Тема 4
Программный инструментарий для решения задач технического зрения
OpenCV

OpenCV

- В языке оригинала
-
- http://locv.ru
- http://habrahabr.ru/search/?q=[opencv]&target_type
- http://robocraft.ru/page/opencv/ → устарело

  Презентация Вадима Писаревского
- http://www.myshared.ru/slide/202801/

OpenCV

- OpenCV= OpenSouce Computer Vision Library

- Библиотека компьютерного зрения с открытым исходным кодом(Open Source Computer Vision Library), содержащая более 500 функций, нацеленных под выполнение задач в реальном времени

OpenCv

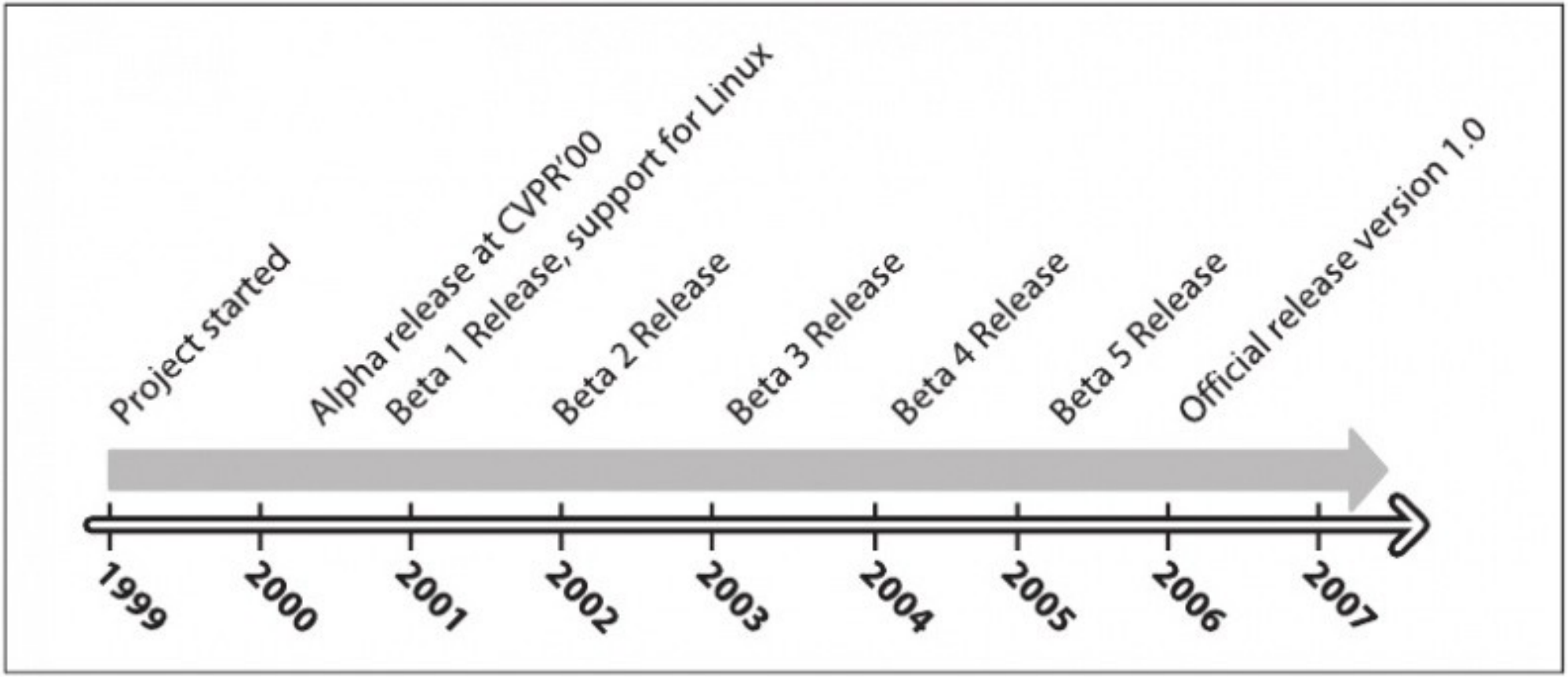
- До первой версии разработкой занималась русская команда Intel в Нижнем Новгороде. Главным в русской команде был Вадим Писаревский, который и по прежнему вкладывает много усилий в OpenCV, вместе с ним Виктор Ерухимов, Валерий Курякин и многие другие.
- Библиотека содержит алгоритмы для захвата и обработки изображений, распознавания образов, калибровки камер,слежения за объектом, 3D и многое другое.

Цели создания OpenCV*

- Утверждение общего стандартного интерфейса компьютерного зрения и для приложений в данной области. Для способствования роста числа таких приложений и создания новых моделей использования PC
- Сделать платформы Intel привлекательными для разработчиков таких приложений за счет дополнительного ускорения OpenCV с помощью Intel Perfomance Libraries.

* презентация Вадима Писаревского

Этапы развития проекта до выпуска версии 1.0



Развитие OpenCV

Архитектура и платформы

- Библиотека написана на C/C++
- Поддерживаются компиляторы:
  - Windows - Microsoft Visual C++, Borland C++, Intel Compiler, MinGW
  - Linux - GCC, Intel Compiler
  - Mac - Intel Compiler, Carbon и др.
- Есть порты под Linux,Win,Mac,Android*,IOS*

  (* могут поддерживаться не все функции)

API

http://opencv.org/downloads.html

- Обертки: C/C++, Python, Java
- Параллелизм: поддержка IPP, CUDA(!), OpenCL

- Следующий релиз под номером 3 доступено в alpha. В нём произойдут значительные изменения, и бинарная совместимость будет нарушена с целью модернизации архитектуры и API библиотеки.

http://www.myshared.ru/slide/202801/#

Функциональность OpenCV в нескольких словах

- Ядро cxcore, также частично используется в Intel Open Source Probabilistic Network Library:
  - Базовые операции над многомерными числовыми массивами
  - Матричная алгебра, математические ф-ции, генераторы случайных чисел
  - DFT, DCT
  - Запись/восстановление структур данных в/из XML/YAML
  - Базовые функции 2D графики
  - Поддержка более сложных структур данных: разреженные массивы, динамически растущие последовательности, графы
- Модуль обработки изображений и компьютерного зрения cv:
  - Базовые операции над изображениями (фильтрация, геометрические преобразования, преобразование цветовых пространств и т.д.)
  - Анализ изображений (выбор отличительных признаков, морфология, поиск контуров, гистограммы)
  - Структурный анализ (описание форм, плоские разбиения, …)
  - Анализ движения, слежение за объектами
  - Обнаружение объектов, в частности лиц
  - Калибровка камер, элементы восстановления пространственной структуры
- Модуль для ввода/вывода изображений и видео, пользовательского интерфейса highgui:
  - Захват видео с камер и из видео файлов, чтение/запись статических изображений.
  - Функции для организации простого UI (сейчас все демо приложения используют HighGUI)
- Экспериментальные и устаревшие функции cvaux:
  - Пространств. зрение: стерео калибрация, само калибрация
  - Поиск стерео-соответствия, клики в графах
  - Нахождение и описание черт лица
  - Сравнение форм, построение скелетонов …
  - Скрытые Марковские цепи
  - Описание текстур

Разработчики о лицензии locv.ru

Открытая лицензия для OpenCV была составлена таким образом, чтобы было возможно создавать коммерческие приложение, используя любые возможности OpenCV. Вы не обязаны делать Ваш проект открытым или публиковать изменения, внесенные вами в библиотеку, хотя мы надеемся, что вы это сделаете. Отчасти из-за таких либеральных условий существует большое сообщество пользователей, включающее в себя такие крупные компании как IBM, Microsoft, Intel, Sony, Siemens, Google, и это далеко не полный список, а также научно-исследовательские центры, такие как Стэнфорд, Массачусетский технологический институт, CMU, Кембридж, и INRIA. Есть также группа на Yahoo где пользователи могут задавать вопросы, насчитывающая более 20 тысяч человек. OpenCV популярна во всём мире, причём большие сообщества пользователей можно найти в Китае, Японии, России, Европе и Израиле.

“Stanley” из Стэнфорда

победитель DARPA Grand Challenge (2005)

Модули OpenCV

CXCORE – Ядро, содержит:

- Базовые структуры
- Матричную алгебру
- Алгоритмы работы с памятью
- Алгоритмы преобразования типов
- Алгоритмы для обработки ошибок
- Функции для записи/чтения XML файлов
- Функции для работы с 2D
-

Модули

CV – Модуль обработки изображений, работа с компьютерным зрением, содержит:

- Функции для работы с изображениями (преобразование, фильтрация и т.д.)
- Функции для анализа изображений (поиск контуров, гистограммы и т.д.)
- Алгоритмы анализа движений, слежение за объектами
- Алгоритмы распознания объектов (лиц, предметов)
- Алгоритмы для калибровки камер
-

- ML – Обучение машин:
- Функции для классификации и анализа данных
-
- HighGUI – Модуль для создания пользовательского интерфейса, отвечает за:
- Создание окон
- Вывод изображений
- Захват видео из файлов и камер
- Чтение/Запись изображений
-

CVCAM – Захват видео с цифровых камер

-

CVAUX – Устаревшие функции:

- Пространственное зрение
- Нахождение и описание черт лица
- Поиск стерео соответствий
- Описание текстур

Документацию на OpenCV можно найти в …/opencv/doc в виде html файлов

OpenCV Ansi / C++

- Большинство функций OpenCV доступны как через ++ объекты, так и через ANSI функции. Намечается тенденция большей «плюсоватости» новых функций, некоторые из них использовать в ANSI становится весьма затруднительно

Основной объект OpenCV - IplImage

- IplImage *img = cvLoadImage(“./test.jpg”);

  IplImage- массив (объект для ++), содержащий точки изображения.

  По сути это объект CvMat, но с некоторыми дополнениями для того, чтобы матрица интерпретировалась как изображение. Изначально эта структура была частью библиотеки обработки изображений(IPL) Intel.

Типы изображений

Таблица 3-2. Типы изображений OpenCV.

Макрос	*Тип пикселя*
IPL_DEPTH_8U	Беззнаковый 8-битный (8u)
IPL_DEPTH_8S	Знаковый 8-битный (8s)
IPL_DEPTH_16S	Знаковый 16-битный (16s)
IPL_DEPTH_32S	Знаковый 32-битный (32s)
IPL_DEPTH_32F	32-битный с плавающей запятой (32f)
IPL_DEPTH_64F	64-битный с плавающей запятой (64f)

```
typedef struct _IplImage {
  int                nSize; // sizeof(IplImage)
  int                ID; // Версия (=0)
  int                nChannels; // Число каналов
  int                alphaChannel; // Альфа-канал
  int                depth; // Глубина в битах
  char               colorModel[4]; // Не используется в OpenCV
  char               channelSeq[4]; // Аналогично
  int                dataOrder; // Расположение каналов
  int                origin; // Начало координат
  int                align; // Выравнивание строк изображения (OpenCV использует
widthStep)
  int                width; // Ширина
  int                height; // Высота
  struct _IplROI*      roi; // ROI
  struct _IplImage*    maskROI; // =0
  void*              imageId; //
  struct _IplTileInfo* tileInfo; //
  int                imageSize; // Память выделенная под изображение
  char*              imageData; // Данные изображения
  int                widthStep; // Число байт в одной строке изображения
  int                BorderMode[4]; // Не используется в OpenCV
  int                BorderConst[4]; // Аналогично
  char*              imageDataOrigin; // Используется для правльного
освобождения памяти
} IplImage;
```

HighGUI

- Для того,чтобы разработчику сконцентрироваться на процессе разработки, а не на окружении, в библиотеке есть все минимально необходимые компоненты:
  - Распаковка и чтения данных из популярных форматов хранения файлов
  - Получение данных с камеры через V4L,dc1934,
  - Работа с окнами и событиями с клавиатуры
  - Преобразования изображений
  - Вывод изображений и видео
  -
  -

```
#include "cv.h"
#include "highgui.h"
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[])
{
	// получаем любую подключённую камеру
	CvCapture* capture = cvCreateCameraCapture(CV_CAP_ANY);
//cvCaptureFromCAM( 0 );
	assert( capture );

	//cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 640);//1280);
	//cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 480);//960);

	// узнаем ширину и высоту кадра
	double width = cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH);
	double height = cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT);
	printf("[i] %.0f x %.0f\n", width, height );

	IplImage* frame=0;

	cvNamedWindow("capture", CV_WINDOW_AUTOSIZE);

	int counter=0;
	char filename[512];
```

Флаги и ключи для компилятора

```
gcc  getimage.c `pkg-config opencv --cflags --libs ` -o test.bin
```

Конвертирование изображения

- void cvConvertImage(const CvArr* src,
  CvArr* dst,
  int flags = 0
  );
- Тип что во что конвертировать задается типом структуры
- Flags умеет переворачивать изображение.

Полезные примитивы cxtypes.h

- cvPoint(X,Y) — точка
- cvPoint2D32f и CvPoint3D32f -
- cvSize(w,h) — свойства объекта длина и ширина
- cvRect(X,Y,X1,Y1) — область прямоугольник
-

```
cvRectangle(
myImg, // Изображение
  cvPoint(10,10), // Верхний левый угол
  cvPoint(60,80), // Нижний правый
  cvScalar(250,255,255) // Цвет
);
```

В ++ - конструктор имеет схожие свойства, отличается заглавной CvPoint

Таблица 3-1. Структура точки, размера, прямоугольника и скаляра

Структура	Содержит	Представляет
`CvPoint`	`int x, y`	Точка изображения
`CvPoint2D32f`	`float x, y`	2D точка
`CvPoint3D32f`	`float x, y, z`	3D точка
`CvSize`	`int width, height`	Размер изображения
`CvRect`	`int x, y, width, height`	Часть изображения
`CvScalar`	`double val[4]`	Значение RGBA

Структура - CvMat

- Прототип создающей матрицу функции выглядит так:

  cvMat* cvCreateMat (int rows, int cols, int type);
- Здесь type может быть любым из длинного списка предопределенных типов, тип задаётся так: CV_<глубина в битах>(S|U|F)C<число каналов>. («CV_8UC1 means an 8-bit unsigned single-channel matrix, CV_32SC2 means a 32-bit signed matrix with two channels», как сказано в Вики)

ROI

ROI и widthStep имеют очень большое практическое значение, поскольку в некоторых ситуациях они ускоряют операции компьютерного зрения, позволяя им обрабатывать лишь часть изображения. ROI и widthStep универсальны в OpenCV, любая функция умеет работать с ними. Для установки и сброса ROI существуют функции cvSetImageROI() и cvResetImageROI().

- Повышает скорость обработки
- Автоматически позволяет задать регион где нужно искать «искомое»

Установили ROI

- cvSetImageROI(src, cvRect(x,y,width,height)); // Устанавливаем ROI
-
- С этого момента все операции будут проводиться именно с этим фрагментом.
- Важно также сбросить ROI перед выводом изображения, иначе на экран выведется только его часть.
- cvResetImageROI(src); // Сбрасываем ROI

WidthStep

- Свойство изображения, которое показывает сколько байт в памяти занимает одна запись, соответствующая одной точке изображения.
- Как выполнить цикл по элементам памяти, как заполнить данные «вручную» ?

Доступ к элементам изображения

```
unsigned char* ptrRGB;
cvSetData(imageIR, dataIR, 640);
ptrRGB=(unsigned char *)(imageRGB->imageData);.

shift_x=0;
shift_y=0;
for(j=0;j<480;j++)
  for(i=0;i<640;i++)
    {
    ptrRGB[(i)*3+(j)*imageRGB->widthStep]=ptrIR[i+j*640];
    ptrRGB[(i)*3+(j)*imageRGB->widthStep+1]=ptrIR[i+j*640];
    ptrRGB[(i)*3+(j)*imageRGB->widthStep+2]=ptrIR[i+j*640];

    }
```

$$\delta = (\text{строка}) \cdot N_{\text{столбцов}} \cdot N_{\text{каналов}} + (\text{столбец}) \cdot N_{\text{каналов}} + (\text{канал})$$

Операции с матрицами и изображениями

Рисование в изображении-line

```
void  cvLine(
  CvArr*   array, // Изображение
  CvPoint  pt1, // Начальная точка
  CvPoint  pt2, // Конечная точка
  CvScalar color, // Цвет (можно установить макрсом CV_RGB())
  int      thickness    = 1, // Толщина
  int      connectivity = 8 // Сглаживание
);
```

Прямоугольники и квадраты

```
void  cvRectangle(
  CvArr*   array, // Изображение
  CvPoint  pt1, // Верхний левый угол
  CvPoint  pt2, // Нижний правый угол
  CvScalar color, // Цвет
  int      thickness = 1 // Толщина
);
```

Круги и эллипсы

```
void  cvCircle (
  CvArr*   array, // Изображение
  CvPoint  center, // Центр окружности
  int      radius, // Радиус
  CvScalar color, // Цвет
  int      thickness    = 1, // Толщина (-1 - залитый)
  int      connectivity = 8 // Сглаживание
);
```

Эллипс

```
void cvEllipse(
  CvArr*   img, // Изображение
  CvPoint  center, // Центр
  CvSize   axes, // Длина большой и малой оси (CvSize(ширина,высота))
  double   angle, // Угол от оси X, против часовой стрелки
  double   start_angle, // Начальный угол (0 Для полного)
  double   end_angle, // Конечный угол (360 для полного)
  CvScalar color, // Цвет
  int      thickness = 1, // Толщина
  int      line_type = 8 // Сглаживание
);
```

Полигоны

```
void cvFillPoly(
  CvArr*    img, // Изображение
  CvPoint** pts, // Массив точек
  int*      npts, // Кол-во точек
  int       contours, //
  CvScalar  color, // Цвет
  int       line_type = 8 // Сглаживание
);

void cvFillConvexPoly(
  CvArr*   img, // Изображение
  CvPoint* pts, // Массив точек
  int      npts, // Кол-во точек
  CvScalar color, // Цвет
  int      line_type = 8  ); // Сглаживание
```

Шрифты и тексты

```
void cvPutText(
  CvArr*        img, // Изображение
  const char*   text, // Текст
  CvPoint       origin, // Нижний левый угол от
//которого отрисовывается текст
  const CvFont* font, // Шрифт
  CvScalar      color // Цвет
);
```

origin это координаты нижнего левого угла от которого будет печататься текст.

```
void cvInitFont(
  CvFont* font, // Указатель на CvFont
  int     font_face, // Шрифт (начертание- фиксированы)
  double  hscale, // Ширина от эталонного шрифта(0.5 или 1.0)
  double  vscale, // Высота от эталонного шрифта(0.5 или 1.0)
  double  shear     = 0, // ???
  int     thickness = 1, // Толщина
  int     line_type = 8 // Сглаживание
);
```

Сериализация данных

- Часть требуется хранить в виде файлов сериализованные массивы данных. В OpenCV для этого предусмотрен механизм сопряжения с XML файлами.
- CvMat A = cvMat(5, 5, CV_32F, the_matrix_data); // Создаём матрицу
- cvSave(“my_matrix.xml”, &A); // Сохраняем
- CvMat* A1 = (CvMat*) cvLoad(“my_matrix.xml”); // Читаем сохранённую ранее матрицу из xml

Если необходимо хранить разношерстные данные

CvFileStorage* cvOpenFileStorage(const char* filename, // Название файла

CvMemStorage* memstorage, // Тип хранилища - для временных данных

//(NULL), для динамических структур (CvSeq)

int flags); // Флаг:

#define CV_STORAGE_READ 0 // Чтение

#define CV_STORAGE_WRITE 1 // Запись

#define CV_STORAGE_WRITE_TEXT CV_STORAGE_WRITE

#define CV_STORAGE_WRITE_BINARY CV_STORAGE_WRITE

#define CV_STORAGE_APPEND 2

Запись. Чтение аналогично.

```
CvFileStorage *fs = cvOpenFileStorage("cfg.xml", 0, CV_STORAGE_WRITE); // Создаём файловое

//хранилище для записи

cvWriteInt(fs, "width", 320); // Записываем два целочисленных значения

cvWriteInt(fs, "height", 240);

cvReleaseFileStorage(&fs); // Освобождаем память
```

Задача на следующие лекции

- Идентифицировать человека на изображении по цвету его кожи.