第三次个人赛 A 题 临时编号：87

智能视频监控分析与车辆计数模型

摘要

在交通管理中，带有智能分析功能的监控系统可以通过区分车辆的外形、动作等特征，做到主动收集、分析统计数据等功能。

对于题目提供的一段监控视频画面，本文首先对其进行预处理，在画面中以左上角为原点**建立直角坐标系**并**提取 7664 帧画面**，然后**基于 LAB 颜色空间调整画面的白平衡**，使用**暗通道去雾**算法对画面进行去雾，利用**双边滤波**去除噪声的同时保留画面的边缘细节，最终得到了 7644 帧处理后的画面，并**重新合成为视频**，方便之后模型的建立。

对于问题一，本文利用 **YOLOv5 算法**，**选择合适的置信度**识别车辆，并通过**反归一化**计算车辆实际坐标，再**根据识别类型和位置过滤噪声产生的错误识别**，仅保留汽车类且出现在道路上的识别结果，成功地**从每一帧画面中识别并提取出所有的车辆位置与大小**，完成了视频的数学模型的构建。

对于问题二，本文在问题一已经建立好的模型基础上，创新的提出了**贪心匹配策略**，**在滑动窗口内基于重合面积以匹配并跟踪车辆**。根据对画面中的车辆按照其位置、速度和大小进行分类的结果，**选择滑动窗口大小为 12 帧、重合面积阈值为 20 平方像素**，以取得最佳跟踪成功率。由于本文模型采用了先进的 YOLO 算法，故选择简单的**检测线方法统计车流量**。在尽量靠近画面中间的情况下，**选择 $y=-7x+1680$ 这条直线作为检测线**，最终在完整视频中**识别到了 275 次车辆跨线行为**。根据**车流量计算公式**，计算出该道路上**车流量为平均每小时 3237.8 辆车**，并取一定时间间隔，作出**车辆数量直方统计图**，实现**可视化分析**。

本文通过**手工计数三次并取平均数**，得到视频中**跨线车辆实际有 256 辆**，本文模型得到的结果与之相比多计算了 19 辆，**误差约为 7.4%**，并提出了三类可能造成误差的原因。最后，本文讨论了该模型的优缺点和改进方向与推广功能。

关键词：YOLOv5 车辆识别跟踪 车流量计算 贪心匹配策略 暗通道去雾 双边滤波

目录

一、 问题背景与重述

1.1 问题背景

视频监控系统是由摄像机、视频录像设备、监视器、网络设备和相关软件组成的一套系统，用于实时监视、记录和管理特定区域或场所的视频图像和数据。它通常用于安全监控、防盗、交通监管、员工监管、公共安全等领域。

传统的视频监控系统具有其固有的缺点，难以实现实时的安全监控和检测管理。在当今快速发展的科技时代，视频智能分析技术（Intelligent Video Analysis）已经成为提升安全生产水平的重要手段。这一技术通过计算机图像视觉分析技术，结合 AI 视觉算法，实现对视频数据的智能分析与处理[1]，实现了对场景中目标的自动识别和追踪，检测效率大幅提高，在检测对象密集、昏暗、远距离和有遮挡等复杂场景下依然能保持良好的准确率。

在交通管理中，带有智能分析功能的监控系统可以通过区分监控对象的外形、动作等特征，做到主动收集、分析数据，并根据预设条件执行报警、记录、分析等动作。

1.2 问题重述

视频分析的关键技术是建立合理的数学模型。题目提供了一段道路监控视频，要求完成下列问题：

（1） 根据提供的视频，建立视频图像的数学模型。

（2） 结合第一问提取出的背景信息，建立计算车流量的数学模型。

二、 问题分析

该问题要求对于一段监控视频画面进行分析并按照两问要求分别建立数学模型。对于监控视频画面，可能由于其画面清晰度较低，同时监控与汽车距离比较远，所以必须先对视频数据进行预处理，方便之后的信息采集与模型建立。常见的数据预处理方法有调整白平衡、直方图均衡、图像滤波等，可以根据之后建立的模型进行合适的预处理方法。

针对问题一：需要建立视频图像的数学模型。可以使用 YOLO（You Only Look Once）

系列计算机视觉模型或者背景差分法，对视频画面中的车辆进行寻找、定位。如果出现错误的识别结果，可能还需要通过位置、大小等特征对于识别结果进行过滤。

针对问题二：需要建立画面中车流量统计的数学模型。车流量是指单位时间内通过某路段的车辆数，单位通常为“辆/小时”。可以使用一定的方法对车辆进行跟踪，然后使用检测线、检测区等方法，对车辆进行计数，最后统计并计算出平均一小时内的车流量数据。

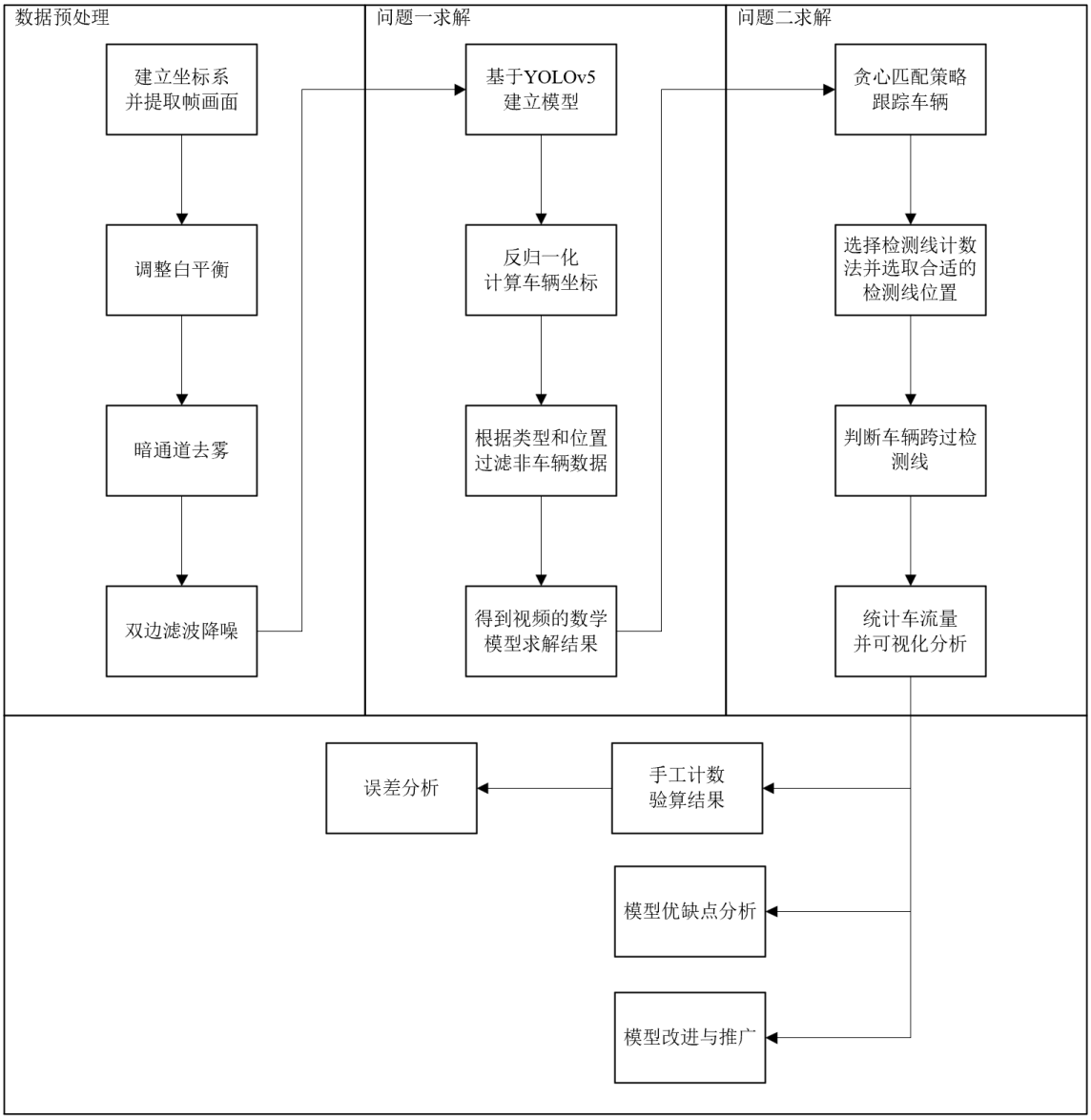


图 1 流程图

三、 问题假设

（1） 假设车辆在视频画面中是连续变化的，不存在视频画面残缺、跳跃的情况。

（2） 假设视频画面稳定无晃动，背景稳定，不会因为来往车辆影响而发生震动。

（3） 假设在视频时间内，光照、天气不会发生剧烈的变化。

（4） 假设道路上车流量密度适中，很少出现车辆紧挨的情况。

（5） 假设道路上除了车辆没有其他运动物体。

四、 符号说明

变量	含义	单位
H	视频帧高度	像素
W	视频帧宽度	像素
$A_{x,y}$	LAB 色彩空间下坐标 (x,y) 处像素的 A 值	/
$B_{x,y}$	LAB 色彩空间下坐标 (x,y) 处像素的 B 值	/
$\overline{A}$	LAB 色彩空间下图像所有像素 A 值的平均值	/
$\overline{B}$	LAB 色彩空间下图像所有像素 B 值的平均值	/
$A'_{x,y}$	白平衡调整后 LAB 色彩空间下坐标 (x,y) 处像素的 A 值	/
$B'_{x,y}$	白平衡调整后 LAB 色彩空间下坐标 (x,y) 处像素的 B 值	/
J^{dark}	有雾图像的暗通道	/
$t(x)$	有雾图像的传输图	/
J^c	去雾图像的景深图	/
T	滑动窗口宽度	帧
D	重合面积阈值	像素^2
k	检测线函数式的一次项系数	/
b	检测线函数式的常数项	/
TV	车流量	辆/小时

五、 模型建立与求解

5.1 数据预处理

题目附件所提供的视频为 2011 年 4 月 13 日星期三中午 11 时 58 分 44 秒至 12 时 03 分 50 秒，某高速公路路口监控画面。视频全长 5 分 5 秒，帧宽度 352 像素，帧高度 288 像素。视频文件大小 27.8MB，码率 1000kbps，视频帧速率 25fps，无声音。共计 7644 帧画面。

5.1.1 建立画面的直角坐标系并提取帧画面

由于在计算机中，画面信息是从左上角开始储存，并且以行为第一维度、以列为第二维度，故我们将画面左上角为原点，以正下为 x 轴正方向，以正右为 y 轴正方向，建立平面直角坐标系，如图 2 所示。

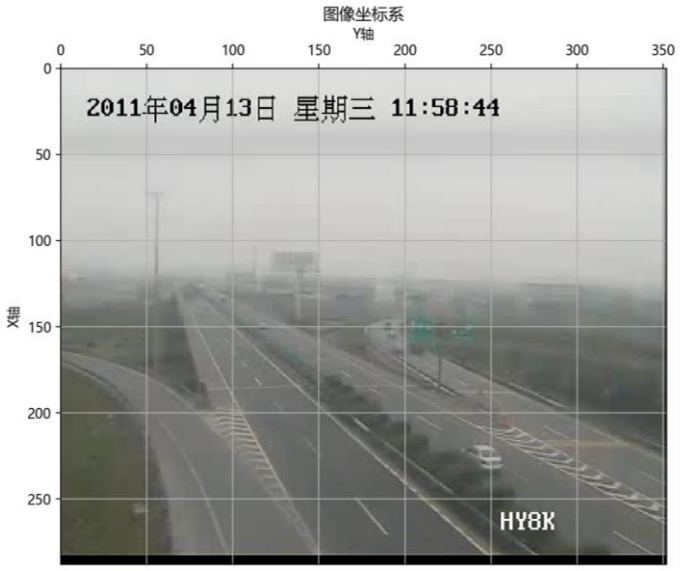


图 2 监控画面的平面直角坐标系示意图

在计算机中，视频由连续的画面构成。本文从原视频文件中提取出共计 7644 帧画面，每 25 帧画面对应了视频文件中的 1 秒钟，并按照上述坐标系规则与 RGB（红绿蓝）三种颜色存储画面信息。

5.1.2 白平衡调整

白平衡[2]是描述显示器中红、绿、蓝三基色混合生成后白色精确度的一项指标。调整白平衡的目的是为了在不同光线环境下还原真实的色彩。不同的光源具有不同的色温，

例如日光下的色温较高，而钨丝灯（白炽灯）下的色温较低。

调整白平衡一般通过调整画面中 R/G/B 分量的比例关系，以准确再现白色。计算机自动调整白平衡一般使用基于 LAB 颜色空间的调整方法，具体步骤如下：

Step1 颜色空间转换

将图像从 RGB 颜色空间转换到 LAB 颜色空间，其中 L 是亮度层，A、B 为色彩层，A 层从绿色到红色，B 层从蓝色到黄色。

从 RGB 色彩空间到 LAB 色彩空间的转换需要经过 XYZ 色彩空间，其转换公式[3]为：

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

再转换至 LAB 色彩空间：

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2}$$

Step2 计算整体偏色

计算 LAB 图像的 A 通道和 B 通道的平均值：

$$\overline{A} = \sum_{x=1}^{H} \sum_{y=1}^{W} A_{x,y} \tag{2}$$

$$\overline{B} = \sum_{x=1}^{H} \sum_{y=1}^{W} B_{x,y} \tag{3}$$

A 通道与 B 通道的平均值表示图像在这两个色彩维度上的整体偏色。

Step3 调整 A 和 B 通道

基于 A 和 B 通道的平均值与中性点（128）之间的差异，对每个像素在 A 和 B 通道上根据其亮度层的值进行调整：

$$A'_{x,y} = \left(\overline{A} - 128\right) \times \frac{L_{x,y}}{255} \times 1.1 \quad x \in [1, H], y \in [1, W] \tag{4}$$

$$B'_{x,y} = \left(\overline{B} - 128\right) \times \frac{L_{x,y}}{255} \times 1.1 \quad x \in [1, H], y \in [1, W] \tag{5}$$

其中，$\left(\overline{A} - 128\right), \left(\overline{B} - 128\right)$是 A 通道与 B 通道相对于中性灰色的偏差

$L_{x,y}$是(x, y)处像素点的亮度值

Step4 颜色空间转换回 RGB

将调整后的 LAB 图像转换回常用的 RGB 颜色空间，即进行公式(1)和公式(2)的逆运算，以便于显示与接下来的处理。

可以通过观察视频画面与其 R/G/B 分量直方图，来对比进行白平衡调整之后的画面效果。下图 3 以视频画面的第一帧为例，其原视图像的$\overline{A} = 126.71 < 128$，$\overline{B} = 130.08 > 128$，说明画面偏黄绿。

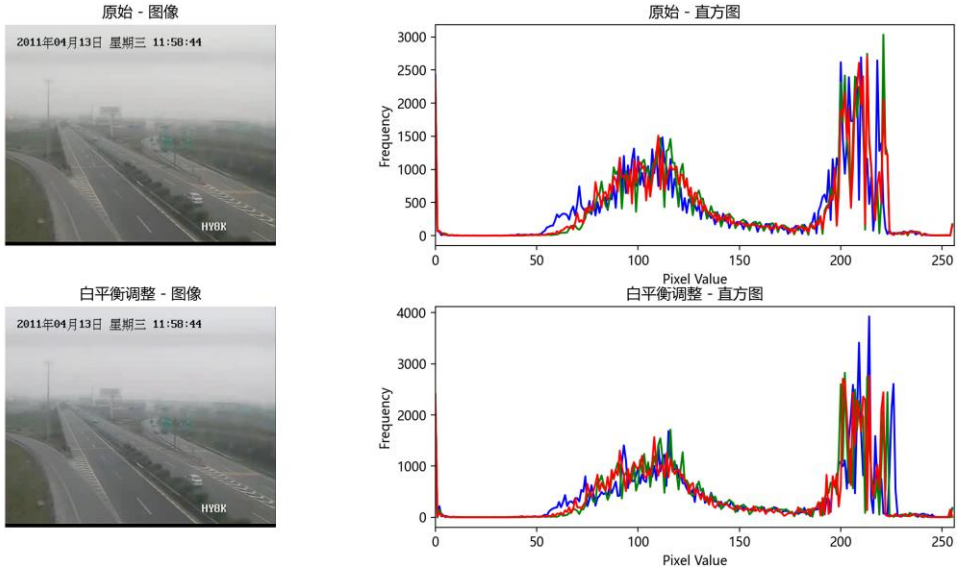


图 3 白平衡调整前后图像及直方图展示

对比两图颜色和直方图可以发现，基于 LAB 色彩空间的白平衡调整很好的解决了原画面的偏色问题。

5.1.3 暗通道去雾

观察视频画面，发现天气为阴天，且画面呈雾状模糊，需对图像进行去雾处理。

暗通道去雾[4]算法是一种基于统计学原理的图像处理技术，由何恺明等人提出。该算法基于一个现象：在绝大多数的非天空的局部区域中，某一些像素总会有至少一个颜色通道（红、绿、蓝）会具有很低的值。这一现象被称为“暗通道先验”。利用这一先验，可以估计出图像中的雾的厚度，并据此恢复无雾的图像。下图 4 为去雾前后对比图。

图 4 暗通道去雾算法去雾前后图像对比

具体去雾步骤如下：

Step1 暗通道计算

定义暗通道 J^{dark} 为原始有雾图像 I 的每个像素在局部窗口内各色彩通道的最小值的最小值：

$$J^{dark}(x) = \min_{y \in \Omega(x)} \left(\min_{c \in \{r,g,b\}} I^c(y) \right) \tag{6}$$

其中，x 是像素位置

$\Omega(x)$ 是以 x 为中心的局部窗口

c 表示颜色通道。

Step2 大气光 A 的估计

选择暗通道图中亮度最高的前 0.1% 的像素，然后在原图中找到这些位置的最亮像素作为大气光 A。

Step3 传输图 $t(x)$ 的估计

传输图描述了每个像素位置的光线被大气散射的程度，计算方法为：

$$t(x) = 1 - \omega \min_{y \in \Omega(x)} \left(\min_{c \in \{r,g,b\}} \frac{I^c(y)}{A^c} \right) \tag{7}$$

其中，ω 是一个保留常数（通常取值约为 0.95），用以保留一些雾的自然外观。

Step4 景深图的恢复

利用传输图和大气光，可以恢复无雾的景深图 J^c：

$$J^c(x) = \frac{I^c(x) - A^c}{\max(t(x), t_0)} + A^c \tag{8}$$

其中，t_0 是一个防止分母为零的小常数，通常取 0.1。

下图 5 中展示了经过白平衡调整后的视频画面第一帧图像进行暗通道去雾前后的图像与直方图，并与原图进行对比。

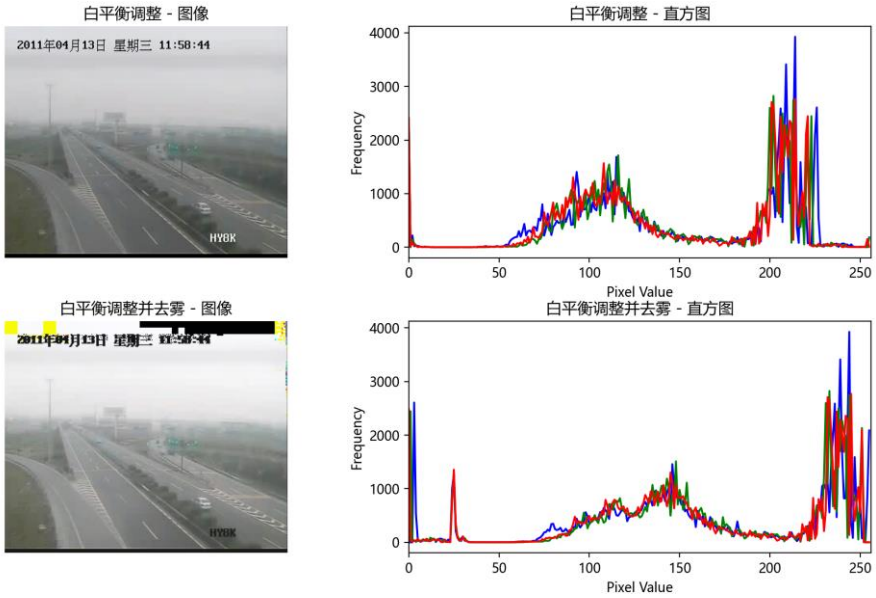


图 5 暗通道去雾效果展示

对比两图颜色和直方图可以发现，暗通道去雾可以使画面清晰，色彩更加丰富，有利于之后的处理。虽然暗通道去雾之后的画面中，可能受到监控画面中文字信息的影响，天空出现了部分黑色色块，但是由于均出现在画面的上半部分，不会遮挡道路汽车，所以该问题可以接受。

5.1.4 双边滤波降噪

双边滤波（Bilateral Filter）是一种非线性滤波技术，能够在去除噪声的同时保留图像的边缘细节。它在计算时考虑了空间邻域和像素值强度的相似性，因此在平滑图像的同时不会模糊边缘，所以相对于其他滤波降噪算法，它不容易将其中较为模糊的汽车图像信息过滤，不会显著降低之后识别的正确率。

双边滤波的核心思想是结合空间域和颜色域的高斯滤波：

1. 空间域滤波：考虑像素在空间上的距离，距离越远的像素对中心像素的影响更小。

2. 颜色域滤波：考虑像素值的相似性，颜色值相近的像素对中心像素的影响更大。双边滤波的输出像素值是输入像素值的加权平均，权重由空间距离和颜色差异共同决定，其公式如下：

$$I_{\text{filtered}}(x) = \frac{1}{W_p} \sum_{y \in \Omega} I(y) \cdot e^{-\frac{|x-y|^2}{2\sigma_s^2}} \cdot e^{-\frac{|I(x)-I(y)|^2}{2\sigma_r^2}} \tag{9}$$

其中：$I_{\text{filtered}}(x)$ 是滤波后的像素值。

$I(y)$ 是邻域 Ω 中的像素值。

$|x-y|$ 是空间距离。

$|I(x) - I(y)|$ 是颜色差异。

σ_s 是空间域的标准差。

σ_r 是颜色域的标准差。

W_p 是归一化因子，确保权重和为 1。

下图 6 中展示了经过白平衡调整和暗通道去雾后的视频画面第一帧图像进行双边滤波前后的图像与直方图，并与原图进行对比。

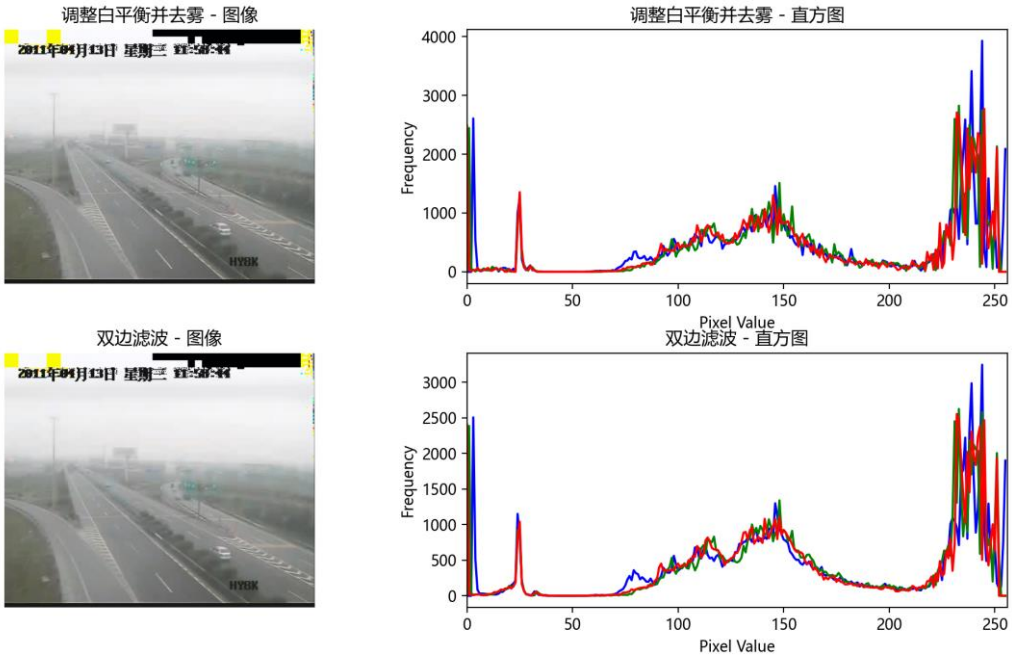


图 6 双边滤波效果展示

观察直方图可以发现，双边滤波在一定程度上减少了直方图中的峰值，使曲线更加平滑，降低了颜色域上的噪音。而图像上的差别难以看清，需要通过放大对比，故选取该帧画面上的白线和车辆进行放大对比。

选择以(148,245)为左上角、以(215,280)为右下角的矩形，对双边滤波前后的图像进行放大，如图 7 所示：

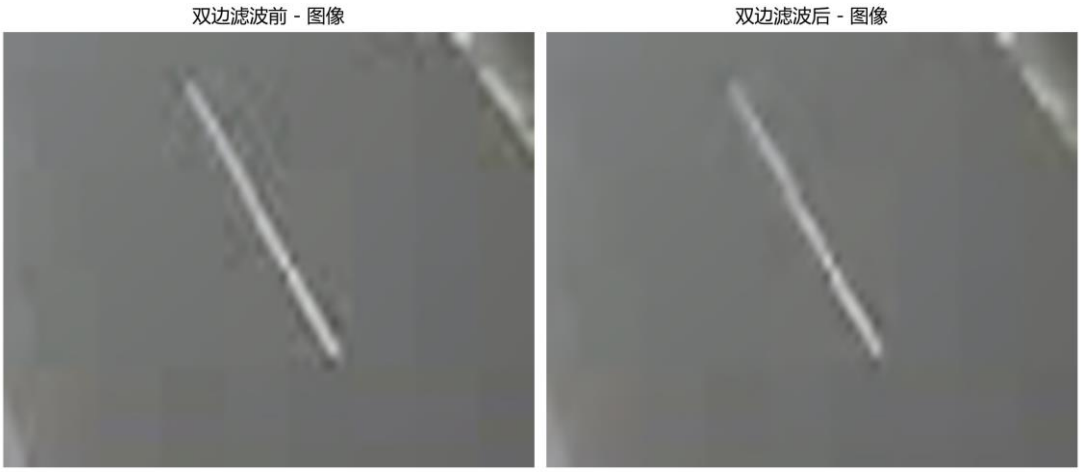


图 7 双边滤波效果对比——车道划分线

选择以(288,211)为左上角、以(238,260)为右下角的矩形，对双边滤波前后的图像进行放大，如图 8 所示：

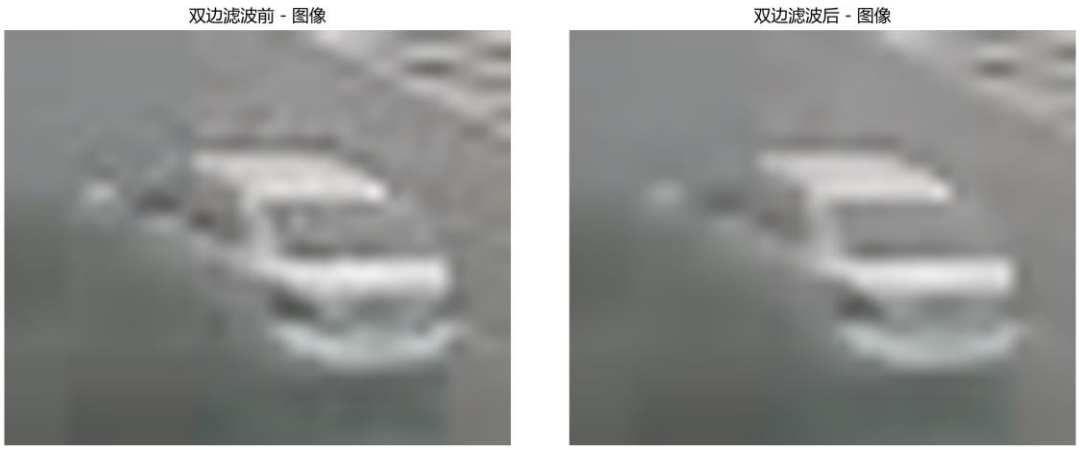


图 8 双边滤波效果对比——白色轿车

可以显著地发现，经过双边降噪之后，图像中车道划分线和白车周围的噪点明显减少，同时完整保留了白线和白车的形状与轮廓。

5.1.5 数据预处理总结

在数据预处理部分，本文在建立画面平面直角坐标系并提取所有帧画面的基础上，对所有帧画面调整了白平衡以还原图像正常色彩，使用了暗通道去雾使图像更加清晰、基于双边滤波对图像除去噪声，并且每一步都做出了与前一步的对比图，最终获得了 7644 帧预处理之后的画面，并重新合成为视频，方便之后模型的建立与求解。效果图总结展示如图 9 所示。

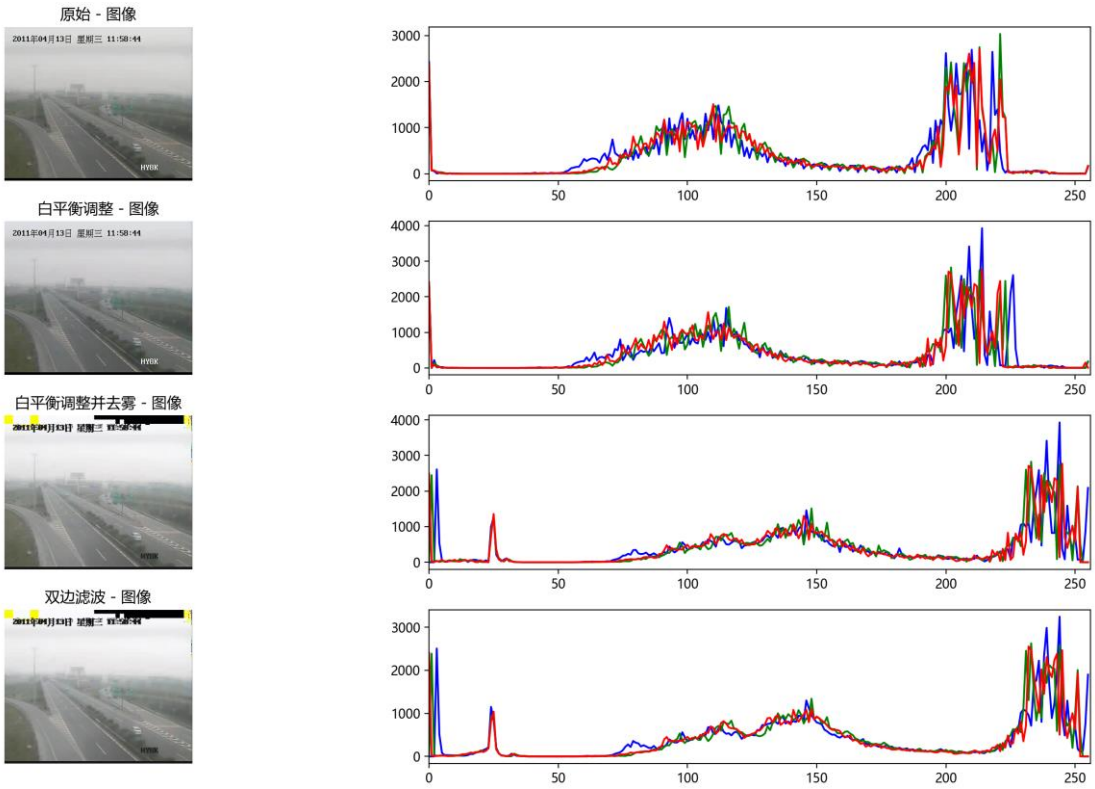


图 9 数据预处理汇总展示图

5.2 问题一模型建立与求解

5.2.1 YOLOv5 网络结构概述

YOLOv5 目标检测算法[5]主要由骨干网络（Backbone）、颈部网络（Neck）、头部网络（Head）三部分组成，结构图如图 10 所示。骨干网络对输入的图像进行提取特征，接着利用颈部网络来融合不同层次的特征，从而获得更丰富的语义信息，并得到三个不同尺度大小的特征图。最后使用头部网络中的三个检测头来对大、中、小三类目标进行预测。

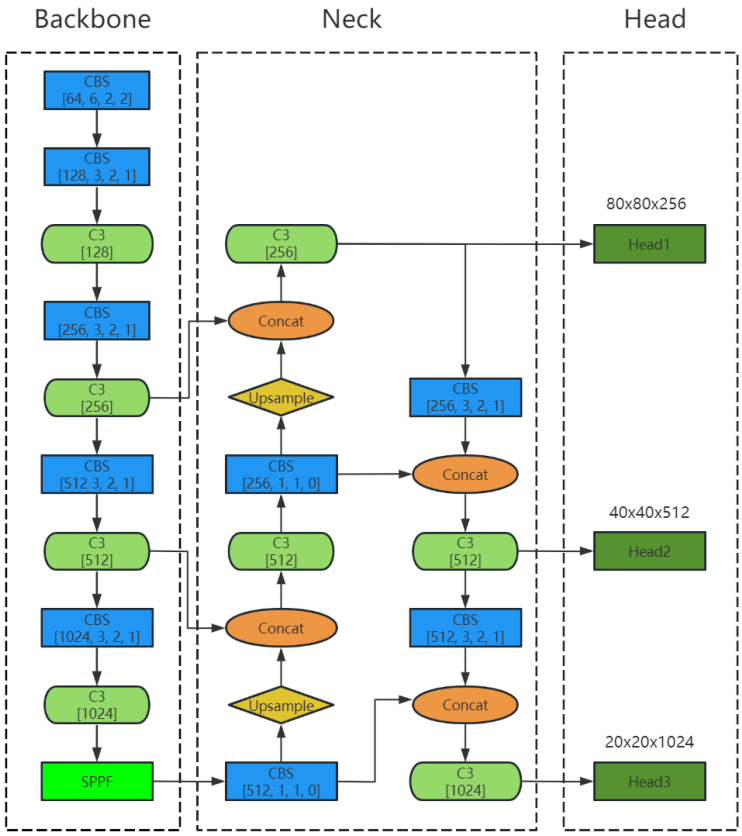


图 10 YOLOv5 模型结构

（1） 骨干网络

骨干网络如图 10 中 Backbone 部分所示，首先输入维度为$640 \times 640 \times 3$的图片，经过两层 CBS 模块后，输出$160 \times 160 \times 128$的特征图。图中的每个 CBS 模块都包含一个四维的向量作为参数，分别代表 CBS 模块中卷积层的输出通道数 c，卷积核大小 k，步长 s，填充 p。CBS 模块由卷积层、批量归一化层、SiLU（Sigmoid Gated Linear Unit）激活函数三部分所组成。其中 SiLU 激活函数是在 Sigmoid 激活函数的基础上增加了线性部分，使其能更好地对线性函数进行拟合，同时还可以有效减少网络在训练过程中发生过拟合的情况。SiLU 激活函数计算公式如公式(10)、(11)所示：

$$f(x) = x \cdot \sigma(x) \tag{10}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

其中：$\sigma(x)$ 为 Sigmoid 激活函数

当输入值 x 大于 0 时，SiLU 函数的输出曲线与 ReLU 函数输出曲线相似；当输入 x 小于 0 时，SiLU 函数的输出曲线形状类似 Sigmoid 函数，但是输出值小于 0。

（2） 颈部网络

YOLOv5 的颈部网络利用 PANet（Path-Aggregation Network，路径聚合网络）结构将浅层和深层的语义信息进行融合，PAN 结构是在 FPN[6]（特征金字塔）的基础上自顶向下地对特征进行融合，YOLOv5 中的 PANet 结构如图 11 所示。

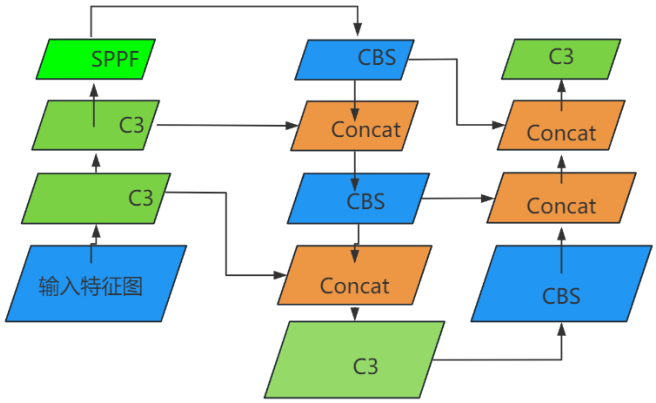


图 11 YOLOv5 中的 PANet 结构

在对图像进行特征提取时，由于浅层网络的感受野较小，提取到的特征中更多包含的是诸如颜色、边缘、轮廓、纹理等细粒度的信息，适合用于小尺度目标检测。随着网络结构的加深，所提取到的深层特征中就包含更多有关图像整体的信息，对大尺度目标的识别能力也就更强。因此将浅层和深层的语义信息进行融合可以使网络检测到更多不同尺度的目标。

（3） 头部网络

YOLOv5 的头部网络中有三个不同尺度的 Anchor，分别用来预测大中小三类目标。在预测层中使用 3 个通道数为 $(4+1+nc)\times na$ 的1×1卷积来分别对这 3 个不同尺度的特征图进行预测。其中：

(a) 4 代表边界框的中心点坐标以及长和宽四种信息；

(b) 1 是指置信度信息；

(c) nc 表示需要检测的目标种类个数，在 COCO 数据集中包含 80 个类别；

(d) na 表示每种尺度的特征图上预设边界框的个数，其中 $na=3$。

因此每个1×1卷积的通道数都为 255。图 10 中的 Head 1 是对维度为$80\times 80\times 256$的特征图进行处理，用来预测小目标；Head 2 处理的特征图维度为$40\times 40\times 512$，用来预测中目标；Head 3 则是处理维度为$20\times 20\times 1024$的特征图，用来预测大目标。

（4） 损失函数

YOLOv5 的总损失包含三部分：分类损失、置信度损失、定位损失。总损失函数计算公式为：

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \tag{12}$$

其中，L_{cls} 为分类损失

L_{obj} 为置信度损失

L_{loc} 为定位损失

(a) 分类损失

采用 BCE Loss（Binary Cross Entropy Loss，二值交叉熵损失）。在计算时并不需要将每个 Grid Cell 的分类损失都进行累加，而是只对预测物体的 GridCell 进行分类损失计算。BCE Loss 计算公式为：

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \times \log\big(p(y_i)\big) + (1-y) \times \log\big(1-p(y_i)\big) \tag{13}$$

其中，y 为二元标签

N 为标签个数

$p(y_i)$ 为输出值等于标签 y 的概率

(b) 置信度损失

与分类损失一样，同样采用二值交叉熵损失，3 个不同尺度（small、medium、large）的预测特征层上的置信度损失使用不同的权重。但与分类损失不同的是，需要对每个 Grid Cell 都进行置信度损失计算，其中不参与预测物体的 Grid Cell 的置信度标签为 0，而负责预测物体的 Grid Cell 的置信度标签为预测边界框（bounding box）与真实框（ground truth）的 IOU（Intersection over Union，交并比）。

$$L_{obj} = 4.0L_{obj}^{\text{small}} + 1.0L_{obj}^{\text{medium}} + 0.4L_{obj}^{\text{large}} \tag{14}$$

其中，L_{obj}^{small} 为小尺度预测特征层的置信度损失

L_{obj}^{medium} 为中尺度预测特征层的置信度损失

L_{obj}^{large} 为大尺度预测特征层的置信度损失

(c) 定位损失

定位损失函数采用 CIOU Loss。CIOU Loss 考虑了两个框之间的重叠面积、中心点距离、长宽比三个因素。CIOU Loss 计算公式为：

$$L_{CIOU} = 1 - CIOU \tag{15}$$

其中，CIOU 用于衡量预测框和真实框之间的重叠程度。

5.2.2 选择合适的置信度

由于视频中的车辆较为模糊，我们需要选择合适的置信度。置信度过高会导致难以识别出视频中的物体（车辆），导致漏检率增加；置信度过低会导致识别出不正确的对象，增加了误检率。

使用公开的预训练模型，选择视频中比较复杂的几帧画面进行解算目标位置，结果如图 12 所示。

第 1216 帧　　第 1279 帧　　第 1391 帧

图 12 部分画面目标位置解算

可以发现，车辆的置信度一般在 0.5 以上，如第 1216 帧画面中的客车和轿车的置信度均为 0.64。而错误识别的置信度一般在 0.1 以下，如第 1216 帧画面右侧的错误识别置信度为 0.06，第 1279 帧画面两处错误识别置信度分别为 0.05 与 0.08。而观察右图可以发现，由于原视频过于模糊，所以对于较远的车辆识别效果可能不太好，例如第 1391 帧画面左侧货车和中间的轿车识别置信度分别只有 0.13 和 0.11。

综合上述因素，本文将临界置信度设置为 0.1，即仅认为置信度高于 0.1 的为车辆，置信度低于 0.1 的不纳入接下来的考虑范畴。虽然将置信度临界值设置偏低会导致误判率增加，但是我们可以通过之后的步骤过滤优化。

5.2.3 反归一化计算坐标

YOLOv5 算法检测车辆位置，最终会得到经过归一化的检测框中心点坐标与矩形的

长宽，记作x', y', a', b'。实际矩形的中心点坐标与长宽记作x, y, a, b。

根据归一化公式：

$$x' = \frac{x - 0}{H - 0} \tag{16}$$

$$y' = \frac{y - 0}{W - 0} \tag{17}$$

可以得到反归一化公式：

$$x = x' \cdot H \tag{18}$$

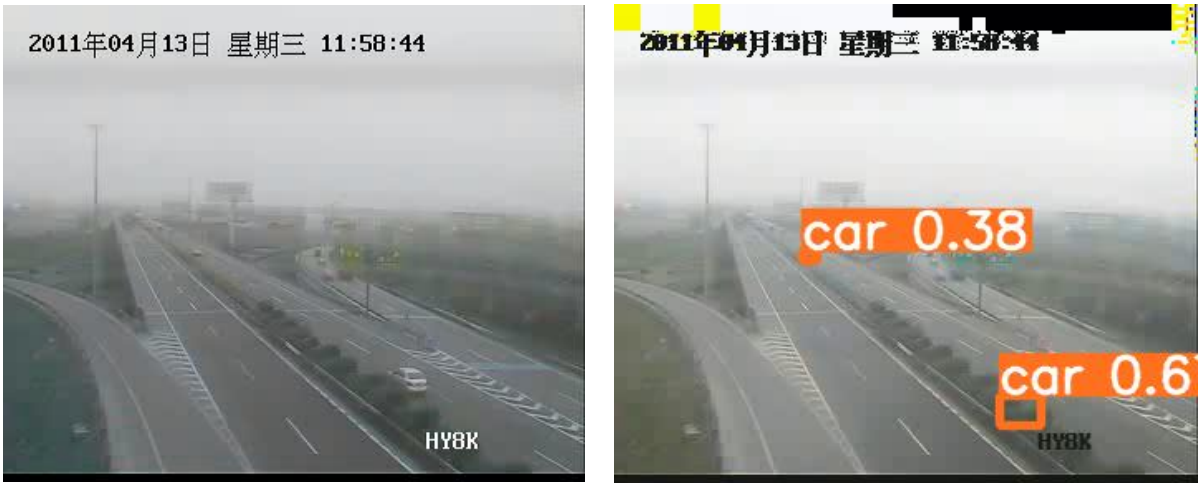
$$y = y' \cdot W \tag{19}$$

同理有：

$$a = a' \cdot H \tag{20}$$

$$b = b' \cdot W \tag{21}$$

以第一帧画面中右下角车辆为例，如图 13 所示：



原图　　YOLOv5 识别结果

图 13 识别结果反归一化示例图

YOLOv5 计算得到的结果为$x' = 0.847, y' = 0.694, a = 0.0625, b = 0.0710$。经过反归一化，得到了矩形中心点坐标为(244,244)，矩形高度为 18、宽度为 25，符合手动计算结果。

5.2.4 根据类型和位置过滤结果

在识别结果中，由于置信度临界值设置得比较低，会导致出现错误识别，例如将天空中的噪音识别成车辆或者其他物体，例如图 14 所示。

图 14 第 104 帧异常识别

可以从两方面过滤识别结果：

1. 标签

   仅保留标签为轿车（car）、客车（bus）和货车（truck）的识别结果，其他识别结果全部忽略。

2. 位置

   根据建立的坐标系（见图 2），仅保留天际线之下区域（$x \geq 130$）的识别结果，忽略天际线以上的识别结果。

5.2.5 YOLOv5 模型求解总结

根据以上步骤对于 YOLOv5 识别结果进行处理，最终对 7644 帧视频画面建立了数学模型，求得画面中所有车辆的位置信息，部分结果展示如下：

第 143 帧　　第 3723 帧　　第 7631 帧

图 15 YOLOv5 模型识别结果部分展示

5.3 问题二模型建立与求解

5.3.1 贪心匹配车辆

本文使用在滑动窗口内寻找车辆、计算重合面积并使用贪心匹配策略来跟踪车辆，为每辆车辆分配一个唯一编号，具体步骤如下：

Step1 贪心寻找最大重合的有编号车辆

在前 T 帧中逐帧寻找所有已经被分配编号的车辆，并且找出与当前车（记作 C）检测框重合面积最大的车辆检测框。

Step2 阈值判定

如果最大重合面积大于重合面积阈值 D，那么可以认为该检测框中车辆与当前车 C 是同一辆车，设置当前车的编号与该车编号相同，结束判断；如果最大重合面积小于阈值 D，则进入下一阶段。

Step3 贪心寻找最大重合的无编号车辆

和 Step1 类似，在前 T 帧中逐帧寻找所有未被分配编号的车辆，并且找出与当前车（记作 C）检测框重合面积最大的车辆检测框。

Step4 阈值判定

如果最大重合面积大于阈值 D，那么可以认为该检测框中车辆与当前车 C 是同一辆车，为当前车 C 和该车分配同一个新编号；如果最大重合面积小于阈值 D，则说明当前车 C 是一个新进入画面的车，先不为其作匹配，等待后续画面的车辆与之匹配。

下图 16 为贪心匹配策略整体流程图。

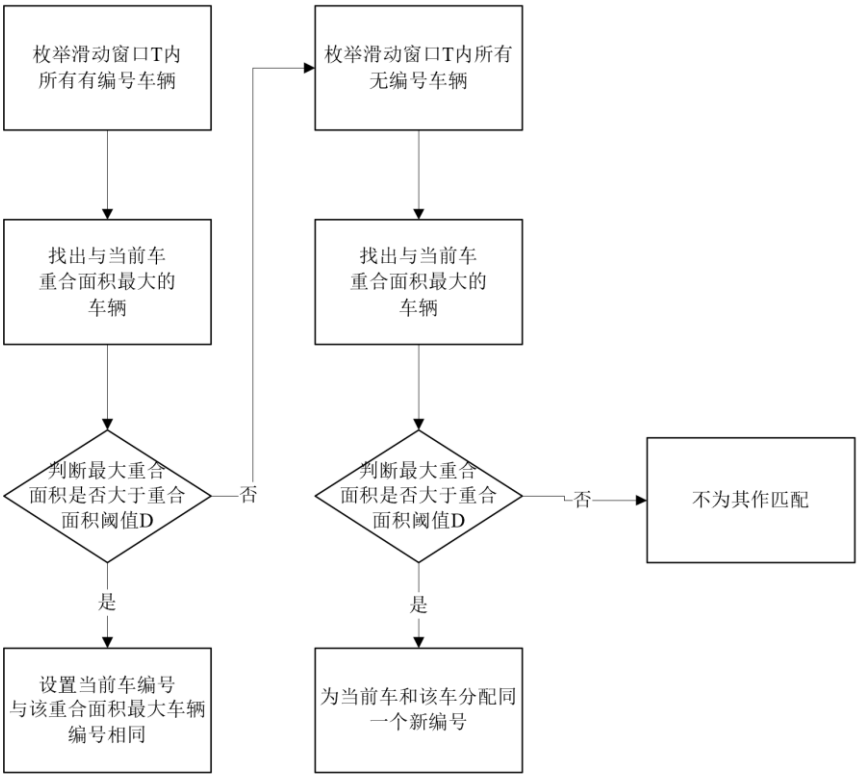


图 16 贪心匹配策略流程图

贪心面积匹配算法的重点在于确定滑动窗口宽度 T 与重合面积阈值 D。如果滑动窗口宽度 T 和重合面积阈值 D 设置太太，会导致前后行驶的车辆发生错误匹配，造成跟踪错误；如果 T 和 D 设置太小，会造成当前车辆难以与过去车辆成功匹配，而被当成一辆新出现在画面中的汽车。

第 3953 帧　　第 3957 帧

图 17 不同位置车辆大小和运动情况

根据“近大远小”透视原理，我们对于画面中的汽车按照其位置、速度和大小分为两类，并基于图 17 估算其数据：

表 1 两类位置的车辆速度和大小估计值

位置	速度	大小
画面中间	慢，大约每秒 4.7 像素	小，大约 109 平方像素
画面下侧	快，大约每秒 16.2 像素	大，大约 800 平方像素

为了最大程度地确保正确匹配，根据上表 1 确定滑动窗口大小 $T = 12$，重合面积阈值 $D = 20$。

5.3.2 选择合适的计数方法

从图像、视频中统计车流量方法有很多，下面列举几种：

（1） 检测线

检测线是设置在交通流场景中的一条垂直于道路方向的直线其宽度覆盖要检测的车道，高度在 1 至 3 个像素之间。由于检测线只有几个像素高，抗噪声能力较差，因此很容易因噪声而导致误检；同时，它对运动目标提取的质量要求很高，旦提取的运动目标中存在“孔洞”或“断层”，则很容易将一辆车检测为两辆，导致多检。

（2） 检测带

检测带是设置在交通流场景中的一条垂直于道路方向的带状区域，其宽度覆盖要检测的车道，高度为 14 个像素左右。由于检测带具有一定的高度，因而该方式的抗噪能力比检测线方式略强。

（3） 虚拟线圈

虚拟线圈是设置在交通流场景图像待检测车道上的矩形或四边形区域，其宽度比一个车道略小，高度一般为一辆小车的长度。它可以根据实际拍摄的交通流场景来相应地调整每个虚拟线圈的位置，从而减少由于车辆遮挡和“粘连"而导致的漏检现象；它的抗噪能力比检测带方式更强，而且也不容易由于提取的运动目标中存在“孔洞”或“断层”而导致多检。

传统的视觉模型需要选择合适的方法才能达到最好的效果，而基于 YOLOv5 的数学模型与贪心跟踪算法可以直接得到每辆车的精确坐标与运动轨迹。所以可以使用检测线

法，并结合基于滑动窗口的持续性目标跟踪算法，跟踪每一辆车并检测是否跨过提前设定好的线条。如果跨过了检测线，则计入总车流量，并且不再跟踪该车辆。

图中共有 4 条主要道路，为了达到最好的效果，我们选择点(240,0)和点(200,350)，作出一条尽可能垂直于四条道路的检测线，如下图 18 所示。

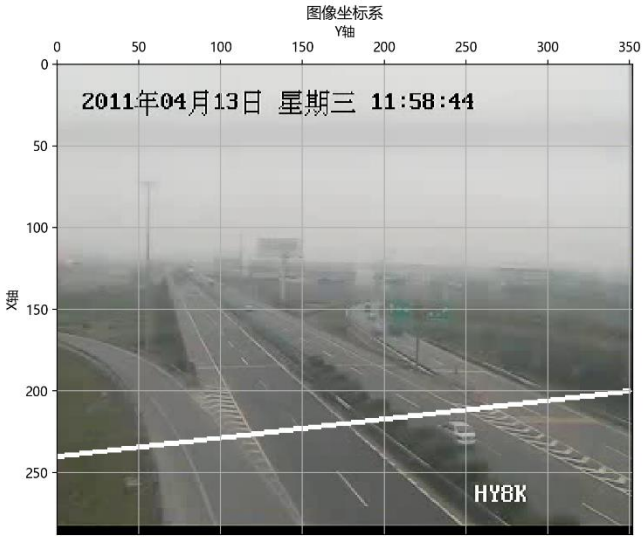


图 18 检测线设置

将两点带入一次函数公式：

$$y = kx + b \tag{22}$$

解出系数 $k = -7, b = 1680$，则检测线函数式为：

$$y = -7x + 1680 \tag{23}$$

5.3.3 判断车辆跨过检测线

使用上述贪心匹配策略，对于每辆车进行跟踪，并判断其在画面中相对于检测线的位置。假设当前某车辆中心坐标为(x, y)，根据公式(23)，若满足$7x + y - 1680 \leq 0$，则称该车处于检测线的上侧；若满足$7x + y - 1680 > 0$，则称该车处于检测线的下侧。

如果某车在当前帧中的位置与在上一帧中的位置分为位于检测线的上下两侧，那么可以认为该车跨过了检测线，计入总车流量中。每辆车仅统计一次跨线行为，第二次出现跨线将不计入车流量。

为了方便展示，本文将已经跨线了的车辆检测框设置为黑色，未跨线且位于检测线上侧的车辆检测框设置为红色、位于检测线下侧的车辆检测框设置为蓝色。图 19 展示部分判定结果。

第 3411 帧　　第 3564 帧　　第 3664 帧

图 19 跨线检测结果部分展示

其中，在第 3411 帧中正确地将检测线（图中白线）上侧的车辆标记为红色，将检测线下侧的车辆标记为蓝色；在第 3564 帧中，有同向行驶的两辆车跨过了检测线，被标记为黑色；在第 3664 帧中，有相向行驶的两辆车跨过了检测线。

5.3.4 统计车流量并可视化分析

统计车流量

通过对每一帧画面进行判断，统计得到共有 275 辆车跨过检测线。根据车流量计算公式：

$$TV = n \times \frac{25}{T} \times 60 \times 60 \tag{24}$$

其中，TV 是车流量，单位为辆/小时

n 是车辆总数，这里取 275

T 是视频帧长度，这里取 7644

25 是视频帧率，即每秒 25 帧画面

60×60 是一小时中的秒数

得到该道路上车流量为平均每小时 3237.8 辆车。该车流量属于中等至较高级别，符合高速公路实际情况。

可视化分析

以 10 秒（250 帧）为间隔，作出车辆数量直方统计图，如图 20 所示。

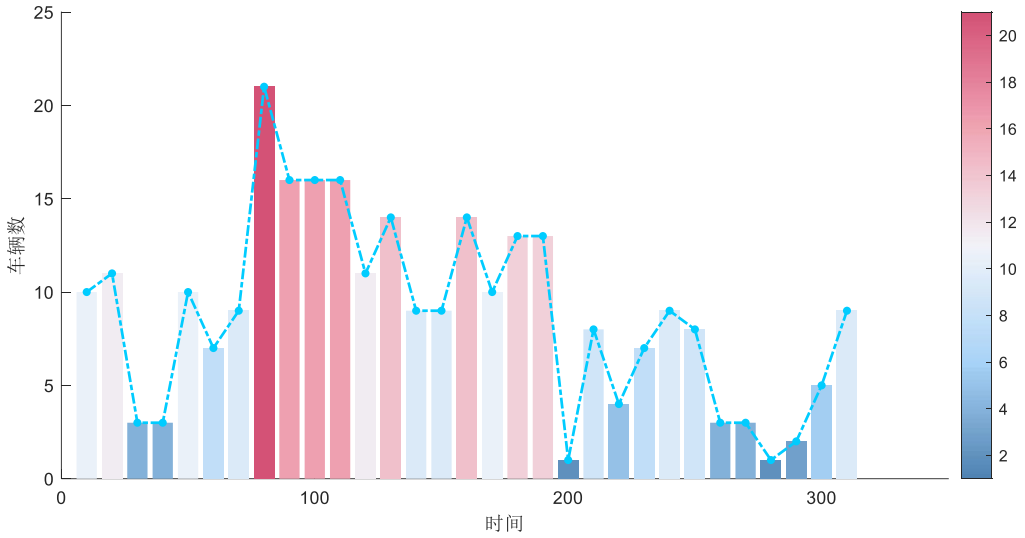


图 20 车流量随时间变化统计图

观察上图可以发现，在 80 秒左右，车流量迎来了一次峰值，之后车流量逐步减小。在 300 秒时，车流量有重新上升的趋势。推测其可能是高速上各车辆车速不同造成的车辆堆积、拥堵导致的。

六、 模型的检验与分析

6.1 模型检验

通过手工计数三次取平均数，得到视频画面中共有 256 辆车通过了检测线，而本文模型检测得到的数量为 275，多统计了 19 辆车，误差大约为 7.4%。

6.2 误差分析

本文模型误差主要来自于三方面：

（1） YOLOv5 模型未识别到车辆

由于原视频较为模糊，可能有部分车辆未被成功识别。根据人工排查，发现大型货车经常出现识别不到的情况，如图 21 所示。这一项误差会导致结果偏小。

图 21 第 5168 帧未识别到货车

（2） YOLOv5 模型识别车辆不稳定

同样由于原视频较为模糊，YOLOv5 模型可能会间断性地识别失败，导致车辆跟踪失败，将一辆车视作多辆。虽然贪心匹配策略可以在滑动窗口内尝试跟踪车辆，能够在一定程度上缓解这种“闪烁”的情况，但是还是无法从根源上避免。这一项误差会导致结果偏大。

（3） 视频画面较复杂

视频中的车辆出现了各种复杂情况。例如对于车辆运输车，模型容易将其识别为多辆车辆，如图 22 所示。这一项误差会导致结果偏大。

图 22 第 129 帧车辆运输车导致结果偏大

七、 模型的优缺点分析

7.1 模型的优点

7.1.1 预处理简单

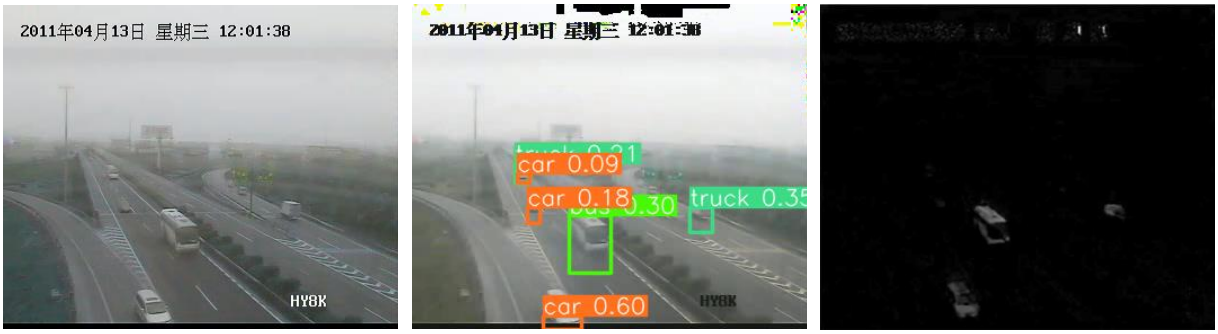
YOLO 在预处理需求方面相对较低。这一特性主要得益于其使用了 CNN，能够直接从原始像素数据中学习特征，而无需复杂的图像预处理步骤。相比之下，传统的对象检

测模型（如背景差分法）通常需要多个预处理步骤（比如对比度增强、直方图平衡化、灰度化、二值化等）以优化图像数据，提高检测的准确性和鲁棒性。

7.1.2 结果更直观

相比于传统模型， 本文基于 YOLO 算法建立的数学模型可以提供图像和视频中识别对象精确的位置，并以边界框的形式直接在原始图像上标出每个检测到的对象，结果更为直观和易于理解。

下图 23 中对比了本文模型与传统模型中背景差分法



原画面　　本文模型结果　　背景差分法结果

图 23 第 4352 帧本文模型与传统模型结果对比

7.1.3 可拓展性强

本文建立的基于 YOLO 算法车辆识别模型不仅可以识别车辆、统计车流量，还可以拓展出其他功能，以下举例两点：

（1） 多维度车流量统计

利用 YOLO 的高精度对象检测能力，可以按照车辆的移动方向进行细分。例如通过分析车辆的运动轨迹，可以区分并计算不同车向的车流量，获得更加准确的车流量信息。

（2） 车辆类型区分

YOLO 模型可以训练以识别不同类型的车辆，如轿车、客车和货车，得到不同车辆的车流量分布，有助于之后的交通分析，对优化道路使用和提高交通效率有所帮助。

7.1.4 求解速度快

本文模型中采用的 YOLO 系列算法相对于其他对象识别算法需要花费的处理时间更

短更迅速、算力需求更小，是目前公认的在对象识别领域的 SOTA（State of the arts，指特定领域技术先进、表现最好的技术方案），在小型计算设备上也可以实现毫秒级别的检测，适用于监控系统。

7.1.5 抗干扰能力强

得益于 YOLO 系列模型出色的抗干扰表现，即使环境中光照、天气发生变化，本文建立的模型也不需要调整参数以适应新环境。相比之下，传统模型（如背景差分法或基于特征匹配的检测系统）通常对环境条件更为敏感，需要对参数进行频繁的调整以适应环境变化。

7.2 模型的缺点

7.2.1 计算速度相对较慢

在计算资源受限的环境中，其计算速度可能仍逊于一些更简单的传统模型。传统方法由于计算过程较为简单，在低性能设备上仍然可以快速地执行。

7.2.2 对模糊视频的处理不足

题目提供的视频分辨率较低、画质压缩严重、光照条件一般，导致画面较为模糊。画面模糊影会影响了图像中物体的边缘和纹理信息，可能会导致模型发生误判或漏判。

八、 模型的改进与推广

8.1 模型的改进

8.1.1 针对性训练 YOLO 模型

由于在模糊监控视频上，YOLO 算法识别车辆表现不佳，故本文建议重新训练 YOLO 模型。可以手动收集监控画面，并制作成车辆数据集，并且在训练阶段引入模糊图像数据，帮助模型提高对模糊画面中的物体检测成功率。

8.1.2 使用匈牙利算法匹配车辆

本文使用贪心匹配的策略跟踪车辆。实际上，将当前帧中检测到的车辆与过去帧中的车辆进行最大化、正确率最高的匹配问题可以看作二分图最大权匹配问题[7]。匈牙利算

法又称为 KM 算法，可以在 $O(n^3)$ 时间内求出二分图的最大权完美匹配。相比于本文使用的贪心匹配策略，匈牙利算法可以得到更精准的跟踪结果，一定程度的避免“闪烁”与“车辆编号交换”等现象。

8.2 模型的推广

8.2.1 车流量计算与分析

可以结合车辆位置和移动方向信息，计算不同方向的车流量，使结果更加精确。

结合车辆的位置和移动方向信息，本文模型还可以区分不同方向的车流量。例如，在交叉口或多车道道路上，模型可以分别统计各个方向的车流量，得到更为详细的交通流向和车流分布结果。

8.2.2 车辆类型分类与统计

还可以对于将汽车分为轿车、客车和货车三类，进一步地分别计算不同类型车辆的车流量

本文使用的 YOLO 算法可以进一步细化并利用其检测能力，区分不同类型的车辆，如轿车、客车和货车，以提供更详细的车流统计信息。

8.2.3 车速测量与安全监控

利用道路上的标识物，如车道划分线（6 米白线与 9 米空缺循环），计算每处位置的实际长度与像素长度之比，以此计算车辆速度，实现车辆测速功能，进而能够检测超速、违规停车行为；还能分析车辆的速度趋势，及时发现拥堵情况。

九、 参考文献

[1] 陈艳敏.智能视频分析平台技术研究及钢铁行业应用[J].信息技术与标准化,2024,(07):90-94+100.

[2] 色彩平衡，https://zh.wikipedia.org/wiki/%E8%89%B2%E5%BD%A9%E5%B9%B3%E8%A1%A1

[3] Color math and programming code examples, https://www.easyrgb.com/en/math.php

[4] Kaiming He, Jian Sun and Xiaoou Tang, "Single image haze removal using dark channel prior," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp. 1956-1963, doi: 10.1109/CVPR.2009.5206515.

[5] 章博闻.面向汽车自动驾驶场景的目标检测与跟踪算法研究[D].重庆交通大学,2024.

[6] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017:2117-2125.

[7] 二分图最大权匹配，https://oi-wiki.org/graph/graph-matching/bigraph-weight-match/

十、 附录

附录 1 数据预处理部分代码

数据预处理部分代码 语言：Python 版本：3.11.8

```python
import os
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['Microsoft YaHei']
mpl.rcParams['axes.unicode_minus'] = False

only_first_frame = False
white_line = False


def display_avi(name):
    print("按 esc 退出播放")
    cap = cv2.VideoCapture(name)
    while cap.isOpened():
        ret, frame = cap.read()
        if ret:
            cv2.imshow("frame", frame)
        else:
            print("视频播放完成！")
```

```
            break

        key = cv2.waitKey(25)
        if key == 27:  # 按esc退出
            break

    cap.release()
    cv2.destroyAllWindows()

def read_video(name):
    cap = cv2.VideoCapture(name)
    num = 1
    frames=[]
    while True:
        success, data = cap.read()
        if not success:
            break
        # im = Image.fromarray(data)
        frames.append(data)
        print("正在读取第"+str(num)+"帧")
        num = num + 1
        if only_first_frame:
            break
    cap.release()
    return frames

def show_coordinate(path):
    image = cv2.imread(path)
    image_height, image_width = image.shape[:2]

    # 创建Matplotlib图像
    plt.figure(figsize=(8, 8))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    # 添加坐标轴标注信息
    plt.xlim(0, image_width)
    plt.ylim(image_height, 0)
    # plt.gca().invert_yaxis()
    ax = plt.gca()
    ax.xaxis.tick_top()
    ax.xaxis.set_label_position('top')
    plt.xlabel('Y轴')
    plt.ylabel('X轴')
```

```
    plt.title('图像坐标系')

    # 显示坐标系
    plt.grid(True)
    plt.show()



def apply_white_balance(frame):
    # 白平衡
    result = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
    avg_a = np.average(result[:, :, 1])
    avg_b = np.average(result[:, :, 2])
    if False:
        print(avg_a,avg_b)
        exit()
    result[:, :, 1] = result[:, :, 1] - ((avg_a - 128) * (result[:, :, 0] / 255.0)
* 1.1)
    result[:, :, 2] = result[:, :, 2] - ((avg_b - 128) * (result[:, :, 0] / 255.0)
* 1.1)
    result = cv2.cvtColor(result, cv2.COLOR_LAB2BGR)
    return result

def dark_channel(image, size=50):
    """计算暗通道"""
    b, g, r = cv2.split(image)
    min_img = cv2.min(cv2.min(r, g), b)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (size, size))
    dark = cv2.erode(min_img, kernel)
    return dark

def atmospheric_light(image, dark):
    """估计大气光"""
    h, w = image.shape[:2]
    num_pixels = h * w
    num_brightest = int(max(np.floor(num_pixels / 1000), 1))
    dark_vec = dark.reshape(num_pixels)
    image_vec = image.reshape(num_pixels, 3)

    indices = dark_vec.argsort()[-num_brightest:]
    brightest_pixels = image_vec[indices]
    A = brightest_pixels.max(axis=0)
    return A
```

```python
def transmission_estimate(image, A, omega=0.8, size=15):
    """估计透射率"""
    norm_img = image.astype(np.float32) / A
    transmission = 1 - omega * dark_channel(norm_img, size)
    return transmission

def guided_filter(I, p, r, eps):
    """引导滤波"""
    mean_I = cv2.boxFilter(I, cv2.CV_64F, (r, r))
    mean_p = cv2.boxFilter(p, cv2.CV_64F, (r, r))
    mean_Ip = cv2.boxFilter(I * p, cv2.CV_64F, (r, r))
    cov_Ip = mean_Ip - mean_I * mean_p

    mean_II = cv2.boxFilter(I * I, cv2.CV_64F, (r, r))
    var_I = mean_II - mean_I * mean_I

    a = cov_Ip / (var_I + eps)
    b = mean_p - a * mean_I

    mean_a = cv2.boxFilter(a, cv2.CV_64F, (r, r))
    mean_b = cv2.boxFilter(b, cv2.CV_64F, (r, r))

    q = mean_a * I + mean_b
    return q

def transmission_refine(image, et):
    """使用引导滤波优化透射率"""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) / 255.0
    r = 60
    eps = 0.0001
    t = guided_filter(gray, et, r, eps)
    return t

def recover(image, t, A, t0=0.7):
    """恢复去雾图像"""
    res = np.empty_like(image, dtype=np.float32)
    t = cv2.max(t, t0)

    for i in range(3):
        res[:, :, i] = (image[:, :, i] - A[i]) / t + A[i]

    return cv2.normalize(res, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
```

```
def dehaze(image):
    """去雾主函数"""
    dark = dark_channel(image)
    A = atmospheric_light(image, dark)
    te = transmission_estimate(image, A)
    t = transmission_refine(image, te)
    result = recover(image, t, A)
    return result

def bilateral_filtered(image):
    # d: 邻域直径
    # sigmaColor: 颜色空间的标准差
    # sigmaSpace: 坐标空间的标准差
    bilateral_filtered_image = cv2.bilateralFilter(image, d=3, sigmaColor=75,
sigmaSpace=75)
    return bilateral_filtered_image

def plot_histograms(images, titles):
    num_images = len(images)
    plt.figure(figsize=(12, 4 * num_images))

    for i, (image, title) in enumerate(zip(images, titles)):
        # 显示图像
        plt.subplot(num_images, 2, 2 * i + 1)
        plt.title(f"{title} - 图像")
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        plt.axis('off')

        # 计算并显示直方图
        plt.subplot(num_images, 2, 2 * i + 2)
        plt.title(f"{title} - 直方图")
        color = ('b', 'g', 'r')
        for j, col in enumerate(color):
            hist = cv2.calcHist([image], [j], None, [256], [0, 256])
            plt.plot(hist, color=col)
            plt.xlim([0, 256])
        plt.xlabel('Pixel Value')
        plt.ylabel('Frequency')

    plt.tight_layout()
    plt.show()
```

```
def show(images,titles):
    num_images = len(images)
    plt.figure(figsize=(12 * num_images, 6))

    for i, (image, title) in enumerate(zip(images, titles)):
        plt.subplot(1,num_images, i+1)
        plt.title(f"{title} - 图像")
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        plt.axis('off')

    plt.tight_layout()
    plt.show()


def output_frame(frame,folder,name):
    if white_line:
        print("draw white line")
        # cv2.line(frame, (0, int(-3.5 * 0 + 875)), (image.shape[1], int(-3.5 *
image.shape[1] + 875)),(255, 255, 255), thickness=2)
        cv2.line(frame, (0, 240), (350, 200), (255, 255, 255), thickness=2)
    if not os.path.exists(folder):
        os.makedirs(folder)
    im = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    im.save(os.path.join(folder, f"{name}.png"))


def frame_to_avi(frames,folder,name,fps=25):
    if not frames:
        print("帧列表为空，无法创建视频。")
        return

        # 获取帧的宽度和高度
    height, width, layers = frames[0].shape
    size = (width, height)

    # 定义编解码器并创建 VideoWriter 对象
    fourcc = cv2.VideoWriter_fourcc(*'XVID')  # 使用 XVID 编码
    out = cv2.VideoWriter(os.path.join(folder,f'{name}.avi'), fourcc, fps, size)

    for frame in frames:
```

```
        out.write(frame)

    out.release()
    print("视频已保存到"+os.path.join(folder,f'{name}.avi'))



avi_name="第三次人赛 A 题附件.avi"

display_avi(avi_name)

frames=read_video(avi_name)
output_frame(frames[0],"show","ord")
show_coordinate("show/ord.png")
# apply_white_balance
for i in range(len(frames)):
    if i % 100 == 0:
        print("正在对第"+str(i+1)+"帧 调整白平衡")
    frames[i]=apply_white_balance(frames[i])
    if only_first_frame:
        break
output_frame(frames[0],"show","white_balance")
plot_histograms(
    [cv2.imread("show/ord.png"),cv2.imread("show/white_balance.png")],
    ["原始", "白平衡调整"]
)
# dehaze
for i in range(len(frames)):
    if i % 100 == 0:
        print("正在对第"+str(i+1)+"帧 去雾")
    frames[i]=dehaze(frames[i])
    if only_first_frame:
        break
output_frame(frames[0],"show","white_balance_and_dehaze")
plot_histograms(

[cv2.imread("show/white_balance.png"),cv2.imread("show/white_balance_and_dehaze.pn
g")],
    ["白平衡调整", "白平衡调整并去雾"]
)

# bilateral_filtered
print(type(frames[0]))
```

```
for i in range(len(frames)):
    if i % 100 == 0:
        print("正在对第"+str(i+1)+"帧 双边滤波")
    frames[i]=bilateral_filtered(frames[i])
    if only_first_frame:
        break
output_frame(frames[0],"show","bilateral_filtered")

plot_histograms(

[cv2.imread("show/white_balance_and_dehaze.png"),cv2.imread("show/bilateral_filtered.png")],
    ["调整白平衡并去雾", "双边滤波"]
)

# 放大对比 1
im0=cv2.imread("show/white_balance_and_dehaze.png")
im1=cv2.imread("show/bilateral_filtered.png")
choose_area=(245,148,280,215)
im0_cropped = im0[choose_area[0]:choose_area[2], choose_area[1]:choose_area[3]]
im1_cropped = im1[choose_area[0]:choose_area[2], choose_area[1]:choose_area[3]]
scale_factor = 4
height, width = im0.shape[:2]
new_width = int(width * scale_factor)
new_height = int(height * scale_factor)
im0_resized = cv2.resize(im0_cropped , (new_width, new_height),
interpolation=cv2.INTER_CUBIC)
im1_resized = cv2.resize(im1_cropped , (new_width, new_height),
interpolation=cv2.INTER_CUBIC)
show([im0_resized,im1_resized],["双边滤波前","双边滤波后"])
cv2.waitKey(0)
cv2.destroyAllWindows()


# 放大对比 2
im0=cv2.imread("show/white_balance_and_dehaze.png")
im1=cv2.imread("show/bilateral_filtered.png")
choose_area=(211,228,238,260)
im0_cropped = im0[choose_area[0]:choose_area[2], choose_area[1]:choose_area[3]]
im1_cropped = im1[choose_area[0]:choose_area[2], choose_area[1]:choose_area[3]]
scale_factor = 4
height, width = im0.shape[:2]
new_width = int(width * scale_factor)
```

```
new_height = int(height * scale_factor)
im0_resized = cv2.resize(im0_cropped , (new_width, new_height),
interpolation=cv2.INTER_CUBIC)
im1_resized = cv2.resize(im1_cropped , (new_width, new_height),
interpolation=cv2.INTER_CUBIC)
show([im0_resized,im1_resized],["双边滤波前","双边滤波后"])
cv2.waitKey(0)
cv2.destroyAllWindows()


# output
print("输出所有图像到“output”")
for i in range(len(frames)):
    output_frame(frames[i], "output", format(i, "05"))
    if only_first_frame:
        break
print("输出所有图像完毕")

frame_to_avi(frames,"show","video_process")


plot_histograms(

[cv2.imread("show/ord.png"),cv2.imread("show/white_balance.png"),cv2.imread("show/
white_balance_and_dehaze.png"),cv2.imread("show/bilateral_filtered.png")],
    ["原始", "白平衡调整", "白平衡调整并去雾","双边滤波"]
)
```

附录 2 问题一 YOLO 模型代码

问题一 YOLO 模型代码 语言：Python 版本：3.11.8

```
import os
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from pylab import mpl
import torch
mpl.rcParams['font.sans-serif'] = ['Microsoft YaHei']
mpl.rcParams['axes.unicode_minus'] = False
from models.common import DetectMultiBackend
import torch
```

```
from models.common import DetectMultiBackend
from utils.general import non_max_suppression, scale_coords
from utils.plots import Annotator, colors, save_one_box
from utils.general import (LOGGER, check_file, check_img_size, check_imshow,
check_requirements, colorstr,
                        increment_path, non_max_suppression, print_args,
scale_coords, strip_optimizer, xyxy2xywh)

now_name="undefined"

save_dir = "yolo_"


def read_video(name):
    cap = cv2.VideoCapture(name)
    num = 1
    frames=[]
    while True:
        success, data = cap.read()
        if not success:
            break
        # im = Image.fromarray(data)
        frames.append(data)
        print("正在读取第"+str(num)+"帧")
        num = num + 1
    cap.release()
    return frames



def preprocess_frame(frame, img_size=640):
    # Convert BGR to RGB and resize
    img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (img_size, img_size))
    # Convert to tensor
    img = torch.from_numpy(img).float() / 255.0
    img = img.permute(2, 0, 1).unsqueeze(0)
    return img

def yolo(video_path):
    weights_path = 'pretrained/yolov5l.pt'
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = DetectMultiBackend(weights_path, device=device)
```

```
    names = model.names
    print(names)
    cap = cv2.VideoCapture(video_path)
    num = 0
    while cap.isOpened():
        num = num + 1
        ret, frame = cap.read()
        if not ret:
            break

        # Preprocess frame and perform detection
        img = preprocess_frame(frame).to(device)
        pred = model(img)
        det = non_max_suppression(pred, conf_thres=0.05, iou_thres=0.45)[0]

        # Rescale boxes from img size to original size
        det[:, :4] = scale_coords(img.shape[2:], det[:, :4], frame.shape).round()

        # Annotator for drawing boxes
        annotator = Annotator(frame, line_width=2, example=str('class_names'))
        save_path = os.path.join(save_dir, (f'frame_{num}') )
        txt_path = os.path.join(save_dir, (f'frame_{num}') )

        if len(det):
            # Process detections
            gn = torch.tensor(frame.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            for *xyxy, conf, cls in det:
                c = int(cls)
                if c not in [2, 5, 7]:
                    continue
                xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
1).tolist()  # normalized xywh
                if xywh[1] < 0.45:
                    continue
                label = f'{names[c]} {conf:.2f}'
                annotator.box_label(xyxy, label, color=colors(c, True))

                # Write to file

                line = (cls, *xywh, conf)
                with open(txt_path + '.txt', 'a') as f:
                    f.write(('%g ' * len(line)).rstrip() % line + '\n')
```

```
                # save_one_box(xyxy, img, file=save_dir / 'crops' / names[c] /
f'{p.stem}.jpg', BGR=True)


        # Show frame
        cv2.imshow('YOLO Detection', annotator.result())
        if cv2.waitKey(1) == ord('q') or cv2.waitKey(1) == 27:
            break
        # Save results (image with detections)
        cv2.imwrite(save_path+".png", annotator.result())

    cap.release()
    cv2.destroyAllWindows()

avi_path="show/video_process.avi"
now_name=input("请输入这一次任务的名字，不要有空格")
save_dir+=now_name
if not os.path.exists(save_dir):
    os.makedirs(save_dir)


yolo(avi_path)
```

附录 3 问题二 YOLO 模型与贪心匹配策略代码

问题二 YOLO 模型与贪心匹配策略代码 语言：Python 版本：3.11.8

```
import os
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from pylab import mpl
import torch
mpl.rcParams['font.sans-serif'] = ['Microsoft YaHei']
mpl.rcParams['axes.unicode_minus'] = False
from models.common import DetectMultiBackend
import torch
from models.common import DetectMultiBackend
from utils.general import non_max_suppression, scale_coords
from utils.plots import Annotator, colors, save_one_box
```

```
from utils.general import (LOGGER, check_file, check_img_size, check_imshow,
check_requirements, colorstr,
                           increment_path, non_max_suppression, print_args,
scale_coords, strip_optimizer, xyxy2xywh)
# 视频尺寸
video_width = 352
video_height = 288
video_len = 7644

# 跟踪设置
T = 12
D = 10
Area = 10


now_name="undefined"

save_dir = "yolo_track_"


def read_video(name):
    cap = cv2.VideoCapture(name)
    num = 1
    frames=[]
    while True:
        success, data = cap.read()
        if not success:
            break
        # im = Image.fromarray(data)
        frames.append(data)
        print("正在读取第"+str(num)+"帧")
        num = num + 1
    cap.release()
    return frames



def preprocess_frame(frame, img_size=640):
    # Convert BGR to RGB and resize
    img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (img_size, img_size))
    # Convert to tensor
    img = torch.from_numpy(img).float() / 255.0
```

```
    img = img.permute(2, 0, 1).unsqueeze(0)
    return img

# 存储每一帧中，所有车辆的信息
mess=[] # 第 i 帧 第 j 辆车的信息，信息格式为 x,y,a,b,ID=0,book 分别为中心点坐标,ID,是否已经有后继车辆
# ID = 0 # 唯一识别符
cross=set() # 已经跨线的车辆 ID

# 反归一化函数
def de_normalize(x,y):
    x *= video_width
    y *= video_height
    return x,y

# 欧拉距离
def dis(x1,y1,x2,y2):
    # print("dis=",((x1-x2)**2+(y1-y2)**2)**0.5)
    return ((x1-x2)**2+(y1-y2)**2)**0.5

def superposition (xc1,yc1,a1,b1,xc2,yc2,a2,b2):
    l1 = xc1 - a1 / 2
    r1 = xc1 + a1 / 2
    t1 = yc1 - b1 / 2
    d1 = yc1 + b1 / 2

    l2 = xc2 - a2 / 2
    r2 = xc2 + a2 / 2
    t2 = yc2 - b2 / 2
    d2 = yc2 + b2 / 2

    x1=max(l1,l2)
    x2=min(r1,r2)
    y1=max(t1,t2)
    y2=min(d1,d2)
    if x2>x1 and y2>y1:
        # print("same area=",(x2-x1)*(y2-y1))
        return (x2-x1)*(y2-y1)
    return 0

def calc_line(x,y):
    # print("x=",x,"y=",y,",在白线的")
    # if 7*x+y-1680>=0:
```

```
        # print("下")
    # else:
        # print("上")
    return 7*x+y-1680>=0

def yolo(video_path):
    ID=0
    weights_path = 'pretrained/yolov5l.pt'
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = DetectMultiBackend(weights_path, device=device)
    names = model.names
    print(names)
    cap = cv2.VideoCapture(video_path)
    num = 0
    while cap.isOpened():
        num = num + 1
        ret, frame = cap.read()
        if not ret:
            break

        # Preprocess frame and perform detection
        img = preprocess_frame(frame).to(device)
        pred = model(img)
        det = non_max_suppression(pred, conf_thres=0.10, iou_thres=0.45)[0]

        # Rescale boxes from img size to original size
        det[:, :4] = scale_coords(img.shape[2:], det[:, :4], frame.shape).round()

        # Annotator for drawing boxes
        annotator = Annotator(frame, line_width=1, example=str('class_names'))

        y1, y2 = 0, 240
        x1 = int(-7 * y1 + 1680)
        x2 = int(-7 * y2 + 1680)

        cv2.line(frame, (x1, y1), (x2, y2), (255, 255, 255), 1)

        save_path = os.path.join(save_dir, (f'frame_{num}') )
        txt_path = os.path.join(save_dir, (f'frame_{num}') )

        now_mess=[]
        if len(det):
            # Process detections
```

```
            gn = torch.tensor(frame.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            for *xyxy, conf, cls in det:
                c = int(cls)
                if c not in [2, 5, 7]:
                    continue
                xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
1).tolist()  # normalized xywh
                if xywh[1] < 0.45:
                    continue

                # tracker
                y,x=de_normalize(xywh[0], xywh[1])
                b,a=de_normalize(xywh[2],xywh[3])
                now_car = [x, y, a, b, 0, 0]
                # print(f"search {max(0,num-1-T)}-{num-1}")
                # print("mess=")
                # print(mess)
                min_t,min_i,max_area=0,0,0

                # find who 4 != 0
                for t in range(max(0,num-1-T),num-1):
                    for i in range(len(mess[t])):
                        # print("t=",t,"i=",i)
                        old_car=mess[t][i]
                        if old_car[5] == True or old_car[4] == 0:
                            continue
                        if
superposition(x,y,a,b,old_car[0],old_car[1],old_car[2],old_car[3]) >= max_area:
                            min_t,min_i,max_area =
t,i,superposition(x,y,a,b,old_car[0],old_car[1],old_car[2],old_car[3])
                if max_area <= Area:
                    for t in range(max(0,num-1-T),num-1):
                        for i in range(len(mess[t])):
                            # print("t=",t,"i=",i)
                            old_car=mess[t][i]
                            if old_car[5] == True:
                                continue
                            if
superposition(x,y,a,b,old_car[0],old_car[1],old_car[2],old_car[3]) >= max_area:
                                min_t,min_i,max_area =
t,i,superposition(x,y,a,b,old_car[0],old_car[1],old_car[2],old_car[3])

                if max_area >= Area:
```

```
                old_car = mess[min_t][min_i]
                # print("choose ",old_car)
                if old_car[4]==0:
                    ID+=1
                    mess[min_t][min_i][4]=now_car[4]=ID
                else:
                    now_car[4]=old_car[4]
                mess[min_t][min_i][5]=True
                # cross white line
                if (now_car[4] not in cross) and
(calc_line(old_car[0],old_car[1]) != calc_line(x,y)):
                    cross.add(now_car[4])


            now_mess.append(now_car)

            def hex_to_bgr(hex_color):
                """将十六进制颜色字符串转换为 RGB 元组"""
                hex_color = hex_color.lstrip('#')  # 移除开头的'#'（如果有）
                return tuple(int(hex_color[i:i + 2], 16) for i in (4, 2, 0))

            label = f'car {now_car[4]}'
            label_color = hex_to_bgr("FF0000")
            if calc_line(now_car[0],now_car[1]):
                label_color = hex_to_bgr("00CCFF")
            if now_car[4] in cross:
                label_color = hex_to_bgr("000000")
            # annotator.box_label(xyxy, label, color=colors(c, True))
            annotator.box_label(xyxy, label, color=label_color)

            # Write to file

            line = (cls, *xywh, conf)
            with open(txt_path + '.txt', 'a') as f:
                f.write(('%g ' * len(line)).rstrip() % line + '\n')


            # save_one_box(xyxy, img, file=save_dir / 'crops' / names[c] /
f'{p.stem}.jpg', BGR=True)
        # print(now_mess)
        mess.append(now_mess)
        print("frame = ", num, "ID =", ID,"Cross = ", len(cross))
        # Show frame
```

```
        cv2.imshow('YOLO Detection', annotator.result())
        if cv2.waitKey(1) == ord('q') or cv2.waitKey(1) == 27:
            break
        # Save results (image and cross message)
        cv2.imwrite(save_path+".png", annotator.result())
        with open(os.path.join(save_dir,'cross.txt'), 'a') as fc:
            fc.write(str(num) + " " + str(len(cross)) + "\n")

    cap.release()
    cv2.destroyAllWindows()

avi_path="show/video_process.avi"
now_name=input("请输入这一次任务的名字，不要有空格")
save_dir+=now_name
if not os.path.exists(save_dir):
    os.makedirs(save_dir)


yolo(avi_path)
```

附录 4 问题二统计图绘制代码

问题二统计图绘制代码 语言：Matlab 软件：MATLAB R2023b

```
data=table2array(readtable("yolo_track_3color\cross.txt"));
data=data(:,2);
data=[0;diff(data)];
P=[];
T=10;

for i=1:T*25:7644
    P=[P,sum(data(i:min(7644,i+T*25-1)))];
end

% 绘制条形图
figure;
hold on;  % 保持当前图形，允许多次绘图

% 定义色彩映射
SIGEWINNE = [81 132 178;
             170 212 248;
             242 245 250;
             241 167 181;
```

```
            213 82 118] / 256;
num_colors = 100;
interp_colors = interp1(linspace(0, 1, size(SIGEWINNE, 1)), SIGEWINNE,
linspace(0, 1, num_colors));
colormap(interp_colors);

% 使用 patch 绘制每个条，并设置颜色
for idx = 1:length(P)
    x = [idx-0.4, idx+0.4, idx+0.4, idx-0.4];
    y = [0, 0, P(idx), P(idx)];
    disp(ceil(P(idx)/max(P)*num_colors));
    patch(x, y, interp_colors(max(1,ceil(P(idx)/max(P)*num_colors)), :),
'EdgeColor', 'none');
end

plot(P, 'Color', [0, 0.8, 1], 'LineStyle', '-.', 'Marker', '.', 'LineWidth', 1.5,
'MarkerSize', 10);


% 设置颜色条和坐标轴标签
colorbar;
caxis([min(P), max(P)]);  % 设置色彩轴的范围为数据的最小值和最大值
xlabel('时间');
ylabel('车辆数');
xticks(0:10:length(P));
xticklabels(arrayfun(@num2str, (0:10:length(P))*10, 'UniformOutput', false));  %
设置自定义的刻度标签
hold off;  % 释放图形保持状态
```