Spec - OpenCV Logging Revamp - revised 2019-03-20

Preface

Disclaimers

- This is a draft.
- This spec has not yet been endorsed by its writers (kinchungwong) or by the OpenCV project maintainers.
- Subject to change.
- The spec and the code contribution shall be seen as independent activities; it is possible for this spec to survive even if the code contribution is rescinded or rejected.

Current draft status

- Awaiting first time public review and proofreading as of `2019-03-20T14:30Z`.

Slangs and shorthands

- [[**TBD**]] stands for "to be determined later"

Internet-accessible links to this draft

- [[**TBD**]]

Main content

Tag information that accompanies logging statements

- Tag information provide additional scope and context accompanying the logging statements.
- Tag information enable easier and more efficient filtering of log messages.
- Use of tag information shall be optional, not forced upon.

Major factors in the design and implementation of tag information

- Injection of "tag information" into existing code throughout the library can be done with minimal change to existing code.
- Compatibility shall be maintained with all pre-existing logging statements throughout the library.

Versioning and "application binary interface" (ABI) - concerns and assumptions

- Intended milestone is 4.1.0
- Will not be backported to 4.0.0 due to need for exporting new API functions on dynamic libraries

- However, a source patch kit will be provided for users of 4.0.x (and possibly 3.4.x) so that the new logging system implementation can be piggybacked into a user's repository. Users of this patch kit are responsible for recompiling and linking the patched library code and to make sure everything works as intended.

New logging macro

- Able to accept and use "tag information" in logging statements.
- Able to inject "tag information" in several ways.
- Logging macro shims are provided, and the C++ syntax shall be backward-compatible with all pre-existing logging statements throughout the library.
- Captures additional information such as "location of code" (source file, line number, function name, class name) as part of logging macro.
- Time-consuming operations (such as string formatting and stream insertion) are deferred until necessary.

Tag name (string identifiers for tags)

- Tags are identified by a string consisting of one or more "identifiers" ("name parts") joined by the period (".") as delimiter.
- Identifiers consist of alphanumeric characters and the underscore ("_").
- Identifiers are case-sensitive. However, it is recommended to use lower-case exclusively.
- OpenCV module names are used as top-level identifiers; additional identifiers will be implemented.
- Example: "imgcodecs.tiff.decoder".
- Example: "imgproc.remap"

Log filtering based on tags

- Each tag is associated with a runtime-configurable log filtering threshold.
- At runtime, each logging statement is accompanied with a tag and a log message level. The log message level is compared against the log filtering threshold associated with the tag.
- Some time-consuming operations, such as string formatting and stream insertion, are not performed if the log message is discarded due to filtering.
- The log filtering threshold associated with each tag can be programmatically configured via the name ("string identifier") of the tag.
- Configuration of log filtering for a multitude of tags can be scheduled at program launch time by reading from a launch configuration (e.g. from the environment or a config file).

Condensing log filtering specification for a multitude of tags into a single line

- The string format is modeled after Android LogCat, with some differences.
- It is also somewhat different from earlier releases of OpenCV.
- [[**TBD**]] Should we preserve backward behavioral compatibility at the consequence of diverging from Android LogCat syntax?

Wildcard matching of log tag names

- Wildcard matching of log tag names must align with the tag name part delimiter, the period (".").
- Example: "*.tiff", "imgcodecs.*"
- Users who need additional log filtering criteria shall do so in a logging backend of the user's choice, by making use of the backend extensibility of the new logging system. Backend extensibility will be discussed soon.

Backend extensibility

- After the logging statement has been filtered and accepted, the new logging system furnishes the following information to the backend of the library user's choice:
    - The message content
    - The name of the tag associated with the message
    - The log level associated with the message
    - Location-of-code (source file, line number, etc.)
    - System time and/or time since application launch
    - Current thread ID (could be system native thread ID or OpenCV's simplified thread ID)
    - [[**TBD**]]
- The library user can configure the backend programmatically at runtime.
- A built-in default backend is provided which prints logging statements to the console output stream.
- The built-in default backend allows for a few printout formatting options. These options are provided so that library users can hide some fields in the printout, in order to reduce the printout verbosity.

String formatters for OpenCV-defined and user-defined data types

- Currently, OpenCV provides string formatters for some OpenCV-defined types. These string formatters are used extensively inside OpenCV's unit testing modules, and they are designed to be interoperable with the "googletest" framework (also known as "gtest").
- As part of the logging revamp, these string formatters will be incorporated into the OpenCV library itself, so that they are available to both library users and to OpenCV's own unit testing modules.

End of document
