Implement hardware-accelerated video I/O in OpenCV

- Author: Vadim Pisarevsky
- Link: The feature request
- Status: **WIP** (e.g. PR adding bitrate control to VideoWriter)
- Platforms: **All**
- Complexity: several man-months

Introduction and Rationale

Currently, videio module in OpenCV has many backends, but only few of them are in reasonably good shape. Currently, the only more or less robust solution to read and write videofiles in OpenCV is via FFMPEG. It's LGPL library, which API changes on a regular basis. Besides, at least for now we do not enable hardware acceleration when we use FFMPEG (or any other backend), so reading or writing high-resolution video may slowdown the processing quite significantly.

The idea is to utilize the available hardware acceleration of video encoding/decoding, and preferably do it via default system API.

Proposed solution

- On Windows try to use MSMF (there is already some code in OpenCV)
- On Mac try to use AVFoundation (there is already some code in OpenCV)
- On Linux try to use gstreamer (ditto).
- As a backup option, try to adjust our FFMPEG wrapper to turn on hardware acceleration in it if possible.

Another important feature of the revised video I/O should be the ability to fine-tune video encoding parameters, most notably bitrate and the use of P- and B-frames (i.e. `all-intra`, `I+P` or `I+P+B` modes). In general, there should be the extendible list of properties passed to the video encoder.

Impact on existing code, compatibility

These are internal changes, so users should not notice any changes other than improved performance. Well, if there are some regression tests that decode video and depend on the exact pixel values in the decoded frames - such tests may start to fail, but that's not good way to write tests anyway.

Possible alternatives

Well, potentially our simplest codec, MJPEG codec, can be optimized for GPU using the OpenCL-accelerated jpeg encoder, but that's likely much slower and power-hungly method comparing to the fixed-function functionality in modern GPU hardware.

References

TBD