Project description

OpenCV on Wheels

Unofficial pre-built OpenCV packages for Python.

Installation and Usage

1. If you have previous/other manually installed (= not installed via `pip`) version of OpenCV installed (e.g. cv2 module in the root of Python's site-packages), remove it before installation to avoid conflicts.
2. Select the correct package for your environment:

   There are four different packages and you should **select only one of them**. Do not install multiple different packages in the same enviroment.

   a. Packages for standard desktop environments (Windows, macOS, almost any GNU/Linux distribution)

   - run `pip install opencv-python` if you need only main modules
   - run `pip install opencv-contrib-python` if you need both main and contrib modules (check extra modules listing from OpenCV documentation)

   b. Packages for server (headless) environments

   These packages do not contain any GUI functionality. They are smaller and suitable for more restricted environments.

   - run `pip install opencv-python-headless` if you need only main modules
   - run `pip install opencv-contrib-python-headless` if you need both main and contrib modules (check extra modules listing from OpenCV documentation)
3. Import the package:

   `import cv2`

   All packages contain haarcascade files. `cv2.data.haarcascades` can be used as a shortcut to the data folder. For example:

   `cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")`
4. Read OpenCV documentation
5. Before opening a new issue, read the FAQ below and have a look at the other issues which are already open.

Frequently Asked Questions

Q: Do I need to install also OpenCV separately?

A: No, the packages are special wheel binary packages and they already contain statically built OpenCV binaries.

Q: Pip fails with `Could not find a version that satisfies the requirement ...`?

A: Most likely the issue is related to too old pip and can be fixed by running `pip install --upgrade pip`. Note that PyPI does not currently support ARM architecture so you can't install these packages for example on Raspberry Pi.

Q: Import fails on Windows: `ImportError: DLL load failed: The specified module could not be found.`?

A: If the import fails on Windows, make sure you have Visual C++ redistributable 2015 installed. If you are using older Windows version than Windows 10 and latest system updates are not installed, Universal C Runtime might be also required.

If the above does not help, check if you are using Anaconda. Old Anaconda versions have a bug which causes the error, see this issue for a manual fix.

Q: I have some other import errors?

A: Make sure you have removed old manual installations of OpenCV Python bindings (cv2.so or cv2.pyd in site-packages).

Documentation for opencv-python

Windows passing Linux macOS passing

The aim of this repository is to provide means to package each new OpenCV release for the most used Python versions and platforms.

Build process

The project is structured like a normal Python package with a standard `setup.py` file. The build process for a single entry in the build matrices is as follows (see for example `appveyor.yml` file):

1. Checkout repository and submodules
   - OpenCV is included as submodule and the version is updated manually by maintainers when a new OpenCV release has been made
   - Contrib modules are also included as a submodule
2. Find OpenCV version from the sources
3. Install dependencies (numpy)
4. Build OpenCV
   - tests are disabled, otherwise build time increases too much
   - there are 4 build matrix entries for each build combination: with and without contrib modules, with and without GUI (headless)
   - Linux builds run in manylinux Docker containers (CentOS 5)
5. Copy each `.pyd/.so` file to cv2 folder of this project and generate wheel
   - Linux and macOS wheels are checked with auditwheel and delocate
6. Install the generated wheel
7. Test that Python can import the library and run some sanity checks
8. Use twine to upload the generated wheel to PyPI (only in release builds)

The `cv2.pyd/.so` file is normally copied to site-packages. To avoid polluting the root folder this package wraps the statically built binary into cv2 package and `__init__.py` file in the package handles the import logic correctly.

Since all packages use the same `cv2` namespace explained above, uninstall the other package before switching for example from `opencv-python` to `opencv-contrib-python`.

Licensing

Opencv-python package (scripts in this repository) is available under MIT license.

OpenCV itself is available under 3-clause BSD License.

Third party package licenses are at LICENSE-3RD-PARTY.txt.

All wheels ship with FFmpeg licensed under the LGPLv2.1.

Linux and MacOS wheels ship with Qt 4.8.7 licensed under the LGPLv2.1.

Versioning

`find_version.py` script searches for the version information from OpenCV sources and appends also a revision number specific to this repository to the version string.

Releases

A release is made and uploaded to PyPI when a new tag is pushed to master branch. These tags differentiate packages (this repo might have modifications but OpenCV version stays same) and should be incremented sequentially. In practice, release version numbers look like this:

`cv_major.cv_minor.cv_revision.package_revision` e.g. `3.1.0.0`

Development builds

Every commit to the master branch of this repo will be built. Possible build artifacts use local version identifiers:

`cv_major.cv_minor.cv_revision+git_hash_of_this_repo` e.g. `3.1.0+14a8d39`

These artifacts can't be and will not be uploaded to PyPI.

Manylinux wheels

Linux wheels are built using manylinux. These wheels should work out of the box for most of the distros (which use GNU C standard library) out there since they are built against an old version of glibc.

The default `manylinux` images have been extended with some OpenCV dependencies. See Docker folder for more info.

Supported Python versions

Python 2.7 is the only supported version in 2.x series. Python 3.x releases follow Numpy releases. For example Python 3.3 is no longer supported by Numpy so support for it has been dropped in `opencv-python`, too.

Currently, builds for following Python versions are provided:

- 2.7

- 3.4
- 3.5
- 3.6