Measurement analysis of speed across multiple methods of grey-scaling

Niels de Waal (1698041), Jasper Smienk(1700502)

April 15, 2018

Contents

1 Target

With this measurement analysis we want to find out how fast each grey-scaling method is and compare them to each other and the default implementation.

These results can help improve the speed of the facial recognition because converting an image to grey-scale is one of the steps that needs to be done. This step needs to be done for every frame to improve the facial recognition success rates. Having to do the grey-scaling every frame could, over time, add up to a lot of computation. While this time could be of high value in applications where the recognition process needs to be done in real-time.

2 Hypothesis

We suspect that of our methods the *decomposition* will be the fastest, as it required very little computation. Followed by *averaging* and lastly *luma*.

We suspect *decomposition* to be the fasted because of the fact that modern cpus have build-in instructions that can take care of the necessary analysis.

We can't say anything about how they will perform against the default implementation, as we don't know how it works.

3 Method

For each grey-scaling method, we will run it 10000 times and see how long it took from the start to the end. From this we will calculate the average time it took.

The test will be run twice for each method, once with the facial recognition, and once without.

We will make sure the tests are run on the same laptop and keep an eye out on the temperature to make sure it does not thermal-throttle.

Specifications of the used laptop:

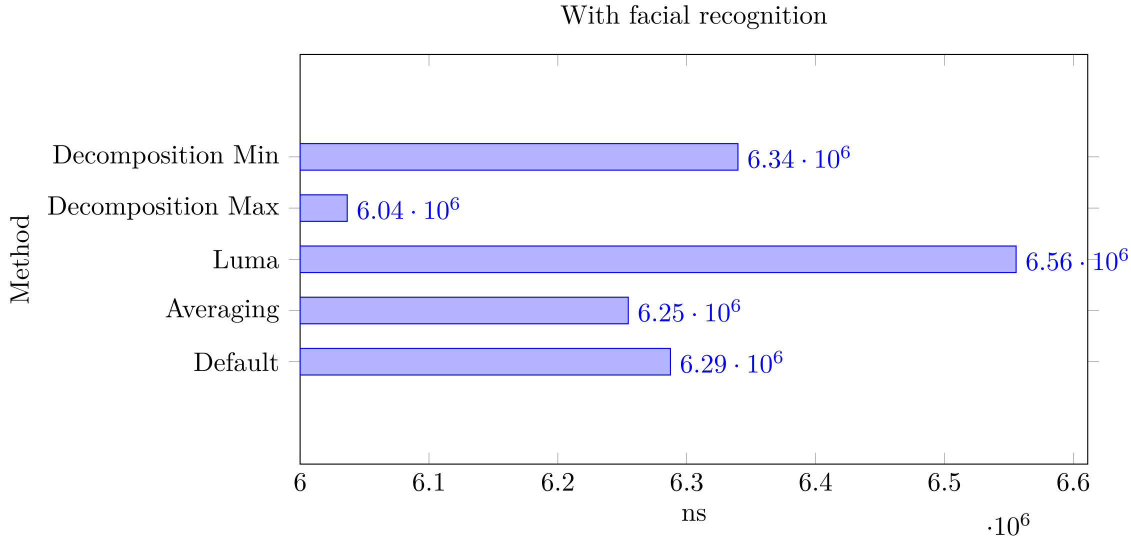
CPU	Intel i7 8550U [1]
RAM	16GB DDR4 2400MHz
OS	Linux 4.15.15-1-ARCH

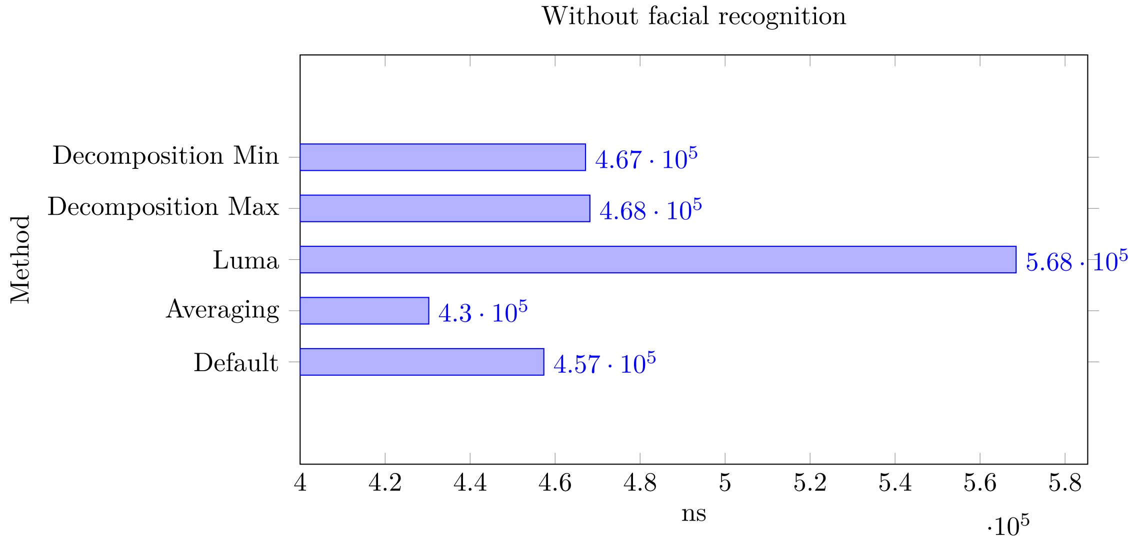
4 Results

4.1 Raw data

Method	Speed total (ns)	Speed greyscale only (ns)
Default	6287400	457354
Averaging	6254653	430275
Luma	6555633	568484
Decomposition (Max)	6036561	468192
Decomposition (Min)	6339750	467159

4.2 Visualized





5 Processing

When looking solely at which method is the fastest grey-scale method, *averaging* comes out ahead. Followed by the default method and both *decomposition* methods. At last is *Luma*, which was expected.

We suspect *luma* to be the slowest because of the necessary floating point calculations involved.

However, *decomposition* max comes out ahead when you also consider the computation time of the facial recognition that follows. Then comes *averaging*, the default and *decomposition* min. *Luma* is last yet again.

6 Conclusion

Depending on what you need, different methods are applicable. If you just want the fastest method, use *averaging*. However, if you want to use it with facial recognition, use *decomposition* max.

An other observation is that the facial recognition with the *Luma* method is noticeable slower, even though the grey-scaling is a small part of the whole process. This means that even though *Luma* makes the (subjectively) best looking images for the human eye, they make the facial recognition slower.

7 Evaluation

7.1 What went well

The project was a very cooperative experience for both parties involved. This cooperation went very well, everyone had their tasks and did them with equal amount of dedication. This made the project an overall pleasant experience.

7.2 What went wrong

We both had a vigorous start with the project, however it turned out to be very hard to keep up this speed. Eventually we started to have to divide our time to other projects that required our attention at the time. This made it so we had to do a lot of work near the deadline.

Also the original plan was to use OpenCL and SIMD for parallelism. Because of the state of the relevant tools, this turned out to be quite a bit harder than was originally thought.

7.3 What could be done differently

A better research on the more ambitious subjects should be done in order to avoid unnecessary delays.

8 Additional information

In this section there will be additional information about the system where the tests where run on.

8.1 LSCPU

```
Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):            8
On-line CPU(s) list: 0-7
Thread(s) per core: 2
Core(s) per socket: 4
Socket(s):         1
NUMA node(s):      1
Vendor ID:         GenuineIntel
CPU family:        6
Model:             142
Model name:        Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Stepping:          10
CPU MHz:           1772.800
CPU max MHz:       4000.0000
CPU min MHz:       400.0000
BogoMIPS:          3984.00
Virtualization:    VT-x
L1d cache:         32K
L1i cache:         32K
L2 cache:          256K
L3 cache:          8192K
NUMA node0 CPU(s): 0-7
Flags:             fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
    pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht
    tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon
    pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
    tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2
    ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe
    popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
    3dnowprefetch cpuid_fault epb invpcid_single pti tpr_shadow vnmi
    flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2
    erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt
    xsavec xgetbv1 xsaves ibpb ibrs stibp dtherm ida arat pln pts hwp
    hwp_notify hwp_act_window hwp_epp
```

References

[1] Intel Ark product specifications https://ark.intel.com/products/122589/Intel-Core-i7-8550U-Processor-8M-Cache-up-to-4_00-GHz