JPEG 图像压缩

1 图像分块 8x8

- 这些小块在整个压缩过程中都是单独被处理，这一步比较简单，将输入的彩色图片分块存储即可。
- 实际代码编写的时候，是处理到某一块的时候才会将其分离出来进行处理。

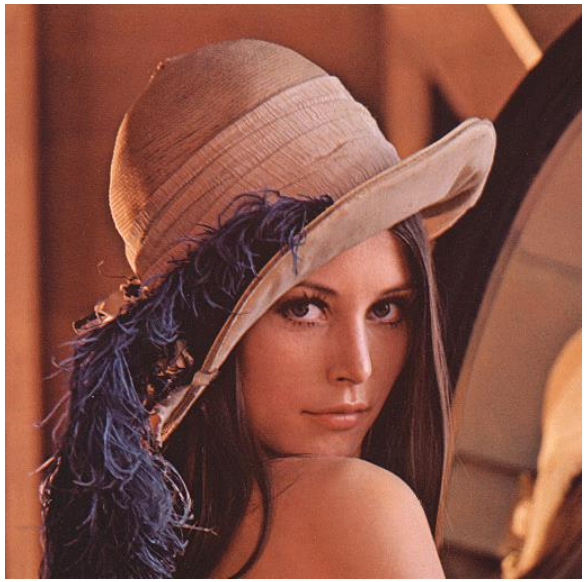
2 RGB转换为YCbCr

- 将RGB转换为YCbCr色彩空间

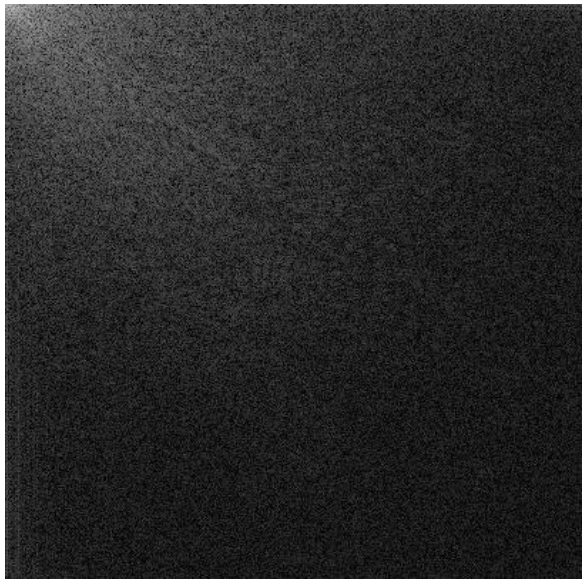
```
// rgb to ycbcr
dct_unit_y[pos] = 0.29900f * r + 0.58700f * g + 0.11400f * b - 128;
dct_unit_u[pos] = -0.16870f * r - 0.33130f * g + 0.50000f * b;
dct_unit_v[pos] = 0.50000f * r - 0.41870f * g - 0.08130f * b;
```

3 DCT变换

- 对每一个 YCbCr 色彩块的三个矩阵做 DCT 变换，变换后的数据左上角为低频部分，右下角为高频部分。



Lena



对于全图进行DCT变换

4 量子化

```
// quantization table
static const unsigned int quantization_table_y[] = {
        [0]: 16,  [1]: 11,  [2]: 10,  [3]: 16,  [4]: 24,  [5]: 40,  [6]: 51,  [7]: 61,
        [8]: 12,  [9]: 12,  [10]: 14,  [11]: 19,  [12]: 26,  [13]: 58,  [14]: 60,  [15]: 55,
        [16]: 14,  [17]: 13,  [18]: 16,  [19]: 24,  [20]: 40,  [21]: 57,  [22]: 69,  [23]: 56,
        [24]: 14,  [25]: 17,  [26]: 22,  [27]: 29,  [28]: 51,  [29]: 87,  [30]: 80,  [31]: 62,
        [32]: 18,  [33]: 22,  [34]: 37,  [35]: 56,  [36]: 68,  [37]: 109,  [38]: 103,  [39]: 77,
        [40]: 24,  [41]: 35,  [42]: 55,  [43]: 64,  [44]: 81,  [45]: 104,  [46]: 113,  [47]: 92,
        [48]: 49,  [49]: 64,  [50]: 78,  [51]: 87,  [52]: 103,  [53]: 121,  [54]: 120,  [55]: 101,
        [56]: 72,  [57]: 92,  [58]: 95,  [59]: 98,  [60]: 112,  [61]: 100,  [62]: 103,  [63]: 99
};
static const int quantization_table_uv[] = {
        [0]: 17,  [1]: 18,  [2]: 24,  [3]: 47,  [4]: 99,  [5]: 99,  [6]: 99,  [7]: 99,
        [8]: 18,  [9]: 21,  [10]: 26,  [11]: 66,  [12]: 99,  [13]: 99,  [14]: 99,  [15]: 99,
        [16]: 24,  [17]: 26,  [18]: 56,  [19]: 99,  [20]: 99,  [21]: 99,  [22]: 99,  [23]: 99,
        [24]: 47,  [25]: 66,  [26]: 99,  [27]: 99,  [28]: 99,  [29]: 99,  [30]: 99,  [31]: 99,
        [32]: 99,  [33]: 99,  [34]: 99,  [35]: 99,  [36]: 99,  [37]: 99,  [38]: 99,  [39]: 99,
        [40]: 99,  [41]: 99,  [42]: 99,  [43]: 99,  [44]: 99,  [45]: 99,  [46]: 99,  [47]: 99,
        [48]: 99,  [49]: 99,  [50]: 99,  [51]: 99,  [52]: 99,  [53]: 99,  [54]: 99,  [55]: 99,
        [56]: 99,  [57]: 99,  [58]: 99,  [59]: 99,  [60]: 99,  [61]: 99,  [62]: 99,  [63]: 99
};
```

$$B_{i,j} = \text{round}\left(\frac{G_{i,j}}{Q_{i,j}}\right) \qquad i,j = 0,1,2,\ldots,7$$

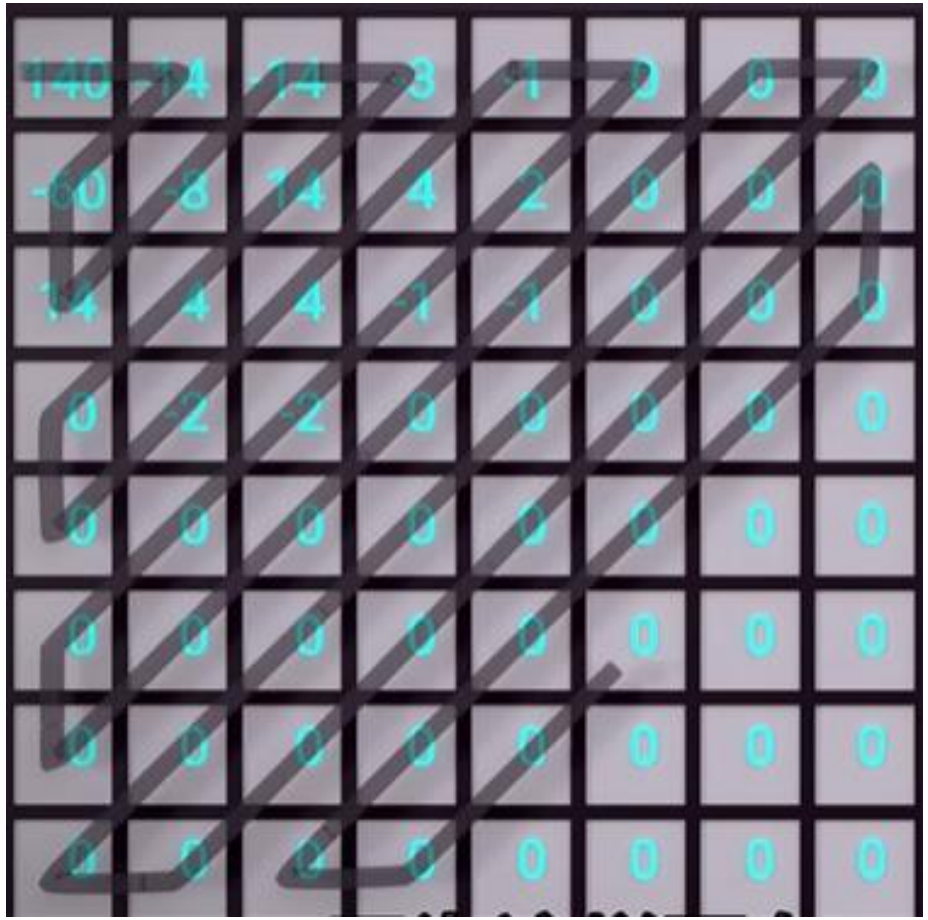
量化矩阵与前面得到的 DCT 矩阵逐项相除并进行四舍五入或取整。

量化矩阵是由预先设置的图像质量分数进行生成的。

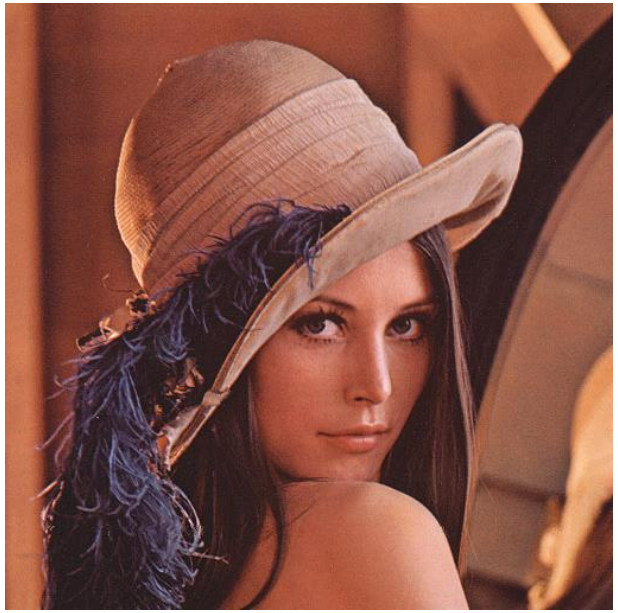
直流分量相对于交流分量来说基本要大很多，而且交流分量中基本会有大量的 0。

5 zigzag 与 霍夫曼编码

- Zigzag 扫描，获得一个尾数大多数为 0 的一块数据。
- 进行霍夫曼编码，采用 JPEG 推荐的亮度、色度，直流、交流的霍夫曼编码表，一共四张。
- 进行数据的霍夫曼编码，之前有写过。



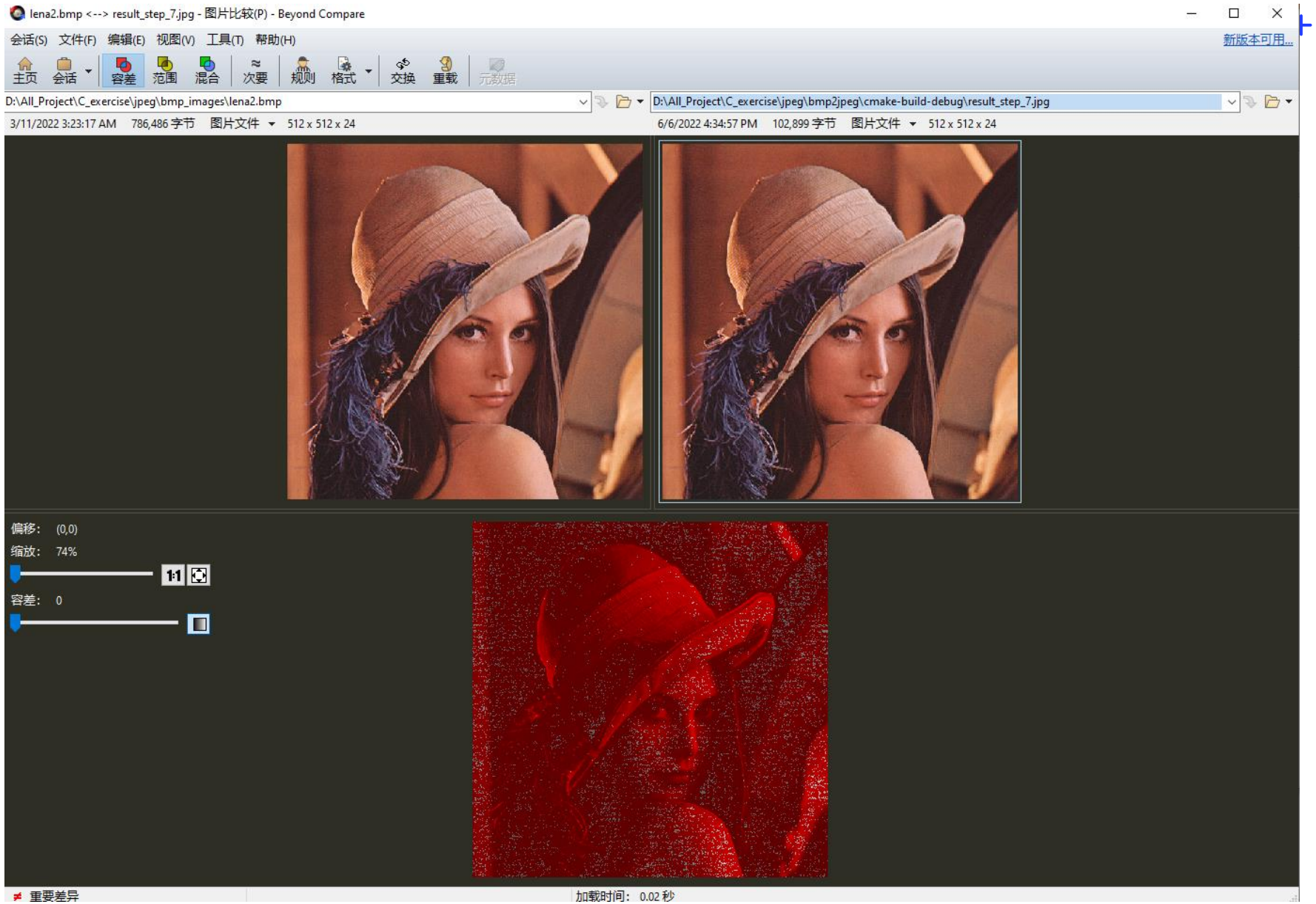
6 效果验证



Lena.bmp

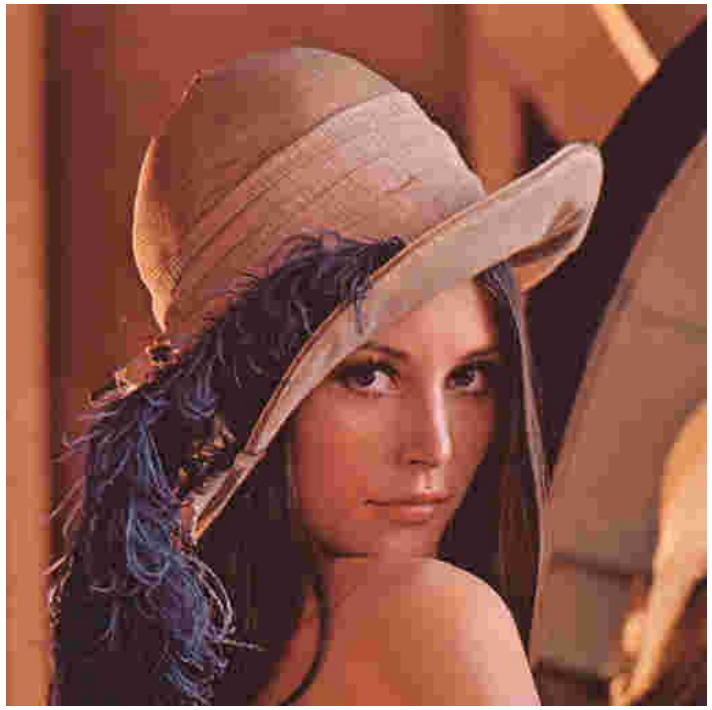
Lena.jpg
Windwos 自带编辑器保存的，
亲测质量数大约为 90

自己实现的过程，质量设置为90

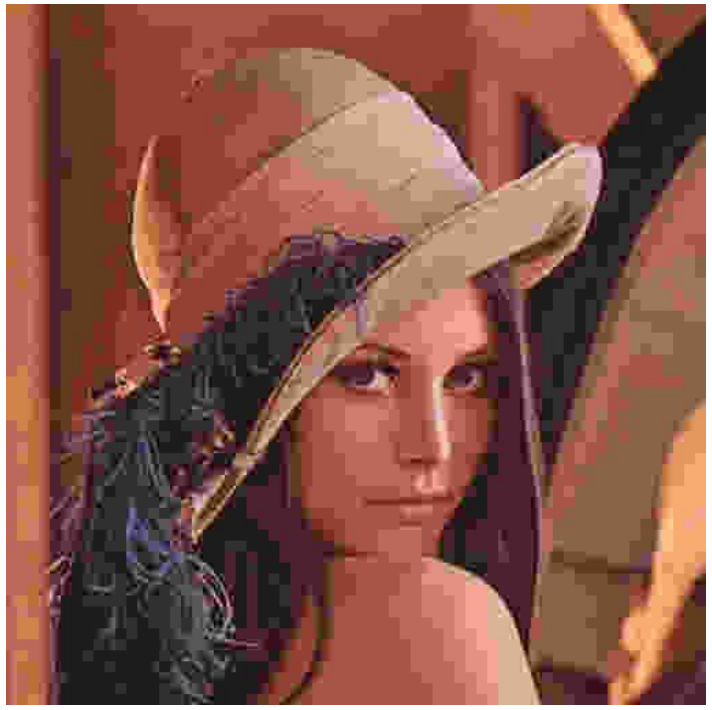


自己实现的图片，质量为 90
与源bmp图像的对比

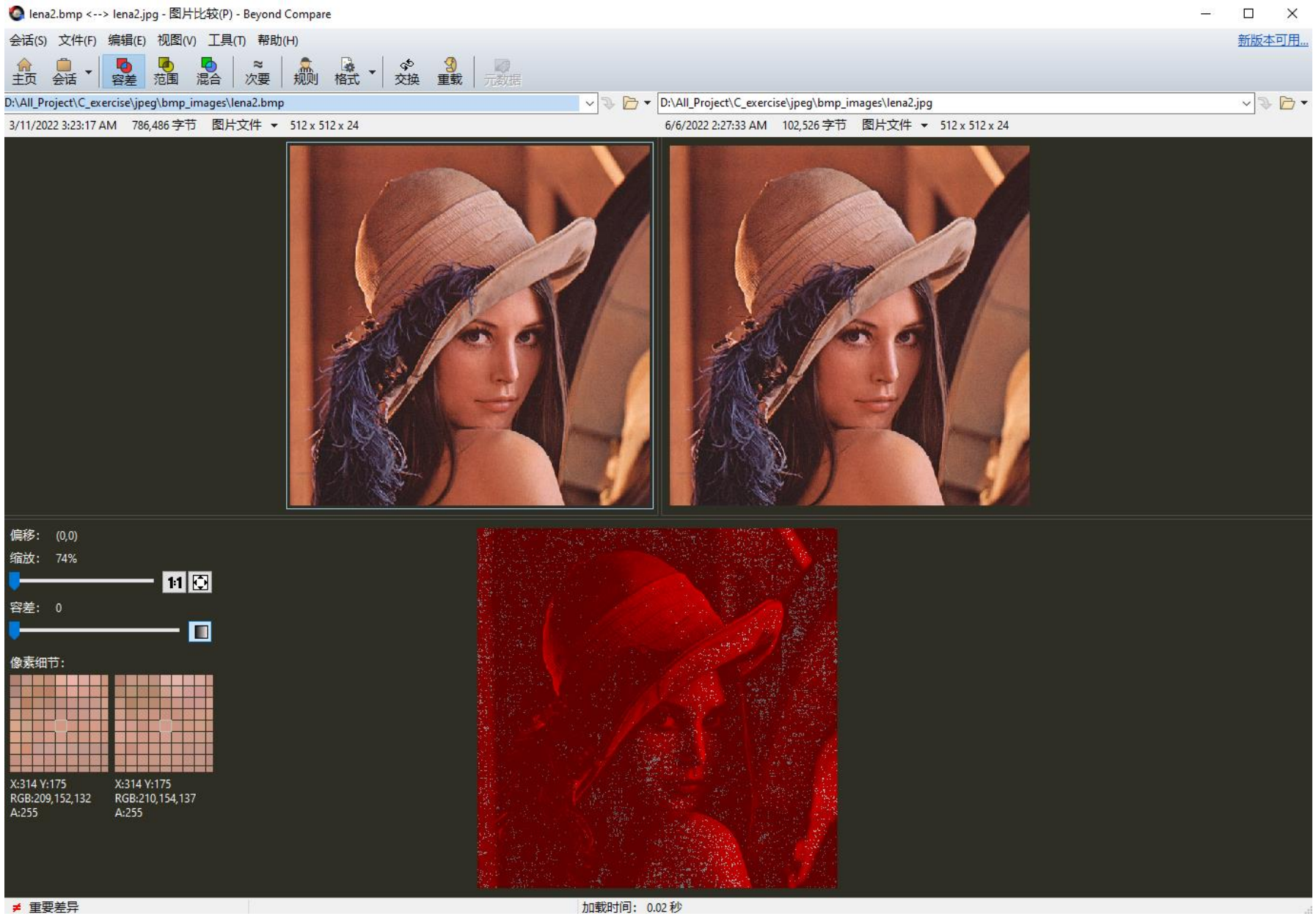
质量为50



质量为10



质量为 5



图片与源
RGB 数据差
异较大

Windwos 编辑器输出，质量为 90
与源bmp图像的对比

7 效率

- 对于整体程序结构和计算方式进行了一部分优化。
- 运行一次 512*512*24 bit 的 Lena 图片，大约耗时 15ms。

谢谢!

19182655 李霄龙