

Начинаем работать с библиотекой OpenCV

Вадим Писаревский, ведущий
инженер компании
ITSEEZ



Содержание

- Общая информация
- Основные ссылки
- Начинаем экспериментировать
- Использование OpenCV из Питона.
- ... и из C/C++
- Дополнительные ссылки, литература

Краткая справка

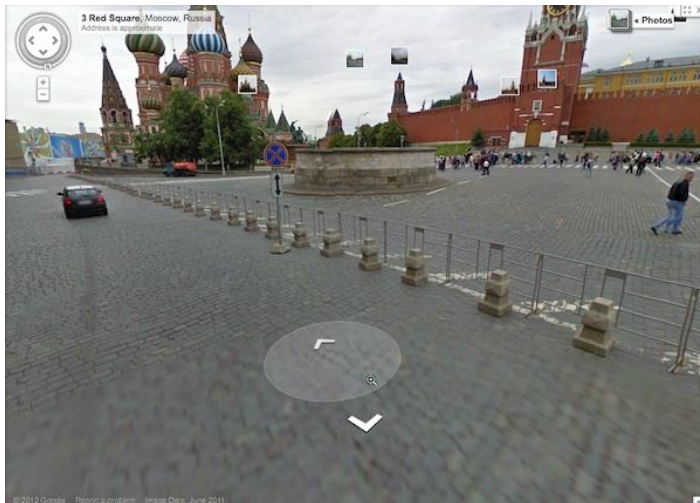
- Страничка: <http://opencv.org>
- Фактически, самая популярная библиотека компьютерного зрения.
- Написана на C/C++, исходный код открыт, включает более 1000 функций и алгоритмов.
- Лицензия BSD (разрешается бесплатное использование дома, для учебы, на работе)
- Разрабатывается с 1998г, сначала в Интел, теперь в компании Itseez при активном участии сообщества.
- >5000000 загрузок (без учета svn/git трафика)
- Используется многими компаниями, организациями, ВУЗами, например в NVidia, Willow Garage, Intel, Google, Stanford ...



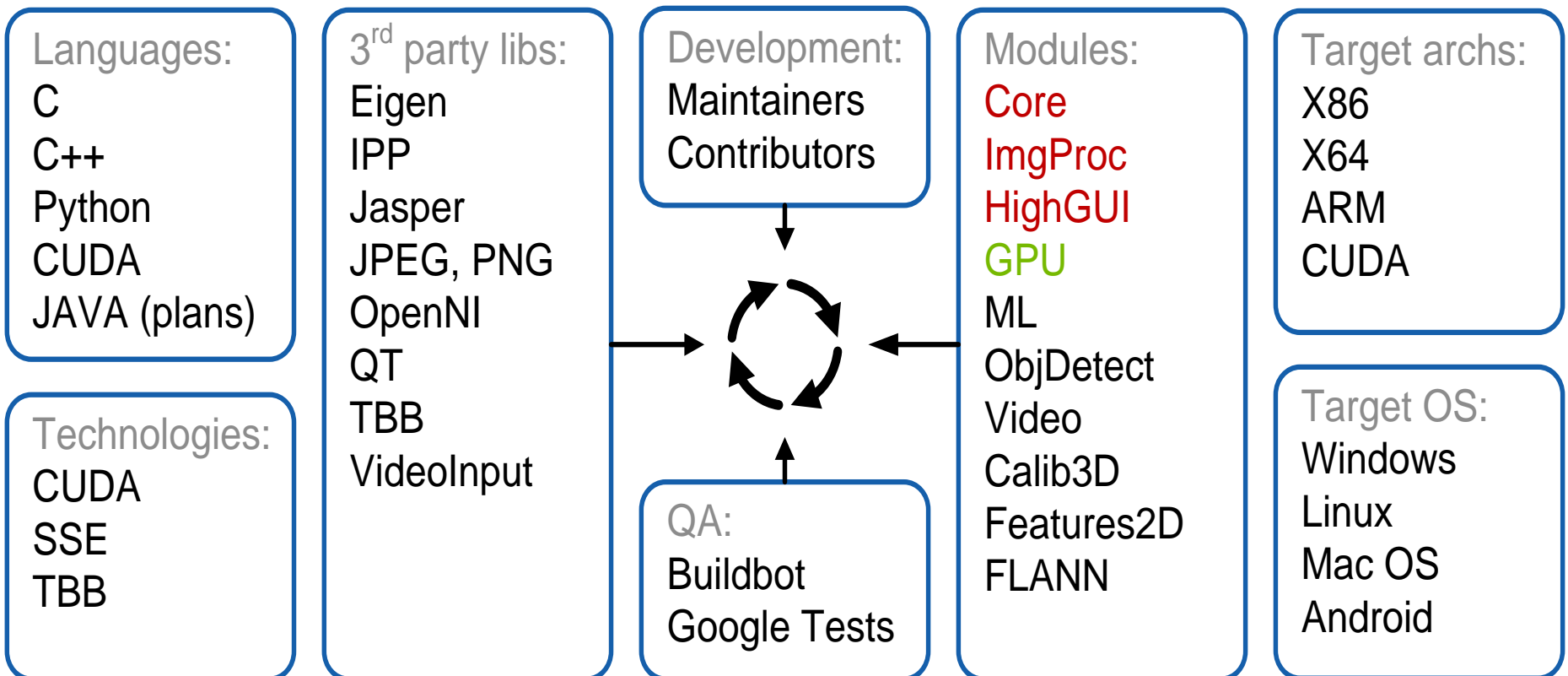
спонсоры

Некоторые примеры использования:

- Система зрения робота PR2, сделанного WillowGarage
- Аудио-визуальная инсталляция в Музее Современного Искусства (Сан-Франциско)
- Контроль качества монет, изготавливаемых Центробанком Китая
- Курсы компьютерного зрения в Стэнфорде
- Панорамы улиц в картах Google



Архитектура и разработка OpenCV



Функциональность

Базовая функциональность

A + B
Ax = B
FFT(A)
<?xml>...

Обработка изображений



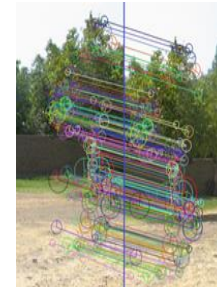
Фильтрация



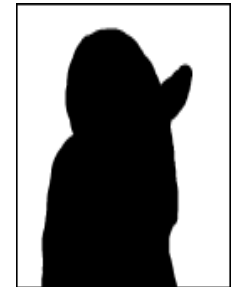
Трансформации



Ребра,
контурный
анализ

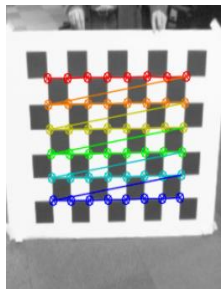


Особые точки

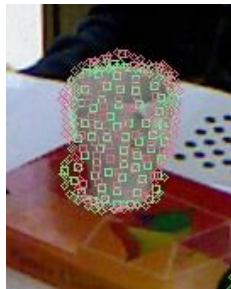


Сегментация

Видео, Стерео, 3D



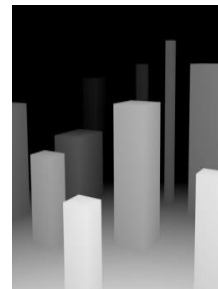
Калибрация
камер



Вычисление
положения в
пространстве



Оптический
поток

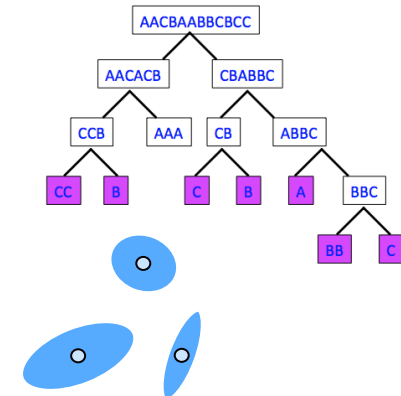


Построение
карты глубины



Нахождение
объектов

Машинное обучение



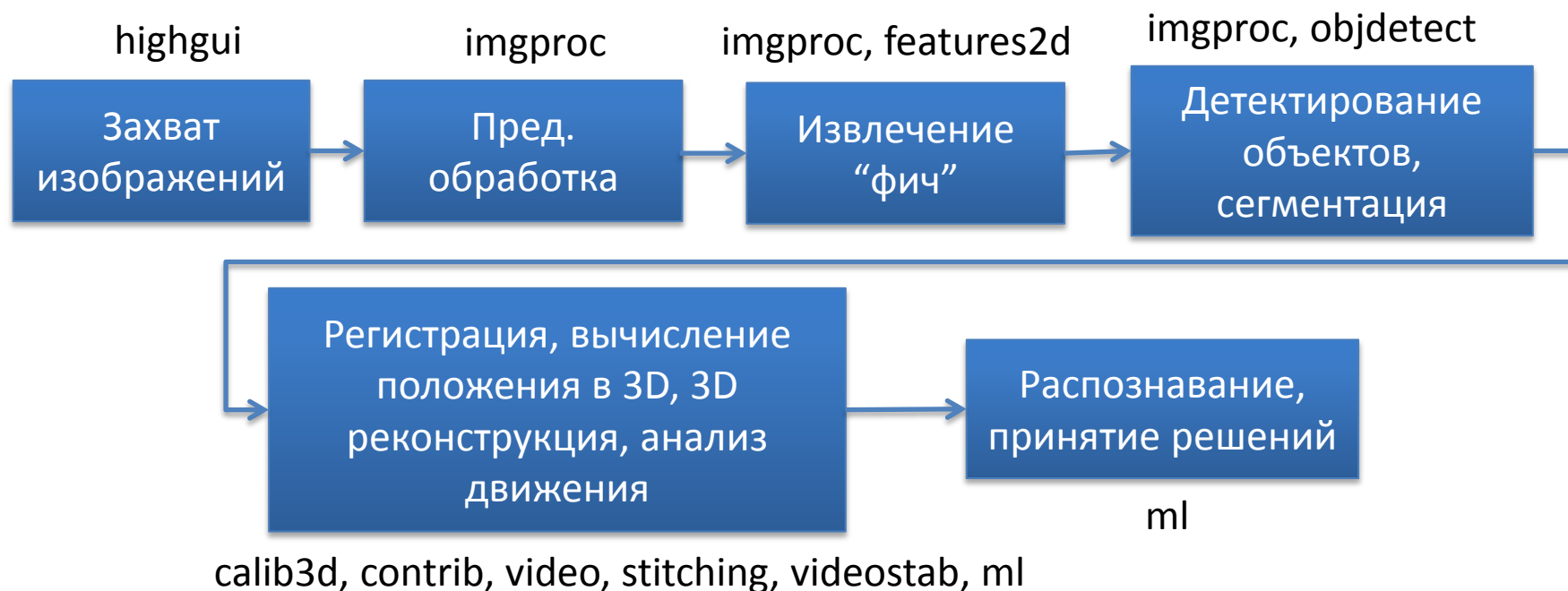
OpenCV в приложениях комп. зрения

OpenCV – базовая, в-целом низкоуровневая библиотека.

Мы создаем строительные блоки, кирпичики для приложений.

Сами приложения предстоит построить вам, пользователям.

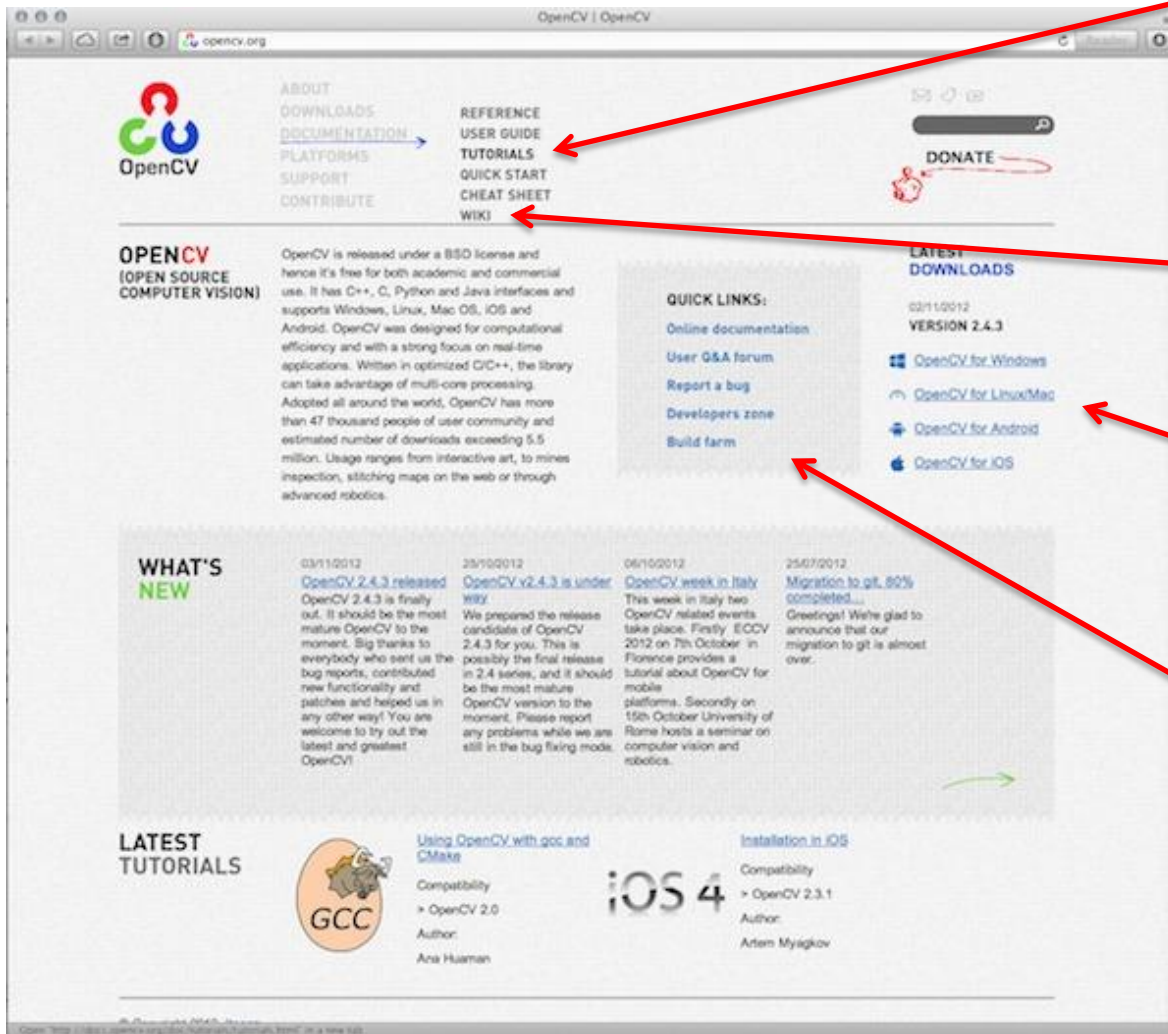
Общая схема типичного приложения CV



С чего начать?

<http://opencv.org>:

Читаем уроки,
Распечатываем
cheatsheet.



Ссылка WIKI ведет на
сайт для разработчиков
<http://code.opencv.org>

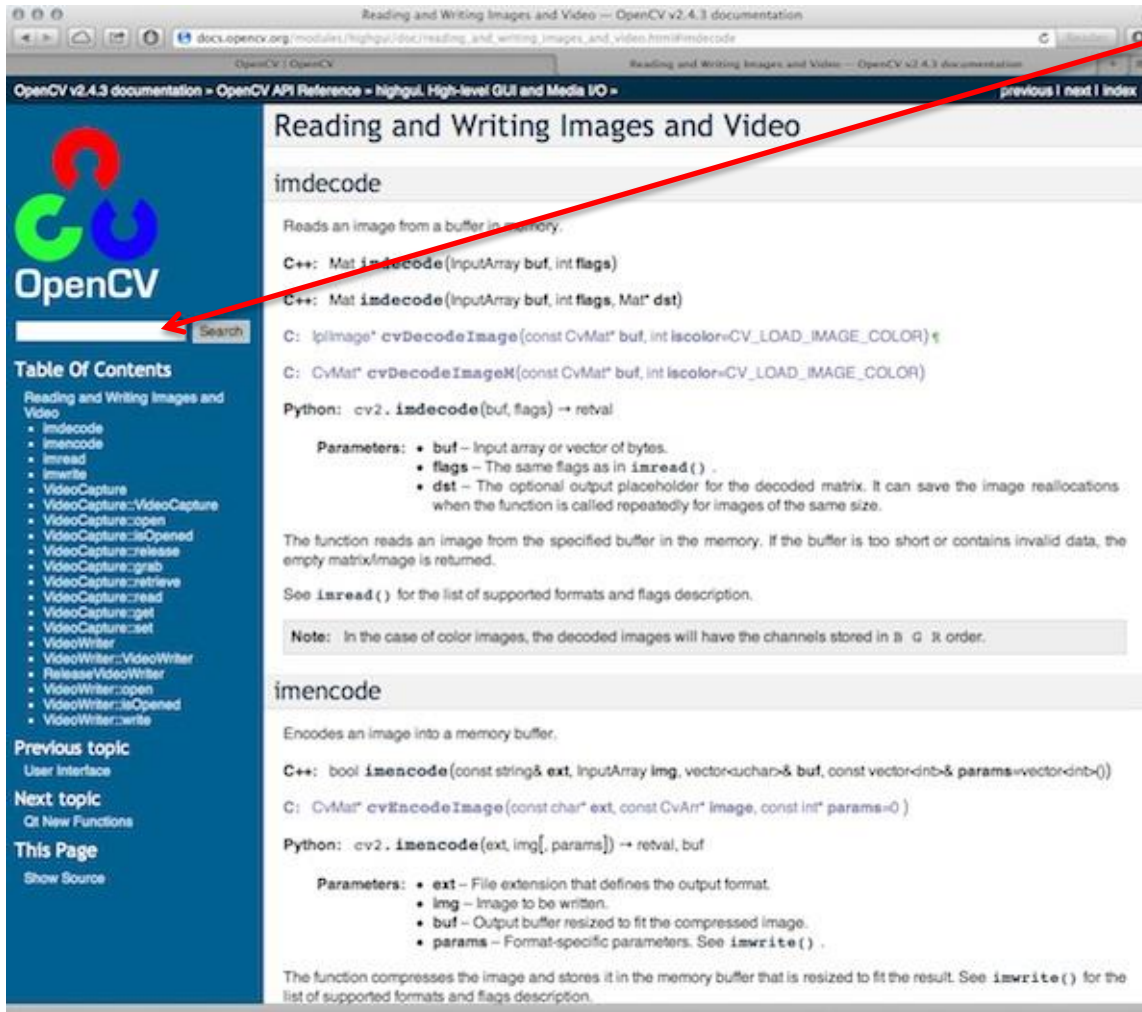
Качаем

Другие сайты OpenCV
(см. дальше)

Сайт с документацией

<http://docs.opencv.org>: справочник, руководство, уроки

Полнотекстовый
поиск по
документации.



OpenCV v2.4.3 documentation » OpenCV API Reference » highgui, High-level GUI and Media I/O »

Reading and Writing Images and Video

imdecode

Reads an image from a buffer in memory.

C++: `Mat imdecode(InputArray buf, int flags)`

C++: `Mat imdecode(InputArray buf, int flags, Mat* dst)`

C: `IplImage* cvDecodeImage(const CvMat* buf, int iscolor=CV_LOAD_IMAGE_COLOR)`

C: `CvMat* cvDecodeImageN(const CvMat* buf, int iscolor=CV_LOAD_IMAGE_COLOR)`

Python: `cv2.imdecode(buf, flags) → retval`

Parameters:

- `buf` – Input array or vector of bytes.
- `flags` – The same flags as in `imread()`.
- `dst` – The optional output placeholder for the decoded matrix. It can save the image reallocations when the function is called repeatedly for images of the same size.

The function reads an image from the specified buffer in the memory. If the buffer is too short or contains invalid data, the empty matrix/image is returned.

See `imread()` for the list of supported formats and flags description.

Note: In the case of color images, the decoded images will have the channels stored in B G R order.

imencode

Encodes an image into a memory buffer.

C++: `bool imencode(const string& ext, InputArray img, vector<uchar>& buf, const vector<int>& params=vector<int>())`

C: `CvMat* cvEncodeImage(const char* ext, const CvArr* image, const int* params=0)`

Python: `cv2.imencode(ext, img[, params]) → retval, buf`

Parameters:

- `ext` – File extension that defines the output format.
- `img` – Image to be written.
- `buf` – Output buffer resized to fit the compressed image.
- `params` – Format-specific parameters. See `imwrite()`.

The function compresses the image and stores it in the memory buffer that is resized to fit the result. See `imwrite()` for the list of supported formats and flags description.

Table Of Contents

- Reading and Writing Images and Video
 - imdecode
 - imencode
 - imread
 - imwrite
 - VideoCapture
 - VideoCapture::VideoCapture
 - VideoCapture::open
 - VideoCapture::isOpened
 - VideoCapture::release
 - VideoCapture::grab
 - VideoCapture::retrieve
 - VideoCapture::read
 - VideoCapture::get
 - VideoCapture::set
 - VideoWriter
 - VideoWriter::VideoWriter
 - ReleaseVideoWriter
 - VideoWriter::open
 - VideoWriter::isOpened
 - VideoWriter::write

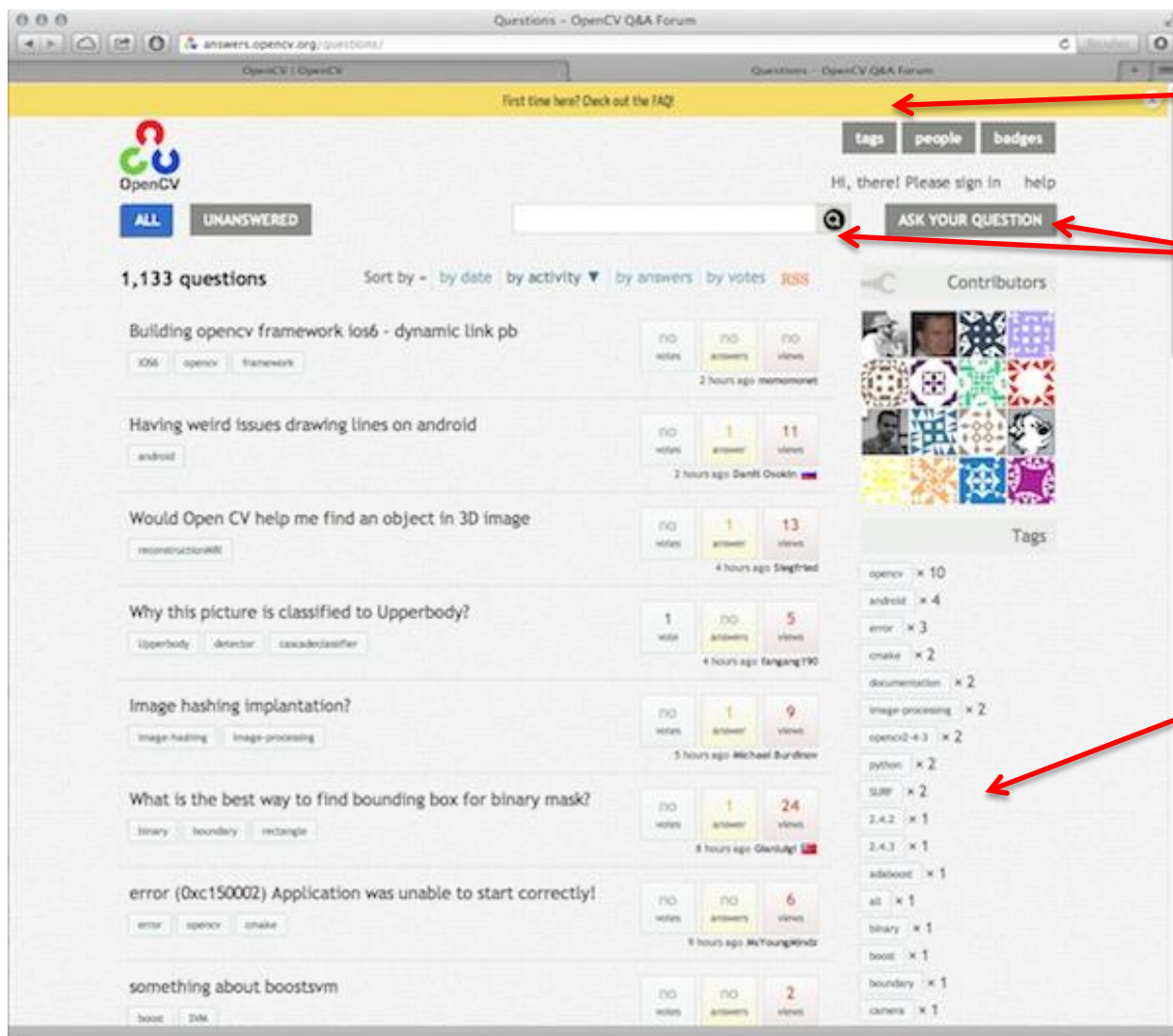
Previous topic
User Interface

Next topic
Qt New Functions

This Page
Show Source

Задать вопрос, найти ответ

<http://answers.opencv.org>: сделан по аналогии со StackOverflow



Начинаем с FAQ

1. Поискать ответы
2. Задать свой вопрос

теги

Самый быстрый старт (на примере Windows)

1. Качаем самораспаковывающийся архив с SourceForge:
<http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.3/>, распаковываем куда-нибудь в домашний каталог
2. Качаем Python 2.7.x с <http://python.org>, качаем расширение Питона для вычислений NumPy:
<http://numpy.scipy.org> (выбираем версию, соответствующую вашему Питону). Ставим и то и другое.
3. Копируем cv2.pyd из распакованного каталога OpenCV в <каталог Python>\Lib\site-packages.

Можно начинать эксперименты!

4. Опционально устанавливаем редактор с поддержкой Питона, например Sublime Text 2.

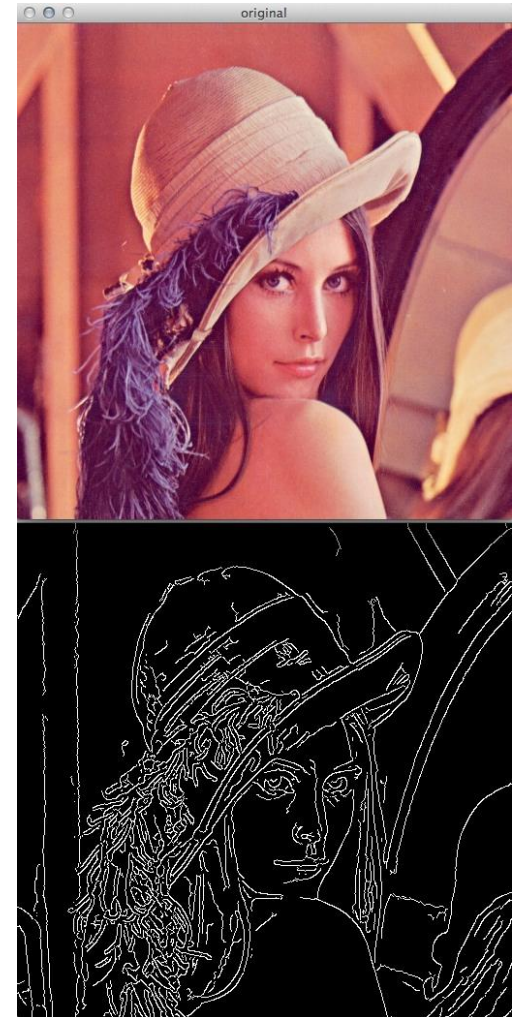
Первая программа

Набираем следующую программу
в текстовом редакторе (first.py):

```
import sys, cv2 as cv
img = cv.imread(sys.argv[1], 1)
cv.imshow("original", img)
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
gray = cv.GaussianBlur(gray, (7, 7), 1.5)
edges = cv.Canny(gray, 0, 50)
cv.imshow("edges", edges)
cv.waitKey()
```

Копируем файл lena.jpg из <opencv>/samples/c
в каталог с first.py, запускаем программу (из Far,
cmd, ...):

```
python first.py lena.jpg
```



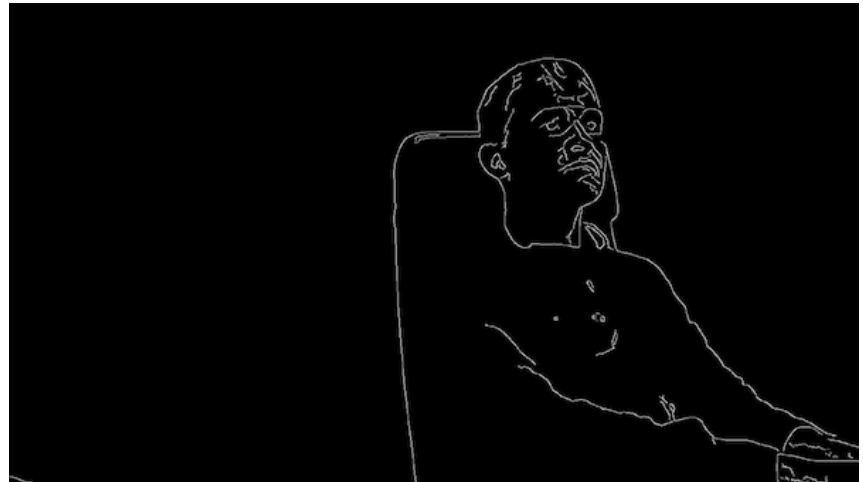
Вторая программа: добавляем видео

Редактируем first.py, сохраняем как second.py:

```
import sys, cv2 as cv
cap = cv.VideoCapture(0)
while True:
    ok, img = cap.read()
    if not ok:
        break
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    gray = cv.GaussianBlur(gray, (7, 7), 1.5)
    edges = cv.Canny(gray, 1, 50)
    cv.imshow("edges", edges)
    if cv.waitKey(30) > 0:
        break
```

Запускаем

python second.py



Третья программа: добавляем детектирование лиц

Редактируем second.py, сохраняем как face.py:

```
import sys, cv2 as cv
cap = cv.VideoCapture(0)
cascade = cv.CascadeClassifier(
    "lbpcascade_frontalface.xml")

while True:
    ok, img = cap.read()
    if not ok:
        break
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    sf = min(640./img.shape[1], 480./img.shape[0])
    gray = cv.resize(gray, (0,0), None, sf, sf)
    rects = cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=4,
                                     minSize=(40, 40), flags=cv.CV_HAAR_SCALE_IMAGE)
    gray = cv.GaussianBlur(gray, (3, 3), 1.1)
    edges = cv.Canny(gray, 5, 50)
    out = cv.cvtColor(edges, cv.COLOR_GRAY2BGR)
    for x, y, w, h in rects:
        cv.rectangle(out, (x, y), (x+w, y+h), (0,0,255), 2)

    cv.imshow("edges+face", out)
    if cv.waitKey(30) > 0:
        break
```



Копируем lbpcascade_frontalface.xml из <OpenCV>/data/lbpcascades в наш каталог. Запускаем:

python face.py

OpenCV+Python: вопросы и ответы

1. Какая часть OpenCV доступна из Питона?

почти вся интересная функциональность

2. А оно быстро работает?

достаточно, особенно если использовать идеологию Матлаба
– векторизация кода, минимум ручной обработки в циклах

3. Как узнать, какие функции использовать и как их использовать?

библиотеки Питон, в том числе наш модуль cv2,
содержат краткую информацию прямо внутри себя:

```
>>> import cv2 as cv
>>> dir(cv) # список функций и классов в cv
>>> cv.resize.__doc__ # описание resize
>>> cap = cv.VideoCapture(0)
>>> dir(cap) # список методов VideoCapture
>>> cap.read.__doc__ # описание метода read
```

в каталоге <OpenCV>/samples/python2 находится большое количество
очень интересных примеров, рекомендуется ознакомиться

Что насчет C/C++?

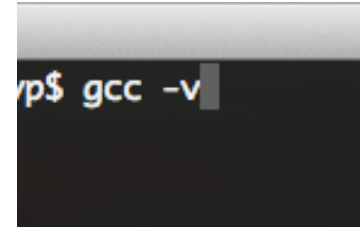
Вам понадобится компилятор, среда разработки на C/C++ *



Visual Studio

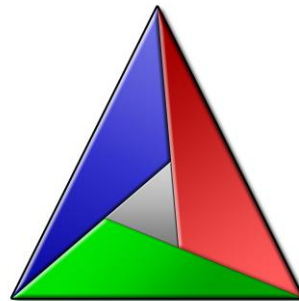


Xcode



GCC + make (Linux)

(*) по возможности, не используйте MinGW, Borland C++, Sun Studio и разные экзотические компиляторы



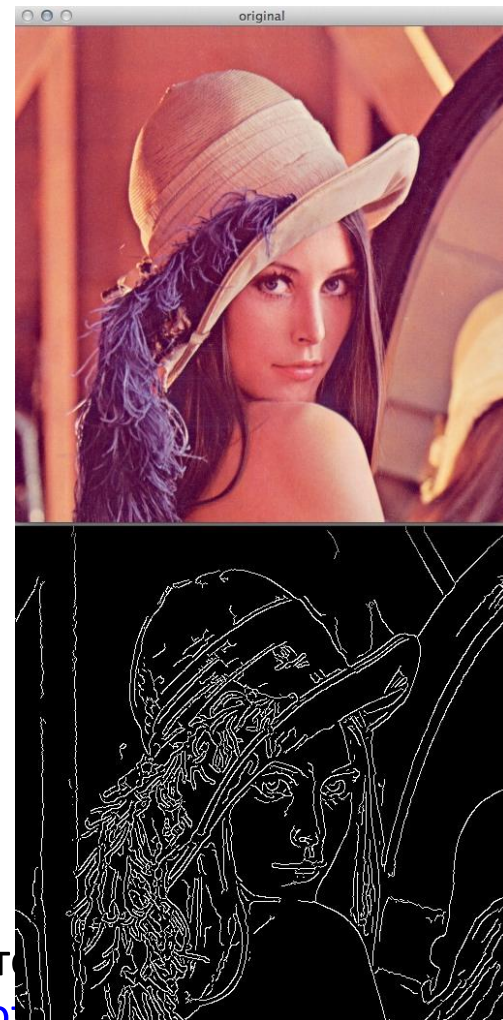
А также CMake
(под Windows необязателен, но желателен).

Первая программа на C/C++

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

using namespace cv;
int main(int argc, char** argv)
{
    Mat img, gray, edges;
    img = imread(argv[1], 1);
    imshow("original", img);
    cvtColor(img, gray, COLOR_BGR2GRAY);
    GaussianBlur(gray, gray, Size(7, 7), 1.5);
    Canny(gray, edges, 0, 50);
    imshow("edges", edges);
    waitKey();
    return 0;
}
```

Как скомпилировать и запустить эту программу? Читайте http://docs.opencv.org/doc/tutorials/introduction/table_of_content_introduction/table_of_content_introduction.html



cv::Mat – многомерный многоканальный массив

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

- Размеры, step
- Счетчик ссылок
- Указатель на данные

```
cv::Mat B = A;
```

- Размеры, step
- Счетчик ссылок
- Указатель на данные

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

- Размеры ROI, step
- Счетчик ссылок
- Указатель на данные

Элементы/Пиксели

RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGB*****

Расположение в памяти матрицы C

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

cv::Mat и std::vector

`std::vector<Point3f>`

XYZXYZXYZXYZXYZXYZXYZXYZXYZXYZXYZXYZXYZXYZ

Массив из N точек



`cv::Mat`

- Размеры, step
- Счетчик ссылок (отсутствует)
- Указатель на данные

`cv::Mat`

RGB
RGB
RGB
...
RGB

Nx1 3-канальное изображение

Работаем с матрицами

```
Mat M(480,640,CV_8UC1); // Создаем полутоновую картинку 640x480
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

// оцениваем “резкость” в выбранном ROI,
// например для реализации автофокуса
Mat_<Vec3b>::iterator it= subM.begin<Vec3b>(),
                      itEnd = subM.end<Vec3b>();

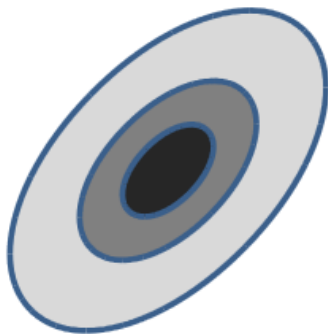
float contrast = 0.f;
for(; it != itEnd; ++it) {
    uchar* ptr = &(*it);
    int dx = ptr[1] - ptr[-1], dy = ptr[subM.step] - ptr[-subM.step];
    contrast += sqrtf((float)(dx*dx + dy*dy));
}
```

Пример использования OpenCV: поиск плоских объектов



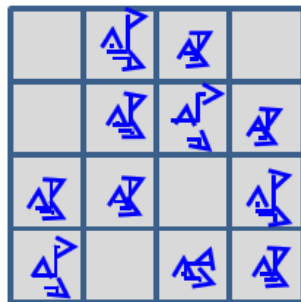
Будем использовать модуль Features2d

Детекторы



- SIFT
- SURF
- FAST
- STAR
- MSER
- HARRIS
- GFTT (Good Features To Track)

Дескрипторы



- SIFT
- SURF
- HoG
- ORB
- FREAK, BRISK
- ...

Сравнение дескрипторов

- BruteForce
- FlannBased
- Bag-Of-Words

Пост-обработка

- Cross check
- Ratio check

Поиск плоских объектов с помощью Features 2D

// Читаем картинки

```
Mat img1 = imread(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
```

```
Mat img2 = imread(argv[2], CV_LOAD_IMAGE_GRAYSCALE);
```

// Находим особые точки на обеих картинках, считаем их описатели

```
Ptr<Feature2D> surf=Algorithm::create<Feature2D>("Feature2D.SURF");
```

```
vector<KeyPoint> keypoints1, keypoints2;
```

```
Mat descriptors1, descriptors2;
```

```
surf->operator()(img1, Mat(), keypoints1, descriptors1);
```

```
surf->operator()(img2, Mat(), keypoints2, descriptors2);
```

// Находим соответствия “в лоб”

```
vector<DMatch> matches;
```

```
BFMatcher(NORM_L2, true).match(descriptors1, descriptors2, matches);
```

// Находим оптимальное преобразование, согласующееся с большинством пар точек.

```
vector<Point2f> pt1, pt2;
```

```
for( size_t i = 0; i < matches.size(); i++ ) {
```

```
    pt1.push_back(keypoints1[matches[i].queryIdx].pt);
```

```
    pt2.push_back(keypoints2[matches[i].trainIdx].pt);
```

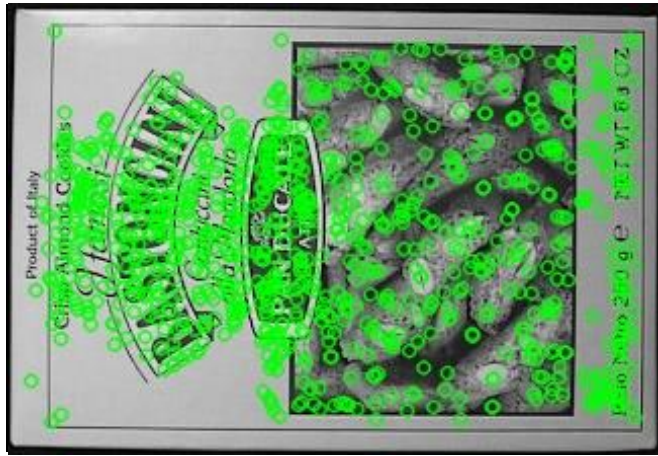
```
}
```

```
Mat H = findHomography(pt1, pt2, RANSAC, 10);
```

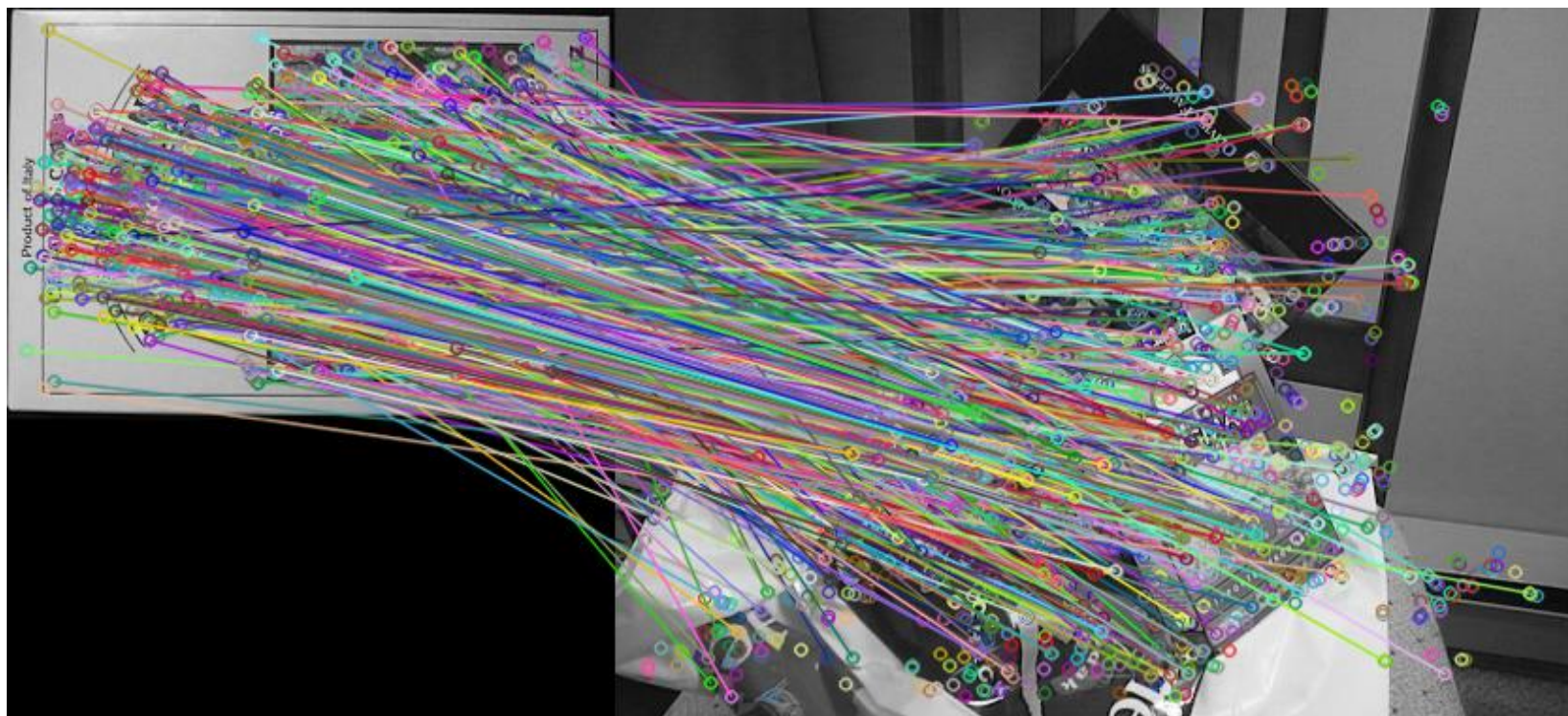
// ***** H – это матрица оптимального перспективного преобразования от img1 к img2. *****

// этот алгоритм также позволяет находить “плоские” текстурные предметы на изображениях

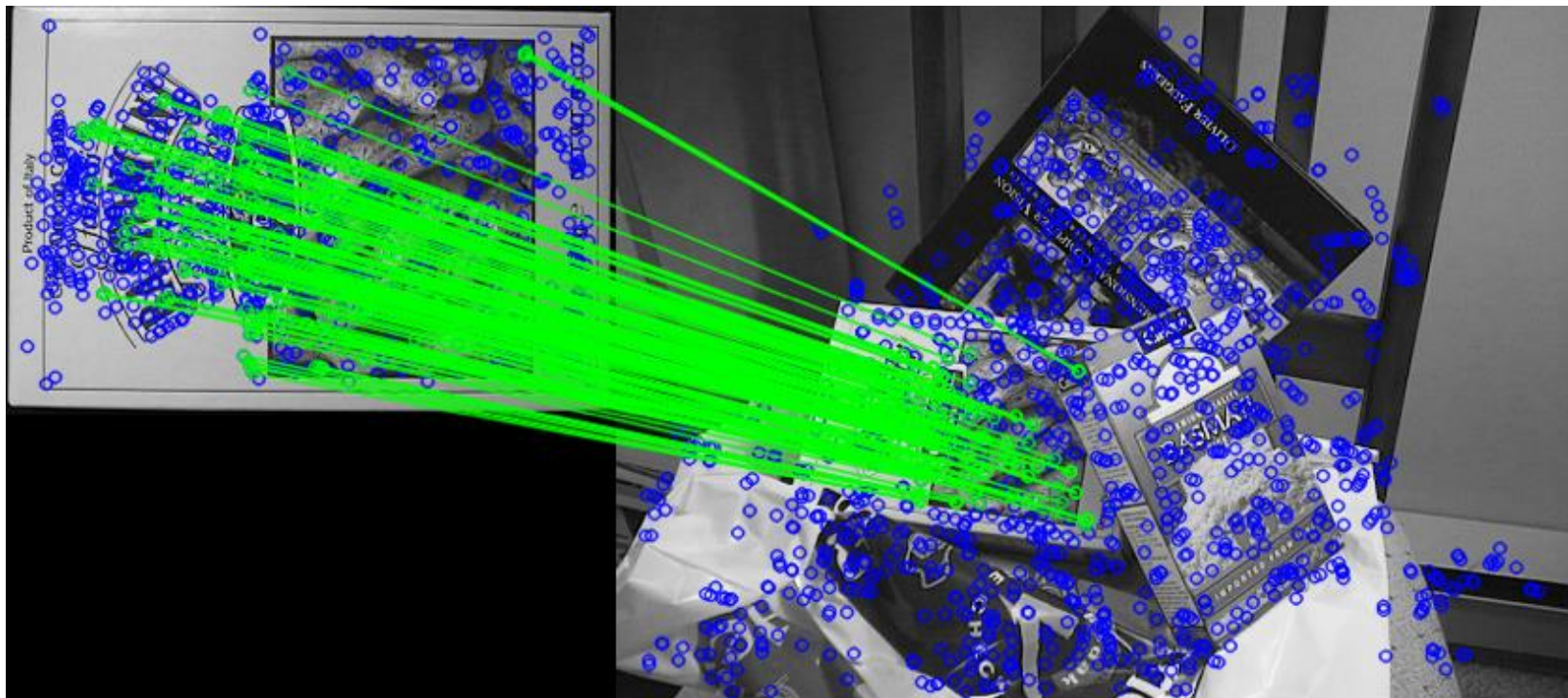
Найденные точки



Пары точек после сравнения дескрипторов



Оставшиеся правильные пары после поиска матрицы гомографии



1. Как теперь найти положение коробки в пространстве?
 - Использовать функцию `solvePnP()`.
2. Что делать в случае не плоских объектов?
 - Найти правильные пары с помощью `findFundamentalMat()` вместо `findHomography()`.
 - После чего опять использовать `solvePnP()`.

Готовый пример

- Загружаем, компилируем OpenCV
- Запускаем `matcher_simple`:
`matcher_simple box.png box_in_scene.png`
(картинки копируем из `opencv/samples/c`)
- Можно покрутить параметры, заменить SURF на SIFT, попробовать использовать свои картинки

Литература, дополнительные ссылки

Лучшая современная книга по компьютерному зрению (на английском) :

<http://szeliski.org/Book/>

“Официальная” книга про

OpenCV:<http://www.amazon.com/Learning-OpenCV-Computer-Vision-Library/dp/0596516134>

(также доступна на Озоне. описывает OpenCV 1.x!)

Сейчас завершается работа над вторым изданием

Примеры кода из книги выше:

<http://examples.oreilly.com/9780596516130/>

“Официальный” учебник по Питону на русском:

http://ru.wikibooks.org/wiki/Учебник_Python_2.6

Numpy – превращает Питон в аналог Матлаба (и необходим для использования OpenCV с Питоном):

<http://docs.scipy.org/doc/>

