Rationale

OpenCV contains a lot of algorithms and provides APIs to different languages. In many cases the library works as black box and it's not easy to understand what went wrong or why particular function or module behaves against developer expectations. The page is collection if ideas and recipes how to debug OpenCV based program. The page will be very useful, if you want to submit a bug and need provide details about found issue.

General

Linkage

OpenCV library is part of popular Linux distributions and can be a part of other 3rd party libraries in your project. So you may have several instances of OpenCV and the project may work in wrong way. In case of issues, please ensure that the project starts with the same OpenCV instance you expected. Windows, Linux and other OSes provides tooling to review list of libraries linked with your dynamic library or app.

For Linux use `ldd`:

```
$ ~/Projects/OpenCV/opencv-master-build/bin$ ldd ./opencv_test_calib3d
        linux-vdso.so.1 (0x00007fffeebb9000)
        libopencv_calib3d.so.4.5 => /home/alexander/Projects/OpenCV/opencv-master-build/lib/
        libopencv_highgui.so.4.5 => /home/alexander/Projects/OpenCV/opencv-master-build/lib/
        libopencv_features2d.so.4.5 => /home/alexander/Projects/OpenCV/opencv-master-build/l
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f6b5aa48000)
        libopencv_imgcodecs.so.4.5 => /home/alexander/Projects/OpenCV/opencv-master-build/li
        libopencv_imgproc.so.4.5 => /home/alexander/Projects/OpenCV/opencv-master-build/lib/
        libopencv_core.so.4.5 => /home/alexander/Projects/OpenCV/opencv-master-build/lib/lib
        libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f6b574a9000)
```

For Windows use Dependency Walker or its analog.

OpenCV Build Options

OpenCV may be built with various options related to media i/o, UI, acceleration frameworks. The library presume original API for compatibility, but some functions may throw "not implemented" exception or behaves differently. The pre-built library can report its build options with cv::getBuildInformation call in C++, Python, Java and other supported languages.

DNN

Model diagnostic tool

OpenCV DNN module uses "Fail Fast" strategy loading model. It means that it stops model initialization with the first found problem. Model diagnostic tool can be useful to debug import-related problems of a network: it will try to import as much layers as possible in spite of errors. This process can show unsupported layers or which supported layers have unexpected parameters, giving an overview of amount of work it would take to get the network running. As of now, this tool supports ONNX(.onnx) and TensorFlow(.pb) models.

Beware, this tool skips series of assertions, it can lead to the expected application crash.

In order to build it, `-DBUILD_APPS=ON` cmake flag should be used. The target name is `opencv_model_diagnostics`.

Usage example:

```
./opencv_model_diagnostics -m=test_net.pb -c=test_net.pbtxt
```

Output example:

Unsupported layer

```
[ERROR:0] global opencv/modules/dnn/src/dnn.cpp (134) addMissing DNN: Node='model_28/tf.expa
[ERROR:0] global opencv/modules/dnn/src/dnn.cpp (3571) getLayerShapesRecursively OPENCV/DNN
[ERROR:0] global opencv/modules/dnn/src/dnn.cpp (3584) getLayerShapesRecursively Exception m
[ERROR:0] global opencv/modules/dnn/src/tensorflow/tf_importer.cpp (2915) parseNode DNN/TF:
```

Layer failure

```
[ERROR:0] global opencv/modules/dnn/src/tensorflow/tf_importer.cpp (2915) parseNode DNN/TF:
[ERROR:0] global opencv/modules/dnn/src/dnn.cpp (3571) getLayerShapesRecursively OPENCV/DNN
[ERROR:0] global opencv/modules/dnn/src/dnn.cpp (3584) getLayerShapesRecursively Exception m
[ERROR:0] global opencv/modules/dnn/src/tensorflow/tf_importer.cpp (2915) parseNode DNN/TF:
```

Optimizing the network to make it easier for OpenCV to import your model

General recommendation

Use any network visualizer(netron, tensorboard) to pinpoint the problematic layer - maybe you can just replace the layer or its attributes and it will work. Nevertheless, you should file an issue.

ONNX

Consider using ONNX-simplifier: there is a constant folding pass, which will remove the layers operating on constants, a fair amount of which aren't

supported, e.g. `Range`, `Size`, `Tile`, etc. You can skip certain optimizers if it introduces new fused layers which aren't supported. Also you can use `onnx` module directly to, for example, check if the model is valid: `onnx.checker.check_model(your_model)`, or to infer its shapes(if netron is having problems): `onnx.shape_inference.infer_shapes(your_model)`.

Tensorflow

Make sure your graph is frozen(in netron there shouldn't be any `Variable` nodes). You can use transform_graph tool to fold constants, for example: `transform_graph --in_graph=<in_model.pb> --out_graph=<out_model.pb> --inputs=<inputs> --outputs=<outputs> --transforms='remove_nodes(op=Identity) strip_unused_nodes fold_constants(ignore_errors=true`

Video I/O & Cameras

OpenCV library uses external video decoding APIs that depend on target platform. The same platform can provide several video i/o capabilities and OpenCV tries the first capable to play back video or open camera in priority order. Backend selection logic and internal state is hidden from the library users, but could be managed with run-time options through environment. `OPENCV_VIDEOIO_DEBUG=1` environment variable enables verbose logging for videoio module. `OPENCV_VIDEOIO_PRIORITY_<backend>=9999` option forces particular priority for particular backend. See more options and details in Query I/O API backends registry description of Doxygen documentation.

Python

Package Initialization

Since OpenCV 4.5.4 Python loader become mandatory part of OpenCV Python distribution. The loader allows to manage binary OpenCV instances and load complex bindings that contain both C++ and Python code (G-API, cv.Mat class, etc). The loader provides debug option to trace loaded binaries and made platform specific decisions. To get python loader trace, you need to add the following line of code BEFORE OpenCV import:

```
sys.OpenCV_LOADER_DEBUG = True
```

Types conversion

Python bindings for OpenCV are generated automatically and work as wrapper for C++ code. It means that C++ and Python calls to OpenCV with the same parameters should return the same results. Unexpected results in Python could be caused by incorrect or unexpected arrays conversion from Numpy array to OpenCV structures. cv.utils.dumpInputArray and

cv.utils.dumpInputArrayOfArrays can help to trace data conversion in auto-generated code.