- Requirement 1. & Requirement 2.

For Bilinear and Bicubic method, I use OPENCV library and C++ in OSX system. The modules in OPENCV I used are "opencv.hpp" and "highgui.hpp", which is for basic module and image showing. Though there is a "resize" function in the OPENCV module "imgproc.hpp" which can scale the image in different methods, I choose to implement them by myself.

For Bilinear, I pick up the nearest 4 pixels from each source pixel. Then I do two times linear interpolation to get two points. Finally, use the two points to do the last time linear interpolation to get each pixel's RGB value.(as Fig 1.)

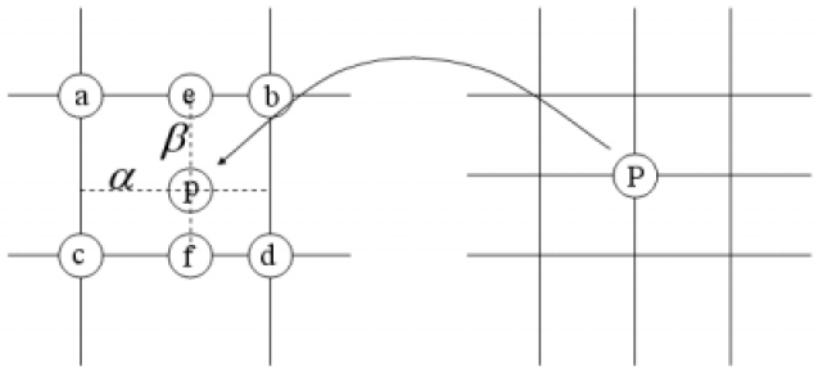


Fig 1.

For Bicubic, I pick up the nearest 16 pixels from each source pixel. Then I do four times cubic interpolation to get four points. Finally, use the four points to do the last time cubic interpolation to get each pixel's RGB value.(as Fig 2.)

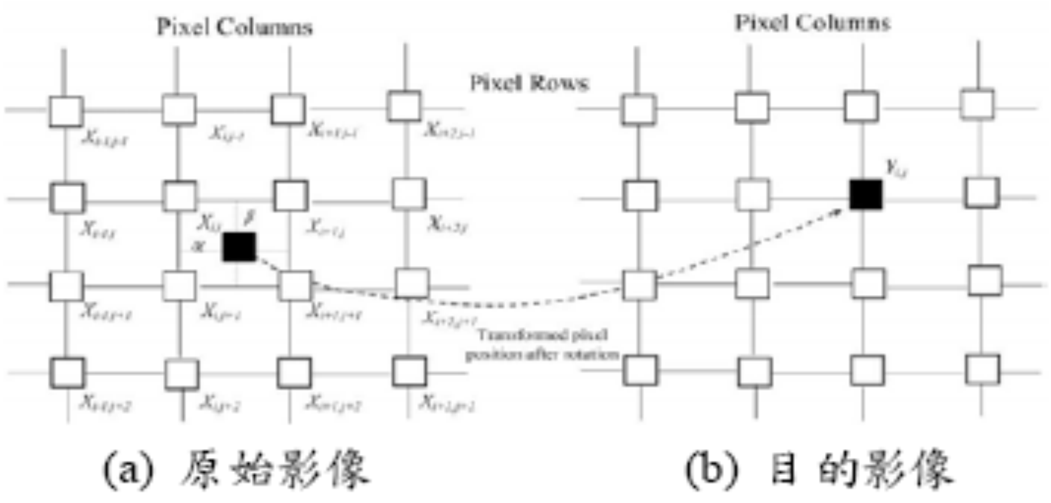


Fig 2.

The develop environment is in OSX with Xcode8 and the language is C++. The way to use this program(Practice) is that execute it with two arguments, ”input image absolute path” and “output image absolute path”.(as Fig 3.) Besides, it won’t handle any runtime error so please enter two arguments correctly.

```
// ./Pratice [input_image_absolute_path] [output_imag_absolute_path]
```

Fig 3.

- Requirement 3. & Requirement 4.

When downscaling (Render Scale over 100%) there's very little appreciable visible difference between the two. Bilinear will run ever so slightly better, but that's about it. When upscaling (Render Scale below 100%), however, the two start to become appreciably different. Bicubic is generally considered preferable as it retains some semblance of sharpness, where-as Bilinear is quite literally just blurring the results. Thus, when downscaling we can use “bilinear” because of the better performance, otherwise we can use “bicubic” because of the better detail information.

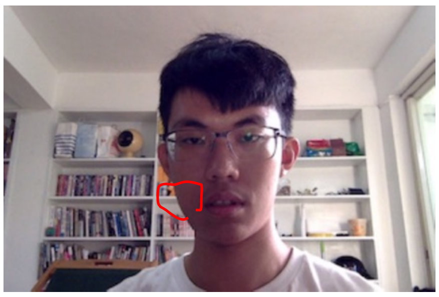


Fig 4.bilinear_resize18

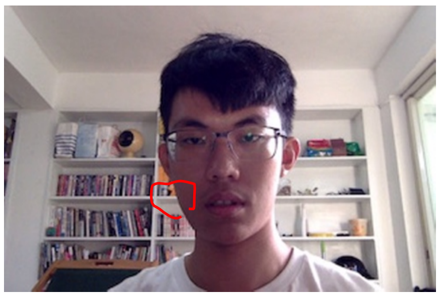


Fig5. bicubic_resize18

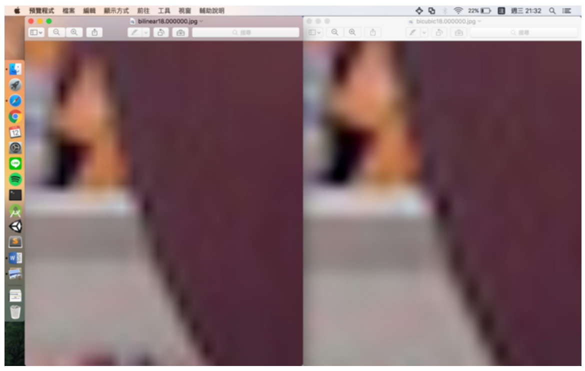
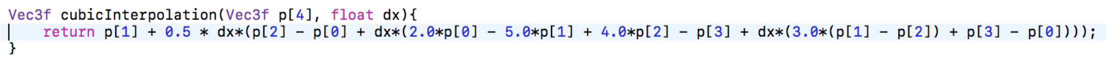


Fig 6. The comparison result

As we can see, the contour information in “bicubic” is more smooth than in “bilinear”.(as Fig 6.)

As for the method of “bicubic” interpolation, I use the formula where p[0] to p[3] are the four points and ‘x’ is the delta value.(as Fig 7.)



```
Vec3f cubicInterpolation(Vec3f p[4], float dx){
    return p[1] + 0.5 * dx*(p[2] - p[0] + dx*(2.0*p[0] - 5.0*p[1] + 4.0*p[2] - p[3] + dx*(3.0*(p[1] - p[2]) + p[3] - p[0])));
}
```

Fig 7.

Finally, I make a comparison between the complexity of bilinear and bicubic method. In Bilinear, I have three times linear interpolation each pixel, thus, the complexity of Bilinear is O(mn*3) = O(mn), where m is the row numbers and n is the column numbers. In Bicubic, I have five times cubic interpolation each pixel, thus, the complexity of Bicubic is O(mn*5) = O(mn). Though their complexity is the same, the actual cost of Bicubic is higher than Bilinear since cubic interpolation has more times of operation and interpolation than linear interpolation.

By. R05922130王瀚磊