Нижегородский государственный университет им. Н.И. Лобачевского

Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP

Лабораторная работа
Сборка и установка библиотеки OpenCV. Использование библиотеки в среде Microsoft Visual Studio

При поддержке компании Intel

Кустикова В.Д.,
кафедра математического обеспечения ЭВМ

Содержание

- Цели и задачи работы
- Основные способы установки библиотеки OpenCV:
  - с использованием установочного файла
  - из исходных кодов с помощью утилиты CMake
- Подготовка среды Microsoft Visual Studio для разработки приложений с использованием OpenCV
- Разработка приложения для демонстрации базовых операций работы с изображениями на примере задачи выделения контуров объекта
- Разработка приложения для демонстрации базовых операций работы с видеоданными на примере задачи детектирования лиц

Библиотека OpenCV (1)

- Библиотека реализована на языках C/C++

- Обертки и интерфейсы для использования функционала библиотеки в других языках программирования:
  - Обертки для вызова функций из Python
  - .NET-обертки в составе библиотеки EmguCV [http://www.emgu.com/wiki/index.php/Main_Page]
  - Java-интерфейс

Библиотека OpenCV (2)

- Основные модули (всего ~20):
  - ***core*** – модуль, содержащий объявление всех структур данных.
  - ***imgproc*** – модуль обработки изображений.
  - ***highgui*** – модуль, позволяющий отображать рабочие изображения, проигрывать видео и создавать простые интерфейсы управления.
  - ***ml*** – модуль, содержащий реализацию некоторых алгоритмов машинного обучения.
  - ***objdetect*** – модуль детектирования объектов.
  - ***video*** – модуль анализа видео.

Цели работы

- Рассмотреть технические этапы подготовки инфраструктуры для выполнения последующих работ.
- Продемонстрировать использование базовых функций библиотеки OpenCV на простых практических примерах.

Задачи работы (1)

- Настройка среды Microsoft Visual Studio с целью использования библиотеки при разработке C/C++ приложений.

- Разработка приложения для определения контуров объектов, демонстрирующего применение некоторых базовых операций обработки изображений:
  - загрузка и сохранение изображений,
  - конвертирование цветового пространства,
  - бинаризация изображения,
  - выделение контуров объекта.

Задачи работы (2)

- Разработка приложения для демонстрации базовых операций обработки видео на примере задачи детектирования лиц с помощью классификатора Хаара:
  - загрузка видео из файла и перехват видеопотока с камеры,
  - извлечение кадров видеопотока,
  - получение и установка свойств видеофайла, сохранение видео.

Тестовая инфраструктура

Операционная система	Microsoft Windows 7
Среда разработки	Microsoft Visual Studio 2010
Библиотека TBB	Intel® Threading Building Blocks 3.0 for Windows, Update 3 (в составе Intel® Parallel Studio XE 2011 SP1)
Библиотеки OpenCV	Версия 2.4.2

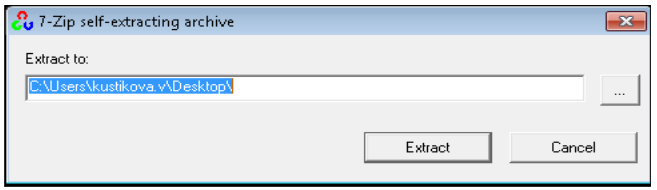
ОСНОВНЫЕ СПОСОБЫ УСТАНОВКИ БИБЛИОТЕКИ OPENCV

Способы установки библиотеки OpenCV

- Установка посредством запуска установочного файла.
- Сборка библиотеки из исходных кодов с использованием утилиты CMake.

Установка OpenCV с использованием установочного файла

- Загрузить инсталлятор с официальной страницы проекта.
- Запустить установочный файл:



- Выбрать путь для извлечения файлов библиотеки и нажать кнопку «**Extract**», чтобы инициировать процесс извлечения.
- Указанная директория будет содержать:
  - заголовочные файлы (вложенная директория **include**)
  - бинарные файлы библиотеки (**build**),
  - исходные коды библиотеки (**modules**),
  - набор примеров использования библиотечных функций (**samples**).

Утилита CMake

- Кроссплатформенная открытая система сборки.
- Применяется с целью управления процессом компиляции посредством использования конфигурационных файлов, не зависимых от компилятора.
- Позволяет генерировать make-файлы и рабочие пространства среды, компилятор которой будет использован для сборки проекта, например, решение и набор проектов для Microsoft Visual Studio.

Сборка библиотеки OpenCV из исходных кодов (1)

- Создать директорию для исходных кодов.
- Загрузить последнюю версию исходных кодов библиотеки с сервера проекта (Git).
- Опционально установить дополнительные программное обеспечение и библиотеки, которые могут потребоваться в процессе сборки OpenCV:
  - Python версии 2.6.x или 2.7.x и NumPy, если необходимо собрать обертки для Python;
  - Intel Threading Building Blocks (TBB);
  - Qt версии 4.6 обеспечивает функции для создания кроссплатформенного (Windows, Linux, Mac) графического интерфейса;

Сборка библиотеки OpenCV из исходных кодов (2)

- Intel Parallel Primitives (IPP) с версии 5.1 до 6.1 для ускорения некоторых вычислительных операций (конвертирование цвета, тренировка классификатора Хаара, функции вычисления быстрого преобразования Фурье (БПФ));
- LiveTeX (или MiKTeX под Windows, MacTeX под Mac) и Sphinx, чтобы собрать документацию OpenCV в форматах PDF и HTML;
- последняя версия NVIDIA CUDA Toolkit, если требуется собрать GPU модуль библиотеки;
- другое ПО и библиотеки, специфичные для Unix-систем, MacOS, Android и iOS.

Сборка библиотеки OpenCV из исходных кодов (3)

- Создать директорию в которую утилита CMake будет генерировать файлы сборки исходных кодов.
- Установить созданную директорию в качестве рабочей.
- Запустить исполняемый файл **cmake** утилиты CMake и создать файлы описания сборки. Директория утилиты CMake среди бинарных файлов содержит файл cmake-gui, который позволяет использовать CMake в графическом режиме.
- Установить настройки для создания сборки.
- Открыть решение **OpenCV.sln**.
- Скомпилировать решение в debug и release режимах. Проект **ALL_BUILD** установлен как рабочий и отвечает за компиляцию всех проектов в решении.

ПОДГОТОВКА СРЕДЫ MICROSOFT VISUAL STUDIO

Создание проекта (1)

- Запустите приложение Microsoft Visual Studio 2010.
- В меню **File** выполните команду **New→Project…**.
- В диалоговом окне **New Project** в типах проекта выберите **Win32**, в шаблонах **Win32 Console Application**, в поле **Name** введите название проекта (для каждого приложения будет использовано свое название), в поле **Solution Name** – название решения **LW_InslallOpenCV**, в поле **Location** укажите путь к папке с лабораторными работами. Нажмите OK.
- В диалоговом окне **Win32 Application Wizard** нажмите **Next** (или выберите **Application Settings** в дереве слева) и установите флаг **Empty Project**. Нажмите **Finish**.

Создание проекта (2)

- В окне **Solution Explorer** в папке **Source Files** выполните команду контекстного меню **Add→New Item…**. В дереве категорий слева выберите **Code**, в шаблонах справа – **C++ File (.cpp)**, в поле **Name** введите имя файла **main**. Нажмите **Add**. В результате выполненной последовательности действий в окне редактора кода Visual Studio будет открыт пустой файл **main.cpp**.
- Создайте заготовку функции **main()** с параметрами командной строки:

```
int main(int argc, char *argv[])
{
    // TODO: source code
    return 0;
}
```

Настройка свойств проекта (1)

- Установка пути до заголовочных файлов библиотеки OpenCV:
  - Выполните команду контекстного меню **Properties**.
  - Откройте вкладку **Configuration Properties→C/C++→General**.
  - Сверху в окне свойств в выпадающем списке **Configuration** выберите значение **All Configurations**, чтобы установить свойство для всех режимов компиляции (**Debug** и **Release**).
  - В поле **Additional Include Directories** укажите пути до заголовочных файлов библиотеки OpenCV.
  - Нажмите кнопку **Apply**, чтобы применить указанное свойство.

Настройка свойств проекта (2)

- Установка путей до подключаемых библиотек:
  - Откройте вкладку **Configuration Properties→Linker→General**.
  - Сверху в окне свойств в выпадающем списке **Configuration** выберите значение **Debug** / **Release** / **All Configurations**. В поле **Additional Library Directories** укажите путь до lib-файлов, скомпилированных в режиме **Debug** или **Release**.

Настройка свойств проекта (3)

- Указание списка подключаемых программных библиотек:
  - Откройте вкладку **Configuration Properties→Linker→Input**.
  - В выпадающем списке **Configuration** выберите значение **Debug** и установите в поле **Additional Dependencies** список lib-файлов (**opencv_core*d.lib**, **opencv_imgproc*d.lib**, **opencv_highgui*d.lib**, **opencv_legacy*d.lib**, **opencv_objdetect*d.lib**, **opencv_video*d.lib**, **opencv_ml*d.lib**).
  - В выпадающем списке **Configuration** выберите значение **Release** и установите в поле **Additional Dependencies** список lib-файлов (имена файлов отличаются отсутствием последнего символа ‘d’).

Подключение заголовочных файлов в исходном коде приложения

- Подключить заголовочный файл **opencv.hpp**, содержащий подключение всех установленных модулей библиотеки.
- Подключить пространство имен **cv**, в которое заключены все функции библиотеки.

```
#include <opencv2\opencv.hpp>
using namespace cv;
```

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ КОНТУРОВ ОБЪЕКТОВ

Задача определения контуров/краев объектов

- Точки, в которых происходит резкий перепад яркости изображения, называются *краями* или *краевыми точками*.
- Резкое изменение яркости изображения представляет интерес по нескольким причинам:
  - Возникают на границах объектов, например, при изображении светлого объекта на темном фоне или наоборот.
  - Следствие изменения отражательной способности на характерных структурах, таких, как разметка пешеходного перехода или пятна на шкуре леопарда.
  - Резкие изменения ориентации поверхности.

Постановка задачи

- Разработать приложение, которое обеспечивает поиск контуров/краев на изображении.
- Обеспечить отображение исходного изображения и результирующего (отрисовка выделенных контуров) изображений.

- ***Примечание:*** предполагается использовать функции библиотеки OpenCV.

Базовые операции работы с изображениями (1)

- Создание изображения:
  - Конструкторы класса **Mat**:

```
// _rows – количество строк
// _cols – количество столбцов
// _type – тип матрицы (CV_8UC1, CV_64FC3 и другие)
Mat(int _rows, int _cols, int _type);
Mat(Size _size, int _type);
```

  - Метод **create**:

```
void create(int _rows, int _cols, int _type);
void create(Size _size, int _type);
```

- Удаление (освобождение памяти) изображения:

```
void release();
```

Базовые операции работы с изображениями (2)

- Загрузка изображения:

```
Mat imread(const string& filename, int flags=1)
```

- **flags>0** означает, что изображение загружается как цветное трехканальное (BGR);
- **flags<0** – автоматически определяется количество каналов при загрузке;
- **flags=0** используется для загрузки изображения в оттенках серого (позволяет сразу получить из цветного изображение в оттенках серого без дополнительной конвертации).

Базовые операции работы с изображениями (3)

- Сохранение изображения:

```
bool imwrite(const string& filename,
             const Mat& img,
             const vector<int>& params=vector<int>())
```

- Вектор целочисленных параметров **params** определяет параметры сохранения в файл. На данный момент доступны:
  - **CV_IMWRITE_JPEG_QUALITY**
  - **CV_IMWRITE_PNG_COMPRESSION**
  - **CV_IMWRITE_PXM_BINARY**

Базовые операции работы с изображениями (4)

- Отображение изображения:
  - Создать окно для отображения

```
void namedWindow(const string& winname, int flags)
```

    - **CV_WINDOW_FREERATIO**, **CV_WINDOW_KEEPRATIO**
    - **CV_GUI_NORMAL**, **CV_GUI_EXPANDED**

  - Отобразить изображение в окне

```
void imshow(const string& winname, const Mat& image)
```

  - Дождаться нажатия какой-либо клавиши, чтобы закрыть окно (по умолчанию в течение ∞)

```
int waitKey(int delay=0)
```

Базовые операции работы с изображениями (5)

- Копирование изображения:

```
Mat  clone() const;

void copyTo( Mat& m ) const;
void copyTo( Mat& m, const Mat& mask ) const;
```

- Замечание: **copyTo** выделяет память для хранения матрицы того же размера, что m.

Базовые операции работы с изображениями (6)

- Конвертирование изображения в другое цветовое пространство:

```
void cvtColor(const Mat& src, Mat& dst,
              int code, int dstCn=0)
```

- Параметр **dstCn** – количество каналов в результирующем изображении
- Параметр **code**:
  - **CV_RGB2GRAY**, **CV_GRAY2RGB** – конвертирование из RGB-пространства в оттенки серого (взвешенная линейная свертка интенсивностей по всем трем каналам) и наоборот (дублирование интенсивности);
  - …

Базовые операции работы с изображениями (7)

- Бинаризация и отсечение изображения:

```
double threshold(const Mat& src, Mat& dst, double thresh,
                 double maxVal, int thresholdType)
```

- Параметр **thresholdType**:
  - **THRESH_BINARY** (dst(x,y)=maxVal, если src(x,y)>thresh, dst(x,y)=0 в противном случае)
  - **THRESH_TRUNC** (dst(x,y)= thresh, если src(x,y)>thresh, dst(x,y)= src(x,y), в противном случае)
  - **THRESH_TOZERO** (dst(x,y)= src(x,y), если src(x,y)>thresh, dst(x,y)=0 в противном случае)
  - **THRESH_BINARY_INV**, **THRESH_TOZERO_INV**

Базовые операции работы с изображениями (8)

- Поиск контуров на бинарном изображении:

```
void findContours(const Mat& image,
                  vector<vector<Point> >& contours,
                  vector<Vec4i>& hierarchy, int mode,
                  int method, Point offset=Point())
void findContours(const Mat& image,
                  vector<vector<Point> >& contours,
                  int mode, int method,
                  Point offset=Point())
```

- **mode** – идентификатор способа восстановления контура.
- **method** – идентификатор метода аппроксимации контура.

Базовые операции работы с изображениями (9)

- Отображение контуров на изображении:

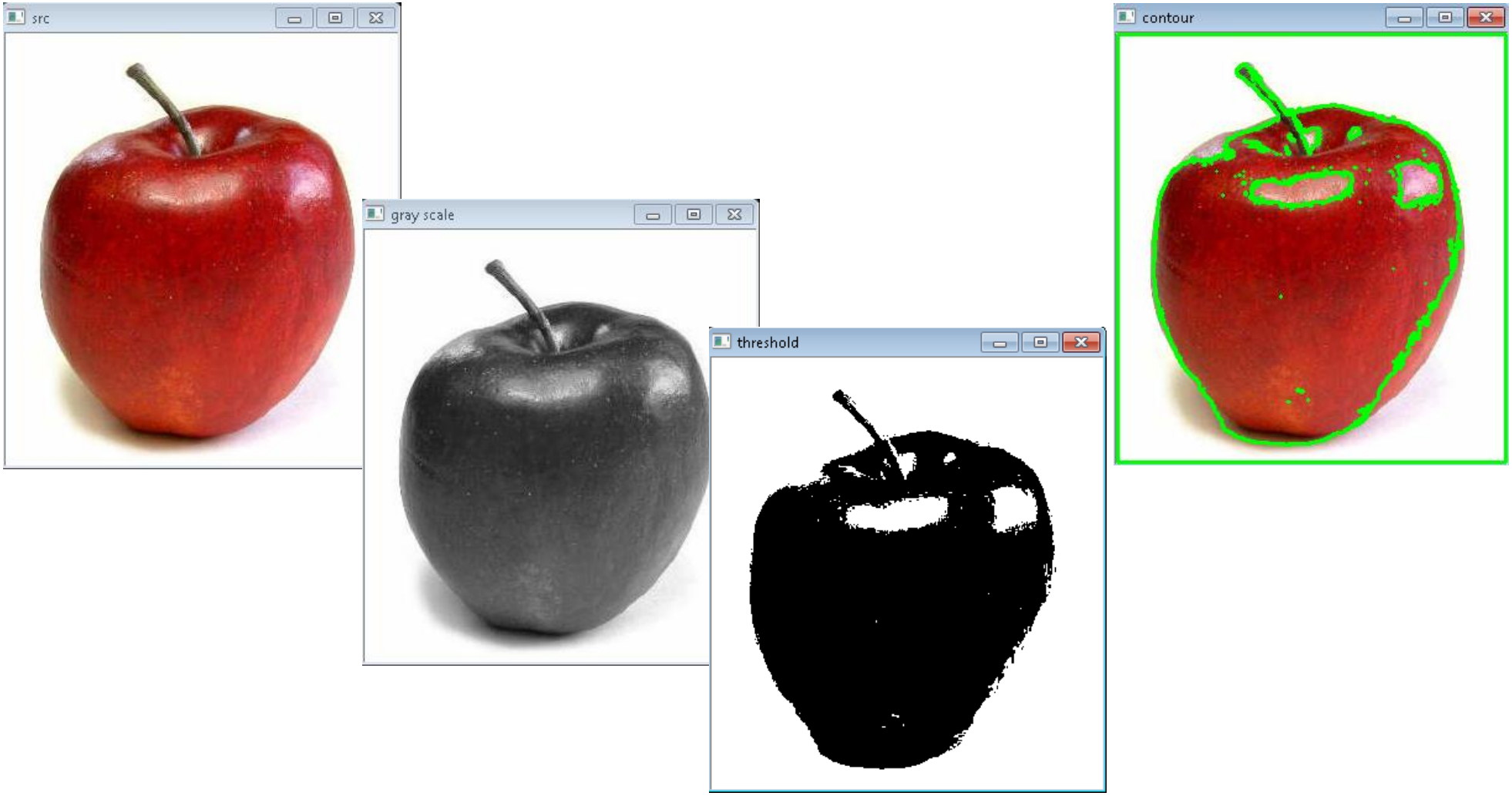
```
void drawContours(Mat& image,
                  const vector<vector<Point> >& contours,
                  int contourIdx, const Scalar& color,
                  int thickness=1, int lineType=8,
                  const vector<Vec4i>&
                          hierarchy=vector<Vec4i>(),
                  int maxLevel=INT_MAX,
                  Point offset=Point())
```

- Отображает контур **contours[contourIdx]** посредством отрисовки линии, имеющей цвет **color** и толщину **thickness**, **lineType** (**8**, **4**, **CV_AA**) – тип связности линии.

Разработка приложения для определения контуров объектов

1. Загрузить исходное изображение (название файла получить через аргументы командной строки).
2. Создать копию изображения, чтобы отобразить изображение с отрисованными контурами и исходное изображение.
3. Конвертировать копию изображения в оттенки серого.
4. Выполнить бинаризацию полученного изображения.
5. Определить контуры на бинаризованном изображении.
6. Отобразить исходное изображение и его копию с отрисованными контурами объектов.
7. Освободить занятые ресурсы (память из-под изображений).

Запуск приложения и анализ результатов



РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ПОИСКА ЛИЦ НА ПОТОКЕ ВИДЕОДАННЫХ

Задача детектирования лиц

- ***Детектирование лиц*** – получение информации о присутствии лиц и их местоположении на изображении.
- Решение задачи детектирования лиц представляет собой совокупность прямоугольников либо окружностей, окаймляющих лица.
- ***Причины сложности задачи:***
  - Значительное расхождение внешнего вида и формы лица.
  - Различие изображения лица при изменении ориентации относительно камеры.
  - Вариативность выражения лица.
  - Возможность перекрытия, различная освещенность.

Постановка задачи

- Разработать приложение, которое позволяет детектировать лица на видео.

- ***Примечание:*** предполагается использовать реализацию метода Виолы-Джонса для поиска лиц на отдельных кадрах видео (изображениях).

Базовые операции работы с видео (1)

- Загрузка видео из файла, перехват видеопотока с камеры:
  - Создать объект типа **VideoCapture**, используя конструктор по умолчанию:

```
VideoCapture();
```

  - Открыть видеопоток:

```
virtual bool open(const string& filename);
virtual bool open(int device);
```

  - Вызвать конструктор с параметрами:

```
VideoCapture(const string& filename);
VideoCapture(int device);
```

Базовые операции работы с видео (2)

- Проверка корректности открытия видеопотока:

```
virtual bool isOpened() const;
```

- Освобождение занятых ресурсов:

```
virtual void release();
```

Базовые операции работы с видео (2)

- Извлечение кадра видеопотока:
  - 1-ый способ:

```
virtual bool grab();
virtual bool retrieve(Mat& image, int channel=0);
```

  - 2-ой способ:

```
virtual VideoCapture& operator >> (Mat& image);
```

Базовые операции работы с видео (3)

- Получение и установка свойств видеофайла:

```
virtual bool set(int propId, double value);
virtual double get(int propId);
```

- Некоторые возможные значения **propId**:
  - **CV_CAP_PROP_POS_MSEC** – текущая позиция в треке в миллисекундах;
  - **CV_CAP_PROP_FRAME_WIDTH (_HEIGHT)** – ширина (высота) кадра в видеопотоке;
  - **CV_CAP_PROP_FPS** – количество кадров, отображаемых в секунду;
  - **CV_CAP_PROP_FRAME_COUNT** – количество кадров.

Базовые операции работы с видео (4)

- Сохранение кадров видеопотока в видеофайл:
  - 1-ый способ:

```
VideoWriter();
virtual bool open(const string& filename, int fourcc,
                  double fps, Size frameSize,
                  bool isColor=true);
```

  - 2-ой способ:

```
VideoWriter(const string& filename, int fourcc,
            double fps, Size frameSize,
            bool isColor=true);
```

Базовые операции работы с видео (5)

- Проверка корректности открытия потока на запись:

```
virtual bool isOpened() const;
```

- Запись изображения в видеофайл:

```
virtual VideoWriter& operator << (const Mat& image);
```

Другие необходимые функции (1)

- Детектирование лиц с использованием реализации классификатора Хаара библиотеки OpenCV:
  - Создание классификатора загрузка модели

```
CascadeClassifier();
bool load(const string& filename);
CascadeClassifier(const string& filename);
```

  - Поиск лиц разного масштаба

```
void detectMultiScale(const Mat& image,
                      vector<Rect>& objects,
                      double scaleFactor=1.1,
                      int minNeighbors=3,
                      int flags=0, Size minSize=Size());
```

Другие необходимые функции (2)

- **image** – это изображение в оттенках серого (**CV_8U**).
- **objects** – вектор результирующих окаймляющих прямоугольников.
- **scaleFactor** – коэффициент масштабирования, который определяет, во сколько раз будет уменьшено изображение на каждом масштабе.
- **minNeighbors** – параметр алгоритма, интенсивность обнаружения лица в заданной позиции (отсекаются одиночные срабатывания классификатора, объединяются близкие срабатывания).
- **flags** не используется и присутствует для совместимости со старым интерфейсом.
- **size** – минимальный размер возможного объекта.

Другие необходимые функции (3)

- Отображение окаймляющих прямоугольников для изображении:

```
void rectangle(CvArr* img, CvPoint pt1, CvPoint pt2,
               CvScalar color, int thickness=1,
               int lineType=8, int shift=0)
```

- **pt1** – координаты левого верхнего угла прямоугольника.
- **pt2** – координаты правого нижнего угла прямоугольника.

- Замечание: если **thickness<0** или **=CV_FILLED**, то отображается закрашенный прямоугольник.

Разработка приложения для детектирования лиц на видео (1)

1. Разобрать аргументы командной строки, чтобы извлечь полное название файла с моделью лица и при необходимости имя видеофайла.
2. Создать классификатор и загрузить модель.
3. Загрузить видео из файла или открыть видеопоток с веб-камеры, если название файла отсутствует в параметрах командной строки.
4. Создать окно для отображения видео.

Разработка приложения для детектирования лиц на видео (2)

5. Разработать цикл обработки последовательности кадров видепотока:
   - получить очередной кадр видео;
   - конвертировать текущий кадр в оттенки серого;
   - продетектировать лица;
   - отрисовать полученные окаймляющие прямоугольники на изображении;
   - отобразить изображение;
   - *Замечание: тело цикла будет выполняться до тех пор, пока не будет достигнут конец трека, либо не будет нажата клавиша **Esc**.*
6. Освободить ресурсы и закрыть видеопоток.

Запуск приложения и анализ результатов

- Демонстрация с перехватом видеопотока с камеры…

Задания для самостоятельной работы (1)

- Модифицируйте приложение для поиска контуров объекта так, чтобы результирующее изображение сохранялось в файл.
- Модифицируйте приложение для поиска контуров объекта так, чтобы отображалось изображение, конвертированное в оттенки серого, и бинаризованное изображение, т.е. то, что показано на рисунке.
- Как будет изменяться результат работы приложения для определения контуров, если изменить способ восстановления контуров? Проведите эксперименты со всеми возможными значениями параметра **mode** в функции **findContours**.

Задания для самостоятельной работы (2)

- Модифицируйте приложения, которое выполняет детектирование лиц на видеопотоке, так, чтобы результат детектирования сохранялся в видеофайл. Примечание: используйте возможности класса **VideoWriter**.
- Снимите видео, где была бы группа людей с разным ракурсом лица. Проанализируйте результаты детектирования на данном видео. Сравните результаты детектирования лиц с использованием других моделей лиц. Модифицируйте приложение так, чтобы добиться максимального качества детектирования. Примечание: комбинируйте результаты детектирования с помощью моделей фаса и профиля.

Задания для самостоятельной работы (3)

- Добавьте в приложение для детектирования лиц возможность детектирования глаз и других частей лица с использованием классификатора Хаара. Примечание: используйте натренированные модели, входящие в состав библиотеки OpenCV.

Авторский коллектив

- Кустикова Валентина Дмитриевна,
  ассистент кафедры
  Математического обеспечения ЭВМ факультета ВМК ННГУ
  valentina.kustikova@gmail.com