Project name: Zooming and Shrinking Images by Pixel Replication and Bilinear Interpolation
Project number: PROJECT 02-04 & 02-05
Course name: Introduction to image processing, HSU, CHIU-TING
Student's name: LU, ZHOU-TAO (ID: x1052078)
Date due: 6th Oct.
Date composed and finished: 2nd Oct; 5th Oct.

Abstract:

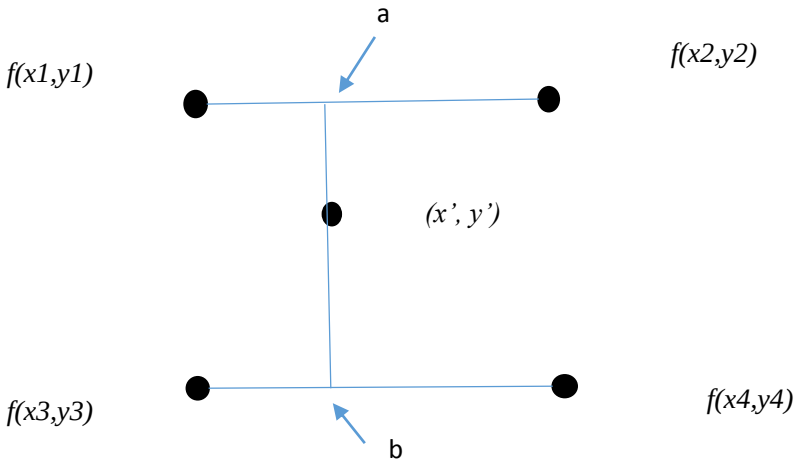
In this project, two totally different image-resizing algorithms, Pixel Replication and Bilinear Interpolation are compared to each other in terms of the performance on resizing. The algorithm, Pixel Replication, is to assign every pixel in the new location by replicating the value of its nearest neighbor, while the other algorithm, Bilinear Interpolation, is to estimate the pixels' value in consideration of its four nearest neighbors. And I select a simple way to assess the performance, which is to reckon the recovery rate of the back-zoomed image.

Implementation:

1. Zooming and Shrinking Images by Pixel Replication
   The for-loop procedure can be separated into two parts, finding the correspond location in the original image of every pixel, and getting the nearest neighbor.
   1) Find the corresponding location.
      Let M denote the number of rows in the original image, and N the number of columns in the original image with λ denoting the scaling factor.
      Then for any coordinate in the resulting image (*x*,*y*), the correspond location (*x'*,*y'*), where *x'*= $\frac{x-1}{round(\lambda M)-1} * (M-1)+1$, y'= $\frac{y-1}{round(\lambda N)-1} * (N-1)+1$. (Note: round(.) stands for the rounding of the number).
   2) Find the nearest neighbor.
      The way to identify the neighbors here is intuitively by *Euclidean* distance. And the best way to identify a specific location's nearest neighbor pixel is just to round the correspond location value, which is to say the neighbor's coordinate is (*round*(*x'*), *round*(*y'*)).

   Then, the rest work is to assign the value by copying the neighbor's intensity.

2. Zooming and Shrinking Images by Bilinear Interpolation
   The only difference is that it needs to find 4 neighbors to estimate the intensity of pixels in resized image by two-dimensional linear estimation.
   The main deduction process is just as following:



Easily, we can get $a = \frac{f(x2,y2)-f(x1,y1)}{y2-y1} * y' + \frac{f(x2,y2)-f(x1,y1)}{y2-y1} * y1 +$ $f(x1,y1)' = a_1+a_2y'$, $b = \frac{f(x4,y4)-f(x3,y3)}{y4-y3} * y' + \frac{f(x4,y4)-f(x3,y3)}{y4-y3} * y3 +$ $f(x3,y3) = b_1+b_2y'$. And $g(x, y) = f(x', y') = \frac{b-a}{x3-x1} * (x' - x1) + a =$ $c_1+c_2x'+c_3y'+c_4x'y'$.

For we've found the 4 neighbors (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) then

$$\begin{cases} f(x1,y1) = c1 + c2 * x1 + c3 * y1 + c4 * x1y1 \\ f(x2,y2) = c1 + c2 * x2 + c3 * y2 + c4 * x2y2 \\ f(x3,y3) = c1 + c2 * x3 + c3 * y3 + c4 * x3y3 \\ f(x4,y4) = c1 + c2 * x4 + c3 * y4 + c4 * x4y4 \end{cases}$$

which is

$$\begin{bmatrix} 1 & x1 & y1 & x1y1 \\ 1 & x2 & y2 & x2y2 \\ 1 & x3 & y3 & x3y3 \\ 1 & x4 & y4 & x4y4 \end{bmatrix} \begin{bmatrix} c1 \\ c2 \\ c3 \\ c4 \end{bmatrix} = \begin{bmatrix} f(x1,y1) \\ f(x2,y2) \\ f(x3,y3) \\ f(x4,y4) \end{bmatrix}$$

With the help of *Matlab*, we can get $\begin{bmatrix} c1 \\ c2 \\ c3 \\ c4 \end{bmatrix}$. Then

$$g(x,y) = \begin{bmatrix} 1 & x' & y' & x'y' \end{bmatrix} \begin{bmatrix} c1 \\ c2 \\ c3 \\ c4 \end{bmatrix}.$$

But, there are some special cases where the computation process cannot be applied directly. That is, when the location is on the certain row or column of the original image, only two neighbors are needed (mono-linear) or even when the location is exactly on the pixel of the original image, replication is enough (replication).

Thus, the for-loop procedure can be revised as following:

1) Find the corresponding location (*x'*, *y'*). (*same as above*)
2) If the location is exactly on the certain pixel, then copy the intensity. Go to step1) for next pixel.
3) If the location is exactly on the certain row or column, but not on a pixel, then bilinear interpolation is degraded to be linear.
   a) Find the nearest neighbors. There are 2 neighbors used to estimate. Similarly, we use floor(.) and ceil(.) to get the neighbors' location, where '.' is *x'* and *y'* for the location is on the certain column and row respectively.
      To simplify, we use the case the location is in the certain column to stated, with the other case staying similar.
   b) Estimate the intensity of target location.
      It is easy to get the equation $\frac{x'-x1}{f(x')-f(x1)} = \frac{x2-x1}{f(x2)-f(x1)}$, then we can get the value f(x') = $\frac{f(x2)-f(x1)}{x2-x1} * x' + \frac{f(x2)-f(x1)}{x2-x1} * x1 + f(x1)$.
      Assign the intensity. Go to step1) for next pixel.
4) If the location is not in the certain row or column, then use bilinear interpolation described above to identify.
   Again, assign the intensity. Go to step1) for next pixel.

Result:

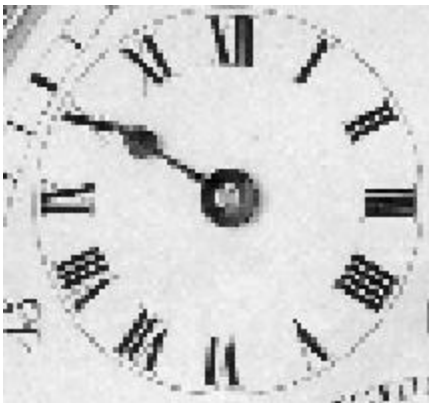
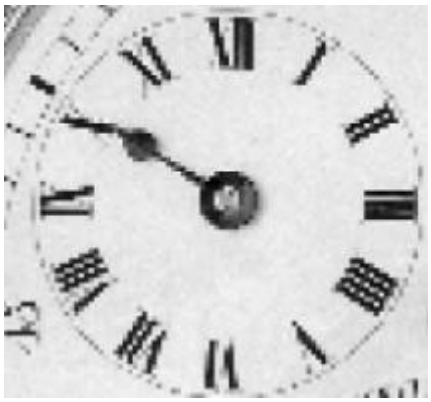
a
b c
d e

FIGURE 1.1 (a) The original image that is required. (b) The image that is shrunk 1/10 by nearest pixel replication. (c) The image that is zoomed back to original resolution by nearest pixel replication. (d) The image that is shrunk to 100dpi by bilinear interpolation. (e) The image that is zoomed back to 1250dpi from image(d) by bilinear interpolation

Discussion of Result:

Let’s just assess the performance by subjective observation.

From the images attached above, it seems similar between (b) and (c) or (d) and (e) in spite of the fact that the number of pixels in the image (b)(or (d)) is just 1/100 that of image (c)(or (e)). But what if the images are enlarged to a same view size? Take the (b) and (c) to compare.

(Note: only parts of enlarged images are posted. The left one is from (b), while the last one is from (c))

Thus, the difference can be clearly observed. There are clear pixel-like squares seen on the left one, while on the right one, it can roughly be seen but not that clear. That is to say, the reason why we cannot distinguish the images (b) and (c) should turn to the limit of our human visual system. That is we cannot tell between two dots standing so closely that it deceives us that there is only one dot. And when the viewed size is enlarged, the size of each pixel is enlarged with it. Progressively, the pixel of the image with fewer pixels first reaches the bound of our visual limit. Thus, the difference is detected.

To compare the performance of 2 algorithms, we need to assess the two zoomed-back image (c) and (e).

At the first glance, it seems that the dark signs in the image (c) shrunk and zoomed by replication is clearer than that in the image (e), which may be described as the higher **contrast**. From the algorithm, we can infer the possible cause. The replication algorithm is to copy the intensity, making the point located in a specific interval $[x, x+1]$ has the same intensity as x location if the point is in the former half interval, and same as $x+1$ location if in the latter half interval. Thus, while zooming back, a sudden decline of intensity will occur in each integer interval, making the signs highlight from the background. However, when the bilinear algorithm is applied, there won’t be a sudden decline in intensity, but the intensity will steadily increase or decrease between each integral intervals.

And surprisingly, that may be the reason why the image zoomed back by bilinear interpolation always **blur in the border** of those dark sign. When we first shrink the image, it is very likely to sample points on two sides of the border, one has high intensity and the other low. After that, when zooming back using bilinear interpolation, the intensity decrease steadily from the high-intensity point to low-intensity point, making the border blur.

So, it reminds me of another way to identify the intensity. Maybe we shouldn't just assume that the actual change of intensity between 2 pixels are linear, instead there should be a **distribution** of the intensity that when the point is more close to the integer point with high intensity, the intensity should be higher, while it should be lower when the point is closer to the integer point with low intensity.

To compare 2 algorithms quantitatively, I define a measurement **recovery rate** as following:

$$REC = 1 - \frac{1}{256} * \frac{1}{n} \sum_{i=1}^{n} |f(i) - g(i)|$$

, where n is the number of pixels in the image.
Of course, the scaling factor here used in both algorithms should be the same.
The instructions are as following :

```
>> delta_nearest = abs(original_image-zoomed_image_nearest);
>> delta_bilinear = abs(original_image-zoomed_image_bilinear);
>> sum_nearest = sum(delta_nearest(:))

sum_nearest =

    34209384

>> sum_bilinear = sum(delta_bilinear(:))

sum_bilinear =

    29959286

>> REC_nearest = 1-sum_nearest/(256*3692*2812)

REC_nearest =

    0.9871

>> REC_bilinear = 1-sum_bilinear/(256*3692*2812)

REC_bilinear =

    0.9887
```

From the response, we can see that the recovery rate of bilinear interpolation is a little bit higher than that of nearest interpolation, which means the algorithm of blinear interpolation can be a little bit better to retain the intensity.