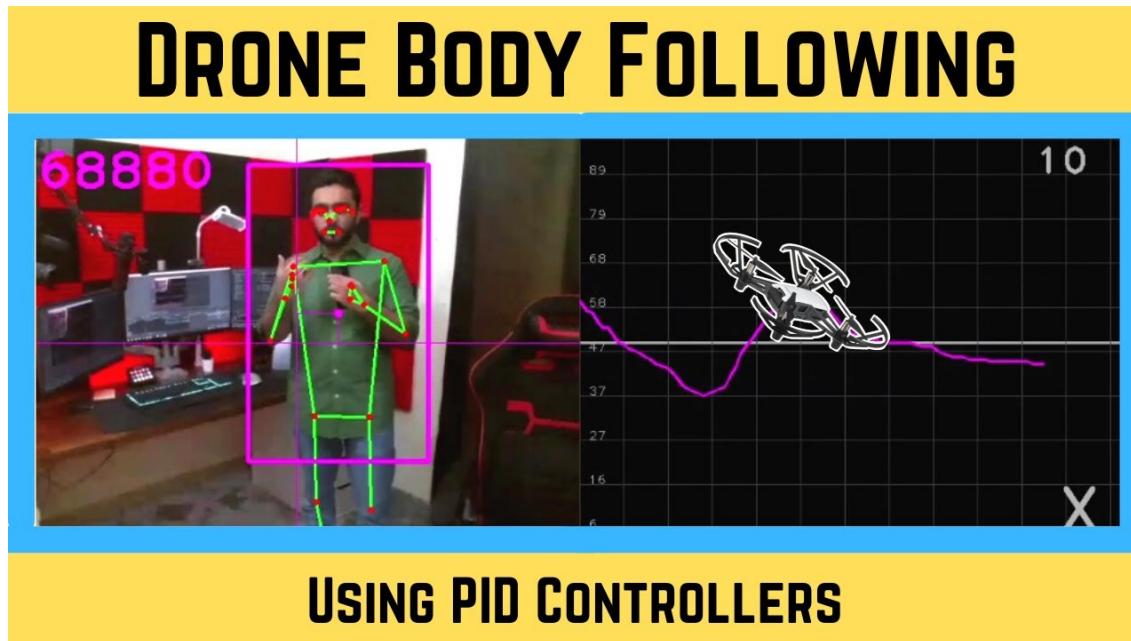


OpenCV Google Summer of Code 2022

[Jump to Project Idea List](#)

Example use of computer vision:



(image from: <https://morioh.com/p/cd512f2bd336>)


- [\[\[Parent of this page|OpenCV_GSoC\]\]](#)
- [\[\[Last year's idea page|GSoC_2021\]\]](#)
- [Jump to Project Idea List](#)

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

[GSoC 2022 Homepage](#)



OpenCV Accepted Projects:

[Mentor only list](#)

-  TBD Spreadsheet of projects link

| Contributor | Title | Mentors | Passed |
|-------------|-------|---------|--------|
|-------------|-------|---------|--------|

Important dates:

| Date (2022) | Description | Comment |
|-------------|-----------------------------------|---|
| February 7 | Organization Applications Open |  |
| February 21 | Organization Application Deadline |  |

| | | |
|-----------------------------|---|-----|
| March 7 | Organizations Announced | 📢 ! |
| April 4 - April 19 | Contributors' Application Period | 👤 |
| May 12 | OpenCV Slot Request | |
| May 20 | Accepted GSoC contributor projects announced | |
| May 20 - June 12 | Community Bonding | |
| June 13 - July 25 | Coding (Phase 1) | |
| July 25 | Phase 1 Evaluations start | |
| July 29 | Phase 1 Evaluations deadline | |
| July 25 - September 4 | Coding (Phase 2) | |
| September 5 - September 12 | Contributors Submit Code and Final Evaluations | |
| September 12 - September 19 | Mentors Submit Final Evaluations | |
| September 20 | Initial Results Announced | |
| September 12 - November 13 | Extended coding phase | |
| November 21 | Deadline for GSoC contributors with extended timeline to submit their work | |
| November 28 | Deadline for Mentors to submit evaluations for GSoC contributors with extended timeline | |

[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 list](#)
- [Contributor+Mentor Mailing List](#)

- IRC Channel: `#opencv` on freenode

OpenCV Project Ideas List:

Mailing list to discuss: [opencv-gsoc-2022 mailing list](#)

Index to Ideas Below

1. [Audiovisual Speech Recognition](#)
2. [FP16 & BF16 for DNN](#)
3. [Improved imgcodecs](#)
4. [Ficus bindings](#) |
5. [Point Cloud Compression](#)
6. [Simple Triangle Rendering](#)
7. [Demo for Android](#)
8. [ONNX/numpy operators](#)
9. [Image augmentation for DNN](#)
10. [Multi-task CV models](#)
11. [Lightweight object detection models](#)
12. [Realtime object tracking models](#)
13. [Improved ArUco module](#)
14. [Lightweight OCR models](#)
15. [RISC-V Optimizations](#)
16. [Enable G-API in OpenCV.js](#)
17. [First Robotics with OAK-D Camera](#)
18. [Multi-grid multi-camera calibration](#)

[Idea Template](#)

All work is in C++ unless otherwise noted.

Ideas:

1. **IDEA: Audiovisual speech recognition demo**

- **Description:** Last summer we brought support for audio I/O and some speech recognition models into OpenCV. This time we want to add more speech recognition examples. One particularly interesting demo would be to recognize speech using joint video+audio input, i.e. use video feed as extra data to improve speech recognition. We would like to support [the following model](#)
- **Expected Outcomes:**
 - Demo that reads video + audio using OpenCV API, analyzes speech and prints the recognized text in realtime in console or separate UI window.
- **Resources:**
 - [Audiovisual Speech Recognition & Lipreading Model](#)
- **Skills Required:** training and using deep learning; basic to good knowledge of NLP; good C++/Python coding skills.
- **Possible Mentors:** Batanina Liubov
- **Duration:** 175 hours

2. **IDEA: FP16 Compute Paths in OpenCV DNN**

- **Description:** Some modern CPUs include hardware support for FP16 arithmetics, which can be used to accelerate various algorithms including computational photography and deep learning inference. OpenCV includes basic support for FP16 data format, but there is no any FP16 compute paths in OpenCV deep learning module. It's suggested to add some basic FP16 kernels to accelerate OpenCV DNN, in particular, convolutions (depth-wise, 1x1, 3x3 ...) and transposed convolutions, max pooling, element-wise operations. The graph engine in OpenCV DNN should be modified to insert FP16<=>FP32 conversions when needed. FP16 compute paths can be utilized either automatically, when the loaded model contains FP16 or INT8 weights, or explicitly, when user sets FP16 precision. Special measures should be taken to preserve accuracy of the original FP32 models. *It's fine to re-use existing kernels from other deep learning engines as long as they have proper license (BSD, MIT, Apache 2).*
- **Expected Outcomes:**
 - Series of patches for OpenCV DNN that add support for FP16 into the inference pipeline, the actual operations and the graph engine.
 - Updated OpenCV DNN units tests to test the performance and accuracy of FP16 compute paths
- **Resources:**
 - [Tencent NCNN with FP16 & BF16 support](#)
 - [Alibaba MNN with FP16 support](#)
 - [XNNPack with lots of FP16 kernels](#)
- **Skills Required:** good software optimization skills, fluent C++ and ARM Assembly, basic understanding of deep learning inference engines
- **Possible Mentors:** Vadim Pisarevsky
- **Duration:** 175 hours

3. IDEA: Improved imgcodecs

- **Description:** It's not a single big project but rather a series of tasks under the "image codecs" umbrella. The goal is to make a series of improvements in the existing opencv_imgcodecs module, including, but not limited to, the following items (please, feel free to propose your ideas):
 - replace libpng with much more compact and equally efficient [libspng](#).
 - probably, insert some bits of [pngcrush](#) into png encoder to achieve better compression
 - enable inline assembly in libjpeg-turbo in order to boost performance.
 - provide optional bit-exactness tests for OpenCV jpeg decoder to make sure that we can read jpeg images and get the same results on each platform. Those tests should only be invoked when OpenCV is built with some known version of libjpeg.
 - provide extra API to read image header separately from the image data (probably, it should be a callback or something like that provided, or there should be some optional limits for image resolution).
 - finalize multi-page image decoder with convenient C++ and Python API: <https://github.com/opencv/opencv/pull/17753>
- **Expected Outcomes:**
 - Series of patches for OpenCV imgcodecs
- **Resources:**
 - TBD
- **Skills Required:** practical expertise in C++ and image codecs.
- **Possible Mentors:** Vadim Pisarevsky, Suleyman Turkmen
- **Duration:** 175 hours

4. IDEA: Ficus Bindings

- **Description:** Extend OpenCV bindings for Ficus programming language to cover more modules. Preferably create a script to generate bindings automatically. Add more examples to use OpenCV from Ficus.
- **Expected Outcomes:**
 - Patches for ficus OpenCV bindings to bring in more functionality
 - Several OpenCV C++/Python examples converted to Ficus
 - Documentation (probably, a chapter in the Ficus tutorial)
- **Resources:**
 - [Ficus at Github](#); inside the doc directory you may find the tutorial.
 - [Current Ficus OpenCV bindings](#)
 - [Object detection example in Ficus](#) that contains build instructions
- **Skills Required:** Mastery experience coding in C/C++, basic knowledge of functional languages, such as Ocaml, SML, Haskell, Rust.
- **Possible Mentors:** Vadim Pisarevsky
- **Duration:** 175 hours

5. IDEA: Point Cloud Compression

- **Description:** Implement algorithm(s) to compress point clouds and optionally compress meshes. In OpenCV 5.x there is new [3D module](#) that contains algorithms for 3D data processing. Point cloud is one of the fundamental representations of 3D data and there is growing number of point cloud processing functions, including I/O, RANSAC-based registration, plane fitting etc. Point cloud compression is another desired functionality to add, where we throw away some points from the cloud while preserving the overall geometry that the point cloud implicitly describes.
- **Expected Outcomes:**
 - Point cloud compression algorithm developed for OpenCV 3d module (need to follow OpenCV coding style)
 - Unit tests and at least 1 example in C++ or Python that demonstrates the functionality
 - Documentation/tutorial on how to use this functionality.
 - (optional, but desired) optional integration of OpenCV with [Google's Draco](#) library that provides rich functionality for 3D data compression.
- **Resources:**
 - [Point cloud compression tutorial from Pointclouds library](#),
 - [Introduction to point cloud compression by ZTE](#)
- **Skills Required:** Mastery experience coding in C/C++, good understanding of clustering algorithms and, more generally, 3D data processing algorithms.
- **Possible Mentors:** Rostislav Vasilikhin
- **Duration:** 175 hours

6. IDEA: Simple triangle rendering

- **Description:** Some 3D algorithms require mesh rendering as their inner part, for example as a feedback for 3d reconstruction. Depending on algorithm type, a depth or a color rendering is required. This function can also be used as a simple debugging tool. Since light, shadows and correct texture rendering are separate huge problems, they are out of scope for this task.
- **Expected Outcomes:**
 - A function that takes as input a set of points, a set of indices that form triangles and produces depth rendering as output
 - Per-vertex color support is desirable. In that case a function should also take vertex colors as input and produce RGB output
 - Optional: accelerated version using SIMD, OpenCL or OpenGL
 - Optional: jacobians of input parameters as a base for neural rendering

- **Resources:** * [Triangle rasterization tutorial](#) * [3D module in OpenCV](#)
- **Skills Required:** mastery plus experience coding in C++; basic skills of optimizing code using SIMD; basic knowledge of 3D math
- **Possible Mentors:** Rostislav Vasilikhin
- **Duration:** 175 hours

7. IDEA: Demo for Android

- **Description:** Computer vision on mobile phones is a very popular topic for many years already. Several years ago we added support for mobile platforms (Android and iOS) into OpenCV. Now it's time to refresh and update the corresponding parts (build scripts, examples, tutorials). Let's use the formula: Modern Android + OpenCV + OpenCV Deep learning module + camera (maybe even use 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, but 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 on any 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 model. For example, if it's object detection demo, it should be possible to replace one model with another, not via UI, just replace the model file and put the new model name into a config file or into 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++ and/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
- **Duration:** 175 or 350 hours, depending on the feature list

8. IDEA: Implement ONNX and numpy operations

- **Description:** Many modern ML frameworks are built in Python or provide Python interface. It's a standard approach to use Python's numpy extension for ML API or at least use numpy as a reference point to describe semantics of tensor processing operations. In particular, a very popular deep learning model representation standard [ONNX](#) often refers to numpy in the description of ONNX operations. Even though OpenCV provides a number of functions (mostly in the core and dnn modules) to operate on multi-dimensional dense arrays, so far we cannot claim that we provide equivalent functionality with similar API. The goal of this project is to fulfil the gap.
- **Expected Outcomes:**

- Extended/new functionality in OpenCV core, dnn modules that matches a subset of numpy and covers most of basic ONNX, TFLITE operations. At minimum, FP32 should be supported. FP16/INT8 support is desirable but optional.
- A set of unit tests to check the new/extended functionality. Probably, the easiest way to test it would be to use OpenCV Python bindings and develop the unit tests in Python where output from OpenCV will be compared with the output from numpy.
- A sample/tutorial to demonstrate this functionality. Probably, there can be a table with the list of numpy operations and equivalent OpenCV functions.
- **Resources:**
 - [ONNX Operators](#). Please note, how often numpy is referred to.
 - [Broadcasting on ONNX/numpy](#).
 - [Numpy User Guide](#)
- **Skills Required:** Mastery experience coding in C/C++, some practical experience with Python and numpy. 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:** Egor Smirnov, Vadim Pisarevsky
- **Duration:** 175 hours

9. IDEA: Image Augmentation to Assist DL Training

- **Description:** To train high-quality deep learning models a lot of data is usually required; data augmentation is one of the easiest ways to increase data variation. Augmentation could be as simple as image flipping, cropping, scaling, blurring etc. but it can also include more complicated transformations, such as style transfer using another deep learning network. In many ML frameworks OpenCV is used anyway for image I/O and simple image processing. It would be natural to add data augmentation functionality as well to further assist implementation of efficient DL training pipelines.
- **Expected Outcomes:**
 1. Analyze which image transformations are widely used for image classification, object detection, semantic and instance segmentation problems.
 - Things that help with data augmentation for training networks
 - Lighting functions
 - spherical or cylindrical views around a planar object
 - noise ...
 - for 3D point clouds
 2. Create a new OpenCV's module (or use an existing one such `datasets` or `dnn` ?) with at least the following functionality:
 - Provide an API to apply single transformations to an Image or batch of Images, Rectangles (i.e. for ground truth for object detection), Masks.
 - Let users combine different transformations in the class object which can apply them with some probability.
 - Custom data transformations which can be included in the augmentation classes.
 3. Write tutorials targeting on Python wrappers due it's the most popular language supported by different DL frameworks right now.
 - These should in particular show use with PyTorch and TensorFlow.
- **Skills Required:** Experience in image processing and deep learning networks training for computer vision problems.
- **Possible Mentors:** TBD

- **Duration:** 175 hours

10. **IDEA: Multi-task CV models**

- **Description:** Deep learning is now used to solve many different CV problems. A typical approach is to train an individual network for each particular task. But there is a demand in models that are a little closer to human brain and that can solve multiple tasks. Multi-task CV models can also be preferable from computational efficiency standpoint, since many similar CV tasks likely use similar features, so, in principle, they can share the "backbone", often the most computationally expensive part of the deep models.
- **Expected Outcomes:**
 1. One or more multi-task CV models trained (or borrowed, if the license is appropriate), quantized if accuracy does not drop too much, and submitted to [OpenCV model zoo](#).
 2. Necessary patches, if any, for OpenCV DNN to support the provided multi-task models. Ideally, there should be API introduced to optionally disable some parts of multi-task CV model and save time by not computing unnecessary parts/heads of the model.
 3. At least one example that will demonstrate the use of such multi-task model.
- **Resources:**
 - [Multi-task CV model for people detection, pose estimation and action recognition](#)
 - [Survey of multi-task DL models, not limited to CV](#)
- **Skills Required:** Good practical experience in DL, including DL training. Mastery C++ and/or Python coding experience.
- **Possible Mentors:** Yuantao Feng, Shiqi Yu
- **Duration:** 175 hours

11. **IDEA: Lightweight object detection models**

- **Description:** There is a great demand on object detection in real life. In a device where computation capacity, power supply and storage are limited, object detection models need to be efficient, compact and meanwhile accurate enough. Lightweight object detection models have been evolving these years. Users can get help from OpenCV Model Zoo when picking lightweight object detection models to reach better performance on their own devices.
- **Expected Outcomes:**
 1. One or more lightweight object detection models trained (or borrowed if the license is appropriate), quantized if accuracy does not drop too much, and submitted to [OpenCV Model Zoo](#).
 2. Necessary patches, if any, for OpenCV DNN to support the provided lightweight object detection models.
 3. Examples in C++ and Python that demonstrate the use of provided models.
- **Resources:**
 - [Real-time Object Detection on COCO](#)
 - [NanoDet](#)
- **Skills Required:** Good Python and C++ coding skills, training and using deep learning, basic knowledge of computer vision
- **Possible Mentors:** Yuantao Feng, Shiqi Yu
- **Duration:** 175 hours

12. **IDEA: Realtime object tracking models**

- **Description:** Deep learning techniques impressively improve the performance of object tracking algorithms. Such algorithms have been deployed on some drones and robot dogs, with which people can record videos in a distance without the concern of losing cameras. Computation

capacity and power supply can be critical on edge devices such as drones and robot dogs. Object tracking models should be real-time and power-efficient enough to run on edge devices.

- **Expected Outcomes:**

1. One or more lightweight object detection models trained (or borrowed if the license is appropriate), quantized if accuracy does not drop too much, and submitted to [OpenCV Model Zoo](#).
2. Necessary patches, if any, for OpenCV DNN to support the provided lightweight object detection models.
3. Examples in C++ and Python that demonstrate the use of provided models.

- **Resources:**

- [Multitarget Real-Time Tracking Algorithms for UAV IoT](#)
- [Real-time person re-identification at the edge: A mixed precision approach](#)

- **Skills Required:** Good Python and C++ coding skills, training and using deep learning, basic knowledge of computer vision

- **Possible Mentors:** Yuantao Feng, Shiqi Yu

- **Duration:** 175 hours

13. IDEA: Improved ArUco module

- **Description:** The goal is to make a series of improvements to the existing opencv_aruco module, such as:

- Improve the reliability of detecting ArUco markers in videos. This goal can be realized using correlation filters or Kalman filters.
- Extension of the functionality of the Aruco module. For example, the ability to create custom ArUco markers (with a company logo or a picture inside).
- Improve the speed of searching ArUco markers. For example, to implement improvements in the search for markers in the dictionary, improve work with the image pyramid, and improve the dynamic choice of algorithm parameters.
- Measure the accuracy and operation time of Aruco markers detection algorithms depending on the parameters used. To do this, create a dataset and write a benchmark.
- Other suggestions are welcome.

- **Expected Outcomes:**

- Series of patches for OpenCV Aruco

- **Resources:**

- [Tracking fiducial markers with discriminative correlation filters.](#)
- [Design, Detection, and Tracking of Customized Fiducial Markers.](#)
- [Speeded Up Detection of Squared Fiducial Markers.](#)
- [An Improvement on ArUco Marker for Pose Tracking Using Kalman Filter.](#)

- **Skills Required:** practical expertise in C++, basic knowledge of data structures and algorithms

- **Possible Mentors:** Panov Alexander

- **Duration:** 175 hours

14. IDEA: Lightweight OCR models

- **Description:** There are several text detection models and text recognition models in OpenCV DNN modules. And OpenCV even provides a separate API for them. For example, OpenCV has API `TextDetectionModel_DB` for DBNet and `TextRecognitionModel` for CRNN-CTC. In this project we need to enhance the ORC of OpenCV. Contributors need to select and provide better lightweight models for text recognition and text detection. And use model quantization techniques to generate usable INT8 or FP16 models that can run in real time. Note that currently only ONNX quantization models are supported by OpenCV.

- **Expected Outcomes:**
 - Provide quantized CRNN-CTC and DBNet to [OpenCV Model Zoo](#). Submit necessary patches to let `TextDetectionModel_DB` and `TextRecognitionModel` support provided quantized models
 - More lightweight(quantized is prefer) text detection or text recognition models are welcome to be submitted to [OpenCV Model Zoo](#). And necessary patches for OpenCV DNN to support provided models.
- **Resources:**
 - [Related Project: PaddleOCR](#)
 - [Text detection](#), [Text recognition](#) and [ORC Doc](#).
 - [ONNX quantize Doc](#) and [ONNX model quantize.py](#).
- **Skills Required:** Good practical experience in deep leaning, basic knowledge in model quantization. Mastery C++ and/or Python coding experience.
- **Possible Mentors:** Zihao Mu, Shiqi Yu
- **Duration:** 175 hours

15. **IDEA: RISC-V Optimizations**

- **Description:** RISC-V is still one of our main target platforms. During past couple of years we brought in some RISC-V optimizations based on RISC-V Vector extension by adding another backend to OpenCV universal intrinsics. The implementation works correctly, but performance is very low. This year we plan to improve the situation. We also plan to continue working on platform-specific optimization of crucial deep learning kernels.
- **Expected Outcomes:**
 - Updated implementation of RISC-V backend of OpenCV's universal intrinsics. The updated implementation will hopefully avoid excessive load/store operations of vector registers to/from memory.
 - New optimized for RISC-V kernels for deep learning operations, including general convolution (+optional relu or relu6), depth-wise convolution, Winograd-based 3x3 convolution, 1x1 convolution, fully connected layer and max pooling.
- **Resources:**
 - [Optimizing Tensorflow Lite for RISC-V](#)
 - [OpenCV Wide Universal Intrinsics Guide](#)
 - [Implementation of wide universal intrinsics for various platforms, including RISC-V](#)
- **Skills Required:** mastery plus experience coding in C++; good skills of optimizing code using SIMD.
- **Possible Mentors:** Mingjie Xing, Vadim Pisarevsky
- **Difficulty:** Hard
- **Duration:** 175 hours

16. **IDEA: Enable G-API in OpenCV.js**

- **Description:** OpenCV.js is a JavaScript binding for a selected subset of OpenCV functions for the web platform. It allows emerging web applications with multimedia processing to benefit from the wide variety of vision functions available in OpenCV. In this project, we want to enable another framework-level module - G-API from OpenCV native to OpenCV.js to provide end-to-end acceleration. OpenCV G-API is a new OpenCV graph-based module that could speed up the execution of OpenCV functions. This new framework utilizes the graph model to capture all operations and the data dependencies in a pipeline, providing pipeline-level optimization. Previous works focus on how to accelerate a single function or module in OpenCV.js, which benefits the development based on the web interactive mode. While for the deployment of an application, such as online video conference app, to client devices, such interactive mode is rarely used by most of

the users who only repeatedly run several same pipelines to obtain image/video processing results, such as background blur. Thus, such graph-based optimization and execution will be useful for the application deployment of OpenCV.js on the web. In this project, we plan to enable the G-API module with several frequently used OpenCV image processing functions in OpenCV.js and create some examples and tests to show the performance improvements.

- **Expected Outcomes:**
 - Enable G-API basic framework in OpenCV.js
 - Enable frequently used OpenCV functions for image processing in the OpenCV.js G-API module.
 - Create some examples and performance tests to show the performance improvement of G-API in OpenCV.js.
- **Resources:**
 - [OpenCV G-API Introduction](#)
 - [OpenCV G-API Module](#)
 - [OpenCV.js Tutorials](#)
 - [OpenCV.js Module](#)
- **Skills Required:** C++, JavaScript, WebAssembly, Deep Learning(Optional)
- **Possible Mentors:** Ningxin Hu, other OpenCV experts?
- **Duration:** 175 hours

17. **IDEA: First Robotics with OAK-D Camera**

- **Description:** Create a python module that makes it easy to use OAK-D Cameras for First Robotics Competition projects.
- **Expected Outcomes:**
 - Person tracking and distance
 - Object tracking and distance
 - April and Aruco tag homography
 - Calibration
 - Object identification
 - Training environment to install on camera
 - Documentation (examples of how to run models on OAK-D. How to tune a model and upload it to get it working on OAK-D in text and in a YouTube tutorial)
- **Resources:**
 - [OAK-D](#): Where to buy the camera
 - [DepthAI](#) Camera programming environment provided by Luxonis
 - [AI models for OAK-D](#) Many models that work with OAK-D
 - [limelightvision.io](#) is an example of a first robotics camera
- **Skills Required:** Mastery of Python, familiarity with OAK-D and OpenCV
- **Difficulty:** Medium
- **Possible Mentors:** Prassana Krishnasamy
- **Duration:** 480 hours

18. **IDEA: Multi-grid multi-camera calibration**

- **Description:** Set up some April Tag or Aruco boards in a non-plannar rigid arrangement. Calibrate one or more cameras (intrinsically) first and then extrinsically by allowing the cameras to see part of the boards. The result will be extrinsic calibration of the cameras and 3D solution for the boards.
- **Expected Outcomes:**
 - Intrinsically Calibrated cameras
 - Extrinsically Calibrated cameras
 - 3D coordinates of the calibration grids

- Documentation (examples of how to calibrate in a text documentation and a YouTube tutorial)
- **Resources:**
 - [OpenCV Calibration documentation](#)
 - [OAK-D](#): Where to buy the camera
 - [DepthAI](#) Camera programming environment provided by Luxonis
- **Skills Required:** Mastery of C++, mathematical knowledge of camera calibration, ability to code up mathematical models
- **Difficulty:** Difficult
- **Possible Mentors:** Jean-Yves Bouguet
- **Duration:** 175-300 hours

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](#).

Contributors

How to Apply

The process is described at [GSoC home page](#)

How contributors will be evaluated once working:

- Contributors will 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 contribution
 - **Phase 2:**
 - You must generate a pull request
 - That builds
 - Has all or most of the planned functionality (but still usable without those missing parts)
 - With OpenCV appropriate Doxygen documentation
 - Includes What the function or net is, what the function or net is used for
 - Has some unit tests
 - Has a tutorial/sample of how to use the function or net and why you'd want to use it.
 - Optionally, but highly desirable: create a (short! 30sec-1min) Movie (preferably on Youtube, but any movie) that demonstrates your project. We will use it to create the final video:
 - [The 2021 Movie](#)
 - [The 2020 Movie](#)
 - [The 2015 Movie](#)
 - [The 2014 Movie](#)
 - [The 2013 Movie](#)
 - **Extended period:**
 - TBD

Mentors:

1. Contact us, preferably in February or early March, 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:
 - [Mentor only list](#)
 - [The Contributor+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 the project proposals and discussions. Discuss the ideas you've chosen.
 - Find likely contributors, ask them to apply to your project(s)
 - You will get a list of contributors who have applied to your project. Go through them and select a contributor or rejecting them all if none suits and joining to co-mentor or to quit this year are acceptable outcomes.
 - Make sure your contributors officially apply through the [Google Summer of Code site](#) prior to the deadline as indicate by the Contributor 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. Contributors 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 contributor **and** we give you a slot **and** the contributor 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
Jia Wu
Yuantao Feng
Zihao Mu

Admins

Gary Bradski
Vadim Pisarevsky
Shiqi Yu

GSoC Org Application Answers

[Answers from our OpenCV GSoC application](#)