

*

Author: Dmitry Kurtaev <dmitry.kurtaev@intel.com>

Date: Tue Jul 6 10:30:00 2021 +0300

Introduction to OpenCV



>>> What is OpenCV?



- * Open Source project on GitHub with >5M downloads per year.

<https://github.com/opencv/opencv>

PyPI downloads 113M

- * The most popular computer vision library with 20 years of history
- * Has modular structure: core, imgproc, calib3d, dnn, stitching, ...
- * Written in C++ but has automatic wrappers in Python, Java, JavaScript, Matlab, GO, PHP
- * Cross-platform and well optimized for R&D

>>> OpenCV basic structures

* `cv::Mat` – images, masks, vector fields, complex values, any data

```
cv::Mat mat(480, 640, CV_8UC3);  
int rows      = mat.rows;      // 480  
int cols      = mat.cols;      // 640  
int channels   = mat.channels(); // 3  
uint8_t* data = mat.data;
```

* `cv::Mat` types: depth (`CV_8U`, `CV_16F`, `CV_32F`, `CV_64F`) + number of channels:

`CV_8U + C + 3 = CV_8UC3`

* `std::cout << mat << std::endl;` - To print `cv::Mat` in console

>>> OpenCV basic structures

+-----+-----+-----+			
* cv::Rect rect;	* cv::Point point;	* cv::Size size;	
int x = rect.x;	int x = point.x;	int w = size.width;	
int y = rect.y;	int y = point.y;	int h = size.height;	
int w = rect.width;			
int h = rect.height;			
+-----+-----+-----+			
* Region of interest (ROI)		[1, 1, 1, [1, 1, 1,	
cv::Mat mat = cv::Mat::ones(5, 3, CV_8UC1);		1, 1, 1, 5, 5, 1,	
std::cout << mat << std::endl;		1, 1, 1, 5, 5, 1,	
		1, 1, 1, 5, 5, 1,	
cv::Mat roi = mat(cv::Rect(/*xywh*/ 0, 1, 2, 3));		1, 1, 1] 1, 1, 1]	
roi.setTo(5);			
std::cout << mat << std::endl;			
+-----+-----+-----+			

>>> IO

---- Read image from file -----

```
cv::Mat image = cv::imread("C:\\Users\\dkurtaev\\Pictures\\example.jpg");
```

---- Read frames from a camera ---+---- Visualize an image -----

```
cv::Mat frame; | cv::namedWindow("My image", cv::WINDOW_NORMAL);  
cv::VideoCapture cap(0); | cv::imshow("My image", image);  
cap >> frame; | cv::waitKey();  
|
```

---- Write image to the file -----+

```
cv::Mat mat(480, 640, CV_8UC3); |  
cv::randu(mat, 0, 255); |  
cv::imwrite("output.png", mat); |
```



>>> Drawings

```
cv::rectangle(image,
               cv::Point(37, 357),           // left-top corner
               cv::Point(196, 408),          // right-bottom corner
               cv::Scalar(0, 255, 0));        // color (BGR)

cv::circle(image,
            cv::Point(37, 357),              // center
            15,                             // radius
            cv::Scalar(255, 0, 0),           // color (BGR)
            cv::FILLED);                     // thickness

cv::putText(image, "Matreshka",
            cv::Point(0, 20),                // position
            cv::FONT_HERSHEY_SIMPLEX,        // font
            0.5,                             // scale
            cv::Scalar(0, 255, 255));         // color (BGR)
```



>>> Methods

* Image2Image

```
cv::Mat src, dst, mask;
```

```
cv::resize(src, dst, cv::Size(1280, 960));
```

```
cv::Canny(src, dst, /*threshold1*/ 100, /*threshold2*/ 200);
```

```
cv::inpaint(src, mask, dst, /*inpaintRadius*/ 3, cv::INPAINT_TELEA);
```

```
std::vector<cv::Mat> images;
```

```
cv::Ptr<Stitcher> stitcher = cv::Stitcher::create(cv::Stitcher::PANORAMA);
```

```
stitcher->stitch(images, dst);
```

>>> Methods

* Image2Data

```
cv::Mat image;
```

```
std::vector<std::vector<cv::Point> > contours;
```

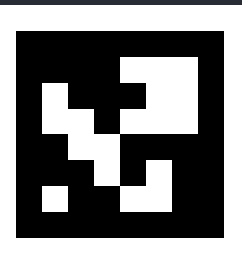
```
cv::findContours(image, contours, cv::RETR_EXTERNAL, cv::CHAIN_APPROX_SIMPLE);
```

```
std::vector<std::vector<cv::Point2f > > corners;
```

```
std::vector<int> ids;
```

```
auto dictionary = cv::aruco::getPredefinedDictionary(cv::aruco::DICT_6X6_250);
```

```
cv::aruco::detectMarkers(image, dictionary, corners, ids);
```



>>> Image processing



```
cv::Mat src = cv::imread("example.jpg"); // src.data = BGRBGR...BGR
```

```
cv::Mat dst(src.size(), CV_8UC1);
```

```
for (int y = 0; y < src.rows; ++y) {  
    for (int x = 0; x < src.cols; ++x) {  
        cv::Vec3b bgr = src.at<cv::Vec3b>(y, x);  
        uint8_t gray = (29 * bgr[0] + 150 * bgr[1] + 77 * bgr[2]) >> 8;  
        dst.at<uint8_t>(y, x) = gray;  
    }  
}
```

```
cv::imshow("grayscale", dst);  
cv::waitKey();
```

>>> Modules

* opencv

core, imgcodecs, imgproc, videoio, highgui
gapi
photo, stitching
features2d, calib3d
flann, ml, dnn
objdetect
video

* opencv_contrib

aruco, bgsegm, img_hash, optflow, quality, rgbd, and many more

>>> Hello World! The simplest eyes detector

```
#include <opencv2/opencv.hpp>

int main(int argc, char** argv) {
    cv::VideoCapture cap(0);
    cv::Mat frame, gray;
    while (cv::waitKey(1) < 0) {
        cap >> frame;
        if (frame.empty())
            break;

        cv::cvtColor(frame, gray, cv::COLOR_BGR2GRAY);

        double minVal;
        cv::Point minLoc;

        cv::minMaxLoc(gray, &minVal, 0, &minLoc);
        cv::circle(frame, minLoc, 10, cv::Scalar(0, 255, 0), cv::FILLED);

        // Exclude found point area from the search.
        cv::circle(gray, minLoc, 50, 255, cv::FILLED);

        cv::minMaxLoc(gray, &minVal, 0, &minLoc);
        cv::circle(frame, minLoc, 10, cv::Scalar(0, 255, 0), cv::FILLED);

        cv::imshow("frame", frame);
    }
    return 0;
}
```

