OpenCV Google Summer of Code (Ideas for GSoC 2020)

2019 Project Page on GSoC site

This is an example from heartbeat.fritz.ai where Saumya Shovan Roy got DNN object detection working on a Raspberry Pi using OpenCV DNN

Mentor, Student, Admin Mailing List

model drive space

OpenCV Accepted Projects:

Mentor only

Student	Title	Mentors	Passed
Yashas Samaga B L	Allow the OpenCV's DNN module to work with CUDA	Davis King	gitgist video
Vedanta Keshav Jha	Alpha Matting	Steven Puttemans Gholamreza Amayeh Sunita Nayak	NA video report
Muskaan	Computer Vision based Alpha Matting	Gholamreza Amayeh Sunita Nayak	gitgist video +info
Diego Velazquez	Curating Deep Nets for the OpenCV DNN Module	Stefano Fabri Edgar Riba	gitgist video

			+info
Fady Essam Baskharon	Data augmentation module	Dmitry Kurtaev Gholamreza Amayeh Edgar Riba	NA
Fanny Monori	Deep learning based super-resolution algorithms based on OpenCV DNN	Vladimir Tyan	gitgist video
Apoorv Goel	DynamicFusion Implementation	Rostislav Vasilikhin	gitgist video
Saiteja Talluri	Facial Landmark Detector	Satya Mallick	gitgist video
Wenzhao Xiang	Improve the performance of JavaScript version of OpenCV (OpenCV.js)	Vitaly Tuzov Ningxin Hu	gitgist video
Xavier Weber	Learning-based Super Resolution	Yida Wang Vladimir Tyan	gitgist video

---

Important dates:

Date (2019)	Description	Comment
January 15	Mentoring organizations begin submitting applications to Google	:ok:
Feb 6	Mentoring organization application deadline	:ok:
February 26	Organizations Announced	:+1: **We are in!**
Feb 27 - Mar 24	Talk to us	:ok:
March 25 - April 9	Apply to OpenCV through GSoC	:+1:
April 9 - May 6	Application Review Period	*slots requested*
May 6	Student Projects Announced by GSoC	👍
May 6 - 27	Community Bonding	:ok:
May 27 - ...	*Coding ...*	:ok:
June 24-28	**Evaluation #1**	👍 All students passed
June 24 - ...	*Coding Continues ...*	:ok:
July 22-26	**Evaluation #2**	👍 9/10 students passed
July 22 - Aug 19	*Coding Continues ...*	:ok:
Aug 19-26	Students submit final code and their evaluations	:ok:
Aug 26-Sept 2	Mentors submit their **Final evaluation**	👍 8/9 students passed

Sept 3	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:

- 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

Student Info

- ### 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

OpenCV Project Ideas List:

Index	to	Ideas	Below
Circular Calibration	Data Augmentation **1** **2**	GPU backend for DNN	Binary Neural Nets
Model Zoo	Point Coordinate Regression	Differential Rendering	Image Processing
April Tags or Geometric April Tags	Optical Flow	Python OpenCV	Depth Fusion
Face Landmarks	Boosted Cascades	Tutorials: **G**eneral; **M**achine Learning	OpenCV GUI
DNN Quantization	DNN Super-Resolution	Deep Alpha Matting **1** **2**	Deep Nets on Video
Add Training DNN			

Add training to OpenCV DNN module

AREAS:					
1: Calibration	2: Data Augment	3: Deep Learning	4: Diff. Ops	5: Fiducials	6: Geometric
7: Language	8: SLAM	9: Tracking/Seg	10: Tutorials	11: GUI	

How to Apply

OpenCV is taking part in GSoC 2019. Below is a list of ideas from the OpenCV developers, staff and Evolution Proposals. ***Note:*** *We have a pre-application form below prior to Mar 25, but you must also register with GSoC and apply to OpenCV there between March 25-April 9. See our timeline below*

1. **Requirements:**
   - You **must** already know how to program fluently in C++!

- Some projects may instead specifically require Python, javascript or Matlab skills
- 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 our
   3. Ideas List to see which projects are of interest to you, and familiarize yourself with
   4. The GSoC 2019 timeline
3. **Sign up:** for the OpenCV_GSoC_2019 mailing list where you can ask questions, exchange and discuss ideas with students and mentors, get announcements etc.
4. **Fill out:** the pre-application form for the one or two (*maximum*) projects you are interested in.
5. **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
6. **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

---

Ideas:

1. **Area: Calibration**
   1. ***IDEA:* Calibration Patterns using Circular Features**
      - ***Description:*** OpenCV currently allows checkerboard, Aruco, mixed and a basic circular calibration grid. Circular patterns have very good detectability across scale, but the Circular calibration feature in OpenCV is not complete because when a circle is viewed in perspective, the center of the circle is not at the center of mass. This introduces a systematic error into calibration with circular patterns. This can be corrected by compensating for perspective distortion.
      - ***Expected Outcomes:***
        - Review the papers on the topic in resources below.
        - Code up a circular calibration module that implements the adjusts the circle center for perspective distortion.
        - Extensive validation (using other patterns and the circular pattern) that shows the circular pattern returning expected intrinsic and distortion camera parameters.
        - Optimizing the module to run efficiently
        - Python interface to the module
        - Resources
          - Geometric Correction of Circular Fiducials; direct pdf link

        - [Valication Code for Circular Correction]()

    - ***Skills Required:*** Solid C++, Coursework and/or experience in camera calibration. Python is a plus because we want calibration to be callable from Python. Mainly, calibration software depends on the details, so being detail oriented is critical as well as the willingness to write thorough test code.
    - ***Mentors:*** Grace Vesom, Gholamreza Amayeh
    - ***Difficulty:*** Medium

2. **Area: Data Augmentation**
    1. ***IDEA:*** **Computer vision data augmentation module**
        - ***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.
        - ***Mentors:*** Dmitry Kurtaev
        - ***Difficulty:*** Medium
    2. ***IDEA:*** **3D Augmentation & April Tags**
        - ***Description:*** 3D computer vision related data is represented mainly in two formats: volumetric data and point clouds. It is usually jointly combined with 2D data processing approaches and 3D approaches for semantic inference and fusion. In this project, we focus on contributing 3D semantic processing modules without dependence of heavy deep architectures. One application would be the April Tags recognition task.
        - ***Expected Outcomes:***

1. Modify TSDF-Fusion (***T****runcated* ***S****igned* ***D****istance* ***F****unction-Fusion*) to produce 3D scenes (point clouds) from a sequence of depth images.
2. On top of TSDF Fusion, implement PCN-like architecture for 3D completion from depth images. Try to make it more efficient by getting rid of convolutional operations for 3D data.
3. C++/CUDA based 3D visualization APIs which are capable for rendering points clouds in the form of .ply or .obj files into high-quality mesh models with realistic shadows. * (Additional) Implement AprilTag for full 6 DOF localization of features from a single image. Try to solve problems introduced by occlusion and bad lighting conditions.

   - resources:
     - TSDF-Fusion
     - PCN
     - AprilTag
     - PointNet
     - FoldingNet

- ***Skills Required:*** Experience in 3D data processing and linear algebra. Good at C++ and familiar with CUDA. Students should be able to understand the basic idea behind 3D point clouds deep learning approaches from basic PointNet.
- ***Mentors:*** Yida Wang
- ***Difficulty:*** Medium-Hard

3. Area: Deep Learning

1. *IDEA:* Allow the OpenCV Deep Neural Net Module (DNN) to work with GPUs

- ***Description:*** The OpenCV Deep Neural Network (DNN) Module takes deepnets found on the net and runs them 5x faster than Pytorch or Tensorflow on CPUs. Programmers find DNN simple to use with powerful results. But, DNN is missing an NVidia GPU backend. This project aims at adding that missing GPU backend.
- ***Expected Outcomes:***
  - Developed plan to port the existing CPU backend to support GPU
    - Review tools NVidia here, Kokkos here, RAJA here, OpenCL here, and CUDA here.
  - Implemented the new GPU backend for DNN
  - A test suite that compares CPU results with GPU results and makes sure they are close
  - Performance as fast or faster than native use of nets on GPUs with TensorFlow and Pytorch
  - Resources
    - In general, you can leverage the CUDA backend for Torch or DLib (Mentor has info)
    - DNN Module
    - DNN Tutorials
    - DNN Wiki -- what's in it
    - DNN Efficiency
    - DNN TensorFlow API
    - DNN Backend Description
- ***Skills Required:***
  - **Must have C++ mastery**

- Good to have:
  - Python and python interface tools for C++ code,
  - GPU/Cuda working experience,
  - coursework or professional work with Deep Nets,
  - Addional plus if you've optimized code before.

- ***Mentors:*** Davis King (Dlib creator)
- ***Difficulty:*** Medium-Hard (not hard if you know the tools/have worked with GPUs and C++ before)

2. ***IDEA:*** **Enable Training and Inference with Binary Neural Networks**
   - ***Description:*** Implement elementary building blocks for binary convolutional neural networks (binary activations and binary weights) for CPU with predictor and trainer modules.
   - ***Expected Outcomes:***
     - Inference module for binary convolutional layers.
     - Trainer module for training ensemble of the networks.
     - Exporter to generate self-contained network code with parameters (weights, etc.)
     - Resources
       - paper: https://arxiv.org/abs/1602.02830
       - paper: https://arxiv.org/abs/1809.03368
       - paper: https://arxiv.org/abs/1603.05279
   - ***Skills Required:*** Skills Required: Coding in C/C++, speed optimization, multi-threading, random number generators, probabilities, Bayesian inference.
   - ***Mentors:*** Michael Tetelman
   - ***Difficulty:*** Medium/Hard

3. ***IDEA:*** **Model Zoo Curate deep nets for the DNN**
   - ***Description:*** The DNN module in OpenCV allows for accomplishing powerful high end vision tasks with just a few lines of code. The goal of this project is to curate more models for ease of use by the OpenCV DNN module and put them in a place that they can be easily accessed such as LFS on Git. The mentor/admins will formalize the storage place.
   - ***Expected Outcomes:***
     - Read through the list of known to work models and major deepnet models in the resource list below. Come up with a target list of 6 new neural network models to add.
     - For each deepnet model in turn
       - Use ONNX or other to allow the model to be run in DNN
       - Write a concise sample code and tutorial showing loading and use of that deepnet model
       - We will allocate a data store for your DNN enabled models so that they may be called by DNN and run. Put your models and other known to run models in this data store.
     - Resources
       - DNN Tutorial
       - List of deepnets known to work with DNN
       - List of major deepnet models
       - Code for DNN Module
       - LFS on Github
   - ***Skills Required:*** Coding in C++ and Python. Experience with deep neural networks.

- ***Mentors:*** Stefano Fabri
- ***Difficulty:*** Easy

4. ***IDEA:*** **Add quantization and pruning functionality 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
   - ***Mentors:*** Tyan Vladimir
   - ***Difficulty:*** Medium-Hard

5. ***IDEA:*** **Deep Learning based Super-Resolution algorithms based on OpenCV DNN**
   - ***Description:*** Project aims to build a super-resolution (SR) module inside OpenCV. Most probably base SR algorithms will be some Deep Learning algorithms (FSRCNN and EDSR) as recovering accuracy of DL algorithms is much better. But Computer Vision based algorithms can also taken in consideration (A+ and etc.) if their usefulness is proven in terms except accuracy.
   - ***Expected Outcomes:***
     - Build Super-Resolution module and implement interface for running different SR algorithms.
     - Add SR datasets to OpenCV datasets module
     - Implement FSRCNN algorithm (fast one) and train model on SR datasets
     - Implement EDSR algorithm (accurate one) and train model on SR datasets
     - Experiments and comparison of different models along with simple bicubic interpolation (or resize?)
     - Additional goals
       - Implement Computer Vision SR algorithm (A+ and etc.)
     - Resources
       - https://arxiv.org/pdf/1608.00367.pdf
       - https://arxiv.org/pdf/1707.02921.pdf
       - https://www.vision.ee.ethz.ch/publications/papers/proceedings/eth_biwi_01165.pdf
   - ***Skills Required:*** Good C++ coding skill, experience in Deep Learning area, experience in Computer Vision area

- ***Mentors:*** Tyan Vladimir
- ***Difficulty:*** Medium

6. ***IDEA:*** **Deep learning based alpha matting**
   - ***Description:*** Project aims at generating a large training dataset for alpha matting, training the model and converting the model to ONNX to be used with OpenCV's DNN module.
   - ***Expected Outcomes:***
     - A large dataset which can be used to train models of alpha matting.
     - Implement the paper: "Deep Image Matting" by Ning Xu et al.
     - Trained model by the use of the generated dataset
     - Experiments comparing results to existing alpha matting algorithms at alphamatting.com
     - Convert model to ONNX and provide a running example in OpenCV's DNN module
     - Resources
       - alphamatting.com Comparison of many methods, datasets etc
   - ***Skills Required:*** Excellent C++ and Python coding skills, Experience in training deep learning models in Tensorflow/Pytorch, Experience in Computer Vision, Experience in Deep Learning
   - ***Mentors:*** Sunita Nayak
   - ***Difficulty:*** Medium-Hard

7. ***IDEA:*** **Computer Vision based Alpha Matting**
   - ***Description:*** Project aims to integrate some of the best computer vision based best alpha matting algorithms into OpenCV.
   - ***Expected Outcomes:***
     - Two different kinds of alpha matting options in OpenCV
     - Experiments comparing results to existing alpha matting algorithms at alphamatting.com
     - Implement the paper: "A Global Sampling Method for Alpha Matting" by Kaiming He et al.
     - Implement the paper: "Designing Effective Inter-Pixel Information Flow for Natural Image Matting" by Yagiz et. al."
     - Resources
       - alphamatting.com Comparison of many methods, datasets etc
   - ***Skills Required:*** Excellent C++ and Python coding skills, Experience in Computer Vision
   - ***Mentors:*** TBD
   - ***Difficulty:*** Medium-Hard

8. ***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:*** Easy/Medium

9. ***IDEA:*** **Add training to OpenCV DNN module**
    - ***Description:*** Current implementation of OpenCV DNN allows only CPU inference (forward pass). This project aims to add standard backpropagation training algorithm with some of state-of-art optimizers to OpenCV DNN module.
    - ***Expected outcomes:***
        - Expand OpenCV DNN module implementing standard backpropagation training algorithm with simple optimizer (for example: SGD with or w/o moments) on CPU for all basic layers
        - Build a topology of some state-of-art network (AlexNet) and train it from the scratch.
        - Training in batches option implementation
    - Additional goals
        - Implement training on GPU
        - Add more optimizers (Adam, Nesterov etc.)
        - Add training support of all layers in OpenCV DNN
        - Resources
            - http://yann.lecun.com/exdb/publis/pdf/lecun-88.pdf
            - https://arxiv.org/pdf/1609.04747.pdf
    - ***Skills Required:***
        - Very high C++ coding skill, very good understanding of Deep Learning theory under the hood, basic Computer Vision knowledge
    - ***Mentors:*** Tyan Vladimir
    - ***Difficulty:*** Hard

4. Area: Differential operators

1. ***IDEA:*** **Point Coordinate regression and DSAC.**
    - ***Description:*** The task is to Implement a differentiable point coordinate regression and differentiable RANSAC into PyTorch Geometry. Open call to propose a differentiable decomposeHomography and help functions for R and T. Context: PyTorch Geometry is a partial reimplementation of OpenCV operators in a differentiable setup.
    - ***Expected Outcomes:***
        - Define and implement an API for proposed methods into PyTorch Geometry.
        - Define and implement DSAC as a generic framework (if possible) to the model (homography, pose, line, etc).
        - Provide unit test, gradient check tests.

        - Write sphinx documentation.
        - Write examples and notebook tutorials.
        - Resources
            - PyTorch Geometry https://github.com/arraiy/torchgeometry
            - point coordinate regression DSNT https://arxiv.org/abs/1801.07372 https://github.com/anibali/dsntnn
            - differentiable randsac dsac https://github.com/vislearn/DSACLine

    - ***Skills Required:*** experience in geometry, computer vision and Python. You must know how to code using PyTorch and understanding about deep neural networks and the autograd engine.
    - ***Mentors:*** Clément Pinard http://perso.ensta-paristech.fr/~pinard/#
    - ***Difficulty:*** Medium/Hard

2. ***IDEA:*** **Create a differentiable rendering module "torchgeometry.render"**
    - ***Description:*** The task is to create a dedicated module into the PyTorch Geometry package based on the existing Neural 3D Mesh Renderer framework. Explore the framework and propose a solid API with all the operators in order to provide as much as possible generic operators to solve computer graphics problems. Context: PyTorch Geometry is a partial reimplementation of OpenCV operators in a differentiable setup.
    - ***Expected Outcomes:***
        - Create a new module in torch geometry and propose an extens API based on the existing code.
        - Provide unit test, gradient check tests.
        - Write sphinx documentation.
        - Write examples and notebook tutorials.
        - Resources
            - PyTorch Geometry https://github.com/arraiy/torchgeometry
            - https://github.com/daniilidis-group/neural_renderer

    - ***Skills Required:*** experience in computer graphics, computer vision and Python. You must know how to code using PyTorch and understanding about deep neural networks and the autograd engine.
    - ***Mentors:*** Anatoly Baksheev
    - ***Difficulty:*** Medium/Hard

3. ***IDEA:*** **Differentiable Image Processing**
    - ***Description:*** Expand the existing PyTorch Geometry image processing module "torchgeometry.image". The idea is to analyze what in OpenCV can be reimplemented in PyTorch in a differentiable manner and coexist with the framework. This will allow to run end-to-end Computer Vision pipelines reusing the computational graph, optimize with JIT or Tensor Comprehensions, run in distributed, etc. Context: PyTorch Geometry is a partial reimplementation of OpenCV operators in a differentiable setup.
    - ***Expected Outcomes:***
        - Explore the OpenCV API and propose what methods can be reimplemented fulfilling the above requirements.
        - Explore what's in tensorflow image that can be used as a base API.
        - Match as much as possible OpenCV API.
            - Some ideas: sobel, laplacian filters, morphologic operators, color space conversions, etc.

        - Provide unit test, gradient check tests.

        - Write sphinx documentation.
        - Write examples and notebook tutorials.
        - Resources
            - PyTorch Geometry https://github.com/arraiy/torchgeometry
            - https://torchgeometry.readthedocs.io/en/latest/image.html
            - https://docs.opencv.org/master/d7/dbd/group__imgproc.html
    - ***Skills Required:*** experience in computer vision, and Python. You must know how to code using PyTorch and understanding about deep neural networks and the autograd engine.
    - ***Mentors:*** Edgar Riba
    - ***Difficulty:*** Medium/Hard

5. **Area: Fiducials**

    1. ***IDEA:* Add April Tag Fiducial detection to OpenCV**
        - ***Description:*** April Tags are fiducial tabs that allow a geometric relation to camera pose via the 4 corners of the tags while yielding over 500 to several thousand uniquely readable tag patterns with error detecting codes. They are used extensively in camera calibration, object tracking, and robotics. They can be slow and hard to detect however. Add April Tags to OpenCV in a highly detectable and computationally efficient way by training MNet to recognize the tags and their corners and then using the April net algorithm at only one scale to find and confirm the ID of the tag.
        - ***Expected Outcomes:***
            - Train MNet to segment April tags to high performance under many views and lightings (Mentor will help with this)
                - It will detect corners in clockwise manner (even if 30% occluded)
                - It will detect ID even if 30% occluded
            - Move network into DNN
            - Use MNet to detect the ID and corners of tags, resize the tag to a canonical size and pass the corner locations and ID to the April Tag algorithm.
            - Modify the April Tag algorithm to take advantage of just one detection at a standard size and orientation together with an expected ID.
            - Resources
                - April Tag paper
                - April Tags are described here
                - April Tag C++ Code
                - Academic Paper describing MNet
        - ***Mentors:*** Gary Bradski
        - ***Skills Required:*** Must and have experience using C++, Python and deep nets.
        - ***Difficulty:*** Hard. Must be able to train deep nets and use ONNX or other to port to DNN curated nets and then add a method for feeding in detected tags to the April tag algorithm (ported to OpenCV).

6. **Area: Geometric Vision**

    1. ***IDEA:* Create a Consolidated Optical Flow Module "optflow"**
        - ***Description:*** Create a dedicated `optflow` module in the main repository consolidating the best available (within the library) collection of sparse and dense optical flow algorithms. We have LK optical flow, various sparse algorithms and the classical Farneback

algorithm along with the better DIS optical flow. There may be some deep optical flow that's available in OpenCV dnn samples.

- ***Expected Outcomes:***
  - rename the optflow module in opencv_contrib to `optical_flow_experimental`
  - Select the best optical flow modules (move those out of the former optflow in opencv_contrib)
  - Make a base class with standard method from which all optical flow estimators will derive. Some attempts to do that can already be found in the video module
  - Even though we consolidate API and move to the uniform object-oriented API, we need to keep the old API for backward compatibility (e.g. `cv::computeOptFlowFarneback` etc.).
  - Resources
    - The feature request for this
    - The optical flow feature request
    - The optflow module in opencv_contrib that includes some new optical flow algorithms.
- ***Skills Required:*** 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.
- ***Mentors:*** Vadim Pisarevsky
- ***Difficulty:*** Medium (if you've worked with vision/opencv before)

7. **Area: Programming Language/Web**

   1. ***IDEA:*** **Improve and Expand Python Version of OpenCV**
      - ***Description:*** The task is to create wrappers / code / tutorials for important modules and update / improve / expand existing tutorials for Python. This includes improving the documentation of newly added modules if required.
      - ***Expected Outcomes:***
        1. Add Python Wrappers for functions not present.
        2. Add python code for samples written in C++ only . C++ Samples , Python Samples
        3. Add tutorials for which Python code is already present as in these python samples
        4. Combine Tutorials for C++ / Python etc into single directory and format the tutorial as shown in this tutorial. Details of documentation are present here.
      - ***Skills Required:*** Mastery of C++ and python. Experience of creating python interface to C++ code. Fluent in written English.
      - ***Mentors:*** Ankit Sachan
      - ***Difficulty:*** Medium

   2. ***IDEA:*** **Improve the performance of JavaScript version of OpenCV (OpenCV.js)**
      - ***Description:*** OpenCV.js makes OpenCV functions available to JavaScript developers by compiling its C++ implementation to WebAssembly (or Asm.js) and exposing them through a JavaScript binding. However the current version of OpenCV.js only runs in single thread and doesn't use SIMD instructions. This causes the performance of OpenCV.js is far below the native version and prevents some real-time use cases to be deployed in web browser. This task aims to improve the performance of OpenCV.js by exploiting parallelism of CPU. This requires to port the parallel implementations of OpenCV source code to web by leveraging Threads and SIMD features of WebAssembly.

- ***Expected Outcomes:***
  - Speedup OpenCV.js by multi-threading
  - Speedup OpenCV.js by SIMD (using WASM)
  - Create OpenCV.js benchmark and profile the performance on multiple platforms
  - Create a real-time demo by leveraging parallel version of OpenCV.js
  - Resources
    - OpenCV.js paper (including earlier experiments for parallelism by Web Worker and SIMD.js
    - WebAssembly Threads
    - WebAssembly SIMD
    - JavaScript module
    - OpenCV.js tutorials
- ***Skills Required:*** C++, JavaScript, HTML, WebAssembly, performance optimization, multi-threading and SIMD. Fluent in written English.
- ***Mentors:*** Ningxin Hu
- ***Difficulty:*** Medium

8. **Area: SLAM / Camera Pose / RGBD / 3D reconstruction**

   1. ***IDEA:*** **KinectFusion improvements**

      - ***Description:*** We have a KinectFusion 3d reconstruction algorithm which works reasonably well. However its practical use is limited by the list of reasons:

        - its space is limited and cannot extend as the camera moves
        - (connected to previous one) its space representation consumes too much memory
        - it cannot process non-rigid scenes like with living people or animals (even slight moves leaves running average artifacts)

        That's why there are 2 main ways of improvement:

        - Large scale implementations of the algorithm like Kintinuous or KinFu Large Scale
        - Implementing DynamicFusion which handles non-rigid scenes (seems more preferable because of already existing code from GSoC 2017) It's up to participant what way to choose.

      - ***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 as a separate class based on existing KinFu class. The set of methods may be extended by the ones to export resulting 3d model in any form (at least points+normals).

        - Resources

            - DynamicFusion which is able to handle non-rigidness by introducing local transformations
            - 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
            - previous attempt to make DynamicFusion
            - Datasets for non-rigid scene
    - ***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:*** from Hard to Extreme (based on the fact that there was one unsuccessful attempt in the past GSoC)

2. ***IDEA: This is a placeholder:*** **We may add an idea for Mobile SLAM 2/13/2019**
    - ideas from http://imagine.enpc.fr/~moulonp/openMVG/coreFeatures.html

9. **Area: Tracking, Human Tracking, and Segmentation**

1. ***IDEA:*** **Facial Landmark Detector**
    - ***Description:*** Facial feature detection and tracking is a high value area of computer vision since humans are interested in what humans are paying attention to, feeling, and enhancing face pictures in selfies etc. At this point, we feel this should just become a standard "built in" ability that computer vision users can just call/rely on. OpenCV already has some code available on for Facial Landmark detection, see Tutorial on Facial Landmark Detector API and Tutorials for face module, but much progress has been made that we want to make available. The task is to create a Facial Landmark detector model with the following requirements:
    - ***Expected Outcomes:***
        - For mobile, make available a smaller model with lesser points (e.g. 5 instead of 68)
        - Run at better Accuracy than Dlib
        - Face Stabilization (filtering and optical flow)
        - Add it to the Python API
        - Resources
            - https://docs.opencv.org/4.0.0/d5/d47/tutorial_table_of_content_facemark.html
            - https://docs.opencv.org/4.0.0/de/d27/tutorial_table_of_content_face.html
            - https://github.com/opencv/opencv_contrib/tree/master/modules/face
    - ***Skills Required:*** mastery level C++ and Python, college course work in Computer Vision that includes work on face features. Best if you have worked with deep networks before.
    - ***Mentors:*** Satya Mallick
    - ***Difficulty:*** Medium

2. ***IDEA:*** **A universal boosted cascades interface**
    - ***Description:*** Boosted cascades, before the era of deep learning, have long been the state-of-the-art for object detection and decision learning. When moving to OpenCV4 future releases, the plan is to remove them because of the old C-API backend and the tons of issues with the current implementation. This project will focus on re-implementing the boosted cascades, through an open and universal interface, that will allow applying

these techniques, combined with newer feature generation algorithms like for example CNNS.

- ***Expected Outcomes:***
  - Re-implement the boosted cascades / softcascades basic algorithm into OpenCV4.0
  - Provide universal feature adding functionality: more than the previous HAAR/LBP/HOG
  - Provide training, detection and evaluation interfaces
  - Resources
    - https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html
    - https://github.com/opencv/opencv/tree/master/apps
- ***Skills Required:*** efficient in C++ coding, college course work in classic computer vision, mediate experience in deep learning can help to include them as feature generators
- ***Mentors:*** Steven Puttemans
- ***Difficulty:*** Medium

10. **Area: Tutorials**

    1. ***IDEA:*** **General Tutorials with Video**
       - ***Description:*** Videos have proven to be one of the most effective and engaging tools for learning.

In this GSoC project, we will be adding videos to the existing OpenCV tutorials. The goal of these videos is to introduce the Computer Vision theoretical concepts behind each tutorial. We will provide our users with a better understanding of the theory and hopefully they will achieve better results when implementing their own projects. * Additionally, we aim for our documentation to be versatile, hence, a second goal of this project is to keep working towards multi-language tutorials (C++, Python, Java, ...) and work on the associated tools. * ***Expected Outcomes:*** 1. Add 1 youtube video to each tutorial (set of tutorials to be defined) 1. Add Java and Python version to the currently C++ only tutorials 1. Add quick navigation (previous/next tutorial) to all the tutorials 1. Contribute to the merge of the python documentation with the general one * ***Skills Required:*** C++, Python, Java, Computer Vision, English. * ***Mentors:*** Joao Cartucho, Steven Puttemans *(if more than one student)* * ***Difficulty:*** Medium

1. ***IDEA:*** **Improve Machine Learning Tutorials and Documentation**
   - ***Description:*** The **D**eep **N**eural **N**etwork (**DNN**) module is new and light on good tutorials. The **M**achine **L**earning (**ML**) Module is old and needs a tutorial refresh. The task is to create tutorials (code example applications and describe) for the machine learning/deep neural network modules and to update / improve / expand existing tutorials (Mostly in C++) for the main OpenCV module as well as some opencv_contrib modules. This includes improving the documentation of newly added modules if required.

   - ***Expected Outcomes:***

     1. Write tutorials for DNN Module.
        1. Code examples and create a tutorial on how the DNN module is structured and create a video explaining the important functions in the DNN module.
        2. Add code samples and write-ups on the usage of ONNX models.
        3. Search for real-time state-of-the-art models and port it to OpenCV.
     2. Improve and expand write-ups / videos for sample code already present in the repository but no tutorials. E.g.:
        1. DNN

            1. Image Classification using Tensorflow
            2. Object Detection using Tensorflow API
            3. Semantic Segmentation
            4. Image Colorization
            5. Neural Style Transfer
            6. OpenPose
        2. C++ code samples present in opencv repo. e.g.
            1. DIS Optical Flow
    3. Add code samples and write-ups for using different algorithms e.g.
        1. ML module
            1. Normal Bayes Classifier
            2. Decision Trees
            3. Random Forests
            4. ANN MLP
            5. K-NN
            6. K-Means
        2. Text Module in OpenCV Contrib
    4. Tutorial Series on using OpenVINO models and DLDT toolkit.

- ***Skills Required:*** Excellent written English, Mastery level C++ knowledge, college course work in Computer Vision, Python knowledge. Working knowledge of deep neural networks.
- ***Mentors:*** Vikas Gupta
- ***Difficulty:*** Easy

2. **Area: User Interface**

    1. ***IDEA:* Improve the OpenCV user interface**
        - ***Description:*** HighGUI is OpenCV's graphical user interface including event triggering. It's cool, but it's old and needs revamping. HighGUI has been improved by using a QT interface at the cost of a large library to link in. We want to replace the QT functionality (mainly zoom to pixel and snapshot) with custom code. We'd also like to add native buttons.
        - ***Expected Outcomes:***
            - Drop QT dependency but retain at least zoom to pixel and snapshot capability with custom code
            - Add buttons as a native capability
            - Allow setting up an adaptive array of display windows w/in one larger window.
            - Retain ability for Python to call HighGUI
            - Resources
                - High GUI documentation
                - High GUI tutorial
        - ***Skills Required:*** C++, Python, user interface experience
        - ***Mentors:*** Vadim Pisarevsky
        - ***Difficulty:*** Easy if you are well familiar with GUI, Medium if not.

- ## 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.

General Information:

▶ Join our OpenCV GSoC 2019 Mailing List ◀

- Program Site for GSoC 2019
- Mailing list for OpenCV GSOC 2019: `opencv-gsoc-2019@googlegroups.com`
- IRC Channel: `#opencv` on freenode
- **Timelines**
    - Timeline for GSoC 2019

For computer vision professionals interested in mentoring

1. Contact us by March 25th 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

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 April 9th.
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.

Staff

Mentors

Mentors are encouraged to look at the ideas list above and choose from that. If you have a specific idea that you are passionate about, then use the github Markdown template below and submit to the mentor's mailing list so that we can see if it should be posted.

```
### Template
1. ### <Descriptive Category such as "Deep Nets">
   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. >
      * ***Mentors:*** < your name goes here >
      * ***Difficulty:*** <Easy, Medium, Hard>
```

Mentors

```
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
```