```python
import os
import torch
import cv2
import numpy as np
from ultralytics import YOLO

# 第一步：配置环境+加载YOLOv8s预训练模型
model = YOLO("yolov8s.pt")

# 哈罗单车专属颜色检测（适配实际颜色特征）
def is_hello_bicycle(image, bbox, blue_threshold=0.55):
    """
    判断检测框内的自行车是否为哈罗单车（适配哈罗蓝的主/辅色）
    :param image: 原始图像（BGR格式，OpenCV读取）
    :param bbox: 自行车的检测框 [x1, y1, x2, y2]
    :param blue_threshold: 蓝色像素占比阈值（默认≥55%判定为哈罗单车）
    :return: bool（是否为哈罗单车）、蓝色像素占比
    """
    # 1. 裁剪检测框区域（优化：缩小5%，排除背景干扰）
    h, w = image.shape[:2]
    x1, y1, x2, y2 = map(int, bbox)
    # 轻微缩小检测框，只保留自行车主体（避免背景蓝色干扰）
    shrink = 0.05
    x1 += int((x2 - x1) * shrink)
    y1 += int((y2 - y1) * shrink)
    x2 -= int((x2 - x1) * shrink)
    y2 -= int((y2 - y1) * shrink)
    # 边界保护
    x1 = max(0, x1)
    y1 = max(0, y1)
    x2 = min(w, x2)
    y2 = min(h, y2)
    bike_roi = image[y1:y2, x1:x2]
    if bike_roi.size == 0:  # 裁剪后无像素，直接返回False
        return False, 0.0

    # 2. 转换为HSV色彩空间（减少光照影响）
    hsv_roi = cv2.cvtColor(bike_roi, cv2.COLOR_BGR2HSV)

    # 3. 定义哈罗单车专属的双区间蓝色HSV范围（核心修改）
    # 区间1：哈罗蓝主色（宝蓝/深蓝）
    lower_blue_main = np.array([105, 70, 40])   # 主色下限（宝蓝）
    upper_blue_main = np.array([125, 255, 255]) # 主色上限
    # 区间2：哈罗蓝辅助色（浅青蓝）
    lower_blue_adj = np.array([95, 50, 60])     # 辅助色下限（浅蓝）
    upper_blue_adj = np.array([110, 255, 255])  # 辅助色上限

    # 4. 生成双区间掩码并合并（覆盖所有哈罗蓝）
    mask_main = cv2.inRange(hsv_roi, lower_blue_main, upper_blue_main)
    mask_adj = cv2.inRange(hsv_roi, lower_blue_adj, upper_blue_adj)
    blue_mask = cv2.bitwise_or(mask_main, mask_adj)  # 合并两个区间的蓝色

    # 5. 计算蓝色像素占比（仅自行车主体）
    total_pixels = blue_mask.size
```

```
    blue_pixels = np.count_nonzero(blue_mask)
    blue_ratio = blue_pixels / total_pixels if total_pixels > 0 else 0.0

    # 6. 调试日志（关键：查看实际占比和阈值）
    print(f"【调试】生效阈值：{blue_threshold:.2f} | 哈罗蓝占比：{blue_ratio:.2%}")

    # 7. 判断是否为哈罗单车
    return blue_ratio >= blue_threshold, blue_ratio

# 第二步：哈罗单车检测主函数
def detect_hello_bicycle(image_path, save_result=True, blue_threshold=0.55):
    """
    用YOLOv8s检测哈罗单车（适配哈罗蓝的实际颜色）
    """
    # 检查图片是否存在
    if os.path.isfile(image_path) and not os.path.exists(image_path):
        print(f"错误：未找到图片文件 → {image_path}")
        return None

    # 读取原始图像
    image = cv2.imread(image_path)
    if image is None:
        print(f"错误：无法读取图片 → {image_path}")
        return None

    # 执行YOLO检测（仅检测自行车）
    results = model(
        image_path,
        device="cpu",
        conf=0.25,
        iou=0.45,
        show_labels=False,
        show_conf=False,
        classes=[1]
    )

    # 筛选哈罗单车
    hello_bike_results = []
    result = results[0]
    annotated_img = result.plot()

    if result.boxes is not None:
        for box in result.boxes:
            bbox = box.xyxy[0].tolist()
            conf = box.conf.item()
            cls_name = result.names[int(box.cls.item())]

            if cls_name == "bicycle":
                # 调用适配哈罗蓝的检测函数
                is_hello, blue_ratio = is_hello_bicycle(image, bbox,
blue_threshold)
                if is_hello:
                    hello_bike_results.append({
                        "bbox": bbox,
                        "confidence": conf,
                        "blue_ratio": blue_ratio
```

```
                    })
                    print(f"检测到哈罗单车 → 置信度：{conf:.2f}，哈罗蓝占比：
{blue_ratio:.2%}")

                    # 绘制专属标签（蓝色框+哈罗单车标注）
                    x1, y1, x2, y2 = map(int, bbox)
                    cv2.rectangle(annotated_img, (x1, y1), (x2, y2), (0, 100,
255), 3)  # 哈罗蓝框
                    label = f"哈罗单车 {conf:.2f} (蓝:{blue_ratio:.2%})"
                    label_size = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX,
0.7, 2)[0]
                    label_y = max(y1 - 15, label_size[1])
                    # 绘制标签背景（半透明黑）
                    cv2.rectangle(annotated_img, (x1, label_y - label_size[1] -
8),
                                  (x1 + label_size[0], label_y + 3), (0, 0, 0),
-1)
                    cv2.putText(annotated_img, label, (x1, label_y),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255),
2)
                else:
                    print(f"普通自行车 → 置信度：{conf:.2f}，哈罗蓝占比：
{blue_ratio:.2%}（<55%，已过滤）")

    # 显示结果
    cv2.imshow("哈罗单车检测结果（适配专属色）", annotated_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    # 保存结果
    if save_result and len(hello_bike_results) > 0:
        save_dir = os.path.join("runs", "detect", "hello_bike_results_v8s")
        os.makedirs(save_dir, exist_ok=True)
        save_path = os.path.join(save_dir, os.path.basename(image_path))
        cv2.imwrite(save_path, annotated_img)
        print(f"哈罗单车检测结果已保存 → {save_path}")
    elif save_result:
        print("未检测到哈罗单车，无需保存结果")

    return hello_bike_results
```

```
IMAGE_PATH = "C://Users//hp//Desktop//bicycle_image.png"  # 替换为你的图片路径
detect_hello_bicycle(
    image_path=IMAGE_PATH,
    save_result=True,
    blue_threshold=0.35  # 强制生效0.55阈值
)
```

```
image 1/1 C:\Users\hp\Desktop\bicycle_image.png: 288x640 11 bicycles, 418.2ms
Speed: 4.3ms preprocess, 418.2ms inference, 3.7ms postprocess per image at shape
(1, 3, 288, 640)
```

```
【调试】生效阈值：0.35 | 哈罗蓝占比：13.18%
普通自行车 → 置信度：0.89，哈罗蓝占比：13.18%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：40.73%
检测到哈罗单车 → 置信度：0.82，哈罗蓝占比：40.73%
【调试】生效阈值：0.35 | 哈罗蓝占比：14.58%
普通自行车 → 置信度：0.79，哈罗蓝占比：14.58%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：16.53%
普通自行车 → 置信度：0.79，哈罗蓝占比：16.53%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：14.91%
普通自行车 → 置信度：0.72，哈罗蓝占比：14.91%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：31.57%
普通自行车 → 置信度：0.63，哈罗蓝占比：31.57%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：17.64%
普通自行车 → 置信度：0.63，哈罗蓝占比：17.64%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：16.52%
普通自行车 → 置信度：0.58，哈罗蓝占比：16.52%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：11.38%
普通自行车 → 置信度：0.58，哈罗蓝占比：11.38%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：24.76%
普通自行车 → 置信度：0.53，哈罗蓝占比：24.76%（<55%，已过滤）
【调试】生效阈值：0.35 | 哈罗蓝占比：9.51%
普通自行车 → 置信度：0.33，哈罗蓝占比：9.51%（<55%，已过滤）
哈罗单车检测结果已保存 → runs\detect\hello_bike_results_v8s\bicycle_image.png
```

```
[{'bbox': [598.1624755859375,
   169.1371307373047,
   662.0811767578125,
   356.8822021484375],
  'confidence': 0.8208448886871338,
  'blue_ratio': 0.4073022312373225}]
```

```
```