8 Face recognition

8.1 experimental goals

This lesson mainly learns the face recognition function, which has more recognition effects than the face detection function.

The reference code path for this experiment is： CanMV\05-AI\face_recog.py

8.2 Preparation before the experiment

Please first import the model file into the memory card, and then insert the memory card into the memory card slot of the K210 module. Please refer to:

appendix：Import the model file into memory card

8.3 experimental procedure

The factory firmware of the module has integrated the AI vision algorithm module. If you have downloaded other firmware, please burn it back to the factory firmware before doing the experiment.

1. Import the libraries and initialize the camera and LCD display.

```
from maix import GPIO, utils
from fpioa_manager import fm
from board import board_info

lcd.init()
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 100)
clock = time.clock()
```

2. The size of the new face feature image is 6464.

```
feature_img = image.Image(size=(64,64), copy_to_fb=False)
feature_img.pix_to_ai()

FACE_PIC_SIZE = 64
dst_point =[(int(38.2946 * FACE_PIC_SIZE / 112), int(51.6963 * FACE_PIC_SIZE / 
112)),
            (int(73.5318 * FACE_PIC_SIZE / 112), int(51.5014 * FACE_PIC_SIZE / 
112)),
            (int(56.0252 * FACE_PIC_SIZE / 112), int(71.7366 * FACE_PIC_SIZE / 
112)),
            (int(41.5493 * FACE_PIC_SIZE / 112), int(92.3655 * FACE_PIC_SIZE / 
112)),
            (int(70.7299 * FACE_PIC_SIZE / 112), int(92.2041 * FACE_PIC_SIZE / 112))
]
```

3. Initialize the parameters related to the face detection model, the model file path is: sd KP Uyolo face detectface detect_320 x 240.kmodel, and use yolo 2 to calculate whether it meets the model requirements.

```
anchor = (0.1075, 0.126875, 0.126875, 0.175, 0.1465625, 0.2246875, 0.1953125,
0.25375, 0.2440625, 0.351875, 0.341875, 0.4721875, 0.5078125, 0.6696875, 0.8984375,
1.099687, 2.129062, 2.425937)
kpu = KPU()
kpu.load_kmodel("/sd/KPU/yolo_face_detect/face_detect_320x240.kmodel")
kpu.init_yolo2(anchor, anchor_num=9, img_w=320, img_h=240, net_w=320 , net_h=240
,layer_w=10 ,layer_h=8, threshold=0.7, nms_value=0.2, classes=1)
```

4. Initialize the LD 5 model parameters at the address of the model:/sd/KPU/face_recognization/ld5.kmodel。

```
ld5_kpu = KPU()
print("ready load model")
ld5_kpu.load_kmodel("/sd/KPU/face_recognization/ld5.kmodel")
```

5. Initialize the relevant parameters of the feature model, and the address of the model is:/sd/KPU/face_recognization/feature_extraction.kmodel。

```
fea_kpu = KPU()
print("ready load model")
fea_kpu.load_kmodel("/sd/KPU/face_recognization/feature_extraction.kmodel")
```

6. New button function, rising edge trigger, the main function is to record the face to be recognized.

```
start_processing = False
BOUNCE_PROTECTION = 50

fm.register(board_info.BOOT_KEY, fm.fpioa.GPIOHS0)
key_gpio = GPIO(GPIO.GPIOHS0, GPIO.IN)
def set_key_state(*_):
    global start_processing
    start_processing = True
    time.sleep_ms(BOUNCE_PROTECTION)
key_gpio.irq(set_key_state, GPIO.IRQ_RISING, GPIO.WAKEUP_NOT_SUPPORT)
```

7. Create a new face recognition variable, where the variable 'record_ftrs': an array representing the recognition of face features; Variable 'THRESHOLD': indicates the threshold of face recognition, beyond which it is considered to be a recognized face; Variable 'recog_flag': indicates the state of whether the detected face has been recognized, recognized as True, not recognized as Flase.

```
record_ftrs = []
THRESHOLD = 80.5
recog_flag = False
```

8. Extract information about detected faces.

```
def extend_box(x, y, w, h, scale):
    x1_t = x - scale*w
    x2_t = x + w + scale*w
    y1_t = y - scale*h
    y2_t = y + h + scale*h
    x1 = int(x1_t) if x1_t>1 else 1
    x2 = int(x2_t) if x2_t<320 else 319
    y1 = int(y1_t) if y1_t>1 else 1
    y2 = int(y2_t) if y2_t<240 else 239
    cut_img_w = x2-x1+1
    cut_img_h = y2-y1+1
    return x1, y1, cut_img_w, cut_img_h
```

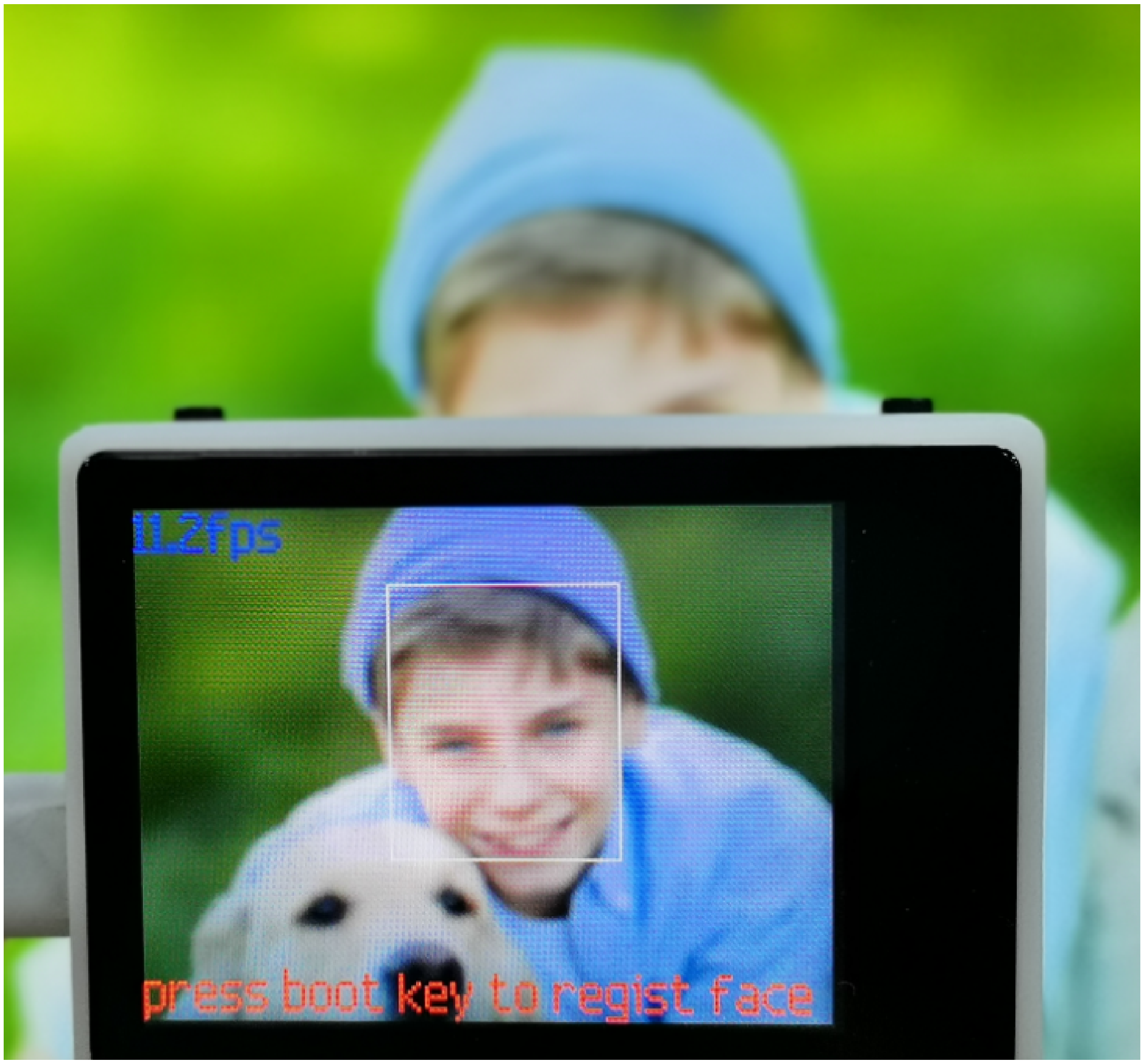
9. New while loop, the main function is to detect the face first, when the BOOT button is recognized, record the face information. When the detected face information is greater than the recognition threshold, it means that it is a recognized face, with a green box, and the recognition score is displayed, otherwise it means an unrecognized face, with a white box.

```
while True:
    gc.collect()
    # print("mem free:",gc.mem_free())
    # print("heap free:",utils.heap_free())
    clock.tick()
    img = sensor.snapshot()
    kpu.run_with_output(img)
    dect = kpu.regionlayer_yolo2()
    fps = clock.fps()
    if len(dect) > 0:
        for l in dect :
            x1, y1, cut_img_w, cut_img_h= extend_box(l[0], l[1], l[2], l[3],
scale=0)
            face_cut = img.cut(x1, y1, cut_img_w, cut_img_h)
            face_cut_128 = face_cut.resize(128, 128)
            face_cut_128.pix_to_ai()
            out = ld5_kpu.run_with_output(face_cut_128, getlist=True)
            face_key_point = []
            for j in range(5):
                x = int(KPU.sigmoid(out[2 * j])*cut_img_w + x1)
                y = int(KPU.sigmoid(out[2 * j + 1])*cut_img_h + y1)
                face_key_point.append((x,y))
            T = image.get_affine_transform(face_key_point, dst_point)
            image.warp_affine_ai(img, feature_img, T)
            feature = fea_kpu.run_with_output(feature_img, get_feature = True)
            del face_key_point
            scores = []
            for j in range(len(record_ftrs)):
                score = kpu.feature_compare(record_ftrs[j], feature)
                scores.append(score)
            if len(scores):
                max_score = max(scores)
                index = scores.index(max_score)
                if max_score > THRESHOLD:
                    img.draw_string(0, 195, "persion:%d,score:%2.1f" %(index,
max_score), color=(0, 255, 0), scale=2)
                    recog_flag = True
                else:
                    img.draw_string(0, 195, "unregistered,score:%2.1f" %(max_score),
color=(255, 0, 0), scale=2)
            del scores
            if start_processing:
                record_ftrs.append(feature)
                print("record_ftrs:%d" % len(record_ftrs))
                start_processing = False

            if recog_flag:
                img.draw_rectangle(l[0],l[1],l[2],l[3], color=(0, 255, 0))
                recog_flag = False
            else:
                img.draw_rectangle(l[0],l[1],l[2],l[3], color=(255, 255, 255))
            del (face_cut_128)
            del (face_cut)

    img.draw_string(0, 0, "%2.1ffps" %(fps), color=(0, 60, 255), scale=2.0)
    img.draw_string(0, 215, "press boot key to regist face", color=(255, 100, 0),
scale=2.0)
    lcd.display(img)
```

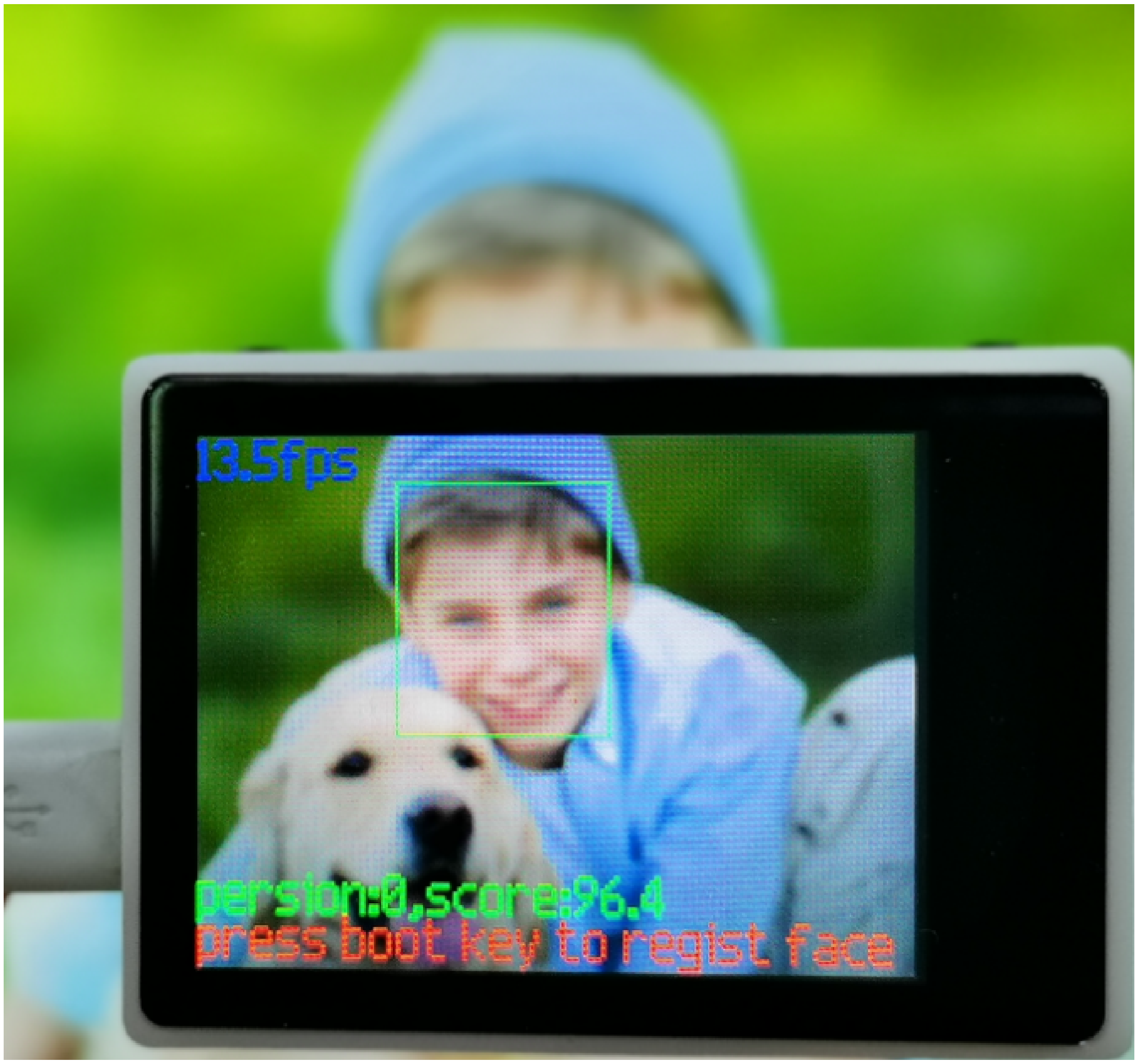
8.4 experimental results

Since the BOOT button is required, do not run it directly in the CanMV IDE, the CanMV IDE cannot detect the BOOT button at present, please download the code as a main.py to the K210 module to run.

Point the camera at the face, and if it is an unrecognized face, a white box will be displayed.



At this time, press the BOOT button in the upper right corner to record the current face information, the white border changes to a green border, and the recognized score is displayed.



8.5 experiment summary

Face recognition needs to use a memory card to load the model file, so you need to import the model file into the memory card in advance, and then insert the memory card into the memory card slot of the K 210 module, if the model file in the memory card cannot be read, an error will be reported.

Since face recognition requires the BOOT button, do not run the face recognition code in the CanMV IDE, download the code as a main.py to the K210 chip, and then press the reset button to start running.

Recognized faces are sorted in order, with the first person recognized as person:0, the second recognized person as person:1, and so on.