

第1章 opencv 与 dlib 介绍.....	3
1.1 opencv 介绍.....	3
1.2 dlib 介绍.....	3
第2章 opencv ubuntu 环境搭建.....	3
2.1 opencv 安装准备工作.....	4
2.1.1 安装环境.....	4
2.1.2 源码获取.....	4
2.2 opencv 具体安装步骤.....	4
2.2.1 安装 opencv 所需依赖库.....	4
2.2.2 解压源码.....	4
2.2.3 配置 opencv.....	4
2.2.4 编译安装 opencv.....	6
2.3 opencv 环境配置.....	6
2.3.1 添加 opencv 库.....	6
2.3.2 使 opencv 配置文件生效.....	6
2.3.3 配置 bash 环境变量.....	7
2.3.4 验证 opencv 环境配置是否成功.....	7
2.4 opencv 测试.....	7
第3章 opencv 基本理论知识.....	8
3.1 Mat 像素统计技术.....	8
3.1.1 opencv 常用类和方法.....	8
3.1.2 opencv 代码编译.....	10
3.1.3 锐化操作.....	11
3.1.4 图像重叠操作.....	13
3.2 计算N维数据关系.....	15
3.2.1 均值.....	15
3.2.2 方差.....	15
3.2.3 标准差.....	16
3.2.4 协方差.....	16
3.2.5 协方差矩阵.....	17
3.3 特征值和特征向量.....	19
3.3.1 特征值与特征向量的定义.....	19
3.3.2 opencv 接口计算特征值与特征向量.....	21
第4章 PCA 原理和应用.....	22
4.1 PCA 原理介绍.....	22
4.1.1 PCA 概念.....	22
4.1.2 opencv 中 PCA 相关接口.....	23

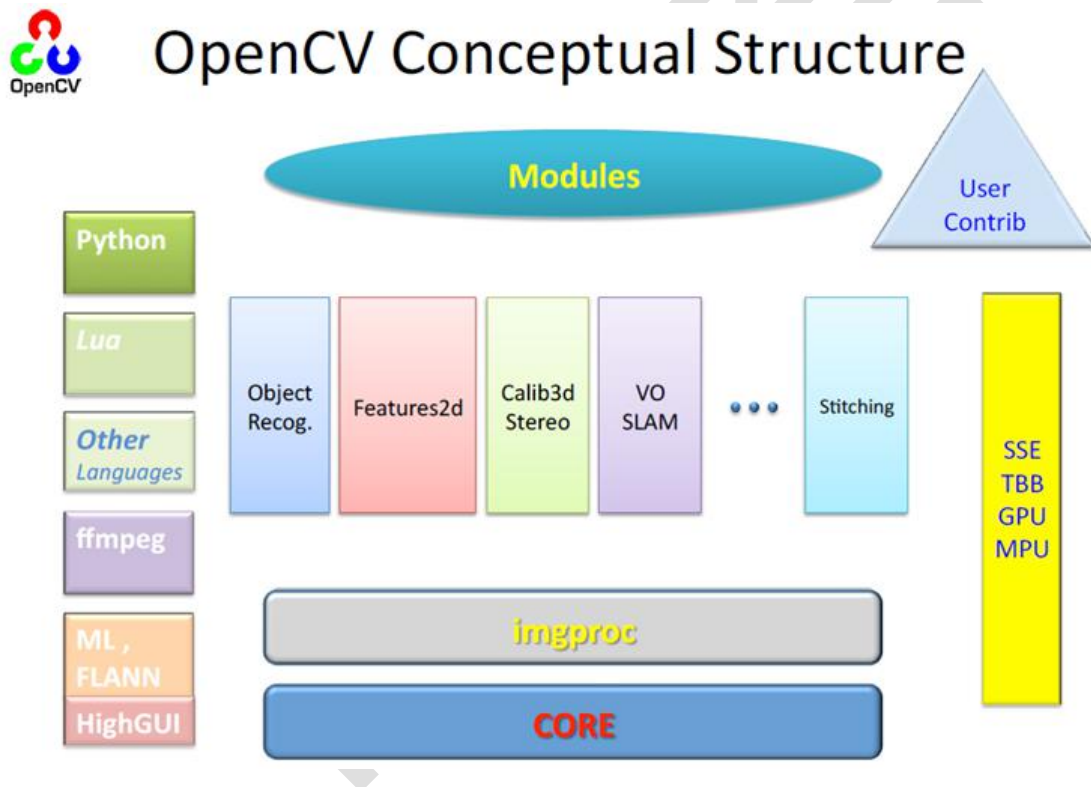
第 5 章 opencv 实时人脸识别应用开发.....	26
5.1 人脸检测.....	26
5.1.1 equalizeHist().....	26
5.1.2 级联分类器 CascadeClassifier.....	26
5.1.3 rectangle().....	27
5.1.3 VideoCapture 类.....	28
5.2 人脸识别.....	28
5.2.1 EigenFace 介绍.....	28
5.2.2 人脸识别算法对比.....	30
5.2.3 opencv 人脸识别案例分析.....	30
5.3 opencv 非特定目标检测之几何图形识别.....	32
5.3.1 准备训练样本素材.....	32
5.3.2 将图片进行灰度处理.....	32
5.3.3 生成 pos.txt 积极图片描述文件.....	33
5.3.4 生成 neg.txt 消极图片描述文件.....	33
5.3.5 生成 vec 文件.....	33
5.3.6 开始样本训练.....	33
5.3.6 对几何图进行识别.....	34
第 6 章 dlib ubuntu 环境搭建.....	34
6.1 dlib 安装准备工作.....	34
6.1.1 安装环境.....	34
6.1.2 源码获取.....	34
6.2 dlib 具体安装步骤.....	34
6.2.1 解压源码.....	35
6.2.2 配置 dlib.....	35
6.2.3 编译安装 dlib.....	35
6.3 dlib 环境配置.....	35
6.3.1 添加 dlib 库.....	35
6.3.2 使 dlib 配置文件生效.....	35
6.3.3 配置 bash 环境变量.....	35
6.3.4 验证 dlib 环境配置是否成功.....	36
6.4 dlib 测试.....	36
第 7 章 dlib 实时人脸识别应用开发.....	36
7.1 人脸检测.....	36
7.2 提取人脸特征点.....	37
7.3 人脸识别.....	38
7.4 dlib 对非特定目标识别之手势识别.....	38
7.4.1 编译训练工具.....	38
7.4.1 训练样本采集.....	38

第 1 章 opencv 与 dlib 介绍

1.1 opencv 介绍

OpenCV 是一个基于 BSD 许可（开源）发行的跨平台计算机视觉库，可以运行在 Linux、Windows、Android 和 Mac OS 操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

OpenCV 目前主要用 C++ 语言开发，它的主要接口也是 C++ 语言封装，但是依然保留了大量的 C 语言接口。该库也有大量的 Python、Java and MATLAB/OCTAVE（版本 2.5）的接口。这些语言的 API 接口函数可以通过在线文档获得。如今也提供对于 C#、Ch、Ruby, GO 的支持



1.2 dlib 介绍

Dlib 是一个现代化的 C++ 工具箱，其中包含用于在 C++ 中创建复杂软件以解决实际问题的机器学习算法和工具。它广泛应用于工业界和学术界，包括机器人，嵌入式设备，移动电话和大型高性能计算环境。Dlib 的开源许可证允许您在任何应用程序中免费使用它。Dlib 有很长的时间，包含很多模块，近几年主要关注在机器学习、深度学习、图像处理等模块的开发。

第 2 章 opencv ubuntu 环境搭建

做真实的自己，用良心做教育

2.1 opencv 安装准备工作

2.1.1 安装环境

Ubuntu16.04 LTS
opencv3.2.0
contrib3.2.0

2.1.2 源码获取

软件百度网盘链接:

链接: https://pan.baidu.com/s/1AQH_yDL3nInC5by1SNUjdA

提取码: 72pk

软件官网下载地址:

<https://github.com/opencv/opencv/archive/3.2.0.tar.gz>

https://github.com/opencv/opencv_contrib/archive/3.2.0.tar.gz

2.2 opencv 具体安装步骤

2.2.1 安装 opencv 所需依赖库

```
sudo apt-get install cmake libgtk2.0-dev libavcodec-dev libavformat-dev  
libjpeg-dev libpng-dev libtiff-dev libtiff4-dev libswscale-dev  
libjasper-dev libcurl4-openssl-dev libtbb2 libdc1394-22-dev
```

2.2.2 解压源码

将两个下载的源码文件拷贝到 Ubuntu 中并解压

```
unzip opencv-3.2.0.zip
```

```
unzip opencv_contrib-3.2.0.zip
```

如果解压失败, 请安装 jar 解压工具:

```
sudo apt-get install default-jdk
```

```
jar xvf opencv-3.2.0.zip
```

2.2.3 配置 opencv

创建编译安装目录:

```
mkdir opencv-3.2.0/mybuild
```

```
mkdir opencv-3.2.0_install
```

通过 cmake 工具生成 Makefile:

千锋教育

```
cd opencv-3.2.0/mybuild
cmake -D CMAKE_BUILD_TYPE=Release -D OPENCV_GENERATE_PKGCONFIG=ON -D
CMAKE_INSTALL_PREFIX=/home/edu/ai/opencv-3.2.0_install -D
OPENCV_EXTRA_MODULES_PATH=/home/edu/ai/opencv_contrib-3.2.0/modules
..
```

cmake 配置途中需要下载

ippicv_2019_lnx_intel64_general_20180723.tgz、
ippicv_linux_20151201.tgz、protobuf-cpp-3.1.0.tar.gz 等文件，还有一堆.i
文件，网速不好的情况可能会比较慢，如果实在太慢可以中断 cmake，直接拷贝
下载好的文件到相应目录，然后再继续执行上面的 cmake 命令：

- 一、ippicv_linux_20151201.tgz 位于
opencv-3.2.0/3rdparty/ippicv/downloads/linux-...
- 二、protobuf-cpp-3.1.0.tar.gz 位于
opencv_contrib-3.2.0/modules/dnn/.download/bd5e3.../v3.1.0/
- 三、那一堆.i 文件位于 opencv_contrib-3.2.0/modules/xfeatures2d/src/

2.2.4 编译安装 opencv

进入 opencv-3.2.0/mybuild 目录完成编译安装(该过程根据不同配置的计算可能
需要 20 分钟左右)：

```
make -j4
make install
```

2.3 opencv 环境配置

2.3.1 添加 opencv 库

打开或创建 opencv.conf 文件，并添加 opencv 安装路径

```
sudo gedit /etc/ld.so.conf.d/opencv.conf
/home/edu/ai/opencv-3.2.0_install/lib <添加内容>
```

2.3.2 使 opencv 配置文件生效

```
sudo ldconfig
```

2.3.3 配置 bash 环境变量

```
sudo gedit ~/.bashrc <在文件末尾添加如下内容>
```

```
export PKG_CONFIG_PATH=/home/edu/ai/opencv-3.2.0_install/lib/pkgconfig
```

第四步：生效配置文件

```
source ~/.bashrc <使环境变量立即生效>
```

2.3.4 验证 opencv 环境配置是否成功

```
pkg-config --cflags --libs opencv
```

2.4 opencv 测试

找到 opencv-3.2.0/samples/cpp/example_cmake 目录, 该目录下面有一个测试程序, 配置编译之前需要修改 CMakeLists.txt, 在文件中添加下面一行:

```
set(OpenCV_DIR /home/edu/ai/opencv-3.2.0/mybuild)
```

然后执行 cmake . 用于生成 makefile

```
cmake .
```

```
stu@qfedu:~/system/opencv-3.2.0/samples/cpp/example_cmake$ ls
CMakeLists.txt  example.cpp
stu@qfedu:~/system/opencv-3.2.0/samples/cpp/example_cmake$ cmake .
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 5.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenCV: /usr/local (found version "3.2.0")
-- OpenCV library status:
--   version: 3.2.0
--   libraries: opencv_calib3d;opencv_core;opencv_features2d;opencv_flann;opencv_highgui;opencv_imgcodecs;opencv_imgproc;opencv_ml;opencv_objdetect;opencv_photo;opencv_shape;opencv_stitching;opencv_superres;opencv_video;opencv_videoio;opencv_videostab;opencv_aruco;opencv_bgsegm;opencv_bioinspired;opencv_ccalib;opencv_datasets;opencv_dnn;opencv_dpm;opencv_face;opencv_freetype;opencv_fuzzy;opencv_line_descriptor;opencv_optflow;opencv_phase_unwrapping;opencv_plot;opencv_reg;opencv_rgbd;opencv_saliency;opencv_stereo;opencv_structured_light;opencv_surface_matching;opencv_text;opencv_tracking;opencv_xfeatures2d;opencv_ximgproc;opencv_xobjdetect;opencv_xphoto
--   include path: /usr/local/include;/usr/local/include/opencv
-- Configuring done
-- Generating done
-- Build files have been written to: /home/stu/system/opencv-3.2.0/samples/cpp/example_cmake
```

会生成 Makefile

```
stu@qfedu:~/system/opencv-3.2.0/samples/cpp/example_cmake$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  CMakeLists.txt  example.cpp  Makefile
```

执行 make

```
make
```



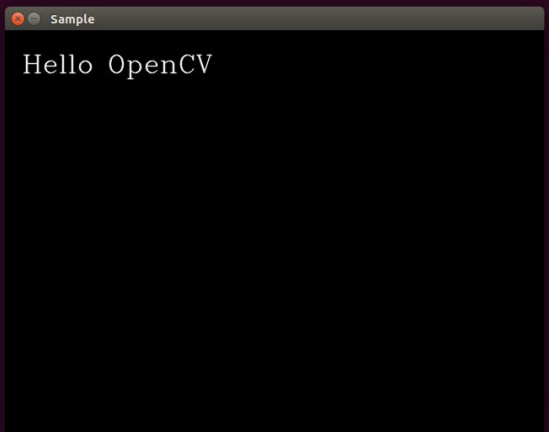
```
stu@qfedu:~/system/opencv-3.2.0/samples/cpp/example_cmake$ make
Scanning dependencies of target opencv_example
[ 50%] Building CXX object CMakeFiles/opencv_example.dir/example.cpp.o
[100%] Linking CXX executable opencv_example
[100%] Built target opencv_example
```

执行生成的可执行文件

```
./opencv_example
```

执行结果

```
lzx@qfedu:~/system/opencv-3.4.7/samples/cpp/example_cmake$ ./opencv_example
Built with OpenCV 3.4.7
VIDEOIO ERROR: V4L: can't open camera by index 0
No capture
□
```



第 3 章 opencv 基本理论知识

3.1 Mat 像素统计技术

3.1.1 opencv 常用类和方法

3.1.1.1 Mat 类

Mat 是一个基本图像容器，也是一个类，数据由两个部分组成：

矩阵头（包含矩阵尺寸，存储方法，存储地址等信息）和一个指向存储所有像素值的矩阵（根据所选存储方法的不同矩阵可以是不同的维数）的指针。

3.1.1.2 imread()

```
Mat imread( const String& filename, int flags = IMREAD_COLOR );
```

功能：读取图片文件中的数据

参数：

filename：图片路径

flags：指定读取图片的颜色类型，注意：opencv 版本不同，宏可能不一样

opencv3.2.0 中

IMREAD_GRAYSCALE 值为 0，表示显示灰度图

做真实的自己，用良心做教育

IMREAD_COLOR 值为 1，表示显示原图

返回值: Mat 类对象

3.1.1.3 imshow()

```
void imshow(const String& winname, InputArray mat);
```

功能: 显示照片

参数:

winname: 显示照片窗口的名称

mat: imread 的返回值

返回值: 无

3.1.1.4 waitKey()

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

功能: 在一个给定的时间内(单位 ms)等待用户按键触发, 如果用户没有按下键, 则一直阻塞

参数:

delay: 用于设置在显示完一帧, 图像后程序等待“delay”ms 再显示下一帧视频, 如果 waitKey(0) 则只会显示第一帧视频

返回值: 按键的 ASCII 码值

3.1.1.5 putText()

```
void putText( Mat& img, const string& text,
              Point org,  int fontFace,
              double fontScale,  Scalar color,
              int thickness=1, int lineType=8 );
```

功能: 在图像上添加文字

参数:

img: 待添加文字的图像

text: 字符串, 不支持中文

org: 待写入的首字符左下角坐标

fontFace: 字体类型, FONT_HERSHEY_SIMPLEX , FONT_HERSHEY_PLAIN , FONT_HERSHEY_DUPLEX 等

fontScale: 字体大小

color: 字体颜色, 颜色用 Scalar (BGR) 表示

Thickness: 字体粗细

lineType: 线型, 默认值是 8

返回值: 无

3.1.2 opencv 代码编译

案例:

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace std;
using namespace cv;

int main(int argc, char const *argv[])
{
    //实例化对象保存图片的信息
    Mat image = imread("./dog.jpg");

    //判断图片是否为空
    if(image.empty()){
        cout << "Do not load image..." << endl;
        return -1;
    }
    //显示图片
    imshow("This is a image", image);

    //等待按键被按下, 如果不按下, 则阻塞
    //waitKey(0);
    #if 1
        //键盘任意键按下或者 5 秒后代码继续运行
        waitKey(5000);
        image = imread("./dog.jpg", 0);
        imshow("This is a image", image);
        waitKey(0);
    #endif
    return 0;
}
```

3.1.2.1 直接命令行编译

```
g++ -o opencv_t t.cpp `pkg-config --cflags --libs opencv`
```

3.1.2.2 通过 Makefile 编译

从 opencv-3.2.0/samples/cpp/example_cmake/拷贝一个 CMakeLists.txt, 将文中三处 example 改成你的 cpp 文件名, 并添加下面一行:

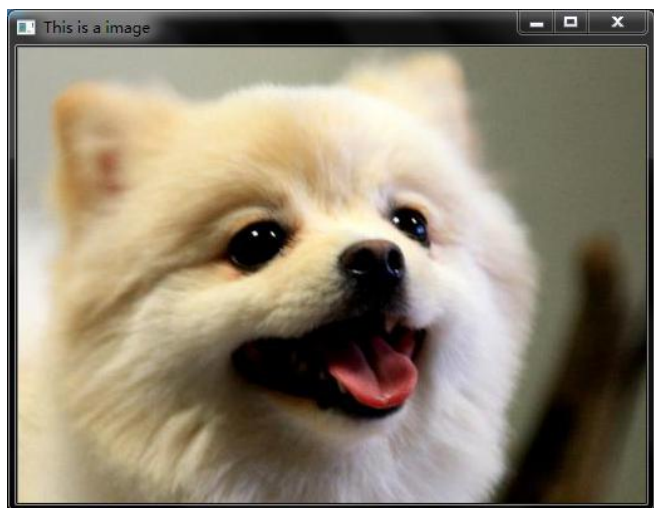
```
set(OpenCV_DIR /home/edu/ai/opencv-3.2.0/mybuild)
```

然后执行配置编译命令:

```
cmake .
```

```
make
```

运行结果:

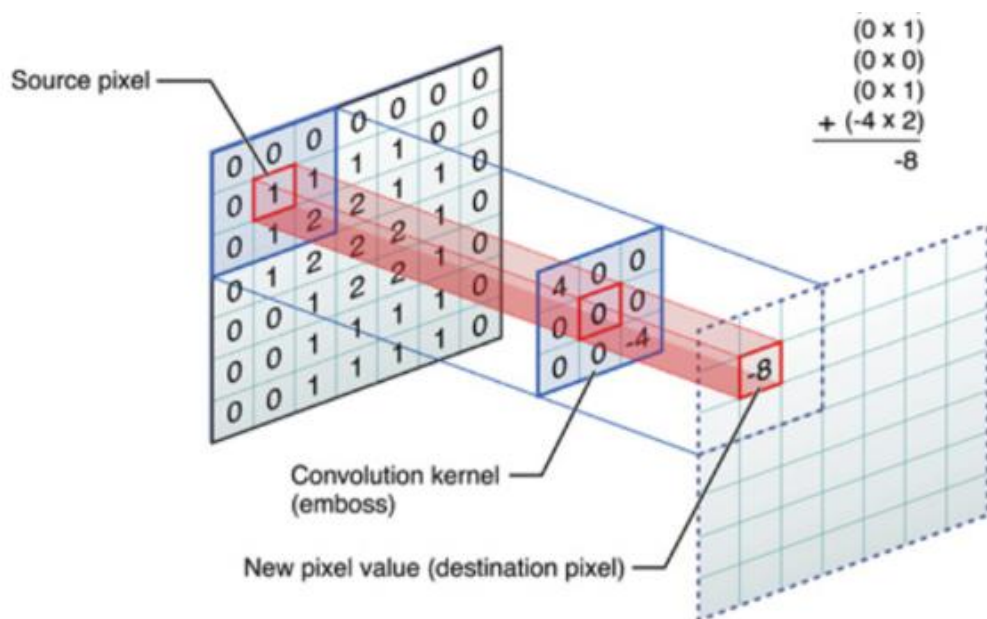


3.1.3 锐化操作

在对像素邻域进行计算时，通常用一个核心矩阵来表示。这个核心矩阵展现了如何将与计算相关的像素组合起来，才能得到预期结果。针对本节使用的锐化滤波器，核心矩阵可以是这样的：

0	-1	0
-1	5	-1
0	-1	0

除非另有说明，当前像素用核心矩阵中心单元格表示，又叫卷积核或相关核，核心矩阵中的每个单元格表示相关像素的乘法系数，结果像素通过核心矩阵得到的结果，即是这些乘积的累加。核心矩阵的大小就是邻域的大小（这里是 3×3 ），计算过程如下：



滤波处理接口 void filter2D()

```
void filter2D( InputArray src, OutputArray dst, int ddepth, InputArray kernel, Point anchor
= Point(-1,-1), double delta = 0, int borderType = BORDER_DEFAULT );
```

功能：通过给定卷积核对图像进行滤波处理

参数：

src：源图像

dst：与 src 相同大小、相同通道数的输出图像

ddepth：目标图像期望的深度（一般保持和源图像一致）

kernel：构造核心矩阵内核（协同图像传入 filter2D）

delta：选填过滤后的像素（一般取缺省值 0）

borderType：边框（一般取缺省值 BORDER_DEFAULT）

案例：

```
#include<iostream>
#include<opencv2/opencv.hpp>
using namespace cv;
using namespace std;

int main(int argc, char const *argv[])
{
    if(argc < 2){
        cout<<"./opencv t *.jpg"<<endl;
        return -1;
    }
    Mat src,dst;
    //读取照片
    src = imread(argv[1]);
    if(!src.data){
        cout<<"could not load image..."<<endl;
    }
}
```

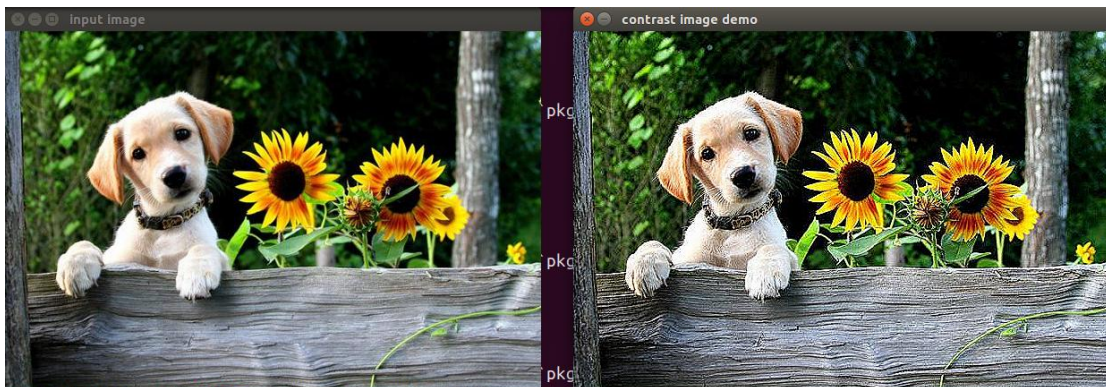
做真实的自己，用良心做教育

```

        return -1;
    }
    imshow("input image", src);
    //锐化方式
    Mat kernel = (Mat_<char>(3,3)<< 0,-1,0,-1,5,-1,0,-1,0);
    filter2D(src, dst, src.depth(), kernel3);
    imshow("contrast image demo", dst);
    waitKey(0);
    return 0;
}

```

执行结果



还可以做如下尝试:

```

Mat kernel = Mat::ones(5,5,CV_32F)/(float)(25); //模糊处理
Mat kernel = (Mat_<int>(2,2)<< 1,0,0,-1); //边缘处理

```

3.1.4 图像重叠操作

像素混合接口 void addWeighted()

```

void addWeighted(InputArray src1, double alpha, InputArray src2, double beta, double gamma,
OutputArray dst, int dtype = -1);

```

功能: 图像合并处理, 合并图片

参数:

src1: 源图像 1

alpha: 图像的权重 (0~1 之间)

src2: 源图像 2

beta: 第二张源图像的权重 (一般为 1.0-alpha)

gamma: 如果两张相加后亮度不理想 可以使用 gamma 使其亮度效果更好, 设置亮度

dst: 两张图像输出的目标图像

dtype: 使用缺省值

案例:

```

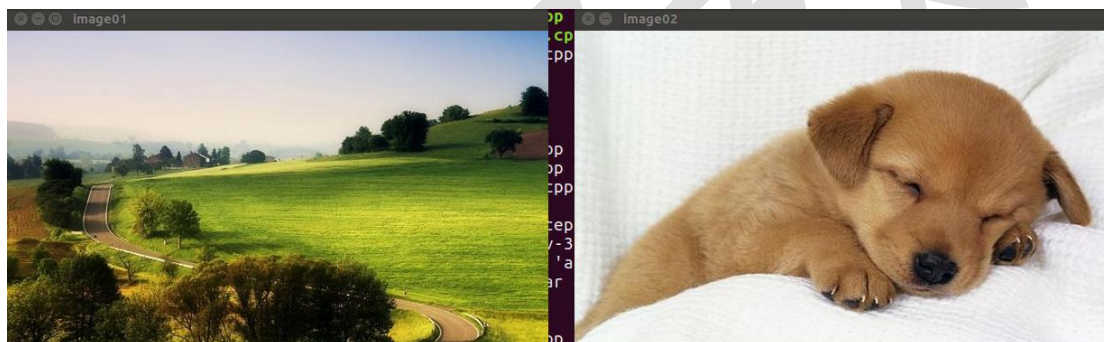
#include<iostream>
#include<opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int main(int argc, char const *argv[])
{

```

做真实的自己, 用良心做教育

```
if(argc < 3){
    cout<<"./opencv t *.jpg *.jpg"<<endl;
    return -1;
}
Mat src1,src2, dst;
//读取第一张照片
src1 = imread(argv[1]);
if(!src1.data){
    cout<<"cound not load image..."<<endl;
    return -1;
}
//读取第二张照片
src2 = imread(argv[2]);
if(!src2.data){
    cout<<"cound not load image..."<<endl;
    return -1;
}
//必须保证图片大小一致
if(src1.rows != src2.rows || src1.cols != src2.cols || src1.type() != src2.type()){
    resize(src1, src1,Size(300,300));
    resize(src2, src2,Size(300,300));
}
double alpha = 0.5;
addWeighted(src1, alpha, src2, 0.6, 0.0, dst);
imshow("src1", src1);
imshow("src2", src2);
imshow("合成", dst);
waitKey(0);
return 0;
}
```

执行结果





3.2 计算N维数据关系

统计学里最基本的概念就是样本的均值、方差、标准差、协方差等，下面我们给定一个包含 n 个样本的集合，分别进行分析。

3.2.1 均值

未经分组的均值计算公式：

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{\sum_{i=1}^n x_i}{n}$$

3.2.2 方差

均值描述的是样本集合的中间点，它告诉我们的信息是有限的，而方差给我们描述的是样本集合的各个样本点到均值之间的平均距离。单一正态总体方差计算公式：

做真实的自己，用良心做教育

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}$$

3.2.3 标准差

方差对平均距离计算了平方，为了还原回原来的数量级，就有了标准差，标准差是对方差开根号，计算公式：

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

标准差描述各个点与均值距离的平均值，值越小表示数据越集中，例如：[0, 8, 12, 20]和[8, 9, 11, 12]，两个集合的均值都是 10，但显然两个集合各个值差别是很大的，计算两者的标准差，前者是 8.3，后者是 1.8，显然后者较为集中。

3.2.4 协方差

如果有另一个样本集合，也就是两个以上的样本集合，那么这两个样本集合间的各个点在时间或空间上有什么关系，其中一个样本中的点会不会像另一个样本中的点保持着一样的变化趋势，均值、方差、标准差都是解决一维内部各数据间的相关性问题（我们村的贫富差距问题）。当出现多维集合时，各个维度间的数据有无关联，可以参照一维的方法，首先将每个维度样本集合中每一个点的数据值减去该维度的平均值，再乘以另外一个维度的同样的差值，最后除以 $n-1$ 就是协方差 (n 就是每个维度样本个数，各维度一样)，这个协方差就可以反映两个维度间各数据的相关性，计算公式：

$$\mu_x = \frac{\sum_{i=0}^n x_i}{n} \quad \mu_y = \frac{\sum_{i=0}^n y_i}{n}$$

$$\text{cov}_{xy} = \frac{\sum_{i=0}^n (x_i - \mu_x)(y_i - \mu_y)}{(n-1)}$$

协方差的结果有什么意义？如果结果为正值，则说明两者是正相关的，如果结果为负值，则说明两者是负相关的，如果结果为 0，则表示两者之间没有关系。协方差只是说明了线性相关的方向问题，即从正无穷到负无穷，不能说明相关的程度，因为这个值可能很大也可能很小，所以还引出了相关系数=两个维度的协方差/(两个维度的标准差)，其值始终在-1 到 1 之间变化。

3.2.5 协方差矩阵

当出现多维数据时，若要对多维数据的相关性进行分析，那么就要用到协方差矩阵。

$$V = \begin{bmatrix} \Sigma x_1^2 / N & \Sigma x_1 x_2 / N & \dots & \Sigma x_1 x_c / N \\ \Sigma x_2 x_1 / N & \Sigma x_2^2 / N & \dots & \Sigma x_2 x_c / N \\ \dots & \dots & \dots & \dots \\ \Sigma x_c x_1 / N & \Sigma x_c x_2 / N & \dots & \Sigma x_c^2 / N \end{bmatrix}$$

Student	Math	English	Art
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

$$= \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix} \quad A$$

$$a = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix} \quad (1/5)$$

$$a = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix} - \begin{bmatrix} 66 & 60 & 60 \\ 66 & 60 & 60 \\ 66 & 60 & 60 \\ 66 & 60 & 60 \\ 66 & 60 & 60 \end{bmatrix} = \begin{bmatrix} 24 & 0 & 30 \\ 24 & 30 & -30 \\ -6 & 0 & 0 \\ -6 & 0 & 30 \\ -36 & -30 & -30 \end{bmatrix}$$

$$a'a = \begin{bmatrix} 24 & 24 & -6 & -6 & -36 \\ 0 & 30 & 0 & 0 & -30 \\ 30 & -30 & 0 & 30 & -30 \end{bmatrix} \begin{bmatrix} 24 & 0 & 30 \\ 24 & 30 & -30 \\ -6 & 0 & 0 \\ -6 & 0 & 30 \\ -36 & -30 & -30 \end{bmatrix} = \begin{bmatrix} 2520 & 1800 & 900 \\ 1800 & 1800 & 0 \\ 900 & 0 & 3600 \end{bmatrix}$$

$$V = a'a/n = \begin{bmatrix} 2520/5 & 1800/5 & 900/5 \\ 1800/5 & 1800/5 & 0/5 \\ 900/5 & 0/5 & 3600/5 \end{bmatrix} = \begin{bmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{bmatrix}$$

通过 opencv 接口完成计算的例子:

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace std;
using namespace cv;

int main(int argc, char const *argv[])
{
    Mat samples = (Mat_<double>(5, 3) << 90, 60, 90, 90, 90, 30, 60, 60, 60, 60, 60, 90,
30, 30, 30);
    Mat cov, mu;
    //mu: 保存均值
    //cov: 保存协方差
    calcCovarMatrix(samples, cov, mu, CV_COVAR_NORMAL | CV_COVAR_ROWS); //CV_COVAR_COLS
    cout << "means : " << endl;
    cout << mu << endl;
    cout << cov << endl;
    cout << "cov : " << endl;
    cout << cov/5 << endl;
    waitKey(0);
    return 0;
}
```

执行结果

```
edu@edu:~/debug/opencv$ ./opencv_t
means :
[66, 60, 60]
[2520, 1800, 900;
 1800, 1800, 0;
 900, 0, 3600]
cov :
[504, 360, 180;
 360, 360, 0;
 180, 0, 720]
edu@edu:~/debug/opencv$
```

3.3 特征值和特征向量

3.3.1 特征值与特征向量的定义

$$Ax = \lambda x$$

设 A 是 n 阶方阵，如果数值 λ 和 n 维非零列向量 x 使关系式 $Ax = \lambda x$ 成立(即只伸缩不旋转)，那么这样的数 λ 称为矩阵 A 的“特征值”。

$$\begin{bmatrix} .8 & .3 \\ .2 & .7 \end{bmatrix} \quad \begin{bmatrix} .70 & .45 \\ .30 & .55 \end{bmatrix} \quad \begin{bmatrix} .650 & .525 \\ .350 & .475 \end{bmatrix} \quad \dots \quad \begin{bmatrix} .6000 & .6000 \\ .4000 & .4000 \end{bmatrix}$$

$A \quad A^2 \quad A^3 \quad A^{100}$

$$Ax = \lambda x$$

$$Ax - \lambda x = 0$$

$$Ax - \lambda Ix = 0$$

$$(A - \lambda I)x = 0$$

定义 设 $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$ ，则

称 $|A - \lambda I| = \begin{vmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} - \lambda \end{vmatrix}$

为矩阵 A 的特征多项式，记作 $f(\lambda)$

做真实的自己，用良心做教育

$$A = \begin{bmatrix} 3 & 6 & -8 \\ 0 & 0 & 6 \\ 0 & 0 & 2 \end{bmatrix} \quad \det(A - \lambda I)$$

$$A - \lambda I = \begin{bmatrix} 3 & 6 & -8 \\ 0 & 0 & 6 \\ 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} = \begin{bmatrix} 3-\lambda & 6 & -8 \\ 0 & -\lambda & 6 \\ 0 & 0 & 2-\lambda \end{bmatrix}$$

$$\begin{aligned} \det(A - \lambda I) &= (3 - \lambda)(-\lambda)(2 - \lambda) + (6)(6)(0) + (-8)(0)(0) \\ &\quad - (0)(-\lambda)(-8) - (0)(6)(3 - \lambda) - (-\lambda)(0)(6) \\ &= -\lambda(3 - \lambda)(2 - \lambda) \end{aligned}$$

特征值有三个 $\lambda = 0, 2, \text{ or } 3$

当特征值为0时

$$(A - \lambda I)\mathbf{x} = \mathbf{0}$$

$$(A - 0I)\mathbf{x} = \mathbf{0}$$

$$A\mathbf{x} = \mathbf{0}$$

$$\begin{bmatrix} 3 & 6 & -8 \\ 0 & 0 & 6 \\ 0 & 0 & 2 \end{bmatrix} \mathbf{x} = \mathbf{0}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_2 \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix}$$

当特征值为2时

$$(A - \lambda I)\mathbf{x} = \mathbf{0}$$

$$(A - 2I)\mathbf{x} = \mathbf{0}$$

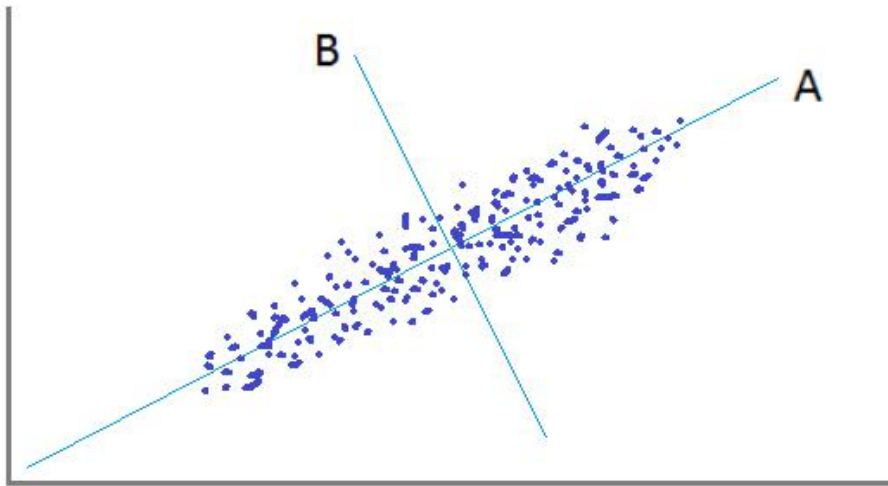
$$\left(\begin{bmatrix} 3 & 6 & -8 \\ 0 & 0 & 6 \\ 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \right) \mathbf{x} = \mathbf{0}$$

$$\begin{bmatrix} 1 & 6 & -8 \\ 0 & -2 & 6 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x} = \mathbf{0}$$

最终得到特征值为2时特征向量

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_3 \begin{bmatrix} -10 \\ 3 \\ 1 \end{bmatrix}.$$

计算得到特征值和特征向量的意义？特征值与特征向量表达了一个线性变换的特征，特征向量将一个矩阵进行正交分解，判断出在哪些方向只拉伸不扭曲来简化计算量，得到了特征值与特征向量就是得到了某个矩阵导致的伸缩比例和伸缩方向，其目的主要用于降维。



上图通过分析特征值与特征向量就可以将二维数据变成一维数据分析。

3.3.2 opencv 接口计算特征值与特征向量

```
bool eigen(InputArray src, OutputArray eigenvalues, OutputArray eigenvectors = noArray());
```

功能：获取特征值和特征向量

参数：

src：原图或者数据

eigenvalues：特征值

eigenvectors：特征向量

案例

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace std;
using namespace cv;

int main(int argc, char const *argv[])
{
    Mat data = (Mat_<double>(2, 2) <<
        1, 2,
        2, 1);
    Mat eigen values, eigen vector;
    eigen(data, eigen values, eigen vector);
    for(int i = 0; i < eigen values.rows; i++){
        printf("eigen value %d: %.3f\n", i, eigen values.at<double>(i));
    }
    cout << "eigen vector: " << endl << eigen vector << endl;
    waitKey(0);
    return 0;
}
```

做真实的自己，用良心做教育

执行结果

```
lzx@qfedu:~/share/work/test/opencv$ ./a.out
eigen value 0: 3.000
eigen value 1: -1.000
eigen vector:
[0.7071067811865475, 0.7071067811865475;
 0.7071067811865475, -0.7071067811865475]
lzx@qfedu:~/share/work/test/opencv$
```

可以将矩阵扩大两倍看看，得到的特征值与特征向量如何变化。

第 4 章 PCA 原理和应用

4.1 PCA 原理介绍

4.1.1 PCA 概念

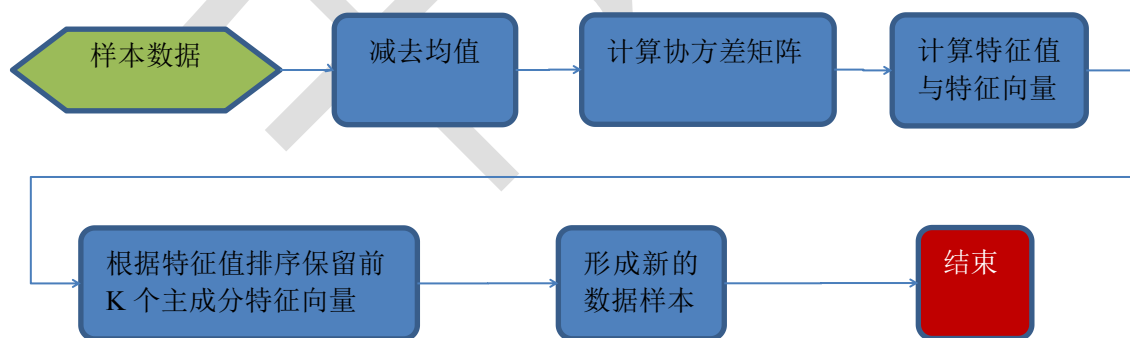
PCA (Principal Component Analysis) 是一种常用的数据分析方法。PCA 通过线性变换将原始数据变换为一组各维度线性无关的表示，可用于提取数据的主要特征分量，常用于高维数据的降维处理。

PCA 特点：

主成分不变

细微损失

高维数据到低维数据



数据互换：

输出数据 = 前 K 个特征向量组合 * 均值调整后的数据

均值调整后的数据 = 前 K 个特征向量行组合^T * 输出数据

原始数据 = 前 K 个特征向量行组合^T * 输出数据 + 均值数据

做真实的自己，用良心做教育

4.1.2 opencv 中 PCA 相关接口

4.1.2.1 cvtColor()

```
void cvtColor( InputArray src, OutputArray dst, int code, int dstCn = 0 );
```

功能:颜色空间转换

参数:

src: 原图

dst: 保存转换后的图

dstCn: 转换方式, 比如 COLOR_BGR2GRAY 转换成灰度图

4.1.2.2 threshold()

```
double threshold( InputArray src, OutputArray dst, double thresh, double maxval, int type );
```

功能:图像的二值化, 就是将图像上的像素点的灰度值设置为 0 或 255, 也就是将整个图像呈现出明显的只有黑和白的视觉效果

参数:

src: 原图

dst: 保存转换后的图

thresh: 设定的阈值

maxval: 当灰度值大于 (或小于) 阈值时将该灰度值赋成的值

type: 当前二值化的方式

4.1.2.3 findContours()

```
void findContours( InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset = Point());
```

功能:轮廓检测

参数:

image: 一般是灰度图或者二值化图, 多是二值化图

contours: 定义为 “vector<vector<Point>> contours”, 是一个双重向量, 每一组点集就是一个轮廓, 有多少轮廓, contours 就有多少元素

hierarchy: 定义为 “vector<Vec4i> hierarchy”, hierarchy 是一个向量, 向量内每个元素都是一个包含 4 个 int 型的数组

mode: 定义轮廓的检索模式

CV_RETR_EXTERNAL, 只检测最外围轮廓, 包含在外围轮廓内的内围轮廓被忽略

CV_RETR_LIST, 检测所有的轮廓, 包括内围、外围轮廓, 但是检测到的轮廓不建立等级关系

CV_RETR_CCOMP, 检测所有的轮廓, 但所有轮廓只建立两个等级关系

CV_RETR_TREE, 检测所有轮廓, 所有轮廓建立一个等级树结构

method: 定义轮廓的近似方法

CV_CHAIN_APPROX_NONE, 保存物体边界上所有连续的轮廓点到 contours 向量内;

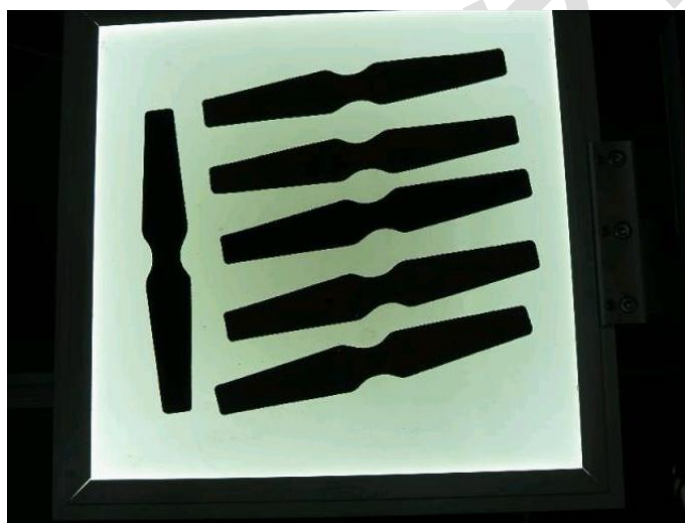
CV_CHAIN_APPROX_SIMPLE, 仅保存轮廓的拐点信息, 把所有轮廓拐点处的点保存入 contours 向量内, 拐点与拐点之间直线段上的信息点不予保留; CV_CHAIN_APPROX_TC89_L1, 使用 teh-Chin1 chain 近似算法;

CV_CHAIN_APPROX_TC89_KCOS, 使用 teh-Chin1 chain 近似算法。

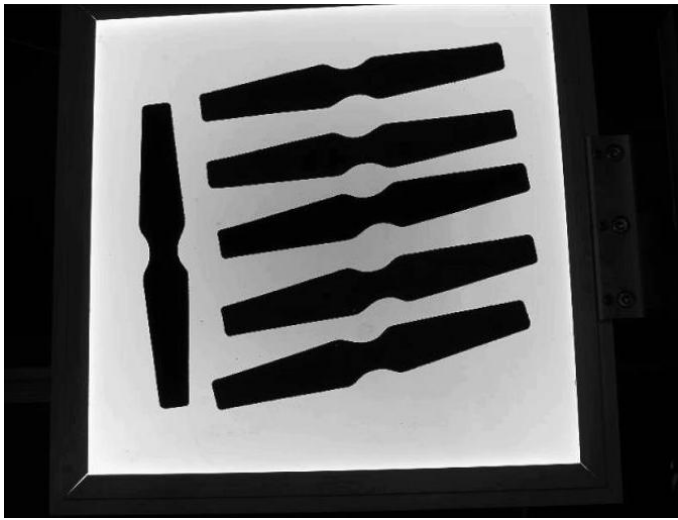
offset: Point 偏移量, 所有的轮廓信息相对于原始图像对应点的偏移量, 相当于在每一个检测出的轮廓点上加上该偏移量, 并且 Point 还可以是负值

案例: [参看基础代码](#)

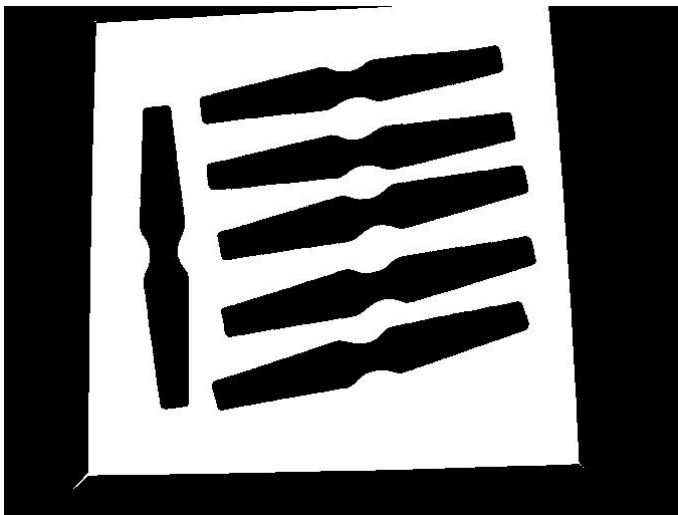
原图:



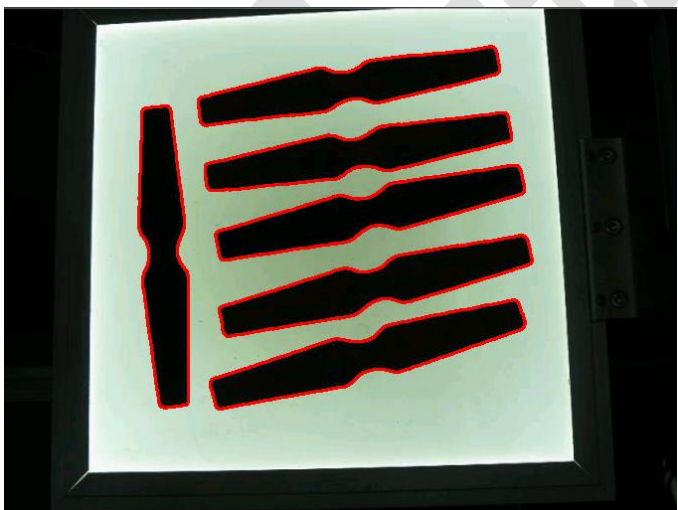
灰度图:



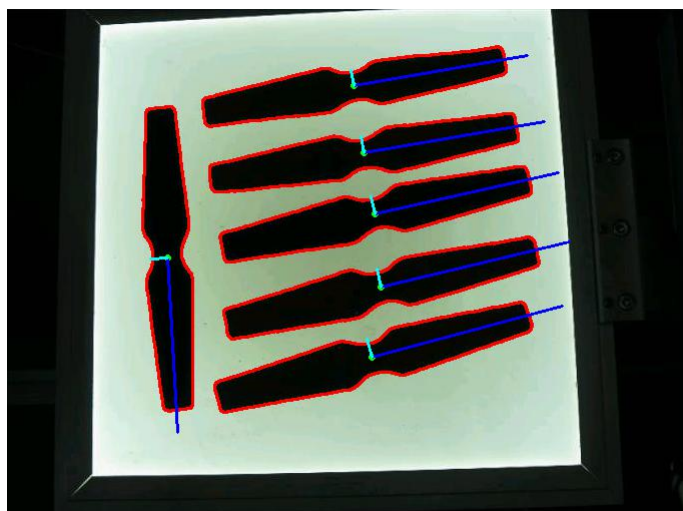
二值化图:



轮廓检测图:



最终图:



第 5 章 opencv 实时人脸识别应用开发

5.1 人脸检测

5.1.1 equalizeHist()

```
void cvEqualizeHist( const CvArr* src, CvArr* dst );
```

功能：直方图均衡化处理

参数：

src：单通道灰度图

dst：得到对比度更强的输出图像

返回值：无

5.1.2 级联分类器 CascadeClassifier

分类器是判别某个事物是否属于某种分类的器件，其结果要么是，要么不是，级联分类器，可以理解为将 N 个单类的分类器串联起来，如果一个事物能属于这一系列串联起来的所有分类器，则最终结果就成立，若有一项不符，则判定为不成立。比如人脸，它有很多属性，我们将每个属性做一个分类器，如果一个模型符合了我们定义的人脸的所有属性，则我们人为这个模型就是一个人脸，比如人脸需要要有两条眉毛，两只眼睛，一个鼻子，一张嘴，一个大概 U 形状的下巴或者是轮廓等等。

常用方法：

一、从文件中加载级联分类器

```
CV_WRAP bool load( const String& filename );
```

做真实的自己，用良心做教育

二、检测级联分类器是否被加载

```
CV_WRAP bool load( const String& filename );
```

三、目标检测方法

```
CV_WRAP void detectMultiScale( InputArray image,
                                CV_OUT std::vector<Rect>& objects,
                                double scaleFactor = 1.1,
                                int minNeighbors = 3, int flags = 0,
                                Size minSize = Size(),
                                Size maxSize = Size() );
```

功能：检测输入图像中不同大小的对象。检测到的对象以矩形列表的形式保存

参数：

image: 包含检测对象的图像的 CV_8U 类型矩阵

objects: 矩形的向量，其中每个矩形包含被检测的对象，矩形可以部分位于原始图像之外

scaleFactor: 指定在每个图像缩放时的缩放比例

minNeighbors: 指定每个候选矩形需要保留多少个相邻矩形

flags: 含义与函数 cvHaarDetectObjects 中的旧级联相同，一般是 0，它不用于新的级联

minSize: 对象最小大小，小于该值的对象被忽略

maxSize: 最大可能的对象大小，大于这个值的对象被忽略

返回值：无

5.1.3 rectangle()

```
void rectangle(Mat& img, Point pt1, Point pt2,
               const Scalar& color, int thickness=1,
               int lineType=8, int shift=0)
```

功能：绘制简单、指定粗细或者带填充的矩形

参数：

Img: 图像来源

pt1: 矩形的一个顶点，一般是左上角

pt2: 矩形对角线上的另一个顶点，一般是右下角

color: 线条颜色 (BGR) 或亮度 (灰度图像) (grayscale image)

Thickness: 组成矩形的线条的粗细程度, 取负值时 (如 CV_FILLED) 函数绘制填充了色彩的矩形

line_type: 线条的类型

shift: 坐标点的小数点位数

返回值: 无

5.1.3 VideoCapture 类

VideoCapture 既支持从视频文件读取, 也支持直接从摄像机等监控器中读取, 还可以读取 IP 视频流, 要想获取视频需要先创建一个 VideoCapture 对象, 对象创建以后, OpenCV 将会打开文件并做好读取准备, 如果打开成功, 我们将可以开始读取视频的帧。

捕获本地视频:

```
VideoCapture cap( "video.mp4" );
```

捕获摄像机等监控器:

```
VideoCapture cap(-1);
```

捕获 IP 视频流:

```
VideoCapture cap( "http://youku.elecfans.com/video.flv" );
```

判断打开是否成功:

```
if (!cap.isOpened()) {  
    cout << "打开失败!!" << endl;  
    return -1;  
}
```

读取一帧图像:

```
if (!cap.read(img)){  
    cap.set(CV_CAP_PROP_POS_FRAMES, 0);  
    continue;  
}
```

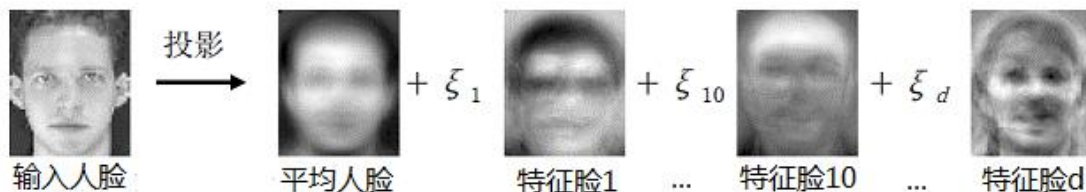
5.2 人脸识别

5.2.1 EigenFace 介绍

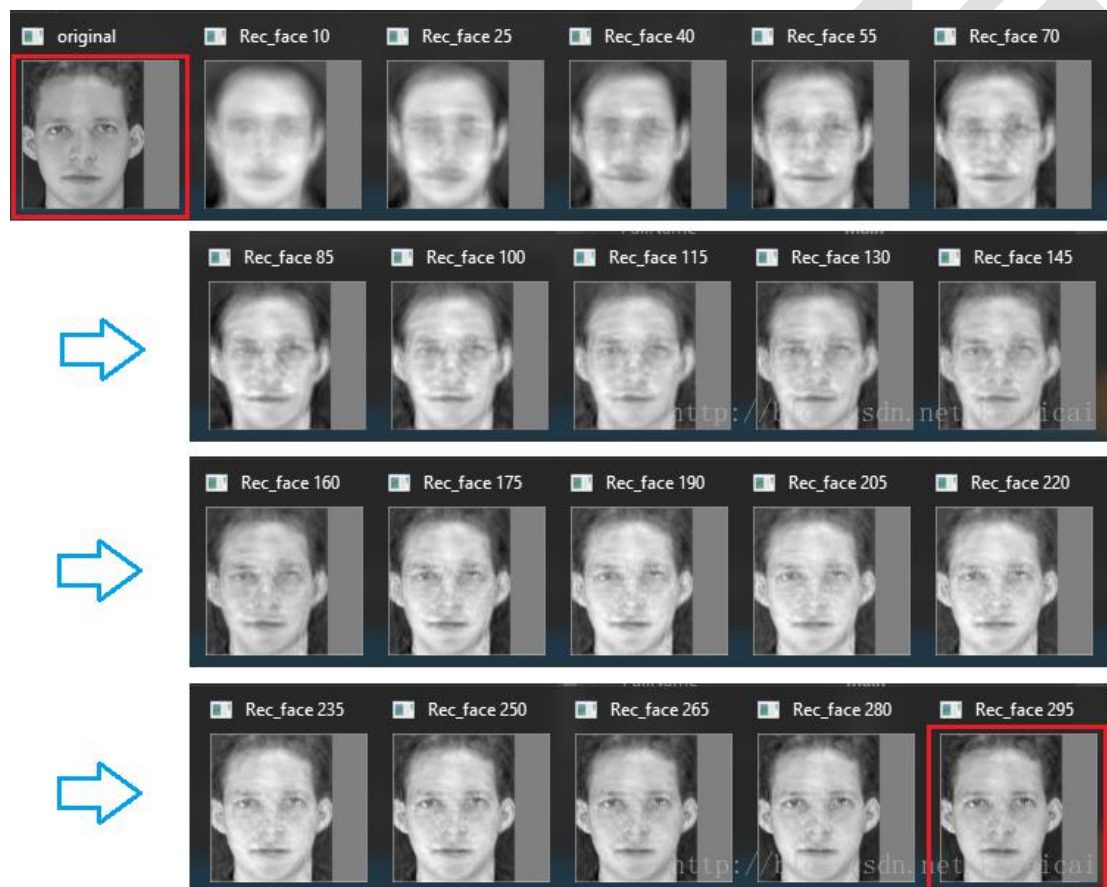
EigenFace 在人脸识别历史上应该是具有里程碑式意义的, 其被认为是第一种有效的人脸识别算法。1987 年 Sirovich and Kirby 为了减少人脸图像的代表采用了 PCA (主成分分析) 的方法进行降维, 1991 年 Matthew Turk 和 Alex Pentland 首次将 PCA 应用于人脸识别, 即将原始图像投影到特征空间, 得到一系列降维图像, 取其主成份表示人脸, 因其主成份中就包含人脸的形状, 估称

EigenFace 为“特征脸”。

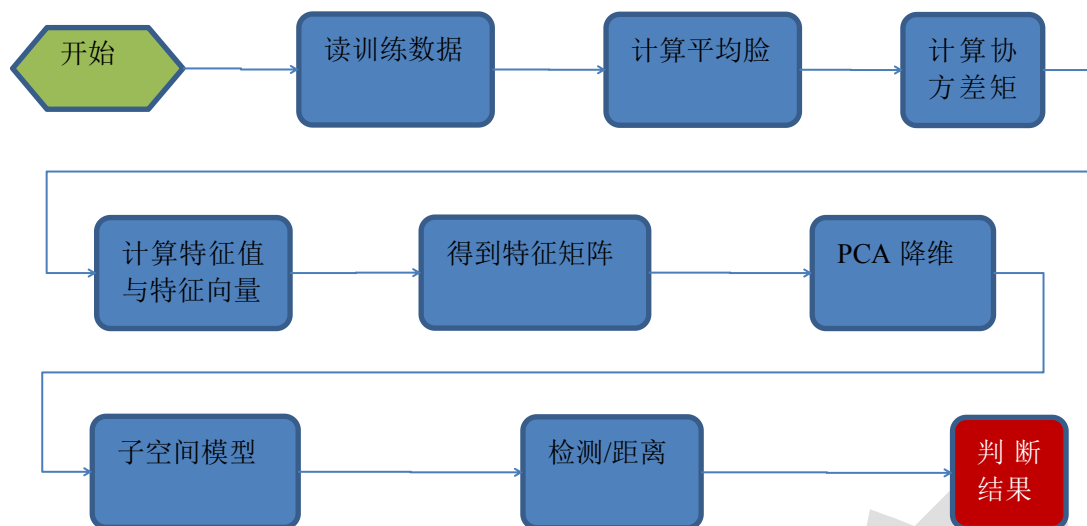
EigenFace 是一种基于统计特征的方法，将人脸图像视为随机向量，并用统计方法辨别不同人脸特征模式。EigenFace 的基本思想是，从统计的观点，寻找人脸图像分布的基本元素，即人脸图像样本集协方差矩阵的特征向量，以此近似的表征人脸图像。具体过程如下：



下面是从第一幅原始图像，原图和每隔 15 个特征向量进行重建后的效果图，很明显随着引入进来的特征向量越来越多，重建后的效果也越来越接近原图：



EigenFace 的工作流程如下：



5.2.2 人脸识别算法对比

前一节主要介绍了 eigenFace，下面再简单介绍两种其它的识别算法：

一、FisherFace

FisherFace 是一种基于 LDA(全称 Linear Discriminant Analysis, 线性判别分析)的人脸识别算法, 而 LDA 是 Ronald Fisher 于 1993 年提出来的, 而 eigenFace 是基于 PCA 实现的, LDA 和 PCA 相同的地方是, 都有利用特征值排序找到主成份的过程, 但是不同的是 PCA 求的是协方差矩阵的特征值, 而 LDA 是求的是一个更为复杂的矩阵的特征值。其中需要注意的是在求均值时, 和 PCA 也是有所不同的, LDA 对每个类别样本求均值, 而 PCA 是对所有样本数据求均值, 来得到平均脸。采用 Fisherface 方法对人脸进行识别对光照、人脸姿态的变化更不敏感, 有助于提高识别效果, 但左右偏转脑袋, 尽量不要超过 15° , 并对上下偏转比较敏感。

二、LBPH

LBPH 是利用局部二值模式直方图的人脸识别算法, LBP 是典型的二值特征描述子, 所以相比前面 EigenFace 和 FisherFace, 更多的是整数计算, 而整数计算的优势是可以通过各种逻辑操作来进行优化, 因此效率较高。另外通常光照对图中的物件带来的影响是全局的, 也就是说照片中的物体明暗程度, 是往同一个方向改变的, 可能是全部变亮或全部变暗, 因此 LBP 特征对光照具有比较好的鲁棒性。

5.2.3 opencv 人脸识别案例分析

5.2.3.1 准备样本

一、建议每个样本人物图像至少 8 张, 拍摄样本图像时, 注意头部尽量不要上下左右摆头。



二、编写样本描述文件 image.txt, 主要描述样本图像路径和样本图像对应的标签, 格式如下:

```
image/face_1.jpg;17
image/face_2.jpg;17
image/face_3.jpg;17
image/face_4.jpg;17
image/face_5.jpg;17
image/face_6.jpg;17
image/face_7.jpg;17
image/face_8.jpg;17
image/face_11.jpg;18
image/face_22.jpg;18
image/face_33.jpg;18
image/face_44.jpg;18
image/face_55.jpg;18
image/face_66.jpg;18
image/face_77.jpg;18
image/face_88.jpg;18
```

三、实现流程

加载样本图像

```
vector<Mat> images;
vector<int> labels;
images.push_back(imread(path, 0));
labels.push_back(num);
```

训练样本, 并给样本打上标签

```
Ptr<BasicFaceRecognizer> model = createEigenFaceRecognizer();
model->train(images, labels);
```

识别比对目标图像, 将找到的样本对应标签返回

```
int label = model->predict(obj_img);
```

识别结果:



还可以通过上面最后介绍的两种识别算法进行测试，对比识别效果：

```
Ptr<BasicFaceRecognizer> model = createFisherFaceRecognizer();
```

```
Ptr<LBPHFaceRecognizer> model = createLBPHFaceRecognizer();
```

5.3 opencv 非特定目标检测之几何图形识别

5.3.1 准备训练样本素材

5.3.1.1 正样本

又叫积极样本，可以自己用画图板，多画几个图形统一放在 pos 目录，如下：



5.3.1.2 负样本

又叫消极样本，你可自己从网上爬一些不相关的图片，也可以自己从电脑找一些，数量大概是正样本的 3 倍以上，尺寸是正样本的 8-12 倍，图片尺寸必须调整为统一大小，最后统一放在 neg 目录。

批量调整图片大小命令

```
find ./neg -name '*.jpg' -exec convert -resize 100X100! {} {} \;
```

5.3.2 将图片进行灰度处理

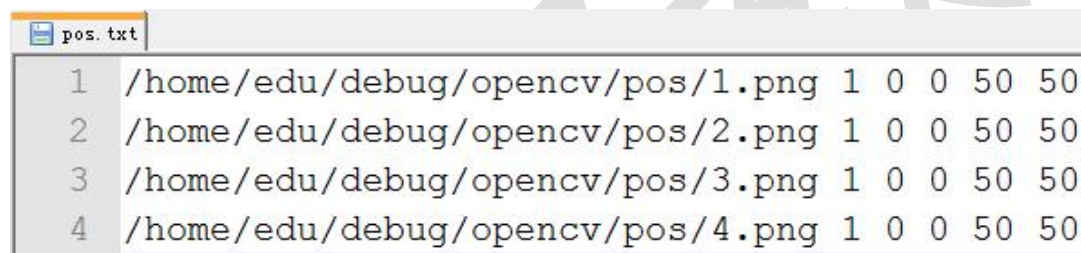
通过下面代码进行批量灰度处理：

```
#include <fstream>
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
```

做真实的自己，用良心做教育

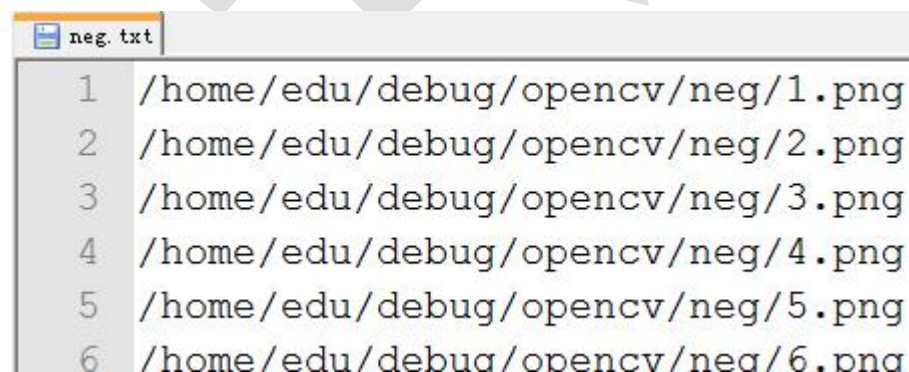
```
using namespace cv;
//g++ -o gray gray.cpp `pkg-config --cflags --libs opencv`
//./gray img 100
int main(int argc, char *argv[])
{
    char src [250];
    char obj [250];
    if(argc != 3){
        printf("./gray path 100(num)\n");
        return 0;
    }
    for(int i=1;i<atoi(argv[2])+1;i++){
        //将数字字母拼接在一起得到读取文件的路径
        sprintf(src, "%s/%d.jpg", argv[1], i);
        cout<<src<<endl;
        sprintf(obj, "%s/%d.jpg", argv[1], i);
        printf("%s\n", obj);
        //从指定路径 buffer 中读取图片
        Mat srcImage = imread(src), grayImage;
        cvtColor(srcImage, grayImage, CV_BGR2GRAY);
        imwrite(obj, grayImage);
    }
    cvWaitKey();
    return 0;
}
```

5.3.3 生成 pos.txt 积极图片描述文件



```
pos.txt
1 /home/edu/debug/opencv/pos/1.png 1 0 0 50 50
2 /home/edu/debug/opencv/pos/2.png 1 0 0 50 50
3 /home/edu/debug/opencv/pos/3.png 1 0 0 50 50
4 /home/edu/debug/opencv/pos/4.png 1 0 0 50 50
```

5.3.4 生成 neg.txt 消极图片描述文件



```
neg.txt
1 /home/edu/debug/opencv/neg/1.png
2 /home/edu/debug/opencv/neg/2.png
3 /home/edu/debug/opencv/neg/3.png
4 /home/edu/debug/opencv/neg/4.png
5 /home/edu/debug/opencv/neg/5.png
6 /home/edu/debug/opencv/neg/6.png
```

5.3.5 生成 vec 文件

```
opencv_createsamples -vec pos.vec -info pos.txt -num 4 -w 50 -h 50
```

5.3.6 开始样本训练

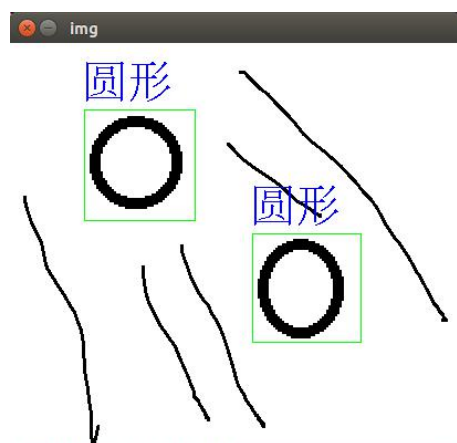
做真实的自己，用良心做教育

```
opencv_traincascade -data xml -vec pos.vec -bg neg.txt -numPos 4 -numNeg 30 -numStages 8  
-w 50 -h 50 -minHitRate 0.999 -maxFalseAlarmRate 0.2 -weightTrimRate 0.95 -featureType  
HAAR -mode ALL
```

上以过程因环境不同，需要反复修改参数来达到最终的训练效果，一般正常训练一个相对稳定的样本模型需要一两天时间。

5.3.6 对几何图进行识别

通过之前讲的人脸识别代码，采用刚刚训练好的 xml 分类器，加载一张几何图看看识别效果：



第 6 章 dlib ubuntu 环境搭建

6.1 dlib 安装准备工作

6.1.1 安装环境

Ubuntu16.04 LTS
dlib-19.19

6.1.2 源码获取

软件官网下载地址：

<http://dlib.net/files/dlib-19.19.tar.bz2>

http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

http://dlib.net/files/shape_predictor_5_face_landmarks.dat.bz2

http://dlib.net/files/dlib_face_recognition_resnet_model_v1.dat.bz2

通过 bunzip2 进行解压

6.2 dlib 具体安装步骤

做真实的自己，用良心做教育

6.2.1 解压源码

将两个下载的源码文件拷贝到 Ubuntu 中并解压

```
tar jxvf dlib-19.19.tar.bz2
```

6.2.2 配置 dlib

创建编译安装目录：

```
mkdir dlib-19.19/mybuild
```

```
mkdir dlib-19.19_install
```

通过 cmake 工具生成 Makefile：

```
cd dlib-19.19/mybuild
```

```
cmake -D CMAKE_BUILD_TYPE=Release -D DLIB_GENERATE_PKGCONFIG=ON -D  
CMAKE_INSTALL_PREFIX=/home/edu/ai/dlib-19.19_install ../dlib
```

6.2.3 编译安装 dlib

进入 dlib-19.19/mybuild 目录完成编译安装(该过程根据不同配置的计算可能需要 20 分钟左右)：

```
make -j4
```

```
make install
```

6.3 dlib 环境配置

6.3.1 添加 dlib 库

打开或创建 dlib-1.conf 文件，并添加 dlib 安装路径

```
sudo gedit /etc/ld.so.conf.d/dlib-1.conf
```

```
/home/edu/ai/dlib-19.19_install/lib <添加内容>
```

6.3.2 使 dlib 配置文件生效

```
sudo ldconfig
```

6.3.3 配置 bash 环境变量

```
sudo gedit ~/.bashrc <在文件末尾添加如下内容>
```

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/home/edu/ai/dlib-19.19_install/lib/pkgconfig
```

第四步：生效配置文件

```
source ~/.bashrc <使环境变量立即生效>
```


6.3.4 验证 dlib 环境配置是否成功

```
pkg-config --cflags --libs dlib-1
```

6.4 dlib 测试

找到 dlib-19.9/examples/ 目录, 该目录下面有很多测试程序, 可以直接编译所有测试程序, 也可以将要测试的程序拷贝出来单独编译。

执行 `cmake .` 用于生成 makefile

```
cmake .
```

执行 `make`

```
make
```

执行生成的可执行文件

```
./dnn_face_recognition_ex faces/bald_guys.jpg
```

执行结果



第 7 章 dlib 实时人脸识别应用开发

7.1 人脸检测

dlib 人脸检测器参考代码:

```
#include <dlib/opencv.h>
#include <dlib/image_processing/frontal_face_detector.h>
```

做真实的自己，用良心做教育


```
#include <dlib/image_processing/render_face_detections.h>
#include <dlib/image_processing.h>
#include <dlib/gui_widgets.h>
#include <opencv/cv.h>
#include <opencv/highgui.h>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>

using namespace dlib;
using namespace std;
using namespace cv;

int main(int argc, char *argv[])
{
    frontal_face_detector detector = get_frontal_face_detector();
    cv::Mat mimg = cv::imread(argv[1]);
    dlib::cv_image<bgr_pixel> img(mimg); //转成 dlib 格式提取特征
    std::vector<dlib::rectangle> faces = detector(img);
    for (unsigned int i = 0; i < faces.size(); ++i)
        cv::rectangle(mimg, cv::Rect(faces[i].left(), faces[i].top(),
                                     faces[i].width(), faces[i].width()), cv::Scalar(0, 0, 255),
                      1, 1, 0); //画矩形框
    cv::imshow("人脸", mimg);
    cv::waitKey(0);
}
```

7.2 提取人脸特征点

dlib 官网有一个训练好的特征模型 shape_predictor_68_face_landmarks.dat, 搭建环境时下载的, 人脸特征提取参考代码:

```
#include <dlib/opencv.h>
#include <dlib/image_processing/frontal_face_detector.h>
#include <dlib/image_processing/render_face_detections.h>
#include <dlib/image_processing.h>
#include <dlib/gui_widgets.h>
#include <opencv/cv.h>
#include <opencv/highgui.h>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/freetype.hpp>

using namespace dlib;
using namespace std;
using namespace cv;

int main(int argc, char *argv[])
{
    frontal_face_detector detector = get_frontal_face_detector();
    shape_predictor sp;
    //提取特征
    deserialize("shape_predictor_68_face_landmarks.dat") >> sp;
    cv::Mat mimg = cv::imread(argv[1]);
```

做真实的自己，用良心做教育

```
dlib::cv_image<bgr_pixel> img(mimg);
std::vector<dlib::rectangle> faces = detector(img);
std::vector<full_object_detection> shapes;
for (unsigned long i = 0; i < faces.size(); ++i)
    shapes.push_back(sp(img, faces[i]));

for (unsigned long j = 0; j < faces.size(); ++j)
if (!shapes.empty()) {
    for (int i = 0; i < 68; i++)//标记特征
        circle(mimg, cvPoint(shapes[j].part(i).x(),
                               shapes[j].part(i).y()), 1,
                cv::Scalar(0, 255, 0), -1);
}
cv::imshow("Dlib 特征点", mimg);
waitKey(0);
}
```

如果要绘制出人脸特征，dlib 提供了 `render_face_detections(shapes)` 接口，可以通过 dlib 自带的 `image_window` 类进行绘制。

7.3 人脸识别

dlib 人脸识别采用了 Resnet 残差神经网络，识别精度高于普通神经网络，同样我们可以到官网去下载训练好的模型 `dlib_face_recognition_resnet_model_v1.dat`，通过 `net()` 接口返回 128 维人脸特征，然后再通过目标图像也同样得到 128 维人脸特征，将两组特征进行对比即可判断出要识别的对象。

具体实现参考代码：从图像、视频、摄像头识别出目标人物

7.4 dlib 对非特定目标识别之手势识别

7.4.1 编译训练工具

一、找到 `imglab` 工具源码目录进行配置编译

```
cd tools/imglab/
cmake .
make
```

编译完成后将生成的 `imglab` 工具拷贝到样本照片所在目录。

二、找到 `examples/train_object_detector.cpp` 样本训练工具，将其拷贝出来单独进行编译，并生成 `train_object_detector` 命令。

7.4.1 训练样本采集

一、拍摄样本照片

dlib 对样本照片并没有太多要求，会做二次处理，样本拍摄建议在光线充足的情况下完成采集，

做真实的自己，用良心做教育

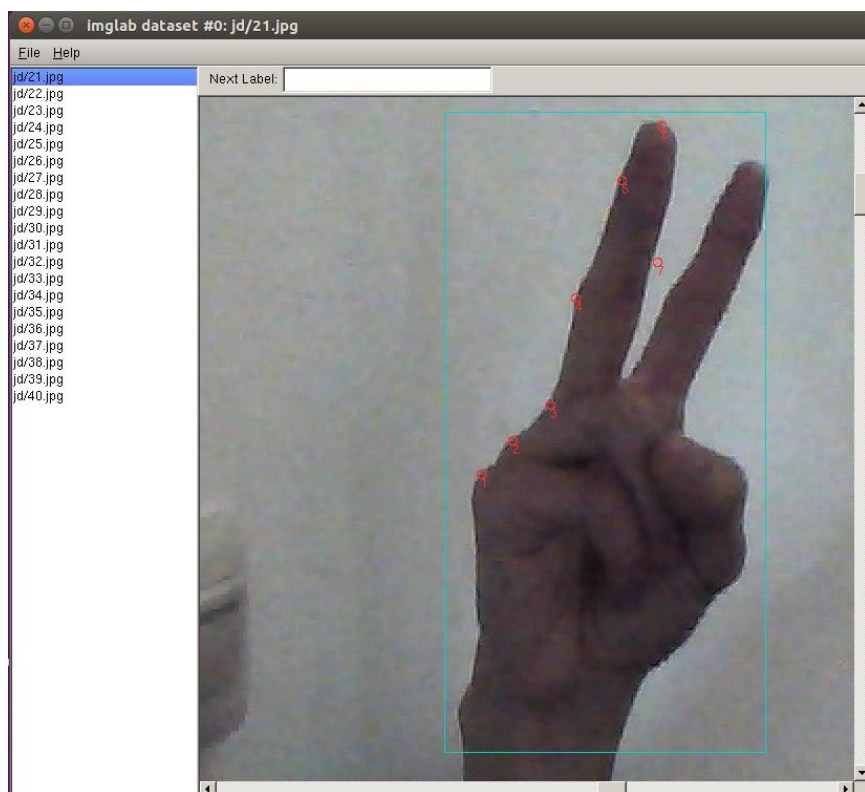
样本数量越多越好，并将拍好的照片统一放在一个目录，比如 img 下面等待处理。

二、生成 xml 描述文件

```
./imglab -c data.xml img
```

三、手动标记检测目标或特征点

在弹出的图形化工具上面对需要检测的特征进行标记，通过 shift 选择识别对象，双击选中对象以后，右键可以标记特征点，标记完成点 File->save 保存结果到 xml 中。



命令如下：

```
./imglab data.xml
```

如果要标记特征点，需要增加参数

```
./imglab mydataset.xml --parts "1 2 3 4 5 6 7 8 9 10"
```

打开 xml 可以查看目标或特征的标记坐标，同时还生成了一个 image_metadata_stylesheet.xml 样式文件。

四、开始训练样本

```
./train_object_detector -tv data.xml
```

训练结束后会生成 object_detector.svm 模型序列，这个模型就可以用于对象检测了。

五、训练模型测试

一般先从训练样本中随便找张照片进行测试，测试如果能够成功找出目标后，再用于实际的目标识别。

做真实的自己，用良心做教育

别:

```
./train_object_detector photo.png
```

具体手势识别程序怎么写，请参看代码！

千锋教育