



Introduction to OpenCV

Vo Hoai Viet

vhviet@fit.hcmus.edu.vn

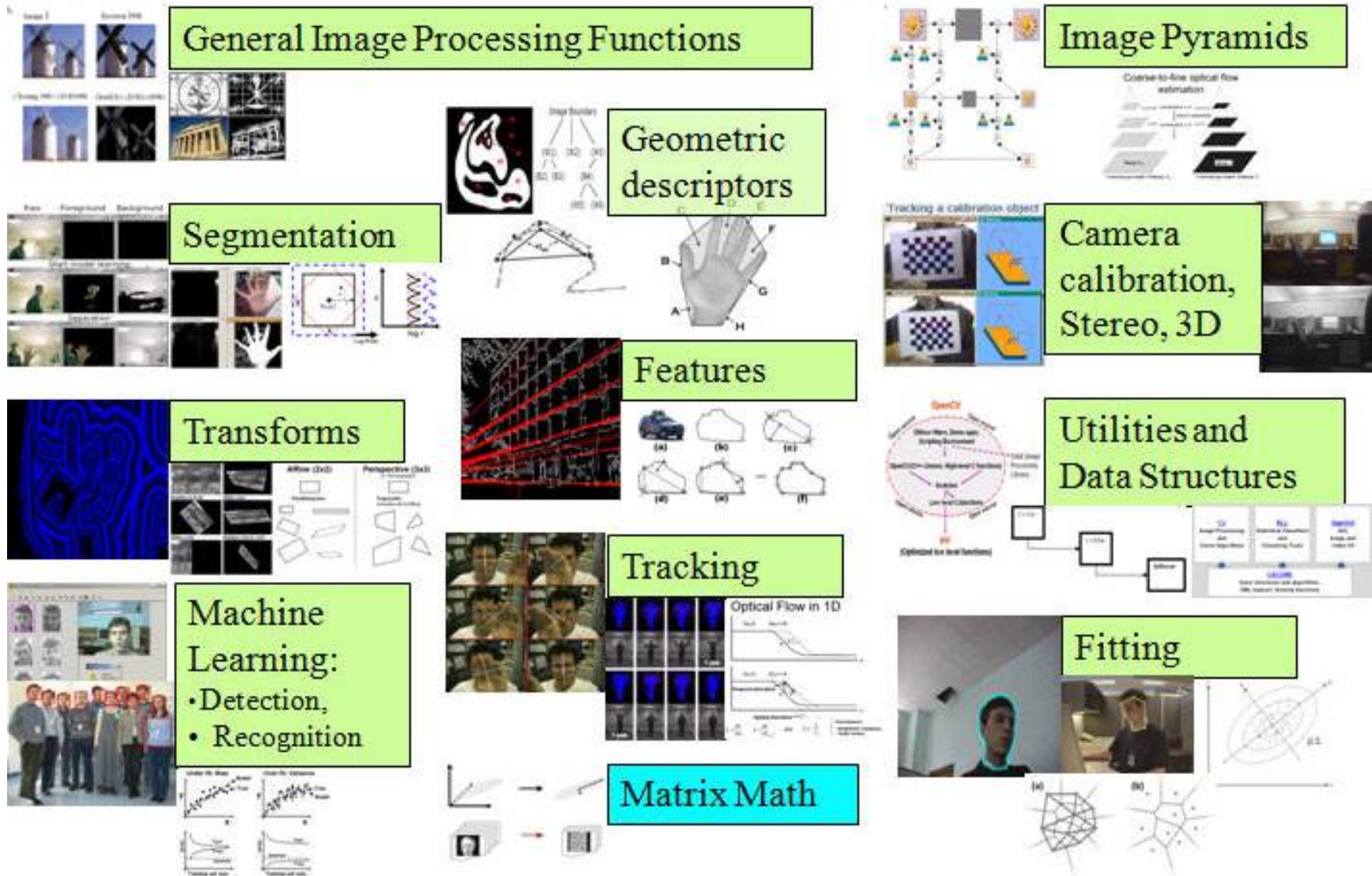
Outline

- What is OpenCV?
- Environment Settings
- Basic structures in OpenCV
- Images Operators
- OpenCV Modules

What is the OpenCV?

- Abbreviation of “Open Source Computer Vision Library” for computer vision and machine learning
- It is released under a BSD license and hence it's free for both academic and commercial use.
- It has C/C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android.
- It was designed for computational efficiency and with a strong focus on real-time applications.

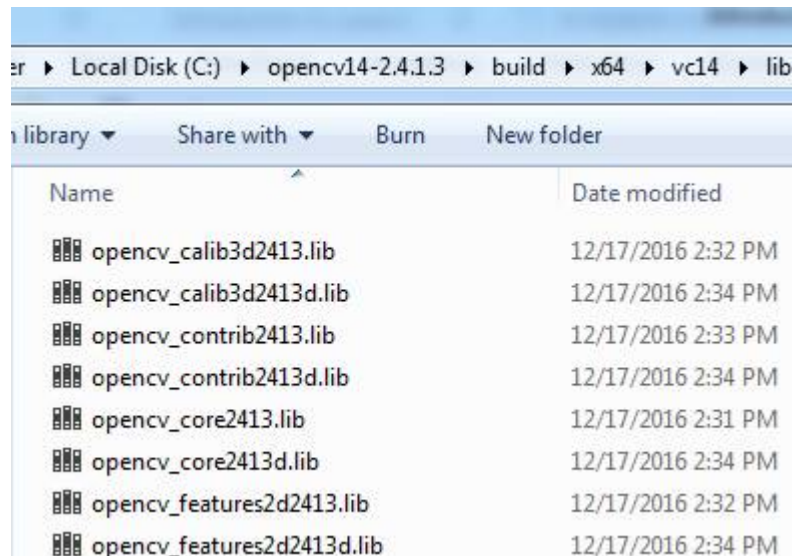
OpenCV Functionality



OpenCV Roadmap

Closing 2.4 series

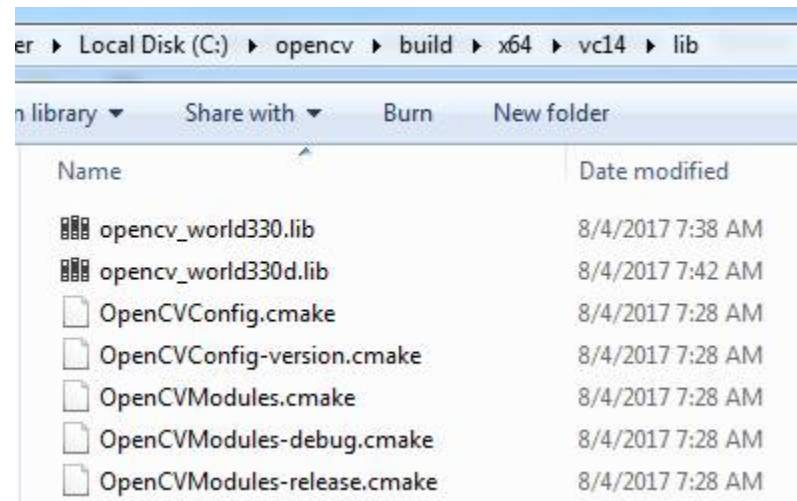
- 2.4.12 in Jul 2015
- 2.4.13 in Apr 2016
- 2.4.13.2 in Dec 2016
- 2.4.13.3 in Aug 2017



Name	Date modified
opencv_calib3d2413.lib	12/17/2016 2:32 PM
opencv_calib3d2413d.lib	12/17/2016 2:34 PM
opencv_contrib2413.lib	12/17/2016 2:33 PM
opencv_contrib2413d.lib	12/17/2016 2:34 PM
opencv_core2413.lib	12/17/2016 2:31 PM
opencv_core2413d.lib	12/17/2016 2:34 PM
opencv_features2d2413.lib	12/17/2016 2:32 PM
opencv_features2d2413d.lib	12/17/2016 2:34 PM

Starting 3.0 series

- 3.0 in Jun 2015
- 3.1 in Dec 2015
- 3.2 in Dec 2016
- 3.3 in Aug 2017



Name	Date modified
opencv_world330.lib	8/4/2017 7:38 AM
opencv_world330d.lib	8/4/2017 7:42 AM
OpenCVConfig.cmake	8/4/2017 7:28 AM
OpenCVConfig-version.cmake	8/4/2017 7:28 AM
OpenCVModules.cmake	8/4/2017 7:28 AM
OpenCVModules-debug.cmake	8/4/2017 7:28 AM
OpenCVModules-release.cmake	8/4/2017 7:28 AM

Introduction to OpenCV

ENVIRONMENT SETTING

Environment Setting

- Download from <http://opencv.org> and compile from source
- Windows: Run executable downloaded from OpenCV website
- Mac OS X: Install through MacPorts/Brew
- Linux: Install through the package manager (e.g. yum, apt) but make sure the version is sufficiently up-to-date for your needs
- OpenCV setting Visual Studio

Introduction to OpenCV

BASIC STRUCTURES

Basic OpenCV Structures

- Mat - image object
- Point - 2D Point
- Size - 2D size structure
- Rect - 2D rectangle object
- Vec – Vector structure

Mat

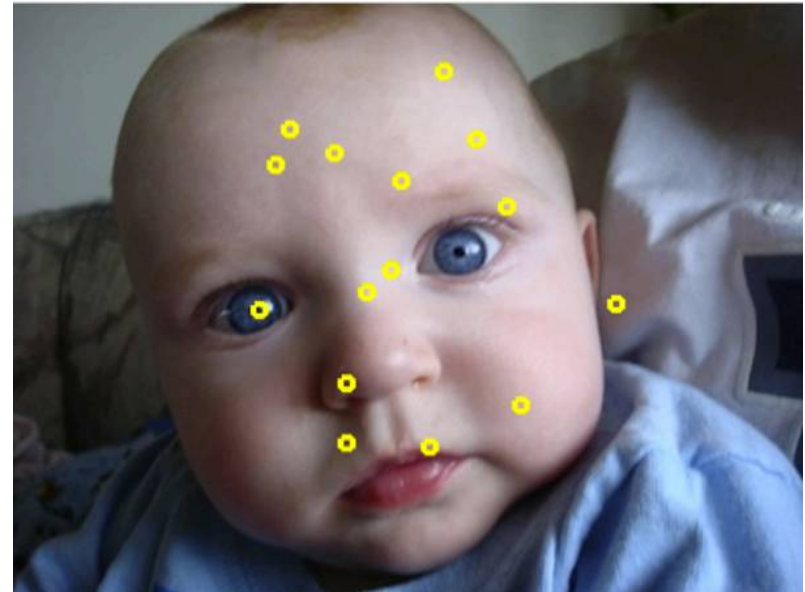
- The primary data structure in OpenCV is the Mat object. It stores images and their components.
- Main items
 - rows, cols - length and width(int)
 - channels - 1: grayscale, 3: BGR
 - depth: CV_<depth>C<num chan>

Mat

- Functions
 - `Mat.at<datatype>(row, col)[channel]` - returns pointer to image location
 - `Mat.channels()` - returns the number of channels
 - `Mat.clone()` - returns a deep copy of the image
 - `Mat.create(rows, cols, TYPE)` - re-allocates new memory to matrix
 - `Mat.cross(<Mat>)` - computes cross product of two matrices
 - `Mat.depth()` - returns data type of matrix
 - `Mat.dot(<Mat>)` - computes the dot product of two matrices

Point

- 2D Point Object
 - `int x, y;`
- Functions
 - `Point.dot(<Point>)` - computes dot product
 - `Point.inside(<Rect>)` - returns true if point is inside



Point

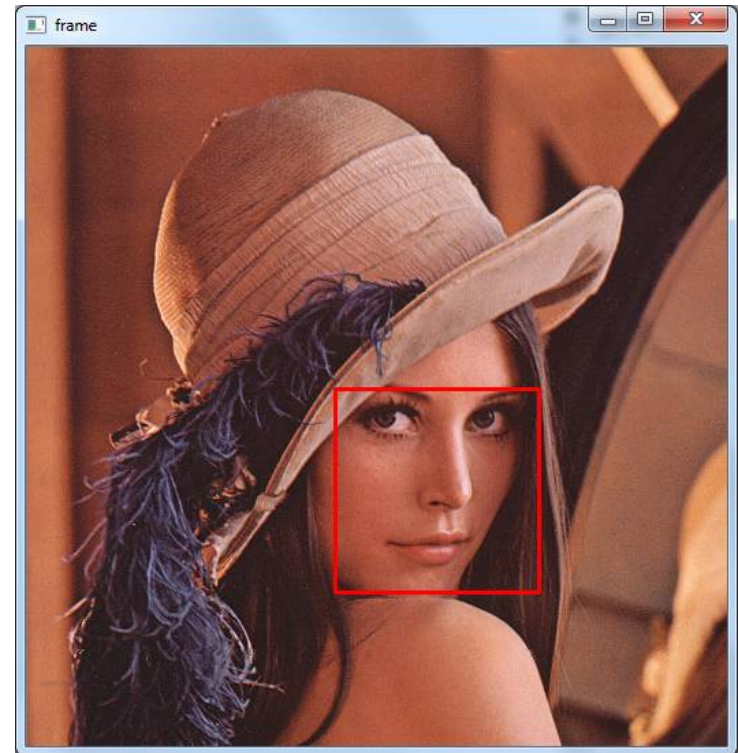
- Math operators, you may use
 - Point operator +
 - Point operator +=
 - Point operator -
 - Point operator -=
 - Point operator *
 - Point operator *=
 - bool operator ==
 - bool operator != double
norm

Size

- 2D Size Structure
 - int width, height;
- Functions
 - Point.area() - returns (width * height)

Rect

- 2D Rectangle Structure
 - int x, y, width, height;
- Functions
 - Point.tl() - return top left point
 - Point.br() - return bottom right point



Vec

- Vector Structure
 - Array 1D of elements
- Math operators, you may use
 - Vec operator []
 - Vec operator +
 - Vec operator +=
 - Vec operator -
 - Vec operator -=
 - Vec operator *
 - bool operator !=
 - bool operator ==

Introduction to OpenCV

IMAGES



Images

- Images in OpenCV are stored in a Mat object
- Consists of a matrix header and a pointer to the matrix containing the pixel values
- Header contains information such as the size of the matrix, the number of color channels in the image, etc.
- Access this information through functions:
 - `Mat.rows()`: Returns the number of rows
 - `Mat.columns()`: Returns the number of columns
 - `Mat.channels()`: Returns the number of channels

Reading, Writing, and Displaying Images

- Read the color image named "foo.png"
 - `Mat foo=imread("foo.png",IMREAD_COLOR);`
- Write the data contained in foo to "bar.png"
 - `imwrite("bar.png",foo);`
- Create a window
 - `namedWindow("display",WINDOW_AUTOSIZE);`
- Display the data stored in foo
 - `imshow("display",foo);`

Image Types

- The type of a Mat object specifies how to interpret the underlying binary data
- Represented as CV <Datatype>C<#Channels>
- Example: CV 8UC3 represents an image with three channels each represented by an 8-bit unsigned integer
 - Mat.type(): Returns an identifier that specifies the underlying arrangement of the data
 - Mat.depth(): Returns the number of bytes for a matrix element which depends on the datatype

Pixel Types

- Also need to know the underlying ordering of the channels
- OpenCV defaults to BGR
- Can also handle other types of channels/orderings such as RGB, HSV or GRAY
- Can access the value for a particular channel at a specific pixel once we have know how to interpret the underlying data
 - `Mat.at<datatype>(row,col)[channel]`: Returns a pointer to the image data at the specified location
- Convert between color spaces using
 - `cvtColor(foo,bar,CV_BGR2GRAY)`

Pixels in Image

- Gray image

	Column 0	Column 1	Column ...	Column m
Row 0	0,0	0,1	...	0, m
Row 1	1,0	1,1	...	1, m
Row,0	...,1, m
Row n	n,0	n,1	n,...	n, m

- Color image

	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	1, m	1, m	1, m
Row,0	...,0	...,0	...,1	...,1	...,1, m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m

Pixel Processing

- Gray image

```
for (int i = 0; i < img.rows; i++)  
{  
    for (int j = 0; j < img.cols; j++)  
    {  
        gray = img.at<uchar>(i, j);  
    }  
}
```

Pixel Processing

- Color image

```
for (int i = 0; i < img.rows; i++)  
{  
    for (int j = 0; j < img.cols; j++)  
    {  
        redChannel = img.at<Vec3b>(i, j)[2];  
        greenChannel = img.at<Vec3b>(i, j)[1];  
        blueChannel = img.at<Vec3b>(i, j)[0];  
    }  
}
```


Drawing Stuff

- Sometimes it is necessary to draw stuff onto the image. Instead of using complicated functions, why not just call a simple function?
- Here are some simple examples...
 - `void circle(image, Point(x,y),int rad, CV_BGR(b,g,r), int thickness=1)`
 - `void ellipse(image, RotatedRect box, CV_BGR(b,g,r), int thickness=1)`
 - `void line(image, Point(x,y), Point(x,y), CV_BGR(b,g,r), int thickness= 1)`
 - `void rectangle(img, Point(x,y), Point(x,y), CV_BGR(b,g,r), int thickness)`

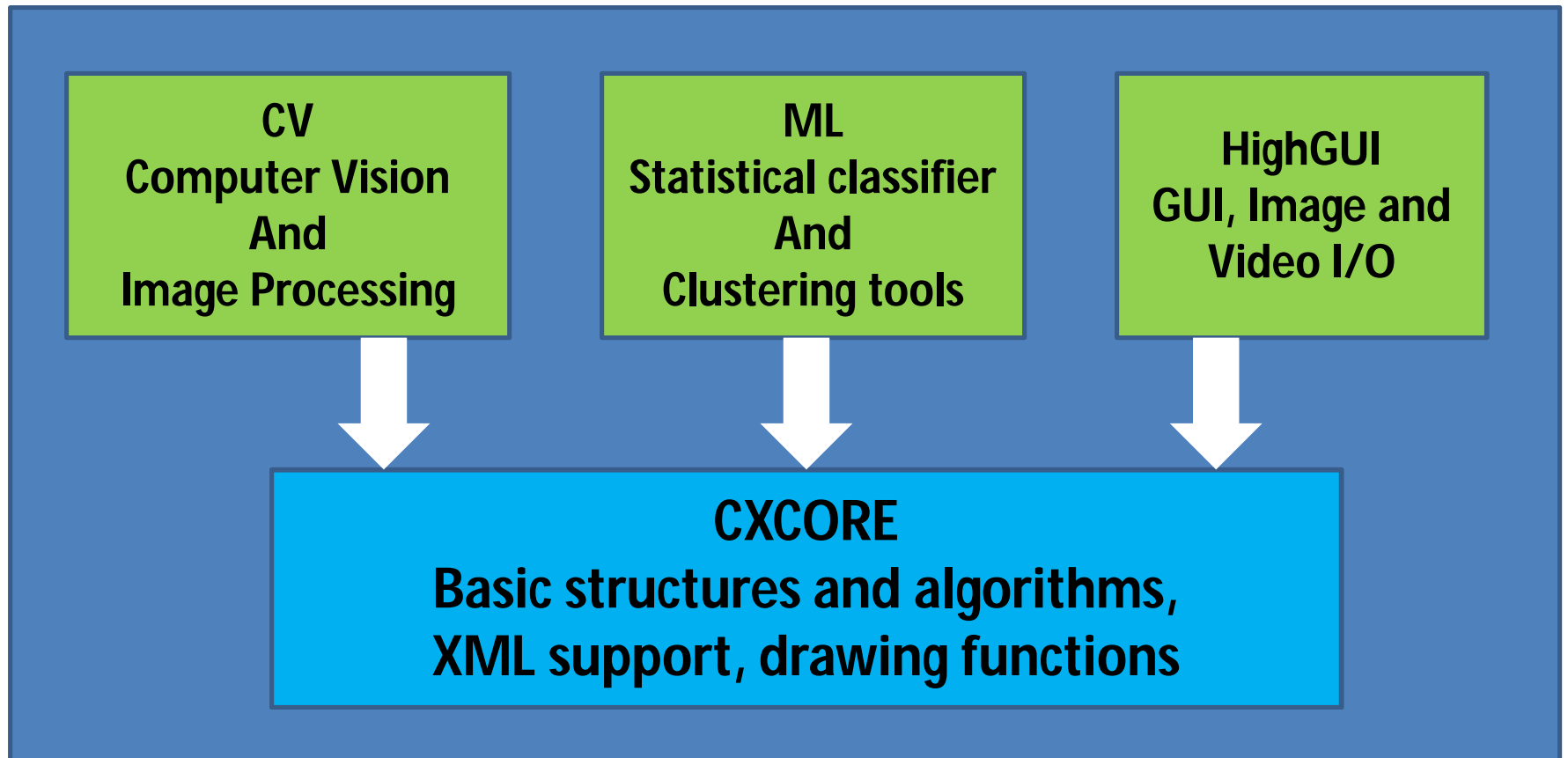
Copying Images

- In order to avoid making unnecessary copies of images, OpenCV uses a reference counting system to manage the underlying matrix data and *performs shallow copies by default*
- The copy constructor and assignment operator only copy the headers and the pointer to the underlying data
- Use `Mat::clone()` and `Mat::copyTo()` to perform a deep copy

Introduction to OpenCV

OPENCV MODULES

OpenCV Architecture



OpenCV Modules

Module	Description
core	basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other module
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.
highgui	an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.
video	a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms

OpenCV Modules

Module	Description
calib3d	basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction
feature2d	salient feature detectors, descriptors, and descriptor matchers
objdetect	detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
gpu	GPU-accelerated algorithms from different OpenCV modules
ml, flann, etc	Machine learning, OpenCL, FLANN, etc.

Open Source

Computer Vision Library

intel



References

- opencv.org
- ...