

数字图像处理实验四

09021220 周天逸

实验内容

1. 对自己选定图像分别进行JPEG 和 JPEG2000压缩, 可以调用函数, 不过需理解调用的函数以及参数。需要理解JPEG 和 JPEG2000压缩算法的思路和差异, 并思考解答当压缩比相同时, 为什么JPEG2000效果更好。

实验步骤

1. 设计思路

1. 本实验通过设置压缩参数k通过用户的输入来决定图像压缩的效果对比。
2. 设置两个按钮来实现不同压缩方法的对比

2. 压缩方法

1. JPEG

1. 通过使用opencv的函数imwrite和设置压缩参数来对imwrite进行设置。

```
vector<int>compression_params;  
compression_params.push_back(IMWRITE_JPEG_QUALITY);
```

2. 在设置是要注意图片的保存后缀为.jpg, 这样才可以保存正确的jpg文件格式

2. JPEG2000

1. 与JPEG类似的方法来初始化压缩参数, 只不过在压缩时要注意后缀应该为.jp2才是正确的格式。否则会出现因为后缀不同而效果不正确。

3. 展示方式

1. 实验最后通过将压缩后的图片转为QImage类似并在QLabel上展示的方式来显示实验结果。

4. 思考压缩效果的不同

1. 压缩算法的不同:

1. JPEG: 使用的是一种称为离散余弦变换 (DCT) 的算法。该算法将图像分解为一系列频率分量, 并对这些分量进行压缩。JPEG压缩在高压比下可能导致失真, 尤其是在重复使用相同图像进行多次压缩时。
2. JPEG2000: 使用的是一种称为小波变换的算法。小波变换能够更好地处理图像的局部特征, 并在高压比下提供更好的图像质量。JPEG2000还支持可变比特率压缩, 允许在不同区域和不同图像部分之间分配不同的比特率, 以获得更高的图像质量。

2. 小波变换比傅里叶变换好

1. 小波变换具有时频局部化的特性, 可以提供信号在时间和频率上的局部信息。这意味着小波变换能够更好地捕捉信号的瞬时特性, 对于非平稳信号的分析更为适用。
2. 小波变换具有多尺度分析的能力, 可以通过改变小波基函数的尺度来分析信号的不同频率成分。这使得小波变换在分析信号的不同频率分量时更为灵活。而傅里叶变换: 提供的是整体频谱信息, 不能轻松地实现多尺度分析。
3. 小波变换具有紧支撑的性质, 它只会分析一个有限区间内的变换。
4. 小波变换克服了窗口大小不随实际信号频率变换, 缺乏离散正交基的缺点。
5. 小波变换时间-频率窗口可变, 低频是宽, 分析高频时先择窄的。

代码展示

```
vector<int>compression_params;
compression_params.push_back(IMWRITE_JPEG_QUALITY);
int k = ui->para->value();
compression_params.push_back(k);

cv::imwrite("D:\\QT\\Qt_File\\ImageCompression\\JPEG_JPEG2000\\output\\file1.jpg",
img,compression_params);
Mat res1=
imread("D:\\QT\\Qt_File\\ImageCompression\\JPEG_JPEG2000\\output\\file1.jpg",0);
imshow("1",res1);
//QDebug()<<res1.channels();
QImage img1((uchar *)res1.data,res1.cols,res1.rows,res1.cols *
1,QImage::Format_Grayscale8);
QPixmap pix2 =change_type(img1);
ui->l2->setText("JPEG");
ui->p2->setPixmap(pix2);
ui->p2->setScaledContents(true);
```

1. 在图片压缩的时候先进行参数初始定义，后通过imwrite函数绘制.之后通过MSE函数计算出与原始图片的均方差。

计算MSE

```
double getMSE(const Mat& I1, const Mat& I2) {
    Mat s1;
    absdiff(I1, I2, s1);
    s1.convertTo(s1, CV_32F);
    s1 = s1.mul(s1);
    Scalar s = sum(s1);
    double mse = s.val[0] / (double)(I1.size().width * I1.size().height);
    return mse;
}
```

计算PSNR误差

```
double getPSNR(const Mat& I1, const Mat& I2) {
    Mat s1;
    absdiff(I1, I2, s1);          // 计算两幅图像的不同
    s1.convertTo(s1, CV_32F);     // 将数据转换成 float，以便进行误差平方

    s1 = s1.mul(s1);              // 计算每个像素的平方误差

    Scalar s = sum(s1);           // 对所有误差求和

    double sse = s.val[0] + s.val[1] + s.val[2]; // 总和
```

```
if (sse <= 1e-10) { // 对于非常小的值我们将返回无限大
    return 0;
} else {
    double mse = sse / (double)(I1.channels() * I1.total());
    double psnr = 10.0 * log10((255 * 255) / mse);
    return psnr;
}
}
```

结果展示

1. 压缩结果展示

原始图片



JPEG



JPEG2000




Choose

JPEG


JPEG2000

2. 均方差展示

原始图片



JPEG



Choose

JPEG

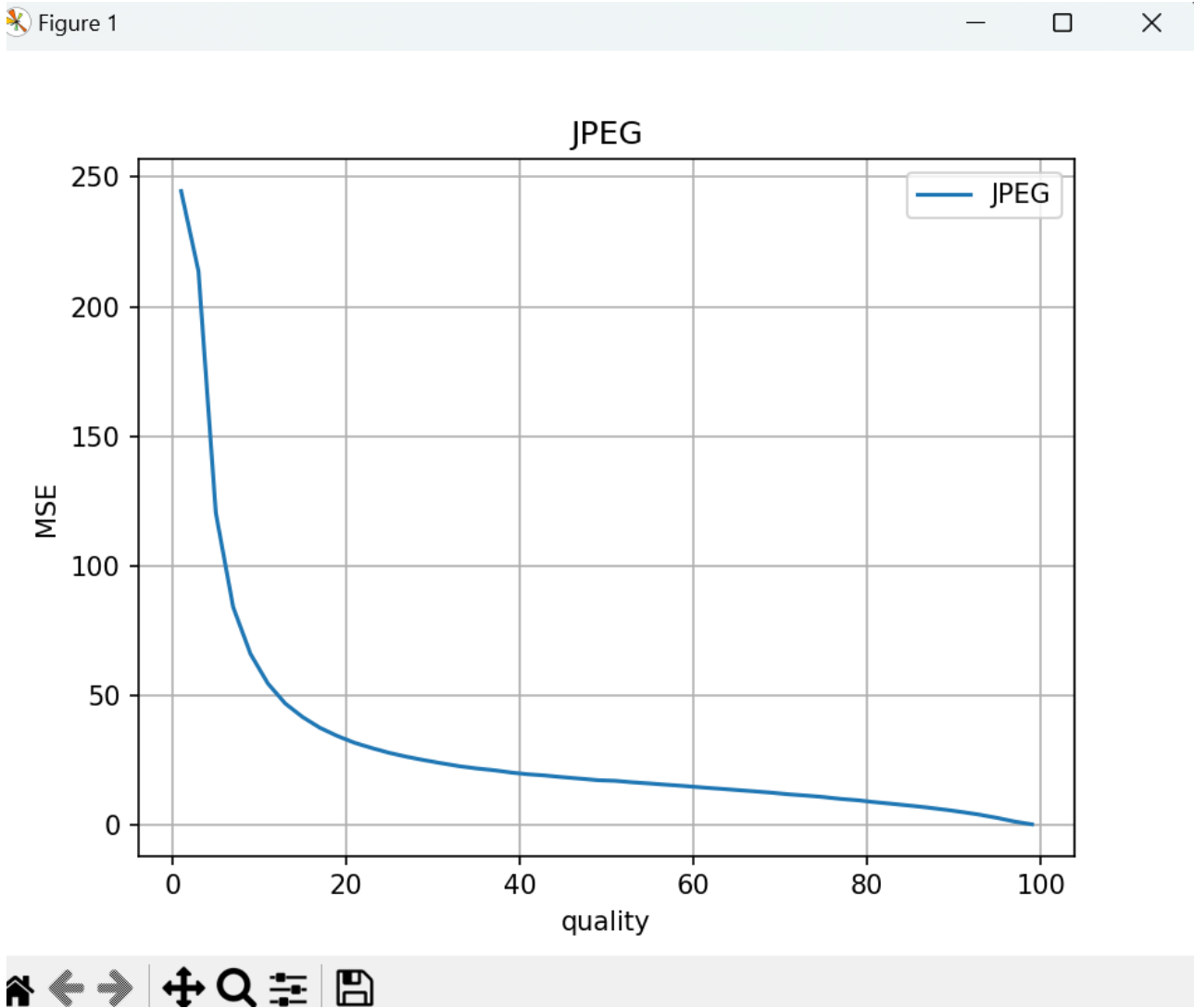
JPEG2000

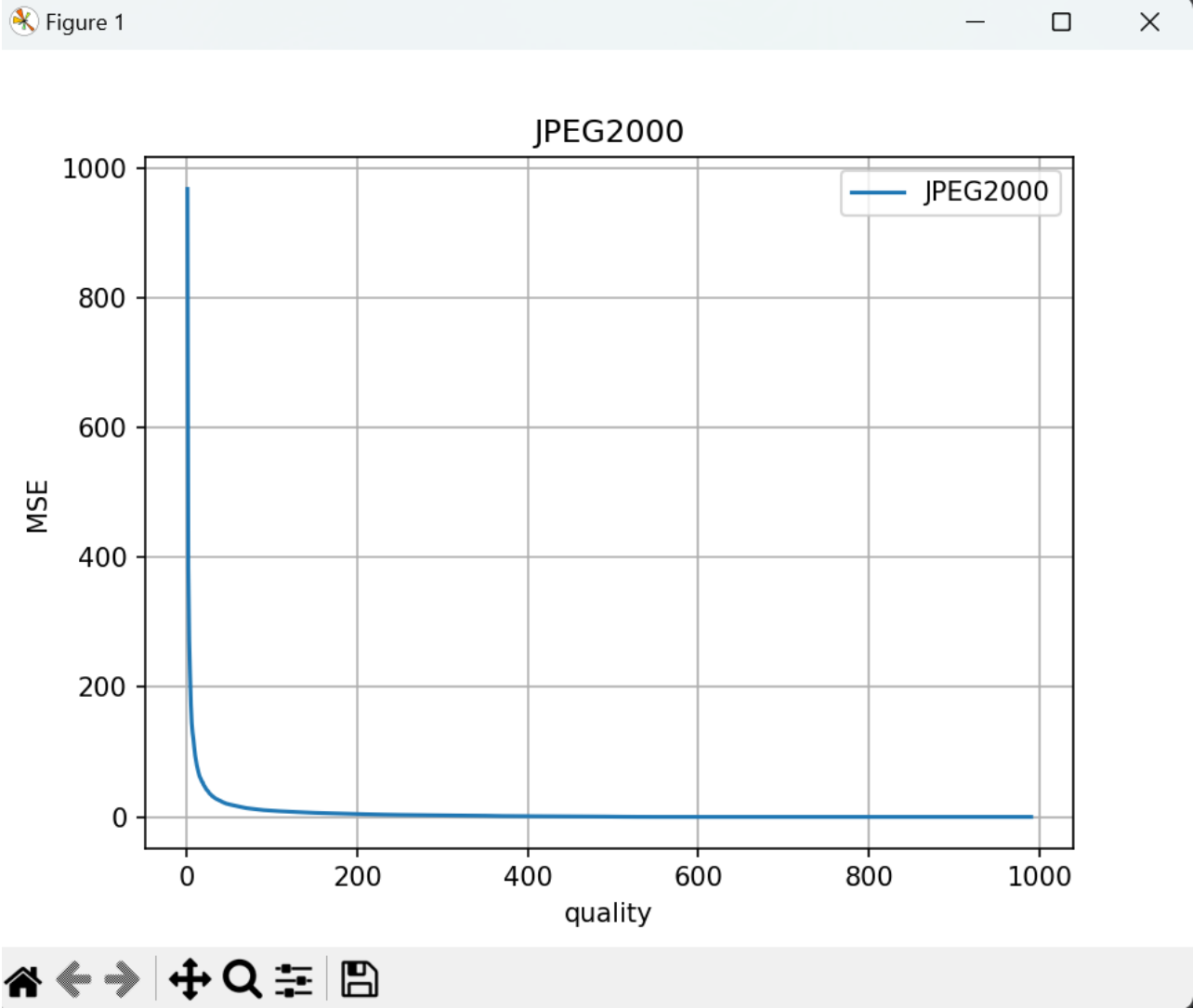
97.48

6

不同的压缩比，MSE的值不一样

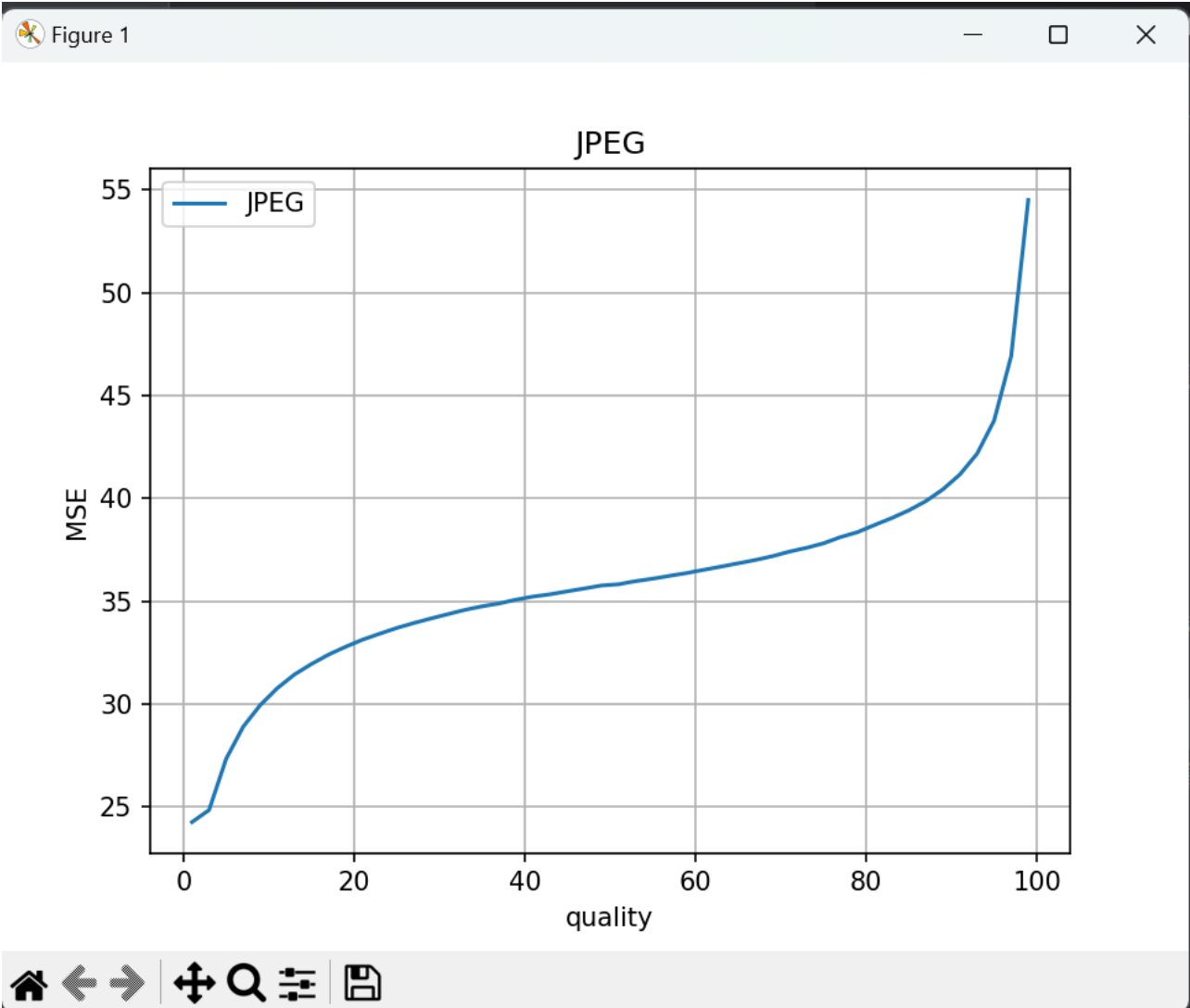
3. 使用MSE比较不同的压缩方式

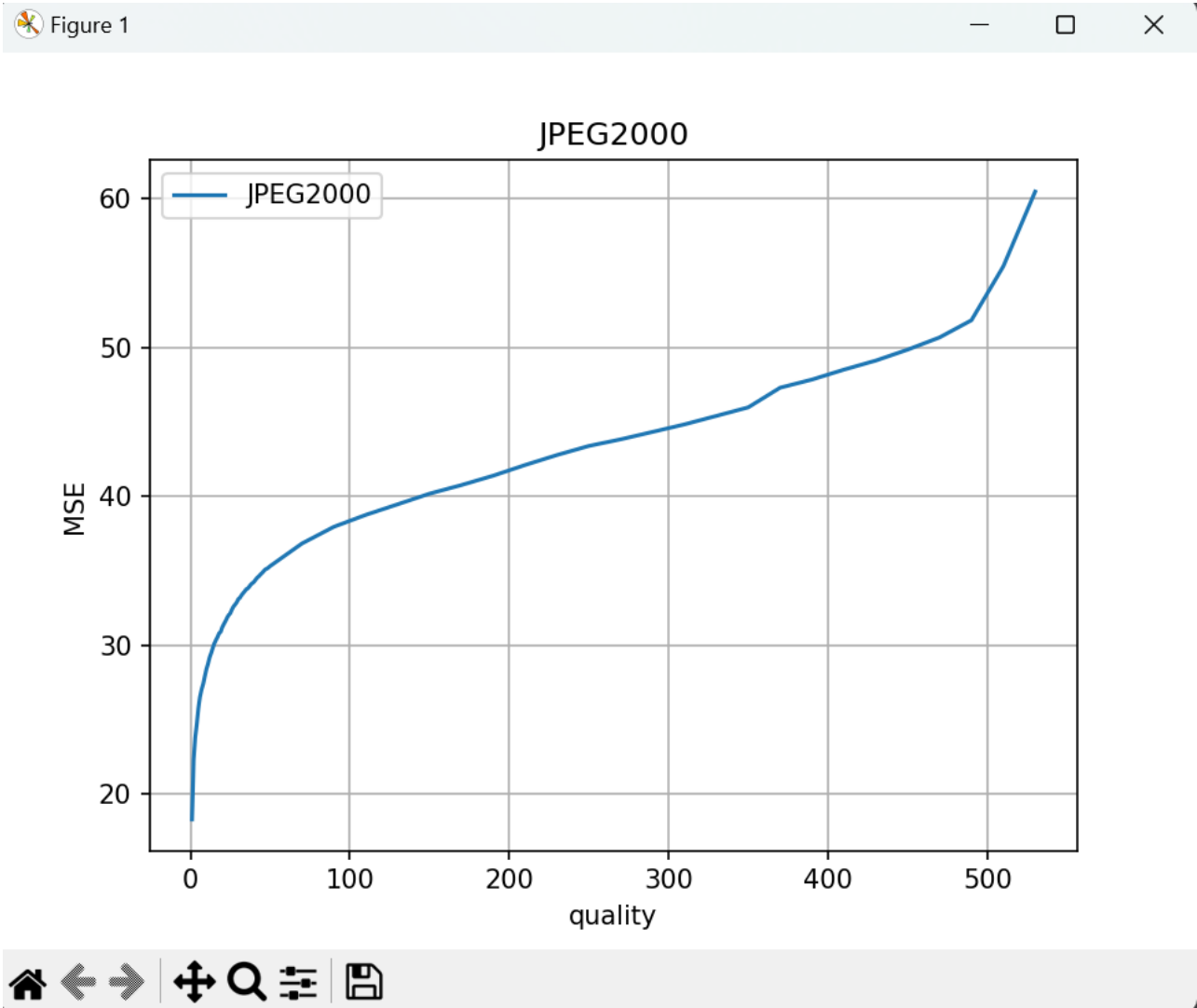




可以发现JPEG2000压缩效率比JPEG更好，在小范围的压缩比变化中，JPEG2000可以快速变小逐渐趋近于0.

4. 使用PSNR进行比较





PSNR (Peak Signal-to-Noise Ratio) 是一种用于衡量图像质量的指标，常常用于评估图像压缩算法的性能。PSNR的值越高，表示图像质量越好。对于图像压缩，PSNR提供了一种量化图像失真的方法，即压缩后图像与原始图像之间的差异。在这方面，PSNR的高值通常意味着压缩算法引入的失真较小，图像质量较高