• Table of Contents

1 Binaries....2
2 Benchmark.exe....2
3 Devices....2
4 Performances....2
4.A Best mode....3
4.B Fast mode....4
5 Recommendations....4

1 Binaries

The binaries are available at https://example.zip and contains **ultimateALPR 2.6.0** (04/27/2020) using **Tensorflow 1.15**. Make sure you're using this version or later.

The binaries come with to versions: **Fast** and **Best**.

- **Fast:** Uses sparse models without native rectification layer. The rectification layer could be activated separately using the configuration options as per https://www.doubango.org/SDKs/anpr/docs/Configuration_options.html#recogn-rectify-enabled. This version is recommended if the license plates are moderately skewed or slanted. This is the default version for ARM devices.
- **Best:** Uses dense models with native rectification layer. The rectification layer is native and always active. This version is slower than fast but is more accurate specially when the plates are distorted. This is the default version for x86_64 devices.

More information about the rectification layer could be found at https://www.doubango.org/SDKs/anpr/docs/Rectification_layer.html .

2 Benchmark.exe

For benchmarking we'll use benchmark application with source code at https://github.com/DoubangoTelecom/ultimateALPR-SDK/tree/master/samples/c%2B%2B/benchmark and documentation at https://github.com/DoubangoTelecom/ultimateALPR-SDK/blob/master/samples/c%2B%2B/benchmark/README.md. You don't need to rebuild the application by yourself the executable is at **WIN64_dist\binaries\windows**.

3 Devices

For the benchmark we'll use 3 devices: **D1** (Desktop, Windows 7, i7-4790K, #4 cores @4Ghz) and **D2** (Laptop, Windows 10, i7-4770HQ, #4cores @2.2Ghz) and **D3** (Server, Ubuntu 18, Intel® Xeon® E3 1230v5, GeForce GTX 1070, #4 cores @3.4Ghz).

- **D1:** https://ark.intel.com/content/www/us/en/ark/products/80807/intel-core-i7-4790k-processor-8m-cache-up-to-4-40-ghz.html
- **D2:** https://ark.intel.com/content/www/us/en/ark/products/83505/intel-core-i7-4770hq-processor-6m-cache-up-to-3-40-ghz.html
- **D3:** https://www.ikoula.com/en/dedicated-server/xeon-gpu

Both D1 and D2 are old 6yr+ (2014) CPUs to make sure everyone can easily found them at the cheapest price possible. D3 is a mid-range GPU (hosting price: €89 / month).

4 Performances

All tests are done on HD images (**1280x720**) using a **#100 loops**. The image decoding is done outside of the loop for the simple reason that the SDK accepts raw images (RGB or YUV) and doesn't provide decoding functions.

4.A Best mode

The is the default mode. The DLL is at **WIN64_dist\binaries\windows\best\ultimateALPR-SDK.dll**.

You can copy it to **WIN64_dist\binaries\windows** to override the default DLL.

	No plate on the image	**#1 plate on the image**
D1	Avg time per frame: **41.92 millis** Avg frame rate per second: **23.85**	Avg time per frame: **55.69 millis** Avg frame rate per second: **17.95**
D2	Avg time per frame: **77.61 millis** Avg frame rate per second: **12.88**	Avg time per frame: **90.87 millis** Avg frame rate per second: **11.01**
D3	Avg time per frame: **9.78 millis** Avg frame rate per second: **102.17**	Avg time per frame: **11.74 millis** Avg frame rate per second: **85.11**

To run **"No plate on the image"** test:

```
WIN64_dist\binaries\windows\benchmark.exe ^
    --positive ../../assets/images/lic_us_1280x720.jpg ^
    --negative ../../assets/images/london_traffic.jpg ^
    --assets ../../assets ^
    --loops 100 ^
    --rate 0.0 ^
    --parallel true
```

To run **"#1 plate on the image"** test:

```
WIN64_dist\binaries\windows\benchmark.exe ^
    --positive ../../assets/images/lic_us_1280x720.jpg ^
    --negative ../../assets/images/london_traffic.jpg ^
    --assets ../../assets ^
    --loops 100 ^
    --rate 1.0 ^
    --parallel true
```

As you can notice, the OCR part is very fast while the detection is slow. Using images with more plates have little impact on the speed.

If you have already used ultimateALPR on ARM devices you can remark that it's faster (relative speed based on CPU frequency) than x86_64 CPU-only. The reason is that on ARM we use INT8 quantized models. On x86_64 we haven't added quantization yet and this why it's recommended to use GPU versions. Adding quantized models for x86_64 and support for Intel MKL is on our roadmap with high priority.

4.B Fast mode

The DLL is at **WIN64_dist\binaries\windows\fast\ultimateALPR-SDK.dll**. You have to copy it to **WIN64_dist\binaries\windows** to override the default DLL.

	No plate on the image	**#1 plate on the image**
D1	Avg time per frame: **43.01 millis** Avg frame rate per second: **23.24**	Avg time per frame: **44.42 millis** Avg frame rate per second: **22.50**
D2	Avg time per frame: **69.23 millis** Avg frame rate per second: **14.44**	Avg time per frame: **74.88 millis** Avg frame rate per second: **13.35**
D3	Avg time per frame: **9.66 millis** Avg frame rate per second: **103.49**	Avg time per frame: **14.18 millis** Avg frame rate per second: **70.51**

As you can notice, there is almost no difference between fast and best when there is no image on the plate. This is expected as they use same detection models but different OCR model as explained above.

5 Recommendations

- Make sure to not include the image decoding in the loop. Decoding is up to you and must not be timed.
- We only load the deep learning models when the detection process is called. Same rule apply to the recognition process. Loading the models is slow and could take few seconds. Do not include the first call in the loop. We've a warm-up function used to force loading the models. You must call the warm-up function before starting the timing as done at https://github.com/DoubangoTelecom/ultimateALPR-SDK/blob/7761433d766291bf68d95e93cb14dff9f14567c5/samples/c%2B%2B/benchmark/benchmark.cxx#L214. We don't load the models at startup for the simple reason that the engine could be used to generate license keys, runtime keys... and such operations don't need detection or recognition operations.
- Activate parallel mode on low-end device (like **D2**) to improve the speed. Parallel mode doesn't change the accuracy. More info about parallel mode at https://www.doubango.org/SDKs/anpr/docs/Parallel_versus_sequential_processing.html#parallelversussequentialprocessing.
- Please check https://www.doubango.org/SDKs/anpr/docs/Improving_the_speed.html on how to improve the speed.