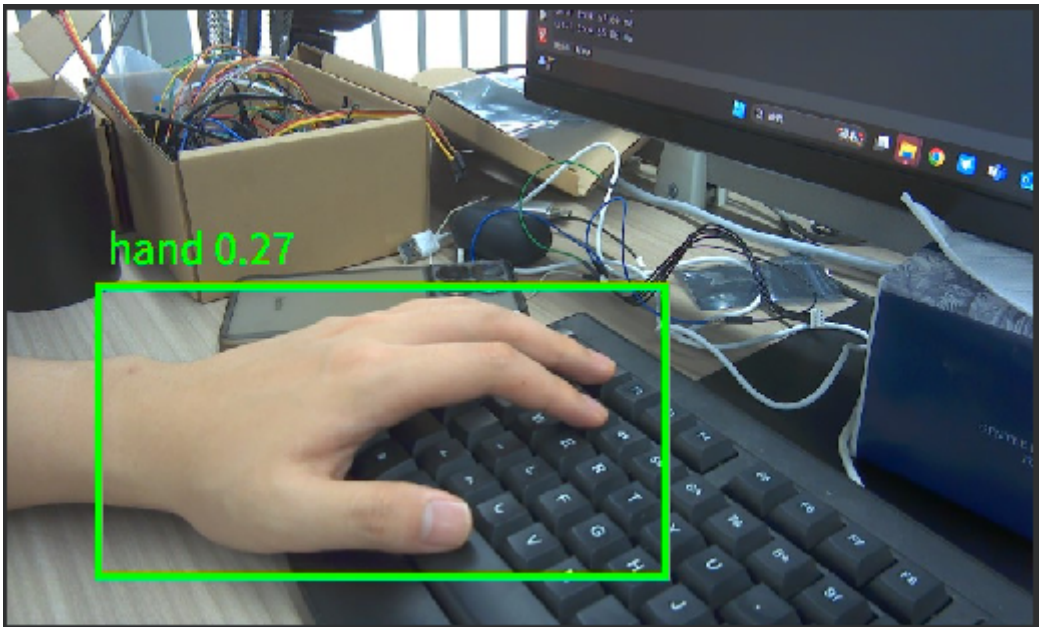
Palm detection

Routine Experiment Effect

In this section, we will learn how to use K230 to implement palm detection.

The example code is in [Source code/08.Body/05.hand_detection.py]

The routine running effect is as follows: K230 will recognize all the hands that appear in the picture.



> Since people's hand postures vary a lot, in order to reduce the possibility of misrecognition, the recognition threshold is set low. This may lead to other things that are not hands being recognized as hands. If you need higher-precision recognition, you can take photos of your hands with K230, annotate them, and train them yourself.

> Serial port output function has been added
>
> After detecting the hand, the following serial output format will be sent
>
> $x,y,w,h#

> The '$' represents the beginning of the data, and the '#' represents the end of the data.
>
> x, y, w, h are the positions of the hand detection frame (resolution is 640*480)

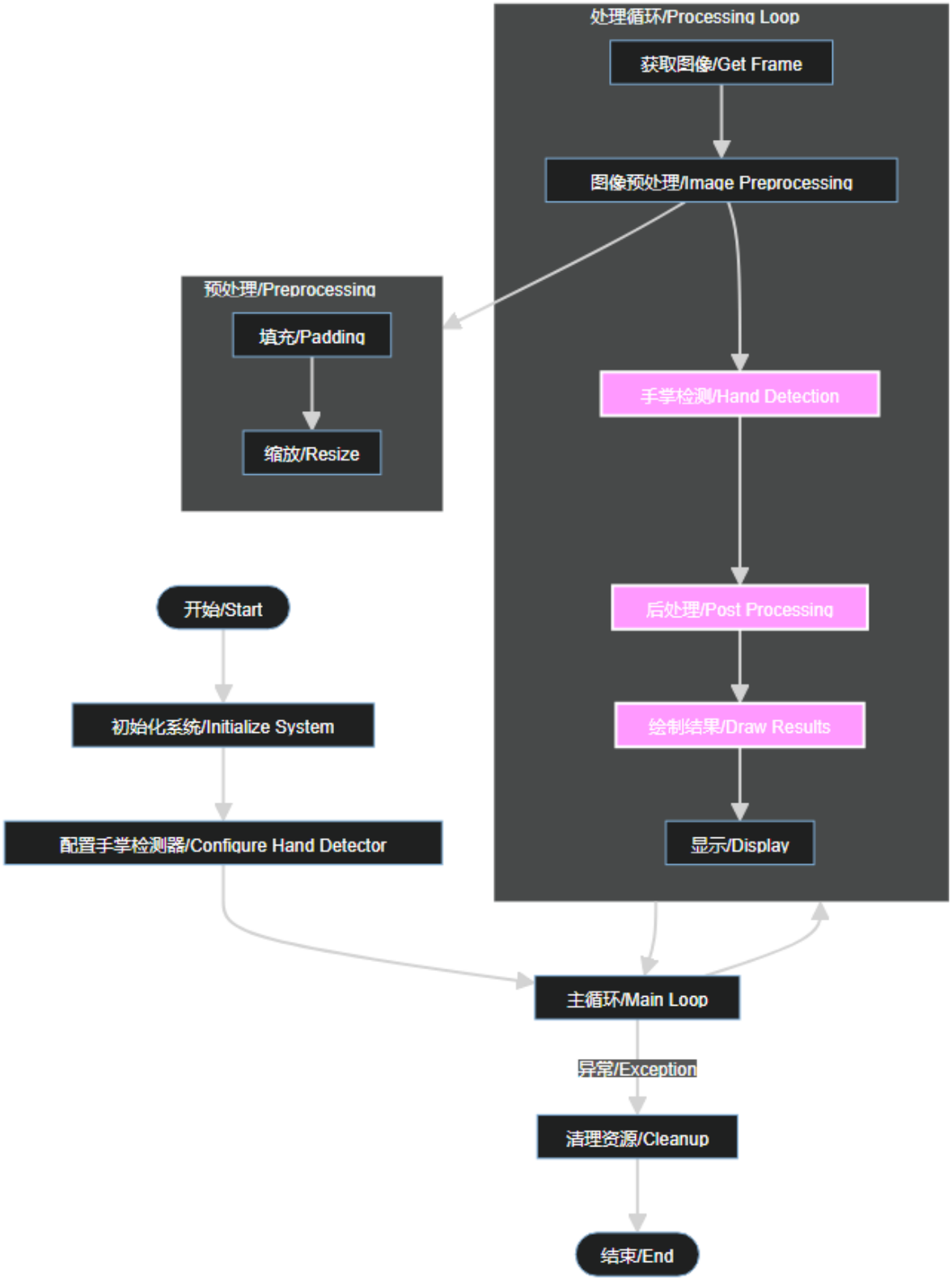
Code Explanation

For the complete code, please refer to the file [Source Code/08.Body/04.hand_detection.py]

```
if __name__=="__main__":

    display_mode="lcd"
    rgb888p_size = [640,360]

    # 根据显示模式设置显示分辨率
    # Set the display resolution according to the display mode
    if display_mode=="hdmi":
        display_size=[1920,1080]
    else:
        display_size=[640,480]

    # 模型路径
    # Model path
    kmodel_path="/sdcard/kmodel/hand_det.kmodel"

    # 其它参数设置
    # Other parameter settings
    confidence_threshold = 0.2
    nms_threshold = 0.5
    labels = ["hand"]
    anchors = [26,27, 53,52, 75,71, 80,99, 106,82, 99,134, 140,113, 161,172,
245,276]   #anchor设置

    # 初始化PipeLine
    # Initialize PipeLine

 pl=PipeLine(rgb888p_size=rgb888p_size,display_size=display_size,display_mode=di
splay_mode)
    pl.create()

    # 初始化自定义手掌检测实例
    # Initialize custom hand detection instance
    hand_det=HandDetectionApp(kmodel_path,model_input_size=
[512,512],labels=labels,anchors=anchors,confidence_threshold=confidence_threshol
d,nms_threshold=nms_threshold,nms_option=False,strides=
[8,16,32],rgb888p_size=rgb888p_size,display_size=display_size,debug_mode=0)
    hand_det.config_preprocess()

    # 主循环，持续检测并显示结果
    # Main loop, continuously detect and display results
    while True:
        with ScopedTiming("total",1):
            # 获取当前帧数据
            # Get current frame data
            img=pl.get_frame()

            # 推理当前帧
            # Infer current frame
```

```
            res=hand_det.run(img)

            # 绘制结果到PipeLine的osd图像
            # Draw results to PipeLine's osd image
            hand_det.draw_result(pl,res)

            # 显示当前的绘制结果
            # Display current drawing results
            pl.show_image()

            # 垃圾回收
            # Garbage collection
            gc.collect()

    # 反初始化
    # Deinitialize
    hand_det.deinit()

    # 销毁PipeLine实例
    # Destroy PipeLine instance
    pl.destroy()
```

flow chart



Brief description of palm detection algorithm

> Palm detection is a basic requirement for many subsequent application scenarios

Common application scenarios

Palm detection algorithms have many practical scenarios in modern life

Biometrics and Authentication

- Access Control System
- Mobile Payment
- Device unlock

Human-computer interaction

- VR/AR controls
- Smart Home Gesture Control
- No touch screen operation

Security Monitoring

- Industrial safety (hazardous area hand gesture warning)
- Driving behavior analysis
- Suspicious behavior detection in public places

Healthcare

- Hand rehabilitation training
- Contactless operation in operating room
- Telemedicine Gesture Interaction

Education and Training

- Sign language recognition teaching
- Instrumental Performance Instruction
- Exercise posture correction

Algorithm Overview

Network structure

Feature extraction network

- Input RGB image of size 512×512
- Use multi-scale feature extraction with 3 feature levels (stride=[8,16,32])
- Feature maps of different scales correspond to detecting targets of different sizes

Detection head design

- Adopting a multi-scale detection strategy
- 9 groups of anchor boxes are preset at each feature map position
- The sizes of anchors from small to large are:
    - Small size: (26×27, 53×52, 75×71)
    - Medium size: (80×99, 106×82, 99×134)
    - Large size: (140×113, 161×172, 245×276)

Detection principle

Preprocessing stage

- The input image is scaled to 512×512
- Keep the aspect ratio, and pad the missing part
- Resize using NCHW format and bilinear interpolation

Testing process

- The network outputs 3 branches, corresponding to 3 different scales
- Each location prediction:
    - Class probability (palm/not palm)
    - Bounding box regression values (x,y,w,h)
    - Confidence score

Post-processing process

- Confidence threshold filtering (confidence_threshold=0.2)
- Execute NMS (nms_threshold=0.5) to remove overlapping boxes
- Bounding box decoding to get the final detection result
- Additional filtering for small and edge objects:
    - Boxes with height < 0.1×displaywidth are filtered
    - Boxes with unreasonable width and position are filtered out