OpenCV & optimisations with Ne[10] for Video Stream Processing

Beau Johnston*
Open Parallel
(Dated: July 31, 2012)

1. ABOUT

The Android application known as NeVideo, was developed to examine the Sepia Toning operation on live video stream. Considerations and Results that were discussed in the report [1] are still applicable, as the video streaming application is an extension of the NeSnap application, where successive image frames are processed. The section on Sepia Toning from [1] Section 2 outlines the algorithm and implementation of the adopted Sepia Toning operation, this is identical to the one used in NeVideo.

2. BENCHMARKING

Benchmarking of NeVideo was performed by examining the execution times of 100 frames, a frame rate was then determined. The adopted metric for analysing the frame rate, is established as Frames Per Second (FPS). A FPS for each of the 100 frames was recorded for both with and without Ne[10]. The results as shown in Figure 1 and 2.

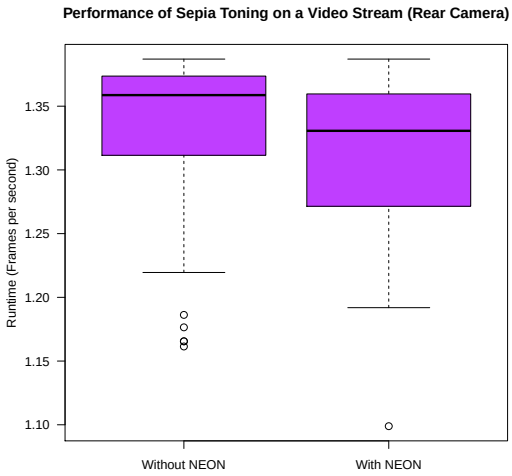

FIG. 1: The framerate per seconds with and without Ne[10] as a box plot.

*Electronic address: beau.johnston@openparallel.com; URL: http://openparallel.com

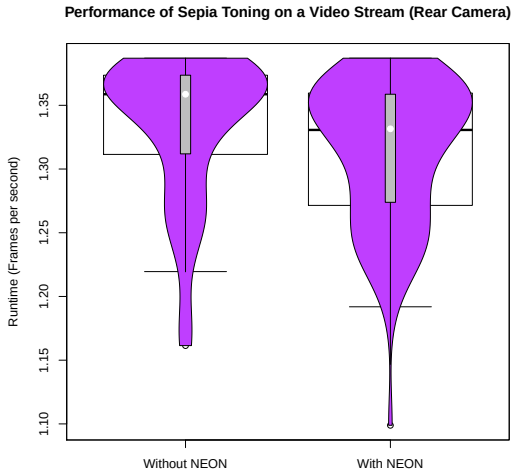

FIG. 2: The framerate per seconds with and without Ne[10] as a violin plot.

From Figure 2 we can see that NeVideo without Ne[10] optimisations achieves 1.33 ± 0.057 FPS whilst the result with Ne[10] is 1.31 ± 0.056. The higher FPS value achieved without Ne[10] is preferred. To confirm that the two means are statistically significantly different we perform a Welch two-sample t test. The results were:

```
> tTest

Welch Two Sample t-test

data:  WO and WN
t = 2.48, df = 198, p-value = 0.006953
alternative hypothesis: true difference \\
in means is greater than 0
95 percent confidence interval:
 0.00662     Inf
sample estimates:
mean of x mean of y
     1.33      1.31
```

Therefore we reject the null hypothesis that the means are the same as the p-value is less than 0.05 (the value below which the differences in the means is deemed statistically significant). In other words the mean runtime performance using Ne[10] is statistically significantly better than without using Ne[10]. It is concluded that using Ne[10] provides a marginal improvement in code execution time for video stream processing when Sepia Toning.

3. CONCLUSIONS

The results correspond to those achieved in [1], which was expected, due to the similar nature of the computation and its implementation. The only difference between projects is that NeVideo operates on a video stream which successively performs identical image processing operations on a series of images. The difference between frame rates of benchmarks of implementations with and without Ne^{10} is statistically, but not computationally significant. We conclude that there are negligible benefits to utilising Ne^{10} with this method of image processing, when being applied on video streams.

4. DISCUSSION & FURTHER CONSIDERATIONS

Additionally to the considerations discussed in [1] (which specifically target image processing optimisations), symmetric multiprocessing (SMP) research should be conducted both for the NeSnap and NeVideo projects.

SMP introduces parallelisation challenges around how image and video processing operations can be partitioned for different computing cores. A majority of this research should be based around examining the overhead of thread based partitioning and linear dependencies amongst image frames (in video stream processing) and pixel regions (in image processing). It is expected that task partitioning for the SMP will theoretically yield a speedup factor of as many cores as are available. For instance, quad core CPU (4 cores are available) bound image processing could optimally be performed in $\frac{1}{4}$ of the time. This is the theoretical optimal case, as there are overheads around the partitioning and synchronisation stages of processing that will impact on the speedup factor.

Task partitioning for image and video processing with SMP is an independent strategy to achieve faster image and video processing, and therefore separate to optimisations with the Ne^{10} library, as such, research has not yet been conducted. Another factor is the complexity of task partitioning and the long time frame that such research involves. Task partitioning for the SMP should be pursued as an unassociated research project.

5. SOURCE CODE

Both Ne^{10} enabled and regular Android NeVideo.apk files are freely available for download https://github.com/openparallel/NeVideo .

All source code can be found here https://github.com/openparallel/VideoStreamProcessor. To use Ne^{10} from source, uncomment `//#define USINGNEON` in `ImageProcessor.h`. To collect 100 FPS of runtime values, use the source from the benchmarking branch. The runtimes will be saved to /sdcard/runtimes.txt.

[1] Johnston, B. (2012). *OpenCV optimisations with Ne^{10} using auto-vectorisation for the Cortex-A9.* Open Parallel.