Fall Detection

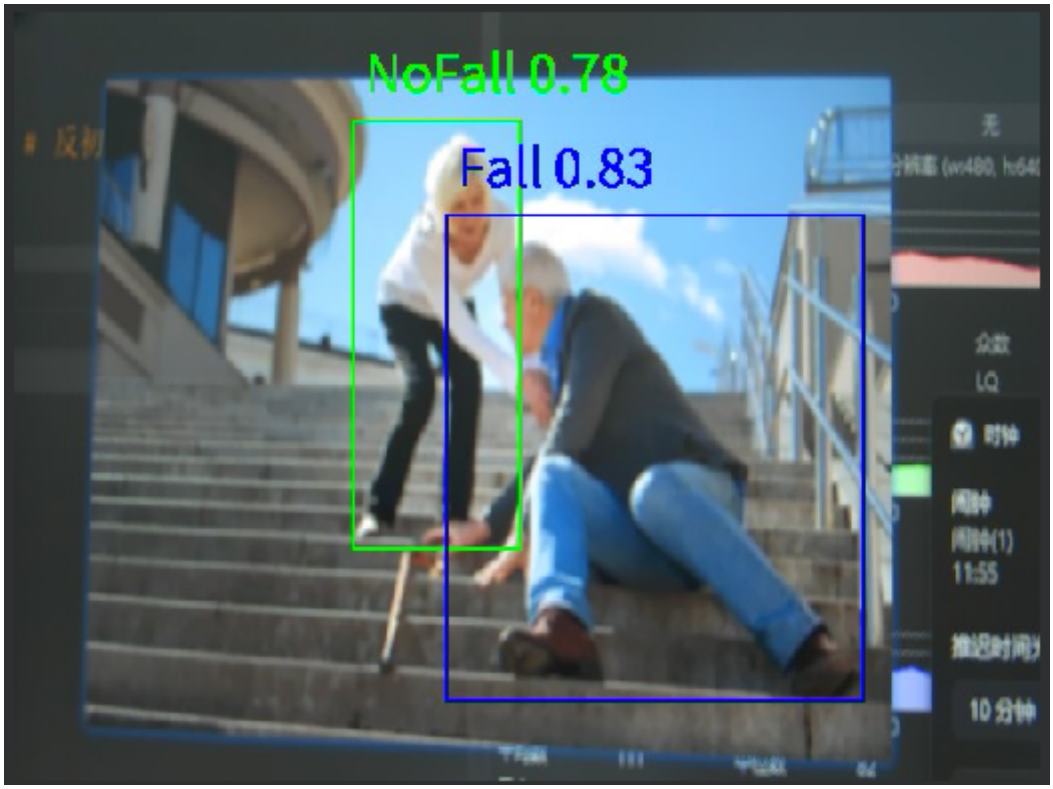
Routine Experiment Effect

> Fall detection is a practical application scenario of the human key point detection we learned in the previous section.

In this section, we will learn how to use K230 to implement the fall detection function.

The example code is in [Source code/08.Body/04.falldown_detect.py]

The results of the routine are as follows: K230 will identify the status of the person in the picture (falling/not falling) and give the corresponding confidence level.



> Serial port output function has been added
>
> After detecting a human body, the following serial output format will be sent
>
> $x,y,w,h,w ...h
>
> The '$' represents the beginning of the data, and the '#' represents the end of the data.
>
> x, y, w, h are the positions of the human body detection frame (resolution is 640*480)

> Let's take a few examples to illustrate:
>
> If the serial port outputs: $489,0,146,253,NoFall,0.38#
>
> This means a person was detected at the position (489,0,146,253).
>
> NoFall means no fall
>
> 0.38 is the confidence level, which ranges from 0 to 1. The higher the confidence level, the greater the possibility that the check is correct.
>
> If the serial port outputs: $483,60,156,386,Fall,0.35#
>
> This means a person was detected at the position (483,60,156,386).
>
> Fall means falling
>
> 0.35 is the confidence level, which ranges from 0 to 1. The higher the confidence level, the greater the possibility that the check is correct.

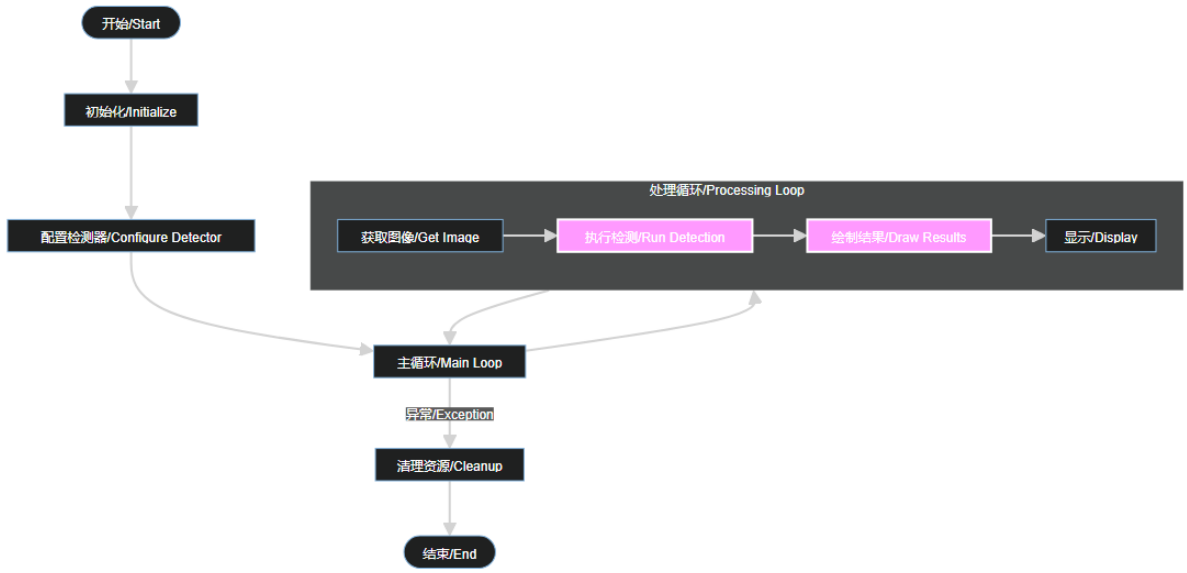
Code Analysis

For the complete code, please refer to the file [Source Code/08.Body/02.person_keypoint_detect.py]

```
if __name__ == "__main__":

    display_mode="lcd"
    # 设置RGB888P图像大小
    # Set RGB888P image size
    rgb888p_size = [640,360]

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

    # 设置模型路径和其他参数
    # Set model path and other parameters
    kmodel_path = "/sdcard/kmodel/yolov5n-falldown.kmodel"  # 模型文件路径 / Model file path
    confidence_threshold = 0.3  # 置信度阈值 / Confidence threshold
    nms_threshold = 0.45  # NMS阈值 / NMS threshold
    labels = ["Fall","NoFall"]  # 模型输出类别名称 / Model output category names

    # anchor设置，用于YOLOv5检测框解码
    # Anchor settings for YOLOv5 detection box decoding
    anchors = [10, 13, 16, 30, 33, 23, 30, 61, 62, 45, 59, 119, 116, 90, 156, 198, 373, 326]

    # 初始化PipeLine，用于图像处理流程
    # Initialize PipeLine for image processing workflow
    pl = PipeLine(rgb888p_size=rgb888p_size, display_size=display_size, display_mode=display_mode)
    pl.create()
```

```
    # 初始化自定义跌倒检测实例
    # Initialize custom fall detection instance
    fall_det = FallDetectionApp(
        kmodel_path,               # 模型路径 / Model path
        model_input_size=[640, 640],  # 模型输入大小 / Model input size
        labels=labels,             # 标签列表 / Label list
        anchors=anchors,           # 锚点设置 / Anchor settings
        confidence_threshold=confidence_threshold,  # 置信度阈值 / Confidence
threshold
        nms_threshold=nms_threshold,  # NMS阈值 / NMS threshold
        nms_option=False,          # NMS选项 / NMS option
        strides=[8,16,32],         # 步长设置 / Stride settings
        rgb888p_size=rgb888p_size, # RGB图像大小 / RGB image size
        display_size=display_size,  # 显示大小 / Display size
        debug_mode=0               # 调试模式 / Debug mode
    )

    # 配置预处理流程
    # Configure preprocessing pipeline
    fall_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 = fall_det.run(img)

            # 绘制结果到PipeLine的osd图像
            # Draw results to PipeLine's osd image
            fall_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，释放内存
            # Garbage collection to free memory
            gc.collect()

    # 反初始化跌倒检测实例
    # Deinitialize fall detection instance
    fall_det.deinit()

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

flow chart



A brief description of the fall detection algorithm

This is a fall detection implementation based on YOLOv5:

1. The overall architecture code implements a fall detection application (FallDetectionApp), which inherits from the AIBase base class. It mainly includes the following functional modules:

- Image preprocessing
- Model Inference
- Post-processing
- Results Visualization

1. Core Process

```
Image input -> preprocessing (AI2D) -> model inference -> post-processing ->
result drawing
```

1. Detailed explanation of key steps

(1) Preprocessing

- Use AI2D for image preprocessing, including:
  - Padding: padding the edges to make the image conform to the model input requirements
  - resize: scale the image to the model input size (640x640)
- Preprocessing keeps the image ratio unchanged and avoids deformation

(2) Model Reasoning

- Use YOLOv5-n as the base network
- The model input is a 640x640 RGB image
- The output includes 3 detection heads, corresponding to 3 different scales of detection

(3) Post-processing

- Use anchor-based detection post-processing

- The following steps are involved:
    - Detection box decoding
    - Confidence filtering (threshold=0.3)
    - NMS processing (threshold=0.45)
- Output the category, confidence and coordinate information of the detection box

(4) Visualization

- Mapping detection results back to display resolution
- Draw bounding boxes and class labels
- Supports two categories: "Fall" (fall) and "NoFall" (no fall)