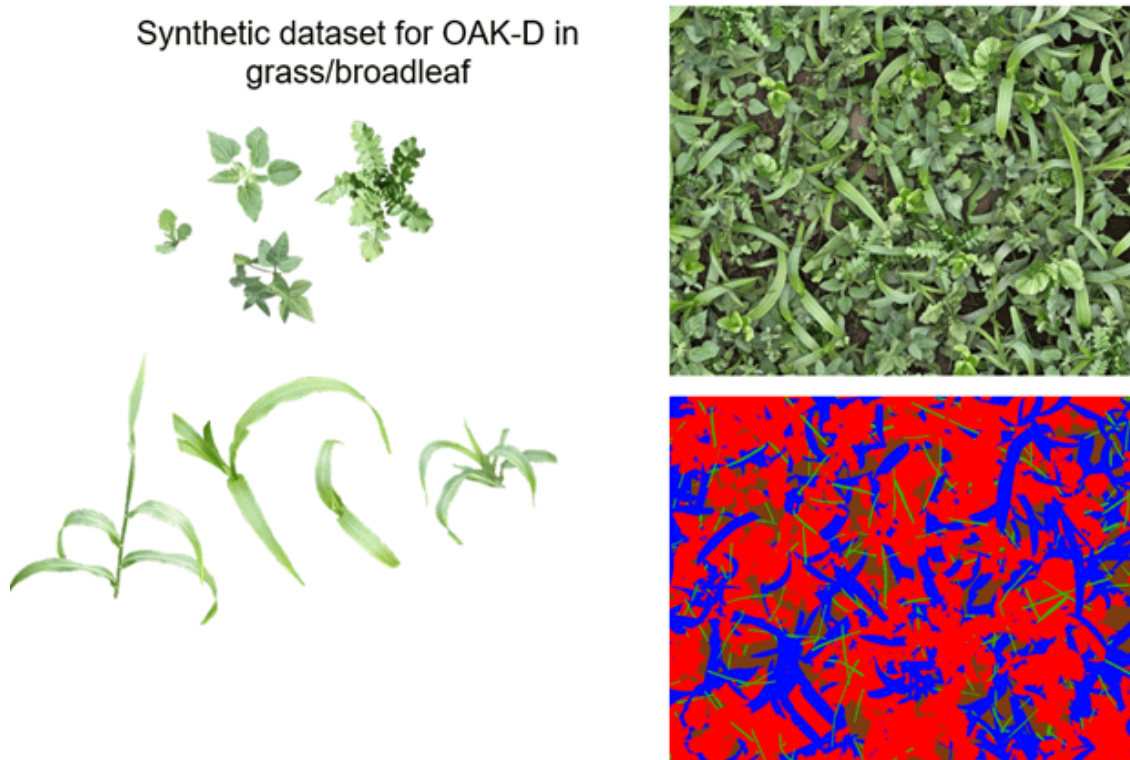


OpenCV Google Summer of Code 2021

Example use of computer vision:



Taken from [Weed control](#) by Paula Giraldo


- [\[\[Parent of this page|OpenCV_GSoC\]\]](#)
- [\[\[Last year's idea page|GSoC_2020\]\]](#)





[Student + Mentor + Project Discussion List](#)

[Student intern application process](#)

OpenCV Accepted Projects:

[Mentor only list](#)

-  TBD Spreadsheet of projects link

Student	Title	Mentors	Passed
Jebastin Nadar	8-bit quantization in DNN Module.	Vadim Pisarevsky	
Archit Rungta	OpenCV bindings for Julia	Tim Holy	
Hanxi Guo	Accelerate OpenCV.js DNN via WebNN	Ningxin Hu	
Zihao Mu	Loop closure algorithm	Rostislav Vasilikhin	

Dmitrii Klepikov	3D samples using OpenGL	Julia Bareeva	👍
Ricardo Antunes	3D samples using GLFW	Shiqi Yu	👍
Shivanshu Tyagi	OpenCV Audio IO module	Batanina Liubov	👍
Liutong HAN	Optimizing OpenCV DNN for RISC-V	Alexander Smorkalov	👍
Aditya Mulgundkar	nuget package for OpenCV	Miguel Lourenço	👍
Duong Dac	Improve the OpenCV Python	Vadim Levin	👍
Rohit Sutradhar	Tutorial for Universal Intrinsic and parallel_for_	Tuzov Vitaly	👍

Important dates:

Date (2021)	Description	Comment
January 29	Organization Applications Open	🕒
February 19	Organization Application Deadline	🕒
March 9	Organizations Announced	👍 <i>We are in!</i>
March 29 - April 13	Student Application Period	🕒
May 3	OpenCV Slot Request	🕒
April 13 - May 12	Slot Allocation	👍 12
May 13 - May 17	Project Selection	👍
May 17	Student Projects Announced	👍
May 17 - June 7	Community Bonding	👍
June 7 - July 12	Coding	👍
July 12 - 16	Evaluations	👍
July 12 - August 16	Coding	👍
August 16 - 23	Students Submit Code and Final Evaluations	👍
August 23 - 30	Mentors Submit Final Evaluations	👍
August 31	Results Announced	👍
September 15	Org Payments go out	🏠

[Timeline](#)

Times:

California switches PST->PDT (*Pacific Standard*->*Pacific Daylight*) Sun, Mar 14 2:00am

[UTC time](#)

[UTC time converter](#)

Resources:

- [GSoC Home Page](#)
- [OpenCV Project Ideas List](#)
- [OpenCV Home Site](#)
- [OpenCV Wiki](#)
- [OpenCV Forum, Questions and Answers](#)
- [[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]]
- [Mentor Only Mailing List](#)
- [Student+Mentor Mailing List](#)
- IRC Channel: `#opencv` on freenode

OpenCV Project Ideas List:

Mailing list to discuss: [opencv-gsoc-2021 mailing list](#)

Index	to	Ideas	Below
Fiducial Tags	Optimize OpenCV DNN for RISC-V	3D samples	Parallel for
Learn Fiducials	Microsoft Nuget	Speech recognition	Quantize & Prune for DNN
Bridge to Open3D	Loop closure algorithm for depth fusion	More flexible core and imgcodecs API	Ficus bindings
Point Cloud Fit	Demo for Android	Extend transpose operator	Type annotations for Python package
Accelerate OpenCV.js DNN via WebNN			
Idea Template			

All work is in C++ unless otherwise noted.

Ideas:

1. IDEA: Create robust visual fiducial tags

- **Description:** Robustly identify standard visual fiducials for April Tags and their orientation. Create a general method of turning a sufficiently textured planar tag into a fiducial that allows for position and ID.
- **Expected Outcomes:**
 - A set of classes and functions that detect the April Tags.
 - Documentation, the regression tests and the samples to describe and demonstrate the implemented functionality.
- **Resources:**

- [AprilTag software under BSD license](#). A note for the future mentors and admins who will integrate the patches. There are LGPL libraries for AprilTag in the net. Please, make sure that the code from the libraries is *not* used in OpenCV, because OpenCV license is not fully compatible with LGPL.
- **Skills Required:** Mastery plus experience coding in C++; training and using deep learning; experience with image processing.
- **Possible Mentors:** Gary Bradski, Gholamreza Amayeh
- **Difficulty:** Hard

2. IDEA: Optimize OpenCV DNN for RISC-V

- **Description:** Last year we brought initial version of wide universal intrinsics for OpenCV that hopefully provide some acceleration for traditional vision algorithms. However, for deep learning in OpenCV we often use specialized kernels in order to achieve the best possible performance.
- **Expected Outcomes:**
 - optimization of deep learning kernels (convolution and such) using RISC-V using native RISC-V vector intrinsics (V extension). See https://github.com/opencv/opencv/blob/master/modules/dnn/src/layers/layers_common.simd.hpp.
 - testing those kernels using QEMU on a subset of OpenCV DNN unit tests.
 - the detailed report that describes the used techniques.
- **Resources:**
 - [Optimizing Tensorflow Lite for RISC-V](#)
 - [OpenCV Wide Universal Intrinsics Guide](#)
 - [Implementation of wide universal intrinsics for various platforms](#)
- **Skills Required:** mastery plus experience coding in C++; basic skills of optimizing code using SIMD.
- **Possible Mentors:** Vadim Pisarevsky
- **Difficulty:** Hard

3. IDEA: 3D samples using OpenGL and GLFW

- **Description:** The newly created OpenCV 3D module includes a lot of interesting functionality that needs some cool samples for demonstration. In OpenCV we have `opencv_vis` module, but it has quite heavy `VTK` dependency that may be unavailable or difficult to install on some platforms. On the other hand, often the basic OpenGL API and some interactive framework on top of it (such as GLFW) are enough to demonstrate 3D objects and scenes, zoom/rotate them etc.
- **Expected Outcomes:**
 - Several samples that demonstrate OpenCV 3D module. Possible ideas: load point cloud from a file, visualize it, construct mesh, draw a mesh, fit planes/spheres/cylinders into the point cloud and visualize them, show the camera trajectory in a scene etc.
 - Preferably, the reusable parts of the samples should be represented as classes that we can further put into the new VTK-independent implementation of `opencv_vis` module.
- **Resources:**
 - [GLFW](#)
 - [OpenCV vis module tutorial](#)
 - [Essential matrix demo](#). For example, it can be modified to include some 3D visualization, as shown here: <https://youtu.be/ROGa2tutPHQ?t=539>
- **Skills Required:** good experience with OpenGL, basic to good knowledge of 3D vision algorithms, good C++/Python coding skills.
- **Possible Mentors:** Shiqi Yu, Vadim Pisarevsky
- **Difficulty:** Medium to Hard

4. **IDEA: Write a tutorial about using universal intrinsics and `cv::parallel_for_` for efficient cross-platform algorithm implementation**

- **Description:** Universal intrinsics is OpenCV way to write cross-platform and yet very efficient code on a variety of platforms. The technique is widely used inside OpenCV, but it's not well-known to many OpenCV users, including contributors, who are supposed to provide high-quality fast code into OpenCV. This tutorial should fill this important missing part of the documentation.
- **Expected Outcomes:**
 - A tutorial + source code that will explain how to use the universal intrinsics. It should also have an overview of what are intrinsics, what are vector (SIMD) instructions. Some tricks (e.g. how to process the "tails" of image rows that do not fit SIMD register), useful intrinsics should be covered as well.
- **Resources:**
 - [OpenCV Wide Universal Intrinsics Guide](#)
- **Skills Required:** good experience in C++, some experience with code optimization. Very good English.
- **Possible Mentors:** Vitaly Tuzov, Shiqi Yu
- **Difficulty:** Medium

5. **IDEA: Learn visual fiducial target**

- **Description:** OpenCV has several working trackers, based on classical image processing approaches, but we want to add something new, more accurate and fast, based on DNN. We want to jointly learn a fiducial image that allows for accurate localization and orientation of a point across scales
- **Expected Outcomes:**
 1. Generate a fiducial image that allows finding it across scales while localizing the center and determining its orientation
 2. A deepnet that robustly finds the fiducial and accurately determines its orientation and center across scales
- **Skills Required:** Expertise in training deep learning networks training for computer vision problems.
- **Mentors:** 🇵🇹
- **Difficulty:** Very Hard

6. **IDEA: Create nuget package for OpenCV and OpenCV contrib.**

- **Description:** Nuget is the standard Microsoft package manager. If done properly, it can become the most convenient way to install OpenCV on Windows. We can also create opencv_contrib nuget package, and thus provide a convenient way for OpenCV users to install experimental OpenCV functionality on Windows.
- **Expected Outcomes:**
 - scripts to automatically generate 2 nuget packages: for OpenCV and OpenCV-contrib. Probably, even finer-grain nuget packages can be created, e.g. one for the main OpenCV and then one per each opencv_contrib module. But 2 will be good enough to start with.
 - publish OpenCV nuget packages at [NuGet Gallery](#).
- **Resources:**
 - [OpenCV 3.x nuget package](#)
 - [An Introduction to NuGet from Microsoft](#)
- **Skills Required:** good expertise in C++ and Windows development.
- **Possible Mentors:** Satya Mallick
- **Difficulty:** Medium

7. **IDEA: Quantization and pruning functionality for OpenCV DNN module**

- **Description:** "Learning compact models for object detection" added SqueezeDet and SqueezeNet models to OpenCV repository. But OpenCV DNN module is still lacks high-level quantization and pruning functionality. Project also includes implementation of re-training (fine-tuning) of quantized and/or pruned models.
- **Expected Outcomes:**
 - 8-bit and 16-bit quantization implementation
 - Iterative pruning with controlled by target sparsity
 - Provide examples of quantized and pruned network and fine-tune it (base is AlexNet or other classification architecture)
 - Provide evaluation of original network and compressed one (accuracy and speed)
- **Additional goals:**
 - N-bit quantization
 - Different operation types as minifloat, dynamic fixed point etc.
 - Further model compression encoding/decoding with Huffman coding
- **Resources:** * <https://arxiv.org/pdf/1806.08342.pdf> * <https://arxiv.org/pdf/1611.06440.pdf>
- **Skills Required:** Very good C++ coding skill, experience in Deep Learning area more than just tutorial, basic Computer Vision knowledge, at least some knowledge about quantization and pruning
- **Possible Mentors:** Tyan Vladimir
- **Difficulty:** Hard

8. **IDEA: Speech recognition samples using Audio IO module**

- **Description:** The newly created OpenCV Audio IO module allows to extend the functionality of DNN module. We want to add networks that work with audio, to start with speech recognition sample. We suggest paying attention to Jasper model.
- **Expected Outcomes:**
 - Integrate required DNN models into OpenCV (may require implementing some additional network layers).
 - Create a speech recognition samples that demonstrate OpenCV DNN and Audio IO modules.
- **Resources:**
 - [paper](#)
 - [repo](#)
- **Skills Required:** training and using deep learning; basic to good knowledge of NLP; good C++/Python coding skills.
- **Possible Mentors:** Batanina Liubov
- **Difficulty:** Medium

9. **IDEA: Bridge to Open3D**

- **Description:** [Open3D](#) is a modern library containing a lot of pipeline-ready algorithms including all parts of KinectFusion, a lot of samples, tutorials and so on. It's very simple to build a SLAM system based on that. Works quite fast for CPU-only code but lacks acceleration like SIMD or OpenCL, as a result it's not suitable for real time tasks. On the other side, 3d module has KinectFusion which

works real time with the help of OpenCL but lacks flexibility of Open3D. Can these two ones be joined together?

- **Expected outcomes:**
 - Converter functions to/from Open3D's data types
 - A tutorial on how to make an app with both OpenCV and Open3D
 - A sample application in Python (1st priority) and/or C++ (2nd priority). Priority can be discussed.
- **Skills required:** good C++/Python coding skills, medium knowledge of 3D vision algorithms
- **Possible Mentors:** Rostislav Vasilikhin
- **Difficulty:** Medium

10. **IDEA: Loop closure algorithm for depth fusion**

- **Description:** The current version of Large Scale Depth Fusion copes with accumulating pose drift by breaking a scene into parts (submaps), estimating local alignments between them and then performing global optimization on pose graph of the submaps. To succeed, such a pose graph should contain edges (local alignments) not only between consequently shot submaps but also between submaps covering the same places in a scene. This is a so-called loop closure problem: an algorithm that takes two shots of a scene (made by a camera or reconstructed from previous data) and tells if they represent the same place. Current Large Scale Depth Fusion pipeline lacks such an algorithm.
- **Expected Outcomes:**
 - A loop closure algorithm for large scale depth fusion pipeline. Possible ways to implement:
 - Feature extraction from depth frames + RANSAC / BoW-like to detect known locations
 - Deep neural network that generates embeddings for depth frames + a code that runs local alignment when a close match encountered
 - Anything else based on existing articles, or, better existing software
- **Resources:**
 - Q.-Y. Zhou, J. Park, and V. Koltun, Fast Global Registration, ECCV, 2016
 - R. Rusu, N. Blodow, and M. Beetz, Fast Point Feature Histograms (FPFH) for 3D registration, ICRA, 2009
- **Skills Required:** good C++ coding skills, medium knowledge of 3D reconstruction algorithms, any experience in model fitting / RANSAC / deep learning is appreciated
- **Possible Mentors:** Rostislav Vasilikhin
- **Difficulty:** Hard

11. **IDEA: Ficus Bindings**

- **Description:** Create bindings for the newly created [Ficus language](#).
- **Expected Outcomes:**
 - A set of ficus modules that would wrap some essential OpenCV functionality, at least enough to run a deep-learning based object detection demo: a subset of core, imgproc, imgcodecs, videoio and dnn modules.
 - The respective short tutorial and 1-2 examples.
- **Skills Required:** Mastery experience coding in C/C++ or Python.
- **Possible Mentors:** Vadim Pisarevsky
- **Difficulty:** Hard

12. **IDEA: Finalize Plane/Sphere/Cylinder Fitting Code**

- **Description:** Last summer during [GSoC 2020](#) we've implemented [draft versions](#) of point cloud fitting algorithms. The goal is to finalize this code, probably optimize it, polish the examples and write the respective tutorial. Note that 3D module has been moved to the main repository, so this functionality is going to become a part of the main, stable OpenCV: <https://github.com/opencv/opencv>. If you are good at software engineering and can turn a good code into an excellent one, you are welcome!
- **Expected Outcomes:**
 - Add shape fitting code into `opencv/3d` module. Regression test should be added, based on some real or synthetic data.
 - At least 1 example that will load a point cloud, fit shapes into it and use OpenGL to visualize it. If the code works fast enough, it can probably be a live "augmented reality" demo with "live" planes fit into RGBD data captured from a depth sensor, e.g. Kinect, Astra etc.
 - (Optional) Support for non-canonical clusters (based on their compactness, for example) would be nice to have as well.
- **Resources:**
 - [A part of Point Cloud Library, which we can probably use as a reference implementation](#)
 - Existing pull request with draft implementation: https://github.com/opencv/opencv_contrib/pull/2584
- **Skills Required:** mastery plus experience coding in C++; basic knowledge of 3D point cloud processing principles, model fitting, RANSAC.
- **Possible Mentors:** Rostislav Vasilikhin, Vadim Pisarevsky
- **Difficulty:** Medium to Hard

13. IDEA: Demo for Android

- **Description:** Computer vision on a mobile phone has been also a hot area. Many years ago we already similar projects for Android and iOS. Now it's time to do this project again. Modern Android + OpenCV + its deep learning module + camera (maybe even with extra depth sensor on phones that have it). It can be a really cool project that will show your skills that that will be extremely helpful for many OpenCV users.
- **Expected Outcomes:**
 - Demo app for Android with all the source code available, no binary blobs or proprietary components.
 - The app should preferably be native (in C++).
 - The app can use camera via native Android API or via OpenCV API. The latter is preferable, not not necessary.
 - The app must use OpenCV DNN module. It should run some deep net using OpenCV and visualize the results (e.g. draw rectangles around found objects, the fancier visualization the better).
 - The app should work in realtime or a reasonably fast phone. If it's slow, it's better to choose some lighter net or run it on a lower resolution.
 - Preferably the app should not be tightly bound to a particular topology. For example, if it's object detection demo, it should be possible to change the topology to another one, not via UI, just replace some file and change it's name in config file or source code.
 - There should be a short tutorial (a markdown file with the key code fragments and some screenshot on how to build it and run).
 - There should be some efforts applied (see <https://github.com/opencv/opencv/wiki/Compact-build-advice>) to make the application compact. That was a separate request from various OpenCV users, and a compact mobile app would be the best example how to do that.

- **Skills Required:** Mastery experience coding in C/C++ or Java, practical experience with developing apps for Android. At least basic understanding of computer vision and deep learning technologies, enough to run a deep net and make it run at realtime speed.
- **Possible Mentors:** Vadim Pisarevsky
- **Difficulty:** Medium to Hard

14. **IDEA: Extend transpose operator for MatND, Mat with channels and extend API**

- **Description:** In image preprocessing and postprocessing tasks for neural networks, the transpose operation is often used, for example, from the [NumPy library](#): `img.transpose(1, 2, 0)`. Unfortunately, when we work in C++, we have no way to repeat this operation using OpenCV in one step. As a workaround, reshaping and memory copying are used, which reduces performance a lot. [See also](#).
- **Expected Outcomes:**
 - implement transpose operator for MatND and GpuMat (reuse existing algorithm from `PermuteLayer` Class if possible or write your own if not)
- **Resources:** * https://docs.opencv.org/master/d2/d3c/classcv_1_1dnn_1_1PermuteLayer.html
- **Skills Required:** C++, CUDA, knowledge of computer vision algorithms
- **Possible Mentors:** Julia Bareeva
- **Difficulty:** Easy

15. **IDEA: Improve Python package with type annotations and type checking**

- **Description:** OpenCV Python interfaces are moving towards the type safety (e.g. strict type checking in argument parsing, overload resolution) and new functionality that is related only to Python might be on the way. Type annotations added with the [PEP 484](#) help to write code that is easier to understand and maintain, moreover IDEs can provide accurate code completion which boosts developer productivity. Type checker like **mypy** can find problems before running the code and catching runtime errors e.g. forgetting to handle a `None` value. Corresponding request from the community: [Python typing stub #14590](#)
- **Expected Outcomes:**
 - Use type information from the header parser provided as `ArgTypeInfo` structure to generate Python typings stubs using [PEP 3107 -- Function Annotations syntax](#). (see [numpy.pyi files](#) as example)
 - Update OpenCV Python package build procedure to pack generated stubs
 - Add type checking mechanism as an optional step (e.g. weekly) of the CI to check Python samples and tests against generated Python package flavored with type hints. This helps keep code in samples and tests always up to date.
- **Resources**
 - [PEP 483 -- The Theory of Type Hints](#)
 - [Pyre is a performant type checker for Python by Facebook](#)
 - [Mypy is a static type checker for Python](#)
- **Skills Required:** Good Python knowledge, basic C++ knowledge
- **Possible Mentors:** Vadim Levin
- **Difficulty:** Medium

16. **IDEA: Accelerate OpenCV.js DNN via WebNN**

- **Description:** OpenCV.js exposes JavaScript API of dnn module that allows web apps to do deep learning model inference in web browsers. This capability enables web developers to create intelligent usages, like image classification, object detection, segmentation and style transfer, with rich web contents, like images, videos and camera streams. Although OpenCV.js dnn module is using WebAssembly with multi-threading and SIMD128 optimization (the GSoC 2017 project), the OpenCV.js dnn module is still 4-8X slower than the native OpenCV. This is because native OpenCV

can leverage more advanced hardware features and even ML-specific accelerators via native instructions and ML APIs, such as AVX2, cuDNN, OpenVINO etc.,. Web Neural Network API (WebNN) is an emerging web standard that allows web apps to access the AI hardware accelerations. WebNN current spec editors are Intel and Microsoft with strong support from Google among others. Our early empirical results indicate that WebNN gets near-native performance on a variety of Intel-based platforms through the open source module oneDNN. We propose to start optimizing OpenCV.js dnn module with WebNN and demonstrate the performance gain.

- **Expected Outcomes:**

- Create the WebNN backend prototype for OpenCV.js dnn module in C++
- Create the WebNN implementation of selected compute intensive operations, like convolution and matrix multiplication in C++
- Compile the OpenCV.js dnn WebNN backend implementation into WebAssembly
- Create the performance test of OpenCV.js dnn module, collect and analyze the performance numbers of different implementations including WebAssembly, WebNN and native.

- **Resources:**

- OpenCV.js module: <https://github.com/opencv/opencv/tree/master/modules/js>
- OpenCV.js dnn tutorials: https://docs.opencv.org/master/d0/db7/tutorial_js_table_of_contents_dnn.html
- WebNN API spec: <https://webmachinelearning.github.io/webnn/>
- WebNN samples: <https://github.com/webmachinelearning/webnn-samples>
- WebNN native implementation: <https://github.com/webmachinelearning/webnn-native>
- WebNN talk @ W3C WebML workshop: https://www.w3.org/2020/06/machine-learning-workshop/talks/access_purpose_built_ml_hardware_with_web_neural_network_api.html

- **Skills Required:** C++, JavaScript, WebAssembly, Deep Learning

- **Possible Mentors:** Ningxin Hu

- **Difficulty:** Medium to Hard

Idea Template:

```
1. ##### _IDEA:_ <Descriptive Title>
* ***Description:*** 3-7 sentences describing the task
* ***Expected Outcomes:***
  * < Short bullet list describing what is to be accomplished >
  * <i.e. create a new module called "bla bla">
  * < Has method to accomplish X >
  * <...>
* ***Resources:***
  * [For example a paper citation] (https://arxiv.org/pdf/1802.08091.pdf)
  * [For example an existing feature request]
    (https://github.com/opencv/opencv/issues/11013)
  * [Possibly an existing related module]
    (https://github.com/opencv/opencv_contrib/tree/master/modules/optflow) that includes
    some new optical flow algorithms.
  * ***Skills Required:*** < for example mastery plus experience coding in C++,
    college course work in vision that covers optical flow, python. Best if you have also
    worked with deep neural networks. >
```

```
* ***Possible Mentors:*** < your name goes here >
* ***Difficulty:*** <Easy, Medium, Hard>
```

• All Ideas Above

1. Have these Additional Expected Outcomes:

- Use the [OpenCV How to Contribute](#) and [Aruco module in opencv contrib](#) as a guide.
 - Add unit tests [described here](#), see also the [Aruco test example](#)
 - Add a tutorial, and sample code
 - see the [Aruco tutorials](#) and how they [look on the web](#).
 - See the [Aruco samples](#)
 - Make a short video showing off your algorithm and post it to Youtube. [Here's an Example](#).
-

Students

How to Apply

[Applicaiton process is here](#)

Pre-application form

<https://forms.gle/U1j4zmXY3gkpgnhE8>

Feel free to submit it. It's the optional form. In order to apply for GSoC 2021, you need to use the official site:

<http://summerofcode.withgoogle.com/get-started/>

But the pre-application form takes just a couple of minutes to complete and basically asks you to submit all the information that we find crucial to consider one's application seriously: 1. resume, 2. link to the source code, 3. application (pre-application form asks for the application draft). If you submit earlier, we will have more time to review your application and can provide some feedback so that you can improve your application before the final deadline.

How students will be evaluated once working:

- Student projects to be paid only if:
 - **Phase 1:**
 - You must generate a pull request
 - That builds
 - Has at least stubbed out (*place holder functions such as just displaying an image*) functionality
 - With OpenCV appropriate Doxygen documentation ([example tutorial](#))
 - 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
 - See [the contribution guild](#)
 - and [the coding style guild](#)
 - the [line descriptor](#) is a good example of student code

- **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 2020 Movie](#)
 - [The 2015 Movie](#)
 - [The 2014 Movie](#)
 - [The 2013 Movie](#)

Mentors:

1. Contact us by March 15th 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!**
- You will also need to get on:
 - [The Mentor Only Mailing List](#)
 - [The Student+Mentor Mailing List](#)
4. You then:
 - Look through the ideas above, choose one you'd like to mentor or create your own and post it for discussion on the mentor list.
 - Go to the opencv-gsoc googlegroups mailing list above and look through student project proposals and discussions. Discuss the ideas you've chosen.
 - Find likely students, ask them to apply to your project(s)
 - You will get a list of students who have applied to your project. Go through them and select a student or rejecting them all if none suits and joining to co-mentor or to quit this year are acceptable outcomes.
 - Make sure your students officially apply through the [Google Summer of Code site](#) prior to the deadline as indicate by the Student Application Period in the [time line](#)

5. Then, when we get a slot allocation from Google, the administrators "*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)
7. Get to work!

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.

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.

Potential Mentors List:

Ankit Sachan
Clément Pinard
Davis King
Dmitry Kurtaev
Dmitry Matveev
Edgar Riba
Gholamreza Amayeh
Grace Vesom
Jiri Hörner
João Cartucho
Justin Shenk
Michael Tetelman
Ningxin Hu
Rostislav Vasilikhin
Satya Mallick
Stefano Fabri
Steven Puttemans
Sunita Nayak
Vikas Gupta
Vincent Rabaud
Vitaly Tuzov
Vladimir Tyan
Yida Wang

Admins

Gary Bradski
Vadim Pisarevsky
Shiqi Yu

GSoC Org Application Answers

[Answers from our OpenCV GSoC application](#)