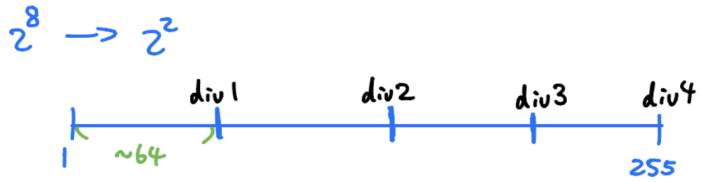
DIP Lab 1

105042015 沈冠妤 外語20

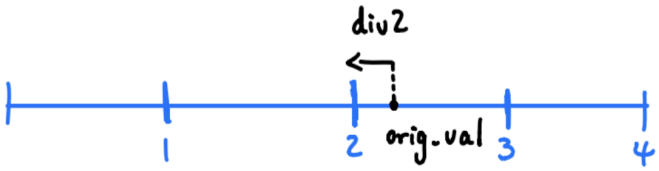
1. Proj02-02 - Reducing the Number of Intensity Levels in an Image (30%)

- **Explanation:**

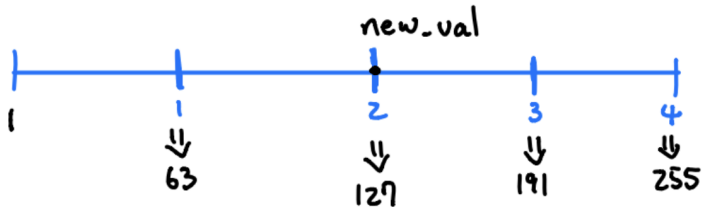
  Divide number 1~255 into given number of divisions, such as 2^7, 2^6, ..., 2^0, to get the range of each division.



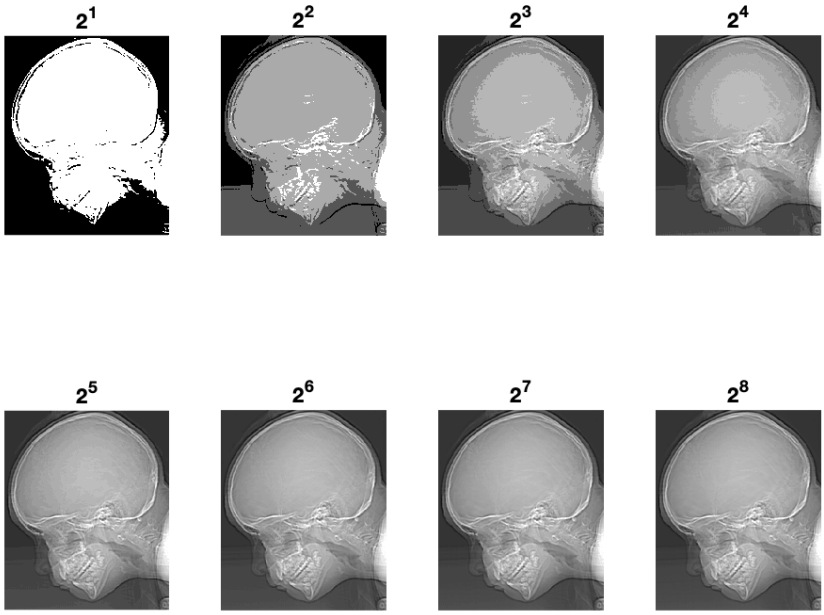
  Let original intensity divided by range of a division. Pick the nearest floor integer to determine the scaled-down value.



  Map the scaled-down value to 1~255 to get the new value.



- **Result:**



- **Comparison:**

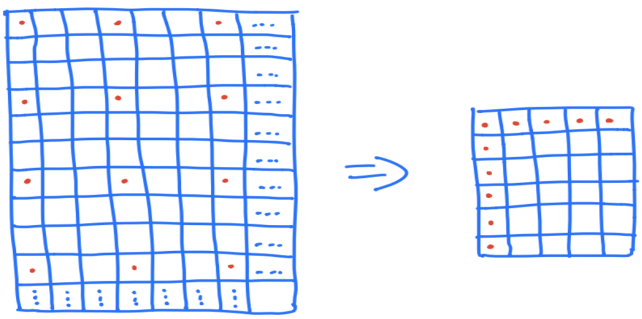
  Images with higher quantization results in a more authetic and blur image, while images with lower quantization appears more sharp and distinct.

2. Proj02-03 – Zooming and Shrinking Images by Pixel Replication (30%)

- **Explanation:**

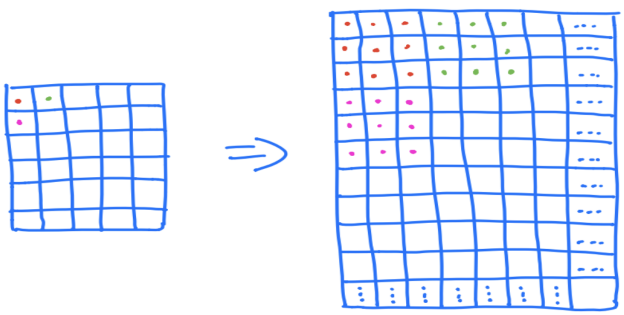
  To shrink image with nearest-neighbor algorithm, each **point(i, j)** of output image maps to **point((i-1)*fac+1, (j-1)*fac+1)** of input image.

  - new_img(1,1) >> orig_img(1,1)
  - new_img(1,2) >> orig_img(1,11)
  - ...
  - new_img(2,2) >> orig_img(11,11)



To zoom image, for each p(i,j) of output image, divide its coordinates by fac and round to the nearest integer, then plus 1 for matlab's indexing rule.

If exceeds boundary, given the closest boundary's value.



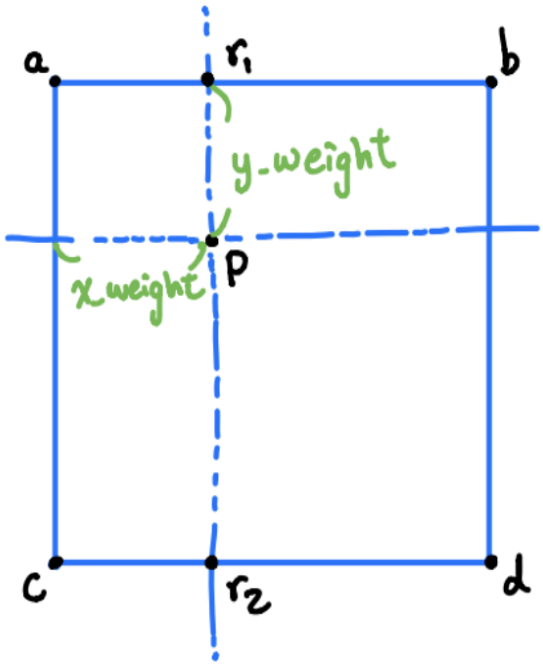
- **Result:**

- **Comparison:**

  Image after shrinking and zooming appears more sharp and distinct. Some data are lost during shrinking, and nearest-neighbor method merely repeats the existing values, so no new values are created during the zooming process.

3. Proj02-04 – Zooming and Shrinking Images by Bilinear Interpolation (40%)

- **Explanation:**



P is the desired point to interpolate.

To implement bilinear interpolation, first find the ratio between input size and output size for both x and y dimension.

For each pixel on the output image, determine the 4 corresponding points on the input image by multiplying the ratio we retrieved before. Mind the fix() and ceil() for each point, and plus 1 for matlab's indexing rule.

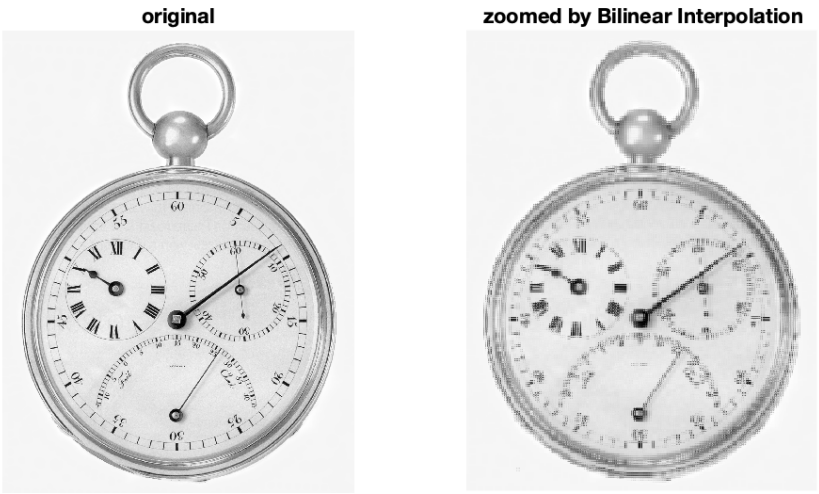
Map P on input image by multiplying the ratios. Get x_weight by subtracting P's x coordinate with a's coordinate. Add abs() to work both for shrinking and zooming. Same for calculating y_weight.

To determine value of r_1, let a*(1-x_weight) + b*x_weight.

To determine value of r_2, let c*(1-x_weight) + d*x_weight.

To determine value of P, let r1*(1-y_weight) + r2*y_weight.

- **Result:**



- **Comparison:**

  Image after shrinking and bilinear interpolation zooming appears more blur than the original image. Comparing to replication method, result of bilinear interpolation seems more natural, less sharp.