How to contribute to OpenCV

We suppose that you've seen the contribute page, and now, as an enthusiastic coder, want to contribute some code. For that purpose OpenCV project now has a mirror on the GitHub, to simplify everybody's life! All the bug fixes, new functionality, new tutorials etc. should be submitted via the GitHub's mechanism of pull requests.

If you are not familiar with the mechanism - do not worry, it's very simple. Keep reading.

Before you start contributing you should

- Make sure you agree to contribute your code under OpenCV (Apache 2.0) license.
- If you are submitting a new algorithm implementation, do a quick search over internet to see whether the algorithm is patented or not. **Note:** All new algorithms should go into opencv_contrib repository by default.
- If you are going to fix a bug, check that it's still exists. This can be done by building the latest 2.4/3.4 branch or the latest master branch, and make sure that the error is still reproducable there. We do not fix bugs that only affect deprecated versions like OpenCV 2.1 for example.
- Make sure that nobody beat you into fixing or reporting the issue by doing a search on the Github OpenCV issues page, and making sure that there isn't someone working on it. In the latter case you might provide support or suggestion in the issue or in the linked pull request.
- If you have a question about the software, then this is **NOT** the right place. You should open up a question at the OpenCV Q&A forum. In order to post a decent question from the start, feel free to read the official forum guidelines.

Before you open up anything on the OpenCV GitHub page, be sure that you are at the right place with your problem.

"Fork & Pull Request model" for code contribution

"Step-by-step" guides for popular systems and clients

- [[Windows_7_and_TortoiseGit_contribution_guide]]

The instruction in brief

1. Install [[Git]].
2. Register at GitHub. Create your fork of OpenCV repository https://github.com/opencv/opencv (see https://help.github.com/articles/fork-a-repo for details).

3. Choose a task for youself. It could be a bugfix or some new code.
4. Choose a base branch for your work. More details about branches and policies are here: [[Branches]]
5. Clone your fork to your computer. **You can install the default pre-commit hook by renaming `opencv/.git/hooks/pre-commit.sample` to `opencv/.git/hooks/pre-commit` - this will prevent you from committing whitespace errors.**
6. Create a new branch (with a meaningful name) from the base branch you chose.
7. Modify/add the code following our [[Coding_Style_Guide]].
8. Run testsuite locally:
    1. get the required sample data by cloning https://github.com/opencv/opencv_extra (choose corresponding branch)
    2. `export OPENCV_TEST_DATA_PATH=<path_to_opencv_extra>/testdata`
    3. execute each test binary from the build directory, e.g. `./bin/opencv_test_core`
9. When you are done, push your branch to your GitHub fork; then create a pull request from your branch to the base branch (see https://help.github.com/articles/using-pull-requests for details).

Making a good pull request

Following these guidelines will increase the likelihood of your pull request being accepted:

1. Before pushing your PR to the repository, make sure that it builds perfectly fine on your local system.
2. Add enough information, like a meaningful title, the reason why you made the commit and a link to the issue page if you opened one for this PR.
3. Scope your PR to one issue. Before submitting, make sure the diff contains no unrelated changes. If you want to cover more than one issue, submit your changes for each as separate pull requests.
4. If you have added new functionality, you should update/create the relevant documentation, as well as add tests for it to the testsuite.
5. Try not to include "oops" commits - ones that just fix an error in the previous commit. If you have those, then before submitting squash those fixes directly into the commits where they belong.
6. Make sure to choose the right base branch and to follow the [[Coding_Style_Guide]] for your code.
7. Make sure to add test for new functionality or test that reproduces fixed bug with related test data. Please do not add extra images or videos, if some of existing media files are suitable.
8. Make sure to add performance test, if you propose optimization or the patch could affect performance of major functionality. See HowToWritePerfTests page to implement good test and HowToUsePerfTests page to run performance tests properly.

Testing and merging pull requests

1. Your pull request will be automatically tested by OpenCV's buildbot (testing status can be checked here: http://pullrequest.opencv.org). If any builders have failed, you should fix the issue. To rerun the automatic builds just push changes to your branch on GitHub. *No need to close pull request and open a new one!*
2. Once all the builders are "green", one of OpenCV developers will review your code. Reviewer could ask you to modify your pull request. Please provide timely response for reviewers (within weeks, not months), otherwise you submission could be postponed or even rejected.

Here is the flow-chart of the process:

[[images/opencv-pr-flow.png]]

Happy End

1. As soon as the reviewer is fine with the pull request and BuildBot likes your code, the special approval comment :+1: or :shipit: is put, which signals OpenCV maintainers that they can merge your pull request.
2. The last, but not least. Make sure you got credits. We try to memorize all the contributions and list major ones in the [[ChangeLog]] and release announcements, but we may forget to do that, unintentionally. Please, do not hesitate to remind us, and we will update opencv.org and the ChangeLog accordingly.

FAQ

Q1. I was asked to change the target branch for my pull request, but why?

Please read this, read the "Choose a base branch..." section.

Q2. I was asked to change the target branch for my pull request, how can I do that?

You can use `git cherry-pick` command to move individual commits between branches. Here is the overall algorithm: you backup your current branch, re-create a new branch properly (from either *2.4*, *3.4* or *master*), cherry-pick necessary commits from backup branch onto re-create branch, change **base** branch of your pull request (via GitHub), finally push changes to your GitHub repository (with `--force/-f` option).

Q3. I was asked to remove whitespace issues, how can I do that?

Just resolve the issues, commit changes, and push them to the same branch you used to create a pull request.

Next time we suggest you to do the following. Install the default pre-commit hook by renaming `opencv/.git/hooks/pre-commit.sample` to `opencv/.git/hooks/pre-commit` - this will prevent you from committing whitespace errors.

```
cd opencv
mv .git/hooks/pre-commit.sample .git/hooks/pre-commit
```

Also, you can check for whitespace errors before the commit using this command:

```
git diff --check
```

Q4. How do I contribute to the documentation.

Please refer Writing documentation for OpenCV tutorial.