Establish a subset of bit-exact functions in OpenCV

- Author: Vadim Pisarevsky
- Link: The feature request
- Status: **WIP** (e.g. bit-exact Gaussian Blur, bit-exact warp affine)
- Platforms: **All**
- Complexity: several man-months

Introduction and Rationale

Normal practice used when developing cross-platform libraries of numerical algorithms is that there is the reference branch (which can, however, be parallel, include various algorithmic optimizations etc.) and then there can be specialized branches of the same algorithm optimized for particular platform, e.g. a branch with AVX or NEON intrinsics, or GPU-accelerated branch etc. Often those optimized branches may give slightly different results, and so the corresponding regression tests check the results within a certain tolerance threshold.

Even if there are no optimized branches and we have just one C++ code, it may give varying results depending on the platform, compiler etc. For example, if certain platform has FMA (fused-multiply-add) instruction and the other does not, the same floating-point processing code will likely produce different results.

At the same time, if we consider some more or less complex pipeline that includes such functions, there is big chance that the results will be very different and there will be no easy way to match the results. For example, if we consider 2d feature matching pipeline (e.g. for image stitching), it includes the following steps:

1. some image proprocessing (RGB to gray conversion, resize, blurring etc.)
2. feature detector
3. descriptor extractor
4. feature matcher and then RANSAC homography esimator.

if the RGB to gray conversion function computes slightly different grayscale image, depending on the platform, the feature detector, even if it's bit-exact algorithm like FAST, may find slightly different feature set, then we get different descriptors; it will result in a different set of matches and correspondingly slightly different homography matrix. Therefore, it would be very difficult to do regression testing of the pipeline and its parts.

Therefore, it would be very useful to establish a set of function, especially the ones usually used in the beginning of typical pipelines, that are guaranteed to give the same results on each single platform.

Proposed solution

1. There should be a list of bit-exact functions, published somewhere in wiki or at docs.opencv.org. Obviously, those will be functions that can be implemented without floating-point operations *inside loops*. Note the last two words. Some functions, e.g. Gaussian Blur on 8-bit images, still need to do some preliminary computations using floating-point arithmetics (e.g. the kernel coefficients in the case of Gaussian Blur), but the actual data processing can then be done completely in fixed-point arithmetics. In this case we are using a variation of softfloat package embedded to OpenCV, i.e. we emulate those operations using integer arithmetics and therefore we guarantee absolutely the same results on each single platform.

2. We [re]implement those functions w/o using floating-point arithmetics inside the loops. Some functions are already de-facto bit-exact, e.g. morphological operations on images; some functions can easily be converted to bit-exact; some may require substantial modifications.

3. The corresponding tests for those functions are modified (or new tests created) to ensure bit-exactness.

4. Preferably, OpenCL kernels should be adjusted as well. Unfortunately, on GPUs floating-point is usually a faster type than integers, and so the performance may drop a bit.

Impact on existing code, compatibility

The behavior of certain functions will change, so some code and some tests that depend on the earlier non-bit-exact implementations may stop working. There can be the following solutions:

1. separate bit-exact variants from the previous ones (separate API or dedicated flags)
2. just ask people to update their tests.

we use combination of the approaches currently. E.g. in the case of resize we use the first option, in the case of GaussianBlur we use the second option.

Possible alternatives

If we use the same source from which both platform-specific CPU and GPU code are automatically generated then bit-exactness is somewhat less important. The most notable example of such universal retargetable code generation technologies is Halide.

References

TBD