OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposed to the C-based OpenCV 1.x API (C API is deprecated and not tested with "C" compiler since OpenCV 2.4 releases)

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- Core functionality (**core**) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (**imgproc**) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Image file reading and writing (**imgcodecs**) - includes functions for reading and writing image files in various formats.
- Video I/O (**videoio**) - an easy-to-use interface to video capturing and video codecs.
- High-level GUI (**highgui**) - an easy-to-use interface to simple UI capabilities.
- Video Analysis (**video**) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- Camera Calibration and 3D Reconstruction (**calib3d**) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (**features2d**) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (**objdetect**) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- Deep Neural Network module (**dnn**) - Deep Neural Network module.
- Machine Learning (**ml**) - The Machine Learning module includes a set of classes and functions for statistical classification, regression, and clustering of data.
- Computational Photography (**photo**) - advanced photo processing techniques like denoising, inpainting.
- Images stitching (**stitching**) - functions for image stitching and panorama creation.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

-
- Main modules:
  - core. Core functionality
  - imgproc. Image Processing

    - imgcodecs. Image file reading and writing
    - videoio. Video I/O
    - highgui. High-level GUI
    - video. Video Analysis
    - calib3d. Camera Calibration and 3D Reconstruction
    - features2d. 2D Features Framework
    - objdetect. Object Detection
    - dnn. Deep Neural Network module
    - ml. Machine Learning
    - flann. Clustering and Search in Multi-Dimensional Spaces
    - photo. Computational Photography
    - stitching. Images stitching
    - gapi. Graph API
- Extra modules:
    - alphamat. Alpha Matting
    - aruco. Aruco markers, module functionality was moved to objdetect module
    - bgsegm. Improved Background-Foreground Segmentation Methods
    - bioinspired. Biologically inspired vision models and derivated tools
    - cannops. Ascend-accelerated Computer Vision
    - ccalib. Custom Calibration Pattern for 3D reconstruction
    - cudaarithm. Operations on Matrices
    - cudabgsegm. Background Segmentation
    - cudacodec. Video Encoding/Decoding
    - cudafeatures2d. Feature Detection and Description
    - cudafilters. Image Filtering
    - cudaimgproc. Image Processing
    - cudalegacy. Legacy support
    - cudaobjdetect. Object Detection
    - cudaoptflow. Optical Flow
    - cudastereo. Stereo Correspondence
    - cudawarping. Image Warping

- cudev. Device layer
- cvv. GUI for Interactive Visual Debugging of Computer Vision Programs
- datasets. Framework for working with different datasets
- dnn_objdetect. DNN used for object detection
- dnn_superres. DNN used for super resolution
- dpm. Deformable Part-based Models
- face. Face Analysis
- fastcv. Module-wrapper for FastCV hardware accelerated functions
- freetype. Drawing UTF-8 strings with freetype/harfbuzz
- fuzzy. Image processing based on fuzzy mathematics
- hdf. Hierarchical Data Format I/O routines
- hfs. Hierarchical Feature Selection for Efficient Image Segmentation
- img_hash. The module brings implementations of different image hashing algorithms.
- intensity_transform. The module brings implementations of intensity transformation algorithms to adjust image contrast.
- julia. Julia bindings for OpenCV
- line_descriptor. Binary descriptors for lines extracted from an image
- mcc. Macbeth Chart module
- optflow. Optical Flow Algorithms
- ovis. OGRE 3D Visualiser
- phase_unwrapping. Phase Unwrapping API
- plot. Plot function for Mat data
- quality. Image Quality Analysis (IQA) API
- rapid. silhouette based 3D object tracking
- reg. Image Registration
- rgbd. RGB-Depth Processing
- saliency. Saliency API
- sfm. Structure From Motion
- shape. Shape Distance and Matching
- signal. Signal Processing

- stereo. Stereo Correspondance Algorithms
- structured_light. Structured Light API
- superres. Super Resolution
- surface_matching. Surface Matching
- text. Scene Text Detection and Recognition
- tracking. Tracking API
- videostab. Video Stabilization
- viz. 3D Visualizer
- wechat_qrcode. WeChat QR code detector for detecting and parsing QR code.
- xfeatures2d. Extra 2D Features Framework
- ximgproc. Extended Image Processing
- xobjdetect. Extended object detection
- xphoto. Additional photo processing algorithms

Detailed Description

Namespaces

namespace	cv::traits

Classes

class	cv::_InputArray This is the proxy class for passing read-only input arrays into OpenCV functions. More...
class	cv::_InputOutputArray
class	cv::_OutputArray This type is very similar to InputArray except that it is used for input/output and output function parameters. More...
class	cv::Algorithm

		This is a base class for all more or less complex algorithms in OpenCV. More...
class	**cv::Complex< _Tp >**	A complex number class. More...
class	**cv::DataDepth< _Tp >**	A helper class for **cv::DataType**. More...
class	**cv::DataType< _Tp >**	Template "trait" class for OpenCV primitive data types. More...
class	**cv::DMatch**	Class for matching keypoint descriptors. More...
class	**cv::Formatted**	
class	**cv::Formatter**	
class	**cv::KeyPoint**	Data structure for salient point detectors. More...
class	**cv::Mat**	n-dimensional dense array class More...
class	**cv::Mat_< _Tp >**	Template matrix class derived from **Mat**. More...
class	**cv::MatAllocator**	Custom array allocator. More...

class	**cv::MatCommaInitializer_< _Tp >** Comma-separated Matrix Initializer. More...
class	**cv::MatConstIterator**
class	**cv::MatConstIterator_< _Tp >** Matrix read-only iterator. More...
class	**cv::MatExpr** Matrix expression representation This is a list of implemented matrix operations that can be combined in arbitrary complex expressions (here A, B stand for matrices (**cv::Mat**), s for a **cv::Scalar**, alpha for a real-valued scalar (double)): More...
class	**cv::MatIterator_< _Tp >** Matrix read-write iterator. More...
class	**cv::MatOp**
struct	**cv::MatSize**
struct	**cv::MatStep**
class	**cv::Matx< _Tp, m, n >** Template class for small matrices whose type and size are known at compilation time. More...
class	**cv::NAryMatIterator** n-ary multi-dimensional array iterator. More...

struct	cv::ParamType< _Tp, _EnumTp >
struct	cv::ParamType< _Tp, typename std::enable_if< std::is_enum< _Tp >::value >::type >
struct	cv::ParamType< Algorithm >
struct	cv::ParamType< bool >
struct	cv::ParamType< double >
struct	cv::ParamType< float >
struct	cv::ParamType< Mat >
struct	cv::ParamType< Scalar >
struct	cv::ParamType< std::vector< Mat > >
struct	cv::ParamType< String >
struct	cv::ParamType< uchar >
struct	cv::ParamType< uint64 >
struct	cv::ParamType< unsigned >
class	cv::Point3_< _Tp > Template class for 3D points specified by its coordinates x, y and z. More...

class	**cv::Point_< _Tp >**
	Template class for 2D points specified by its coordinates x and y. More...
class	**cv::Range**
	Template class specifying a continuous subsequence (slice) of a sequence. More...
class	**cv::Rect_< _Tp >**
	Template class for 2D rectangles. More...
class	**cv::RotatedRect**
	The class represents rotated (i.e. not up-right) rectangles on a plane. More...
class	**cv::Scalar_< _Tp >**
	Template class for a 4-element vector derived from **Vec**. More...
class	**cv::Size_< _Tp >**
	Template class for specifying the size of an image or rectangle. More...
class	**cv::SparseMat**
	The class **SparseMat** represents multi-dimensional sparse numerical arrays. More...
class	**cv::SparseMat_< _Tp >**
	Template sparse n-dimensional array class derived from **SparseMat**. More...
class	**cv::SparseMatConstIterator**
	Read-Only Sparse Matrix Iterator. More...
class	**cv::SparseMatConstIterator_< _Tp >**
	Template Read-Only Sparse Matrix Iterator Class. More...

class	cv::SparseMatIterator Read-write Sparse Matrix Iterator. More...
class	cv::SparseMatIterator_< _Tp > Template Read-Write Sparse Matrix Iterator Class. More...
class	cv::TermCriteria The class defining termination criteria for iterative algorithms. More...
class	cv::UMat
struct	cv::UMatData
class	cv::Vec< _Tp, cn > Template class for short numerical vectors, a partial case of **Matx**. More...

Typedefs

typedef Complex< double >	cv::Complexd
typedef Complex< float >	cv::Complexf
typedef const _InputArray &	cv::InputArray
typedef InputArray	cv::InputArrayOfArrays
typedef const _InputOutputArray &	cv::InputOutputArray
typedef InputOutputArray	cv::InputOutputArrayOfArrays

typedef Mat_< uchar >	cv::Mat1b
typedef Mat_< double >	cv::Mat1d
typedef Mat_< float >	cv::Mat1f
typedef Mat_< int >	cv::Mat1i
typedef Mat_< short >	cv::Mat1s
typedef Mat_< ushort >	cv::Mat1w
typedef Mat_< Vec2b >	cv::Mat2b
typedef Mat_< Vec2d >	cv::Mat2d
typedef Mat_< Vec2f >	cv::Mat2f
typedef Mat_< Vec2i >	cv::Mat2i
typedef Mat_< Vec2s >	cv::Mat2s
typedef Mat_< Vec2w >	cv::Mat2w
typedef Mat_< Vec3b >	cv::Mat3b
typedef Mat_< Vec3d >	cv::Mat3d
typedef Mat_< Vec3f >	cv::Mat3f

typedef Mat_< Vec3i >	cv::Mat3i
typedef Mat_< Vec3s >	cv::Mat3s
typedef Mat_< Vec3w >	cv::Mat3w
typedef Mat_< Vec4b >	cv::Mat4b
typedef Mat_< Vec4d >	cv::Mat4d
typedef Mat_< Vec4f >	cv::Mat4f
typedef Mat_< Vec4i >	cv::Mat4i
typedef Mat_< Vec4s >	cv::Mat4s
typedef Mat_< Vec4w >	cv::Mat4w
typedef Matx< double, 1, 2 >	cv::Matx12d
typedef Matx< float, 1, 2 >	cv::Matx12f
typedef Matx< double, 1, 3 >	cv::Matx13d
typedef Matx< float, 1, 3 >	cv::Matx13f
typedef Matx< double, 1, 4 >	cv::Matx14d
typedef Matx< float, 1, 4 >	cv::Matx14f

typedef Matx< double, 1, 6 >	cv::Matx16d
typedef Matx< float, 1, 6 >	cv::Matx16f
typedef Matx< double, 2, 1 >	cv::Matx21d
typedef Matx< float, 2, 1 >	cv::Matx21f
typedef Matx< double, 2, 2 >	cv::Matx22d
typedef Matx< float, 2, 2 >	cv::Matx22f
typedef Matx< double, 2, 3 >	cv::Matx23d
typedef Matx< float, 2, 3 >	cv::Matx23f
typedef Matx< double, 3, 1 >	cv::Matx31d
typedef Matx< float, 3, 1 >	cv::Matx31f
typedef Matx< double, 3, 2 >	cv::Matx32d
typedef Matx< float, 3, 2 >	cv::Matx32f
typedef Matx< double, 3, 3 >	cv::Matx33d
typedef Matx< float, 3, 3 >	cv::Matx33f
typedef Matx< double, 3, 4 >	cv::Matx34d

typedef Matx< float, 3, 4 >	cv::Matx34f
typedef Matx< double, 4, 1 >	cv::Matx41d
typedef Matx< float, 4, 1 >	cv::Matx41f
typedef Matx< double, 4, 3 >	cv::Matx43d
typedef Matx< float, 4, 3 >	cv::Matx43f
typedef Matx< double, 4, 4 >	cv::Matx44d
typedef Matx< float, 4, 4 >	cv::Matx44f
typedef Matx< double, 6, 1 >	cv::Matx61d
typedef Matx< float, 6, 1 >	cv::Matx61f
typedef Matx< double, 6, 6 >	cv::Matx66d
typedef Matx< float, 6, 6 >	cv::Matx66f
typedef const _OutputArray &	cv::OutputArray
typedef OutputArray	cv::OutputArrayOfArrays
typedef Point2i	cv::Point
typedef Point_< double >	cv::Point2d

typedef Point_< float >	cv::Point2f
typedef Point_< int >	cv::Point2i
typedef Point_< int64 >	cv::Point2l
typedef Point3_< double >	cv::Point3d
typedef Point3_< float >	cv::Point3f
typedef Point3_< int >	cv::Point3i
template<typename _Tp > using	cv::Ptr = std::shared_ptr<_Tp>
typedef Rect2i	cv::Rect
typedef Rect_< double >	cv::Rect2d
typedef Rect_< float >	cv::Rect2f
typedef Rect_< int >	cv::Rect2i
typedef Scalar_< double >	cv::Scalar
typedef Size2i	cv::Size
typedef Size_< double >	cv::Size2d

typedef Size_< float >	cv::Size2f
typedef Size_< int >	cv::Size2i
typedef Size_< int64 >	cv::Size2l
typedef std::string	cv::String

Enumerations

enum cv::AccessFlag {
  cv::ACCESS_READ =1<<24 ,
  cv::ACCESS_WRITE =1<<25 ,
  cv::ACCESS_RW =3<<24 ,
  cv::ACCESS_MASK =ACCESS_RW ,
  cv::ACCESS_FAST =1<<26
}

enum struct cv::Param {
  cv::Param::INT =0 ,
  cv::Param::BOOLEAN =1 ,
  cv::Param::REAL =2 ,
  cv::Param::STRING =3 ,
  cv::Param::MAT =4 ,
  cv::Param::MAT_VECTOR =5 ,
  cv::Param::ALGORITHM =6 ,
  cv::Param::FLOAT =7 ,
  cv::Param::UNSIGNED_INT =8 ,
  cv::Param::UINT64 =9 ,
  cv::Param::UCHAR =11 ,
  cv::Param::SCALAR =12
}

enum cv::UMatUsageFlags {
  cv::USAGE_DEFAULT = 0 ,
  cv::USAGE_ALLOCATE_HOST_MEMORY = 1 << 0 ,
  cv::USAGE_ALLOCATE_DEVICE_MEMORY = 1 << 1 ,
  cv::USAGE_ALLOCATE_SHARED_MEMORY = 1 << 2 ,

cv::__UMAT_USAGE_FLAGS_32BIT = 0x7fffffff
}

Usage flags for allocator. More...

Functions

template<typename _Tp , int m>	
static double	cv::determinant (const Matx< _Tp, m, m > &a)
template<typename _Tp , typename ... A1>	
static Ptr< _Tp >	cv::makePtr (const A1 &... a1)
InputOutputArray	cv::noArray ()
	Returns an empty InputArray or OutputArray.
template<typename _Tp , int m, int n>	
static double	cv::norm (const Matx< _Tp, m, n > &M)
template<typename _Tp , int m, int n>	
static double	cv::norm (const Matx< _Tp, m, n > &M, int normType)
template<typename _Tp , int cn>	
Vec< _Tp, cn >	cv::normalize (const Vec< _Tp, cn > &v)
template<typename _Tp , int m, int n>	
static bool	cv::operator!= (const Matx< _Tp, m, n > &a, const Matx< _Tp, m, n > &b)
template<typename _Tp , int m, int n, int l>	
static Matx< _Tp, m, n >	cv::operator* (const Matx< _Tp, m, l > &a, const Matx< _Tp, l, n > &b)

template<typename _Tp , int m, int n>

static Vec< _Tp, m > cv::operator* (const Matx< _Tp, m, n > &a, const Vec< _Tp, n > &b)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator* (const Matx< _Tp, m, n > &a, double alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator* (const Matx< _Tp, m, n > &a, float alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator* (const Matx< _Tp, m, n > &a, int alpha)

template<typename _Tp >

Vec< _Tp, 4 > cv::operator* (const Vec< _Tp, 4 > &v1, const Vec< _Tp, 4 > &v2)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator* (const Vec< _Tp, cn > &a, double alpha)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator* (const Vec< _Tp, cn > &a, float alpha)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator* (const Vec< _Tp, cn > &a, int alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator* (double alpha, const Matx< _Tp, m, n > &a)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator* (double alpha, const Vec< _Tp, cn > &a)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator* (float alpha, const Matx< _Tp, m, n > &a)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator* (float alpha, const Vec< _Tp, cn > &a)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator* (int alpha, const Matx< _Tp, m, n > &a)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator* (int alpha, const Vec< _Tp, cn > &a)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > & cv::operator*= (Matx< _Tp, m, n > &a, double alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > & cv::operator*= (Matx< _Tp, m, n > &a, float alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > & cv::operator*= (Matx< _Tp, m, n > &a, int alpha)

template<typename _Tp >

Vec< _Tp, 4 > & cv::operator*= (Vec< _Tp, 4 > &v1, const Vec< _Tp, 4 > &v2)

template<typename _Tp , int cn>

static Vec< _Tp, cn > & cv::operator*= (Vec< _Tp, cn > &a, double alpha)

template<typename _Tp , int cn>

static Vec< _Tp, cn > & cv::operator*= (Vec< _Tp, cn > &a, float alpha) template<typename _Tp , int cn>

static Vec< _Tp, cn > & cv::operator*= (Vec< _Tp, cn > &a, int alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator+ (const Matx< _Tp, m, n > &a, const Matx< _Tp, m, n > &b)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator+ (const Vec< _Tp, cn > &a, const Vec< _Tp, cn > &b)

template<typename _Tp1 , typename _Tp2 , int m, int n>

static Matx< _Tp1, m, n > & cv::operator+= (Matx< _Tp1, m, n > &a, const Matx< _Tp2, m, n > &b)

template<typename _Tp1 , typename _Tp2 , int cn>

static Vec< _Tp1, cn > & cv::operator+= (Vec< _Tp1, cn > &a, const Vec< _Tp2, cn > &b)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator- (const Matx< _Tp, m, n > &a)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator- (const Matx< _Tp, m, n > &a, const Matx< _Tp, m, n > &b)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator- (const Vec< _Tp, cn > &a)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator- (const Vec< _Tp, cn > &a, const Vec< _Tp, cn > &b)

template<typename _Tp1 , typename _Tp2 , int m, int n>

static Matx< _Tp1, m, n > & cv::operator-= (Matx< _Tp1, m, n > &a, const Matx< _Tp2, m, n > &b)

template<typename _Tp1 , typename _Tp2 , int cn>

static Vec< _Tp1, cn > & cv::operator-= (Vec< _Tp1, cn > &a, const Vec< _Tp2, cn > &b)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator/ (const Matx< _Tp, m, n > &a, double alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > cv::operator/ (const Matx< _Tp, m, n > &a, float alpha)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator/ (const Vec< _Tp, cn > &a, double alpha)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator/ (const Vec< _Tp, cn > &a, float alpha)

template<typename _Tp , int cn>

static Vec< _Tp, cn > cv::operator/ (const Vec< _Tp, cn > &a, int alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > & cv::operator/= (Matx< _Tp, m, n > &a, double alpha)

template<typename _Tp , int m, int n>

static Matx< _Tp, m, n > & cv::operator/= (Matx< _Tp, m, n > &a, float alpha)

template<typename _Tp , int cn>

static Vec< _Tp, cn > & cv::operator/= (Vec< _Tp, cn > &a, double alpha)

template<typename _Tp , int cn>	
static Vec< _Tp, cn > &	cv::operator/= (Vec< _Tp, cn > &a, float alpha)
template<typename _Tp , int cn>	
static Vec< _Tp, cn > &	cv::operator/= (Vec< _Tp, cn > &a, int alpha)
static String &	cv::operator<< (String &out, const Mat &mtx)
static String &	cv::operator<< (String &out, Ptr< Formatted > fmtd)
template<typename _Tp , int m, int n>	
static bool	cv::operator== (const Matx< _Tp, m, n > &a, const Matx< _Tp, m, n > &b)
template<typename _Tp >	
static _InputArray	cv::rawIn (_Tp &v)
template<typename _Tp >	
static _InputOutputArray	cv::rawInOut (_Tp &v)
template<typename _Tp >	
static _OutputArray	cv::rawOut (_Tp &v)
static std::string	cv::toLowerCase (const std::string &str)
static std::string	cv::toUpperCase (const std::string &str)
template<typename _Tp , int m, int n>	
static double	cv::trace (const Matx< _Tp, m, n > &a)

Shorter aliases for the most popular specializations of Vec<T,n>

typedef Vec< uchar, 2 >	cv::Vec2b
typedef Vec< uchar, 3 >	cv::Vec3b
typedef Vec< uchar, 4 >	cv::Vec4b
typedef Vec< short, 2 >	cv::Vec2s
typedef Vec< short, 3 >	cv::Vec3s
typedef Vec< short, 4 >	cv::Vec4s
typedef Vec< ushort, 2 >	cv::Vec2w
typedef Vec< ushort, 3 >	cv::Vec3w
typedef Vec< ushort, 4 >	cv::Vec4w
typedef Vec< int, 2 >	cv::Vec2i
typedef Vec< int, 3 >	cv::Vec3i
typedef Vec< int, 4 >	cv::Vec4i
typedef Vec< int, 6 >	cv::Vec6i
typedef Vec< int, 8 >	cv::Vec8i

typedef Vec< float, 2 >	cv::Vec2f
typedef Vec< float, 3 >	cv::Vec3f
typedef Vec< float, 4 >	cv::Vec4f
typedef Vec< float, 6 >	cv::Vec6f
typedef Vec< double, 2 >	cv::Vec2d
typedef Vec< double, 3 >	cv::Vec3d
typedef Vec< double, 4 >	cv::Vec4d
typedef Vec< double, 6 >	cv::Vec6d