



Обзор OpenCV

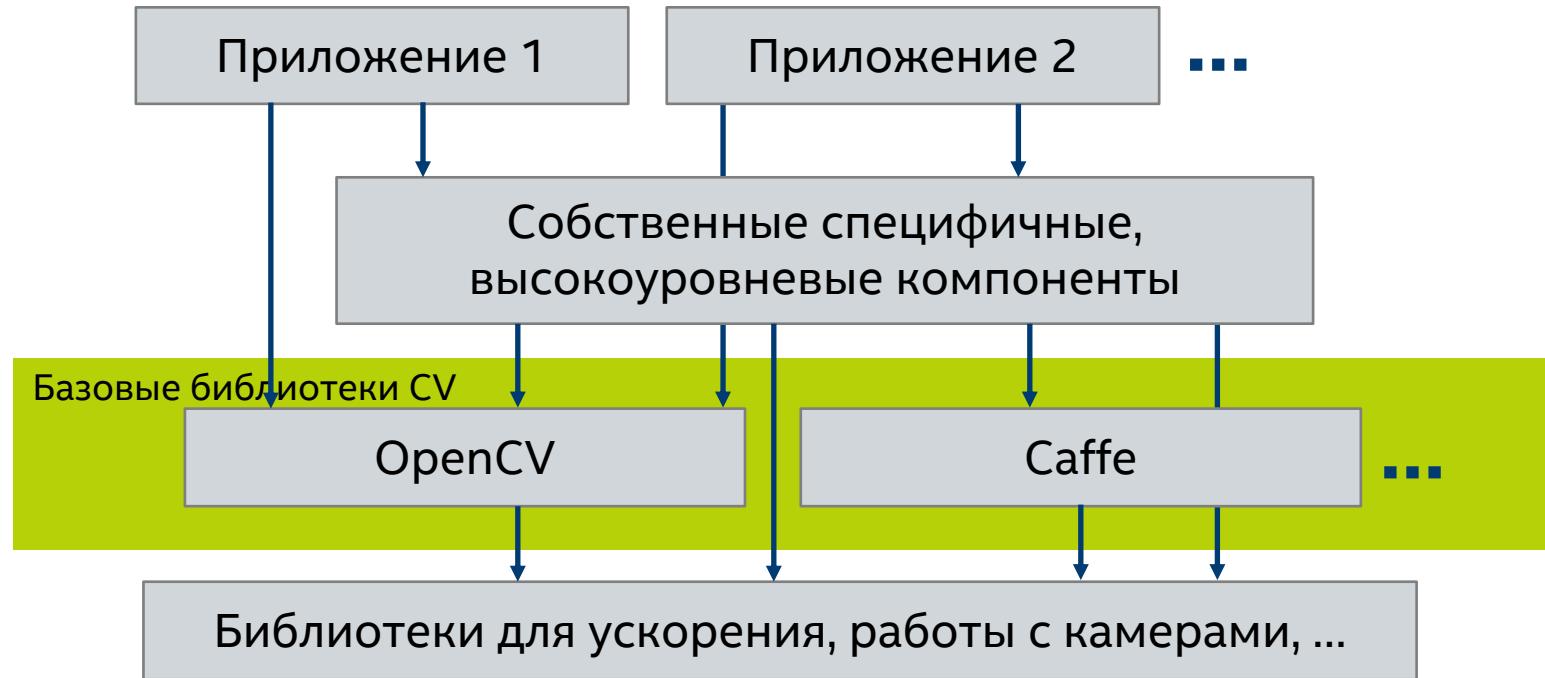
Вадим Писаревский, руководитель проекта OpenCV

Краткая справка

- Страница: <http://opencv.org>
- Самая популярная библиотека компьютерного зрения. Используется многими компаниями и учебными учреждениями.
- Написана на C/C++, исходный код открыт, включает более 2000 функций/алгоритмов.
- Лицензия BSD (разрешается бесплатное использование дома, для учебы, на работе)
- Разрабатывается с 1998г, сначала в Intel, потом в Itseez и снова в Intel, при активном участии сообщества.
- >18,000,000 загрузок (без учета трафика с github)



OpenCV как базовая библиотека CV



Карта проекта

Языки:

C/C++
Python
Java
Javscript

(Matlab, Ruby,
Haskell, C#, Lua,
Closure)

Ускорение:

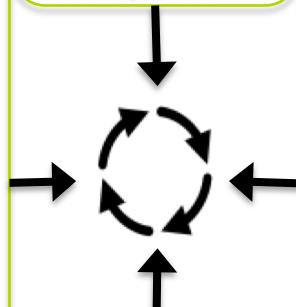
OpenCL
CUDA
parallel for
(OpenMP, TBB...)
SIMD (SSE, AVX,
NEON), HAL (IPP,
Carotene), Halide

Использование сторонних библиотек:

zlib, png, tiff
jpeg, jasper,
webp, gdcm,
Gtk, Qt,
Cocoa, Win32
OpenNI, V4L ...
(20+ backends),
ffmpeg,
MediaSDK,
gstreamer,
AVFoundation,
intel ipp, tbb
eigen
Tesseract (OCR)
VTK
libmv
protobuf

Разработка:

Основная
команда,
Сообщество



Инфраструктура:

CMake,
Doxygen,
Wrapper
generators,
Github, Buildbot,
GTest

Модули:

"main" opencv:
core, imgproc,
photo, video,
calib3d,
features2d,
ml, dnn,
objdetect, shape
highgui, viz
stitching,
superres
videostab ...

opencv_contrib:
rgbd, tracking
text, reg
face, bioinspired
ximgproc, xphoto
xfeatures2d

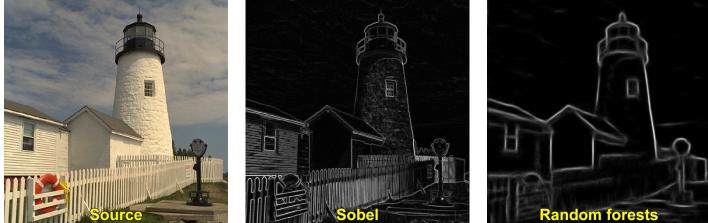
Архитектуры:

x86/x64
ARM,
PowerPC,
(MIPS, Sparc, ...)

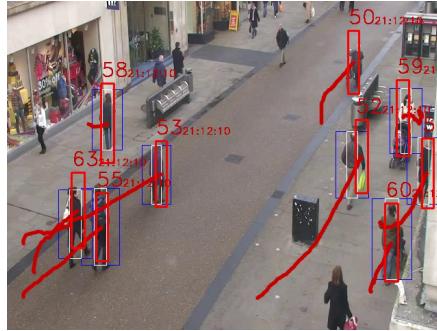
ОС:

Windows
Linux
macOS
Android
iOS
(QNX, BSD, ...)

Примеры функциональности (1)



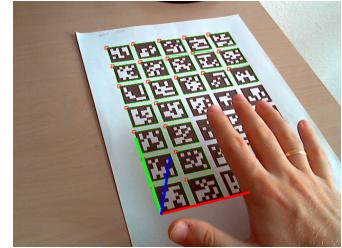
Базовая и продвинутая обработка изображений
(`imgproc`, `contrib/ximgproc`)



Трекинг объектов (`contrib/tracking`)



Детектирование и распознавание текста
(`contrib/text`)



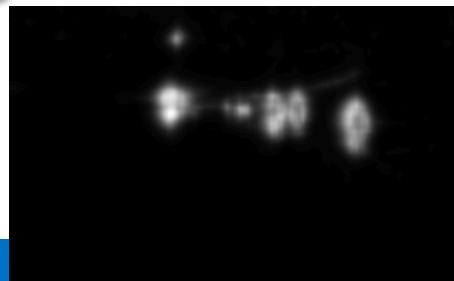
Калибрация камер;
вычисление положения и
ориентации камеры
(`calib3d`, `contrib/aruco`)



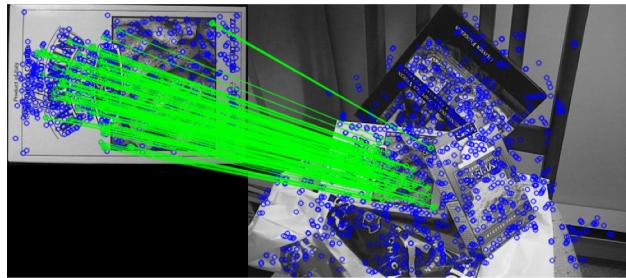
HDR и вычитание фона с помощью
эмulationи сетчатки “bioinspired” (`contrib/bioinspired`)

Parvo

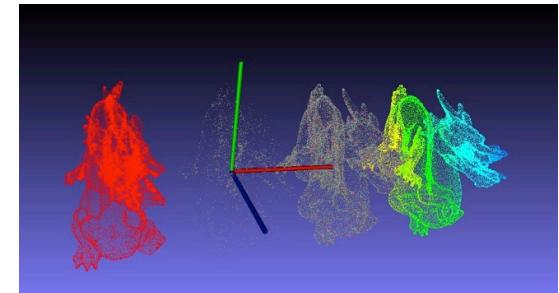
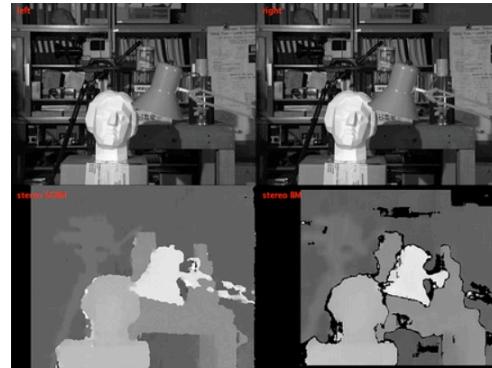
Magno



Примеры функциональности (2)



Нахождение особых точек и
их сопоставление (**features2d**)

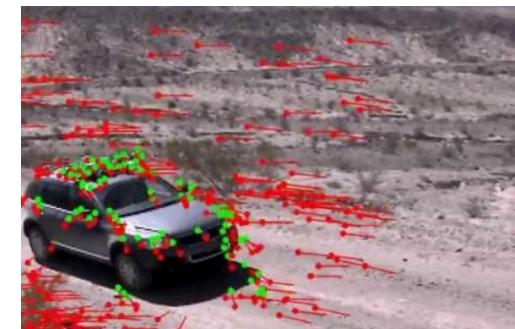


Визуализация 3D данных (**viz**)

Определение карты глубины по стерео (**calib3d**)



HDR, шумоподавление и другие алгоритмы
вычислительной фотографии (**photo, contrib/xphoto**)



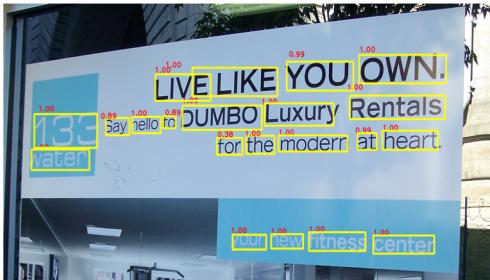
Разреженный и плотный оптический поток (**video**)

Примеры функциональности (3): deep learning

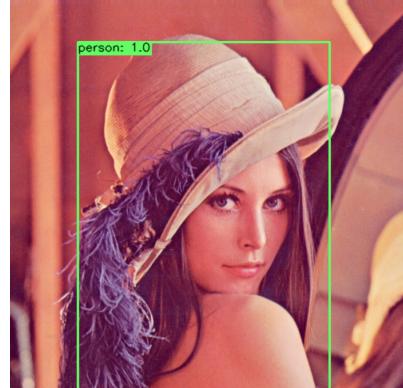


space shuttle: 0.9998
airliner: 0.0001
...

Классификация объектов
(50-100FPS)



Нахождение текста
(10FPS)



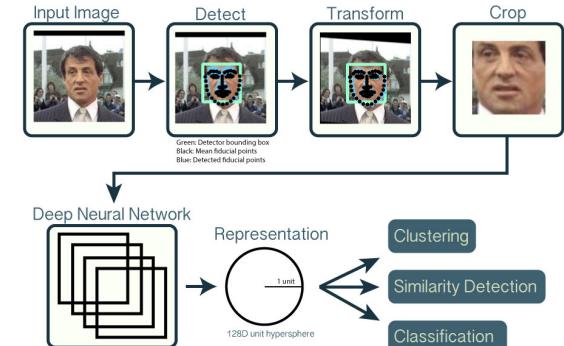
Детектирование объектов
(20-50FPS)



Интересные фотоэффекты



Определение позы человека с помощью OpenPose
(2-3FPS)



Распознавание лиц используя OpenFace

Разработка OpenCV

Open Source Computer Vision Library <http://opencv.org>



opencv c-plus-plus computer-vision deep-learning image-processing Manage topics

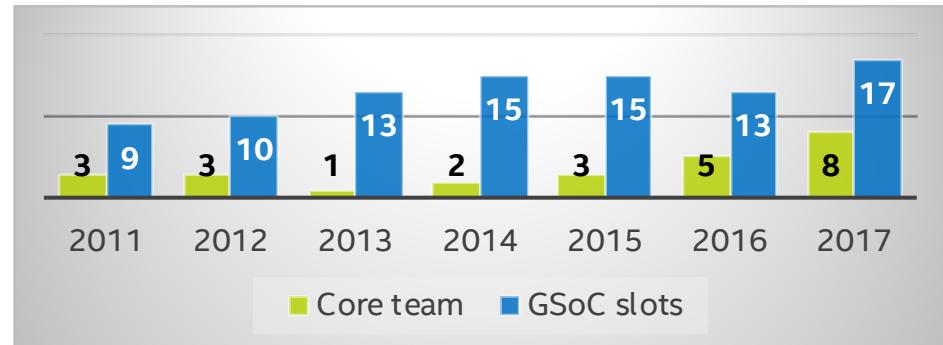
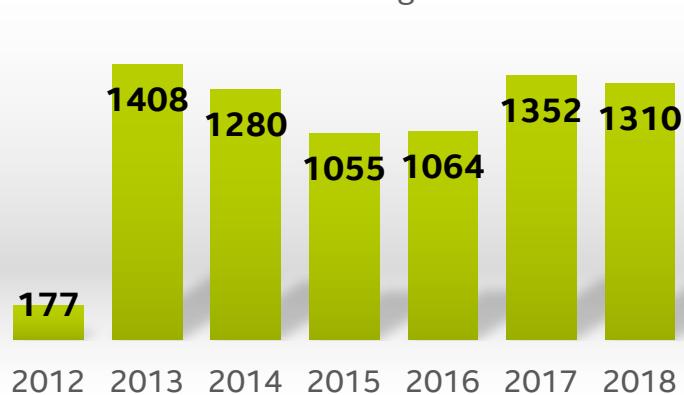
21,181 commits

2 branches

51 releases

722 contributors

PRs merged



>5 patches per working day!

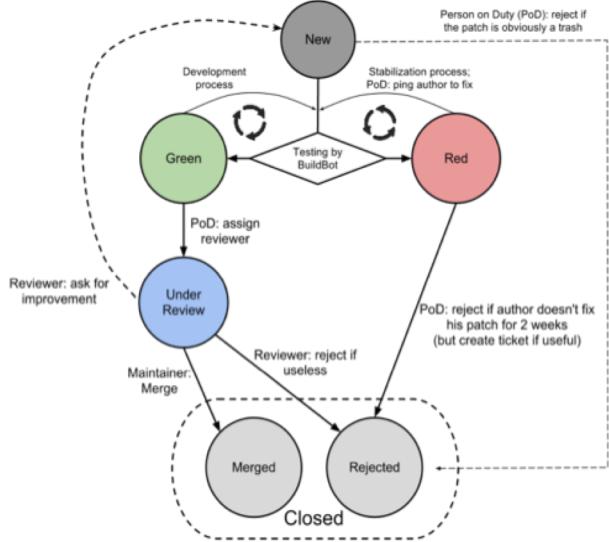
>70% of patches from community

27/32 opencv_contrib modules is 100% contributed code

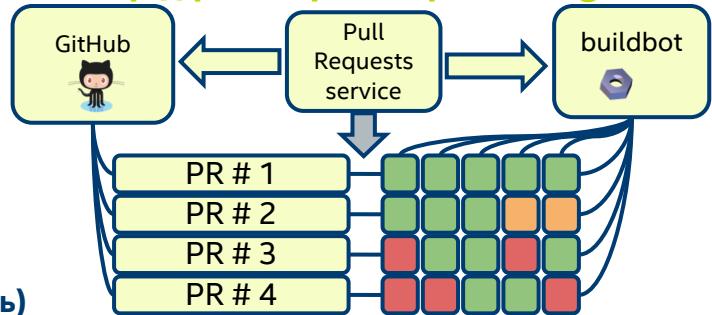


Непрерывная интеграция патчей

<http://pullrequest.opencv.org>



Матрица тестирования (часть)



	Win	Linux (U LTS)	MacOS	iOS	Android
x86/x64	+	+	+	+ (build only)	+ (build only)
arm32/aarch64	-	+	-	+ (build only)	+ (build only)
OpenCL	+	+	+ (disabled)	-	-
CUDA	-	+ (build only)	-	-	-
Python bindings	+	+	+	-	-
Java bindings	+	+	+	-	+ (build only)
ABI check	-	+	-	-	-
valgrind	-	+ (nightly)	-	-	-



С чего начать?

<http://opencv.org>:

The screenshot shows the official OpenCV website at <http://opencv.org>. At the top, there's a navigation bar with links to About, News, Releases, Platforms, Books, Links, and License. Below the navigation is a main content area with a paragraph about the BSD license and usage across various platforms. To the right of the content is a sidebar titled "Quick Links" containing links to Online documentation, Tutorials, User Q&A forum, Report a bug, Build farm, Developer site (which is highlighted with a red arrow), and Wiki. A "Donate" button is also in the sidebar. At the bottom, there's a "Latest news" section with several articles and social media sharing icons.

Документация,
уроки и т.д.

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



Быстрый старт

1. Устанавливаем компилятор C/C++, cmake (<http://cmake.org>)
2. Клонируем репозиторий с github (<http://github.com/opencv/opencv>), конфигурируем с помощью cmake и строим

```
$ mkdir ocv.build && cd ocv.build && cmake -G "Unix Makefiles" ..../opencv  
$ make -j8
```

3. Создаем каталог с проектом и кладем туда следующий CMakeList.txt:

```
cmake_minimum_required(VERSION 3.1)  
#set(CMAKE_CXX_STANDARD 11) # for OpenCV 4.0 and later  
project(myopencv_samples)  
find_package(OpenCV REQUIRED)  
include_directories(${OpenCV_INCLUDE_DIRS})  
add_executable(myopencv_app1 app1_main.cpp) # add other .cpp and .h files here  
target_link_libraries(myopencv_app1 ${OpenCV_LIBS})
```

4. Создаем main.cpp (см. дальше). Можно взять один из готовых примеров из opencv/samples/cpp.
5. Указываем cmake, где найти OpenCVConfig.cmake и генерируем проект или Makefile's.
6. Строим проект из среды или командной строки



Первая программа: находим ребра

```
01 #include "opencv2/opencv.hpp"
02 using namespace cv;
03 int main(int argc, char** argv)
04 {
05     Mat img, gray, edges;
06     img = imread(argv[1], 1);
07     imshow("original", img);
08     cvtColor(img, gray, COLOR_BGR2GRAY);
09     GaussianBlur(gray, gray, Size(7, 7), 1.5);
10     Canny(gray, edges, 0, 50);
11     imshow("edges", edges);
12     imwrite("result.png", edges);
13     waitKey();
14     return 0;
15 }
```



Настраиваем параметры

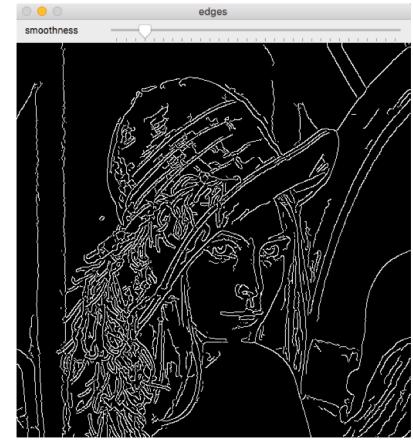
```
#include "opencv2/opencv.hpp"

using namespace cv;

Mat img, gray, edges;
int smoothness = 15;

static void on_smoothness(int, void*) {
    cvtColor(img, gray, COLOR_BGR2GRAY);
    double sigma = smoothness*0.1 + 1;
    int ksize = cvRound(sigma*4+1)|1;
    GaussianBlur(gray, gray, Size(ksize, ksize), sigma);
    Canny(gray, edges, 0, 50);
    imshow("edges", edges);
}

int main(int argc, char** argv) {
    img = imread(argc > 1 ? argv[1] : "lena.jpg", 1);
    imshow("original", img); imshow("edges", img);
    createTrackbar("smoothness", "edges", &smoothness, 30, on_smoothness, 0);
    on_smoothness(0, 0);
    waitKey(); return 0;
}
```



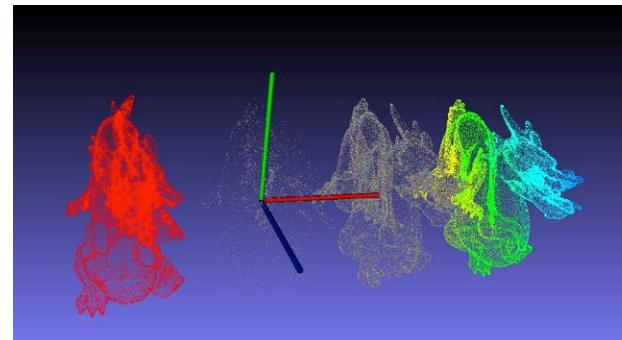
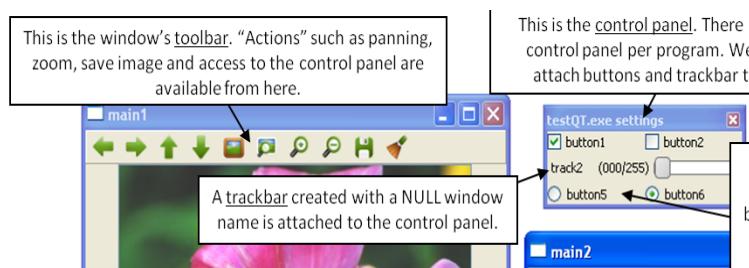
Добавляем видео

```
#include "opencv2/opencv.hpp"
using namespace cv;
int main(int argc, char** argv) {
    Mat img, gray, edges;
    int smoothness = 15;
    bool firstframe = true;
    VideoCapture cap(0); // для чтения из видеофайла заменить на VideoCapture cap(argv[1])
    for(;;) {
        cap >> img; if(img.empty()) break;
        cvtColor(img, gray, COLOR_BGR2GRAY);
        double sigma = smoothness*0.1 + 1;
        int ksize = cvRound(sigma*4+1)|1;
        GaussianBlur(gray, gray, Size(ksize, ksize), sigma);
        Canny(gray, edges, 0, 50);
        imshow("edges", edges);
        if(firstframe) {
            createTrackbar("smoothness", "edges", &smoothness, 30, 0, 0);
            firstframe = false;
        }
        if(waitKey(30) >= 0) break;
    }
    return 0;
}
```



HighGUI (=ui+imgcodec+videoio)

- Окна с “памятью”
- Обработка нажатий клавиш.
- Обработка событий от мыши (см. <https://github.com/opencv/opencv/blob/master/samples/cpp/camshiftdemo.cpp>)
- Слайдеры.
- Чтение/запись изображений
- Чтение/запись видео
- В случае наличия Qt – много дополнительных средств (тулбар, кнопки, зум, значения пикселей ...)
- См. также модуль VIZ для визуализации 3D данных: <http://habrahabr.ru/company/itseez/blog/217021/>



cv::Mat – многомерный многоканальный массив

`cv::Mat A(h, w, CV_8UC3);`

- Размеры, step
- Счетчик ссылок (=1)
- Указатель на данные

`cv::Mat B = A;`

- Размеры, step
- Счетчик ссылок (=2)
- Указатель на данные

`cv::Mat C=A(roi);`

- Размеры ROI, step
- Счетчик ссылок (=3)
- Указатель на данные

Элементы/Пиксели

RGBRGBRGBRGBRGBRGBRGBRGBRG
RGBRGBRGBRGBRGBRGBRGBRGBRG

Расположение в памяти матрицы С

RGBRGBRGBRGB*****RGBRGBRGBRGB*****...
...
...

cv::Mat – универсальный тип данных

BGR/RGB: CV_8UC3



grayscale/bayer: CV_8UC1 mask: CV_8UC1



HSV/Lab/...: CV_8UC3



matrix/tensor: CV_32F, CV_64F

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots \\ a_{3,1} & a_{3,2} & a_{3,3} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

dense motion field: CV_32FC2



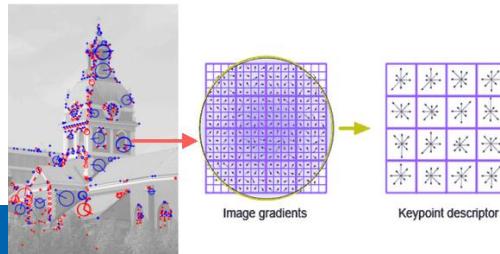
hi-quality photo: CV_16UC3, CV_32FC3



depth/RGBD: CV_32F, CV_32FC4



feature descriptors: CV_8U/CV_32F



???

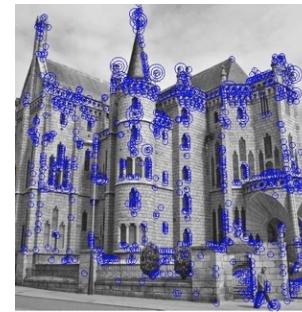
CV_MAKETYPE(
CV_8U ... CV_64F,
1 .. 512)

std::vector – еще один универсальный тип

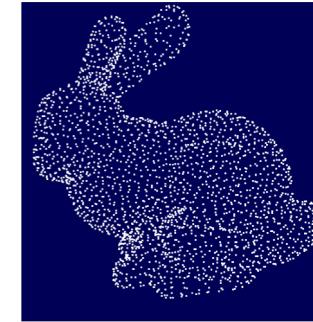
sparse motion field:
vector<Point2f>



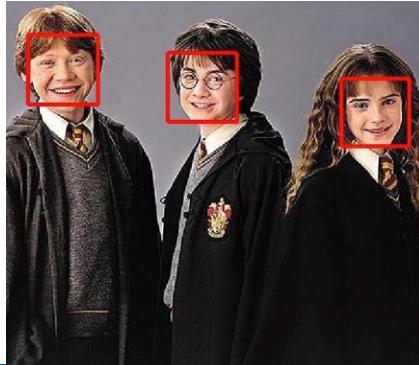
keypoints: vector<KeyPoint>



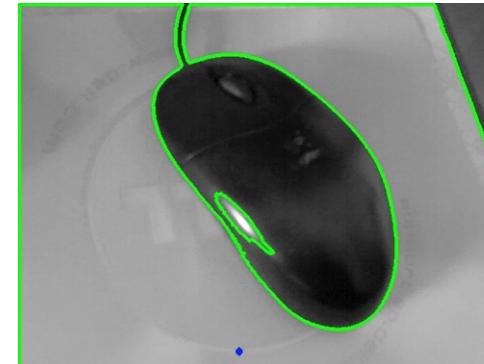
pointcloud: vector<Point3f>



object detection: vector<Rect>,
vector<Detection>



contours: vector<vector<Point>>



Представление некоторых std::vector<> как cv::Mat

Массив из N точек

- cv::Mat a(v);
- Размеры (Nx1), step(=12)
- Счетчик ссылок (отсутствует)
- Указатель на данные



Nx1
3-канальное
изображение

```
namespace cv {  
CV_EXPORTS_W Scalar mean(InputArray src, InputArray mask = noArray())  
}  
...  
cv::Mat img(...); Scalar mean_brightness = mean(img);  
std::vector<Point2f> ptset = ...; Scalar mass_center = mean(ptset);
```

Работаем с матрицами

```
Mat M = Mat::zeros(480,640,CV_8UC1); // Создаем 8-битное одноканальное изображение 640x480, заполненное нулями  
Rect roi(100, 200, 20, 20); // Определяем ROI  
Mat subM = M(roi); // "выделяем" ROI в отдельную матрицу без копирования  
subM.at<uchar>(y,x)=255; // изменяем пиксель в строке y и столбце x ROI и (x+100, y+200) в исходном изображении  
float a = CV_PI/3;  
Mat R = (Mat_<float>(2, 2) << cos(a), -sin(a), sin(a), cos(a)); // заполняем матрицу по элементам  
float iRdata[] = {cos(a), sin(a), -sin(a), cos(a)};  
Mat iR = Mat(2, 2, CV_32F, R2data); // альтернативный вариант (данные не копируются)  
CV_Assert(norm(R*iR, Mat::eye(3, 3, CV_32F), NORM_L2) <= 0.01); // используем матричные выражения  
Mat img = imread("lena.jpg", 1); // читаем картинку как цветную (BGR)  
Mat planes[3], noise(img.size(), CV_32F);  
cvtColor(img, img, COLOR_BGR2YUV); // конвертируем RGB => YUV  
split(img, planes); // разделяем на отдельные цветовые плоскости  
randn(noise, Scalar::all(0), Scalar::all(30)); // генерируем гауссовый шум  
GaussianBlur(noise, noise, Size(3, 3), 0.5); // сглаживаем его  
add(planes[0], noise, planes[0], noArray(), CV_8U); // добавляем шум к яркостной компоненте  
merge(planes, 3, img); // объединяем каналы обратно  
cvtColor(img, img, COLOR_YUV2BGR); // превращаем картинку снова в BGR
```



core: операции над матрицами, “мини Matlab”

Mat::zeros, Mat::eye, Mat::ones, Mat::setTo, randu, randn – заполнение/ инициализация матриц элементов, объединение и выделение отдельных каналов.

Mat::operator (), **Mat::row**, **Mat::col**, **Mat::rowRange**, **Mat::colRange**, **Mat::diag**, **Mat::reshape** – выделение частей матриц и изменение формы без копирования

Mat::copyTo, **Mat::clone**, **Mat::repeat**, **vconcat**, **hconcat**, **flip**, **transpose**, **randShuffle**, **split**, **merge**, **mixChannels** – копирование и различные перемешивания

Mat::convertTo, **normalize** – преобразование к другому диапазону и/или к другому типу данных

add, **subtract**, **multiply**, **divide**, **absdiff** – поэлементные арифметические операции

bitwise_and, **bitwise_or**, **bitwise_xor**, **bitwise_not** – логические операции

compare, **min**, **max** – поэлементное сравнение, минимум, максимум

sum, **mean**, **meanStdDev**, **norm**, **minMaxLoc** – сбор статистики по матрице, сравнение матриц

gemm, **Mat::dot**, **Mat::cross**, **solve**, **eigen**, **SVD**, **determinant**, **trace**, **solvePoly**, **solveLP**,
MinProblemSolver – линейная алгебра, нахождение корней полиномов, решение задач оптимизации

exp, **log**, **sqrt**, **pow**, **cartToPolar**, **polarToCart** – стандартные поэлементные математические операции

reduce, **sort**, **sortIdx** – операции над строками и столбцами

dft, **dct** – дискретные ортогональные преобразования

http://docs.opencv.org/2.4/opencv_cheatsheet.pdf – здесь перечислено гораздо больше функций



Собственные функции обработки изображений

```
// повышаем резкость изображения
void unsharpMask(const Mat& src, Mat& dst, int sigma, int thresh, float scale)
{
    // проверяем что тип тот что нужно
    CV_Assert( src.type() == CV_8UC3 );

    // создаем сглаженную версию изображения
    Mat smoothed;
    GaussianBlur(src, smoothed, Size(cvRound(sigma*3)|1, cvRound(sigma*3)|1), sigma, sigma);
    dst.create(src.size(), src.type());

    // перебираем все пиксели с помощью вложенных циклов
    for( int i = 0; i < src.rows; i++ )
    {
        // каждый канал обрабатывается независимо, поэтому просто умножаем ширину на 3
        for( int j = 0; j < src.cols*3; j++ )
        {
            uchar pix = src.at<uchar>(i, j); // доступ к элементу изображения
            uchar spix = smoothed.at<uchar>(i, j);
            int diff = pix - spix;
            // основная формула алгоритма "unsharp mask"
            float result = abs(diff) < thresh ? (float)pix : pix + scale*diff;
            // сохраняем результат, аккуратно переводя float => uchar
            dst.at<uchar>(i, j) = saturate_cast<uchar>(result);
        }
    }
}
```



FileStorage: записываем структурированные данные в YAML

```
{  
FileStorage fs("test.yml", FileStorage::WRITE);  
  
fs << "intval" << 5  
    << "realval" << 3.1  
    << "str" << "ABCDEFGH";  
fs << "mtx" << Mat::eye(3,3,CV_32F);  
fs << "mylist" << "[" << 1 << 2 << 3  
                << 4 << 5 << "]";  
fs << "date" << "{:" << "month" << 12  
                << "day" << 31  
                << "year" << 2015  
                << "}";  
const uchar arr[] = {0, 1, 1, 0, 1, 1, 0, 1};  
fs << "bitmask" << "[:";  
fs.writeRaw("u", arr, 8);  
fs << "]";  
}
```

```
%YAML 1.0  
---  
intval: 5  
realval: 3.100000000000001e+00  
str: ABCDEFGH  
mtx: !!opencv-matrix  
    rows: 3  
    cols: 3  
    dt: f  
    data: [ 1., 0., 0., 0., 1., 0., 0., 0., 1. ]  
mylist:  
    - 1  
    - 2  
    - 3  
    - 4  
    - 5  
date: { month:12, day:31, year:2015 }  
bitmask: [ 0, 1, 1, 0, 1, 1, 0, 1 ]
```

FileStorage: записываем структурированные данные в XML

```
{  
// записываем в XML, одновременно сжимая  
результат  
FileStorage fs("test.xml.gz", FileStorage::WRITE);  
  
fs << "intval" << 5  
    << "realval" << 3.1  
    << "str" << "ABCDEFGH";  
fs << "mtx" << Mat::eye(3,3,CV_32F);  
fs << "mylist" << "[" << 1 << 2 << 3  
                << 4 << 5 << "]";  
fs << "date" << "{:" << "month" << 12  
                << "day" << 31  
                << "year" << 2015  
                << "}";
const uchar arr[] = {0, 1, 1, 0, 1, 1, 0, 1};  
fs << "bitmask" << "[:";  
fs.writeRaw("u", arr, 8);  
fs << "]";  
}
```

```
<?xml version="1.0"?>  
<opencv_storage>  
<intval>5</intval>  
<realval>3.1000000000000001e+00</realval>  
<str>ABCDEFGH</str>  
<mtx type_id="opencv-matrix">  
    <rows>3</rows>  
    <cols>3</cols>  
    <dt>f</dt>  
    <data>  
        1. 0. 0. 0. 1. 0. 0. 0. 1.</data></mtx>  
<mylist>  
    1 2 3 4 5</mylist>  
<date><month>12</month>  
    <day>31</day>  
    <year>2015</year></date>  
<bitmask>  
    0 1 1 0 1 1 0 1</bitmask>  
</opencv_storage>
```



FileStorage: записываем данные и считываем их обратно

```
{  
FileStorage fs("test.xml.gz", FileStorage::WRITE);  
  
fs << "intval" << 5  
    << "realval" << 3.1  
    << "str" << "ABCDEFGH";  
fs << "mtx" << Mat::eye(3,3,CV_32F);  
fs << "mylist" << "[" << 1 << 2 << 3  
                << 4 << 5 << "]";  
fs << "date" << "{:" << "month" << 12  
                << "day" << 31  
                << "year" << 2015  
                << "}";  
const uchar arr[] = {0, 1, 1, 0, 1, 1, 0, 1};  
fs << "bitmask" << "[:";  
fs.writeRaw("u", arr, 8);  
fs << "]";  
}  
  
{  
FileStorage fs("test.xml.gz", FileStorage::READ);  
  
int intval = (int)fs["intval"];  
double realval; fs["realval"] >> realval;  
string str = (string)fs["str"];  
  
Mat mtx; fs["mtx"] >> mtx;  
  
// считывание вектора. Длинный вариант  
vector<int> mylist;  
FileNode mylist_node = fs["mylist"];  
size_t n = mylist_node.size();  
FileNodeIterator mylist_it = mylist_node.begin();  
for( i = 0; i < n; i++, ++it)  
    mylist.push_back((int)*it);  
  
FileNode dn = fs["date"];  
int month = (int)dn["month"],  
    day=(int)dn["day"],  
    year=(int)dn["year"];  
  
// считывание вектора. Короткий вариант  
vector<uchar> bitmask; fs["bitmask"] >> bitmask;  
}
```



Примеры решения некоторых основных задач к. зрения с использованием OpenCV

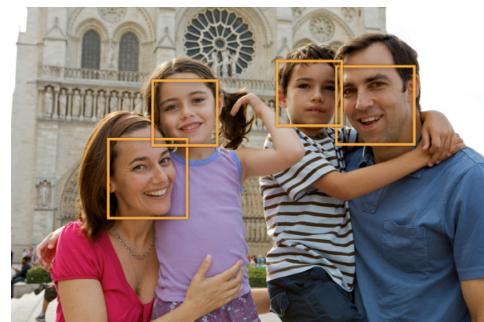
Сопоставить изображения



Отделить движущиеся объекты от неподвижного фона

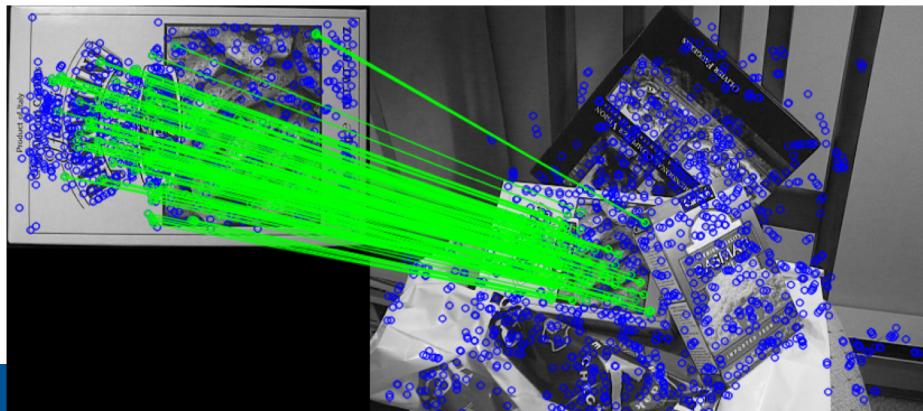


Найти объекты
заданного класса
(например, лица)



Сопоставление плоских объектов: features2d+ calib3d

```
Mat img[2], desc[2]; vector<KeyPoint> kpt[2];
Ptr<ORB> f2d = ORB::create();
for( int k = 0; k < 2; k++ ) {
    img[k] = imread(format("image%d.png", k+1), 0);
    f2d->detectAndCompute( img[k], Mat(), kpt[k], desc[k], false );
}
vector< DMatch > matches;
FlannBasedMatcher matcher(/* params ... */);
BFMatcher(NORM_HAMMING,true).knnMatch( desc[0], desc[1], matches, 1 );
vector<Point2f> points[2];
for( size_t i = 0; i < matches.size(); i++ ) {
    points[0].push_back(kpt[matches[i].queryIdx].pt);
    points[1].push_back(kpt[matches[i].trainIdx].pt);
}
Mat H, inliers;
H = findHomography( points[0], points[1], RANSAC, 3.0, inliers );
```



HDR + вычитание фона: opencv_contrib/bioinspired

Input video

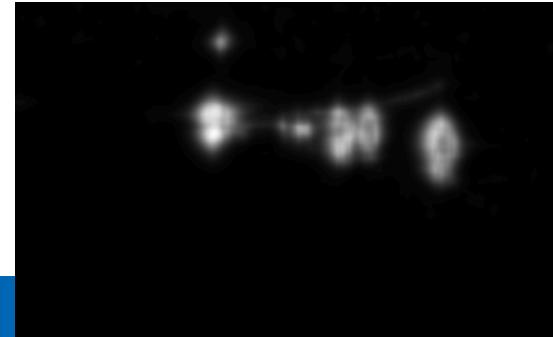


Parvo (details)



```
#include "opencv2/bioinspired.hpp"
//...
VideoCapture cap(argv[1]);
Ptr<bioinspired::Retina> retina;
Mat frame, parvo, magno;
for(;;) {
    cap >> frame; if(frame.empty()) break;
    if (!retina)
        retina = bioinspired::Retina::create(frame.size());
    retina->run(frame);
    retina->getParvo(parvo);
    retina->getMagno(magno);
    imshow("Parvo", parvo); imshow("Magno", magno);
    if(waitKey(30) != -1) break;
}
```

Magno (motion)



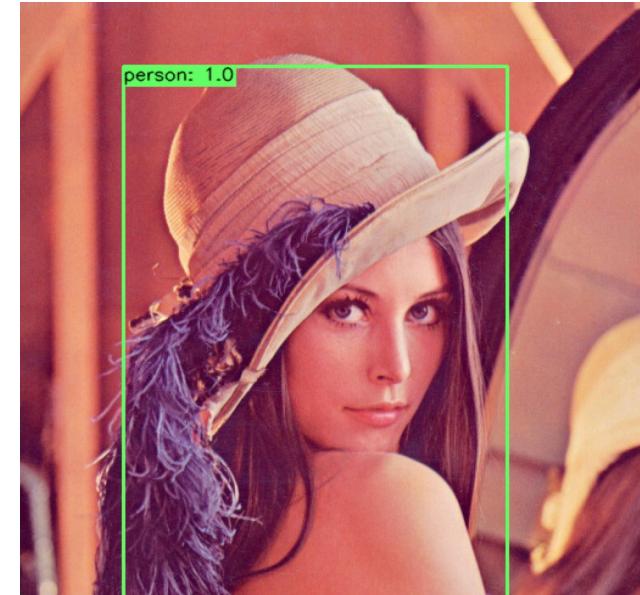
Использование OpenCV из Python

- Обертки генерируются полуавтоматически (требуется указать, какие функции надо оборачивать с использованием макросов)
- Вся основная функциональность доступна из Python.
- Используется собственный парсер C++ заголовков, написанный Python.
- Поддерживается Python 2.7.x и Python 3.x
- Особая обработка:
 - вложенных namespaces (превращаются в под-модули)
 - Mat – превращаются в «родные» массивы numpy
 - Input/OutputArray – OutputArray превращаются в возвращаемые значения (скаляр или tuple)
 - исключений – ловятся и превращаются в исключения Python
 - Ptr<> – оборачивается в простой указатель с Питоновским подсчетом ссылок



Детектирование объектов: OpenCV DNN + Python

```
import sys, numpy as np, cv2 as cv
size0 = 300
classNames = ('background', 'aeroplane', 'bicycle', 'bird', 'boat',
    'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable',
    'dog', 'horse', 'motorbike', 'person', 'pottedplant',
    'sheep', 'sofa', 'train', 'tvmonitor')
net = cv.dnn.readNetFromCaffe('MobileNetSSD_deploy_generated.prototxt',
    'MobileNetSSD_deploy_generated.caffemodel')
cap = cv.VideoCapture(0)
while True:
    ret, frame = cap.read(); if not ret: break
    frame_r = cv.resize(frame, (size0, size0))
    blob = cv.dnn.blobFromImage(frame_r, 1./127.5, (size0, size0), 127.5)
    net.setInput(blob)
    detections = net.forward()
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence < 0.2: continue
        class_id = int(detections[0, 0, i, 1])
        x0 = int(detections[0, 0, i, 3] * frame.shape[1])
        y0 = int(detections[0, 0, i, 4] * frame.shape[0])
        x1 = int(detections[0, 0, i, 5] * frame.shape[1])
        y1 = int(detections[0, 0, i, 6] * frame.shape[0])
        cv.rectangle(frame, (x0, y0), (x1, y1), (100, 255, 100))
        label = classNames[class_id] + ": " + str(round(confidence, 2))
        lsize, bl = cv.getTextSize(label, cv.FONT_HERSHEY_SIMPLEX, 0.5, 1)
        cv.rectangle(frame, (x0,y0), (x0+lsize[0], y0+lsize[1]+bl), (100,255,100), -1)
        cv.putText(frame, label, (x0, y0 + lsize[1]), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv.LINE_AA)
    cv.imshow("detections", frame)
    if cv.waitKey(30) >= 0: break
```

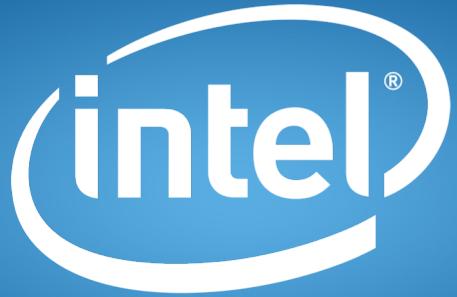


Будущее развитие

- OpenCV 4.x (октябрь – ноябрь 2018):
 - Переход на C++ 11
 - Замена устаревших алгоритмов на более продвинутые, в частности на алгоритмы использующие deep learning.
 - Дальнейшее увеличение производительности на CPU и GPU

Лекцию и исходный код примеров можно найти здесь:
http://github.com/vpisarev/opencv_lectures





experience
what's inside™