**1. 项目背景**

**项目名称**：交通流量监控系统

**项目目标**：通过计算机视觉技术（YOLO、ByteTrack等），实现对交通流量的实时监控与分析。项目目标包括目标检测、追踪与计数，并通过Web界面进行可视化展示。

**2. 工具与环境配置**

**注：所有代码均为ChatGPT所提供，不同系统或其他情况下可能无法使用**

**无论你使用什么操作系统，建议你使用Python虚拟环境来隔离项目依赖，以防与系统环境冲突。——ChatGPT**

**2.1 所需工具及库**

1. **OpenCV**：

**功能**：用于实时目标检测的最新YOLO模型，YOLOv8相比之前的版本具有更高的准确性和速度。

**安装代码**：

pip install opencv-python

1. **YOLOv8 (You Only Look Once)**：

**功能**：用于实时目标检测的深度学习模型。

**安装代码**：

pip install ultralytics

1. **ByteTrack**：

**功能**：用于目标追踪，实现交通参与者的连续追踪。

**安装代码**：

git clone https://github.com/ifzhang/ByteTrack.git

cd ByteTrack

pip install -r requirements.txt

某些依赖包（如 numpy）要求 **Python 3.9 或更高**，还有一些需要 **Python 3.10** 或更高版本。——**ChatGPT**

1. **Supervision**：

**功能**：用于目标计数和可视化，便于展示交通流量统计结果。

**安装代码**：

pip install supervision

1. **Streamlit**：

**功能**：用于开发Web界面，展示原始视频和处理后的视频。

**安装代码**：

pip install streamlit

**3. 技术学习与文档资源**

**3.1 YOLOv8技术学习**

**模型简介**：YOLO是一种高效的实时目标检测算法，能够在单次推理中预测多个目标的类别和边界框。

**学习资源**：

[YOLOv8官方文档](https://docs.ultralytics.com) （）

YOLOv8代码结构：

* + - models/：包含YOLOv5模型架构的代码。
    - data/：存放数据集的配置文件。
    - detect.py：YOLOv8的推理脚本，用于在视频或图像上进行目标检测。

**快速示例**：

（python）

from ultralytics import YOLO

model = YOLO('yolov8n.pt') # 加载YOLO模型

results = model('input\_image.jpg') # 对图像进行目标检测

results.show() # 展示检测结果

* **YOLOv8模型类型**：
  + yolov8n.pt：nano版本，适用于对速度要求较高的任务
  + yolov8s.pt：small版本，平衡速度与精度
  + yolov8m.pt：medium版本，适用于精度要求更高的任务
  + yolov8l.pt：large版本，提供最高精度，但推理速度稍慢

**3.2 ByteTrack技术学习**

* **算法简介**：ByteTrack是一种基于目标检测结果进行多目标追踪的算法，能够实现交通参与者在不同帧中的连续追踪。
* **学习资源**：
  + [ByteTrack官方文档](https://github.com/ifzhang/ByteTrack)
  + **ByteTrack工作流程**：
    1. 目标检测（YOLO） -> 2. ByteTrack追踪 -> 3. 轨迹生成
  + **示例代码**：

python3 tools/track.py --source path\_to\_video --yolo-weights yolov8n.pt

**3.3 OpenCV基础**

* **简介**：OpenCV是一个计算机视觉库，用于读取、处理和显示视频流。
* **常用操作**：
  + 读取视频流：

（Python）

import cv2

cap = cv2.VideoCapture('video.mp4') # 打开视频文件

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

cv2.imshow('Frame', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

* + 捕获摄像头数据：

（Python）

cap = cv2.VideoCapture(0) # 打开摄像头

**4. GitHub项目管理**

**4.1 仓库创建与设置**

1. **创建仓库**：
   * 使用GitHub创建新的项目仓库，名称为“Traffic-Monitoring-System”。
   * 初始化README.md文件，描述项目背景和目标。
2. **分支管理**：
   * 使用 main 分支进行稳定代码的维护。
   * 使用 dev 分支进行开发过程中的代码提交。
3. **团队协作流程**：
   * 每个组员通过创建独立的功能分支进行开发（如 feature/yolo-detection），完成开发后通过Pull Request合并到 dev 分支。
   * 代码通过审核后，再合并到 main 分支。

**4.2 Git基础操作**

1. **克隆仓库**：git clone https://github.com/username/Traffic-Monitoring-System.git
2. **创建分支并提交代码**：

git checkout -b feature/your-feature

git add .

git commit -m "Added YOLO detection"

git push origin feature/your-feature

1. **Pull Request**： 在GitHub页面上发起Pull Request，等待组长或其他成员审核代码并合并到 dev 分支。
2. **合并代码**： 确保没有冲突后，将代码合并到主分支。