OpenCV 3D module

- Authors: Rostislav Vasilikhin, Vadim Pisarevsky, Mihai Bujanca
- Link: #18267
- Status: **Draft**
- Platforms: **All**
- Complexity: **TBD** (estimated complexity; certain amount of man-weeks, man-months etc.)

Introduction and Rationale

While a lot of 2D tasks in computer vision are now successfully solved by deep neural networks, 3D-related problems are still an area where traditional algorithms can be more successful. While there are a lot of 3D algorithms in OpenCV already, they are spread around the library between modules of both main and contrib repositories.

OpenCV community could benefit a lot if a modern OpenCV module covering various 3D vision problems is added to the main repository.

Name

Here are some possible options for the module and namespace names:

- module= `opencv_d3` , namespace= `cv::d3` (used in this document)
- module= `opencv_3d` , namespace= `cv::d3` or `cv::_3d`
- module= `opencv_3d` , namespace= `cv3d` (that is, a parallel to `cv` namespace is suggested here; the shortest to type, but maybe inconvenient a bit)
- module= `opencv_geom` , namespace= `cv::geom` (shortcut for "geometry")
- module= `opencv_space` , namespace= `cv::space` etc.

Proposed Functionality

Several algorithm families should be joined in the module:

- KinectFusion and RGBD odometry
- SLAM and SFM
- Open3D Integration
- Quaternion Support
- Split `calib3d` Module
- New RANSAC Framework
- Plane/Shape Fitting
- Point Cloud Simplification
- Import/Export for popular formats
- 3D sensors support
- Visualization and OpenGL support
- Some algorithms: octree, filtering, keypoints, features, matching, segmentation, etc.
- ther contrib modules integration

That's a lot of functionality. The goal for OpenCV 5.0 is to introduce a reasonable API and put all the code together. The actual implementations could be polished and replaced with better ones in further OpenCV 5.x, 6.x etc. releases.

Details

The module will be put to the “next” branch of OpenCV, i.e. the branch for OpenCV 5.x. All the functionality will be put into `cv::d3` namespace, however some existing functions, extracted from `calib3d` module, will also be put into cv namespace to provide backward compatibility.

The `opencv_d3` module will depend on `opencv_core`, maybe on `opencv_imgproc`.

The key data structures (matrices, point sets etc.) will be using C++ and OpenCV standard structures: `cv::Mat`, `std::vector<Vec3f>` etc. The data structure should be compatible with DNN module. The 3D data can be input to DNN models directly.

Let’s now discuss the planned functionality a bit.

KinectFusion and RGBD odometry

Surface normal calculation and ICP algorithms can be taken from the opencv_rgbd module. The module contains several ICP algorithms with such features as RGB matching, separate rotation/translation optimization, tunable parameters and so on. They are based on various articles including KinectFusion.

Talking about KinectFusion, it is patented, so we’d better not move the whole pipeline into the main repo.

As a solution, we can provide a user with KinectFusion parts such as ICP, TSDF and depth preprocessing. Moreover, there are already publicly available algorithms in `rgbd`:

- several variants of ICP with such features as RGB matching, separate rotation/translation optimization, tunable parameters and so on. They are based on various articles including KinectFusion.
- 2 variants of TSDF:
  - TSDFVolume from KinFu, stable and fast
  - HashTSDF from LargeScaleKinFu, based on Voxel Hashing / InfiniTAM (GSoC-2020), slow and not so stable yet, but accuracy/perf tests writing is in progress now
- DynamicFusion: slow and buggy, in the middle of remaking. Comes as a whole pipeline, has a little sense to be decomposed into parts.

SLAM and SFM

SFM module in its current state is hard to build (a lot of 3rdparty dependencies), it has poor accuracy and it’s based on outdated library `libmv` - an old library from Blender, w/o updates in repo since 2017. It's better to implement it from scratch, for example, based on ORB SLAM or something else.

A SLAM system requires some sparse optimization, for example, for loop closure on large scenes.

- There is a bundle adjustment in the `stitching` module but I’m not sure if the way it’s done makes it applicable for that task.
- Also, there will be a pose graph in Large Scale Depth Fusion (GSoC-2020). Loop closure will be supported but is not planned yet.

Sparse optimization should go together with other 3D algorithms.

Bridge to Open3D

TBD, needs discussion

Open3D is pretty cool stuff! There are a lot of pipeline-ready algorithms including all parts of KinectFusion, a lot of samples, tutorials and so on. It's very simple to build a SLAM system based on that. Works quite fast for CPU-only code but lacks acceleration like SIMD or OpenCL, as a result it’s not suitable for real time tasks.

On the other side, `rgbd` module has KinectFusion which works real time with the help of OpenCL but it lacks flexibility of Open3D. Can these two ones be joined together?

Quaternion Support

Quaternions are such a must-have in computer vision that they were independently implemented in different parts of OpenCV but none of that implementation still has no public interface. The proposal is to create the one based on the upcoming DynamicFusion implementation.

There is an unmerged source with such features as:

- Special classes for quaternion of arbitrary norm and of unit norm (which represent rotations and can make some math faster)
- Dual quaternions and unit dual quaternions classes for the same reason
- Required tools: to/from affine transformations, exp, log, operator overloading, etc.
- Derivatives
- DQB for pose averaging

This can be augmented by more useful features like templated data type, splines and SLERP, and add some tests on all that functionality.

Part of opencv_calib3d module

Geometry-related part is to be put into the 3D module. Calibration-related part should be extended with `opencv_contrib/ccalib`, `opencv_contrib/mcc` and other such modules.

New RANSAC Framework

Related PR: https://github.com/opencv/opencv/pull/17683

Functions based on that like `find{Homography|FundamentalMat|EssentialMat}`, `solvePnP`, `triangulate`, etc. will be one of the cornerstones of the new module

Maksym Ivashechkin: I want to suggest two small things which can be useful

- As I see in OpenCV is currently implemented linear triangulation method. Another option is to have non-linear triangulation, which would be a slower but more precise. For example, when someone would need to have a better reconstruction regardless of time.
- It is also good to have multi-model RANSAC. I don't have much experience with it but there are already methods which allow finding multiple models without explicitly running several times the same RANSAC and removing inliers. It can be useful for multi-plane or multi-homography fitting.

Plane/Shape Fitting

https://github.com/opencv/opencv_contrib/pull/2584/files

Point Cloud Simplification

TBD by Mihai Bujanca OSPP-2020 project

Import/Export for popular formats

TO BE DISCUSSED

`opencv_d3` should be able to read/write popular point cloud/mesh formats. The formats to support are:

- PLY, OBJ (partially implemented in opencv_viz)
- OpenVDB (?) for TSDF export

3D sensors support

Nowadays there is a growing number of 3D sensors, lidars, ToF sensors, pure stereo cameras or the ones equipped with structured light etc. And because of autonomous cars, robots, drones etc. the interest is huge. We may put some initial functionality to process point clouds from such sensors into `opencv_d3`. At the same time, it's planned to gradually extend `opencv_videoio` to grab/integrate data from such 3D sensors.

Visualization and OpenGL support

OpenGL support (not just interoperability as it's now) is highly required for rendering-based algorithms such as DynamicFusion. Visualization can be done in the way it's done in `viz` module but w/o VTK as a backend. Probably, visualization will be put into a separate module, not `opencv_d3`.

Other contrib modules integration

Several 3D-related things are to be left in contrib repo:

- `cnn_3dobj`: CNN for 3D object classification and pose estimation
  - according to @paroj not a good thing to be included in 3D module (link)
  - no reasons claimed to include this algorithm to 3D module
- `surface_matching`:
  - patent problems