

- 项目概述篇（案例演示）
 - 完整项目运行演示
 - 人脸特征点 (68,72,84,194)
 - 项目系统架构设计



2.3 应用架构

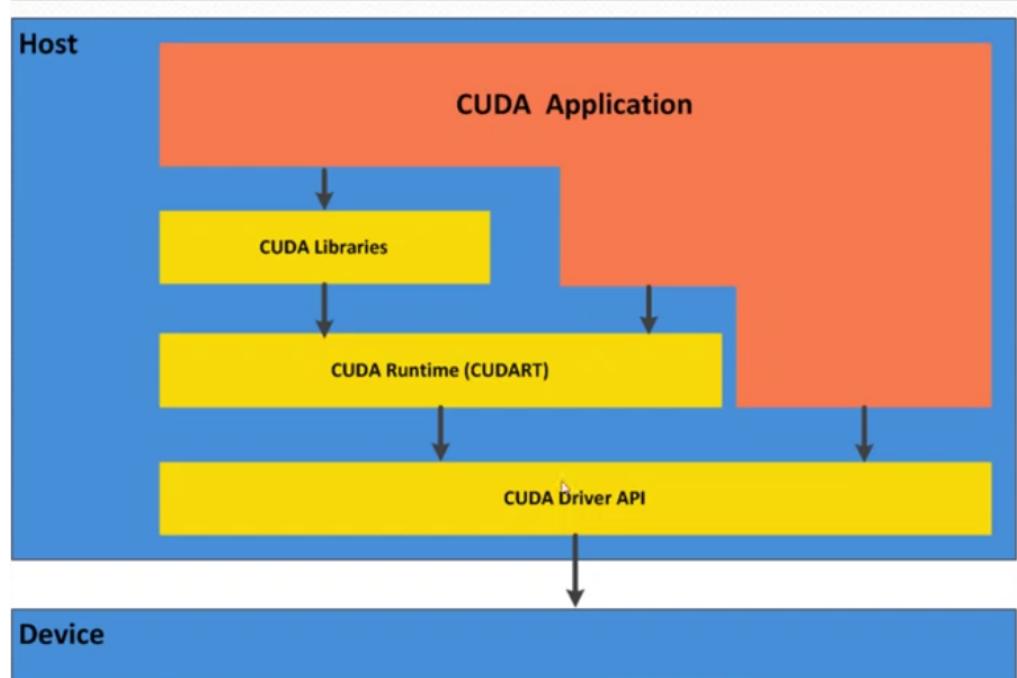


2.4 数据架构

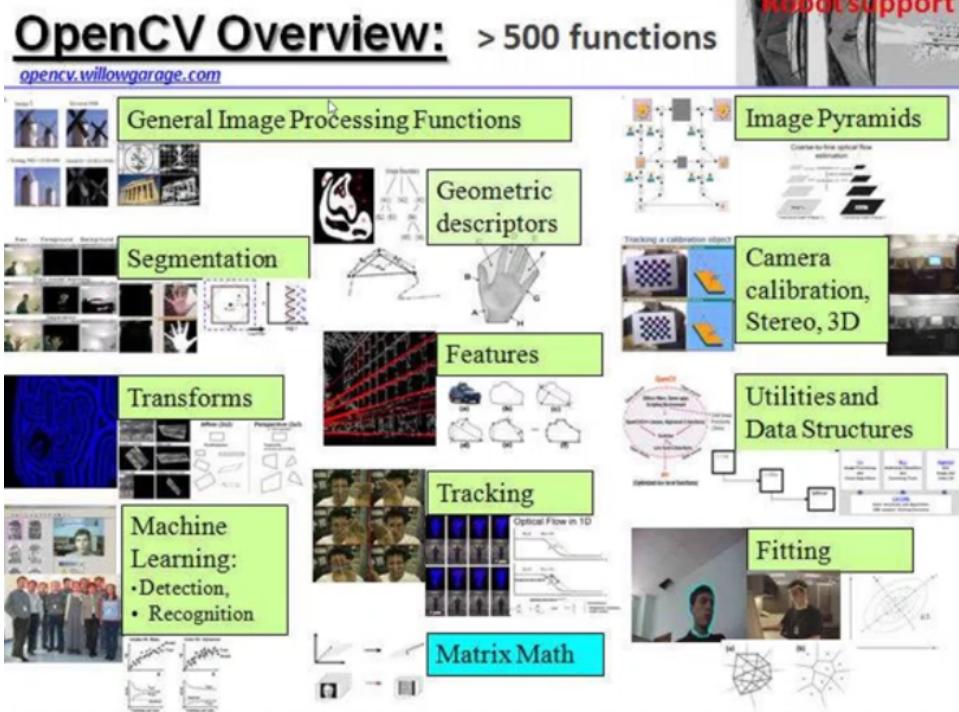


- 项目关键技术说明

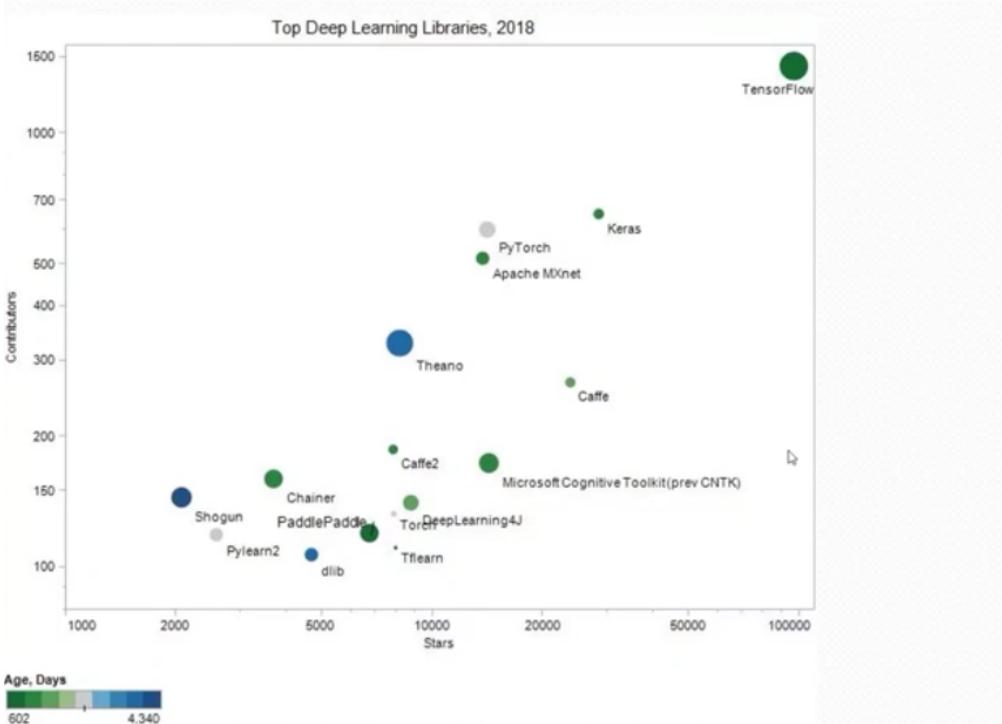
3.2 人脸识别的关键技术：并行计算框架（cuda）



3.2 人脸识别的关键技术：专业图像处理框架

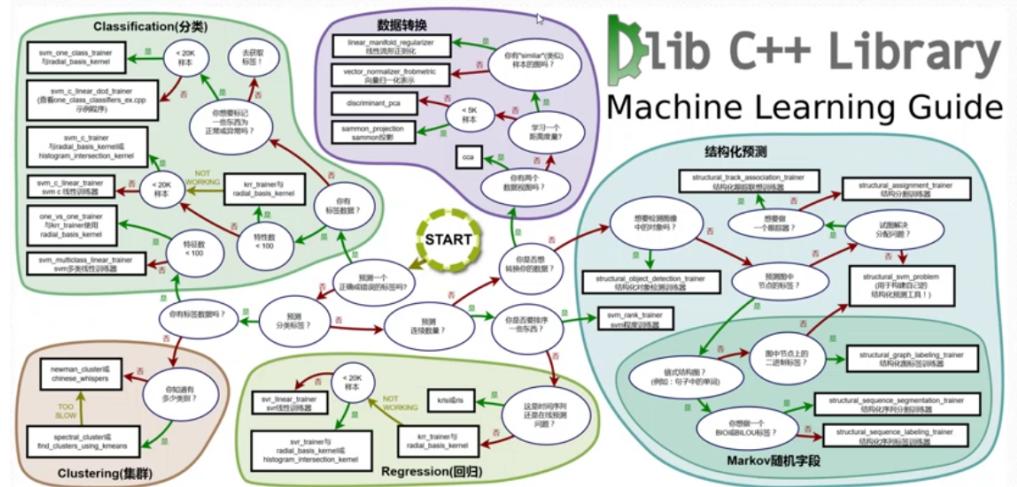


3.2 人脸识别的关键技术：深度学习框架



3.2 人脸识别的关键技术：深度学习框架

网易云课堂



3.2 人脸识别的关键技术：算法模型

网易云课堂

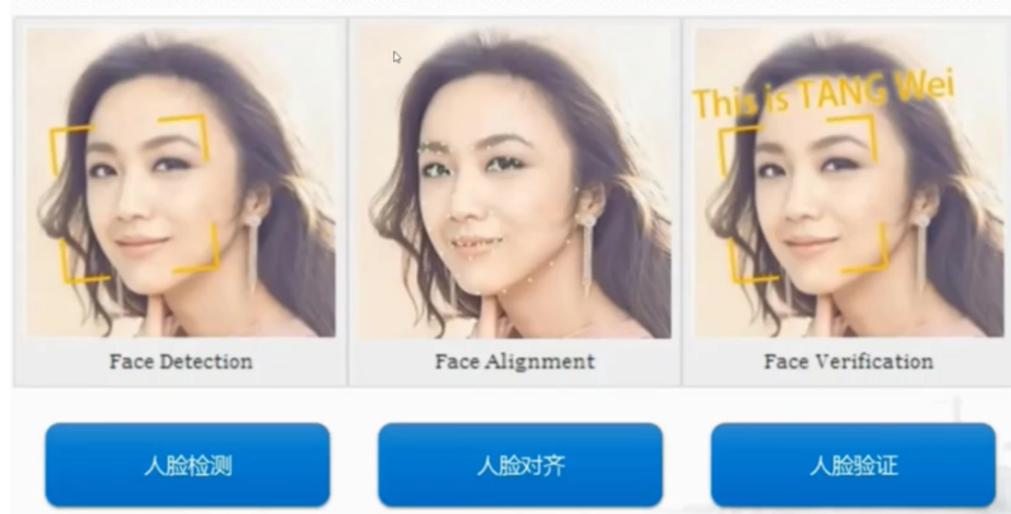
Simile classifiers ¹¹¹	0.8472 ± 0.0041
Attribute + Simile classifiers ¹¹¹	0.8554 ± 0.0035
Multiple LE + comp ¹¹¹	0.8445 ± 0.0046
Associate-Predict ¹¹¹	0.9057 ± 0.0056
Tom-vs-Pete ¹¹¹	0.9310 ± 0.0125
Tom-vs-Pete + Attribute ¹¹¹	0.9330 ± 0.0128
combined Joint Bayesian ¹¹¹	0.9242 ± 0.0108
high-dim LBP ¹¹¹	0.9517 ± 0.0113
DFID ¹¹¹	0.8402 ± 0.0044
TL Joint Bayesian ¹¹¹	0.9623 ± 0.0108
Face.com ^{12011b¹²}	0.9130 ± 0.0030
Face++ ¹²	0.9950 ± 0.0036
DeepFace-ensemble ¹²	0.9735 ± 0.0025
ConvNet-RBM ¹²	0.9252 ± 0.0038
POOF-gradi ¹²	0.9313 ± 0.0040
POOF-HOG ¹²	0.9280 ± 0.0047
FR-FCN ¹²	0.9645 ± 0.0025
DeepID ¹²	0.9745 ± 0.0026
GaussianFace ¹²	0.9852 ± 0.0066
DeepID2 ¹²	0.9915 ± 0.0019
TCC ¹²	0.9333 ± 0.0124
DeepID2+ ¹²	0.9947 ± 0.0012
betaface.com ¹²	0.9953 ± 0.0009
DeepID3 ¹²	0.9953 ± 0.0010
dkface ¹²	0.9551 ± 0.0013
Uni-Ute ¹²	0.9900 ± 0.0032
FaceNet ¹²	0.9963 ± 0.0009
Biad ¹²	0.9977 ± 0.0006
AuthenMetric ¹²	0.9977 ± 0.0009
MMDFR ¹²	0.9902 ± 0.0019
CW-DNA-1 ¹²	0.9950 ± 0.0022
Face++ ¹²	0.9967 ± 0.0007
JustMeTalk ¹²	0.9887 ± 0.0016

pose+shape+expression augmentation ¹²	0.9807 ± 0.0060
ColorReco ¹²	0.9940 ± 0.0022
Asaphus ¹²	0.9815 ± 0.0039
Daream ¹²	0.9968 ± 0.0009
Dahuia-Facemaque ¹²	0.9978 ± 0.0007
Skytop Gao ¹²	0.9630 ± 0.0023
CNN-3DMM estimation ¹²	0.9235 ± 0.0129
SamTech Facequest ¹²	0.9971 ± 0.0018
KY2 Robot ¹²	0.9895 ± 0.0020
THU CV-AI Lab ¹²	0.9973 ± 0.0008
di-biE ¹²	0.9938 ± 0.0027
Aureus ¹²	0.9920 ± 0.0030
YouTu Lab_Tencent ¹²	0.9960 ± 0.0023
Orion Star ¹²	0.9965 ± 0.0032
Yunfu WiseSight ¹²	0.9943 ± 0.0045
PingAn AI Lab ¹²	0.9980 ± 0.0016
Turing123 ¹²	0.9940 ± 0.0040
Design ¹²	0.9968 ± 0.0030
VisionLabs V2 ¹²	0.9978 ± 0.0007
Deepmark ¹²	0.9923 ± 0.0016
Force InfoSystems ¹²	0.9973 ± 0.0028
RoadSense ¹²	0.9982 ± 0.0007
CM-CV&AR ¹²	0.9963 ± 0.0039
sensingtech ¹²	0.9970 ± 0.0008
Glasseye ¹²	0.9983 ± 0.0018
icarvision ¹²	0.9977 ± 0.0030
Eason Electron ¹²	0.9983 ± 0.0006
younshifu ¹²	0.9975 ± 0.0006
RemarkFace ¹²	0.9963 ± 0.0030

○ 项目业务需求分析

4.1 项目需求概述

网易云课堂

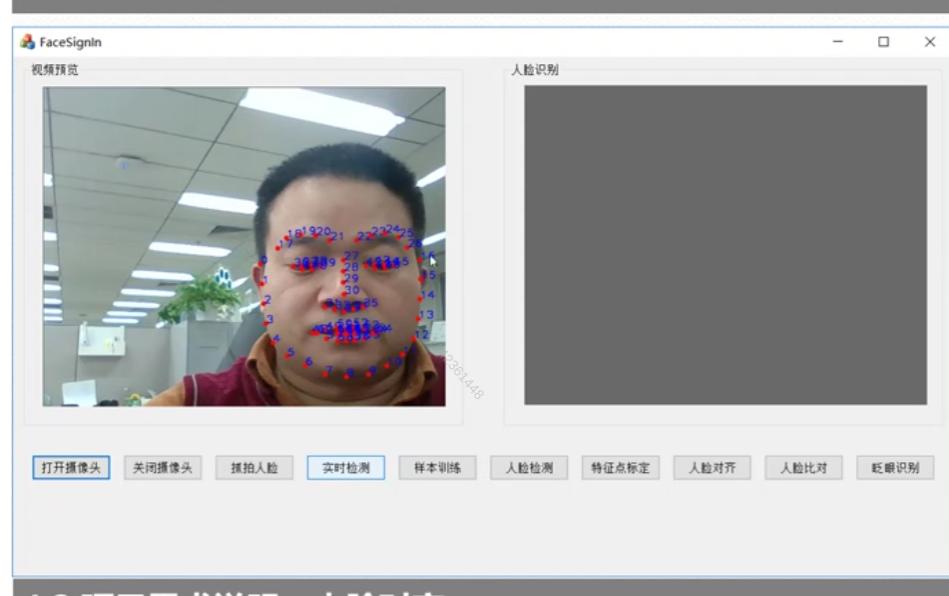


4.2 项目需求说明：人脸检测



- 人脸对齐：人脸特征点的标注（标注出特征点）->位置的矫正->特征向量的比对（特征点信息映射到128维向量，再判定欧式距离，如何在阈值之内就是同一个人）->活体检测（常见就是眨眼张嘴点头）

4.2 项目需求说明：人脸对齐



4.2 项目需求说明：人脸识别



- 项目业务流程分析

- 样本标注->模型训练->模型应用



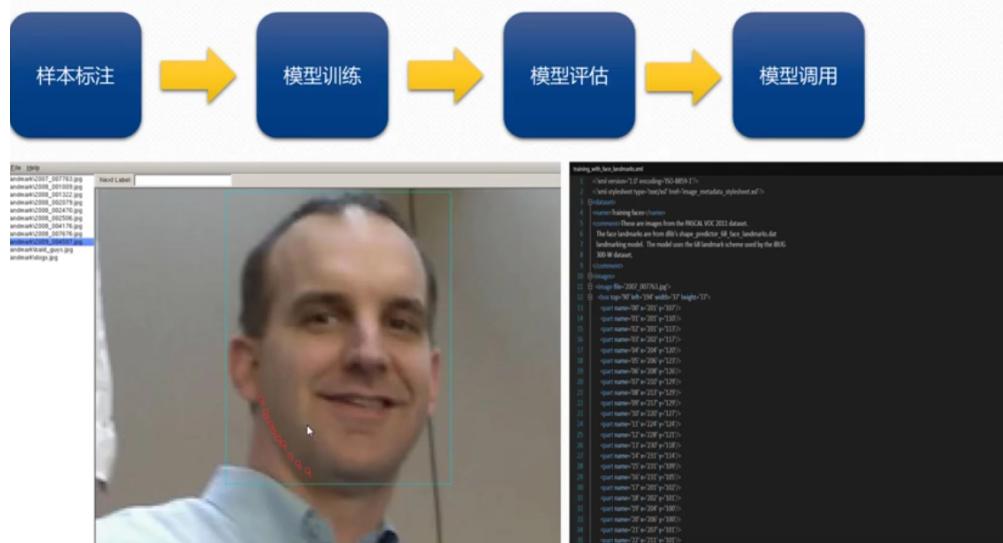
- 5.2 对应如下5.3



■ 5.3对应的核心过程（看图对应其中内容）

5.4 模型训练流程

网易云课堂



● 环境部署篇（技术框架）

- 项目开发环境概述

1.2 开发环境说明

软件环境	版本	备注
操作系统	Win10	操作系统
C++ IDE	Visual Studio 2015	应用开发
Python	3.6.5	模型训练
Dlib	19.16	算法框架
cuda	9.0	模型加速
opencv	3.4.1	图像框架

■ a

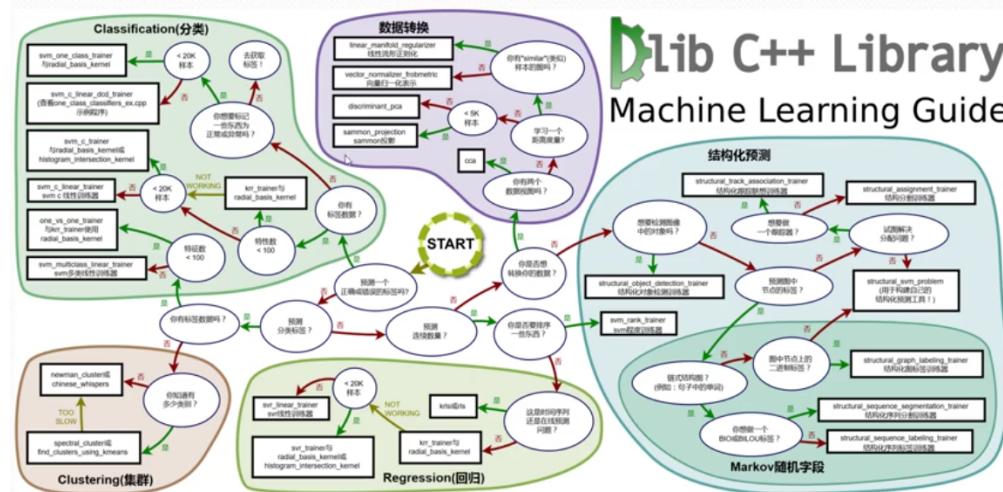
■

- Dlib框架源码编译

- 包含分类、集群、回归

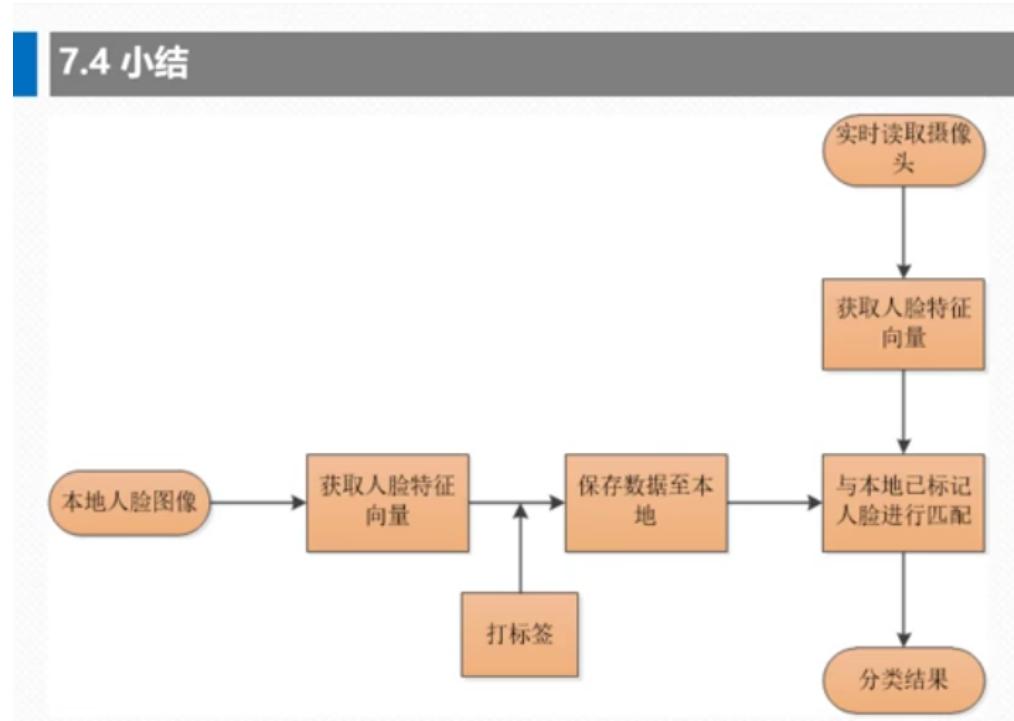
2.1 dlib概述

网易云课堂

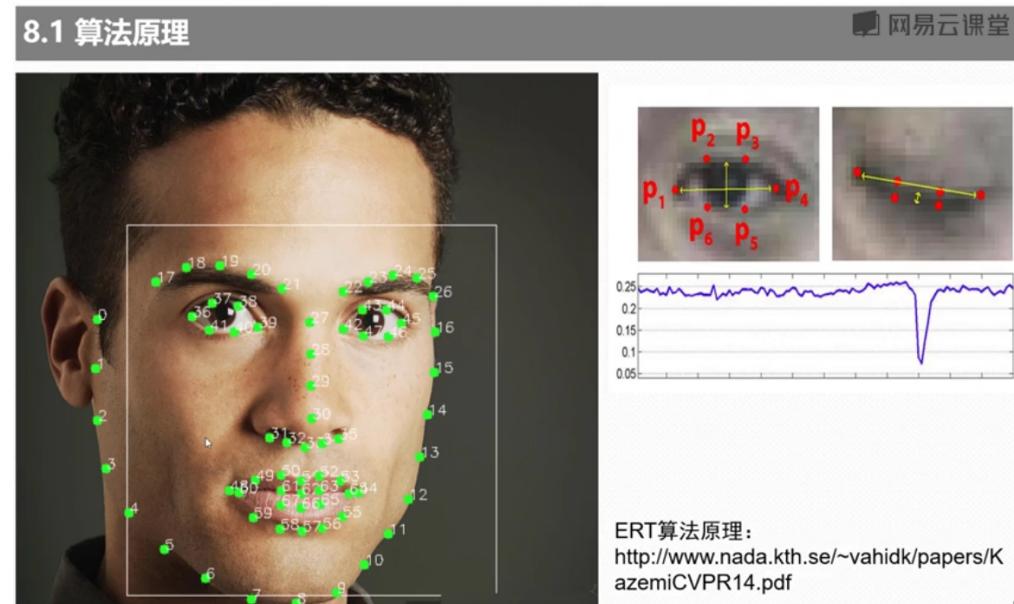


- 项目工程文件创建
- 项目开发环境配置之Cuda
- 项目开发环境配置之OpenCV

- 项目开发环境配置之Dlib
- 项目性能优化配置
- 程序设计篇（关键技术）
 - 实时视频采集程序设计
 - 实时图像抓拍程序设计
 - 实时人脸检测程序设计
 - 实时人脸特征点标定程序设计
 - 实时人脸特征点对齐程序设计
 - 实时目标跟踪程序设计
 - 实时人脸对比程序设计

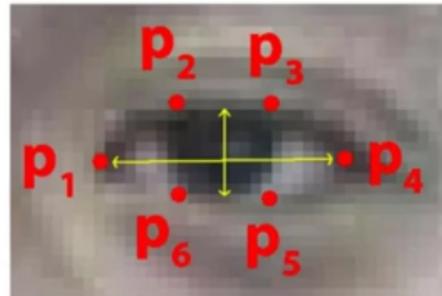


- 实时活体检测之眨眼识别



8.1 算法原理：EAR

每只眼睛由6个(x, y)坐标表示，从眼睛的左角开始，然后围绕该区域的其余部分顺时针显示：



基于这个描述，我们应该抓住重点：这些坐标的宽度和高度之间有一个关系。

Soukupová和Čech在其2016年的论文“使用面部标志实时眼睛眨眼检测”的工作，我们可以推导出反映这种关系的方程，称为眼睛纵横比 (EAR)：

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

图4：眼长宽比方程

其中 $p1, \dots, p6$ 是2D面部地标位置。

8.2 程序设计

```
833 //眨眼检测
834 void CFaceSignInDlg::OnBnClickedBtnFaceeye()
835 {
836     if (m_beye)
837     {
838         m_beye = 0;
839     }
840     else
841     {
842         m_beye = 1;
843     }
844 }
```

- 设置阈值，就是发生眨眼行为

8.2 程序设计

```
//眨眼行为检测:左眼-上下四个点
int y37 = shapes[0].part(37).y();
int y38 = shapes[0].part(38).y();
int y40 = shapes[0].part(40).y();
int y41 = shapes[0].part(41).y();
int x36 = shapes[0].part(36).x();
int x39 = shapes[0].part(39).x();

//1 / 2 * [(y41 + y40) - (y37 + y38)] / (x39 - x36)
//float flg = 1/2* [(y41 + y40) - (y37 + y38)]/(x39 - x36);
int y1 = abs(y37 - y41);
int y2 = abs(y38 - y40);
int x1 = abs(x39 - x36);
//长宽比
float flg = (y1 + y2) / (2.0 * x1);

CString str;
str.Format("EAR: %.2f", flg);
m_RegResult.SetWindowText(str);
}
```

```
// shapes[0].part(i).x(); //68个
// zhanzl:显示特征点的数字
putText(frame, to_string(i), cvPoint(shapes[0].part(i).x(), shapes[0].part(i).y()), C
    )
}
//眨眼检测
if (m_beye)
{
    //眨眼行为检测:左眼-上下四个点
    int y37 = shapes[0].part(37).y();
    int y38 = shapes[0].part(38).y();
    int y40 = shapes[0].part(40).y();
    int y41 = shapes[0].part(41).y();
    int x36 = shapes[0].part(36).x();
    int x39 = shapes[0].part(39).x();

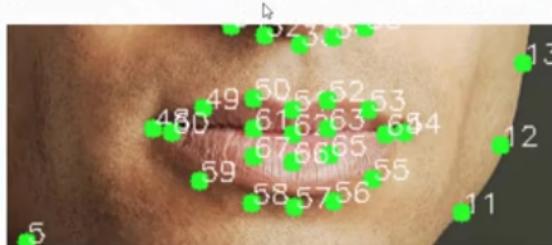
    //...
    int y1 = abs(y37 - y41);
    int y2 = abs(y38 - y40);
    int x1 = abs(x39 - x36);
    //长宽比
    float flg = (y1 + y2) / (2.0 * x1);    I

    CString str;
    str.Format("EAR: %.2f", flg);
    m_RegResult.SetWindowText(str);
}
//张嘴检测: Abs(Y50 - Y58) + Abs(Y52-56) / (2.0 * ABS(x48-x54))
if (m_bmouth)
{
    int y50 = shapes[0].part(50).y();
    I
```

- 实时活体检测之张嘴识别

9.1 算法原理

- 点位选择：50、52、58、56 纵坐标；48、54横坐标；



- 计算公式：长宽比

- $$\text{Abs}(Y_{50} - Y_{58}) + \text{Abs}(Y_{52} - Y_{56}) / (2.0 * \text{ABS}(x_{48} - x_{54}))$$

9.3 程序设计

```
811     void CFaceSignInDlg::OnBnClickedBtnFacemouse()
812     {
813         if (m_bmouth)
814         {
815             m_bmouth = 0;
816         }
817         else
818         {
819             m_bmouth = 1;
820         }
821     }
822
823
824     if (m_bmouth)
825     {
826         int y50 = shapes[0].part(50).y();
827         int y52 = shapes[0].part(52).y();
828         int y56 = shapes[0].part(56).y();
829         int y58 = shapes[0].part(58).y();
830
831         int x48 = shapes[0].part(48).x();
832         int x54 = shapes[0].part(54).x();
833
834         int y1 = abs(y50 - y58);
835         int y2 = abs(y52 - y56);
836         int x1 = abs(x48 - x54);
837
838         //长宽比
839         float flg = (y1 + y2) / (2.0 * x1);
840
841         CString str;
842         str.Format("Mouth EAR: %.2f", flg);
843         m_RegResult.SetWindowText(str);
844     }
```

```

if (_bmouth)
{
    int y50 = shapes[0].part(50).y();
    int y52 = shapes[0].part(52).y();
    int y56 = shapes[0].part(56).y();
    int y58 = shapes[0].part(58).y();

    int x48 = shapes[0].part(48).x();
    int x54 = shapes[0].part(54).x();

    int y1 = abs(y50 - y58);
    int y2 = abs(y52 - y56);
    int x1 = abs(x48 - x54);

    //长宽比
    float flg = (y1 + y2) / (2.0 * x1);

    CString str;
    str.Format("Mouth EAR: %.2f", flg);
    m_regResult.SetWindowText(str);
}

```

- 人脸聚类程序设计
- 模型训练篇（模型训练）
 - 人脸区域检测样本标注（imagelab源码编译和人脸检测标注方法）

1.2 ImgLab编译说明

```

README.txt
3 to train an object detector (e.g. a face detector) since it allows you to
4 easily create the needed training dataset.
5
6 You can compile imglab with the following commands:
7 cd lib/tools/imglab
8 mkdir build
9 cd build
10 cmake ..
11 cmake --build . --config Release
12 Note that you may need to install CMake (www.cmake.org) for this to work. On a
13 unix system you can also install imglab into /usr/local/bin by running
14 sudo make install
15 This will make running it more convenient.
16
17 Next, to use it, lets assume you have a folder of images called /tmp/images.
18 These images should contain examples of the objects you want to learn to
19 detect. You will use the imglab tool to label these objects. Do this by
20 typing the following command:
21 ./imglab -c mydataset.xml /tmp/images
22 This will create a file called mydataset.xml which simply lists the images in
23 /tmp/images. To add bounding boxes to the objects you run:
24 ./imglab mydataset.xml
25 and a window will appear showing all the images. You can use the up and down
26 arrow keys to cycle though the images and the mouse to label objects. In
27 particular, holding the shift key, left clicking, and dragging the mouse will
28 allow you to draw boxes around the objects you wish to detect.
29
30 Once you finish labeling objects go to the file menu, click save, and then
31 close the program. This will save the object boxes back to mydataset.xml. You
32 can verify this by opening the tool again with:
33 ./imglab mydataset.xml
34 and observing that the boxes are present.
35
36
37 imglab can do a few additional things. To see these run:

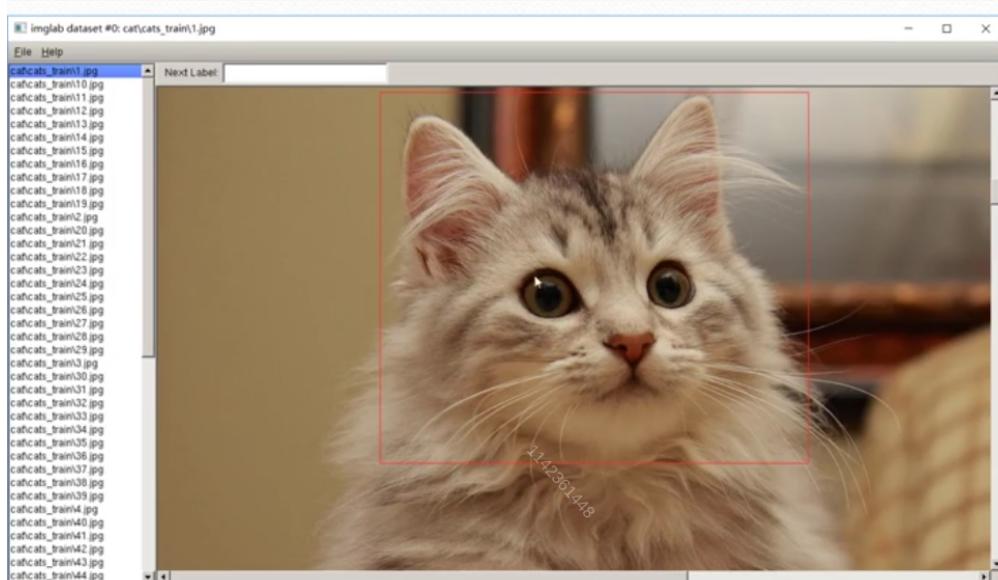
```

1.3 Imglab源码编译

1.4 ImgLab运行

- 创建记录标签的配置文件:
 - imglab.exe -c mydataset.xml cat//cats_train
 - 进行图像检测区域标注:
 - imglab.exe mydataset.xml

1.5 ImgLab标注方法：shift+鼠标左键



- ## ○ 人脸区域检测模型训练

2.2 程序设计：参数设置

```
1      # -*- coding: utf-8 -*-
2      import os
3      import sys
4      import glob
5      import dlib
6      import cv2
7
8      # options用于设置训练的参数和模式
9      options = dlib.simple_object_detector_training_options()
10     # Since faces are left/right symmetric we can tell the trainer to train a
11     # symmetric detector. This helps it get the most value out of the training
12     # data.
13     options.add_left_right_image_flips = True
14     # 支持向量机的C参数，通常默认取为5.自己适当更改参数以达到最好的效果
15     options.C = 5
16     # 线程数，你电脑有4核的话就填4
17     options.num_threads = 4
18     options.be_verbose = True
19
20     # 获取路径
21     current_path = os.getcwd()
22     train_folder = current_path + '/data/cats_train/'
23     test_folder = current_path + '/data/cats_test/'
24     train_xml_path = train_folder + 'cat.xml'
```

2.1 样本数据

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<?xml-stylesheet type='text/xsl' href='image_metadata_stylesheet.xsl'?>
<dataset>
<name>imglab dataset</name>
<comment>Created by imglab tool.</comment>
<images>
    <image file='1.jpg'>
        <box top='62' left='264' width='549' height='472'/>
    </image>
    <image file='10.jpg'>
        <box top='187' left='371' width='902' height='661'/>
    </image>
    <image file='11.jpg'>
        <box top='73' left='147' width='292' height='255'/>
    </image>
    <image file='12.jpg'>
        <box top='4' left='63' width='75' height='79'/>
        <box top='4' left='162' width='122' height='130' style="z-index: 1000;"/>
    </image>
    <image file='13.jpg'>
        <box top='45' left='221' width='325' height='375'/>
    </image>
    <image file='14.jpg'>
        <box top='6' left='209' width='680' height='678'/>
    </image>
    <image file='15.jpg'>
        <box top='200' left='1003' width='342' height='362'/>
    </image>
    <image file='16.jpg'>
        <box top='73' left='470' width='256' height='270'/>
    </image>
    <image file='17.jpg'>
        <box top='184' left='442' width='644' height='521'/>
```

2.2 程序设计：接口调用

```
pycode - [E:\project\faceid\pycode] - ..\mod_facedetect_train.py - PyCharm 2017.1.5
文件 (F) 编辑 (E) 视图 (V) 导航 (N) 代码 (C) 重构 (R) 运行 (U) 工具 (T) VCS (S) 窗口 (W) 帮助 (H)

21 current_path = os.getcwd()
22 train_folder = current_path + '/data/cats_train/'
23 test_folder = current_path + '/data/cats_test/'
24 train_xml_path = train_folder + 'cat.xml'
25 test_xml_path = test_folder + 'cats.xml'
26
27 print("training file path:" + train_xml_path)
28 # print(train_xml_path)
29 print("testing file path:" + test_xml_path)
30 # print(test_xml_path)
31
32 # 开始训练
33 print("start training:")
34 dlib.train_simple_object_detector(train_xml_path, "detector.svm", options)
35
36 print("") # Print blank line to create gap from previous output
37 print("Training accuracy: {}".format(
38     dlib.test_simple_object_detector(train_xml_path, "detector.svm")))
39
40 print("Testing accuracy: {}".format(
41     dlib.test_simple_object_detector(test_xml_path, "detector.svm")))
```

■ a

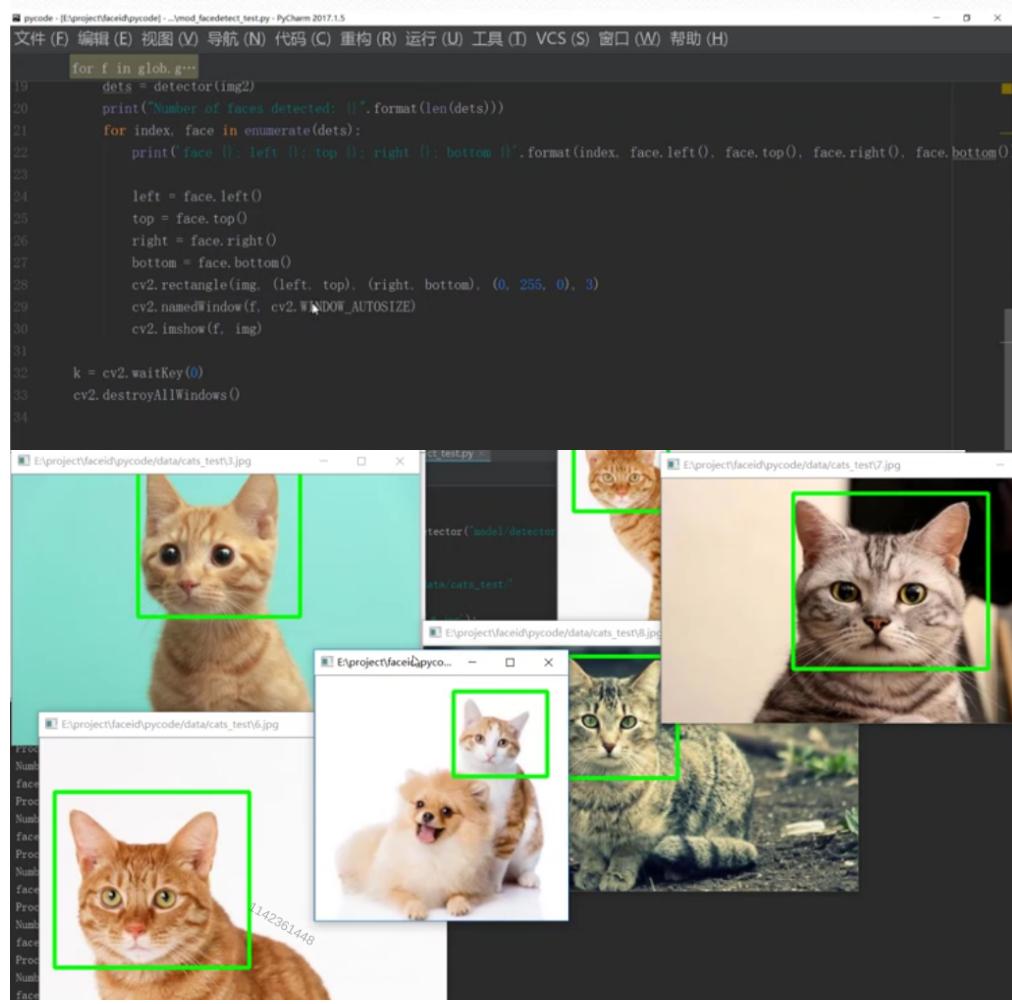
■

- 人脸区域检测模型测试

3.2 程序设计

```
for f in glob.glob('*.*'):
    for index, face in enumerate(faces):
        print('Processing file: {}'.format(f))
        img = cv2.imread(f, cv2.IMREAD_COLOR)
        b, g, r = cv2.split(img)
        img2 = cv2.merge([r, g, b])
        dets = detector(img2)
        print('Number of faces detected: {}'.format(len(dets)))
        for index, face in enumerate(dets):
            print('face [{}]: left [{}], top [{}], right [{}], bottom [{}].format(index, face.left(), face.top(), face.right(), face.bottom())
3
4         left = face.left()
5         top = face.top()
6         right = face.right()
```

3.2 程序设计



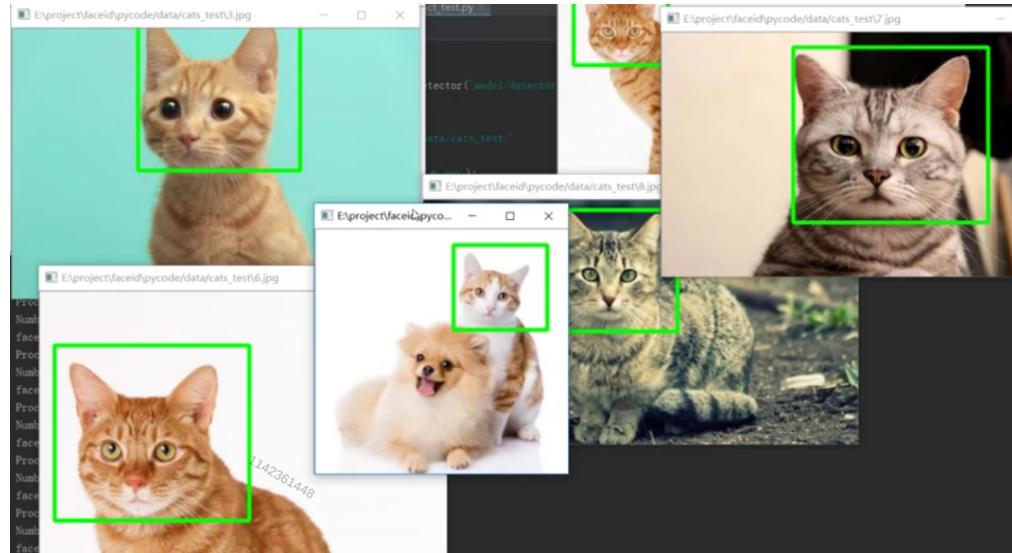
```
pycode - E:\project\faceid\pycode - Vmod_facedetect test.py - PyCharm 2017.1.5
文件 (F) 编辑 (E) 视图 (V) 导航 (N) 代码 (C) 重构 (R) 运行 (U) 工具 (T) VCS (S) 窗口 (W) 帮助 (H)
for f in glob.glob('...'):
    dets = detector(img2)
    print("Number of faces detected: {}".format(len(dets)))
    for index, face in enumerate(dets):
        print('face {}: left {}: top {}: right {}: bottom {}'.format(index, face.left(), face.top(), face.right(), face.bottom()))
        ...
        left = face.left()
        top = face.top()
        right = face.right()
        bottom = face.bottom()
        cv2.rectangle(img, (left, top), (right, bottom), (0, 255, 0), 3)
        cv2.namedWindow(f, cv2.WINDOW_AUTOSIZE)
        cv2.imshow(f, img)

    k = cv2.waitKey(0)
    cv2.destroyAllWindows()
```

The screenshot shows a PyCharm interface with a code editor containing a Python script for face detection. The script uses the `glob` module to iterate through files in a directory, loads an image using `cv2.imread`, initializes a face detector with `detector = cv2.CascadeClassifier('model/haarcascade_frontalface_default.xml')`, and then iterates through the detected faces. For each face, it prints its bounding box coordinates and draws a green rectangle around it using `cv2.rectangle`. The script then displays the image with the detected faces using `cv2.imshow` and waits for a key press with `cv2.waitKey(0)` before closing all windows with `cv2.destroyAllWindows()`. Below the code editor, there are four windows showing the results of the face detection on different images: a small orange cat, a larger orange cat, a small dog, and a large cat.

a

- 人脸特征点标注样本标注



3.1 数据标注方法

- • 创建记录标签的配置文件:
 - imglab.exe -c training_with_face_landmarks.xml landmark
- • 进行图像检测区域标注:
 - imglab.exe training_with_face_landmarks.xml

3.1 数据标注方法

- • 创建记录标签的配置文件:
 - imglab.exe -c training_with_face_landmarks.xml landmark
- • 进行图像检测区域标注:
 - imglab.exe training_with_face_landmarks.xml

3.1 数据标注方法

- • 创建记录标签的配置文件:
 - imglab.exe -c training_with_face_landmarks.xml landmark
- • 进行图像检测区域标注:
 - imglab.exe training_with_face_landmarks.xml

3.3 数据样本结构

training_with_face_landmarks.xml

```
1  <?xml version='1.0' encoding='ISO-8859-1'?>
2  <?xml-stylesheet type='text/xsl' href='image_metadata_stylesheet.xsl'?>
3  <dataset>
4    <name>Training faces</name>
5    <comment>These are images from the PASCAL VOC 2011 dataset.
6      The face landmarks are from dlib's shape_predictor_68_face_landmarks.dat
7      landmarking model. The model uses the 68 landmark scheme used by the iBUG
8      300-W dataset.
9    </comment>
10   <images>
11     <image file='2007_007763.jpg'>
12       <box top='90' left='194' width='37' height='37'>
13         <part name='00' x='201' y='107' />
14         <part name='01' x='201' y='110' />
15         <part name='02' x='201' y='113' />
16         <part name='03' x='202' y='117' />
17         <part name='04' x='204' y='120' />
18         <part name='05' x='206' y='123' />
19         <part name='06' x='208' y='126' />
20         <part name='07' x='210' y='129' />
21         <part name='08' x='213' y='129' />
22         <part name='09' x='217' y='129' />
23         <part name='10' x='220' y='127' />
24         <part name='11' x='224' y='124' />
25         <part name='12' x='228' y='121' />
26         <part name='13' x='230' y='118' />
27         <part name='14' x='231' y='114' />
28         <part name='15' x='231' y='109' />
29         <part name='16' x='231' y='105' />
30         <part name='17' x='201' y='102' />
31         <part name='18' x='202' y='101' />
32         <part name='19' x='204' y='100' />
33         <part name='20' x='206' y='100' />
34         <part name='21' x='207' y='101' />
35         <part name='22' x='211' y='101' />
```

3.5 更多数据样本-194

网易云课堂



官方地址：<http://www.ifp.illinois.edu/~vuongle2/helen/>

3.5 更多数据样本

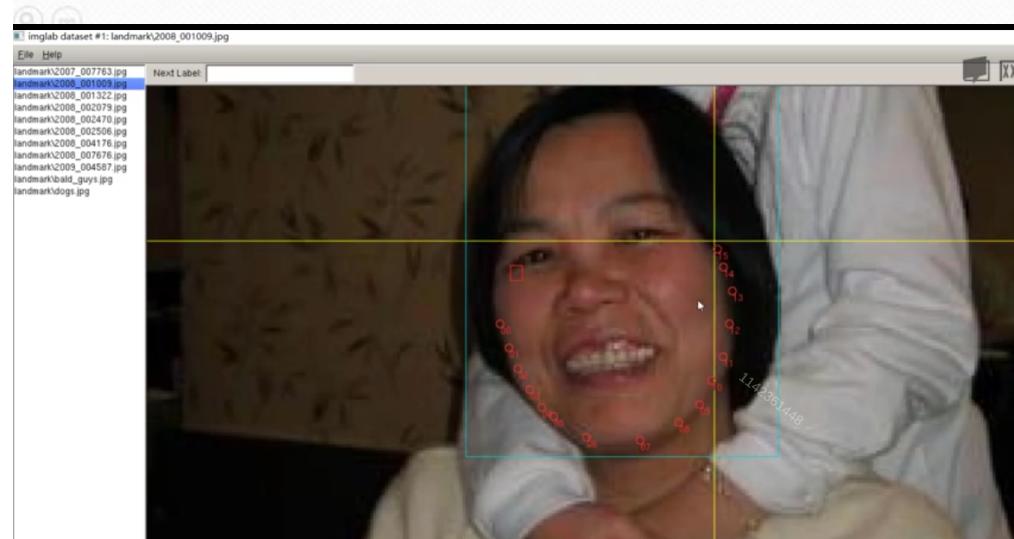
Download Link:

https://www.dropbox.com/s/jk98moqm8vopp5b/training_with_face_landmarks_2000.zip?dl=0

Steps:

1. Download Train images - part 1, Train images - part 2, Train images - part 3 & Train images - part 4 from <http://www.ifp.illinois.edu/~vuongle2/helen/>. (each part has 500 images, so total of 2000 images)
2. Place all the images & the training_with_face_landmarks.xml in a common folder (Ex: Folder name "train").
3. In the project train_shape_predictor_ex.cpp, comment out these sections/lines
 - a. Reference to testing_with_face_landmarks.xml and its usage b. trainer set_oversampling_amount(100) c. trainer set_nu(0.05) d. trainer set_tree_depth(3)
4. Set command line argument as ./train
5. Run the project
6. It will generate sp.dat of size approx 178 MB (took 2 hours approx in my laptop)
7. Use this sp.dat file in face_landmark_detection_ex project Command line Ex: sp.dat Example_image.jpg
8. In face_landmark_detection_ex.cpp, a. In a loop of 194, call draw_solid_circle function b. using save_png function save the landmark output as a file.

官方地址: <http://www.ifp.illinois.edu/~vuongle2/helen/>



a

- 人脸特征点标注模型训练

5.1 训练样本

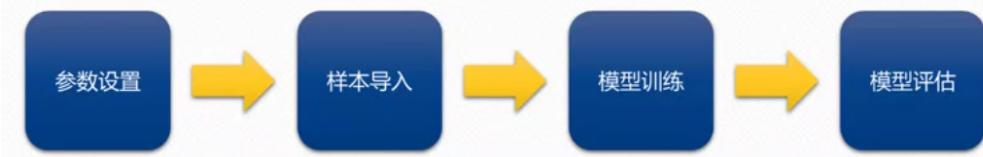
电脑 > project (E:) > project > faceid > pycode > data > landmark



5.1 训练样本

```
training_with_face_landmarks.xml
1  <?xml version='1.0' encoding='ISO-8859-1'?>
2  <?xmlstylesheet type='text/xsl' href='image_metadata_stylesheet.xsl'?>
3  <dataset>
4      <name>Training faces</name>
5      <comment>These are images from the PASCAL VOC 2011 dataset.
6          The face landmarks are from dlib's shape_predictor_68_face_landmarks.dat
7          landmarking model. The model uses the 68 landmark scheme used by the iBUG
8          300-W dataset.
9      </comment>
10     <images>
11         <image file='2007_007763.jpg'>
12             <box top='90' left='194' width='37' height='37'>
13                 <part name='00' x='201' y='107' />
14                 <part name='01' x='201' y='110' />
15                 <part name='02' x='201' y='113' />
16                 <part name='03' x='202' y='117' />
17                 <part name='04' x='204' y='120' />
18                 <part name='05' x='206' y='123' />
19                 <part name='06' x='208' y='126' />
20                 <part name='07' x='210' y='129' />
21                 <part name='08' x='213' y='129' />
22                 <part name='09' x='217' y='129' />
23                 <part name='10' x='220' y='127' />
24                 <part name='11' x='224' y='124' />
25                 <part name='12' x='228' y='121' />
26                 <part name='13' x='230' y='118' />
27                 <part name='14' x='231' y='114' />
28                 <part name='15' x='231' y='109' />
29                 <part name='16' x='231' y='105' />
30                 <part name='17' x='201' y='102' />
31                 <part name='18' x='202' y='101' />
32                 <part name='19' x='204' y='100' />
33                 <part name='20' x='206' y='100' />
34                 <part name='21' x='207' y='101' />
35                 <part name='22' x='211' y='101' />
```

5.2 程序逻辑



5.3 程序设计

网易云课

```
# 01-参数设置
options = dlib.shape_predictor_training_options()
options.oversampling_amount = 300
options.nu = 0.05
options.tree_depth = 2
options.be_verbose = True

# 02-导入打好了标签的xml文件
training_xml_path = os.path.join(faces_path, "training_with_face_landmarks.xml")
# 03-进行训练，训练好的模型将保存为predictor.dat
print(training_xml_path)
dlib.train_shape_predictor(training_xml_path, "model/predictor.dat", options)
# 04-打印在训练集中的准确率
print
"\nTraining accuracy: [0]".format(dlib.test_shape_predictor(training_xml_path, "model/predictor.dat"))
```

```
[ode] -\vmod_facelandmark_train.py - PyCharm 2017.1.5
) 代码 (G 重构 (R 执行 (E 工具 (T VCS (S 窗口 (W 帮助 (H)
K_train.py mod_facelandmark_train.py - PyCharm 2017.1.5
mod_facelandmark_train.py < mod_facelandmark_train.py <
pycode

13     current_path = os.getcwd()
14     faces_path = current_path + '/data/landmark'
15
16     # 01-参数设置
17     options = dlib.shape_predictor_training_options()
18     options.oversampling_amount = 300
19     options.nu = 0.05
20     options.tree_depth = 2
21     options.be_verbose = True
22
23     # 02-导入打好了标签的xml文件
24     training_xml_path = os.path.join(faces_path, "training_with_face_landmarks.xml")
25     # 03-进行训练，训练好的模型将保存为predictor.dat
26     print(training_xml_path)
27     dlib.train_shape_predictor(training_xml_path, "model/predictor.dat", options)
28     # 04-打印在训练集中的准确率
29     print
30     "\nTraining accuracy: ({})".format(dlib.test_shape_predictor(training_xml_path, "model/predictor.dat"))
31
32     # 05-导入测试集的xml文件
33     testing_xml_path = os.path.join(faces_path, "testing_with_face_landmarks.xml")
34     # 打印在测试集中的准确率
35     print
36     "\nTesting accuracy: ({})".format(dlib.test_shape_predictor(testing_xml_path, "model/predictor.dat"))
37
```

5.3 程序执行

The screenshot shows the PyCharm IDE interface with the following details:

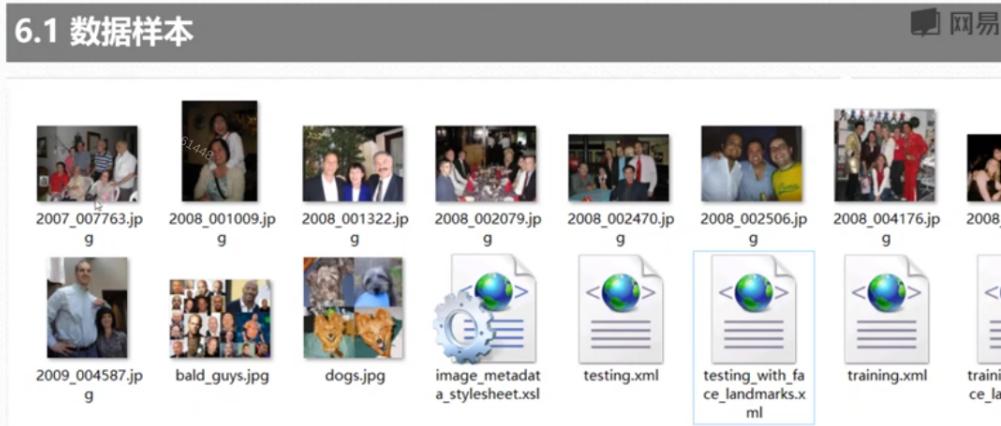
- Project Structure:** The project is named "pycode" and contains a "data" folder with subfolders "data", "face", "face_cascade", "eye_cascade", and "model".
- Code Editor:** The file "mod_faceLandmark_train.py" is open, showing Python code for training a face landmark detector. It imports cv2, numpy, and dlib, and defines paths for current and faces. It sets training options like shape predictor, overampling amount, and tree depth.
- Run Tab:** The "mod_faceLandmark_train" configuration is selected, and the command "D:\Program Files\Python36\python.exe E:/project/FaceID/pycode/mod_faceLandmark_train.py" is displayed.
- Output Tab:** The terminal output shows the execution of the script, including training with cascade depths 10 and 2, and fitting trees with a depth of 3.

共享查看

此电脑 > project (E) > project > faceid > pycode >

名称	修改日期	类型	大小
.idea	2019/3/4 8:33	文件夹	
data	2019/3/3 6:47	文件夹	
model	2019/3/3 9:35	文件夹	
eye_actiion.py	2019/2/26 8:13	JetBrains PyCharm	5 KB
mod_facedetect_test.py	2019/2/26 8:25	JetBrains PyCharm	1 KB
mod_facedetect_train.py	2019/2/26 8:25	JetBrains PyCharm	2 KB
mod_facelandmark_test.py	2019/3/3 9:41	JetBrains PyCharm	2 KB
mod_facelandmark_train.py	2019/3/3 9:20	JetBrains PyCharm	2 KB

- 人脸特征点标注模型测试



6.2 模型文件

电脑 > project (E) > project > faceid > pycode > model

名称	修改日期	类型	大小
detector.svm	2019/2/26 8:16	SVM 文件	48 KB
predictor.dat	2019/3/3 9:15	DAT 文件	16,217 KB
shape_predictor_68_face_landmarks.dat	2018/9/12 20:32	DAT 文件	97,358 KB



6.4 程序设计

```
import cv2
import dlib
import glob

# 01-路径设置
current_path = os.getcwd()
faces_path = current_path + '/data/landmark'

# 模型加载
predictor = dlib.shape_predictor("model/predictor.dat")
detector = dlib.get_frontal_face_detector()
print("Showing detections and predictions on the images in the faces folder...")
for f in glob.glob(os.path.join(faces_path, "*.jpg")):
    print("Processing file: {}".format(f))
    img = cv2.imread(f)
    img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # 02-人脸区域检测
    dets = detector(img2, 1)
    print("Number of faces detected: {}".format(len(dets)))
```

6.4 程序设计

网易云课堂

```
for index, face in enumerate(dets):
    print('face [{}]: left [{}]; top [{}]; right [{}]; bottom [{}].format(index, face.left(), face.top(), face.right(), face.bottom()))'
    ...
    shape = predictor(img, face)
    # print(shape)
    # print(shape.num_parts)
    for index, pt in enumerate(shape.parts()):
        print('Part [{}]: [{}].format(index, pt)')
        pt_pos = (pt.x, pt.y)
        # 04-特征点绘制
        cv2.circle(img, pt_pos, 2, (255, 0, 0), 1)
    # print(type(pt))
    # print("Part 0: {}, Part 1: {} ...".format(shape.part(0), shape.part(1)))
    cv2.namedWindow(f, cv2.WINDOW_AUTOSIZE)
    cv2.imshow(f, img)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- 偏了，需要优化训练样本

6.5 结果输出

网易云课堂



```
1 ...
2
3 import os
4 import cv2
5 import dlib
6 import glob
7
8 # 01-路径设置
9 current_path = os.getcwd()
10 faces_path = current_path + '/data/landmark'
11
12 # 模型加载
13 predictor = dlib.shape_predictor("model/predictor.dat")
14 detector = dlib.get_frontal_face_detector()
15 print("Showing detections and predictions on the faces in the faces folder...")
16 for f in glob.glob(os.path.join(faces_path, "*_*.jpg")):
17     print("Processing file: {}".format(f))
18     img = cv2.imread(f)
19     img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
20     # 02-人脸区域检测
21     dets = detector(img2, 1)
22     print("Number of faces detected: {}".format(len(dets)))
23     for index, face in enumerate(dets):
24         print('face {}: left {}; top {}; right {}; bottom {}'.format(index, face.left(), face.top(), face.right(), face.bottom()))
25
26     ...
27     shape = predictor(img, face)
28     # print(shape)
29     # print(shape.num_parts)
30     for index, pt in enumerate(shape.parts()):
31         print('Part {}: {}'.format(index, pt))
32         pt_pos = (pt.x, pt.y)
33         # 04-特征点绘制
34         cv2.circle(img, pt_pos, 2, (255, 0, 0), 1)
35     # print(type(pt))
36     # print("Part 0: {}, Part 1: {}".format(shape.part(0), shape.part(1)))
37     cv2.namedWindow(f, cv2.WINDOW_AUTOSIZE)
38     cv2.imshow(f, img)
39
40     cv2.waitKey(0)
41     cv2.destroyAllWindows()
```

- 往往训练样本不够，跟源码无关



- 人脸注册