Exercise 1: Full Adder Simulation

Jincheng Yu

March 7, 2023

1 Introduction

In the daily life, changing the size of picture is a very normal and demanded function for phones or cameras. But if the size of picture varies in proportion directly, the picture will cause distortion which can be easily recognized by eyes especially for the enlarging conduct. Therefore, some methods to make the enlarged or squeezed picture clearer and smooth are necessary. Image interpolation is one of most usual method of resizing image. It doesn't increase more image information but increase the pixels according to the known image information.

In this experiment there are three algorithm to solve the problem: nearest neighbor interpolation, bilinear interpolation and bicubic interpolation. The nearset interpolation is the algorithm that selects the nearest point and doesn't consider the other points nearby. The bilinear interpolation consider the nearest 4 points and use three times linear transform to get the destination point. The bicubic interpolation use 16 points to solve the differential equation to calculate the destination point. We will use these methods to enlarge and shrunk the same picture and compare their performance to analyse their difference.

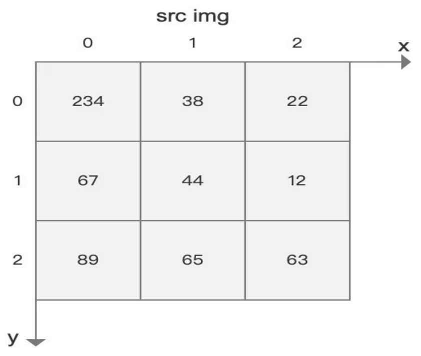
2 Nearest Interpolation

The nearest interpolation algorithm choose the value of the nearest point in origin figure as the value of resized image point. We define dst_width and dst_height are the width and height of the destination image, src_width and src_height are the width and height of the source image, dstX and dstY are the coordinate of the destination point, srcX and srcY are the coordinate of the source point. The destination point and source point have the following relation:

$$srcX = dstX * (src_width/dst_width)$$

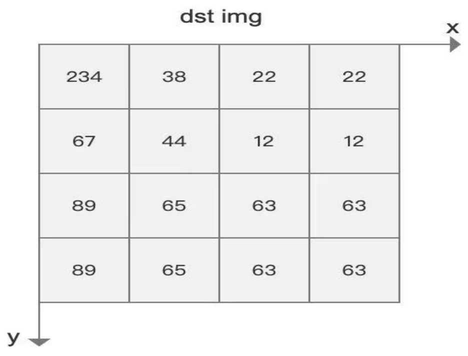
$$srcY = dstY * (src_height/dst_height)$$

In the Fig.1. we can see an example of nearest interpolation. This figure shows resize a 3*3 image to a 4*4 image and the third column is identical to the fourth column due to the nearest interpolation algorithm.



src img
0
1
2
x
y
234
38
22
67
44
12
89
65
63

(a) Source image



dst img
x
y
234
38
22
22
67
44
12
12
89
65
63
63
89
65
63
63

(b) Destination image

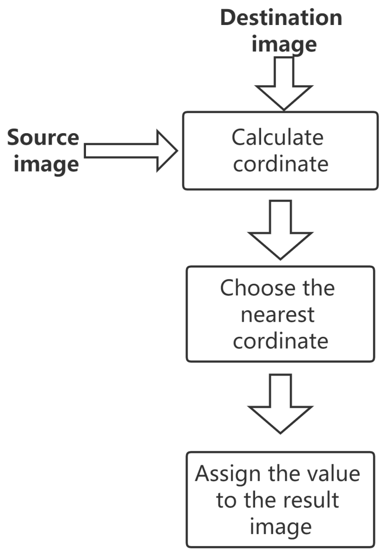
Figure 1: The process of nearest interpolation

From the figures and equations we can discover the defect of this method of choosing point. When the dstX = 0, the srcX = 0 and when dst = dst_width - 1 the srcX is less than src_width - 1, which means the $srcX \in [0, dst\_width - 1)$, $srcY \in [0, dst\_height - 1)$ and the selected points deviated to left and up in general. Therefore, it's significant to shift the geometric center of selected points to the center of entire figure.
The improved equation:

$$srcY = (dstY + 0.5) * (src_height/dst_height) - 0.5$$

$$srcX = (dstX + 0.5) * (src_width/dst_width) - 0.5$$

The geometric center of source image is $(\frac{src\_width-1}{2}, \frac{src\_height-1}{2})$ and the geometric center of destination image is $(\frac{dst\_width-1}{2}, \frac{dst\_height-1}{2})$. And we can find that srcX=$\frac{src\_width-1}{2}$ when dstX=$\frac{dst\_width-1}{2}$ in above equations which means the geometric center of destination image correspond the the geometric center of source image through above transform.



Destination image
Source image
Calculate cordinate
Choose the nearest cordinate
Assign the value to the result image

Figure 2: Flow-chart: nearest interpolation

3 Bilinear Interpolation

Bilinear interpolation is an algorithm that considers four points in source image near the point transformed from the destination image. The algorithm derived from linear interpolation.

The principle of linear interpolation shows above. There are two points (x1,y1),(x2,y2) and one interpolation point (x,y) in the middle. They have the relation that $\frac{x-x1}{y-y1} = \frac{x2-x}{y2-y}$ so we can derive the equation of coordinates that $x = \frac{x-x1}{x2-x1} * x2 + \frac{x2-x}{x2-x1} * x1$, $y = \frac{y-y1}{y2-y1} * y2 + \frac{y2-y}{y2-y1} * y1$.

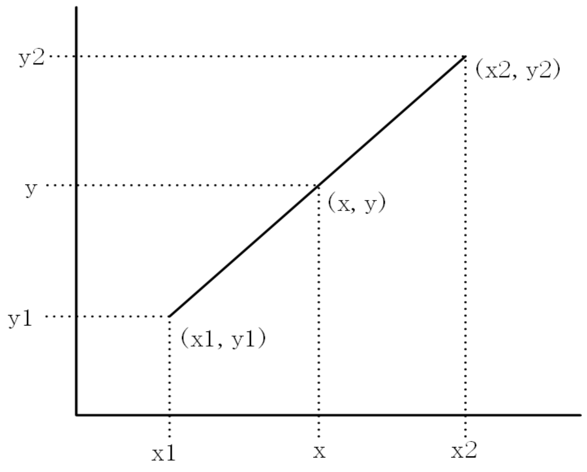


Figure 3: Linear interpolation

The point transform between source image and destination image is identical with the nearest interpolation algorithm. Bilinear interpolation calculates four points near the transformed point.

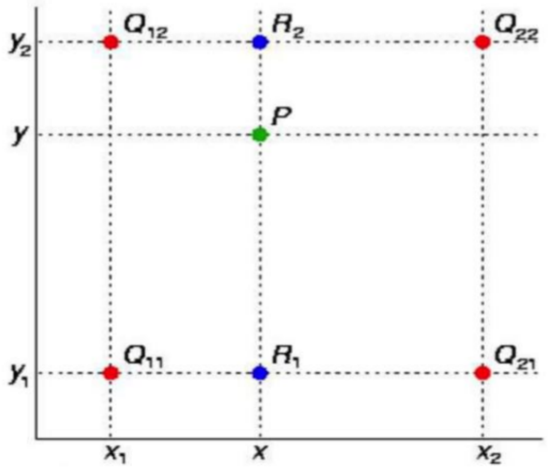
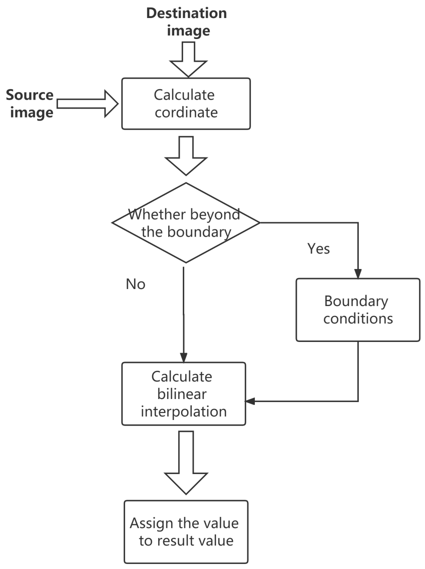


Figure 4: The bilinear interpolation

As shown in the figure above, Q12, Q22, Q11 and Q21 are known, but the points to be interpolated are P points. First, in the x-axis direction, R1 and R2 are interpolated, and then P points are interpolated according to R1 and R2. The equation shows below: $f(x,y) = \frac{f(Q11)}{(x2-x1)(y2-y1)}(x2-x)(y2-y) + \frac{f(Q21)}{(x2-x1)(y2-y1)}(x-x1)(y2-y)$



Destination image
Source image
Calculate cordinate
Whether beyond the boundary
Yes
No
Boundary conditions
Calculate bilinear interpolation
Assign the value to result value

Figure 5: Flow-chart: linear interpolation

4 Bicubic Interpolation

In the process of bicubic interpolation, we first find out the corresponding pixel (x, y) of pixel (X, Y) in source image, and then use the 16 pixel points closest to the pixel (x, y) in source image as the parameter to calculate the pixel value at destination image (X, Y), and use the BiCubic basis function to calculate the weight of 16 pixel points. Through the Bicubic basis function we can

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{for } |x| \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{for } 1 < |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

get the weight of all 16 points. And the value of the 16 points can be calculate by the following function:

$$f(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} f(x_i, y_j) W(x - x_i) W(y - y_j)$$

5 Experiment

In the experiment nearest neighbor interpolation and bilinear interpolation algorithms are manually written in Python and bicubic interpolation algorithm is directly used the function "interp2d" from packet "scipy".

The algorithm of nearest neighbor interpolation shows below:

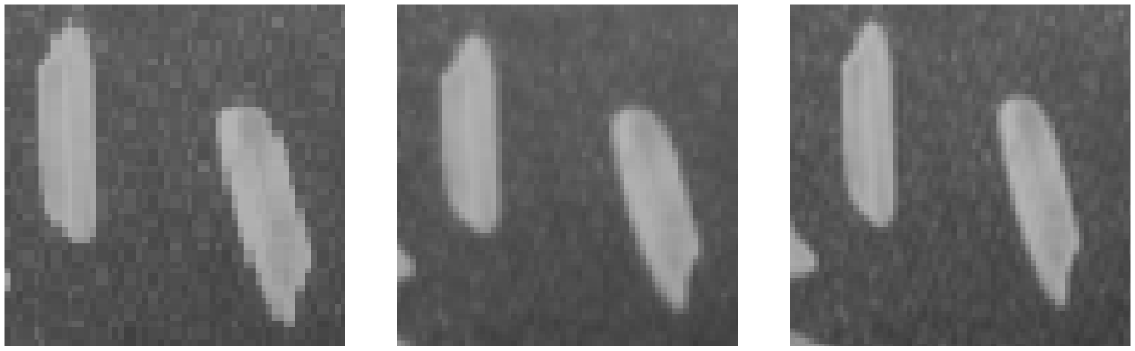
```
  Data: source image Is and dimension of desired image dim
  Result: destination image Id
1 Convert image to matrix;
2 for $i \leftarrow 0$ to $dim[0] - 1$ do
3     for $j \leftarrow 0$ to $dim[1] - 1$ do
4         Calculate the nearest point Im[x,y];
5         Id[i,j]= Im[x,y];
6     end
7 end
8 Convert matrix to image;
```

The algorithm of bilinear interpolation shows below:

```
   Data: source image Is and dimension of desired image dim
   Result: destination image Id
 1 Convert image to matrix;
 2 for $i \leftarrow 0$ to $dim[0] - 1$ do
 3     for $j \leftarrow 0$ to $dim[1] - 1$ do
 4         Calculate the nearest point Im[x,y];
 5         if $Im[x, y]$ beyond the boundary then
 6             Let $Im[x, y]$ be the points on the boundary
 7         end
 8         Choose four points around Im[x,y];
 9         Calculate Id[i,j] by bilinear interpolation;
10     end
11 end
12 Convert matrix to image;
```

In this report, the size of zooming image is 333x333 and that of shrank image is 179x179 with 256x256 original image. Thus, the enlarging ratio is 1.3 and the shrank ratio is 0.7. The result of bicubic interpolation and bilinear interpolation can hardly be distinguished and they both have better performance than nearest neighbour interpolation. To observe the difference clear, I selected the same area in the interpolated images for comparison. By enlarging the images their difference are more easier to observe.

From the figures we can see that the figure of nearest neighbour interpolation has clear mosaics



(a) Nearest neighbour interpolation (b) Bilinear interpolation (c) Bicubic interpolation

Figure 6: Three interpolation methods on the same part after enlarging interpolation

and the figures of bilinear interpolation and bicubic interpolation are more smoother. Comparing the bilinear interpolation and bicubic interpolation we can see the figure of bilinear interpolation is more blurred and the figure of bicubic interpolation is more clear. From the perspective of time, the time for these algorithm are shown in the table.1. The time cost: Bilinear > Nearest neighbour > Bicubic. The result does not conform the expectation because the number of calculation steps of bicubic interpolation is the most in the three method. The reason for the results may because the function ”interp2d” has optimized the algorithm to a low time complexity. But we still can find that the time cost of bilinear algorithm is much more than it of nearest neighbour algorithm, although the interpolated image of bilinear algorithm is better than it of nearest algorithm. Considering the principle, we can infer the time cost for bicubic algorithm is more than the others when all of them has been optimized, but the image quality of bicubic algorithm is better than the others.

Table 1: The time cost for interpolation process

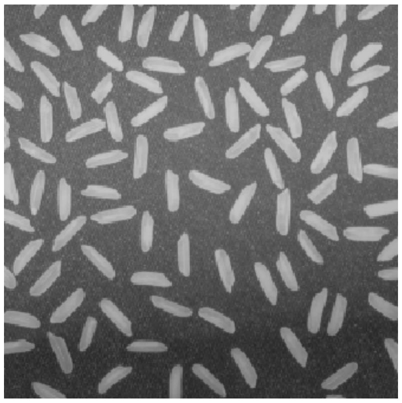
Nearest Neighbour	Bilinear	Bicubic
0.0496s	0.682s	0.006s

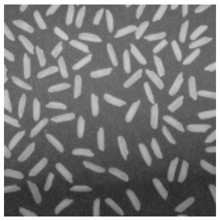
6 Conclusion

Comparing the three interpolation algorithms, we can see the quality of interpolated image: bicubic>bilinear>nearest. The time cost of the interpolation process: nearest<bilinear<bicubic. The result shows the trade-off between performance and time cost for algorithms. Each algorithm has its own uniqueness so it is best to choose the appropriate algorithm accoriding to the specific situation.

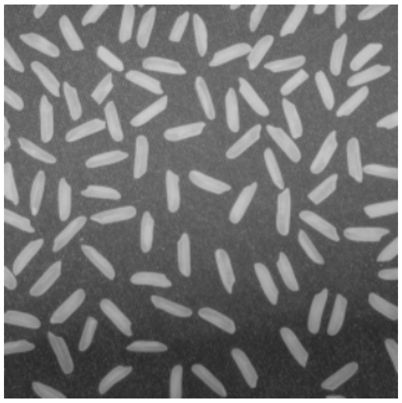
(a) Shrunk nearest intepolation

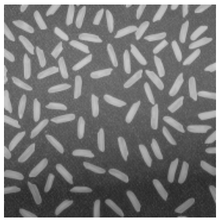
(b) Origin image


(c) Enlarged nearest intepolation


(d) Shrunk bilinear intepolation

(e) Origin image


(f) Enlarged bilinear intepolation


(g) Shrunk bicubic intepolation

(h) Origin image

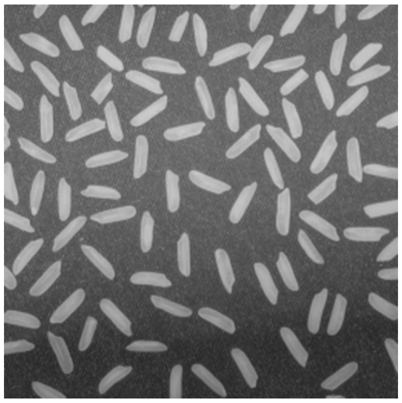

(i) Enlarged bicubic intepolation

Figure 7: Interpolated images by three algorithms