OpenCV Google Summer of Code (GSoC 2018)

Update. This time we take a break - OpenCV was not accepted into GSoC 2018. See you next year :)

This is an example of the guided filter in opencv_contrib

>>> Skip directly to the ideas list below <<<

General Information:

- Program Site for GSoC 2018
- Mailing list for OpenCV GSOC 2018: `opencv-gsoc-2018@googlegroups.com`
  - Group mailing list site
- IRC Channel: `#opencv` on freenode
- **Timelines**
  - Timeline for GSoC 2018

Important dates:

Date	Description	Comment
January 4 17:00 UTC	Mentoring organizations begin submitting applications to Google	:ok:
January 23 17:00 UTC	**Mentoring organization application deadline**	:heavy_check_mark:
January 23 - February 11	Google program administrators review organization applications	:skull:
~~February 12~~	~~List of accepted mentoring organizations published~~	
~~February 12 - March 12	~~Participants discuss ideas with mentoring organizations~~	
~~March 12 16:00 UTC	~~Student application period begins~~	
~~March 27 16:00 UTC	~~**Student application deadline**~~	
~~April 23 16:00 UTC	~~Accepted student proposals announced~~	
~~Community Bonding Period	~~Students get to know mentors, read documentation, get up to speed to begin working on their projects~~	
~~May 14 Coding begins!~~	~~Yeah!~~	

~~June 11 16.00 UTC~~	~~Mentors and students can begin submitting Phase 1 evaluations~~	
~~June 15 16.00 UTC~~	~~**Phase 1 Evaluation deadline**~~	
~~Work Period~~	~~Students work on their project with guidance from Mentors~~	
~~July 9 16.00 UTC~~	~~Mentors and students can begin submitting Phase 2 evaluations~~	
~~July 13 16.00 UTC~~	~~**Phase 2 Evaluation deadline**~~	
~~Work Period~~	~~Students continue working with guidance from Mentors~~	
~~August 6 - 14 16.00 UTC~~	~~Final week. Students submit their final work product and their final mentor evaluation~~	
~~August 14 - August 21 16.00 UTC~~	~~**Mentors submit final student evaluations**~~	
~~August 22~~	~~Final results of Google Summer of Code 2018 announced~~	
~~October~~	~~Mentor Summit at Google~~	

Times:

UTC to PDT (California uses PST in the winter (from Nov 1st) and PDT in the summer (from March 8)).

UTC time

UTC time converter

Resources:

- OpenCV official Site
- OpenCV wiki
- [[How to do a pull request/How to Contribute Code|How_to_contribute]]
- Source Code can be found at GitHub/opencv and GitHub/opencv_contrib
- [[Developer meeting notes|Meeting_notes]]

How you will be evaluated if you are an accepted student

- Student projects to be paid only if:
  - **Phase 1:**
    - You must generate a pull request
      - That builds
      - Has at least stubbed out functionality
      - With OpenCV appropriate Doxygen documentation
        - Includes What the function or net is, what the function or net is used for
      - Has at least stubbed out unit test
      - Has a stubbed out example/tutorial of use that builds
  - **Phase 2:**
    - You must generate a pull request
      - That builds

        - Has basic functionality
        - With OpenCV appropriate Doxygen documentation
            - Includes What the function or net is, what the function or net is used for
        - Has basic unit test
        - Has a tutorial of how to use the function or net and why you'd want to use it.
    - **End of summer:**
        - A full pull request
            - Full Doxygen documentation
            - A good unit test
            - Example of use/tutorial of the code or net
        - Create a (short!) Movie (preferably on Youtube, but any movie) that demonstrates your code
            - We use this to create an overall summary. Past years:
                - The 2015 Movie
                - The 2014 Movie
                - The 2013 Movie

For students interested in applying

1. You **must** already know how to program fluently in C++
    - Some projects may instead specifically require Python, javascript or Matlab skills
2. Ask to join the OpenCV GSoC Forum List
    - Discuss projects ideas below or your own ideas with OpenCV mentors on the list now and March 27.
    - Always title your proposal with what you want to do (example: *Implement Patch Match Stereo Algorithm*)
        - **NOTE:** The above is to discuss proposals with mentors. **BUT**, when the application period starts, you must still sign up with Google Summmer of Code and submit your proposal to the OpenCV organization. If not, you will not show up on the database where we can select you as a student.
3. In March 12, Go to the GSoC site and sign up to be a student with OpenCV
4. Post the project from below or your own agreed on project on the GSoC to the mailing list.
    - Include Name, google email, age
    - Include how you think you are qualified to accomplish this project (skills, courses, relevant background)
    - Include Country of origin, school you are enrolled in, Professor you work with (if any)
    - Include a projected timeline and milestones for the project
5. Once (and if!) OpenCV gets accepted as GSoC org this year, and we are told how many slots we will get **and** you've signed up for a project with us in March before the March 27 deadline: **Then:**
    - We will weigh the students and projects against the mentors we gather and the mentor's interests and choose which students/project to pursue.
    - Accepted students will be posted on the GSoC site in May (and we will notify the accepted students ourselves).
    - Students are paid over the summer by Google **IF** the mentor accepts the student's work. There are several milestone based go-nogo points.
6. We've had a lot of experience with this: Please only propose projects that you already know how to do.

- It is impossible for a mentor to train you in how to do the task while helping you do it. This **always** results in failure.
- Mentors exist to keep you on track, and ensure good code, but not to teach you a new area of vision or AI.
- This is also not a *research* period. If you are a top student with publications in top conferences from a top professor in a top University known to us, we may let you code your research area. This is very rare.
- Documentation, tests and tutorial are as vital as the code. We'd rather have rougher code then miss these items because if these are not in place, your code will be useless to everyone.

For computer vision professionals interested in mentoring

1. Contact us on the opencv-gsoc googlegroups mailing list above and ask to be a mentor (or we will ask you in some known cases)
2. If we accept you, we will post a request from the Google Summer of Code OpenCV project site asking you to join.
3. You must accept the request and **you are a mentor!**
4. You then:
    - Go to the opencv-gsoc googlegroups mailing list above and look through student project proposals, find a student and project you like and work with them to refine a realistic proposal that they can implement in a summer (you have to judge whether the student is capable - **absolutely no non-coders in the language you need, typically C++, accepted!** Summer is too short to learn to code and get something done).
        - you may also find (good) students and get them to apply, perhaps to your pet project idea
    - Once you find or create a project proposal that you want to mentor
        - several students might try for the same project
        - alternatively, you might have to convince a student to change projects to one you like or recruit an external student to join Google Summer of Code and apply to your project
    - But, always encourage students to officially apply through the Google Summer of Code site - it helps us and them.
5. We later get a slot allocation from Google, the administrators then *"spend"* the slots in order of priority influenced by whether there's a capable mentor or not for each topic.
6. Students must finally actually accept to do that project (some sign up for multiple organizations and then choose)
    - Sheesh!

If you are accepted as a mentor **and** you find a suitable student **and** we give you a slot **and** the student signs up for it, **then** you are an actual mentor! Otherwise you are **not a mentor** and have no other obligations.

- Thank you for trying.
- You may contact other mentors and co-mentor a project.

It sounds harder than it is.

You get paid a modest stipend over the summer to mentor, typically $500 minus an org fee of 6%.

Several mentors donate their salary, earning ever better positions in heaven when that comes.

2018 Accepted Projects and Mentors

Alphabetic by project title

Student	Title	Mentor(s)

2018 Project Ideas:

Have to make compliant with Defining a Project Ideas list from the Mentor Guide

Students may propose their own projects

- You must give us a clear summary of what you want to do and
- Proof that you already have the experience to do this project However, below are some of our priorities for this year Contact us and/or discuss ideas on the mailing list.

These are **not** in order of priority

1. Tutorials

We have so much good code in opencv_contrib as well as OpenCV that needs better documentation, tutorials, examples of use etc. Particularly:

- Our deep net module
  - Curating useful deepnet vision models from the net
  - Documenting the whole process of getting it to run in the [DNN module (https://github.com/garybradski/opencv_contrib/tree/master/modules/dnn)
  - Show how to use OpenCV with TensorFlow, Keras and/or Pytorch!
- Expand the python documentation and/or tutorials, see the Python idea below
- Make tutorials on more difficult topics such as
  - Calibration, stereo calibration, calibration of fisheye lens etc
  - Random Forests
  - Any topics from opencv_contrib
    - Particularly computational vision, new detector/descriptors, new filters, text recognition, depth sensing

2a. Deep Learning

2a. Improving OpenCV Deep learning functionality

OpenCV has a deep neural network module (DNN module). You are welcome to base your proposal on various options with it. We want better performance, more tests, tutorials, python/java wrappers etc.

Improving DNN module ideas:

- Advanced visualizing of deep learning models (with input-output blobs dimensions, layer types, connections and so on). It can be either based on third-party libraries solution or built on OpenCV's drawing functions own implementation.
- Implementing various network compression algorithms, such as quantization, factorization-based compression etc. That will likely involve both scripts for TF/Caffe and the corresponding modifications in DNN to support such compressed models.
- Enable supporting of the most popular deep learning architectures and classes of architectures that DNN does not support yet: DenseNet, GAN's etc. Implement missing layers, check output accuracy, write samples for online-available models.

2b. Curating and optimizing valuable published deep neural networks and turning them into compact models.

We'd like to learn **compact** (that can be fit on mobile, i.e. models that take a few megabytes of disk or memory space) models, see SqueezeNet for inspiration. Particularly for tasks such as:

- People recognition, instance finding, segmentation
- Background segmentation
- Object tracking
- Car instance, segmentation
- Bike instance, segmentation
- Animals, particularly dogs and cats outside and inside.
- Face tracking and/or finding face features
- Text finding, reading
- Floors, Walls, Ceilings, Windows, Doors, Tables, Chairs
- April tag detectors
- Corner point detectors
- Others....

2c. Reading ONNX

- Implement ONNX format parser. We already support Caffe 1 (with various extensions), TF, Torch and part of Darknet. Adding yet another parser is time-consuming but quite straight-forward thing.
- We have found a good reference starting point for this project here.
    - Basically, we need to parse this network structure and fill out the DNN data structures that come with a download of OpenCV. There are several people working on DNN right now, so one of them will mentor this through to finish.

3a. Add April Tags to OpenCV, show examples of calibration using them

April Tags

3b. Same as 3, but improve Aruco and Charuco tags.

- The code for detecting Aruco and Charuco is here

4. Working, well-done SLAM

We have tried to get in reliable building blocks for SLAM (Simultaneous Localization and Mapping) for years. We've made some progress and have good pieces, but it's not there yet.

- Please don't even think about doing this project unless you are already doing SLAM in your advanced PhD work. - This is a hard algorithm to get right and it needs lots of well thought out engineering and you just won't unless you are already working under a leading advisor in the field and/or are a published author in a top conference.
- We have two great mentors in this area, but the student has to have mature knowledge of the area and have mature programming skills in C++

5. Script to help people port OpenCV 1.0 (C) and 2.0 (C++) to OpenCV 3

- Write a script and/or use clang to help people move their OpenCV 1.0 or 2.0 code to OpenCV 3.

6. Improve and expand the python version of OpenCV.

The python wrapper to OpenCV is absolutely great thanks to several years of continual improvement in GSoC and continued efforts thereafter. This is one of the great success stories of OpenCV. But we want even more.

- Wrap more functions
- Add more documentation
- Add more tutorials and fairly complete applications
  - Calibration, face ID, feature detection and identification
  - One area is to show how to use OpenCV python with TensorFlow, Keras and/or Pytorch
- Add python wrappers to opencv_contrib functions

7. Extend translation of OpenCV into Java script. OpenCV.js

Last year, a concerted University + student effort enabled us to create OpenCV.js, OpenCV in javascript. This allows running computer vision natively inside a browser! This year, we want to improve the great effort that started last year. OpenCV.js

- Needs more documentation and tutorials
- Needs more working example code.
- See:
  - https://github.com/ucisysarch
  - Examples
    - http://ucisysarch.github.io/opencvjs/examples/img_proc.html
    - http://ucisysarch.github.io/opencvjs/examples/face_detect.html

8. HighGUI

HighGUI is OpenCV's simple graphical user interface including event triggering. It's cool, but it's old. It needs expanding.

- We, the admins, actually place a high priority of getting the simple basic stuff to work well, so we very much encourage application to this project
- HighGUI stands relatively alone and so can be easily changed without a deep knowledge of computer vision.
- Some of it's needs are:
  - Add simple buttons and sliders
  - It has an optional QT interface, but it involves some messy loading and linking into a very large external code base for what is relatively simple additional functionality.
    - Add the QT functionality (mainly zoom to pixel and snap shot) w/o the QT dependency;
      - one option is to use OpenGL and draw various controls manually.

9. Improve data labeling

Write better primitives for marking, segmenting, storing image (and possibly 3D!) data

But maybe just work with and improve the VATIC code (contact authors?)

- https://github.com/cvondrick/vatic
- http://web.mit.edu/vondrick/vatic/

10. OpenCV for interaction

- Eye tracking
- Hand or finger detection
- gesture recognition

- human skeleton based on 2d or 3d

11. OpenCV in Augmented and Virtual Reality

- As above, eye tracking, hand tracking, gesture recognition or human tracking
- Finding/segmenting planar surfaces
- Tracking different kind of markers
- Work better with the ARToolkit or Apple's tool kit ARKit

12. Improve 3D visualization for vis

The vis module is very useful, but it needs:

- Easy way to directly enter cv::Mat containing depth or disparity into a point cloud for vis
- Point cloud to ply converter
- Once in vis, automatic color coded surface normal visualization see here
    - And also here
- Tutorials showing how to use its features
    - Add onto here
- Or, maybe we just need to document/improve and bring out functionality as in docs here
- Test codes for functionality are here

13. Optimizing various functions

This is very general and always relevant topic. OpenCV is known for speed, but there is always room for improvement, especially in the experimental part (opencv_contrib). You are welcome to come up with concrete proposals on optimizing a particular functionality.

- Optimisation can be done using OpenCL, parallelisation, SSE/AVX or universal intrinsics.
    - It can also be "algorithmic" optimization, which, despite its name, means that the algorithm basically stays the same, but you find some ways to eliminate some unnecessary computations, unnecessary memory allocations etc.
    - For example, we would be interested in optimizing
        - image processing (imgproc, opencv_contrib/ximgproc),
        - TLD tracker (opencv_contrib/tracking),
        - computational photography (photo and opencv_contrib/xphoto),
        - dnn module (opencv_contrib/dnn),
        - feature detectors (features2d and opencv_contrib/xfeatures2d),
        - non-dnn-based object detection (opencv_contrib/dpm and opencv_contrib/xobjdetect).
        - SGBM stereo correspondence algorithm
    - You are welcome to propose for optimization some other functionality, though.

14. Halide language

OpenCV 4.0 is going to be written in Halide so that the compiler can be used to quickly optimize OpenCV on all the different hardware platforms that are proliferating.

- See the Halide project here
- We need:
    - Optimizing exists algorithms or some parts using Halide.
    - Extend implementations for GPU targets.
    - Deep learning layers and topologies.
    - Automatic domain-specific scheduling.

15. Better image & video encoding/decoding

- Bringing significant improvements into gstreamer backend of videoio
- Hardware-accelerated video codecs via libyami (https://github.com/01org/libyami)
- Replacing jasper with openjpeg to encode and decode Jpeg 2000 images
- Any other good suggestions/projects to reduce OpenCV dependency of ffmpeg

---

Staff

Admins

```
Gary Bradski
   CTO Arraiy, CEO/Co-founder OpenCV
   Founder of OpenCV
   Organized the Vision Team for Stanley the Robot that
      won the 2005 DARPA Grand Challenge and started
      the autonomous driving industry. Now in the
      Smithsonian Museum
   Startups (leading role):
      Video Surf -- Founding shares, sold to Microsoft 2010
      Industrial Perception -- Founder, CTO sold to Google 2013
      Magic Leap -- Founded Silicon Valley office, built to 110 employees
      Arraiy -- Founder CTO
      garybradski(+)gmail.com
Vadim Pisarevsky
    OpenCV team lead at Intel
    Co-founder of OpenCV
    Former principal engineer at Itseez.inc (sold to Intel, 2016)
    vadim.pisarevsky(+)gmail.com
```

Potential mentors

```
- Alexander Mordvintsev zzznah(+)gmail.com
- Sid Bao baoyingze(+)gmail.com
- Vincent Rabaud vincent.rabaud(+)gmail.com
- Vadim Pisarevsky vadim.pisarevsky(+)gmail.com
- Maksim Shabunin maksim.shabunin(+)gmail.com
- Adrian Kaehler therealadrian(+)gmail.com
- Terry Boult tboult(+)vast.uccs.edu
- spmallick spmallick(+)taaz.com
- Serge Belongie sjb344(+)cornell.edu
- Stefano s.fabri10(+)gmail.com
- Prasanna pras.bits(+)gmail.com
- Pablo Alcantarilla pablofdezalc(+)gmail.com
- Bence Magyar mw.mzperx(+)gmail.com
- Manuele manuele.tamburrano(+)gmail.com
- Grace Vesom grace.vesom(+)gmail.com
- Open Source Computer Vision Library (OpenCV) garybradski(+)gmail.com
- Douglas Lee dougabug(+)gmail.com
```

- Claudia Rapuano c.rapuano(+)gmail.com
- Antonella Cascitelli antonellacascitelli(+)gmail.com
- Anatoly Baksheev anatoly.baksheev(+)intel.com
- Alexander alexander.shishkov(+)intel.com
- Alexander Smorkalov alexander.smorkalov(+)intel.com
- Alexander Bovyrin alexander.bovyrin(+)intel.com