

K210 实现人脸关键点检测教程

第一章：K210 芯片介绍

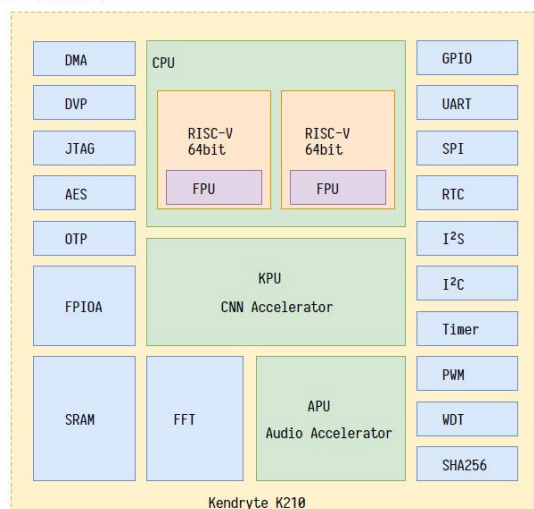
K210 是嘉楠科技 2018 年 9 月开始商用基于 RISC—V 架构和神经网络加速器 KPU 的双核 64 位 CPU，采用台积电 28nm 制程实现运行频率在 400MHz，算力达到 1TOPS。

K210 芯片主要是基于 SiFive 和 UC Berkeley 开源的 Rocket Core 实现 ARM Cortex-A5 性能。该芯片下一代 K510 芯片会支持 Linux、GCC、Glibc。K510 同样采用 28nm 工艺但在芯片架构方面使用 RSIC-V 的多核异构处理架构。同时改良 KPU 加入全新设计计算模块 GNNE。第三代芯片计划采用 12nm 工艺实现 5G 云计算与边缘计算。目前该芯片用在计算机视觉边缘计算、语音识别、STEAM 教育方面。

同时 A1166 是嘉楠科技在 2013 年基于 ASIC 研发 16nm、7nm 的区块链和比特币矿机芯片。

该芯片的主要开发板厂家有常见的 Sipeed 和 PyAi-K210 及 BPI-K210。

1.2 系统架构



K210 包含 RISC-V 64 位双核 CPU，每个核心内置独立 FPU。K210 的核心功能是机器视觉与听觉，其包含用于计算卷积人工神经网络的 KPU 与用于处理麦克风阵列输入的 APU。同时 K210 具备快速傅里叶变换加速器，可以进行高性能复数 FFT 计算。因此对于大多数机器学习算法，K210 具备高性能处理能力。

K210 内嵌 AES 与 SHA256 算法加速器, 为用户提供基本安全功能。

K210 拥有高性能、低功耗的 SRAM，以及功能强大的 DMA，在数据吞吐能力方面性能优异。

K210 具备丰富的外设单元, 分别是: DVP、JTAG、OTP、FPIOA、GPIO、UART、SPI、RTC、I²S、I²C、WDT、Timer 与 PWM, 可满足海量应用场景。

概述

Kendryte K210 是集成机器视觉与机器听觉能力的系统级芯片 (SoC)。使用台积电 (TSMC) 超低功耗的 28 纳米先进制程, 具有双核 64 位处理器, 拥有较好的功耗性能, 稳定性与可靠性。该方案力求零门槛开发, 可在最短时效部署于用户的产品中, 赋予产品人工智能。

Kendryte K210 定位于 AI 与 IoT 市场的 SoC, 同时是使用非常方便的 MCU。

Kendryte 中文含义为勘智, 而勘智取自勘物探智。这颗芯片主要应用领域为物联网领域, 在物联网领域进行开发, 因此为勘物; 这颗芯片主要提供的是人工智能解决方案, 在人工智能领域探索, 因此为探智。

- 具备机器视觉能力
- 具备机器听觉能力
- 更好的低功耗视觉处理速度与准确率
- 具备卷积神经网络硬件加速器 KPU, 可高性能进行卷积神经网络运算
- TSMC 28nm 先进制程, 温度范围-40°C 到 125°C, 稳定可靠
- 支持固件加密, 难以使用普通方法破解
- 独特的可编程 IO 阵列, 使产品设计更加灵活
- 低电压, 与相同处理能力的系统相比具有更低功耗
- 3.3V/1.8V 双电压支持, 无需电平转换, 节约成本

https://blog.csdn.net/gmz_syy

1.1 AI 解决方案

1.1.1 机器视觉

Kendryte K210 具备机器视觉能力, 是零门槛机器视觉嵌入式解决方案。它可以在低功耗情况下进行卷积神经网络计算。

该芯片可以实现以下机器视觉能力:

- 基于卷积神经网络的一般目标检测
- 基于卷积神经网络的图像分类任务
- 人脸检测和人脸识别
- 实时获取被检测目标的大小与坐标
- 实时获取被检测目标的种类

https://blog.csdn.net/gmz_syy

1.1.2 机器听觉

Kendryte K210 具备机器听觉能力。芯片上自带高性能麦克风阵列音频处理器, 可以进行实时声源定向与波束形成。

该芯片可以实现以下机器听觉能力:

- 声源定向
- 声场成像
- 波束形成
- 语音唤醒
- 语音识别

https://blog.csdn.net/gmz_syy

1.1.3 视觉/听觉混合解决方案

Kendryte K210 可结合机器视觉和机器听觉能力，提供更强大的功能。一方面，在应用中既可以通过声源定位和声场成像辅助机器视觉对目标的跟踪，又可以通过一般目标检测获得目标的方向后辅助机器听觉对该方位进行波束形成。另一方面，可以通过摄像头传来的图像获得人的方向后，使得麦克风阵列通过波束形成指向该人。同时也可以根据麦克风阵列确定一个说话人的方向，转动摄像头指向该人。

https://blog.csdn.net/gmq_xyy

Sipeed maix dock k210 操作教程：

第一步：下载烧录工具 Kflash_gui

https://dl.sipeed.com/MAIX/tools/kflash_gui/kflash_gui_v1.6.5

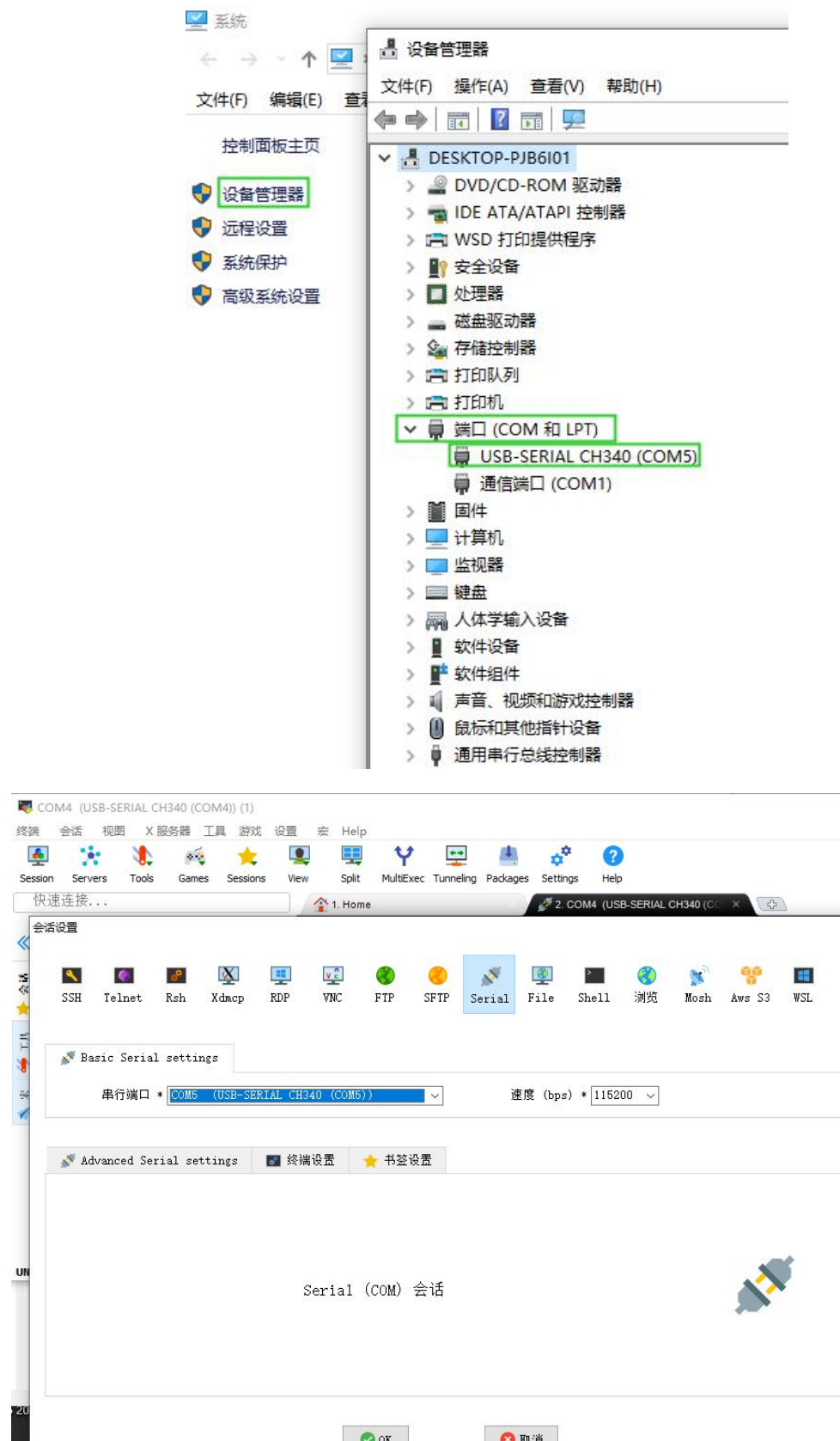
https://github.com/sipeed/kflash_gui/releases/tag/v1.5.3

第二步：下载烧录驱动更新文件 ken_gen.bin

第三步：连接好板卡使用烧录工具烧录驱动文件



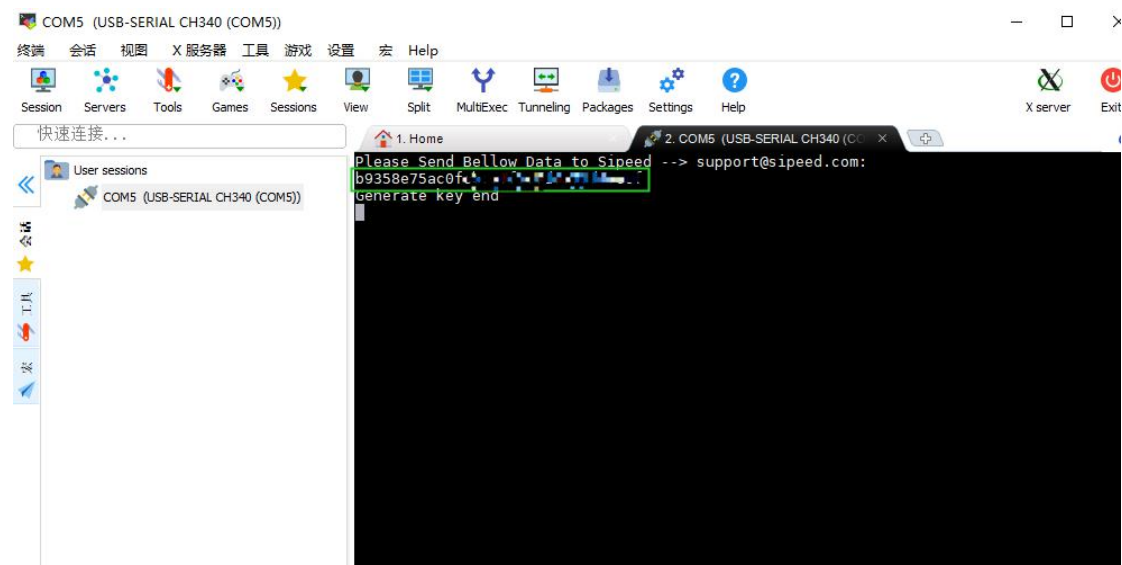
第四步：安装 COM 端口访问工具获取端口机器码



在此处就是获取的机器码

如图无法获取按住板上蓝色的灯（复位键）后还没法获取时候重新烧录一下 ken_en.bin

文件，该文件类似一个 key 验证码然后获取机器码原理是加密狗原理。



第五步：获取官方测试模型

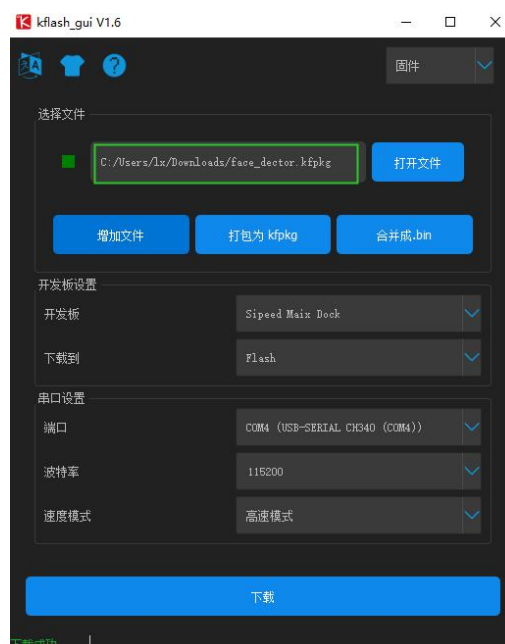
<https://www.maixhub.com/index.php/index/user/download/id/22>

输出机器码验证或者在 MaixHub 网站找到自己想测试的模型文件夹 kfpkg 文件解压后如下：

FD_b9358e75ac0fcbca1f5c2d7a06d4e...	2020/8/28 1:43	SMODEL 文件
FE_mbv1_0.5_b9358e75ac0fcbca1f5c2...	2020/8/28 1:44	SMODEL 文件
flash-list	2020/8/28 1:44	JSON 源文件
KP_chwise_b9358e75ac0fcbca1f5c2d7...	2020/8/28 1:43	SMODEL 文件
maixpy_face_ide.bin	2019/11/29 17:36	BIN 文件

根据以上文件的文件名称可以看出本次下载的为三个权重文件一个配置文件 json, 一个简化版的 maixpy 的 bin 文件。

第六步：使用烧录工具烧录模型驱动程序 kfpkg 包















第七步：安装程序 IDE 开始测试

<https://dl.sipeed.com/MAIX/MaixPy/ide/v0.2.5>

🏠 / MAIX / MaixPy / ide / v0.2.5








Show entries Search:

#	Name	Size	Modified	Download Count	MD5 File
6	 readme.txt	2.00KB	4 weeks ago	0	
5	 maixpy-ide-mac-0.2.5_2.dmg	102.94MB	4 weeks ago	0	
3	 maixpy-ide-windows-0.2.5.exe	85.50MB	4 weeks ago	0	
4	 maixpy-ide-windows-0.2.5-installer-archive.7z	64.90MB	4 weeks ago	0	
2	 maixpy-ide-linux-x86_64-0.2.5.run	81.12MB	4 weeks ago	0	
1	 maixpy-ide-linux-x86_64-0.2.5-installer-archive.7z	56.45MB	4 weeks ago	0	

这里还可以下载一个模型

<https://github.com/sipeed/MaixPy/releases>

▼ Assets 7

	elf.7z	7.4 MB
	face_model_at_0x300000.kfpkg	329 KB
	maixpy_v0.3.2_full.bin	2.09 MB
	maixpy_v0.3.2_minimum.bin	813 KB
	maixpy_v0.3.2_no_lvgl.bin	1.52 MB
	Source code (zip)	
	Source code (tar.gz)	

本教程使用的模型和代码是通过注册登录获取

<https://www.maixhub.com/index.php/index/user/download/id/22>

 首页 官方网站 模型训练 在线编译 发布模型

Eric

请填写32位机器码

选择机器码 ▼

请填写32位机器码

点我获取机器码

提交

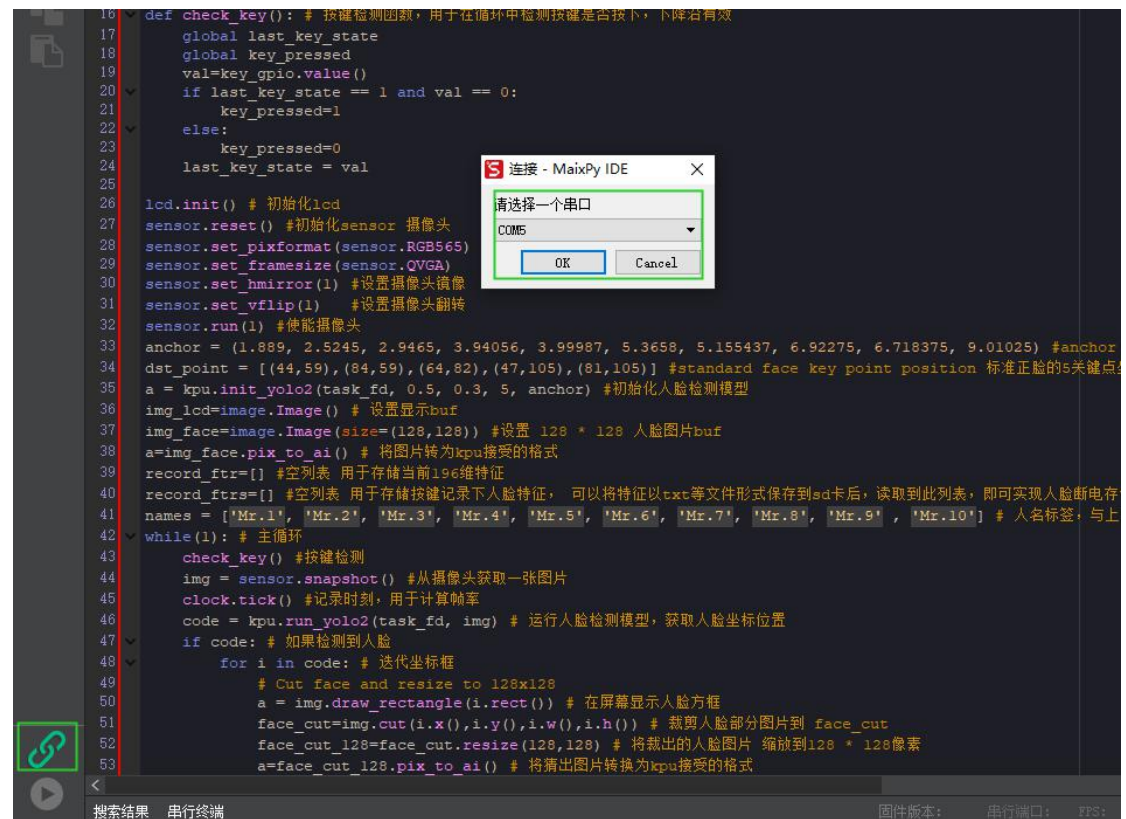
程序运行代码下载：

https://github.com/sipeed/MaixPy_scripts/blob/master/machine_vision/demo_face_recognition.py

按 boot 键可以录入人脸，录入人脸后会按顺序分配名字，识别到后会显示出来

第八步：安装板卡 IDE 运行代码

将下载好的代码读入 IDE 然后选择



```
# -*- coding: utf-8 -*-
# @Time      : 2020/8/28 16:02
# @Author    : Jungang An!
# @FileName  : facepoint.py
# @Software  : PyCharm

import sensor, image, lcd # import 相关库
import KPU as kpu
import time
from Maix import FPIOA, GPIO

task_fd = kpu.load(0x200000) # 从 flash 0x200000 加载人脸检测模型
task_ld = kpu.load(0x300000) # 从 flash 0x300000 加载人脸五点关键点检测模型
task_fe = kpu.load(0x400000) # 从 flash 0x400000 加载人脸 196 维特征值模型
clock = time.clock() # 初始化系统时钟，计算帧率
```

```

key_pin=16 # 设置按键引脚 FPIO16
fpioa = FPIOA()
fpioa.set_function(key_pin,FPIOA.GPIO7)
key_gpio=GPIO(GPIO.GPIO7,GPIO.IN)
last_key_state=1
key_pressed=0 # 初始化按键引脚 分配 GPIO7 到 FPIO16
def check_key(): # 按键检测函数,用于在循环中检测按键是否按下,下降沿有效
    global last_key_state
    global key_pressed
    val=key_gpio.value()
    if last_key_state == 1 and val == 0:
        key_pressed=1
    else:
        key_pressed=0
    last_key_state = val

lcd.init() # 初始化 lcd
sensor.reset() #初始化 sensor 摄像头
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_hmirror(1) #设置摄像头镜像
sensor.set_vflip(1) #设置摄像头翻转
sensor.run(1) #使能摄像头
anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437,
6.92275, 6.718375, 9.01025) #anchor for face detect 用于人脸检测的 Anchor
dst_point = [(44,59),(84,59),(64,82),(47,105),(81,105)] #standard face
key point position 标准正脸的 5 关键点坐标 分别为 左眼 右眼 鼻子 左嘴角 右嘴角
a = kpu.init_yolo2(task_fd, 0.5, 0.3, 5, anchor) #初始化人脸检测模型
img_lcd=image.Image() # 设置显示 buf
img_face=image.Image(size=(128,128)) #设置 128 * 128 人脸图片 buf
a=img_face.pix_to_ai() # 将图片转为 kpu 接受的格式
record_ftr=[] #空列表 用于存储当前 196 维特征
record_ftrs=[] #空列表 用于存储按键记录下人脸特征, 可以将特征以 txt 等文件形式保
存到 sd 卡后, 读取到此列表, 即可实现人脸断电存储。
names = ['Mr.1', 'Mr.2', 'Mr.3', 'Mr.4', 'Mr.5', 'Mr.6', 'Mr.7', 'Mr.8',
'Mr.9', 'Mr.10'] # 人名标签, 与上面列表特征值一一对应。
while(1): # 主循环
    check_key() #按键检测
    img = sensor.snapshot() #从摄像头获取一张图片
    clock.tick() #记录时刻,用于计算帧率
    code = kpu.run_yolo2(task_fd, img) # 运行人脸检测模型, 获取人脸坐标位置
    if code: # 如果检测到人脸
        for i in code: # 迭代坐标框
            # Cut face and resize to 128x128

```



```

a = img.draw_rectangle(i.rect()) # 在屏幕显示人脸方框
face_cut=img.cut(i.x(),i.y(),i.w(),i.h()) # 裁剪人脸部分图片到
face_cut

face_cut_128=face_cut.resize(128,128) # 将裁出的人脸图片 缩放到
128 * 128 像素

a=face_cut_128.pix_to_ai() # 将猜出图片转换为 kpu 接受的格式
#a = img.draw_image(face_cut_128, (0,0))
# Landmark for face 5 points
fmap = kpu.forward(task_ld, face_cut_128) # 运行人脸 5 点关键点
检测模型

plist=fmap[:] # 获取关键点预测结果
le=(i.x()+int(plist[0]*i.w() - 10),
i.y()+int(plist[1]*i.h())) # 计算左眼位置, 这里在 w 方向-10 用来补偿模型转换
带来的精度损失

re=(i.x()+int(plist[2]*i.w()), i.y()+int(plist[3]*i.h())) #
计算右眼位置

nose=(i.x()+int(plist[4]*i.w()), i.y()+int(plist[5]*i.h()))
#计算鼻子位置

lm=(i.x()+int(plist[6]*i.w()), i.y()+int(plist[7]*i.h())) #
计算左嘴角位置

rm=(i.x()+int(plist[8]*i.w()), i.y()+int(plist[9]*i.h())) #
右嘴角位置

a = img.draw_circle(le[0], le[1], 4)
a = img.draw_circle(re[0], re[1], 4)
a = img.draw_circle(nose[0], nose[1], 4)
a = img.draw_circle(lm[0], lm[1], 4)
a = img.draw_circle(rm[0], rm[1], 4) # 在相应位置处画小圆圈
# align face to standard position
src_point = [le, re, nose, lm, rm] # 图片中 5 坐标的位置
T=image.get_affine_transform(src_point, dst_point) # 根据获得
的 5 点坐标与标准正脸坐标获取仿射变换矩阵

a=image.warp_affine_ai(img, img_face, T) #对原始图片人脸图片进
行仿射变换, 变换为正脸图像

a=img_face.ai_to_pix() # 将正脸图像转为 kpu 格式
#a = img.draw_image(img_face, (128,0))
del(face_cut_128) # 释放裁剪人脸部分图片
# calculate face feature vector

fmap = kpu.forward(task_fe, img_face) # 计算正脸图片的 196 维特
征值

feature=kpu.face_encode(fmap[:]) #获取计算结果
reg_flag = False
scores = [] # 存储特征比对分数
for j in range(len(record_ftrs)): #迭代已存特征值
    score = kpu.face_compare(record_ftrs[j], feature) #计算当

```

前人脸特征值与已存特征值的分数

```
scores.append(score) #添加分数总表
max_score = 0
index = 0
for k in range(len(scores)): #迭代所有比对分数，找到最大分数和索引
    if max_score < scores[k]:
        max_score = scores[k]
        index = k
    if max_score > 85: # 如果最大分数大于 85， 可以被认定为同一个人
        a = img.draw_string(i.x(),i.y(), ("%s :%2.1f" %
(names[index], max_score)), color=(0,255,0),scale=2) # 显示人名 与 分数
    else:
        a = img.draw_string(i.x(),i.y(), ("X:%2.1f" % (max_score)),
color=(255,0,0),scale=2) #显示未知 与 分数
    if key_pressed == 1: #如果检测到按键
        key_pressed = 0 #重置按键状态
        record_ftr = feature
        record_ftrs.append(record_ftr) #将当前特征添加到已知特征列表
    break
fps =clock.fps() #计算帧率
print("%2.1f fps"%fps) #打印帧率
a = lcd.display(img) #刷屏显示
#kpu.memtest()
```

