Self-learning object recognition

Routine Experiment Effect

In this section, we will learn how to use K230 to simply save the feature data of an object and perform basic recognition.

Let's take "mobile phone" and "ultrasonic module" as examples.

Run the example program in this section, place the [mobile phone] and [ultrasonic module] in front of the K230 camera respectively, and the K230 will start to record the features of the object at the center of the camera and save it as a binary bin file.

> The current routine has added serial port output
>
> For the protocol format, please refer to [Routine Communication Protocol.xlsx] in the document.

Code Explanation

Code structure

Initialization phase:

- Loading tracking templates and live models
- Initialize the tracker
- Set preprocessing parameters
- Initialize the AI2D preprocessor
- Configuring Tracking Thresholds

Main loop phase:

- Image acquisition and preprocessing
- Template reasoning and real-time reasoning
- Tracking algorithm processing
- Results
- Resource Recycling

Exit Processing:

- Abnormality Check
- Resource Cleanup

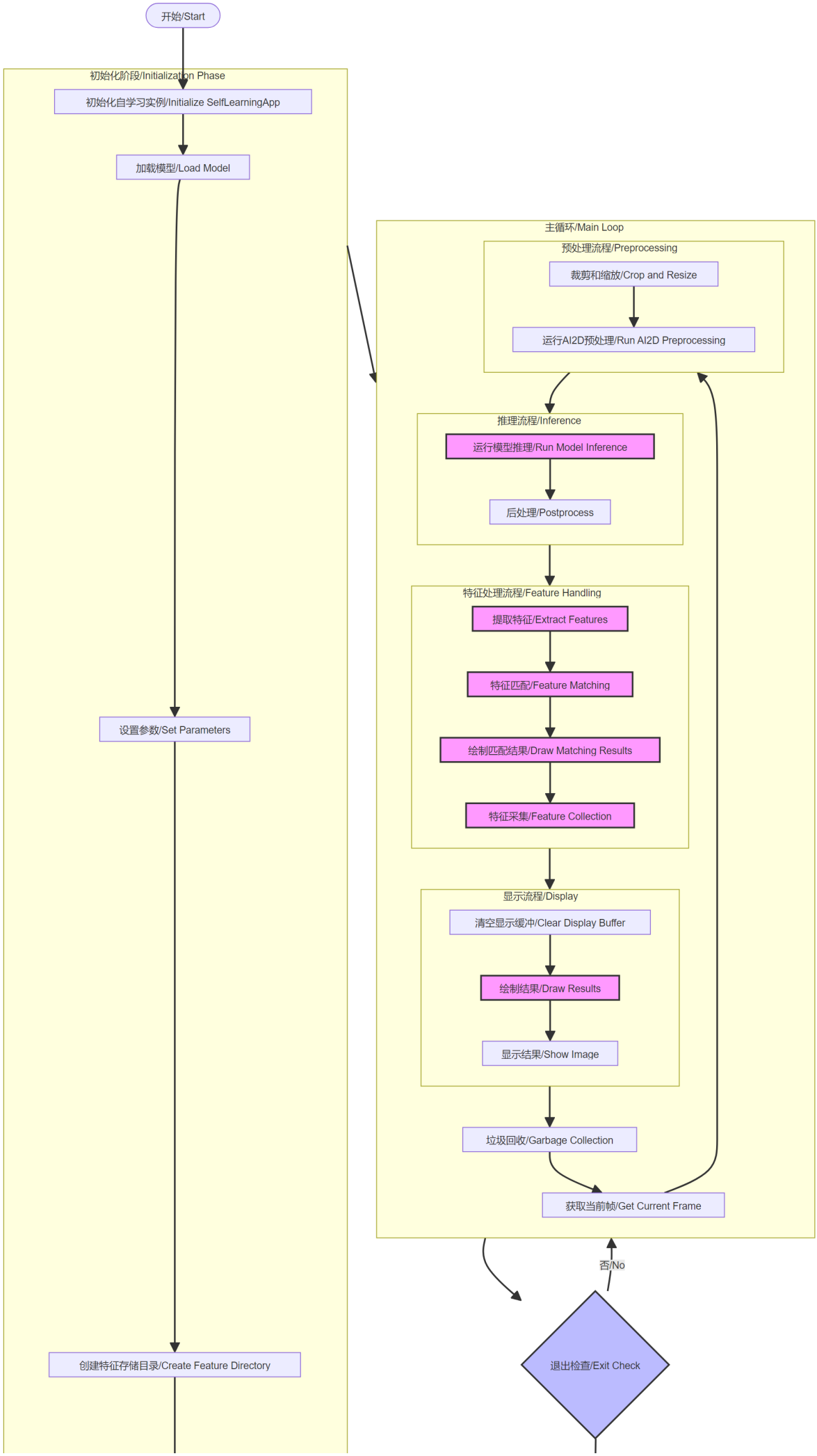
Key Code

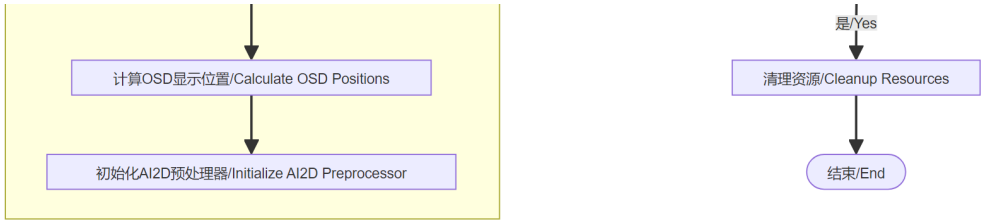
For the complete code, please refer to the file [Source Code/09.Scene/05.self_learning.py]

```
# 自学习例程执行函数
# Self-learning demo execution function
def exce_demo(pl, recong_only=False):
    """
    自学习例程主要流程：
      1. 初始化自学习实例
      2. 启动图像采集、推理、特征采集及后续匹配展示
    Main execution flow for self-learning demo:
      1. Initialize self-learning instance.
      2. Start continuous frame capture, inference, feature collection, and
matching display.

    参数：
      pl (PipeLine): PipeLine实例，用于图像采集和显示 // Pipeline instance for frame
capture and display.
      recong_only (bool): 是否仅作为识别模式，不进行特征采集 // Whether to run in
recognition-only mode.
    """
    global sl, database_path

    # 获取显示和图像参数
    # Retrieve display and image sizes
    display_mode = pl.display_mode
    rgb888p_size = pl.rgb888p_size
    display_size = pl.display_size

    # 模型路径设置
    # Set model path
    kmodel_path = "/sdcard/kmodel/recognition.kmodel"
    # 特征存储路径
    # Database (feature storage) path.
    database_path = "/sdcard/utils/features/"
    # 其它参数设置，例如模型输入尺寸、类别标签、top_k以及匹配阈值
    # Other parameters: model input size, labels, top_k and threshold.
    model_input_size = [224,224]
    labels = ["耳机", "超声波模块"]
    # labels = ["earphone", "ultrasonic"]
    top_k = 3
    threshold = 0.45

    try:
        # 初始化自学习实例
        # Initialize the SelfLearningApp instance
        sl = SelfLearningApp(
            kmodel_path,
            model_input_size=model_input_size,
            labels=labels,
            top_k=top_k,
            threshold=threshold,
            database_path=database_path,
            rgb888p_size=rgb888p_size,
            display_size=display_size,
            debug_mode=0,
```

```
            recong_only=recong_only
        )
        # 配置预处理操作
        # Configure the preprocessing operations
        sl.config_preprocess()
        # 无限循环处理每一帧数据
        # Infinite loop to process each frame
        while True:
            with ScopedTiming("total", 1):
                # 获取当前帧图像数据
                # Capture the current frame from the sensor
                img = pl.get_frame()
                # 推理当前帧图像，得到输出特征
                # Run inference on the current frame to get features
                res = sl.run(img)
                # 绘制结果到Pipeline的OSD图像中（包含特征采集框等信息）
                # Draw the results (feature collection box and matching info)
onto the pipeline's OSD image.
                sl.draw_result(pl, res)
                # 显示当前的图像及绘制的OSD信息
                # Display the image with the drawn OSD information.
                pl.show_image()
                # 主动调用垃圾回收，释放内存
                # Explicitly call garbage collection to free memory.
                gc.collect()
    except Exception as e:
        # 捕获异常后打印提示信息
        # Print message when an exception is caught.
        print("自学习例程退出")
    finally:
        # 最终退出时执行反初始化
        # Finally, deinitialize the demo.
        exit_demo()

# 退出例程，反初始化资源并清除特征保存文件夹
# Exit routine: cleanup resources and remove stored feature files
def exit_demo():
    global sl, database_path
    # 删除features文件夹下所有文件，并删除文件夹本身
    # Remove all files within the features directory and the directory itself
    stat_info = os.stat(database_path)
    # 判断目录是否为文件夹（目录标志位0x4000）
    # Check if database_path is a directory based on stat info (0x4000 indicates
a directory)
    if (stat_info[0] & 0x4000):
        list_files = os.listdir(database_path)
        for l in list_files:
            os.remove(database_path + l)
    os.rmdir(database_path)
    # 执行自学习实例的反初始化，释放资源
    # Deinitialize the self-learning instance and free up resources.
    sl.deinit()
```

flow chart


开始/Start
初始化阶段/Initialization Phase
初始化自学习实例/Initialize SelfLearningApp
加载模型/Load Model
设置参数/Set Parameters
创建特征存储目录/Create Feature Directory
主循环/Main Loop
预处理流程/Preprocessing
裁剪和缩放/Crop and Resize
运行AI2D预处理/Run AI2D Preprocessing
推理流程/Inference
运行模型推理/Run Model Inference
后处理/Postprocess
特征处理流程/Feature Handling
提取特征/Extract Features
特征匹配/Feature Matching
绘制匹配结果/Draw Matching Results
特征采集/Feature Collection
显示流程/Display
清空显示缓冲/Clear Display Buffer
绘制结果/Draw Results
显示结果/Show Image
垃圾回收/Garbage Collection
获取当前帧/Get Current Frame
否/No
退出检查/Exit Check


计算OSD显示位置/Calculate OSD Positions
初始化AI2D预处理器/Initialize AI2D Preprocessor

是/Yes
清理资源/Cleanup Resources
结束/End