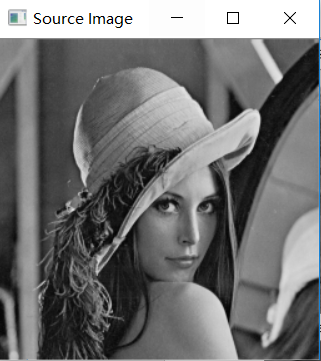
 

**Fig. 6.** Alternative Line Reduction (Left: Source Image, Right: Resized Image)

And then follows the fractional linear reduction using bilinear interpolation to reduce the image to 0.75 of its size, as shown in figure 7.

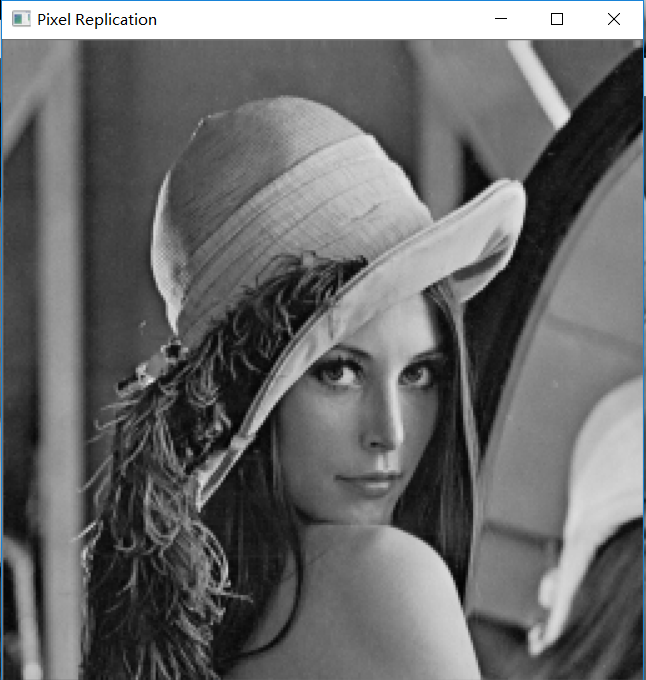
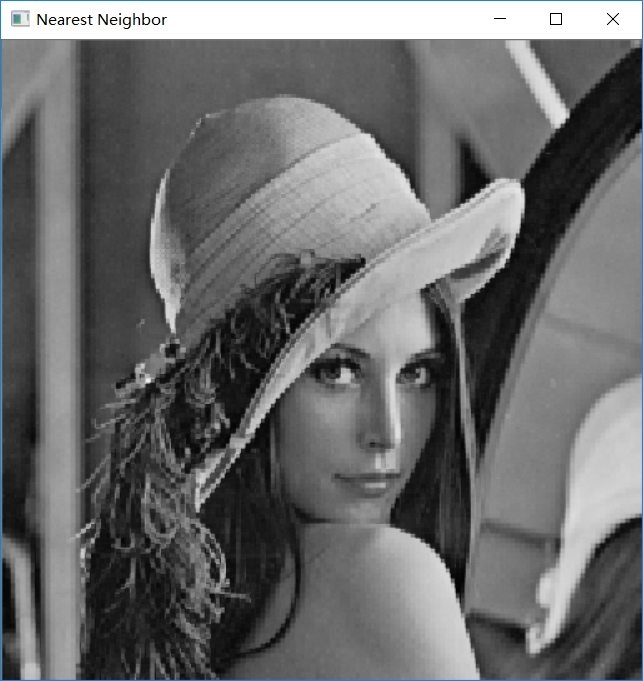
 

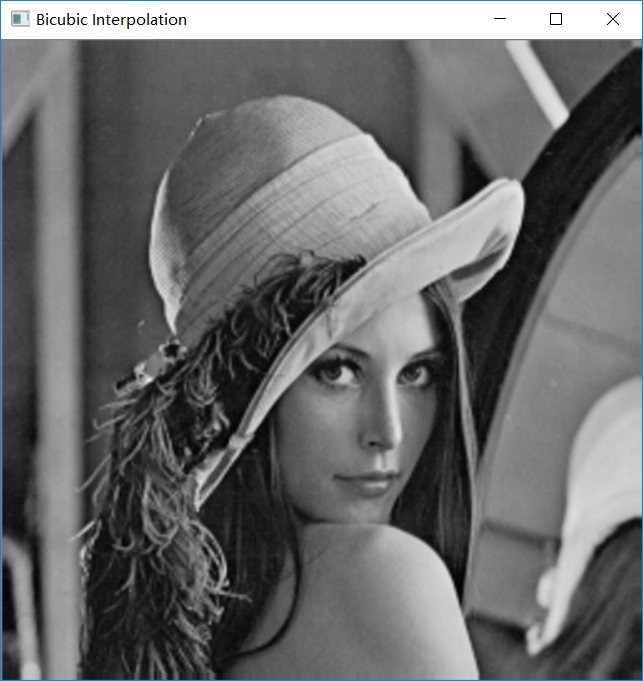
 

**Fig. 7.** Fractional Linear Reduction (Left: Source Image, Right: Resized Image)

From figure 6 and figure 7, we can see that the algorithms perform well in image reduction, no matter the reduction number is an integer or a fraction.

Then we implemented the image enlargement. The size of the images twice using pixel replication, nearest neighborhood, bilinear interpolation and bicubic interpolation respectively, shown in figure 8.

**Fig. 9.** Image Enlargement(Lena, from left to right, from up to down: pixel replication, nearest neighborhood, bilinear interpolation, bicubic interpolation)

Another example is given in figure 10.    

**Fig. 10.** Image Enlargement(Goldhill, from left to right, from up to down: pixel replication, nearest neighborhood, bilinear interpolation, bicubic interpolation)

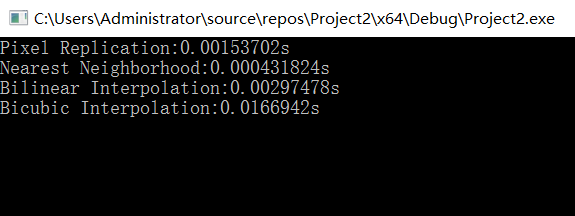
From figure 9 and figure 10, we could see clearly that using bicubic interpolation allows us to get the smoothest enlarged image. And bilinear interpolation is a bit less smoother than bicubic interpolation. Using pixel replication or nearest neighborhood is much worse than the other two interpolation methods. Especially when using nearest neighborhood, you could see some mosaic-like pixel blocks.

Besides, figure 11 shows two examples of fractal linear expansion using bilinear interpolation.

**Fig. 11.** Fractal linear expansion (size: x1.5, left: goldhill, right: lena)

Now, we are going to discuss the running time of each algorithm. Set the enlargement as x1.5 and then we got the running time of each algorithm, as shown in figure 12. We can see that bicubic interpolation is the most time-consuming while nearest neighborhood costs the least time.



**Fig. 12.** Running time

# Conclusion

In this lab, we implemented four image resizing algorithms—pixel replication, nearest neighborhood, bilinear interpolation and bicubic interpolation. We reduced and enlarged the images using those algorithms. And we found that the bicubic interpolation has the best effect while the nearest neighborhood has the poorest performance. On the other side, bicubic interpolation costs the time most while nearest neighborhood is the fastest. Therefore, when resizing images, we need to take this trade-off into consideration.

1. Chengpin Luo is with Electrical and Electronic Engineering Department, Southern University of Science and Technology, 518055, CHINA (Student Number: 11510880, e-mail: 11510880@mail.sustc.edu.cn). [↑](#footnote-ref-1)