License Plate Detection

Routine Experiment Effect

In this section, we will learn how to use K230 to implement the license plate detection function. License plate detection can be used in many scenarios, especially when combined with license plate recognition, which we will learn in the next section. We connect to the IDE and run the routine to identify the license plate.

【Original image】

【Detection】

> 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:

- Configure basic parameters
- Initializing Pipeline
- Loading the model
- Setting up the AI2D preprocessor

Main loop phase:

- Image acquisition and preprocessing
- Model inference and post-processing
- Results
- Resource Recycling

Exit Processing:

- Abnormality Check
- Resource Cleanup

Code Analysis

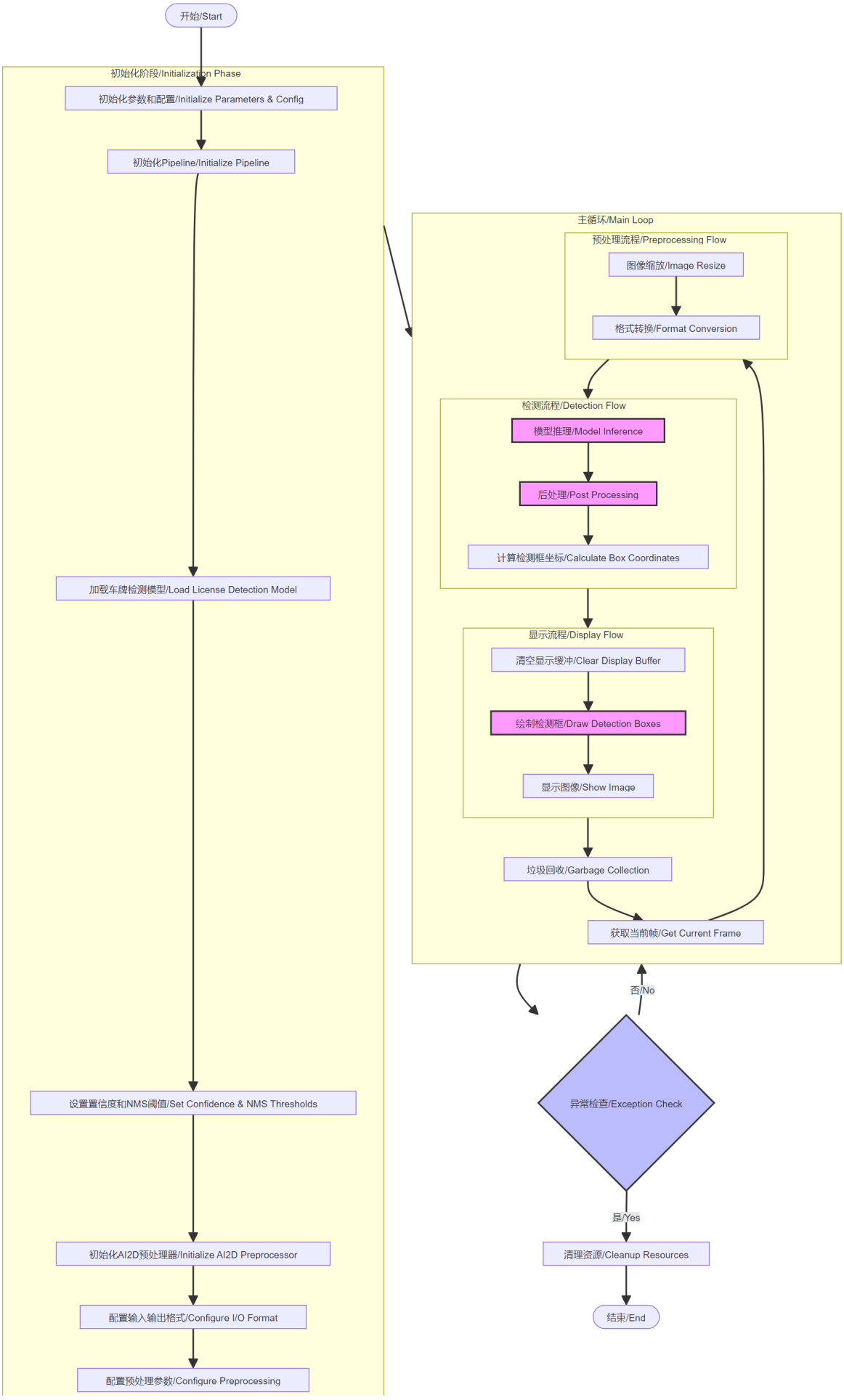
For the complete code, please refer to the file [Source Code/09.Scene/06.licence_det.py]

```
# 车牌检测类 License Plate Detection Class
class LicenceDetectionApp(AIBase):
    """
    车牌检测应用类，继承自AIBase
```

```
    License plate detection application class, inherited from AIBase
    """
    def __init__(self, kmodel_path, model_input_size, confidence_threshold=0.5,
nms_threshold=0.2, rgb888p_size=[224,224], display_size=[1920,1080],
debug_mode=0):
        """
        初始化函数 Initialization function
        参数 Parameters:
            kmodel_path: 模型路径 Model path
            model_input_size: 模型输入尺寸 Model input size
            confidence_threshold: 置信度阈值 Confidence threshold
            nms_threshold: NMS阈值 NMS 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
        """
        super().__init__(kmodel_path, model_input_size, rgb888p_size,
debug_mode)
        self.kmodel_path = kmodel_path
        self.model_input_size = model_input_size
        self.confidence_threshold = confidence_threshold
        self.nms_threshold = nms_threshold
        # 确保宽度是16的倍数 Ensure 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AI2D实例用于图像预处理 Initialize AI2D instance for image
preprocessing
        self.ai2d = Ai2d(debug_mode)
        self.ai2d.set_ai2d_dtype(nn.ai2d_format.NCHW_FMT,
nn.ai2d_format.NCHW_FMT, np.uint8, np.uint8)

    def config_preprocess(self, input_image_size=None):
        """
        配置图像预处理参数 Configure image preprocessing parameters
        """
        with ScopedTiming("set preprocess config", self.debug_mode > 0):
            ai2d_input_size = input_image_size if input_image_size else
self.rgb888p_size
            # 配置双线性插值方法 Configure bilinear interpolation method
            self.ai2d.resize(nn.interp_method.tf_bilinear,
nn.interp_mode.half_pixel)
            self.ai2d.build([1,3,ai2d_input_size[1],ai2d_input_size[0]],
[1,3,self.model_input_size[1],self.model_input_size[0]])

    def postprocess(self, results):
        """
        后处理函数 Postprocessing function
        对模型输出结果进行处理 Process model output results
        """
        with ScopedTiming("postprocess", self.debug_mode > 0):
            det_res = aidemo.licence_det_postprocess(results,
                                                     [self.rgb888p_size[1],
self.rgb888p_size[0]],
                                                     self.model_input_size,
                                                     self.confidence_threshold,
                                                     self.nms_threshold)
```

```
            return det_res

    def draw_result(self, pl, dets):
        """
        绘制检测结果 Draw detection results
        参数 Parameters:
            pl: PipeLine实例 PipeLine instance
            dets: 检测结果 Detection results
        """
        with ScopedTiming("display_draw", self.debug_mode > 0):
            if dets:
                pl.osd_img.clear()
                point_8 = np.zeros((8), dtype=np.int16)
                for det in dets:
                    # 坐标转换 Coordinate conversion
                    for i in range(4):
                        x = det[i * 2 + 0] / self.rgb888p_size[0] *
self.display_size[0]
                        y = det[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)
            else:
                pl.osd_img.clear()
```

flow chart


开始/Start
初始化阶段/Initialization Phase
初始化参数和配置/Initialize Parameters & Config
初始化Pipeline/Initialize Pipeline
加载车牌检测模型/Load License Detection Model
设置置信度和NMS阈值/Set Confidence & NMS Thresholds
初始化AI2D预处理器/Initialize AI2D Preprocessor
配置输入输出格式/Configure I/O Format
配置预处理参数/Configure Preprocessing
主循环/Main Loop
预处理流程/Preprocessing Flow
图像缩放/Image Resize
格式转换/Format Conversion
检测流程/Detection Flow
模型推理/Model Inference
后处理/Post Processing
计算检测框坐标/Calculate Box Coordinates
显示流程/Display Flow
清空显示缓冲/Clear Display Buffer
绘制检测框/Draw Detection Boxes
显示图像/Show Image
垃圾回收/Garbage Collection
获取当前帧/Get Current Frame
异常检查/Exception Check
否/No
是/Yes
清理资源/Cleanup Resources
结束/End