Discussion of Different Interpolations

Junkai Qiu

Introduction

In image geometric transformation, it usually need to involved operation like translation, rotation and zoom, etc. For the first two transformations, there is always a pixel of original image correspond to a pixel of output image. However, for the zoom, we need to add or delete some pixel in order to output a new image. Therefore, we need to introduce a technique called interpolation into image process.

Interpolation

Took RGB mode as an example, in each channel, the pixel color intensity is discrete. So the typical way is to create continuous curves upon the discrete pixel values, transplant it into the new image platform and resample for new pixel. We generally take backward-warping way, set up a table to connect the pixels of input and output images and make sure there is always a certain value for the new pixel. Otherwise, if we infer the map from input, it’s possible to find a pixel of output with no intensity value. In addition, a good interpolation algorithm should implement the smooth change between two given pixels and avoid distortion.

Nearest Neighbor Interpolation

It’s the simplest interpolation algorithm. It just directly chooses the closest pixel’s intensity value as the interpolation result. Through backward-warping, acquire target float coordinate, simply round the coordinate value and use that integer as the result pixel value.
The formula can be simply stated as:

$$\mathrm{f(i+a,j+b)} = \begin{cases} (i,j), & if(a \le 0.5, b \le 0.5) \\ (i+1,j), & if(a > 0.5, b \le 0.5) \\ (i,j+1), & if(a \le 0.5, b > 0.5) \\ (i+1,j+1), & if(a > 0.5, b > 0.5) \end{cases}$$

Without any calculation, nearest neighbor interpolation is simple and efficient, but the output often has low quality (mosaic situation).

Bilinear Interpolation

Just as its name implies, bilinear interpolation is combination of linear interpolation in two directions. Assumed the changes between (a,i) and (a+1,i), c(I,b)and(I,b+1) are linear, then we have

$$c(x,b) = c[a+1] * (x-a) + c[a] * (1+a-x);$$
$$c(x,b+1) = c[a+1][b+1] * (x-a) + c[a][b+1] * (1+a-x);$$

Now we know c(x,b) and c(x,b+1), we can continue and know:

$$c(x,y) = c(x,b+1) * (y-b) + c(x,b) * (1+b-y)$$

Or the well-known formula may be clearer:

$$f(i+u,j+v) = (1-u) * (1-v) * f(i,j) + (1-u) * v * f(i,j+1) + u * (1-v) * f(i+1,j) + u * v * f(i+1,j+1)$$

Bilinear Interpolation is anti-aliasing, but it doesn't consider the pixels gradient change. Its low pass filtering characteristic will damage high-frequency segment and make the contour vague.

Cubic Convolution Interpolation

It's upgrade of bilinear interpolation. It uses cubic polynomial to approach ideal interpolation function:

$$S(x) = \begin{Bmatrix} 1-2|x|^2+|x|^3, & 0 \le |x| < 1 \\ 4-8|x|+5|x|^2-|x|^3, & 1 \le |x| < 2 \\ 0, & |x| > 2 \end{Bmatrix}$$

Use the weighting interpolation of the near 16 pixels, the formula is

$$f(x,y) = f(i+u,j+v) = ABC,$$

$$A = \begin{bmatrix} S(1+v) \\ S(v) \\ S(1-v) \\ S(2-v) \end{bmatrix}^T \quad B = \begin{bmatrix} f(i-1,i-1) & f(i-1,j) & f(i-1,i+1) & f(i-1,j+2) \\ f(i,i-1) & f(i,j) & f(i,i+1) & f(i,j+2) \\ f(i+1,i-1) & f(i+1,j) & f(i+1,i+1) & f(i+1,j+2) \\ f(i+2,i-1) & f(i+2,j) & f(i+2,i+1) & f(i+2,j+2) \end{bmatrix}$$

$$C = \begin{bmatrix} S(1+u) \\ S(u) \\ S(1-u) \\ S(2-u) \end{bmatrix}$$

Cubic Convolution Interpolation has the best performance among algorithms list above, but it certainly need more time in calculation. It not only considers the influence of the value in neighbor pixels, but also their change rates. So it has a smoother margin and higher precision, hence lose little image quality.

Summary

These 3 algorithms are suitable for most transformation like rotation, simple linear transform, and non-linear transform and so on. Programmer should consider the time factor, quality requirement and other specific conditions to choose algorithm in order to reach the best experiment performance.