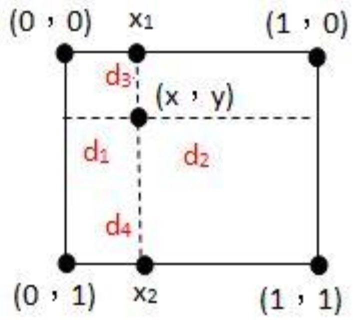
Scaling

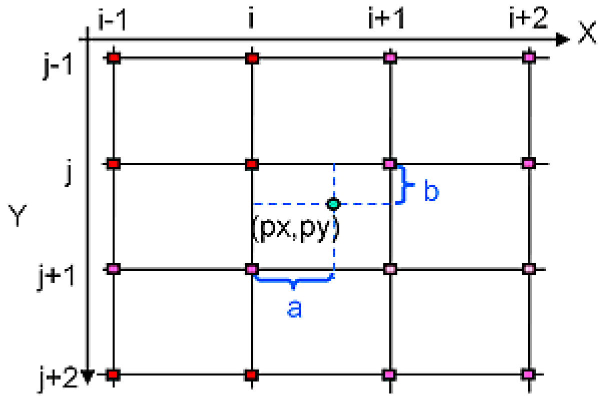
Discussion on bilinear interpolation



Bilinear interpolation uses the **nearest four points** to predict the value of the target (x_t, y_t). Define f(x, y) means the value at that point, i.e. f(0, 0) = 0, f(1, 1) = 255. In bilinear interpolation, we compute the linear equation between x and f(x, y) for every row first. Therefore, we need to get these equations. $\mathrm{f}(\mathrm{x}, 0) = a_0 x + b_0$ and $\mathrm{f}(\mathrm{x}, 1) = a_1 x + b_1$

We could get a_0 and b_0 by f(0, 0) and f(1, 0) and get a_1 and b_1 by f(0, 1) and f(1, 1). After the computation, we can get two equation f(x, 0) and f(x, 1). We put x_t (x value of target) to get two point: f(x_t, 0) at point x_1 and f(x_t, 1) at point x_2. Again, we could compute the equation: $\mathrm{f}(x_t, \mathrm{y}) = a_3 y + b_3$ by f(x_t, 0) and f(x_t, 1). Finally, we have f(x_t, y) and just put y_t in it to get f(x_t, y_t).

Bicubic interpolation



Bicubic interpolation uses the same concept of bilinear interpolation, but bicubic interpolation uses the most **nearest 16 points** to predict the value of target F(x', y'). Using the concept of bilinear, I compute the following equation for each row first.

$\mathrm{F}(\mathrm{x}, \mathrm{j}-1) = a_0 x^3 + b_0 x^2 + c_0 x + d_0$

$\mathrm{F}(\mathrm{x}, \mathrm{j}) = a_1 x^3 + b_1 x^2 + c_1 x + d_1$

$\mathrm{F}(\mathrm{x}, \mathrm{j}+1) = a_2 x^3 + b_2 x^2 + c_2 x + d_2$

$\mathrm{F}(\mathrm{x}, \mathrm{j}+2) = a_3 x^3 + b_3 x^2 + c_3 x + d_3$

After computing these equations, we could put x' in and get 4 points: F(x', j-1), F(x', j), F(x', j+1) and F(x', j+2). Now, we could compute the equation:

$$\mathrm{F}(\mathrm{x}', \mathrm{y}) = a_4 x^3 + b_4 x^2 + c_4 x + d_4$$

Put y' into F(x', y) and get F(x', y') at the end.

Some method I use in program

There is an issue that the result after read bitmap array is **one-dimensional array**. If the file is a 24-bit BMP image, we will get an array like this : [RGBRGBRGB…], every pixel has three values. However, the previous discussions are based on two-dimensional array, so I calculate (x, y) from 1-D array by:

$$x = \ floor\left(\frac{index}{byte\ \ per\ \ pixel}\right)\ \ \mathrm{mod}\ \ width$$

$$y = floor\left(floor\left(\frac{index}{color\ \ number\ \ per\ \ pixel}\right) \div width\right)$$

$$\text{where } index \text{ is the index of current element in an array}$$

$$\text{and } color\ number\ per\ pixel \text{ is typically calculated by } \frac{bits\ \ per\ \ pixel}{8}$$

$$\text{because we use one byte to represent value of a color.}$$

After getting (x, y), we could continue to go on our algorithm above, for scaling-up task, we could map this (x, y) to original image by (x/1.5, y/1.5).

Reference

[1] Bilinear interpolation (Wiki)

[2] Bicubic interpolation (Wiki)