

5.6 人脸检测

一、实验目的(以 OV2640 为例)

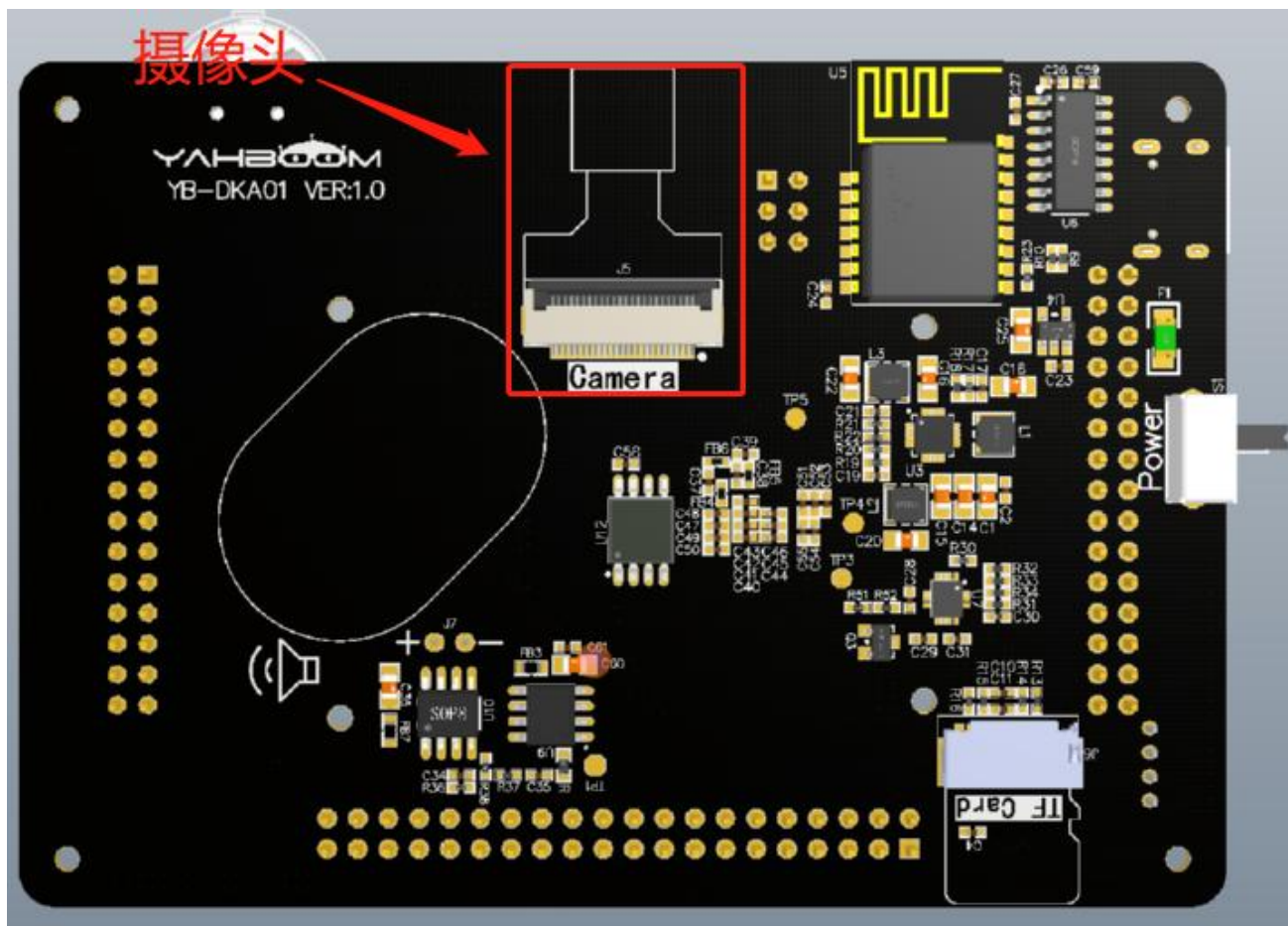
本节课主要学习 K210 如何人脸检测，然后通过 LCD 显示屏实时圈住人脸。

二、实验准备

1. 实验元件

OV2640 摄像头/OV9655 摄像头/GC2145 摄像头

LCD 显示屏



2. 硬件连接

K210 开发板出厂默认已经安装好摄像头和显示器，只需要使用 type-C 数据线连接 K210 开发板与电脑即可。

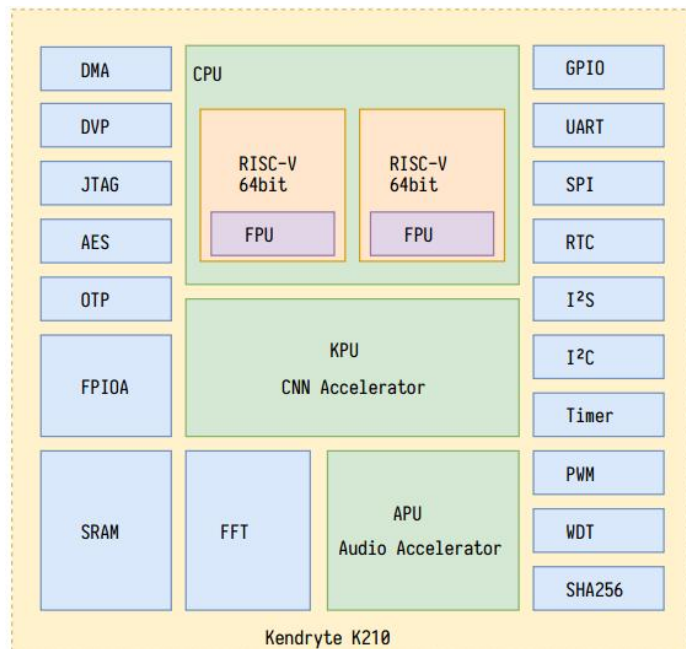
三、实验原理

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

这节我们主要介绍人脸检测的原理和实现方法。下图为 K210 核心图：



KPU 介绍

KPU 是通用的神经网络处理器，它可以在低功耗的情况下实现卷积神经网络计算，实时获取被检测目标的大小、坐标和种类，对人脸或者物体进行检测和分

类。使用 kpu 时，必须结合 model compiler。

KPU 特点:

- 支持主流训练框架按照特定限制规则训练出来的定点化模型，对网络层数无直接限制，
- 支持每层卷积神经网络参数单独配置，包括输入输出通道数目、输入输出行宽列高。
- 支持两种卷积内核 1x1 和 3x3
- 支持任意形式的激活函数
- 实时工作时最大支持神经网络参数大小为 5.5MiB 到 5.9MiB
- 非实时工作时最大支持网络参数大小为 (Flash 容量-软件体积)

K210 实时人脸识别方案步骤:

人脸检测是检测出图片中的人脸，并能够标示出人脸的位置。人脸检测技术主要完成了两件工作：

第一，判断图片中是否包含人脸区域；

第二，如果图片中存在人脸，将人脸的位置预测出来。

四、实验步骤

1、代码流程

系统内部初始化部分:

- 系统时钟初始化

- 串口初始化
- 硬件引脚初始化
- IO 电压设置
- 系统中断初始化
- Flash 初始化

外部硬件初始化

- Lcd 初始化
- Ov2640 初始化

人脸检测初始化

- 模型加载
- 人脸检测层配置初始化

人脸检测业务逻辑层

- 等待摄像头采集完成
- 传入摄像头采集的图像到 KPU 运行模型
- 等待 KPU 处理完成
- 获取 KPU 最终处理的结果
- 把 KPU 处理的结果带入区域层计算最终位置
- 根据获取的人脸个数进行逐一标记

2、核心代码如下：

```
int main(void)
{

    sysclock_init();    /* 系统时钟初始化*/
    uarths_init();      /* 串口初始化*/
```

```

hardware_init();    /* 硬件引脚初始化*/
io_set_power();     /* 设置 IO 口电压*/
plic_init();        /* 系统中断初始化 */

printf("flash init\n");
w25qxx_init(3, 0);    /* flash init */
w25qxx_enable_quad_mode(); /* flash 四倍模式开启*/

/*kmodel 加载方式: 1: 分开烧录模式 2: 直接与代码合并编译*/
#if LOAD_KMODEL_FROM_FLASH
    model_data = (uint8_t*)malloc(KMODEL_SIZE + 255);
    uint8_t *model_data_align = (uint8_t*)((uintptr_t)model_data+255)&(~255));
    w25qxx_read_data(0xA00000, model_data_align, KMODEL_SIZE, W25QXX_QUAD_FAST);
#else
    uint8_t *model_data_align = model_data;
#endif

// 初始化 LCD
lcd_init();
lcd_draw_picture_half(0, 0, 320, 240, (uint32_t *)logo);
lcd_draw_string(100, 40, "Hello Yahboom!", RED);
lcd_draw_string(100, 60, "Demo: Face Detect!", BLUE);
sleep(1);

ov2640_init();

/* init face detect model  KPU 任务句柄 kmodel 数据*/
//加载 kmodel, 需要与 nncase 配合使用
if (kpu_load_kmodel(&face_detect_task, model_data_align) != 0)
{
    printf("\nmodel init error\n");
    while (1);
}
//人脸层配置参数
face_detect_rl.anchor_number = ANCHOR_NUM;
face_detect_rl.anchor = g_anchor;
face_detect_rl.threshold = 0.7;
face_detect_rl.nms_value = 0.3;
region_layer_init(&face_detect_rl, 20, 15, 30, 320, 240);

printf("REGION LAYER INIT, FREE MEM: %ld\r\n", (long)get_free_heap_size());

```

```

sysctl_enable_irq();
/* system start */
printf("System start \n");

while (1)
{
    g_dvp_finish_flag = 0;
    while (g_dvp_finish_flag == 0);    //等待采集中断 使能

    /* run face detect */
    g_ai_done_flag = 0;
    // 运行 kmodel    KPU 任务句柄    源数据    DMA 通道    完成后回调函数    回调的参数
    kpu_run_kmodel(&face_detect_task, g_ai_red_buf_addr, DMAC_CHANNEL5, ai_done,
    NULL);
    while(!g_ai_done_flag);    //等待 KPU 处理完成

    float *output;
    size_t output_size;
    // 获取 KPU 最终处理的结果    KPU 任务句柄    结果的索引值    结果    大小(字节)
    kpu_get_output(&face_detect_task, 0, (uint8_t **)&output, &output_size);

    /*算法检测人脸*/
    face_detect_rl.input = output;
    region_layer_run(&face_detect_rl, &face_detect_info);

    /*根据返回值进行人脸圈住 */
    for (uint32_t face_cnt = 0; face_cnt < face_detect_info.obj_number; face_cnt
    ++)
    {
        draw_edge((uint32_t *)display_buf_addr, &face_detect_info, face_cnt, RED);
    }
    /* display result */
    lcd_draw_picture(0, 0, 320, 240, (uint32_t *)display_buf_addr);
}

return 0;
}

```

3、代码编译方法：

把本课程资料中的 face_detection 文件夹复制到 SDK 中的 src 目录下，
然后进入 build 目录，删除 build 目录下所有文件，最后运行以下命令编译。

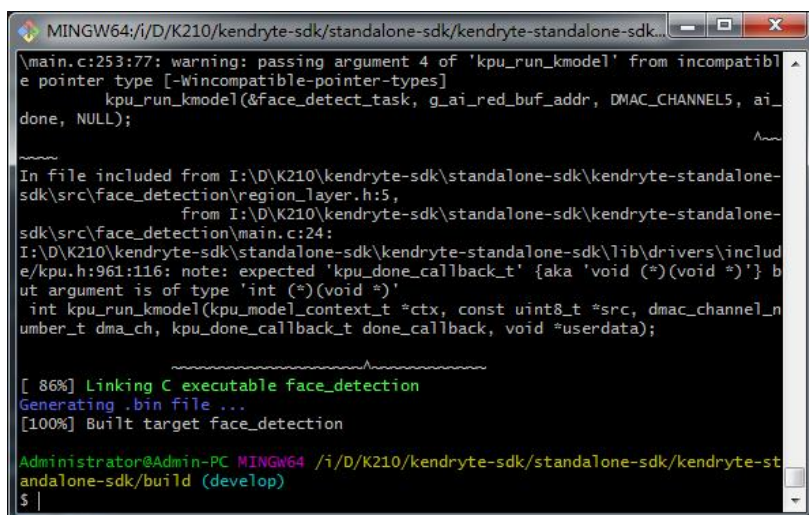
```
cmake .. -DPROJ=face_detection -G "MinGW Makefiles"
```

```
make
```

编译完成后，在 build 文件夹下会生成 face_detection.bin 文件，

4、代码烧录方法

打开 kflash 将 face_detection.bin 文件烧录到 K210 开发板上。



```

MINGW64/i/D/K210/kendryte-sdk/standalone-sdk/kendryte-standalone-sdk...
\\main.c:253:77: warning: passing argument 4 of 'kpu_run_kmodel' from incompatibl
e pointer type [-Wincompatible-pointer-types]
    kpu_run_kmodel(&face_detect_task, g_ai_red_buf_addr, DMAC_CHANNELS, ai_
done, NULL);
~~~~~
In file included from I:\\D\\K210\\kendryte-sdk\\standalone-sdk\\kendryte-standalone-
sdk\\src\\face_detection\\region_layer.h:5,
               from I:\\D\\K210\\kendryte-sdk\\standalone-sdk\\kendryte-standalone-
sdk\\src\\face_detection\\main.c:24:
I:\\D\\K210\\kendryte-sdk\\standalone-sdk\\kendryte-standalone-sdk\\lib\\drivers\\includ
e\\kpu.h:961:116: note: expected 'kpu_done_callback_t' {aka 'void (*)(void *)'} b
ut argument is of type 'int (*)(*)(void *)'
    int kpu_run_kmodel(kpu_model_context_t *ctx, const uint8_t *src, dma_channel_n
umber_t dma_ch, kpu_done_callback_t done_callback, void *userdata);
~~~~~
[ 86%] Linking C executable face_detection
Generating .bin file ...
[100%] Built target face_detection

Administrator@Admin-PC MINGW64 /i/D/K210/kendryte-sdk/standalone-sdk/kendryte-st
andalone-sdk/build (develop)
$ |

```

扩展部分：

如果想把代码和模型文件分开烧录，需要修改代码

```
#define LOAD_KMODEL_FROM_FLASH 0
```

为

```
#define LOAD_KMODEL_FROM_FLASH 1
```

然后重新生成 bin 文件，这个时候我们需要把模型文件和 bin 文件打包成一个
kfpkg 文件在烧录。.bin 文件是固件内容，作为参数传给烧录软件，软件会默认

烧录到 flash 开头，完成后重启即可运行；但是有时候我们需要烧录其它二进制文件到 flash，比如烧录模型、文件系统或者自己定义的其它数据，这时需要指定烧录的地址，光是 .bin（二进制）文件烧录工具不知道我们想把数据烧录到 flash 的哪里，打包一个 .kfpkg 格式的文件则是为了实现这个目的。

kfpkg 由 3 部分组成：

flash-list.json 文本文件，.bin 文件列表以及烧录地址等信息

*.bin 固件

. 其他文件(二进制文件)

比如我们想同时下载名为 face_detection.bin 的固件，以及 detect.kmodel 的其它文件到 Flash 的 0xA00000 地址，则需要写一个 flash-list.json 文件，内容如下：

```
{
  "version": "0.1.0",
  "files": [
    {
      "address": 0,
      "bin": "face_detection.bin",
      "sha256Prefix": true
    },
    {
      "address": 0x00A00000,
      "bin": "detect.kmodel",
      "sha256Prefix": false
    }
  ]
}
```

注意在代码中需要根据 kfpkg 中的地址写对，否则无法读取模型文件。


```
/*kmodel加载方式： 1： 分开烧录模式 2： 直接与代码合并编译*/  
#if LOAD_KMODEL_FROM_FLASH  
    model_data = (uint8_t*)malloc(KMODEL_SIZE + 255);  
    uint8_t *model_data_align = (uint8_t*)((uintptr_t)model_data+255)&(~255);  
    w25qxx_read_data(0xA00000, model_data_align, KMODEL_SIZE, W25QXX_QUAD_FAST);  
#else  
    uint8_t *model_data_align = model_data;  
#endif
```

编写完成后，压缩这三个文件为 zip，然后修改扩展名为.kfpkg，下载直接选择此 kfpkg 文件即可下载运行程序。

四、实验现象

LCD 显示器先显示图片 logo 和文字，一秒后打开摄像头采集的画面，并且实时检测人脸并框住。

五、实验总结

1. 人脸检测利用的是摄像头采集画面后再进行处理，最后显示到 LCD 上。
2. 需要下载.kfpkg 文件到 K210 开发板上。