opencv 和 libjpeg-turbo 的性能比较。

本文主要对比了 `opencv` （两种模式）和 `libjpeg-turbo` 在 raw 转换为 jpeg 格式上的性能差异。

- `opencv`

  内部使用 `libjpeg` 进行压缩，`opencv` 可通过 `cv::SetUseOptimized(bool onoff)` 方法开启或关闭优化（加速）

- `libjpeg-turbo`

  `libjpeg-turbo` 是 `libjpeg` 的复刻， 且通过 SIMD 指令加速了 jpeg 的压缩和加压缩过程。

测试

测试环境

- CPU : Intel i5-4590(4) @ 3.700GHz
- OS : win10
- IDE: Visual studio 2017
- opencv(4.5.1-release), libjpeg-turbo(2.1.0)
- 测试图片

  ```
  分辨率：8192*5000
  位深度：24
  格式：Bmp
  大小：117MB
  ```

相关变量

- 执行时间 $time$
  在执行体前后分辨执行 `clock()` 函数，得到 开始时间 t_1 和 结束时间 t_2 。

  $$time = (t_2 - t_1)/CLOCKS\_PER\_SEC$$

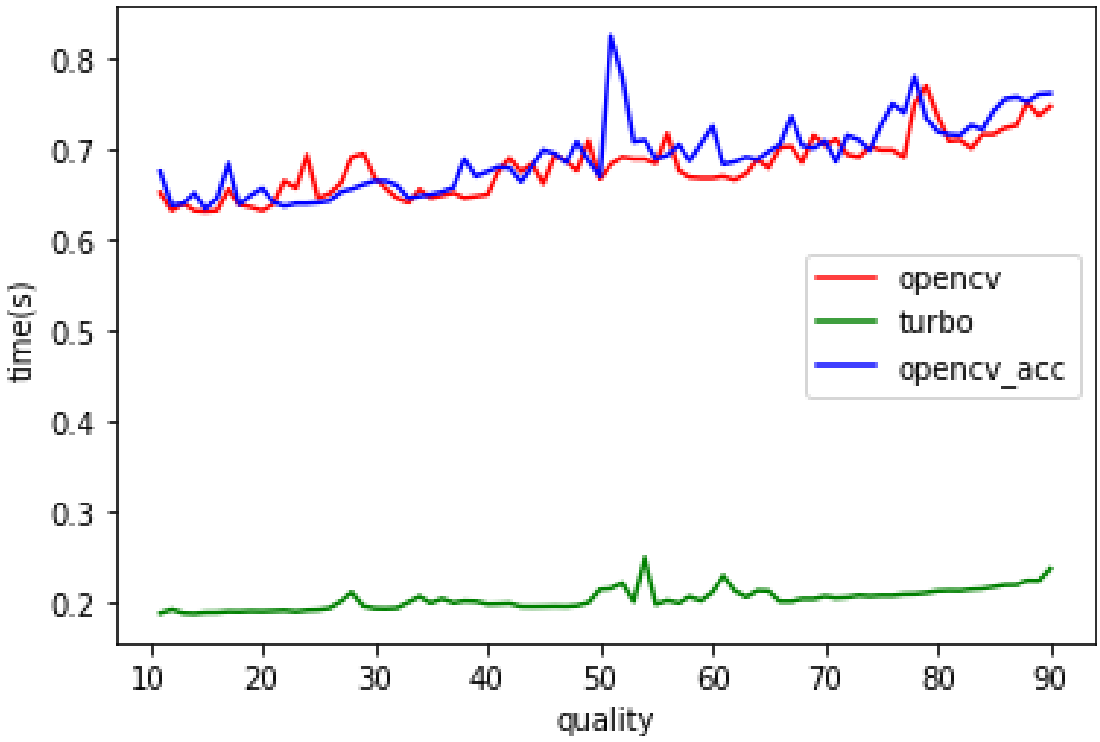
- 吞吐量 tp
  设图片读取进内存后所占的内存空间为s，压缩图片的执行时间为t，则$tp = \frac{s}{t}$

注意事项

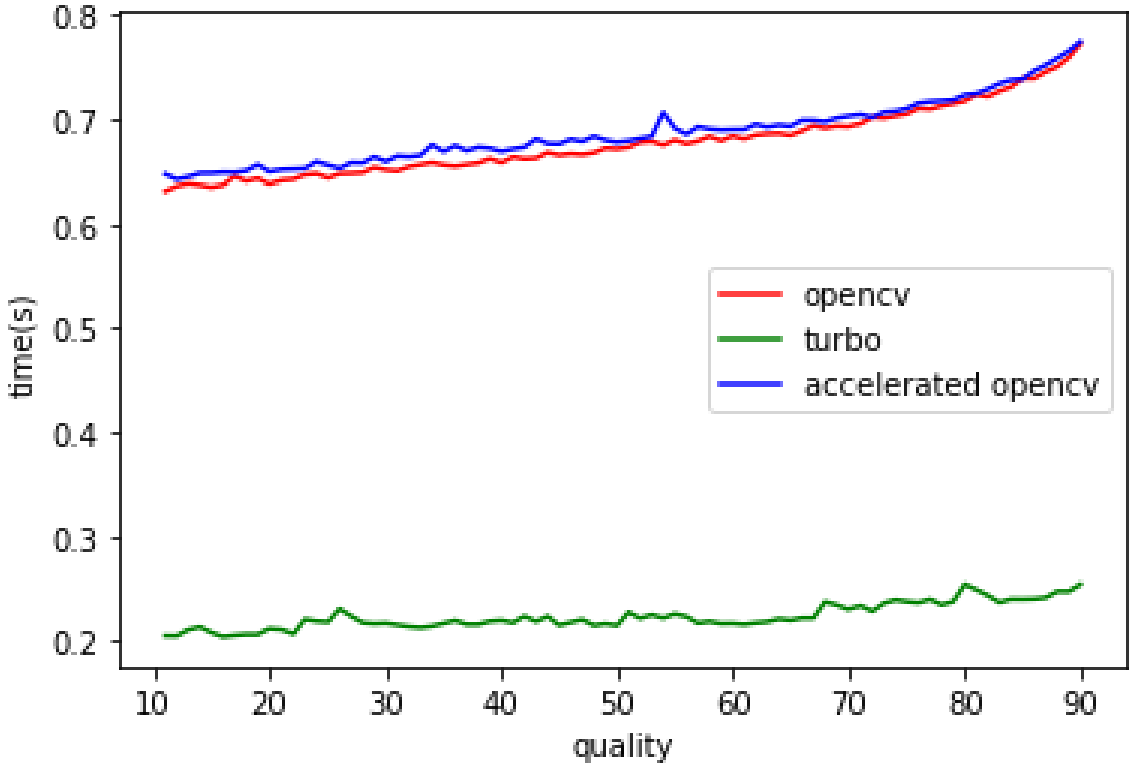
1. 相关变量随压缩进行趋于稳定。
2. opencv的debug版本和release版本性能差距太大，实际测试中使用release版本(release版本的吞吐量是debug版本的3倍左右)。

结果

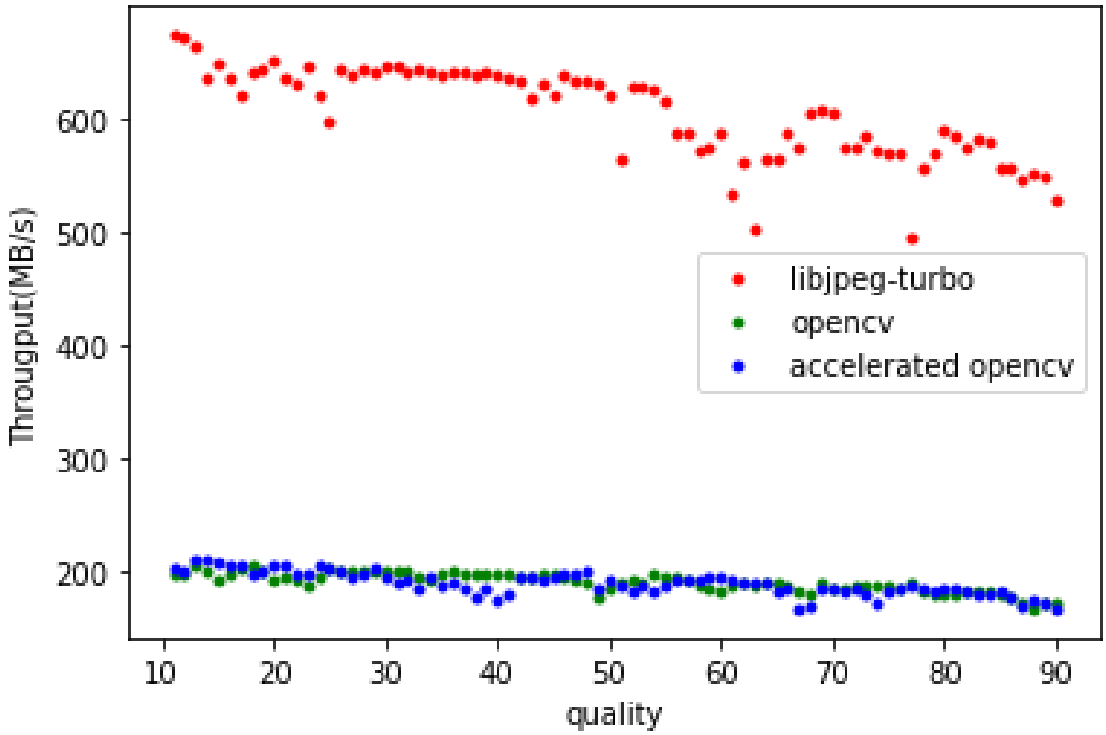
- 执行时间-压缩质量（不写入文件）



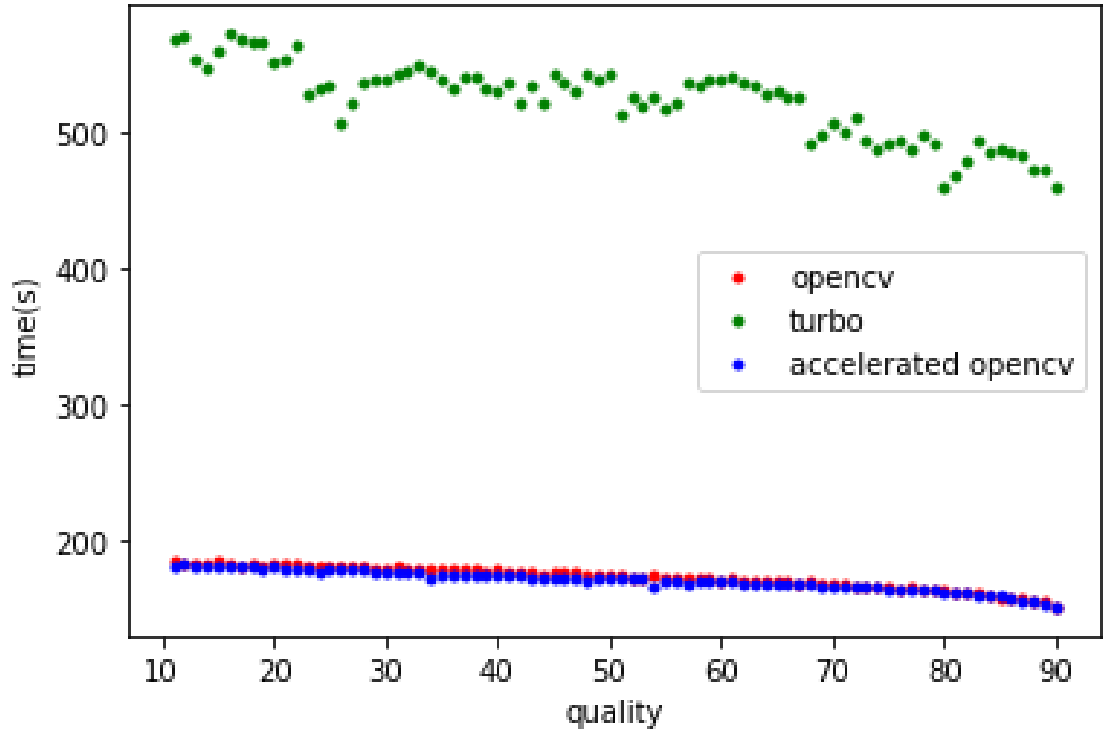
- 执行时间-压缩质量（写入文件）



- 吞吐量-压缩质量（不写入文件）

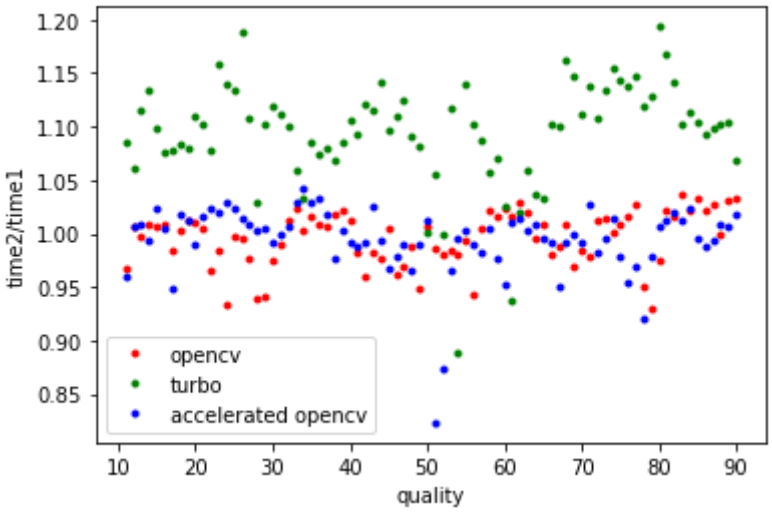


- 吞吐量-压缩质量（写入文件）



`opencv`加速效果不明显。`libjpeg-turbo`的效果明显好于opencv, 由于opencv内部使用的`libjpeg`进行压缩，而`libjpeg-turbo`是`libjpeg`的优化版本。

写入图片和不写入图片所用时间的比值 随质量的变换如下图所示。



对于`libjpeg-turbo`而言，写入图片平均多使用`9%`的时间。而对于`opencv`而言，写入图片和不写入图片所花时间相近，这表明`imencode`方法和`imwrite`方法的速度相近。

对于`libjpeg-turbo`而言，具体几个点的值：

quality	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
平均吞吐量（MB/s）	600	599.69	591.21	588.98	582.96	566.86	543.30	501.50

- CPU 使用率

观测方式：不断对图片进行压缩操作，使用资源管理器观测60秒的平均CPU使用率，待CPU使用率稳定后，记录当前的平均CPU使用率。

	opencv	**accelated opencv**	**libjpeg-turbo**
cpu使用率	24.47%	24.55%	24.36%

三者的cpu使用率没有太大区别。