5 Face detection

5.1 experimental goals

This lesson mainly learns the face detection function, analyzes the picture collected by the camera, compares the model, frames it if there is a face, and prints the relevant information.

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

5.2 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

5.3 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.

```
import sensor, image, time, lcd
from maix import KPU

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

2. To initialize the parameters related to KPU, kpu needs to load the kmodel file. The path of the model file required for this experiment is:/sd/KPU/yolo_face_detect/yolo_face_detect.kmodel, And use yolo2 to calculate conformity to the model requirements. od_img is the image of the neural network, the size is 320*256, which is used to store the camera image and pass it to KPU for calculation.

```
od_img = image.Image(size=(320,256))

anchor = (0.893, 1.463, 0.245, 0.389, 1.55, 2.58, 0.375, 0.594, 3.099, 5.038, 0.057,
0.090, 0.567, 0.904, 0.101, 0.160, 0.159, 0.255)
kpu = KPU()
kpu.load_kmodel("/sd/KPU/yolo_face_detect/yolo_face_detect.kmodel")
kpu.init_yolo2(anchor, anchor_num=9, img_w=320, img_h=240, net_w=320 , net_h=256
,layer_w=10 ,layer_h=8, threshold=0.7, nms_value=0.3, classes=1)
```

3. Create a new while loop, pass the image to the KPU for calculation, use the yolo 2 neural network algorithm to solve, and finally obtain the position information of the face, and then frame the face.

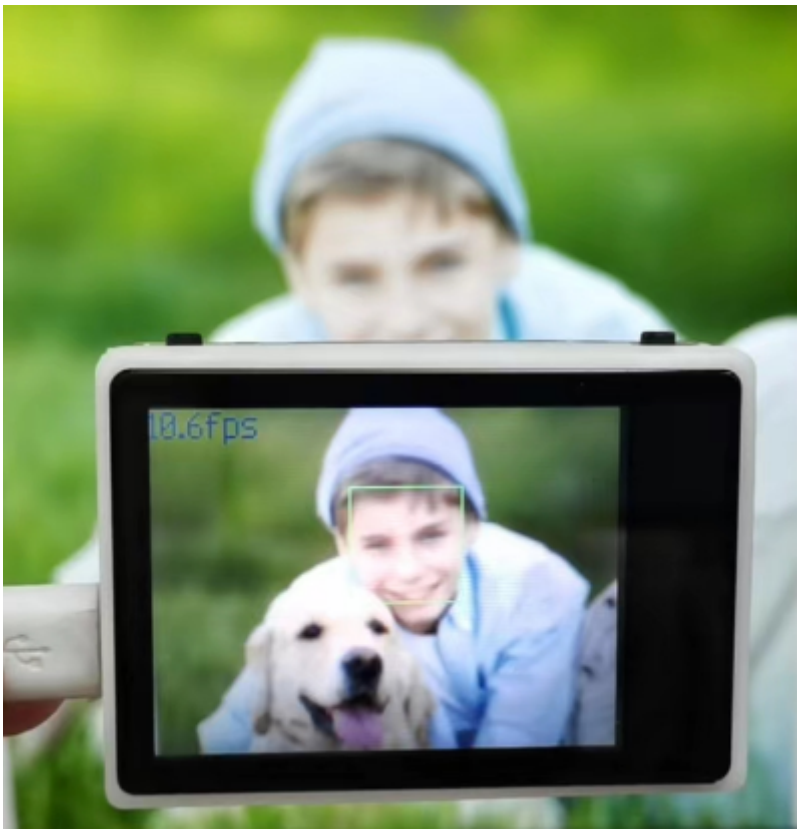
```
while True:
    clock.tick()
    img = sensor.snapshot()
    a = od_img.draw_image(img, 0,0)
    od_img.pix_to_ai()
    kpu.run_with_output(od_img)
    dect = kpu.regionlayer_yolo2()
    fps = clock.fps()
    if len(dect) > 0:
        print("dect:",dect)
        for l in dect :
            a = img.draw_rectangle(l[0],l[1],l[2],l[3], color=(0, 255, 0))
    a = img.draw_string(0, 0, "%2.1ffps" %(fps), color=(0, 60, 128), scale=2.0)
    lcd.display(img)

kpu.deinit()
```

5.4 experimental results

Connect the K210 module to the computer through the microUSB data cable, CanMV IDE click the connect button, after the connection is completed click the Run button to run the routine code. You can also download the code as main.py and run it in the K210 module.

Wait for the system initialization is completed, the LCD displays the camera picture, use the camera to shoot the face, when the face is detected, the screen will appear green frame the face out, and the serial terminal at the bottom of the IDE prints the detected face information.




```
dect: [[134, 54, 78, 82, 0, 0.7314994]]
dect: [[134, 53, 79, 83, 0, 0.7314994]]
dect: [[122, 39, 99, 109, 0, 0.7744358]]
dect: [[126, 39, 92, 110, 0, 0.7876179]]
dect: [[127, 39, 91, 110, 0, 0.8548551]]
dect: [[126, 38, 92, 110, 0, 0.7744358]]
dect: [[122, 38, 99, 110, 0, 0.8002282]]
dect: [[135, 54, 79, 83, 0, 0.8237426]]
dect: [[125, 39, 92, 110, 0, 0.7744358]]
dect: [[121, 38, 99, 110, 0, 0.7160898]]
dect: [[110, 28, 85, 77, 0, 0.7160898]]
```

5.5 experiment summary

Face detection requires 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.

At present, the threshold value for detecting faces is threshold=0.7, and if you need to detect faces more accurately, you can adjust the threshold appropriately.