License Plate Recognition

Routine Experiment Effect

In this section, we will learn how to use K230 to realize the license plate recognition function. License plate recognition is based on the license plate detection we talked about in the previous section. When we detect that there is a license plate on the screen, we regard the area of the license plate as the ROI area that needs to be recognized.

Let's use the picture from the previous section as an example:

> 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

1. Initialization phase:
    - Loading detection and recognition models
    - Initialize the character dictionary
    - Setting up the AI2D Processor
    - Configuring Threshold Parameters
2. Main processing flow:
    - Detection stage: Detect the license plate position and crop
    - Recognition stage: preprocessing, model reasoning, character recognition
    - Display stage: drawing detection boxes and recognition results
3. Other treatments:
    - Resource Management
    - Exit Check
    - Results

Code Analysis

For the complete code, please refer to the file [Source Code/09.Scene/07.licence_det_rec.py]

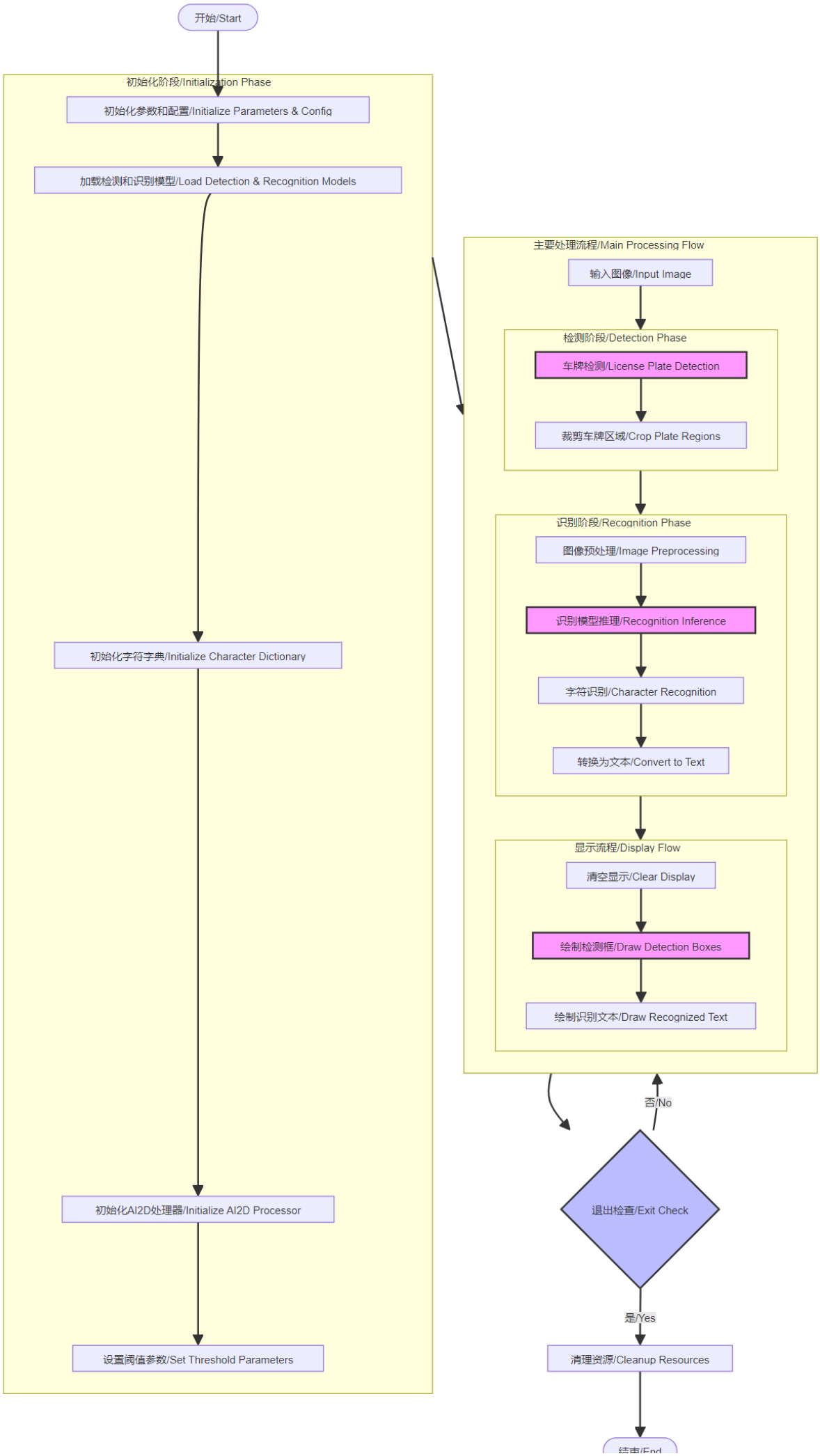
```
class LicenceRec:
    """
    车牌检测和识别的整合类，包含检测和识别两个功能模块
```

```
    Integrated class for license plate detection and recognition, including
detection and recognition modules
    """
    def __init__(self, licence_det_kmodel, licence_rec_kmodel, det_input_size,
rec_input_size,
                 confidence_threshold=0.25, nms_threshold=0.3, rgb888p_size=
[1920,1080],
                 display_size=[1920,1080], debug_mode=0):
        """
        初始化函数
        Initialization function

        参数说明 Parameters:
        licence_det_kmodel: 车牌检测模型路径 Path to license plate detection model
        licence_rec_kmodel: 车牌识别模型路径 Path to license plate recognition
model
        det_input_size: 检测模型的输入尺寸 Input size for detection model
        rec_input_size: 识别模型的输入尺寸 Input size for recognition model
        confidence_threshold: 置信度阈值 Confidence threshold for detection
        nms_threshold: 非极大值抑制阈值 Non-maximum suppression threshold
        rgb888p_size: 输入图像尺寸 Input image size
        display_size: 显示尺寸 Display size
        debug_mode: 调试模式 Debug mode
        """
        # 初始化成员变量 Initialize member variables
        self.licence_det_kmodel = licence_det_kmodel
        self.licence_rec_kmodel = licence_rec_kmodel
        self.det_input_size = det_input_size
        self.rec_input_size = rec_input_size
        self.confidence_threshold = confidence_threshold
        self.nms_threshold = nms_threshold

        # 确保图像宽度是16的倍数 Ensure image width is multiple of 16
        self.rgb888p_size = [ALIGN_UP(rgb888p_size[0],16), rgb888p_size[1]]
        self.display_size = [ALIGN_UP(display_size[0],16), display_size[1]]
        self.debug_mode = debug_mode

        # 初始化检测和识别模型 Initialize detection and recognition models
        self.licence_det = LicenceDetectionApp(
            self.licence_det_kmodel,
            model_input_size=self.det_input_size,
            confidence_threshold=self.confidence_threshold,
            nms_threshold=self.nms_threshold,
            rgb888p_size=self.rgb888p_size,
            display_size=self.display_size,
            debug_mode=0)
        self.licence_rec = LicenceRecognitionApp(
            self.licence_rec_kmodel,
            model_input_size=self.rec_input_size,
            rgb888p_size=self.rgb888p_size)

        # 配置检测模型的预处理 Configure preprocessing for detection model
        self.licence_det.config_preprocess()

    def run(self, input_np):
        """
        执行车牌检测和识别的主要流程
        Main pipeline for license plate detection and recognition
```

```
        参数 Parameters:
        input_np: 输入图像(numpy array格式) Input image in numpy array format

        返回 Returns:
        det_boxes: 检测到的车牌位置 Detected license plate positions
        rec_res: 识别的车牌字符 Recognized license plate characters
        """
        # 执行车牌检测 Perform license plate detection
        det_boxes = self.licence_det.run(input_np)

        # 对检测到的区域进行预处理 Preprocess detected regions
        imgs_array_boxes = aidemo.ocr_rec_preprocess(
            input_np,
            [self.rgb888p_size[1], self.rgb888p_size[0]],
            det_boxes)
        imgs_array = imgs_array_boxes[0]
        boxes = imgs_array_boxes[1]

        # 对每个检测到的车牌进行识别 Recognize each detected license plate
        rec_res = []
        for img_array in imgs_array:
            # 配置预处理参数 Configure preprocessing parameters
            self.licence_rec.config_preprocess(
                input_image_size=[img_array.shape[3], img_array.shape[2]])
            # 执行识别 Perform recognition
            licence_str = self.licence_rec.run(img_array)
            rec_res.append(licence_str)
            gc.collect()  # 垃圾回收 Garbage collection

        return det_boxes, rec_res

    def draw_result(self, pl, det_res, rec_res):
        """
        在图像上绘制检测和识别结果
        Draw detection and recognition results on image

        参数 Parameters:
        pl: PipeLine对象 PipeLine object
        det_res: 检测结果 Detection results
        rec_res: 识别结果 Recognition results
        """
        # 清除上一帧的绘制内容 Clear previous frame drawings
        pl.osd_img.clear()

        if det_res:
            # 创建坐标数组 Create coordinates array
            point_8 = np.zeros((8), dtype=np.int16)

            # 遍历每个检测到的车牌 Iterate through each detected plate
            for det_index in range(len(det_res)):
                # 坐标转换 Coordinate conversion
                for i in range(4):
                    x = det_res[det_index][i * 2 + 0] / self.rgb888p_size[0] *
self.display_size[0]
                    y = det_res[det_index][i * 2 + 1] / self.rgb888p_size[1] *
self.display_size[1]
                    point_8[i * 2 + 0] = int(x)
```

```
                    point_8[i * 2 + 1] = int(y)

                # 绘制检测框 Draw detection box
                for i in range(4):
                    pl.osd_img.draw_line(
                        point_8[i * 2 + 0],
                        point_8[i * 2 + 1],
                        point_8[(i+1) % 4 * 2 + 0],
                        point_8[(i+1) % 4 * 2 + 1],
                        color=(255, 0, 255, 0),
                        thickness=4
                    )

                # 绘制识别结果文本 Draw recognition result text
                pl.osd_img.draw_string_advanced(
                    point_8[6],
                    point_8[7] + 20,
                    40,
                    rec_res[det_index],
                    color=(255,255,153,18)
                )
```

flow chart


开始/Start
初始化阶段/Initialization Phase
初始化参数和配置/Initialize Parameters & Config
加载检测和识别模型/Load Detection & Recognition Models
初始化字符字典/Initialize Character Dictionary
初始化AI2D处理器/Initialize AI2D Processor
设置阈值参数/Set Threshold Parameters
主要处理流程/Main Processing Flow
输入图像/Input Image
检测阶段/Detection Phase
车牌检测/License Plate Detection
裁剪车牌区域/Crop Plate Regions
识别阶段/Recognition Phase
图像预处理/Image Preprocessing
识别模型推理/Recognition Inference
字符识别/Character Recognition
转换为文本/Convert to Text
显示流程/Display Flow
清空显示/Clear Display
绘制检测框/Draw Detection Boxes
绘制识别文本/Draw Recognized Text
否/No
退出检查/Exit Check
是/Yes
清理资源/Cleanup Resources
结束/End