Maintaining OpenCV 3.x

- Author: Vadim Pisarevsky
- Link: TBD
- Status: **Draft**
- Platforms: All
- Complexity: a few man-weeks per year (hopefully)

Introduction and Rationale

OpenCV 3.x is the release series maintained since 2015 June. It's been an evolutionary and noticeable improvement over 2.4.x and brought the following major features:

- Revised OpenCL acceleration, a.k.a. T-API, with semi-automatic fallback to CPU.
- Subset of IPP (IPPICV) is included into the library by default (though, it can be explicitly excluded). It accelerates some basic core & imgproc OpenCV functionality noticeably.
- Introduced opencv_contrib repository with experimental functionality. At the moment, the repository includes over 30 modules.
- Introduced OpenCV HAL, low-level synchronous (i.e. CPU-oriented) API that can be used to accelerate various OpenCV functions on different platforms. Currently, there is ARM acceleration backend by NVidia. Part of IPPICV acceleration is also done via HAL API.
- A major cleanup of the library has been done where we mostly replaced internal C-like code that used OpenCV 1.x API with the new versions using C++ API. The external API has also been cleaned, and we now extensively use the “open interface + hidden implementation” pattern.
- ML module has been completely rewritten using the new-style C++ API.
- A lot of new functionality has been added, mostly as user contributions and results of GSoC 2015, 2016 and 2017.
- Doxygen replaced RST (ReStructuredText) as the documentation generation tool. For big textual chunks of documentation, such as the tutorials, we now use markdown.

At the same time, many fundamental aspects of the library have been retained:

- The library is built up as a collection of modules with well-defined directory structure (include, src, test, perf, samples etc. subdirectories)
- CMake is used as configuration and build system.
- Retained the same set of basic types: `cv::Mat`, `std::vector<>`, `cv::Matx<>`, `cv::Vec<>`, `cv::Ptr<>`. `cv::Mat`-like `cv::UMat` type has been added to implement T-API.
- The basic modules `core`, `imgproc`, `highgui`, i.e. the mostly used ones, are pretty much compatible with OpenCV 2.x (even though `highgui` has been split into `highgui`, `imgcodecs` and `videoio`).
- Python bindings were extended to support Python 3. This is intended to be a continuously updated meta-proposal, gathering information on what to do with OpenCV 2.4.x, which is in the maintenance mode for a few years already (since OpenCV 3.0).

We plan to support OpenCV 3.x for a few more years. As it's done with OpenCV 2.4, we will switch to the low-profile maintenance mode as soon as OpenCV 4.0 is released. It means that no new features will be added to OpenCV 3.0, just bug fixes.

Proposed solution

- Create 3.x (name?) branch and review/accept PRs coming to it. For now (2018 Spring) we plan to do it for at least a few more years. Later on, there can be some volunteers engaged to review the patches.
- Define the period when 3.x will reach the end-of-life and what it will mean for us. One possible scenario is that at some point OpenCV core team will stop creating backports of various patches for 3.x. But we can still

review and integrate patches from users, because it's low traffic and correspondingly low cost for us.

- One question to discuss is whether we should do 1-time or regular synchronization of some files in OpenCV 3.x and 4.x, such as video I/O code etc. Usually, it is this non-algorithmic code that needs backports in order to support new OSes, updated APIs etc. This synchronization could simplify propagation of fixes to all OpenCV versions.

Impact on existing code, compatibility

The upcoming OpenCV 3.x releases are going to be compatible with the previous ones. Users should not be affected.

Possible alternatives

There is no clear alternative - we need some support period anyway.

References

1. OpenCV master branch, from which we are going to fork 3.x branch at some point