

*

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

Date: Tue Jul 7 10:30:00 2020 +0300

Introduction to OpenCV



>>> What is OpenCV?

- * Open Source project on GitHub with >3M downloads per year.
<https://github.com/opencv/opencv>
- * 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, ObjC/Swift, 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;
```

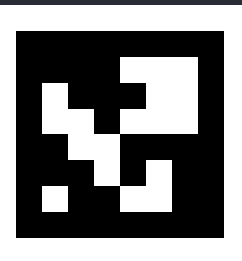
```
cv::findContours(image, contours, cv::RETR_EXTERNAL, cv::CHAIN_APPROX_SIMPLE);
```

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

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

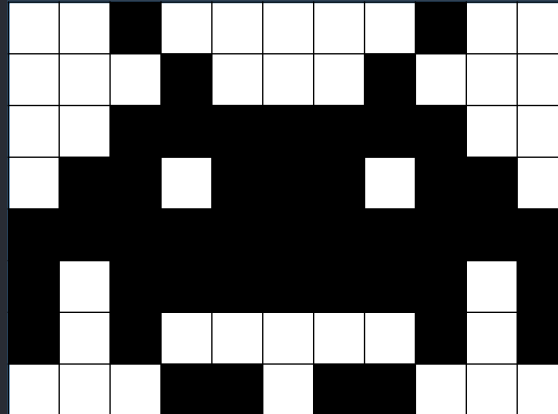
```
auto dictionary = cv::aruco::getPredefinedDictionary(cv::aruco::DICT_6X6_250);
```

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



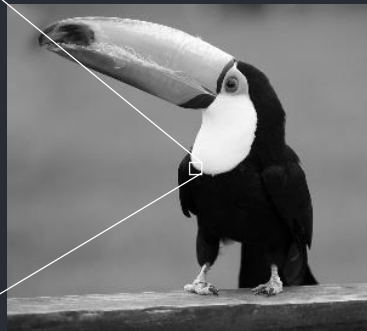
>>> Image representation

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |



binary

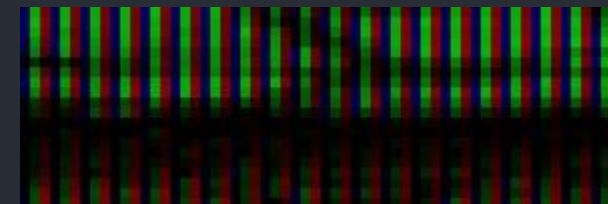
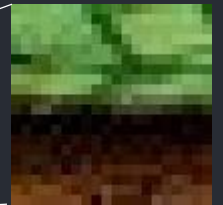
| | | | | | | | | | |
|----|----|-----|-----|-----|-----|-----|-----|-----|-----|
| 25 | 32 | 140 | 223 | 225 | 228 | 229 | 228 | 228 | 228 |
| 16 | 16 | 133 | 207 | 220 | 224 | 225 | 224 | 224 | 224 |
| 16 | 16 | 56 | 158 | 212 | 218 | 220 | 222 | 222 | 222 |
| 14 | 14 | 12 | 81 | 188 | 208 | 213 | 216 | 217 | 217 |
| 13 | 12 | 12 | 19 | 95 | 175 | 190 | 199 | 208 | 212 |
| 13 | 12 | 10 | 11 | 27 | 79 | 167 | 183 | 193 | 198 |
| 12 | 13 | 10 | 12 | 12 | 16 | 34 | 119 | 162 | 187 |
| 11 | 12 | 11 | 13 | 12 | 17 | 14 | 21 | 41 | 60 |
| 9 | 9 | 12 | 11 | 13 | 10 | 14 | 15 | 20 | 21 |
| 9 | 9 | 13 | 12 | 10 | 13 | 13 | 14 | 10 | 11 |



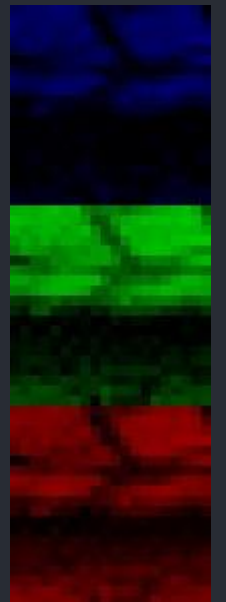
grayscale



color



BGR interleaved



BGR planar

>>> 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 = (0.114f * bgr[0] + 0.587f * bgr[1] + 0.299f * bgr[2]);
        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

>>> Install

- * Linux: build from source or download with Intel OpenVINO
- * Windows: <https://github.com/opencv/opencv/releases> or Intel OpenVINO
- * Python (any OS): `pip install opencv-python`
- * JavaScript: <https://docs.opencv.org/master/opencv.js>

>>> 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;
}
```



>>> Useful links

- * OpenCV online documentation: <https://docs.opencv.org/master/>
- * Community forum: <https://answers.opencv.org/> (or just <https://stackoverflow.com/questions/tagged/opencv>)