OpenCV Google Summer of Code 2020


OpenCV Ideas Page

Heat maps from OpenCV in Python by Intel IoT.

- *[[Parent page|OpenCV_GSoC]]*
- *[[Last year|GSoC_2019]]*

Mentor, Student, Admin Mailing List: TBD

Student intern application process

model drive space: TBD

OpenCV Accepted Projects:

- Mentor only
- Sheet

Student	Title	Mentors	Passed
Yoshio Kato	Better SIFT in the main repository	Vitaly Tuzov	👍
Akash Sharma-2	Depth Fusion for large scale environments	Rostislav Vasilikhin	👍
Zhiming Zeng	Develop OpenCV.js DNN modules for promising web use cases together with their tutorials	Ningxin Hu	👍
Jiang Hao	GPU-enabled OpenCV.js	Zhiwen Wu	👍
Ajit Pant	Implement Macbeth Chart detector and a better detector for AprilTag	Gary Bradski Gholamreza Amayeh	👍
Maksym Ivashechkin	Improvement of Random Sample Consensus	Riba Edgar Vadim Pisarevsky	👍
Archit Rungta	OpenCV bindings for Julia	Sayan Sinha Vadim Pisarevsky	👍
Kun Liang	OpenCV.js: WASM SIMD optimization 2.0	Vitaly Tuzov Ningxin Hu	👍
Joy2myself	Optimize OpenCV for RISC-V	Alexander Smorkalov	👍
Devansh Batra	Point Clouds Model Fitting Implementation	Mihai Bujanca	👍
Jin Yeob	Real-time Single Object Tracking using Deep Learning	Liubov	👍

Chung		Batanina Stefano Fabri Ilya Elizarov	
Wenqing Zhang	Revise/improve Text & Digit Recognition Samples	Shiqi Yu	👍
Zihao Mu	Revise/improve Text & Digit Recognition Samples	Vladimir Tyan	👍
Anastasia Murzova	Write a tutorial about training a network in TF or PyTorch and then running it in OpenCV	Anna Petrovicheva	👍

Important dates:

Date (2020)	Description	Comment
January 14	Mentoring organizations begin submitting applications to Google	👍
Feb 5	Mentoring organization application deadline	👍
February 20	Organizations Announced **WE ARE IN!**	🏅
Feb 20 - Mar 15	Talk to us	👍
Mar 16 - Mar 31	Apply to OpenCV through GSoCgoogle.com/	👍
Mar 31 - Apr 27	Application Review Period	👍
Apr 21	OpenCV Slot Requests	:boom: **14** :boom:
Apr 30	OpenCV tell Google Selected Projects	👍
May 4	Student Projects Announced by GSoC	👍
May 4 - June 1	Community Bonding	👍
Jun 01 - Aug 24	*Coding Period*	👍
Jun 29 - Jul 3	**Evaluation #1**	💯 **%**
Jun 29 - ...	...*Coding Continues* ...	👍
Jul 27 - Jul 31	**Evaluation #2**	💯 **%**
Jul 31 - Aug 24	...*Coding Continues*	👍
Aug 24 - Aug 31	Students submit final code and their evaluations	👍
Aug 31 - Sep 7	**Final Evaluation #3**	🤸
Sep 8	Results announced	

Timeline

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:

- GSoC Home Page
- OpenCV Project Ideas List
- OpenCV official Site
- OpenCV wiki
- 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-2020

Index	to	Ideas	Below
1: Fix Bindings	2: RANSAC	3: Fiducial Tags	4: Calibration
5: ONNX for DNN	6: RISC-V	7: Better SIFT	8: Text Rec.
9: Point Cloud Fit	10: TF/Pytorch=>OpenCV	11: Parallel for	12: DL Trackers
13: Julia Bindings	14: Microsoft Nuget	15: Large Scale Depth Fusion	16: Support Audio I/O module
17: QR-codes algebra	18: Deepnets on Video	19: Improve and document OpenCV.js	20:Quantize & Prune for DNN
21: Data Augmentation	:	:	:

All work is in C++ unless otherwise noted.

Ideas:

1. ***IDEA:*** **Fix Bindings Everywhere**
    - ***Description:*** OpenCV has been updated to C++11, not all binding to Python, Javascipt etc work. We want to automate the process and fix this problem. Maybe we should use one of parser generator tool for Python in order to make the OpenCV header parser less "ad-hoc".
    - ***Expected Outcomes:***
        - Updated Python, Java etc. bindings to handle the new C++11 API correctly and produce more efficient code for the bindings.

        - Extended Python, Java, Javascript tests to test more corner cases of the new bindings generator. (https://github.com/opencv/opencv_contrib/tree/master/modules/optflow) that includes some new optical flow algorithms.
    - ***Resources:***
        - Parser generators for Python.
    - ***Skills Required:*** mastery experience coding in C++ and Python
    - ***Possible Mentors:*** Vadim Levin, Alexander Smorkalov
    - ***Difficulty:*** Hard (devil is in the details)

2. ***IDEA:*** **Improve RANSAC in OpenCV**
    - ***Description:*** OpenCV has fallen behind the state of the art RANSAC techniques, as described in the paper below.
    - ***Expected Outcomes:***
        - improved implementation of RANSAC in OpenCV (perhaps, `calib3d` module).
        - the tests and documentation for the new implementation.
        - possibly, a parallel solvePnP implementation using the new algorithm.
        - a report comparing the old and the new implementation.
    - ***Resources:***
        - Slides where different RANSAC implementations are compared
        - Better RANSAC: implementation in Python
    - ***Skills Required:*** mastery experience coding in C++, familiarity with the basic RANSAC algorithm.
    - ***Possible Mentors:*** Edgar Riba
    - ***Difficulty:*** Medium

3. ***IDEA:*** **Create robust visual fiducial tags**
    - ***Description:*** Robustly identify standard visual fiducials for April Tags and MacBeth Charts. 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 aforementioned calibration rigs.
        - 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.**
        - Various references on Macbeth Color Chart. **A note for the mentors and admins. Please, carefully watch the licenses and the possible patent violation!**
    - ***Skills Required:*** Mastery plus experience coding in C++; experience with image processing.
    - ***Possible Mentors:*** Gary Bradski, Gholamreza Amayeh
    - ***Difficulty:*** Hard

4. ***IDEA:*** **Calibration Improvements**
    - ***Description:*** Allow for and end to end of extrinsic calibration of intrinsically calibrated cameras using a set of calibration targets place in the scene. Recover the camera extrinsics and the geometry of the calibration patterns.
    - ***Expected Outcomes:***
        - the new calibration algorithm in opencv_contrib/ccalib module.

        - documentation, the regression tests and optionally the sample to demonstrate the new functionality.
    - ***Resources:*** * TBD
    - ***Skills Required:*** mastery plus experience coding in C++, familiarity with multi-view geometry, camera calibration techniques.
    - ***Possible Mentors:*** Reza Amayeh
    - ***Difficulty:*** Hard

5. ***IDEA:*** **Fully support ONNX for DNN**
    - ***Description:*** Choose a sufficiently recent version of ONNX and support DNN running all models of that version of ONNX.
    - ***Expected Outcomes:***
        - A series of patches brining better support for ONNX to OpenCV DNN
        - Extended regression tests to properly test the new functionality
    - ***Resources:*** * Repository of different ONNX models
    - ***Skills Required:*** mastery experience coding in C++; familiarity with Deep Learning, basic knowledge of ONNX, protobuf etc. A definite plus would be a practical experience with internals of OpenCV DNN or other deep learning inference tool (Tensorflow, PyTorch etc.)
    - ***Possible Mentors:*** Dmitry Kurtaev
    - ***Difficulty:*** Hard

6. ***IDEA:*** **Optimize OpenCV for RISC-V**
    - ***Description:*** OpenCV provides a convenient method to port many optimized kernels at once to a new CPU, as long as that CPU supports SIMD instructions. We use so-called *Wide Universal Intrinsics* for that. By adding implementation of the wide universal intrinsics for RISC-V we can make OpenCV run pretty efficiently on RISC-V CPUs.
    - ***Expected Outcomes:***
        - intrin_riscv.hpp added to `opencv_core` module. It does not have to implement all of the intrinsics. We can start with vector single-precision floating-point arithmetics that would be sufficient for OpenCV DNN acceleration.
        - tweaking OpenCV CMake scripts to let OpenCV cross-compile for RISC-V (Linux OS).
        - make the *wide universal intrinsics* regression tests pass on the software simulator (QEMU)
    - ***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, Alexander Smorkalov
    - ***Difficulty:*** Hard

7. ***IDEA:*** **Better SIFT in the main repository**
    - ***Description:*** In 2020 March the patent on one of the most popular feature detection algorithm, SIFT, expires. So, we can move the implementation from opencv_contrib/xfeatures2d to the main OpenCV repository (opencv/features2d) in the late spring or summer. We can also optimize and improve it further, probably create bit-exact implementation.
    - ***Expected Outcomes:***
        - Transfer SIFT implementation, together with the tests, to the main OpenCV repository.
        - Convert Intel-specific AVX2 intrinsics used in the implementation to wide universal intrinsics.

    - (optional) Convert it to bit-exact implementation that does not use floating-point operations. Maybe just the detector part.
    - Add A-SIFT into the C++ interface. Right now there is Python script to compute A-SIFT on top of SIFT. It can be converted to C++ for better efficiency (including parallel processing of the transformed images).
  - ***Resources:***
    - Current SIFT Implementation in OpenCV
    - OpenCV Wide Universal Intrinsics Guide
    - A-SIFT Example in Python
  - ***Skills Required:*** mastery plus experience coding in C++; basic skills of optimizing code using SIMD; basic knowledge about feature points in images, what they are about, how to use them.
  - ***Possible Mentors:*** Vitaly Tuzov, Vadim Pisarevsky
  - ***Difficulty:*** Medium

8. ***IDEA:*** **Revise/improve Text & Digit Recognition Samples**
  - ***Description:*** In OpenCV there are several quite boring samples on digit and text recognition, digits.cpp, digits.py, letter_recog.cpp ... . In the modern deep learning era they can be done in much better way. In particular, instead of using classical algorithms in letter_recog.cpp, we can extend OpenCV deep learning-based text detection sample and run some network to recognize the detected text, at least English text. In digits.cpp/.py we can take the live image from camera, use connected component analysis to detect potential regions with each digit (let's assume we snow to the camera a white paper sheet with hand-written digits in black/blue/green/red etc. color) and then run a classical Yann LeCun's LeNet-5 network, which is very compact, so it can be included into the repository or easily downloaded from the net. To make the recognition more stable, each region may be rotated several times, i.e. run through the network under different angles.
  - ***Expected Outcomes:***
    - Completely rework digits.cpp (digits_video.py does something like that, but it's broken now) to use a live feed from camera instead of static images. Segment digits, run LeNet-5 or similar convolutional network to recognize the digits. Display the recognized digits as overlays on top of the original digits.
    - Remove letter_recog.cpp from opencv/samples/cpp. Instead, add opencv/samples/dnn/text_recog.cpp (or text_recog.py) that will not only detect text, but will recognize it.
  - ***Resources:***
    - Current Python Samples including digits_video.py and digits.py
    - Classical LeNet-5 Architecture for hand-written digit recognition
  - ***Skills Required:*** mastery plus experience coding in C++ and Python; good knowledge of basic image processing (connected components analysis) and deep learning.
  - ***Possible Mentors:*** Jia Wu
  - ***Difficulty:*** Medium (just digits) to Hard (digits & text recognition)

9. ***IDEA:*** **Add Robust Plane/Sphere/Cylinder Fitting into 3D Point Clouds**
  - ***Description:*** It's time to start extending OpenCV 3D handling capabilities. One of the important components of 3D Point Cloud processing pipeline is detection of some canonical 3D shapes that make up the cloud. In the urban environments, such as homes, offices, factories, cities, such canonical shapes (planes mostly, sometimes cylinders, occasionally spheres) often occur and they play important role in the 3D scene analysis. In OpenCV we have `opencv_contrib/rgbd` module with some algorithms for 3D cloud processing. Plane, cylinder and sphere fitting would be a useful addition to this set of algorithms.

- ***Expected Outcomes:***
  - Add plane fitting into `opencv_contrib/rgbd' module. It should not be just a single plane fitting. It should be segmentation, where the found plane is "subtracted" from the scene and another plane (or other shape) fitting algorithm is applied to the remaining point cloud. Regression test should be added, based on some real or synthetic data.
  - Similarly to the above item, cylinder and sphere fitting should be added as well. In general, we should be able to decompose the scene into a number of planes, cylinders, sheres and the "other" objects.
  - (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
- ***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:*** Hard

10. ***IDEA:*** **Write a tutorial about training a network in TF or PyTorch and then running it in OpenCV**
    - ***Description:*** TensorFlow (TF) and PyTorch are some of the most popular deep learning frameworks. TF exports networks in its own format; PyTorch supports export of the trained models into ONNX format, which is also very popular. OpenCV is becoming quite popular tool for Deep Learning Inference. It would be very useful to describe clearly how to train a model in TF or PyTorch so that it can be then run in OpenCV.
    - ***Expected Outcomes:***
      - Write a tutorial that will explain the whole pipeline step by step. Take object detection as the example task. ONNX does not provide all those standard layers for object detection, but OpenCV covers this missing part. Describe it clearly.
      - The tutorial(s) should include some basic information (or a link) how to configure PyTorch/TF, get dataset, train the detection network, and then how to export it to ONNX. Full script should be provided. In the case of TensorFlow it's easier, but still some "finalization" of the network may be needed to run it smoothly with OpenCV.
      - The tutorial(s) should also include description and the whole OpenCV-based example that will read the trained network and will run it on the live video stream from camera.
    - ***Resources:***
      - PyTorch
      - ONNX
      - Object Detection using TensorFlow
      - OpenCV Object Detection Example. For the tutorial it's too complex. Much simpler one is needed.
    - ***Skills Required:*** good practical experience in DeepLearning, including TF and/or PyTorch. Good Python coding skills. Very good English.
    - ***Possible Mentors:*** Dmitry Kurtaev, Jia Wu
    - ***Difficulty:*** Medium

11. ***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

12. ***IDEA:*** **Deep learning based visual trackers**
    - ***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. The first step is to analyze and evaluate performance of selected tracker. Then port/import required DNN model in OpenCV DNN (may require implementing some additional network layers) and finally, analyze OpenCV tracking API and integrate DNN tracker into OpenCV contrib tracking module.
    - ***Expected Outcomes:***
      1. Refactor existing tracking functionality considering modern deep learning based approaches
      2. Research currecn state-of-the-art of DL-based trackers. Compare by accuracy, efficiency, training code availability and pre-trained networks, license.
    - ***Skills Required:*** Experience in image processing and deep learning networks training for computer vision problems.
    - ***Mentors:*** Ilya Elizarov, Dmitry Kurtaev, Liubov Batanina
    - ***Difficulty:*** Medium

13. ***IDEA:*** **Add automatically generated Julia bindings for OpenCV**
    - ***Description:*** Julia language is becoming a popular solution for computing, for research, as it's similar to Python and provides built-in array support.
    - ***Expected Outcomes:***
      - Automatically generated bindings for Julia + a few samples (probably translated from Python). The same OpenCV header parser as for Python, Java etc. should be used. Python bindings generator can be used as a starting point.
    - ***Resources:***
      - Julia Bindings for OpenCV. Evolution Proposal
      - Python Bindings for OpenCV
      - Python Samples for OpenCV
    - ***Skills Required:*** good expertise in C++ and Julia, very good expertise in Python.
    - ***Possible Mentors:*** Sayan Sinha
    - ***Difficulty:*** Hard

14. ***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

15. ***IDEA:*** **Depth fusion algorithm for large scale**
    - ***Description:*** The `rgbd` module lacks a 3d reconstruction algorithm for large scale scanning. This cannot be done by both existing KinectFusion and DynamicFusion because their capture space is limited and cannot extend as the camera moves and their space representation consumes too much memory. There are popular algorithms which solve that problem by changing the space representation to less memory consuming and less space limited, like VoxelHashing, Kintinuous, Surfels, etc. It's up to participant what one to choose and to implement.
    - ***Expected Outcomes:*** GSoC participant is expected to build a version of the chosen algorithm that conforms to OpenCV standards:
        - it has (maybe slow) CPU version
        - it has tests w/ captured depth or rendered scene as it's done in OpenCV's KinFu version
        - it builds, it works and produces fine results at least on one dataset (better if this works in live mode)
        - it has as few dependencies as it is possible (ideal case is no dependencies outside of OpenCV)

      This should be done inside `rgbd` module using its primitives and infrastructure (which can be extended, of course).
    - ***Resources***
        - Kintinuous large-scale version of KinFu which is based on pose-graph
        - KinFu Large Scale this one is based on marching cubes
        - rgbd module containing existing KinFu implementation
        - RGB-D SLAM dataset
        - Open3D's ScalableTSDF
        - ElasticFusion
        - Point-Based fusion
    - ***Skills Required:*** good understanding of 3d math, experienced in C++, experience in computer vision or image processing (at least one successful project), a lot of patience
    - ***Mentors:*** Rostislav Vasilikhin
    - ***Difficulty:*** Hard

16. ***IDEA:*** **Support Audio IO module**
    - ***Description:*** At the moment, OpenCV can works with inference deep neural network with help DNN module. But DNN module has limitation, it work with only images. In addition, exist much

network which work with audio. That why proposed support audio input/output in OpenCV for extension functionality of library and more specifically extension functionality of DNN module.

- ***Expected Outcomes:*** GSoC participant is expected to support audio input/output module for Windows or MacOS family operating system. Will require a study Windows Core Audio APIs (WASAPI) and MacOS Core Audio.
- ***Resources***
  - WASAPI - Windows audio I/O API.
  - Core Audio - MacOS audio I/O API.
- ***Skills Required:*** experienced in C++, experience in computer vision and OpenCV library.
- ***Mentors:*** Nesterov Alexander
- ***Difficulty:*** Medium

17. ***IDEA:*** **Implementation of QR-codes decoder and encoder**
    - ***Description:*** OpenCV has well-working QR codes detection and decoding algorithm. Currently it uses third party library for decoding and doesn't have functionality for creation of QR-codes. The first task will be implementation decoder with help of OpenCV functionality. This will help to solve second task – QR-code encoder (tasks are interconnected). Student will need to research QR-codes decoding algorithms, error correction rules, data character encoding methods and to program decoder and encoder.
    - ***Expected Outcomes:***
      - Integration of decoding functionality of QR codes that will pass existing tests.
      - Create encoding functionality and API with regression and performance tests
      - New functionality has no 3rdparty dependencies and no code replacements from other libraries
    - ***Skills Required:*** experienced in C++, OOP, experience of using OpenCV library.
    - ***Mentors:*** Grace Vesom
    - ***Difficulty:*** Medium

18. ***IDEA:*** **Python deep learning inference on video**
    - ***Description:*** OpenCV's DNN module allows high-level inference on individual images. But, performing inference on video requires producing much boilerplate code and skills not directly relevant to computer vision. The goal of this project is to develop a high-level helper class in python to perform optimized inference on videos (eg, pose detection, emotion detection) with data storage (eg, output and bounding boxes) in dataframes for easy access.
    - ***Expected outcomes:***
      - Review the papers on the topic in the resources below
      - Define and implement an API for proposed methods
      - Optimize batch processing of video input for neural networks
      - Implement the Model Zoo for video use cases
      - Handle optimal neural network inference output for further processing
      - Write examples and tutorials
    - Resources
      - DNN Tutorial

        - List of DNNs
        - List of major DNN models
    - ***Skills Required:***
        - Coding in Python. Experience with deep neural networks.
    - ***Mentors:*** Justin Shenk
    - ***Difficulty:*** Medium

19. ***IDEA:*** **Improve and Document OpenCV.js**
    - ***Description:*** OpenCV.js is Javascript

interface to OpenCV, standalone javascript module that includes essential parts of OpenCV, including core functionality, image processing, deep learning etc. It has been created and further developed during GSoC 2017 and 2019, and we want to continue this good tradition. Several projects can be done, including better WASM optimization, GPU-based acceleration, and the new use cases & tutorials. Please, find the detailed proposals here: https://github.com/huningxin/opencv/wiki/OpenCV-GSoC-2020-Proposal * ***Expected outcomes:*** * Improved, more efficient OpenCV.js (Note that OpenCV.js is generated completely automatically). * New tutorials and examples of how to use OpenCV.js in real-world scenarious. * Resources * See the link above * ***Skills Required:*** * C++, Javascript, Web technologies, experience with GPU-based code optimization etc. * ***Mentors:*** Ningxin Hu, Vitaly Tuzov * ***Difficulty:*** Medium to Hard

1. ***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 prunned 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

2. ***IDEA:*** **Data Augmentation**
    - ***Description:*** Deep learning networks are hungry for data and data augmentation is one of the easiest ways to increase data variation. Augmentation could be as simple image flipping, cropping

and scaling on up to more complicated transformations such style transfer using another deep learning network. For computer vision problems, OpenCV is often used for reading images in most of training scenarios, so why we'd like to enhance data reading with simple to use data augmentation techniques as well.

- ***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:*** Edgar Riba, Dmitry Kurtaev
- ***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

OpenCV is taking part in GSoC 2020. Above is the list of ideas from developers, staff and Evolution Proposals.

1. **Requirements:**
    - You **must** already know how to program fluently in C++!
    - Some projects may instead specifically require Python or javascript
    - Some projects may require knowledge of Deep Nets and perhaps one or more of the standard packages:
        - PyTorch, TensorFlow, MXNet. Familiarity with OpenCV's DNN (here for code and here for samples) and ONNX deep net exchange is a big plus
2. **Please familiarize yourself with the:**
    1. OpenCV Developer's site, the
    2. User's site, read through the ideas list below to see which projects are of interest to you, and familiarize yourself with:
    3. The GSoC 2020 timeline
3. **Sign up:** for the OpenCV GSoC 2020 mailing list where you can ask questions, exchange and discuss ideas with students and mentors, get announcements etc.
4. **Optional steps mentors may take:**
    - Mentors may contact you for hangouts or other means of live interview
    - You may be asked for proof of coding
    - You will probably be asked for a full project plan
5. **How to enhance your application:**
    1. Contributing to OpenCV is a **big plus**. Some suggestions:
        1. Fixing an bug/issue or
        2. Very advanced people can already start delivering code for the idea they like to their mentor
        3. More realistically and still **great**: REALLY learn some OpenCV function and contribute a well-written tutorial on it

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 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
    - The Proposals Spreadsheet TBD

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 before March 16th.

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