


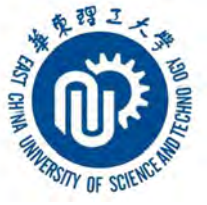

中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生数学建模竞赛

学　　校	华东师范大学
参赛队号	**20102690219**
队员姓名	**1.**宋智超
	2.刘旗洋
	3.季仁杰

中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生

数学建模竞赛

题 目 能见度估计与预测

摘 要：

能见度对于交通出行的影响重大，特别是在高速公路和机场等对于能见度较为敏感的区域，能够准确获知能见度的数值和变化趋势尤为重要。而能见度易受雾和霾的影响，因此研究雾的变化机制并做出预测极具现实意义。传统的仪器探测法成本高，覆盖范围小，而基于视频的能见度检测能弥补仪器测量的不足之处，但是预测精度不高，需要进一步改进和完善。本文针对探索大雾演化规律，预测大雾变化趋势进行了建模研究。

能见度与气象因子的关系（问题一）根据问题的要求，首先分析各个单因子的分布情况，在分析因子之间的相关性，采用 Lasso 方法得到各因子的重要性，将相关性与重要性相结合，确定参与建模的影响因子。经过测试，选择相对湿度(RH)、温度(TEMP)、2 分钟平均风速(WS2A)三个变量建立三次多项式拟合，并且考虑白天和夜间的气象条件差异，分别建立三变量在全天、白天和夜间三种场景下的三次多项式关系式。根据上述方法建立的全天数据关系式拟合优度(R^2)能够达到 0.94 以上。

基于视频数据的能见度深度学习模型（问题二）根据问题要求，首先对原始数据进行数据匹配和图像校正预处理，得到 1860 条对应数据，并基于特征工程的思想，使用尺度不变特征变化(SIFT)提取图像特征，得到 1860×15766 维特征向量集，考虑到数据分布的不均衡性，对训练集和验证集的划分进行调整以保证输入数据的分布均衡。之后，输入全连接神经网络深度学习模型，损失函数快速收敛，经过 200 轮训练，最终训练得到的模型在训练集和验证集上的 R^2 分别为 0.94 和 0.86，表明模型具有较好的泛化能力。

不依赖能见度观测值的能见度估计算法（问题三）根据问题要求，首先对图像进行裁剪预处理以去除水印对能见度估计的干扰，比较了单目图像计算景深法、公路标线可见长度推算法和暗通道先验算法的可行性，最终利用暗通道先验规律和高速公路车道分隔线间距，设计了一种通过大气散射系数预测气象光学视程的算法，得到图像对应时段的能见度估算值，范围在 105 米至 140 米之间，总体呈现波动上升的趋势，最后绘制出能见度随时间变化的曲线。

大雾变化趋势预测模型（问题四）根据问题的要求，首先对问题三中得到的能见度随时间变化曲线进行统计特征分析，再采用二次指数平滑法，寻找时间序列之间相互影响的内在规律，模型预估的气象光学视程达到 150 米及以上的时间为当日 9 时 5 分前后。

关键词：能见度，雾，多项式回归，SIFT 特征提取，全连接神经网络，图像建模，暗通道先验，指数平滑法，趋势预测

目录

一、问题背景与问题重述

1.1 问题背景

能见度是指视力正常的人能将目标物从背景中识别出来的最大距离。能见度易受到气象条件，空气质量的影响，其中较为关键的影响因子是雾和霾。而能见度对于交通出行、生产生活至关重要，在能见度较低的情况下，高速公路会采取封道措施，而机场也会限制航班起降的流量甚至禁止航班起降。因此，能见度预测对于交通管理部门和航空公司极具现实意义。

传统的能见度测量方法应用较为广泛的有人工目测和仪器检测两大类，人工目测主观性过强，激光能见度仪的检测方法更为可信，但是激光能见度仪覆盖的检测范围小，安装维护的成本过高，无法完全覆盖全国高速扩张的高速路网。因此，研究者提出了利用广泛覆盖的交通视频路况数据间接计算得到能见度数值的方法，此方法成本较低，覆盖范围广泛，应用前景广阔。但是现有的研究大都针对的是视频图像中截取的帧图像，忽略了视频信息的时间连续性，且算法的精度一般较低。而视频信息的时间连续性对于估计大雾的变化过程具有参考意义，可以利用视频包含的丰富信息不断优化能见度计算算法并对未来路况进行预测。

1.2 问题重述

基于上述研究背景，题目提供了机场 AMOS 观测数据，机场视频数据和高速公路视频截图数据 3 个相关数据，本文需要解决以下四个问题：

问题 1：能见度与近地面的气象因素关系构建。

雾的形成本质上是一个气象过程，在水汽充足、微风及大气稳定的情况下，相对湿度达到 100% 时，空气中的水汽便会凝结成细微的水滴悬浮于空中，使地面水平的能见度下降。因此，题目要求借助机场的 AMOS 观测数据中的气压、温度、湿度、风速等气象因子，建立其与能见度之间的关系式。

问题 2：基于视频数据的能见度估计深度学习模型构建。

基于视频数据的能见度估计是能见度测量发展的新方向，拥有巨大的优化提升空间。题目要求根据机场视频数据和机场 AMOS 观测数据建立深度学习模型，并对能见度估计的精度进行评估。

问题 3：建立不基于实测数据的能见度估计算法。

建立覆盖高速路网的激光能见度仪是不现实的，需要在缺少实测数据的情况下仅仅根据道路视频信息间接计算能见度，在这种情况下，可以利用视频中不同景深的物体的亮度差异，计算得到估计的能见度结果，题目要求使用高速公路视频截图，实现不基于实测数据的能见度估计算法，并对绘制能见度随时间变化曲线。

问题 4：大雾变化趋势模型构建。

问题 3 中得到了能见度随时间变化的曲线，可以利用该曲线估计大雾变化的趋势，题目要求建立相关模型，预测大雾将加强或减弱，何时消散。

二、模型假设与符号说明

2.1 模型假设

假设一、摄像头与两个参考点的距离的差值约等于两个参考点之间的距离。
假设二、在一幅影像内，大气散射系数 β 的值相同。

2.2 符号说明

表 2-1 符号说明

符号	含义
R^2	决定系数
$RMSE$	均方根误差
MAE	平均绝对误差
v	含噪声的原始图像
$\tilde{u}$	去除噪声后的图像
w	像素点x和y间的相似度
$V(x)$	以x为中心的矩形邻域
$Z(x)$	归一化系数
h	高斯函数平滑参数
V_{in}^{y}	Gamma 增强的输入图像
V_{out}	Gamma 增强的输出图像
$G(x,y,\sigma)$	高斯卷积核函数
m	梯度的幅度大小
θ	梯度的方向
$I(x)$	原始图像像元值
$J(x)$	无雾情况下图像像元值
A	大气光照亮度
$t(x)$	大气透过率
ω	系数
σ	大气消光系数
α	吸收系数
β	散射系数

$R(x)$	所选像元区域
p_R	所选区域的像元数量
D	参考点与摄像头的距离
ACF	自相关函数
$PACF$	偏自相关函数
sgn	符号函数

三、问题一的建模与求解

3.1 任务一问题分析

任务一要求建立一个描述能见度与地面气象观测之间的关系模型，气象数据包括温度、湿度和风速等，能见度数据包括 RVE 和 MOR。由于 AMOS 文件给出的数据类型较多且数量不同，首先需要对数据进行统计，去除异常数据。其次，由于数据类型较多，我们需要对其进行筛选，选择出与能见度具有高相关性的数据类型。除此之外，考虑气象因素外，白天和晚上由于光学条件本身不同，能见度也具有较大的差异，因此，需要将白天、晚上的模型分开讨论。

具体流程图如图 3-1 所示。

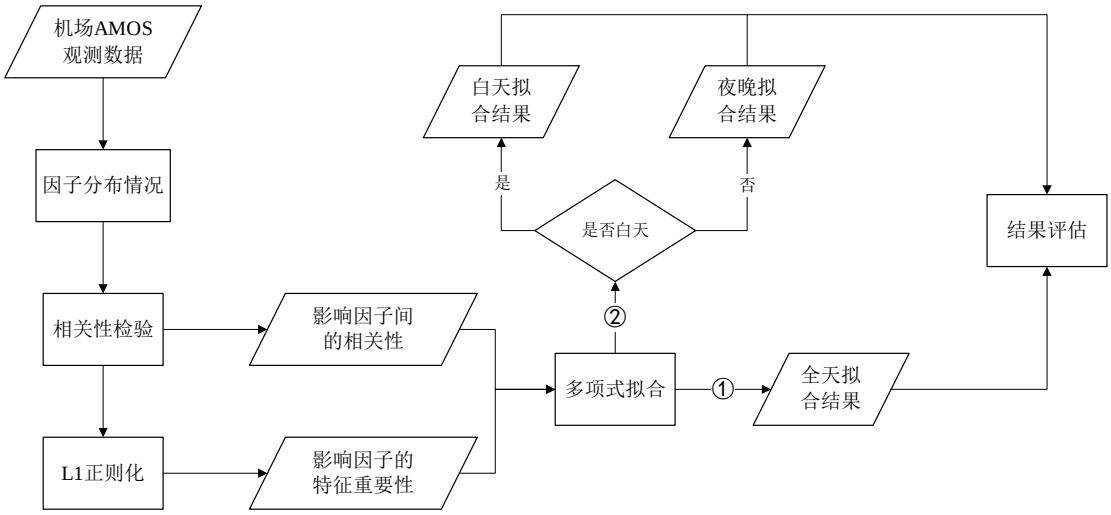

图 3-1 任务一流程图

3.2 任务一数据分析及预处理

首先对给出的 AMOS 数据集进行原始数据分析。PTU 数据的时间分辨率为 1min，包括气压、温度、湿度等气象数据；VIS 数据的时间分辨率为 15s，包括 RVR 能见度、MOR 能见度、灯光等能见度数据；WIND 数据的时间分辨率为 15s，包括风向、风速等气象数据。3 个数据文件的数据量和时间分辨率如表 3-1 所示。

表 3-1 AMOS 数据文件基本情况

时间	文件名	数据总量（行）	时间分辨率
20191216	PTU_R06_15.his	1442	1min
	VIS_R06_15.his	5757	15s
	WIND_R06_15.his	5757	15s
20200313	PTU_R06_12.his	1442	1min
	VIS_R06_12.his	5757	15s
	WIND_R06_12.his	5757	15s

通过对数据进行统计和分析，发现两天的 PTU 数据共有 2884 个，且 VIS 在 1min 内变化较小，且数据质量较高。此外，两天的数据虽然在不同的年份和季节，但具有较好的一致性。我们认为取 1min 的时间分辨率可以满足建模的数量和分辨率要求，因此通过时间匹配的方法，提取出 3 类数据的共有部分。由于有小部分样本在记录时有所缺失，针对 2879

个完整的样本进行建模。

AMOS 数据集给出了较多的气象数据及能见度数据，包括：

表 3-2 AMOS 数据基本情况

英文名称	中文名称	单位
WS2A	2 分钟平均风速	m/s
CW2A	2 分钟平均垂直风速	m/s
WD2A	2 分钟平均风向	度
PAINS	本站气压	hPa
QFE	飞机着陆地区最高点气压	hPa
QNH：	修正海平面气压	hPa
TEMP：	气温	℃
RH	相对湿度	%
DEWPOINT	露点温度	℃
MOR_1A	能见度	m

显然，各类气象数据对雾的形成与消散作用、影响能力均不同，有必要从诸多数据中挑选出重要的影响因素。各影响因子的数据分布如下图 3-2 所示。

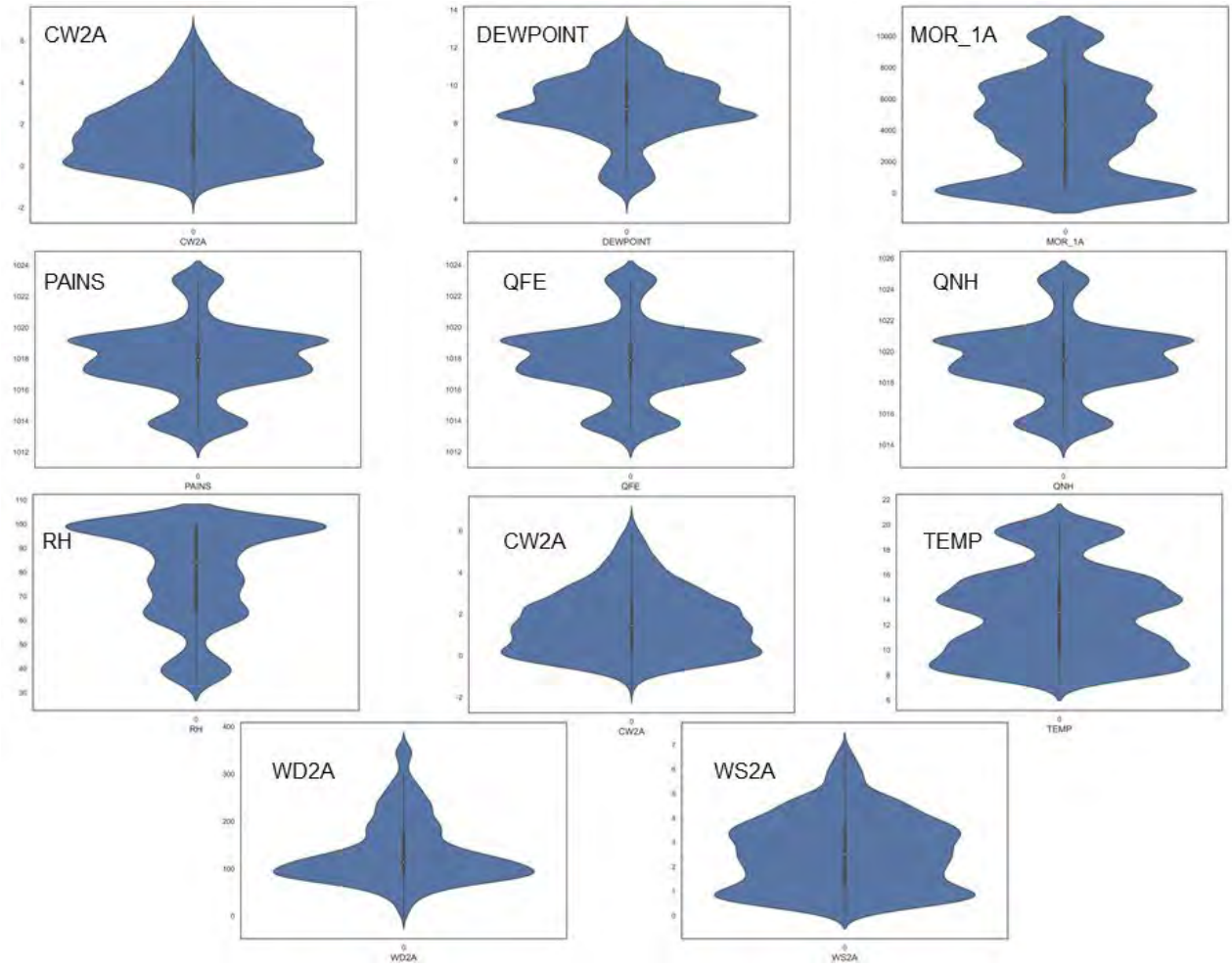

图 3-2 各影响因子的分布情况

查看各因素间的散点图，并计算得到各影响因子之间的相关性如图 3-3 和图 3-4 所示。

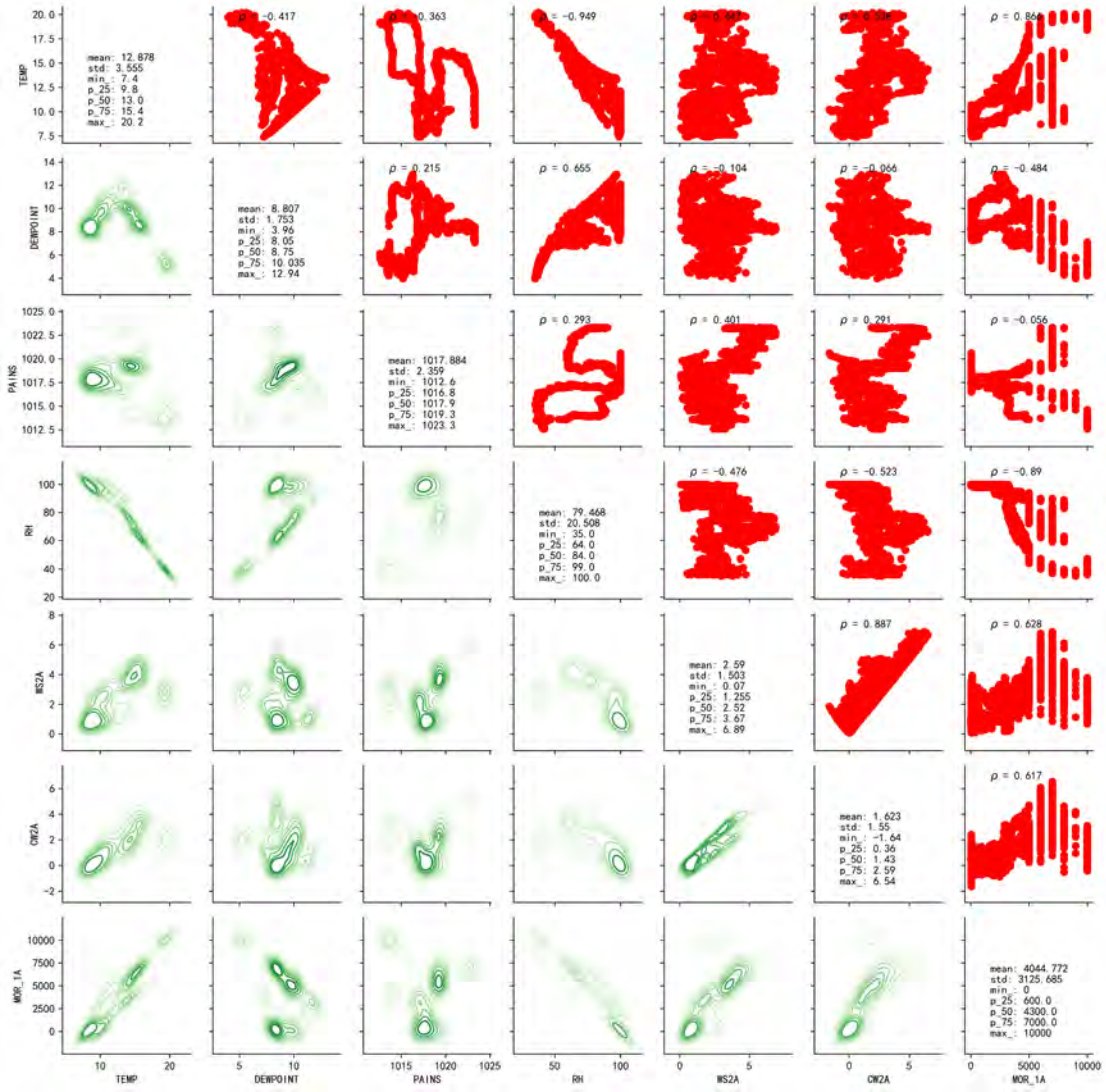

图 3-3 多变量组图（Pairplot）

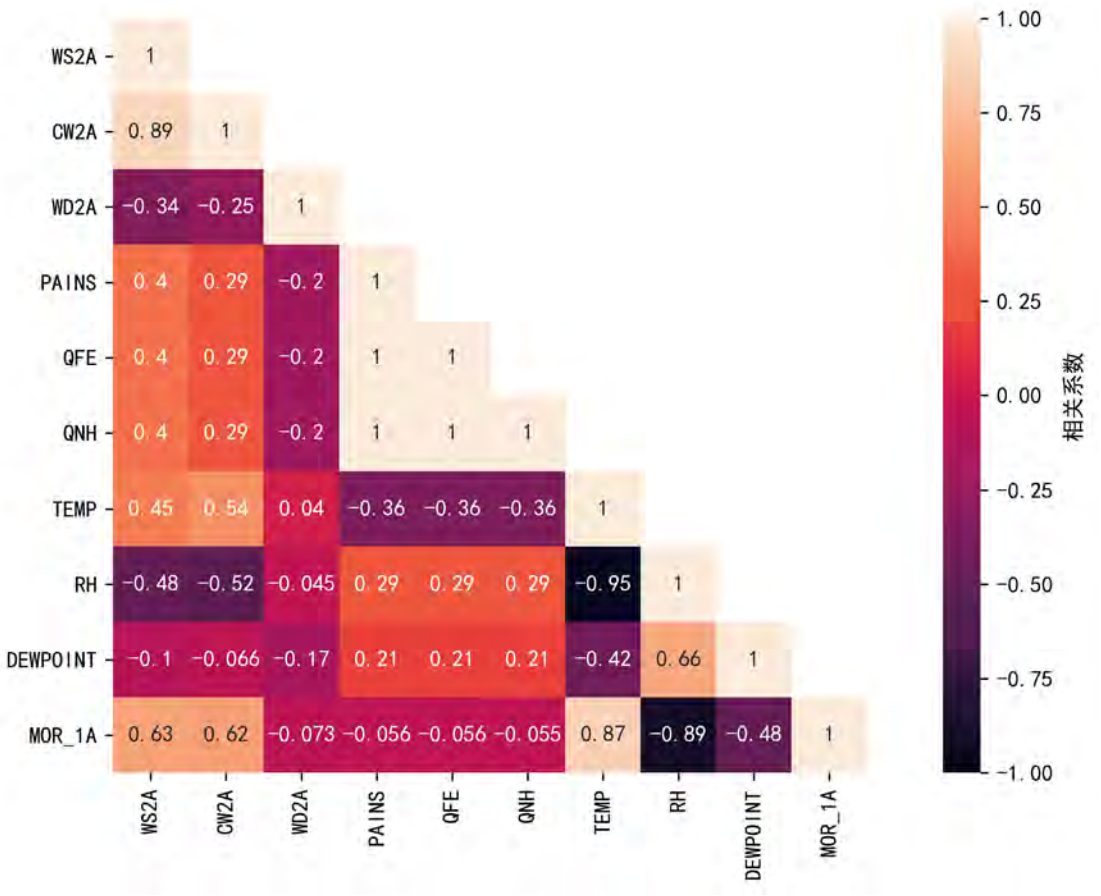

图 3-4 各影响因子间的相关性

通过计算相关性表明，与 MOR_1A 相关性由强到弱如表 3-3 所示。

表 3-3 与 MOR_1A 相关性由强到弱排名

影响因子	相关系数（绝对值）	正相关/负相关
RH	0.89	-
TEMP	0.87	+
WS2A	0.63	+
CW2A	0.62	+
DEWPOINT	0.48	-
WD2A	0.073	-
PAINS	0.056	-
QFE	0.056	-
QNH	0.055	-

由上述图表可知，对 MOR_1A（能见度）具有较大影响的气象因素为：

（1）RH（相对湿度）：由图 3-5 可知，能见度与相对湿度具有较强的负相关性（-0.89），当相对湿度达到 100%后，能见度接近于 0m，而当相对湿度在 40%左右时，能见度已经超出了量程 10000m。

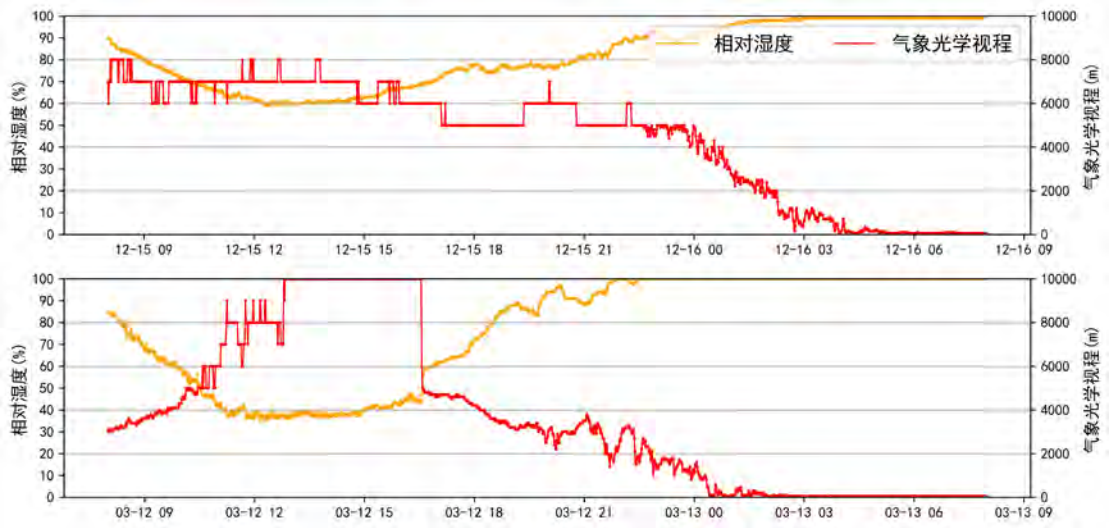

图 3-5 能见度与相对湿度相关性结果

（2）TEMP（气温）：由图 3-6 可知，能见度与气温具有较强的正相关性（+0.87），当气温较低时，能见度与气温的相关性更强，这在 2020 年的数据中更加明显。

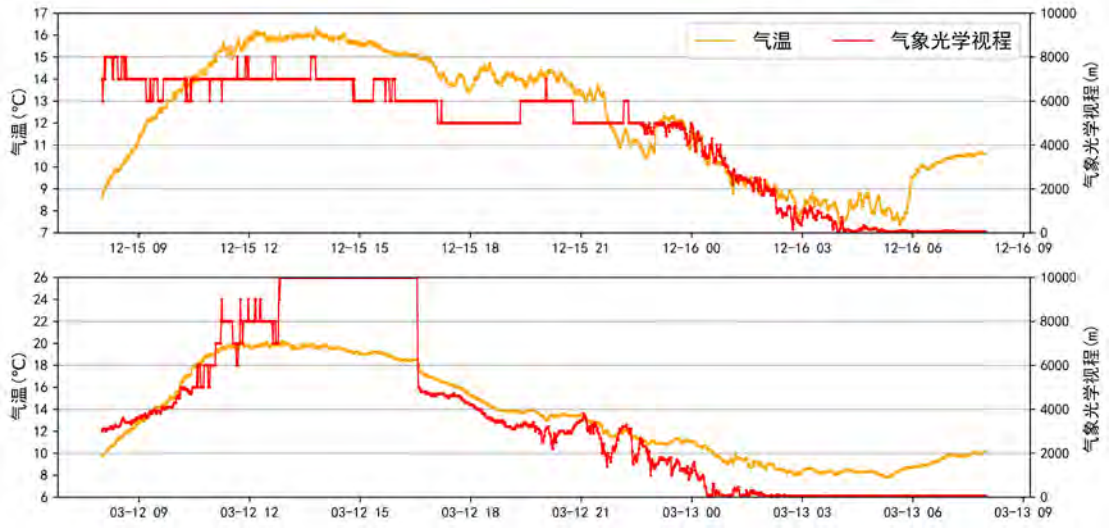

图 3-6 能见度与气温相关性结果

（3）WS2A（2 分钟平均风速）&CW2A（2 分钟平均垂直风速）：由图 3-7 可知，能见度与平均风速、垂直平均风速均有较强的正相关性（+0.63、+0.62），同时，平均风速和垂直平均风速两者间的相关性也较高（+0.89）。

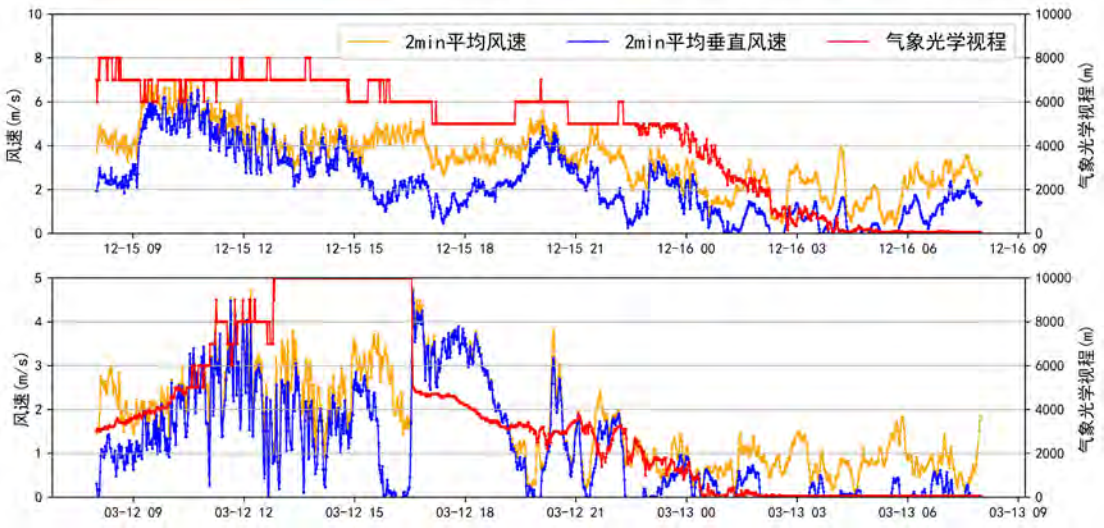

图 3-7 能见度与风速相关性结果

（4）DEWPOINT（露点温度）：由图 3-8 可知，能见度与露点温度具有一定的负相关性（-0.48），但在不同日期、不同时间段内相关性具有差异性。

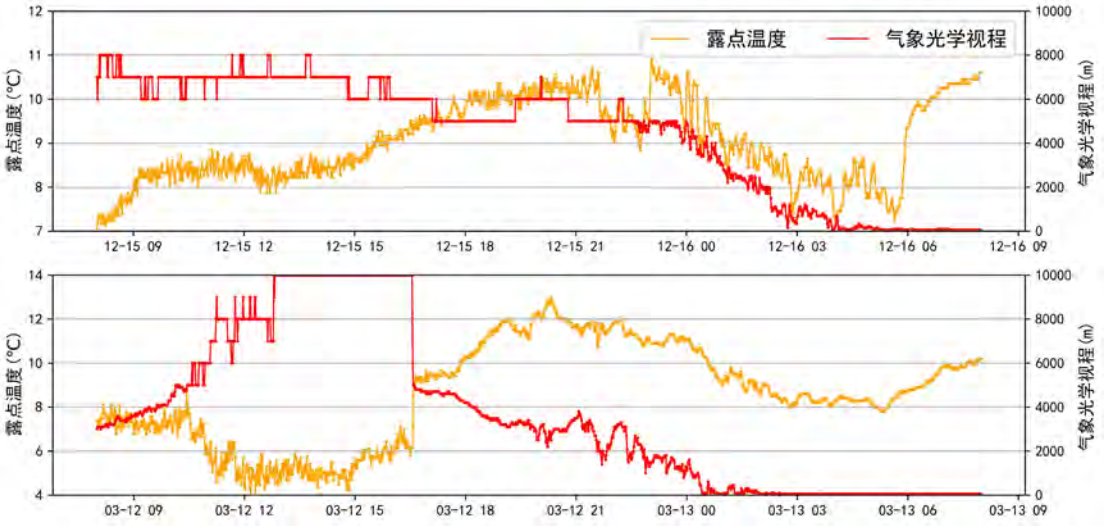

图 3-8 能见度与露点温度相关性结果

（5）PAINS（本站气压）：由图 3-9 可知能见度与气压之间的相关性较差，在不同日期中，能见度对气压的响应情况并不相同。如在 12.15 的数据中，两者近于正相关，而在 3.13 的数据中，两者又近于负相关，这可能受其他气象因子影响，或者在不同季节，气压对该地的能见度的影响情况不一。其他两个气压与能见度的相关性接近，相关性均较低，可见气压并不是影响能见度的主要原因。

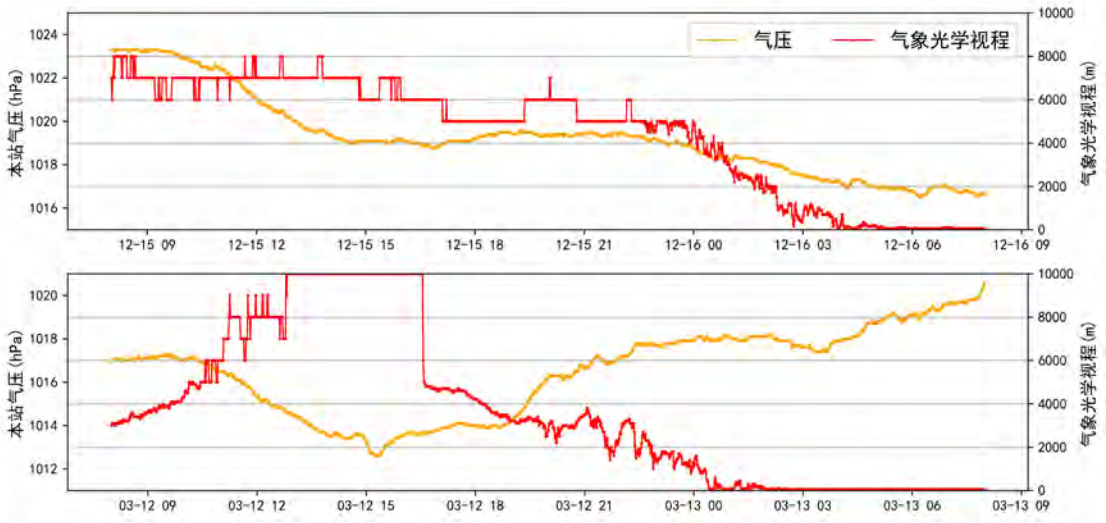

图 3-9 能见度与气压相关性结果

通过 L1 正则化（Lasso）计算各影响因子的特征重要性。L1 正则化往往会使学习后的模型较为稀疏，这个特性使得 L1 正则化成为一种很好的特征选择方法，Python 中的 Scikit-learn 为线性回归提供了 Lasso，为分类提供了 L1 逻辑回归。通过计算，得到如图 3-10 所示的结果。

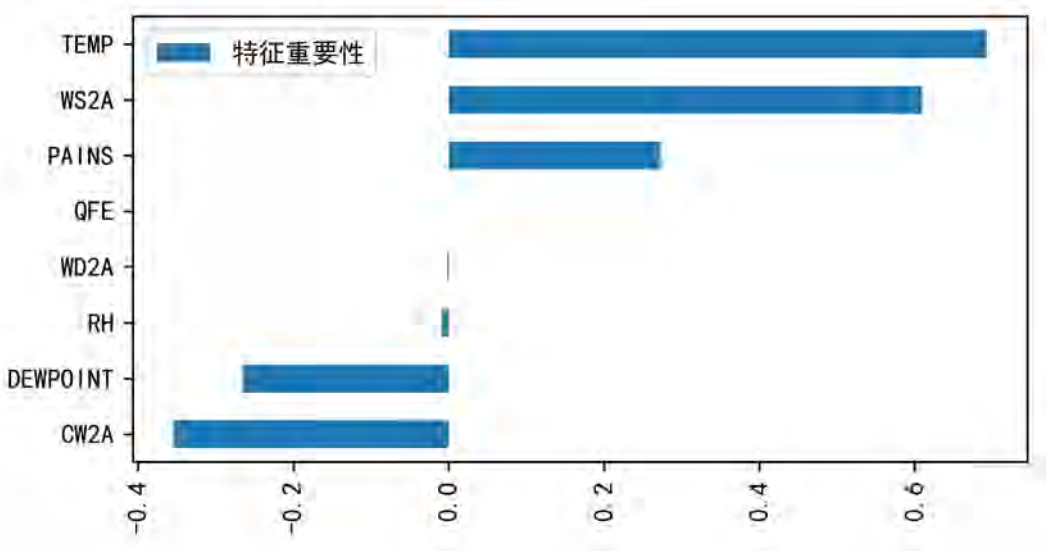

图 3-10 各影响因子的特征重要性

由图可知，TEMP、WS2A 和 PAINS 呈现出明显的正向重要性，而 DEWPOINT 和 CW2A 则呈现明显的负向重要性，QFE、WD2A 和 RH 则并不突出。

综合考虑上述两个方法的结果，TEMP 和 WS2A 同时具有较好的相关性和特征重要性，应当在模型参数中。尽管 PAINS 表现出较高的特征重要性，但其相关性过低，在不同天（季节）的相关性具有较大差异，故而不考虑。DWPOINT 作为重要的气象指标，可由 TEMP 和 RH 推导求得，并非直接观测值，故不考虑。CW2A 与 WS2A 间具有较高的相关性，由于雾的形成、消散主要考虑水平方向的风力大小，故不考虑 CW2A。其他参数，在两个指标上均未体现出重要作用。

3.3 模型建立

通过观察能见度与气象因子之间的相关性，选择 “RH 相对湿度”、“TEMP 温度”、“WS2A 2 分钟平均风速”作为影响因子进行建模，以减少模型的复杂程度，降低其他不相关因子的异常干扰。

通过观察，发现尽管能见度与 3 个影响因子之间具有较高的相关性，分别达到了 0.89、0.87 和 0.63，但直接进行线性拟合的效果较差，模型的泛化能力较差。选用非线性模型可以使得模型适用于更为广泛的数据，多元多项式拟合的方法属于此类，具有较好的拟合效果。

经过测试，三次多项式的拟合效果最好，三元三次多项式拟合的公式如下：

$$\begin{aligned} y = {} & b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_{11}{x_1}^2 \\ & + b_{22}{x_2}^2 + b_{33}{x_3}^2 + b_{12}x_1x_2 \\ & + b_{13}x_1x_3 + b_{23}x_2x_3 + b_{111}{x_1}^3 \\ & + b_{222}{x_2}^3 + b_{333}{x_3}^3 \\ & + b_{123}x_1x_2x_3 + b_{112}{x_1}^2x_2 \\ & + b_{113}{x_1}^2x_3 + b_{221}{x_2}^2x_1 \\ & + b_{223}{x_2}^2x_3 + b_{331}{x_3}^2x_1 \\ & + b_{332}{x_3}^2x_2 \end{aligned} \tag{3-1}$$

式中，x_1、x_1、x_1分别表示 RH 相对湿度、TEMP 温度、WS2A 2 分钟平均风速；b_i为多项式系数。

使用拟合优度（R^2）对多项式拟合的结果进行评估，公式如下：

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} \tag{3-2}$$

使用均方根误差（$RMSE$）对多项式拟合的结果进行评估，是用来衡量观测值同真值之间的偏差，公式如下：

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)}^2 \tag{3-3}$$

使用平均绝对误差（MAE）对多项式拟合的结果进行评估，能更好地反映预测值误差的实际情况，公式如下：

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|(y_i - \hat{y}_i)| \tag{3-4}$$

3.4 模型结果

不考虑白天与晚上本身亮度对能见度影响造成差异时，b_i结果依次为：

[1.8590482332e+10 -1.88122212e+07 -5.48781654e+07 9.83003079e+06
7.04697608e+04 3.57621151e+04 3.35147003e+04 5.39999766e+04
-1.93843900e+04 4.37121933e+03 2.93977361e+01 -6.95718377e+01
-3.98385426e+00 -1.69745632e+01 -3.30681712e+01 7.62910936e+00
-1.77122081e+01 9.55647218e+00 -4.26074222e+00 -1.54523326e+00]

12.15 和 3.12 两天整体的拟合优度达到了 0.94 和 0.96。

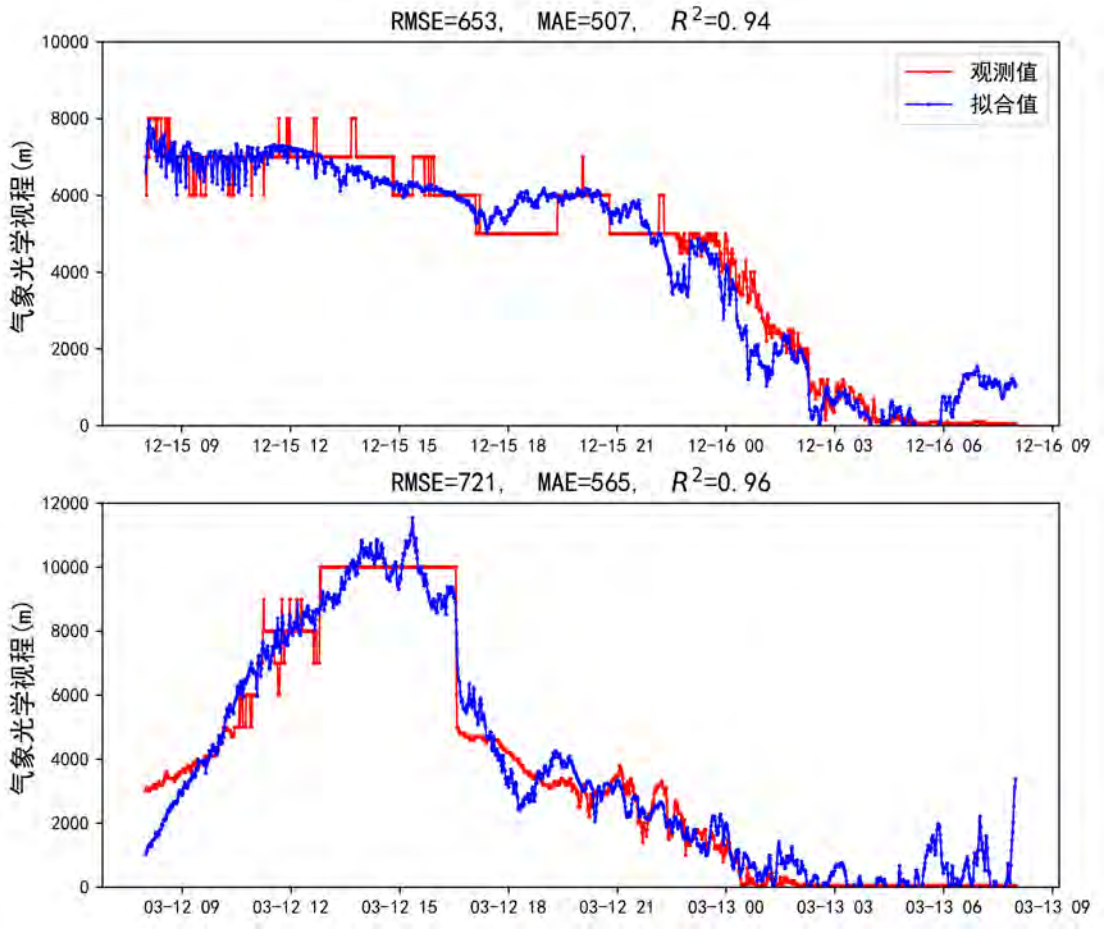

图 3-11 能见度拟合值与观测值

（上：12 月 15 日全天；下：3 月 12 日全天）

仅考虑白天时，b_i结果依次为：

[6.214886060e+09 -8.25569933e+07 -1.80403129e+07 3.08751576e+06
-7.79194598e+04 1.62642420e+05 4.93004773e+03 1.74540258e+04
-6.68205212e+03 2.34564668e+04 -4.51425366e+01 7.70993357e+01
-2.08671582e+00 -8.01002128e+01 -5.47685522e+00 2.11436722e+01
-5.62839777e+00 3.57675577e+00 -2.24364298e+01 -1.44511317e+01]

12.15 和 3.12 两天白天（8:00-20:00）的拟合优度达到了 0.69 和 0.94。

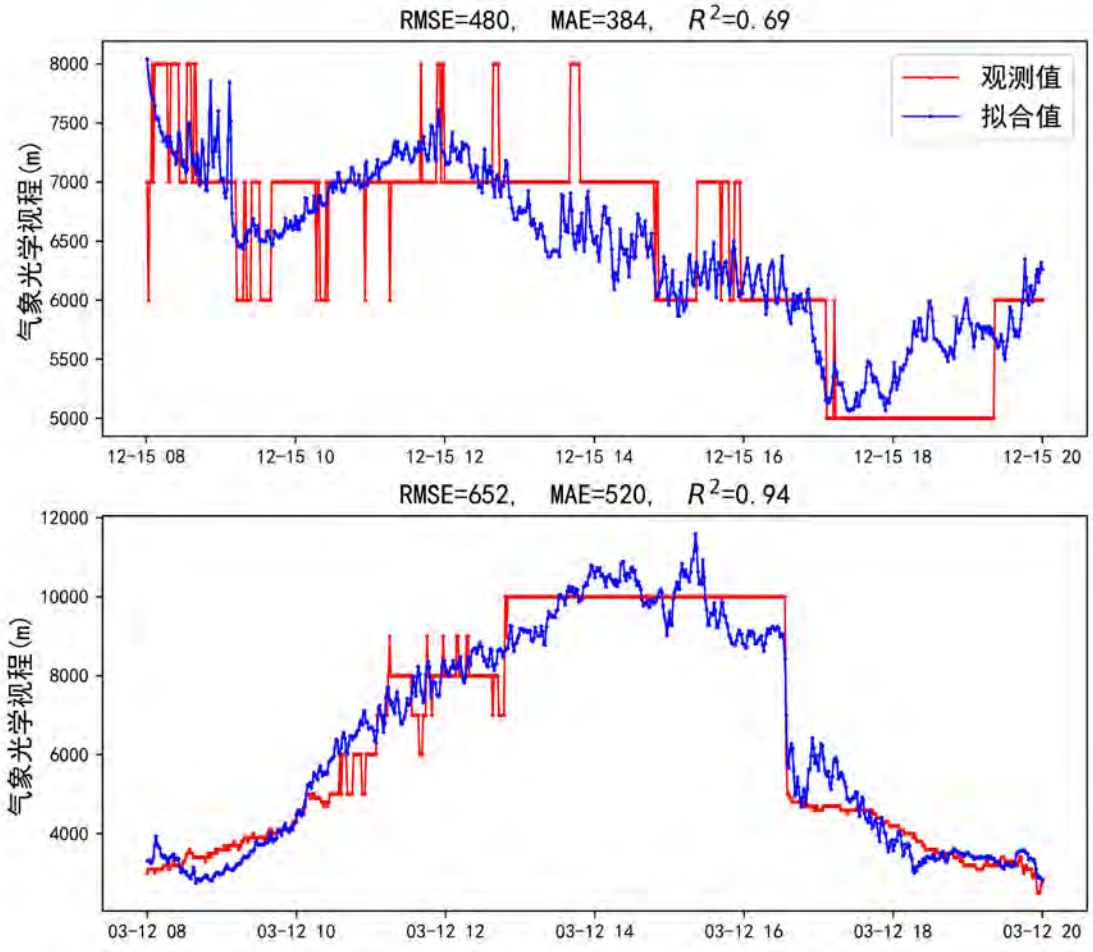

图 3-12 能见度拟合值与观测值

（上：12 月 15 日白天；下：3 月 12 日白天）

仅考虑晚上时，b_i结果依次为：

[1.9639045393e+10 -7.64828825e+07 -5.78489400e+08 -5.78341255e+07
4.43286456e+04 1.49334367e+05 3.25556802e+04 5.68014942e+05
1.10964664e+05 1.22001863e+05 -3.84920073e+00 -4.37288500e+01
6.71511022e+00 -7.28913420e+01 -3.18237248e+01 -8.47356258e+00
-1.85914585e+02 -5.31949949e+01 -1.19904196e+02 4.14832098e+00]

12.15 和 3.12 两天晚上（20:00-次日 8:00）的拟合优度达到了 0.94 和 0.81。

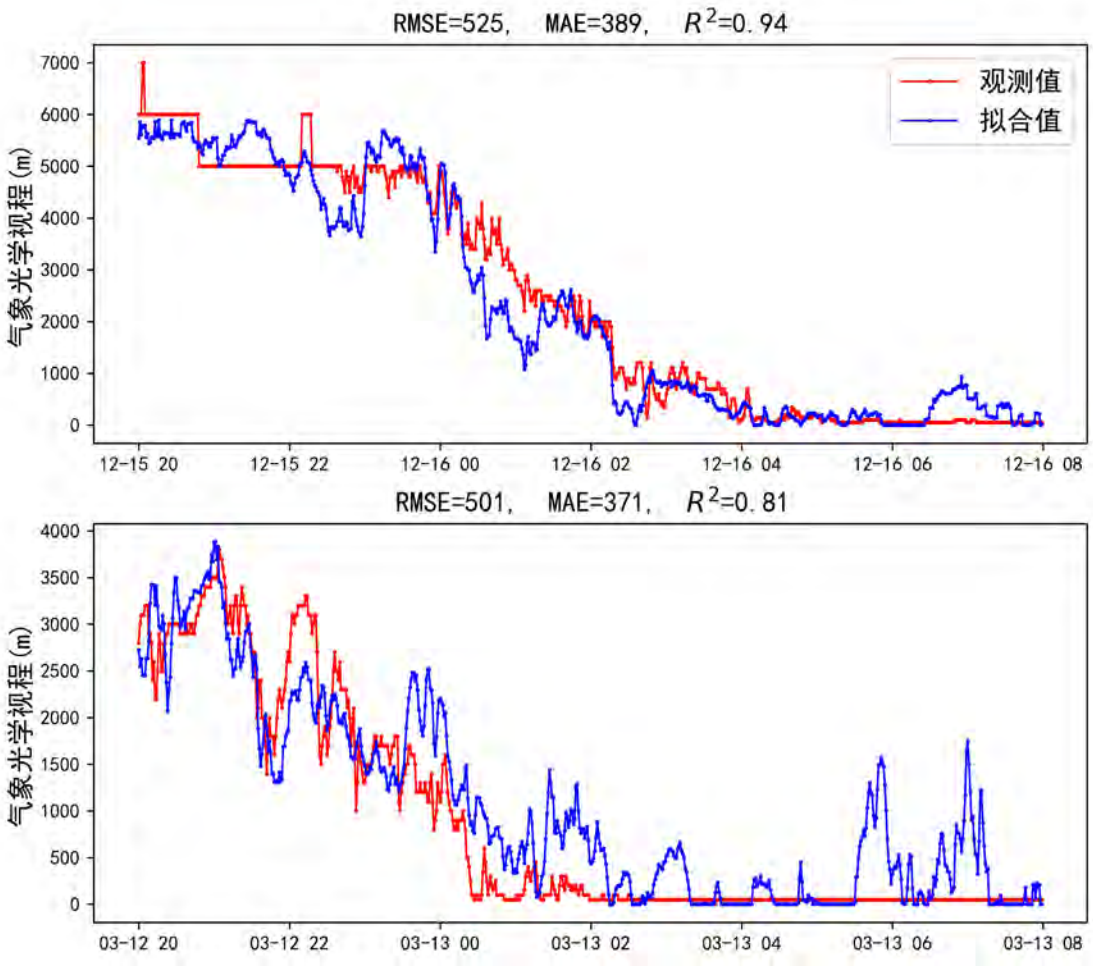

图 3-13 能见度拟合值与观测值

（上：12 月 15 日晚上；下：3 月 12 日晚上）

综上所述，模型精度如下：

表 3-4 不同时间的拟合结果精度验证

时间	R^2	$RMSE$	MAE
12.15 全天	0.94	653	507
3.12 全天	0.96	721	565
12.15 白天	0.69	480	384
3.12 白天	0.94	652	520
12.15 晚上	0.94	525	389
3.12 晚上	0.81	501	371

从上表中可以发现，三元三次多项式拟合具有较好的效果。考虑全天数据时，拟合优度能够达到 0.94、0.96。尽管区分白天、晚上后，拟合优度R^2有一定下降，但从$RMSE$、MAE两个指标上来看，均有明显的改进。从拟合结果上来看，12.15 白天R^2下降的主要原因是 AMOS 传感器的观测数据存在明显异常，具体表现为在短时间内存在较大的突变。

综上所述，考虑白天、晚上不同光照条件下的三元三次多项式拟合效果较好，能够满足模型需求。

四、问题二的建模与求解

4.1 任务二问题分析

任务二提供了某机场的视频数据和 AMOS 观测的能见度数据，要求建立基于视频数据的能见度估计深度学习模型，并对其精度进行评估。视频数据是是由一组连续的图像构成的，其每幅图像具有时间属性，也表现出一定的空间范围的信息，属于常见的时空数据。通过视频数据可知，每秒有 25 帧图像，时间跨度是 2020 年 3 月 13 日 0 时 0 分 26 秒开始，当日 11 时 47 分 48 秒结束。视频的观测点是固定的，但是视频的观测视角和视野范围约在 9 时 13 分后存在变化。另外，由于雾气较重、能见度较低，视频部分数据无法体现景物信息，目标观测物体已完全被景物遮挡。需要指出的是，由于观测设备质量、大气颗粒物等影响，视频的部分图像存在色彩、亮度突变的异常情况，同时也存在大量的噪声点。视频数据属于二维数据，而机场 AMOS 数据属于一维数据，时间分辨率为 15 秒，时间跨度是 2020 年 3 月 13 日 0 时 0 分 26 秒开始，次日 7 时 59 分 45 秒结束，两者的时间跨度和分辨率和不同。基于以上的情况分析，需要根据 AMOS 能见度观测时间提取视频中的关键帧，以此将某时刻的图像和能见度进行匹配。

为了建立视频数据和能见度数据之间的深度学习模型，神经网络（Artificial Neural Networks，ANN）是必不可少的工具，其基于梯度下降进行参数寻优，具有较强的学习能力。通常对于二维图像数据，卷积神经网络（Convolutional Neural Networks，CNN）对于图像结构化特征提取具有很大优势，也广泛应用于目标检测、语义分割等场景。但由于本题的图像数据存在上述异常情况，其局部特征对于能见度的反演能力有限，并且图像数据量有限，直接基于图像进行深度学习并不合适。因此，需要对图像数据进行预处理，以提高图像的质量，增强图像的特征，减少噪声等随机信息对模型收敛的影响。基于特征工程的思想，本文先提取每一幅图像的特征信息，再采用全连接神经网络进行学习，这样的深度学习模型具有收敛速度快、计算成本低、可解释性高的优势。

4.2 任务二数据分析与预处理

首先对给出的机场视频数据（Fog20200313000026.mp4）进行分析，视频总时长 11 时 33 分 28 秒，视频分辨率为 1280×720，图像为三通道图像，帧率为 25 帧/秒，拍摄的时间跨度是 2020 年 3 月 13 日 0 时 0 分 26 秒开始，当日 11 时 47 分 48 秒结束。通过 Adobe Premiere Pro 软件观察数据发现，该视频帧率和左上角水印时间匹配存在些许差异，每秒约为 24~26 帧。为了将视频数据与能见度数据进行时间上的匹配，首先获取能见度和视频的共同观测部分，即从 2020 年 3 月 13 日凌晨 0 时 0 分 30 秒至上午 7 时 59 分 45 秒。其中，在 0 时 47 分至 1 时 1 分间视频数据缺失，另在 1 时 39 分 45 秒、4 时 11 分 15 秒 AMOS 数据缺失的情况。在此基础上，根据 AMOS 能见度的时间分辨率，时间（每隔 15 秒）提取出重要的关键帧，最终将连续的视频数据转换为 1860 幅图像，分辨率仍保留为 1280×720。

通过这 1860 幅图像可以发现，在大雾天气下，可能由于视频传感器通道和质量问题，图像出现了紫色、蓝色、黄色等异常情况，如图 4-1 所示。另外，大雾天气下，水汽和气溶胶颗粒分布在空气中，对夜间灯光会产生不稳定的散射，相邻两幅图像的背景灯光亮度会产生较大变化，如图 4-2 所示。

图 4-1 异常一：图像颜色

图 4-2 异常二：相邻图像灯光亮度

4.3 图像预处理（去噪、增强、掩膜）

基于以上分析，首先利用不同滤波器对图像进行去噪，通过比较确定适用于的本题的滤波算法。一般图像的噪声可以分为椒盐噪声和高斯噪声，椒盐噪声就是在图像上随机出现黑色白色的像素；高斯噪声是指它的概率密度函数服从高斯分布（即正态分布）的一类噪声，与椒盐噪声相似，区别在于椒盐噪声是出现在随机位置、噪点深度基本固定的噪声，

高斯噪声与其相反，是几乎每个点上都出现噪声、噪点深度随机的噪声。本题的数据同时存在两种噪声，且不同图像的噪声分布范围大且不稳定。

图像去噪是指去除噪声图像中的噪声，从而还原真实图像。但是，由于噪声、边缘和纹理都是高频分量，在去噪过程中很难区分它们，因此去噪后的图像不可避免地会丢失一些细节。总体而言，从有噪声的图像中恢复有意义的信息以获得高质量的图像是一个重要的问题，分别采用中值滤波、均值滤波和高斯滤波，其去噪结果如图 4-3 所示。

中值滤波是一种非线性空间滤波器，它将图像滤波器包围的区域中像素的统计排序的中位数值代替中心像素的值；均值滤波器的输出是包含在滤波掩模领域内像素的简单平均值；高斯滤波是一种线性平滑滤波，整幅图像的每一个像素点的值，都由其本身和邻域内的其他像素值经过加权平均计算后得到，适用于消除高斯噪声。但是三种方法对于能见度较低的图像并不适用。

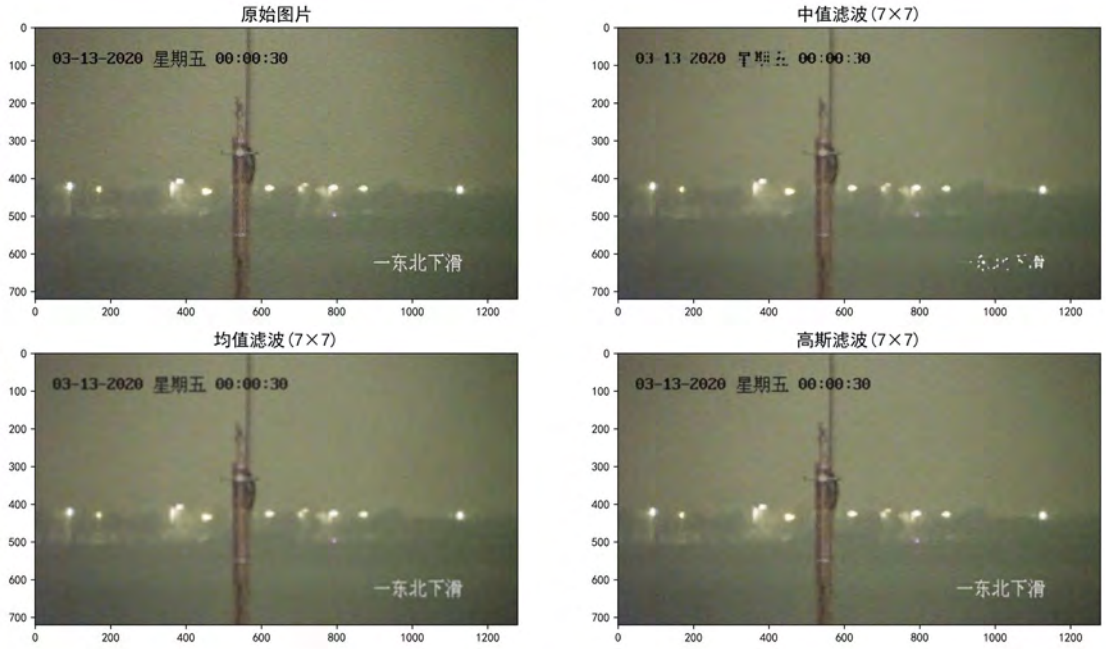

图 4-3 三种滤波方法的比较

非局部均值（Non-Local Means）是近年来提出的一项新的去噪算法，其充分利用了图像中的冗余信息，并且能最大程度地保持图像的细节特征。基本思想是利用整幅图像进行去噪，它以图像块为单位在图像中寻找相似区域，再对这些区域进行加权平均平均，能够较好地滤除图像中的噪声。该算法适用于短时间的图像序列，对本题去噪具有较大的优势。

设含噪声图像为v，去噪后的图像为$\tilde{u}$。$\tilde{u}$中像素点x的灰度值可通过如下公式进行计算：

$$\tilde{u}(x) = \sum_{y \in I} w(x,y) \times v(y) \tag{4-1}$$

其中，$w(x,y)$表示像素点x和y间的相似度，由以x和y为中心的矩形邻域$V(x)$、$V(y)$间的距离$\|V(x) - V(y)\|^2$决定。

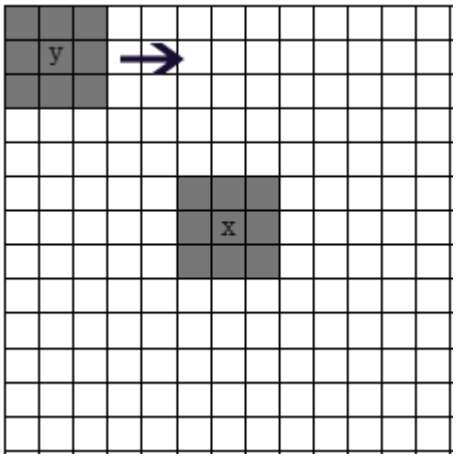

图 4-4　非局部均值算法示意图

$$w(x,y)=\frac{1}{Z(x)}exp\left(-\frac{\|V(x)-V(y)\|^2}{h^2}\right) \tag{4-2}$$

其中，

$$Z(x)=\sum_{y}exp\left(-\frac{\|V(x)-V(y)\|^2}{h^2}\right) \tag{4-3}$$

$$\|V(x)-V(y)\|^2=\frac{1}{d^2}\sum_{\|Z\|_\infty\le ds}\|v(x+z)-v(y+z)\|^2 \tag{4-4}$$

$Z(x)$为归一化系数，h为平滑参数，决定过滤器强度，可以控制高斯函数的衰减程度。h取值越大，高斯函数变化越平缓，去噪效果越好，但图像会越模糊；h取值越小则相反，能够保留更多细节，但也会残留较多的噪声点。在本模型h取值 10，能够较好的去除噪声，同时保留更多细节，结果如图 4-5 所示。

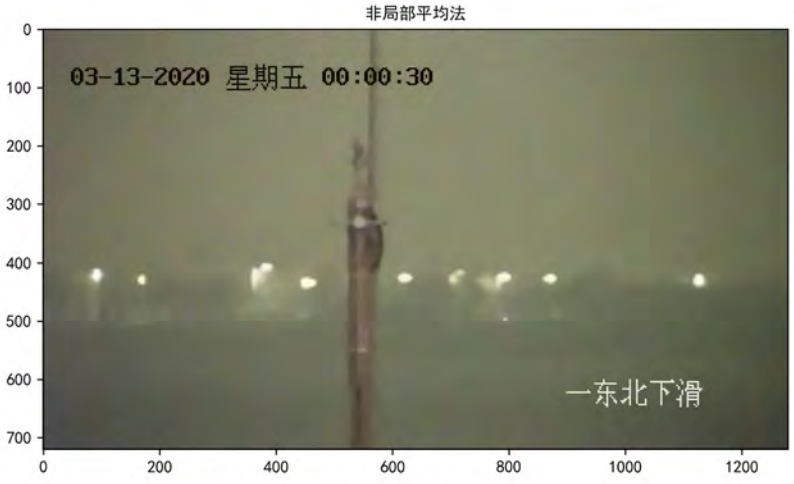

图 4-5 非局部平均滤波法去噪效果

由于大气中水汽含量高、能见度较低，视频监控数据的对比度有待加强，以突出前景

物体和背景的差异。在本模型中，采用了 Gamma 增强和提高对比度结合的方法。

Gamma 增强采用指数函数对图像的灰度值进行变换，基本公式如下：

$$V_{out} = AV_{in}^{y} \tag{4-5}$$

其中，A是一个常数，通常取 1，当$y > 1$时，Gamma 增强对图像的灰度分布直方图具有拉伸的作用，反之则为收缩。本题的 A 取值为 2。

由于 Gamma 增强后的图像背景明显变暗，背景的差异较不明显，再进行图像亮度的对比度的调整，对图像进行混合加权，得到最终的效果图。

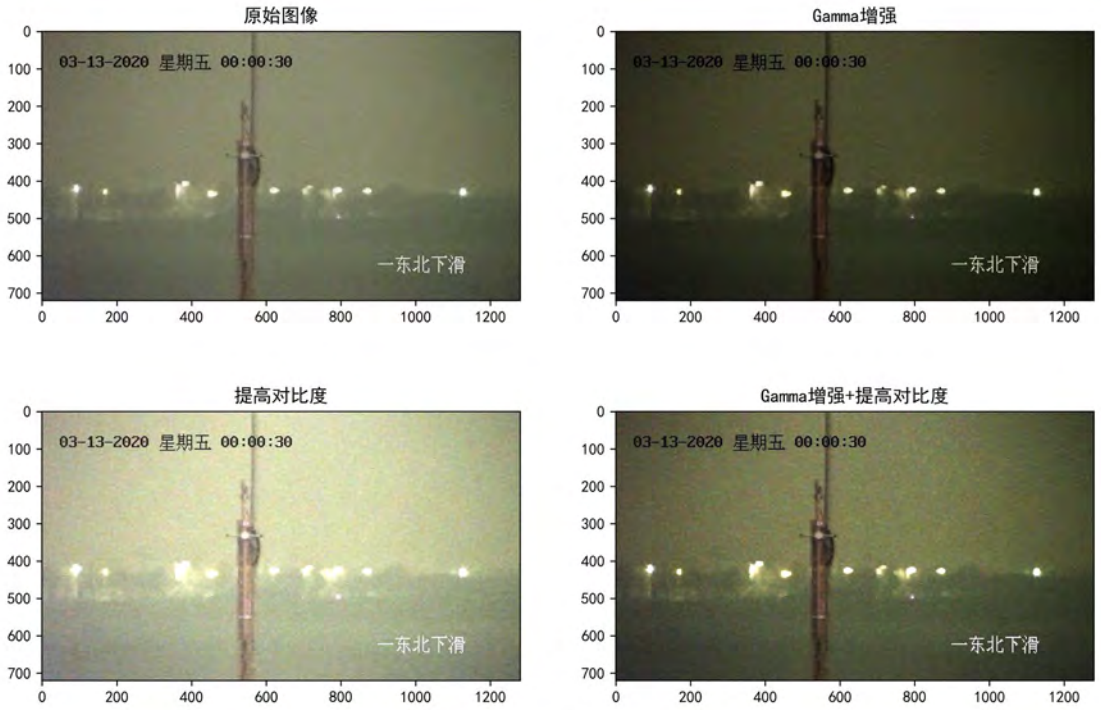

图 4-6 图像增强的方法比较

由于图像左上角的观测时间会发生改变，以及右下角的文字本身不属于图像内容，故予以进行掩膜处理。

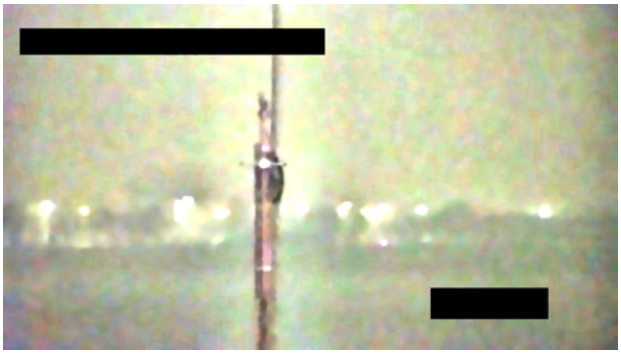

图 4-7　图像掩膜结果

4.4 尺度不变特征变换（SIFT）特征提取

尺度不变特征变换（Scale Invariant Feature Transform，SIFT）是一种特征提取方法，其提取的是一种非常稳定的局部特征，具有放缩不变性、旋转不变性，抗光照变化，抗视点变化的优点。SIFT 算法一共由 4 个部分组成：尺度空间极值检测、特征点定位、特

征方向赋值、特征点描述。

在查找特征点之前，需要得到所有的尺度空间中的极值点作为候选，一般使用线性的高斯卷积核进行尺度空间的变换，并建立尺度空间。

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \tag{4-6}$$

$$L(x,y)=G(x,y,\sigma)*I(x,y) \tag{4-7}$$

完成尺度空间的建立及对应尺度空间中的极值点选取后，需要对所有的候选点进行下一步的筛选，剔除低对比度的特征点和不稳定的边缘响应点后，即完成特征点的定位。随后计算每个特征点的局部特征，得到梯度幅度大小和梯度方向。

$$\begin{cases} m(x,y)=\sqrt{[L(x+1,y)-L(x-1,y)]^2+[L(x,y+1)-L(x,y-1)]^2} \\ \theta(x,y)=\arctan\dfrac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)} \end{cases} \tag{4-8}$$

最后，只需要使用特征向量对特征点进行描述，即可得到对于一幅图像的特征提取过程。

SIFT 特征提取具有多量性，即使少数的几个物体也可以产生大量的 SIFT 特征向量；可扩展性，可以很方便的与其他形式的特征向量进行联合。对于能见度较低，观测物体较少的图像具有较大优势。在提供的数据中挑选了图像质量相对较高的第一帧影像（00:00:30）进行特征点的提取，总共从我们的图像中提取了 123 个特征点，如图 4-8 所示。之后从后续所有样本中提取相同位置的特征点，并计算所有特征点的特征向量，展平在一维空间上作为后续的输入向量。通过对 1860 张图片的提取，我们总共获得了一个 1860 ×15766 维的矩阵作为所有样本特征。

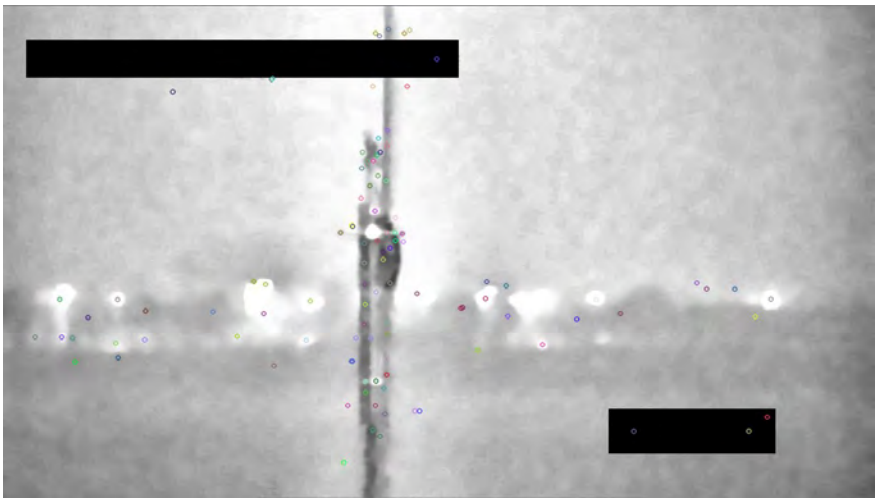

图 4-8 SIFT 算法的图像特征点检测结果

4.5 训练样本的分布及选取

在与视频成功匹配的结果共 1860 条能见度数据中，可以看到能见度的全部样本数据为 0-50m 区间内占比极高（如图 4-9），高达 78.6%，而 1000m 以上能见度的样本只占全部数据的 1.5%。需要特别强调的是，训练集的样本不均衡势必会导致深度学习的预测结果趋于样本多的那类情况，这样的模型虽然拟合精度高，但其实属于一种错误的学习，其并不能有效得将图像与能见度建立联系。由于深度学习的学习能力很强，很容易出现过拟合现象，而样本的不均衡可能掩盖了这种过拟合现象。

为了得到可靠的深度学习模型，输入的训练数据至关重要，本文先去除大部分 50m 能见度的样本，以保证训练数据的能见度分布较为均衡。但是，将所有 50m 能见度的样本作为测试集来评价模型，是不合理的。因此，在样本均衡的能见度中进行分层抽样，将 70%的样本作为真正的训练集，剩余的 30%样本归入测试集，这样的测试集也包含了不同的能见度样本。

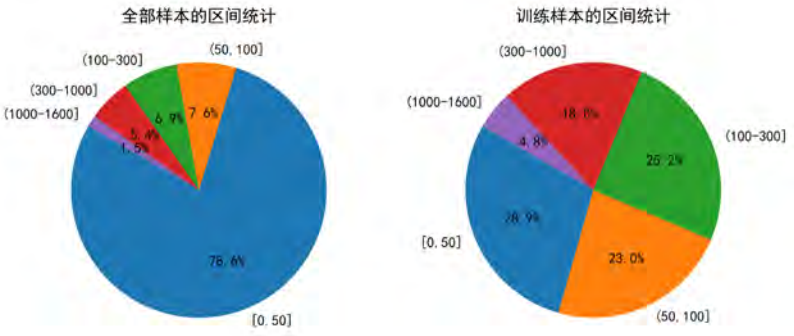

图 4-9 全部样本数据与训练样本数据的标签区间分布

4.6 模型实现与评估

经过图像特征点的提取，训练样本和测试样本的划分后，即获取了深度学习模型的输入数据。传统 BP 神经网络算法及其改进算法都是非完全全连接神经网络算法，具有收敛速度慢，泛化能力差等不足。本文构建的是全连接神经网络模型，为了防止过拟合的情况发生，我们在全连接层之间设置了 Dropout 层，使得在训练过程中使部分连接层停止工作，以避免其对于某些局部特征的过度依赖，这样可以进一步提高模型的泛化能力。

经过 200 轮次的训练，得到了全连接神经网络模型在训练集和验证集上的损失函数变化曲线（图 4-10 所示），损失函数呈现明显的下降趋势，并随着训练的深入达到一个平衡的状态，可以看到在训练集和验证集上，损失函数的表现具有一致性，说明我们的模型并没有出现明显的过拟合现象。

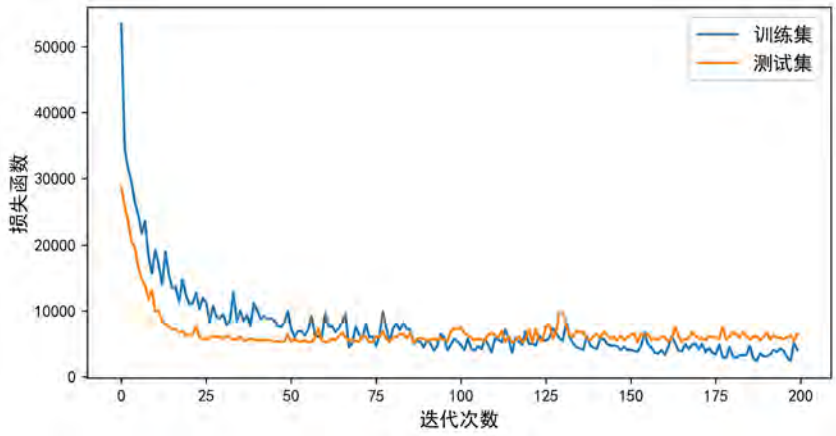

图 4-10 模型在训练过程中的损失函数变化

图 4-11 展示了模型经过训练后在训练集和验证集上的预测结果和实际结果的比较，并计算三个模型的评估指标：平均绝对误差（MAE）、均方根误差（RMSE）和拟合优度（R^2）。模型在训练集上的拟合优度 R^2 值为 0.94，说明模型对于训练集上的特征进行了很好的学习，而在验证集上的拟合优度 R^2 值为 0.86，也证明了模型的泛化能力较强，能较好地根据图像预测不同能见度。模型评估时的 RMSE 为 80，MAE 为 47，较训练集略有上升。但需要指出的是，随着能见度的升高，深度学习模型的模拟误差也在增大，而且预测值比实际值偏低，这可能与能见度高的训练样本较少有关。

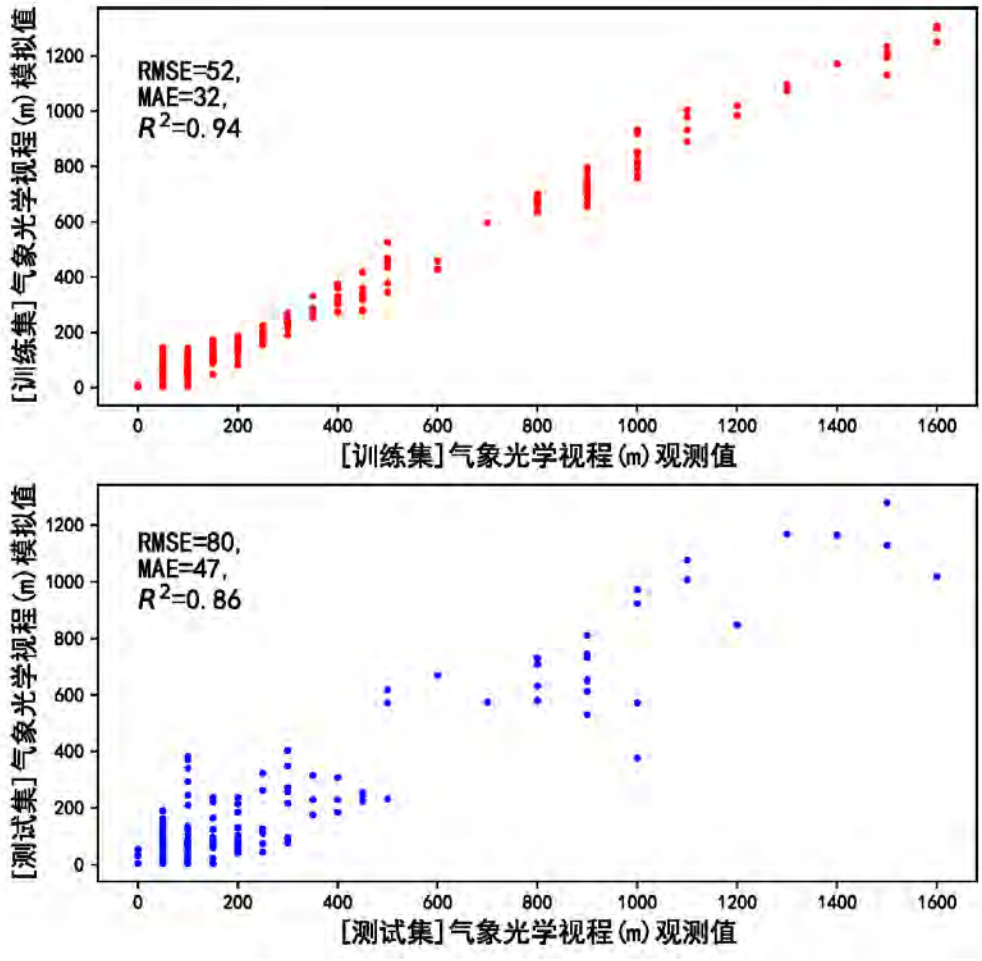

图 4-11 全连接神经网络模型在训练集和验证集上的预测结果的比较

4.7 问题二结果总结

图 4-12 展示了基于视频图像的能见度深度学习模型实现和技术路线，我们进行了以下的工作：（1）对视频进行了帧提取，并利用影像去噪、影像增强的手段对数据质量进行修正；（2）对标签数据进行统计和划分，避免了某一类数据在训练集和验证集中过于密集的情况；（3）尝试使用 SIFT 算子对图像特征点进行提取，并结合神经网络提出我们的神经网络深度学习模型。

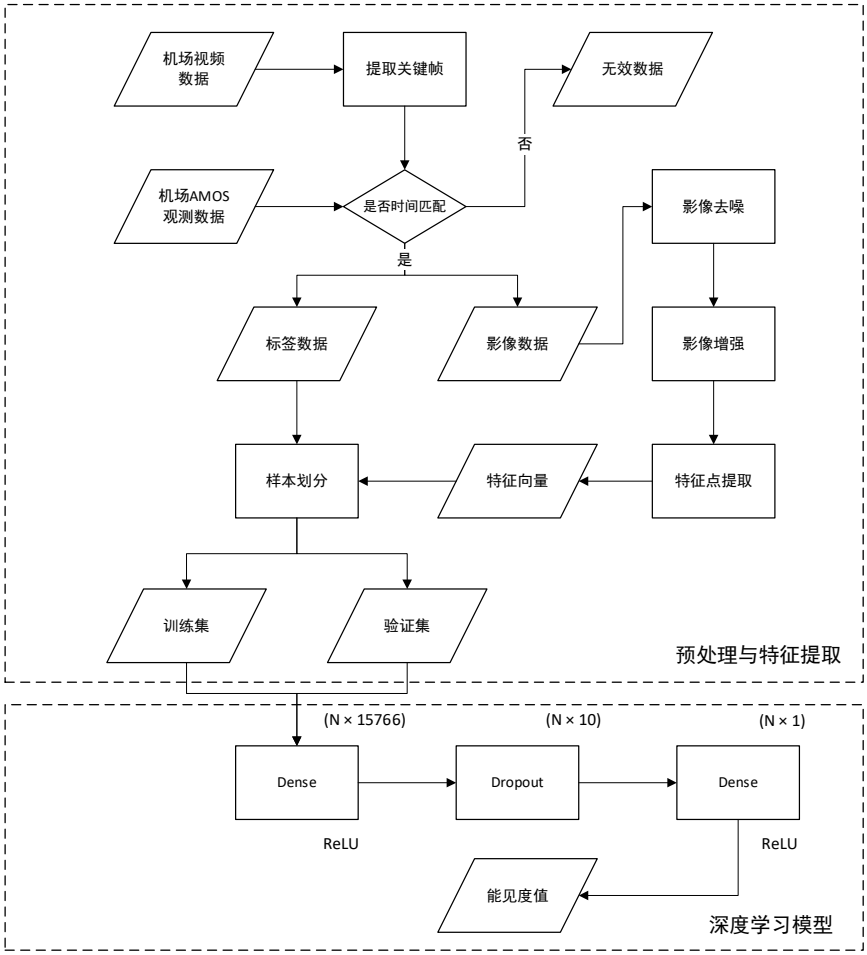

图 4-12 基于图像特征的深度学习模型构建流程

最后对我们的模型进行评估，可以看到该模型在训练集和验证集上均有出色的表现，可以用于视频数据（切分为帧图像）的能见度预测。但也需要指出，受限于训练数据的质量和特征提取方法，模型的精度和复杂度可能还有提升的空间。

特别的是，本题本质上虽属于回归问题，但仍需考虑训练数据（能见度）的分布不均衡带来的影响，由于 50m 能见度的数值占全部样本的 75%以上，存在较为严重的不均衡，本次深度学习模型的训练数据的能见度分布是较为均衡的，不会出现因 50m 能见度数据过多而导致全部预测成 50m 的情况，模型具有较强的泛化能力。

五、问题三的建模与求解

5.1 问题分析

题目要求在不依赖能见度仪观测数据的前提下，仅根据高速公路的 100 张视频监控图像进行建模估算能见度。通过截图发现，这 100 张图像的能见度均较低，景物也较为单调。如何充分利用图像中的特征以及如何分析具有时间序列的图像是进行建模求解的关键。本文尝试了多种不同的方法：（1）直接根据单目图像计算景深（Depth of Field）；（2）选定感兴趣区域（Region of Interest，ROI），计算所有 ROI 区域的平均亮度值的变化；（3）通过图像增强算法，利用边缘检测算子获取公路标线的可见长度，以此来推算能见度；（4）基于暗通道先验算法，根据不同景深的物体对比度差异，计算图像的大气消光系数和大气透射率，并根据大气消光系数估算能见度。需要指出的是，方法（1）对周边环境的丰富度依赖较大，低能见度的高速公路可能并不适用；方法（2）得到的变化曲线难以和能见度直接建立映射关系，但只能间接体现能见度的变化趋势；方法（3）从数学角度出发，需要对图像进行建模，寻找灭点进行透视变换，获取像素距离与实际距离的关系；方法（4）从基于图像特征进行物理建模，且区域的选取相对比较灵活，可以充分利用图像的特征。本文讨论了不同建模方法的优缺点以及遇到的问题，并最终选用了方法（4）进行建模。

5.2 数据预处理

所用的视频数据是 2016 年 4 月 14 日 6 点 30 分 26 秒至 7 点 39 分 11 秒的 100 张高速公路监控视频截图。由于目视解译发现图像的变化不够明显，大部分区域被雾气笼罩，能见度较低。为了更好地分析图像上不同像素点随时间的变化幅度，根据 100 张图像作了方差计算，发现第 35 张图像出现了一辆车子。除去第 35 张后的重新进行方差计算，发现变化幅度较大的区域仍然位于公路线条。

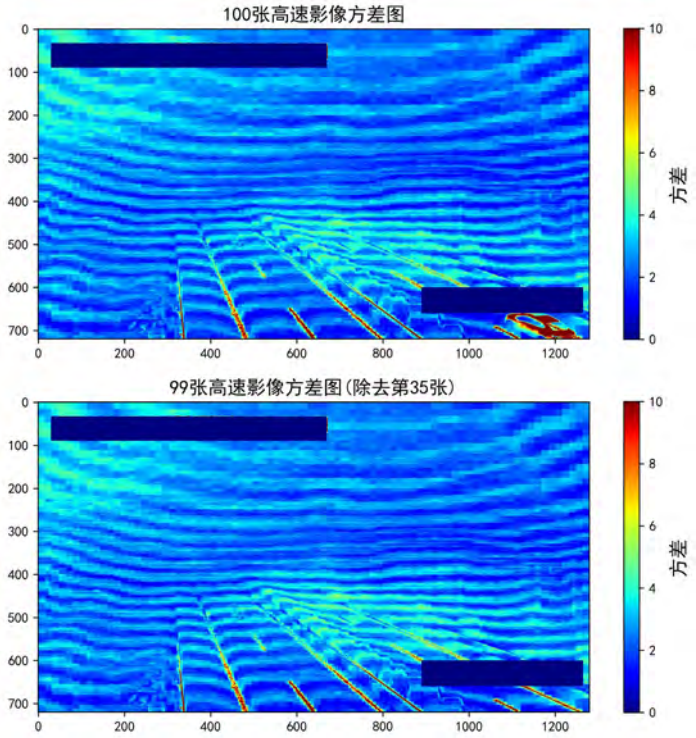

图 5-1 高速视频图像的方差

与机场视频图像类似，高速图像数据中存在左上角黑色的时间和右下角白色的道路信息的文字和数字标记。但不同于之前问题二中的处理，本题中水印所在的上部和左侧包含的图像信息可利用价值不高，因此采用了图像裁剪，直接从原始图像中删除水印区域，以下方红色框线内的范围作为研究区域，裁剪前后对比如图 5-2。

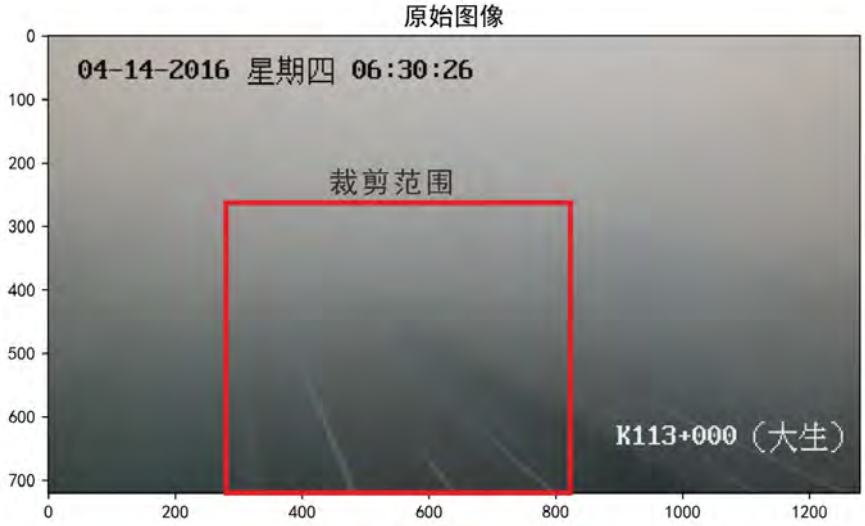

图 5-2 裁剪范围对比图

5.3 单目深度估计

图像是立体场景的投影，投影只捕获了平面信息，因此单目深度估计（Monocular Depth Estimation）很具挑战性。最新的单目深度估计算法 Monodepth2 是一种利用深度学习的方法（https://github.com/nianticlabs/monodepth2），自监督学习图像中的逐像素的尺度深度的技术。该深度学习模型已发布预训练模型，在 KITTI 数据集的基准上取得了高质量的推演结果[1]。对于裁剪之后的图像计算景深的结果如图 5-3，发现深度估计图像的变化不平滑，会出现“孔洞”的现象，这不符合实际情况。这是由于低能见度的图像中存在的景物信息较少，景物之间缺乏对比，得到的景深结果效果不佳，因此该方法在本题不合适。

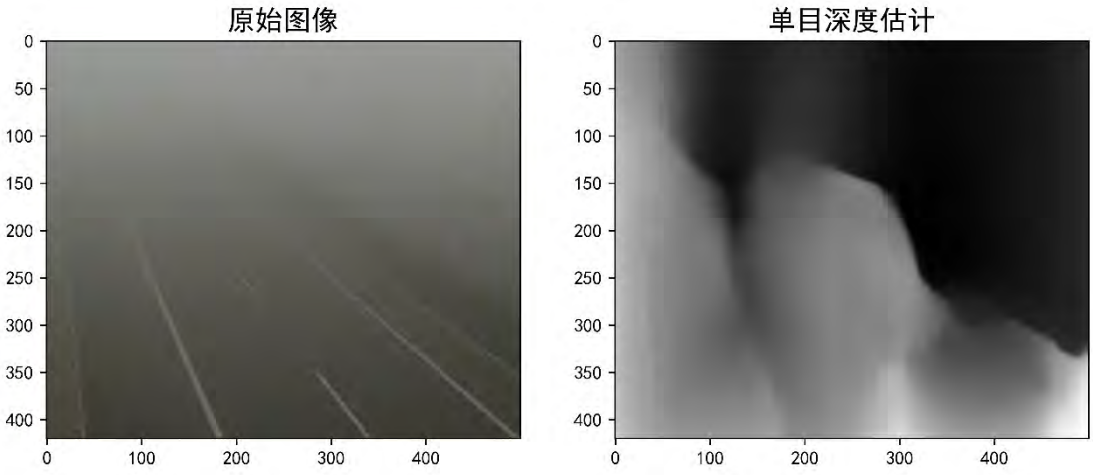

图 5-3 原始图像与单目深度估计图

5.4 公路标线可见长度估计算法

在大雾天气下，公路标线是具有明显变化特征的物体，但由于高速公路的标线与背景

颜色较为接近，因此边缘检测算法难以准确地捕捉道路可见距离的最远点。为了更好的进行公路标线检测，需要先对原始图像进行增强。Retinex 是以色感一致性（颜色恒常性）为基础的，即物体的颜色是由物体对长波（红色）、中波（绿色）、短波（蓝色）光线的反射能力来决定的，而不是由反射光强度的绝对值来决定的，物体的色彩不受光照非均匀性的影响。不同于传统的线性、非线性的图像增强方法，Retinex 可以在动态范围压缩、边缘增强和颜色恒常三个方面达到平衡，因此可以对各种不同类型的图像进行自适应的增强。此题选用了基于 Retinex 理论算法的改进算法——彩色保护多尺度（MSRCP）[2]，得到结果如图 5-4 所示，公路标线得到了显著的突出效果。

图 5-4　彩色保护多尺度（MSRCP）图像增强效果图

图像边缘检测常用算法有 Sobel、Prewitt、Prewitt、Canny 算子，其中 Canny 算法更为优异，提出了（1）基于边缘梯度方向的非极大值抑制；（2）双阈值的滞后阈值处理。但是为了更有效得提取公路标线，希望对直线进行检测，此处需要用到霍夫变换（HoughLine），用于分离图像中特定形状的特征，可以容忍特征边界描述的误差，并且不受噪声的干扰，在本题中具有较好的优势，能够较为精确地反映道路的可见长度（图 5-5）。

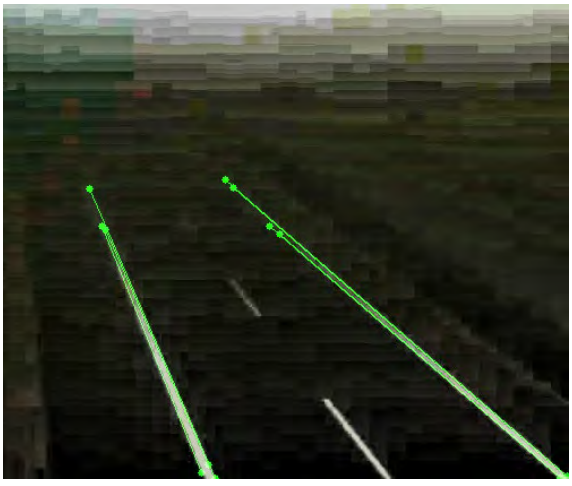

图 5-5 Canny 算子与霍夫变换（HoughLine）直线检测算法效果图

为了建立图像坐标系和真实坐标系之间的关系，则必须知道图像上某一段的实际距离。高速公路上的标线是有标准的，一般宽约为 20cm，长度为 6m，间隔为 9m。这对于本题基于单目图像测算实际距离具有很大帮助。对于测算距离而言，需要找到图像的灭点（Vanishing Point，立体图形各条边的延伸线所产生的相交点），如图 5-6 所示。

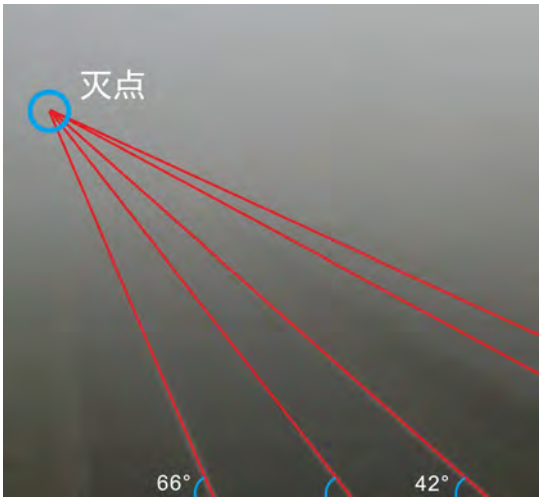

图 5-6 高速视频截图的灭点示意图

对于霍夫变换直线检测算法的结果进行透视变换（如图 5-7），但在本题中，由于高速道路的监控设备高度未知，故无法通过透视变化后的图片与实际距离产生映射关系。因此较难直接估算出能见度，故采用以下暗通道先验法进行建模。

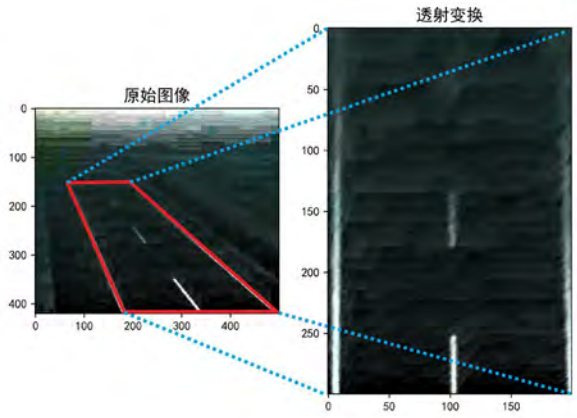

图 5-7 透视变换算法效果图

5.5 暗通道先验法

暗通道先验法（Dark Channel Prior，DCR）是何恺明提出的一种突破性的图像去雾算法[3]，经过多次的验证，何恺明总结了在非天空的局部区域中，某一些像素及其周围区域会存在至少一个颜色通道具有很低的值的规律。在实际的计算中，取图像中每个像素点 RGB 三通道的最小值，并进行一次最小值滤波，得到暗通道灰度图（如图 5-8）。

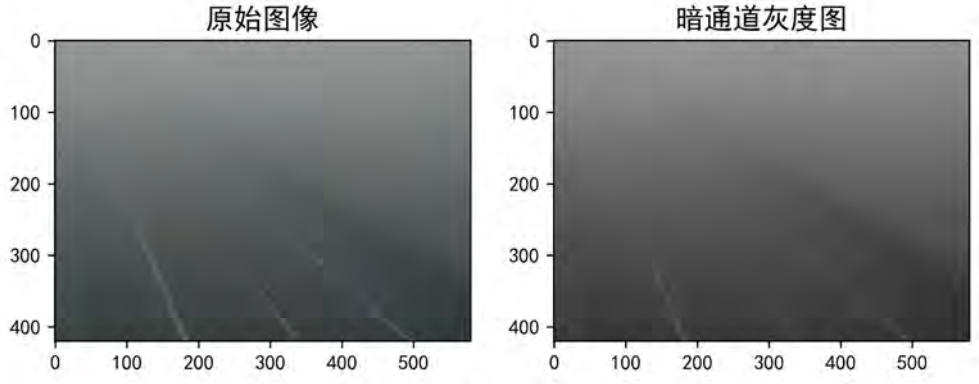

图 5-8 暗通道灰度图

在计算机视觉领域，雾图形成模型应用广泛（公式 5-1），结合暗通道先验法，可将公式 5-1 等式两边移项变形后同时取最小值运算，将无雾图像 $J_{(x)}$ 的用暗通道先验的结果 5-3 代替，得到公式 5-4。

$$I(x) = J(x)t(x) + A(1 - t(x)) \tag{5-1}$$

其中，$I(x)$ 为原始图像像元值，$J(x)$ 为无雾情况下图像像元值，A 为全球大气光成分（atmospheric light），$t(x)$ 为像元处的大气透过率（transmission from scene to camera）。

大气透射率也可由以下公式计算：

$$t(x) = e^{-\beta d(x)} \tag{5-2}$$

其中β是大气散射的参数，d 是景深。这个公式说明景深会随着距离的增大而呈现出指数型的衰减。因为如果我们得到了这个透射率，我们就可以根据这一规律得到景物的深度。

暗通道先验规则为：

$$J^{dark}(x) = \min_{c\in\{r,g,b\}} \left(\min_{y\in\omega(x)} J^c(y) \right) \approx 0 \tag{5-3}$$

$$\min_{y\in\Omega(x)} \left(\frac{I^c(y)}{A^c} \right) = t(x) \min_{y\in\Omega(x)} \left(\frac{J^c(y)}{A^c} \right) + (1 - t(x)) \tag{5-4}$$

将无雾图像 $\min_{y\in\Omega(x)} \left(\frac{I^c(y)}{A^c} \right)$ 的用暗通道先验的结果代入消去，得到如下公式：

$$t(x) = 1 - \min_{c\in\{r,g,b\}} \left(\min_{y\in\Omega(x)} \left(\frac{I^c(y)}{A^c} \right) \right) \tag{5-5}$$

我们可以根据该公式推算像素值周围的大气透射率，因为大气中或多或少的存在部分

的雾霾颗粒，因此需要对公式 5-5 的计算结果进行限制，通用的做法是在减数项前添加系数 ω ，得到公式 5-6，根据文献[4]，一般取 $\omega = 0.95$ 。

$$t(x) = 1 - \omega \min_{c\in\{r,g,b\}} \left(\min_{y\in\Omega(x)} \left(\frac{I^c(y)}{A^c} \right) \right) \tag{5-6}$$

5.5 模型建立

通过公式 5-6 可以推知图像中的大气透射率，在气象上，有利用大气消光系数 σ 估计气象能见度的公式 5-7，而大气消光系数由两部分组成（公式 5-8），分别是散射系数 β 和吸收系数 α ，但是当观察目标与观察者较近时，大气的吸收效果可以忽略，因此，我们假设在摄像头和高速公路的距离内，大气消光系数约等于大气散射系数，推得公式 5-9。

$$MOR = \frac{log(F/F_0)}{-\sigma} = \frac{log\ (0.05)}{-\sigma} \tag{5-7}$$

$$\sigma = \alpha + \beta \tag{5-8}$$

$$MOR = \frac{log\ (0.05)}{-\sigma} = \frac{log\ (0.05)}{-\beta} \tag{5-9}$$

我们将估算大气能见度的问题转化为计算大气散射系数 β ，通过相关文献的搜索[5]，可以利用图像的局部区域的像元特征和距离观测位置的距离推算散射系数（公式 5-10）。

$$\beta = -\frac{\sum_{i\in R(x)} ln\ t(i)}{|\,p_R\,|\,D} \tag{5-10}$$

其中，$R(x)$ 为所选像元区域，p_R 为所选区域的像元数量，D 为参考点与摄像头的距离。

在公式 5-10 中，我们可以通过公式 5-6 计算图像像元的大气透射率，而 p_R 是选择的 ROI 的像元总数。只需要知道摄像头距离观测位置的实际距离即可计算出该图的气象能见度。但是，由于题目中并未提供详细的数据，而通过透射变换实现像素距离与实际距离的变换过于复杂，我们设计了一种基于公式 5-10 的散射系数估算方法。

因为不确定摄像头距离，我们从图中选择了两个参考点（图 5-10），利用两参考点的距离差值来消除摄像头与参考点距离这个变量，假设我们所选择的参考点的距离差值与两条分隔线之间的距离相等，而一幅图像上的大气散射系数也假设不变，可得

$$\beta = -\frac{\sum_{i\in R_1(x)} ln\ t(i)}{|\,p_{R_1}\,|\,D_1} \tag{5-11}$$

$$\beta = -\frac{\sum_{i\in R_2(x)} ln\ t(i)}{|p_{R_2}|\, D_2} \tag{5-12}$$

此时两式移项相减，可得：

$$\beta\ (D_2 - D_1) = -\frac{\sum_{i\in R_2(x)} \ln t(i)}{|p_{R_2}|} + \frac{\sum_{i\in R_1(x)} ln\ t(i)}{|p_{R_1}|} \tag{5-13}$$

$$D_2 - D_1 = d \tag{5-14}$$

$$\beta = -\frac{\sum_{i\in R_2(x)} \ln t(i)}{|p_{R_2}|\,(D_2 - D_1)} + \frac{\sum_{i\in R_1(x)} ln\ t(i)}{|p_{R_1}|\,(D_2 - D_1)} \tag{5-15}$$

在题目提供的图像中，高速公路的车道线的位置是固定不变的，而根据查阅的资料，车道分隔线的线长为 6 米，相隔 9 米出现。由公式 5-10 可得，大气散射系数如公式 5-15 所示，而能见度公式为 5-16。

$$MOR = \frac{log\ (0.05)}{-\beta} = \frac{log\ (0.05)}{\frac{\sum_{i\in R_2(x)} \ln t(i)}{|p_{R_2}|\,(D_2 - D_1)} - \frac{\sum_{i\in R_1(x)} ln\ t(i)}{|p_{R_1}|\,(D_2 - D_1)}} \tag{5-16}$$

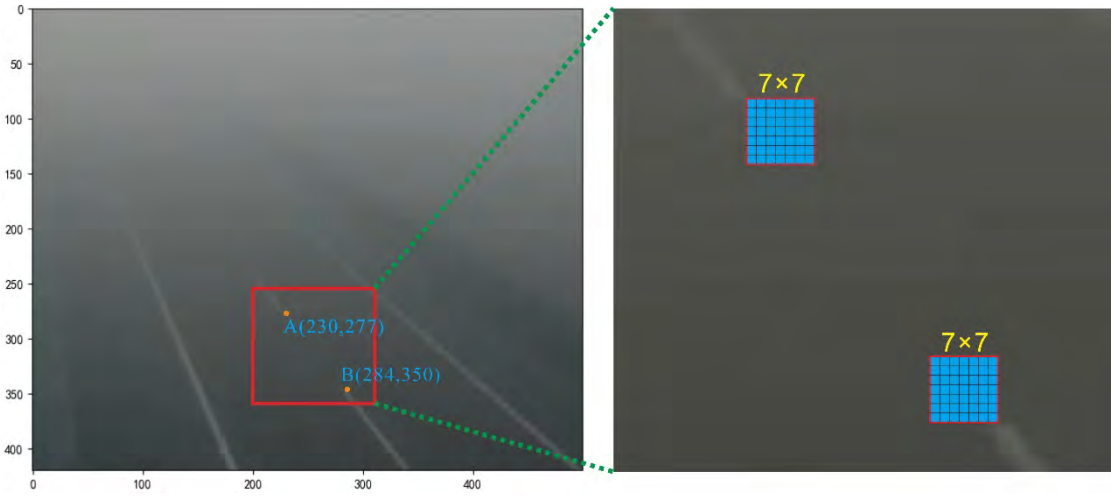

图 5-9 车道分隔线间隔

5.7 模型实现与评估

模型的实现如流程图 5-10 所示，可分为图像裁剪、暗通道提取、大气光照计算、参考点邻域像元提取、大气透射率计算、能见度计算几个关键步骤。

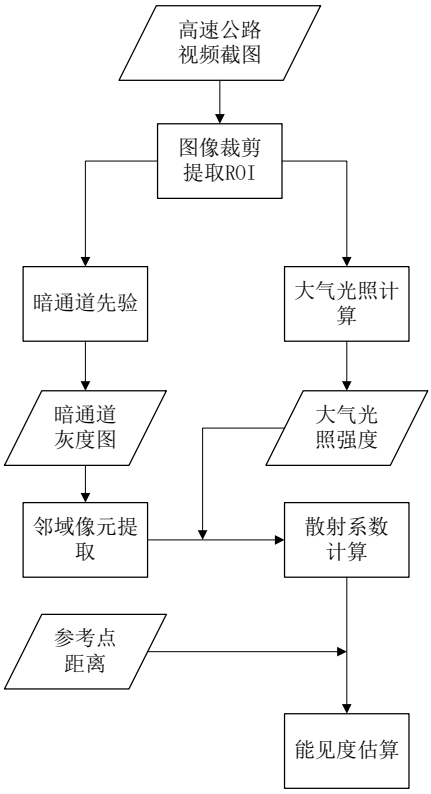

图 5-10 基于视频截图的能见度估算技术路线图

在实际操作的过程中，参考点选择在前后两条道路车道分隔线的间隔处，采用 7 ×7 的窗口提取邻域像元，计算大气透射率。在计算过程中，需要获得大气光照强度，一般选择图像直方图中亮度值最高的 0.1%的像元亮度平均值代替，首先需要对每一幅进行裁剪，以避免白色的路段位置信息水印对大气光照强度的干扰。裁剪出需要的 ROI 后，我们再进行直方图计算，得出每幅图像的大气光照强度，用于后续操作。

因为提取了参考点的邻域信息，我们对于两者之间的距离进行假设，忽略提取邻域像元的误差，将两个参考点的距离等同于两条车道线之间的距离，即相差 9 米，计算得到最后的气象能见度 MOR。

经过对 100 幅影像的计算，我们得到了能见度随时间变化的曲线（图 5-11），根据我们的预测，在此时段内，高速公路该路段的能见度范围在 105 米到 140 米的范围内变动，符合实际的情况，变化趋势也与目测结果相吻合。

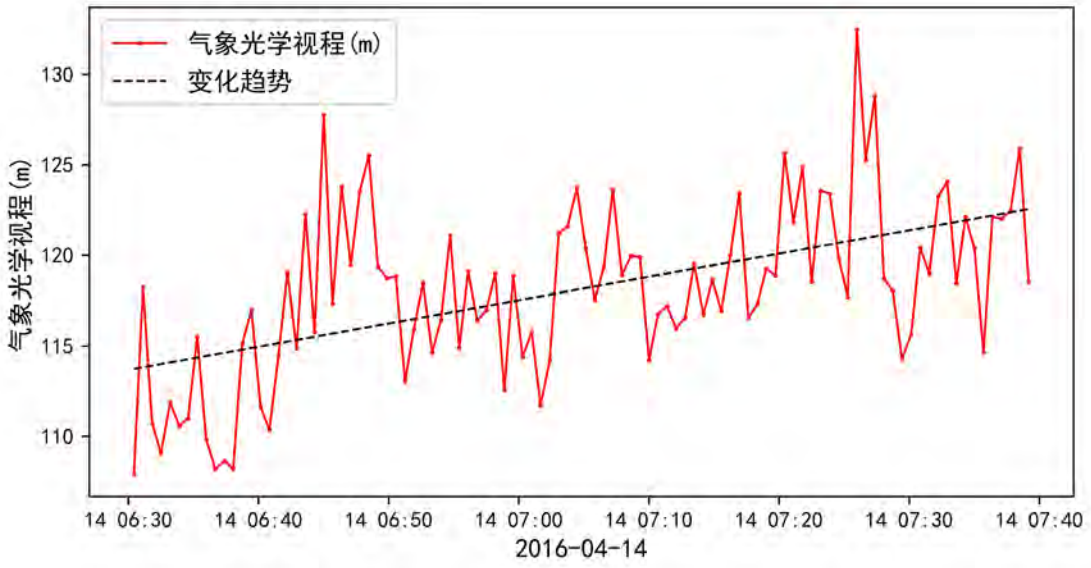

图 5-11 能见度随时间变化曲线

5.6 模型优缺点评价

我们提出的基于视频截图的能见度估算模型通过不同景深的像素对比度，建立基于大气辐射传输的能见度估计模型，其充分利用了图像中的固有信息，不需要依靠相机的高度、角度和参数，是一种计算简单、方便高效的估计方法。该模型的建立基础具有明确的物理意义，并且能够定量得反演能见度的变化，在现实中有较高的参考价值。模拟结果显示高速能见度处于波动上升的趋势，但是该模型也有一定的不足之处，其简化了实际距离计算中的复杂关系，并在邻域像元的提取中引入了一定的人为误差，还需要进一步完善和提升。

六、问题四的建模与求解

6.1 问题分析

问题三中得到了高速公路该路段的能见度随时间变化的趋势，题目要求从现有的 100 个数据出发，建立模型对大雾的发展变化进行预测，并估算能见度达到 150 米时需要的时间。对于 100 个相邻数据间隔约 40 秒的能见度数据，其数据的样本量过少且时间跨度短，对于未来的趋势变化预测具有很大的挑战。考虑到大雾具有其本身的发展演变规律（起雾-能见度降低-维持低能见度-能见度上升-散雾），而这 100 组数据难以体现大雾的变化趋势。因此，本题的重点不在于选用什么模型，而在于对时间序列的分析和探索，通过变化趋势、自相关、平稳性等方面进行分析，利用单位根检验（Dickey-Fuller test）、自相关（Auto Correlation Function , ACF）和偏自相关（Partial Auto Correlation Function, PACF）等指数来反映时间序列的特征。由于样本数据有限，模型难以准确地描述未来变化，但考虑到能见度的变化具有明显的趋势性，随着时间的推移在不断的提高，初步判断大雾处于逐步消散的阶段。因此，采用了二次指数平滑法预测大雾消散过程的发展变化，预计 9 时 05 分前后能见度达到 150 米以上。

6.2 平稳性检验

时间数据的稳定性可以通过 ADF（Augmented Dickey-Fuller Test）单位根检验进行判断，即在一定置信水平下，假设时序数据非稳定，则序列具有单位根。其优点在于它透过纳入一阶向下差分项，排除了自相关的影响。因此对于一个平稳的时序数据，就需要在给定的置信水平上显著，拒绝原假设。对能见度时间序列的单位根检验显示，P 值低于 0.05，所以可以拒绝原假设，说明数据是平稳的。

表 6-1 能见度时间序列的 ADF 单位根检验统计量

英文名称	中文含义	值
Test Statistic	代表检验统计量	-3.270392
p-value	代表 p 值检验的概率	0.016255
Number of Observations Used	样本数量	98.000000
Critical Value (1%)	显著性水平为 1%的临界值	-3.498910
Critical Value (5%)	显著性水平为 5%的临界值	-2.891516
Critical Value (10%)	显著性水平为 10%的临界值	-2.582760

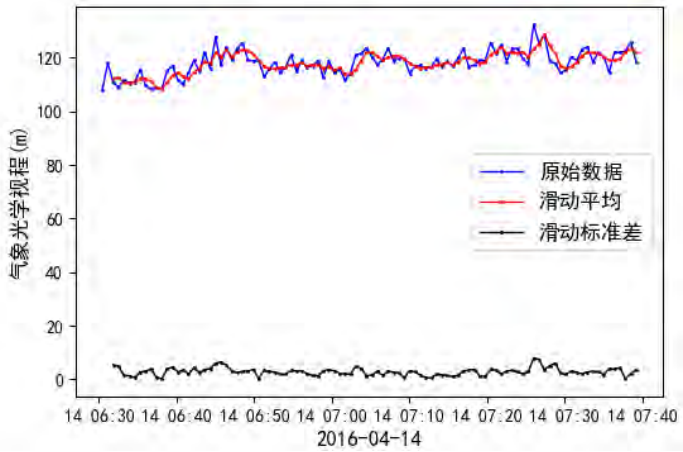

图 6-1 能见度时序的滑动平均和滑动标准差

6.3 自相关（ACF）与偏自相关（PACF）

自相关函数（决定 q 值）反映了同一序列在不同时序的取值之间的相关性。x(t)同时还会受到中间 k-1 个随机变量 $x_{t-1},x_{t-2},\cdots,x_{t-k+1}$ 的影响而这 k-1 个随机变量又都和 x(t-k)具有相关关系，所 以自相关系数 p(k)里实际掺杂了其他变量对 x(t)与 x(t-k)的影响。偏自相关函数（决定 p 值），剔除了中间 k-1 个随机变量$x_{t-1},x_{t-2},\cdots,x_{t-k+1}$ 的干扰之后 x(t-k)对 x(t)影响的相关程度。

$$ACF(k)=\frac{\mathrm{Cov}(y_t,y_{t-k})}{\mathrm{Var}(y_t)} \tag{6-1}$$

$$PACF(k)=\begin{cases}\mathrm{cor}\,(x_1,x_0)=r_1 & \text{if } k=1\\ \mathrm{cor}\left(x_k-x_k^{k-1},x_0-x_0^{k-1}\right) & \text{if } k\geq 2\end{cases} \tag{6-2}$$

图 6-2 展示了能见度的变化时序、直方图、自相关和偏自相关，从直方图中可以看出，此时段的大气能见度集中在 115 至 120 米的区间内。自相关和偏相关可用来分析拖尾和截尾（拖尾是指序列以指数率单调递减或震荡衰减，截尾指序列从某个时点变得非常小）。结果展示，自相关系数是 7 阶拖尾，偏自相关系数则没有拖尾和截尾，因此该序列不适合用于自回归（AR）、移动平均（MA）、自回归滑动平均模型（ARMA）等模型。

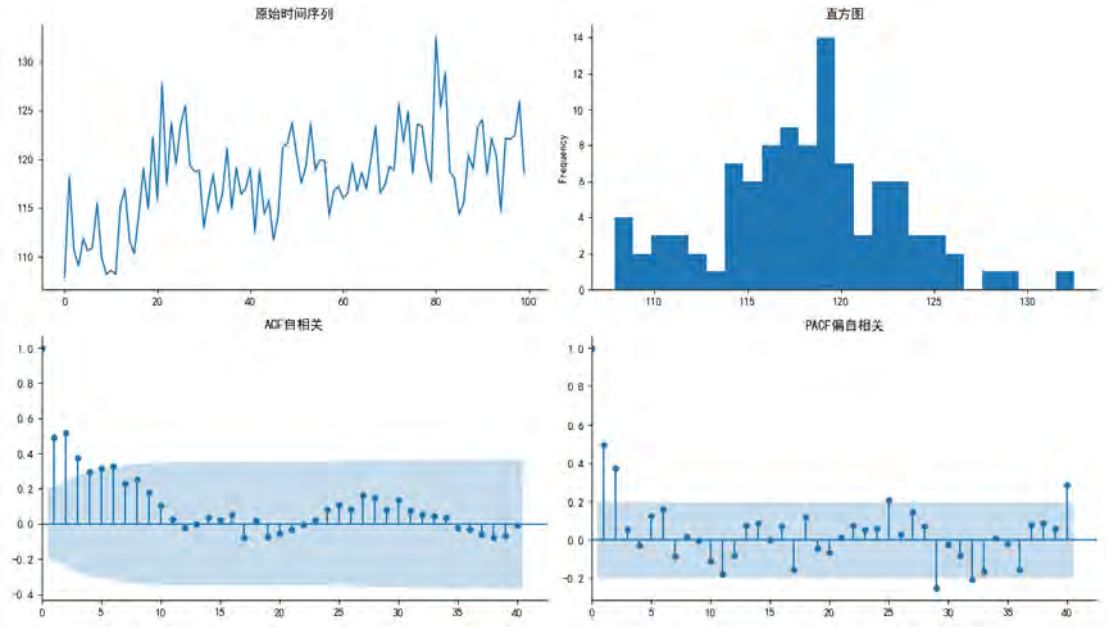

图 6-2 能见度的变化时序、直方图、自相关和偏自相关

6.4 趋势检验

Man-Kendall 趋势检验法是一种非参数检验方法，假设有一个时间序列$X=\{x_1,x_2,\cdots,x_n\}$，统计值 S 的计算过程为

$$sgn\,(x_j - x_i) = \begin{cases} 1 & x_j > x_i \\ 0 & x_j = x_i \\ -1 & x_j < x_i \end{cases} \tag{6-3}$$

$$S = \sum_{j=i+1}^{n} \sum_{j=i+1}^{n} sgn\,(x_j - x_i) \tag{6-4}$$

式中，sgn () 为符号函数；Mann-Kendall 统计量 Z 值计算式为

$$Z = \begin{cases} \dfrac{S-1}{\sqrt{n(n-1)(2n+5)/18}} & S > 0 \\ 0 & S = 0 \\ \dfrac{S+1}{\sqrt{n(n-1)(2n+5)/18}} & S < 0 \end{cases} \tag{6-5}$$

式中，Z>0 表示序列呈上升趋势，Z<0 表示序列呈下降趋势，Z＝$Z_{1-\alpha/2}$ 的绝对值越大，说明该时间序列的变化趋势越显著。给定显著性水平 α=0.05，经查正态分布表可知，对应的置信度水平 1.96；给定显著性水平 α=0.01，对应的置信度水平 2.58。|Z|值大于临界值 1.96 时，上升或下降变化趋势在统计学上为显著；|Z|值大于临界值 2.58 时，变化趋势为极显著；否则，变化趋势不显著。

MK 趋势检验结果表明：该能见度序列具有显著的上升趋势，Z 值为 5.39，通过了 0.01 的显著性检验。

6.5 指数平滑法预测

指数平滑法（Exponential Smoothing, ES）实际上是一种特殊的加权移动平均法。其特点是：（1）进一步加强了观察期近期观察值对预测值的作用，对不同时间的观察值所赋予的权数不等，使预测值能够迅速反映实际的变化。权数之间按等比级数减少，此级数之首项为平滑常数 a,公比为(1- a)；（2）指数平滑法对于观察值所赋予的权数有伸缩性，可以取不同的 a 值以改变权数的变化速率。因此，运用指数平滑法，可以选择不同的 a 值来调节时间序列观察值的即趋势变化的平稳程度。

指数平滑法有几种不同形式：一次指数平滑法针对没有趋势和季节性的序列，二次指数平滑法针对有趋势但没有季节性的序列，三次指数平滑法针对有趋势也有季节性的序列。

Holt 扩展了简单的指数平滑，方法包括预测方程和两个平滑方程（一个用于水平，一个用于趋势），公式如下：

预测方程：$\hat{y}_{t+h|t}\& = \ell_t + hb_t$

水平方程：$\ell_t\& = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$

趋势方程：$b_t\& = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$

由于 100 个能见度数据显示，能见度数据有一定的波动规律，该方法可以在较少数据

的情况下预测较长时间的结果，正好符合题目的要求。在问题 3 求解的最后，得到了能见度随时间变化的曲线（如图 5-6），从图上的趋势线中可以看到在该时段内，能见度呈整体上升趋势，说明雾气是呈消散的趋势，因此我们可以预测该路段的能见度最终会达到 150 米以上，即雾气消散。

图 6-3 展示了使用二次方法得到的能见度预测结果，可以看到结果拟合的曲线也是呈现上升的趋势，且增长速度明显加快，说明该路段的雾气将逐渐消散，红色水平参考线来确定能见度等于 150 米的情况，当能见度的值震荡上升超过红色线且不再与其相交时，对应的时间点为 9 点 05 分，说明此时雾已消散。

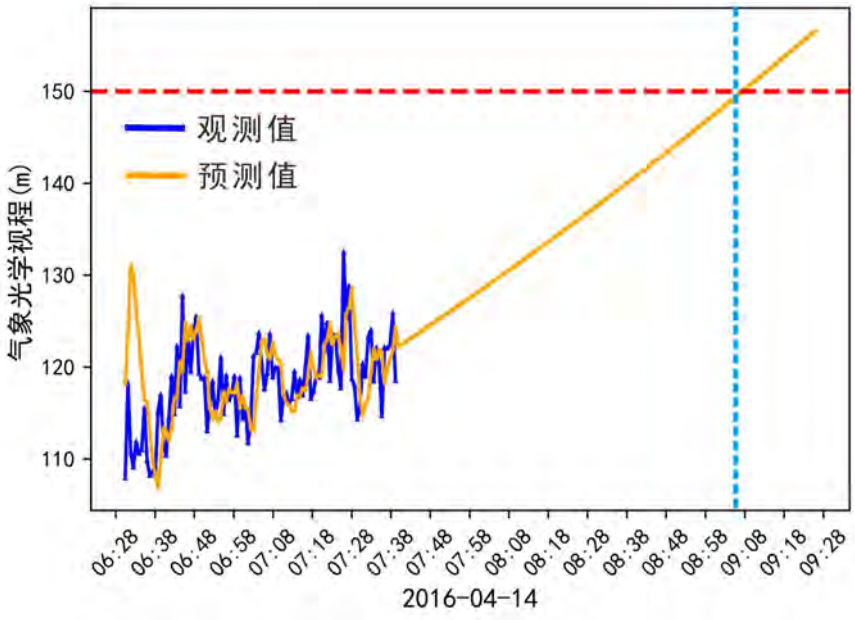

图 6-3 二次指数平滑预测结果

七、参考文献

[1] Godard C, Mac Aodha O, Firman M, et al. Digging into self-supervised monocular depth estimation[C]//Proceedings of the IEEE international conference on computer vision. 2019: 3828-3838.

[2] 段立春,刘超,钟玮,陈丽清,蒋慕蓉. 一种利用大气散射模型实现图像去雾的方法 A Method of Image Dehazing Using Atmospheric Scattering Model[J]. 图像与信号处理, 06(02): 78-88, 2017.

[3] Lee S, Yun S, Nam J H, et al. A review on dark channel prior based image dehazing algorithms[J]. EURASIP Journal on Image and Video Processing, 2016(1): 4, 2016.

[4] He K, Sun J, Tang X. Single image haze removal using dark channel prior[J]. IEEE transactions on pattern analysis and machine intelligence, 33(12): 2341-2353, 2010.

[5] 周凯. 基于道路监控视频的雾霾能见度检测方法研究[D].南京邮电大学,2017.

八、附录

8.1 基于 SIFT 特征的深度学习.py

```
# -*- coding: utf-8 -*-
import cv2
import pandas as pd
import numpy as np
import os
from sklearn import svm

df = pd.read_excel('data.xlsx')
y = df['MOR_1A']

X = []
filenames = sorted((fn for fn in os.listdir('./airport5') if fn.endswith('.jpg')))
filenames.sort(key=lambda x: int(x[0:-4]))

sift = cv2.xfeatures2d.SIFT_create()

for file in filenames[0:1]:
    img = cv2.imread('airport5/' + file)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    kp = sift.detect(gray, None)
    # 显示关键点
    ret = cv2.drawKeypoints(gray, kp, img)
    cv2.imshow('ret', ret)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

for file in filenames:
    img = cv2.imread('airport5/' + file)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # 找出关键点并绘图
    kp, des = sift.compute(gray, kp)
    des = des.reshape((-1,))

    # 使用关键点找出 SIFT 特征向量
    X.append(list(des))

X = pd.DataFrame(X)

# 划分训练集和测试集
```

```
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.datasets import boston_housing
from keras.layers import Dense, Dropout
from keras.utils import multi_gpu_model
from keras import regularizers  # 正则化
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

x_train, x_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3, stratify=y,
                                                    random_state=2020)


model = Sequential()  # 初始化，很重要！
model.add(Dense(units=10,  # 输出大小
                activation='relu',  # 激励函数
                input_shape=(x_train.shape[1],)  # 输入大小，也就是列的大小
                )
          )

model.add(Dropout(0.2))  # 丢弃神经元链接概率
model.add(Dense(units=15,
                kernel_regularizer=regularizers.l2(0.01),  # 施加在权重上的正则项
                activity_regularizer=regularizers.l1(0.01),  # 施加在输出上的正则项
                activation='relu',  # 激励函数
                bias_regularizer=keras.regularizers.l1_l2(0.01)  # 施加在偏置向量上的正则项
                )
          )
model.add(Dense(units=1,
                activation='linear'  # 线性激励函数 回归一般在输出层用这个激励函数
                )
          )
print(model.summary())  # 打印网络层次结构

model.compile(loss='mse',  # 损失均方误差
              optimizer='adam',  # 优化器
              )

```

```
history = model.fit(x_train, y_train,
                    epochs=200,  # 迭代次数
                    batch_size=200,  # 每次用来梯度下降的批处理数据大小
                    verbose=2,  # verbose：日志冗长度，int：冗长度，0：不输出训练过程，1：输出训练进度，2：输出每一个 epoch
                    validation_data=(x_test, y_test)  # 验证集
                    )
fig = plt.figure(figsize=(8, 4))
ax = fig.add_subplot(111)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.ylabel('损失函数', fontsize=12)
plt.xlabel('迭代次数', fontsize=12)
plt.legend(['训练集', '测试集'], loc='upper right', fontsize=12)
plt.savefig('损失曲线.jpg', dpi=600, bbox_inches='tight')
plt.show()

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

# 散点图
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(211)
predict_y = model.predict(x_train)
RMSE = np.sqrt(mean_squared_error(y_train, predict_y))
MAE = mean_absolute_error(y_train, predict_y)
R2 = r2_score(y_train, predict_y)
plt.scatter(y_train, predict_y, c='red', s=5)
plt.xlabel('[训练集]气象光学视程(m)观测值', fontsize=14)
plt.ylabel('[训练集]气象光学视程(m)模拟值', fontsize=14)
plt.text(0, 900, 'RMSE={},\nMAE={},\n$R^2$={}'.format(int(RMSE), int(MAE), round(R2, 2)), fontsize=14)

ax = fig.add_subplot(212)
predict_y = model.predict(x_test)
RMSE = np.sqrt(mean_squared_error(y_test, predict_y))
MAE = mean_absolute_error(y_test, predict_y)
R2 = r2_score(y_test, predict_y)
plt.scatter(y_test, predict_y, c='blue', s=5)
plt.xlabel('[测试集]气象光学视程(m)观测值', fontsize=14)
plt.ylabel('[测试集]气象光学视程(m)模拟值', fontsize=14)
plt.text(0, 900, 'RMSE={},\nMAE={},\n$R^2$={}'.format(int(RMSE), int(MAE), round(R2, 2)), fontsize=14)

plt.savefig('模拟结果.jpg', dpi=600, bbox_inches='tight')
```

```
plt.savefig('模拟结果.pdf', bbox_inches='tight')
plt.show()

```

8.2 机场数据的预处理.py

```
import cv2
import os
import numpy as np

filenames=sorted((fn for fn in os.listdir('./airport1') if fn.endswith('.jpg')))
outPutDirName='airport2/'
if not os.path.exists(outPutDirName):
    os.makedirs(outPutDirName)

# 非局部平均法去噪声(airport2)
for file in filenames:
    img = cv2.imread('airport1/'+file)
    dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21)
    b1, g1, r1 = cv2.split(dst)
    img_denoise = cv2.merge([r1, g1, b1])
    cv2.imwrite('airport2/'+file,img_denoise)
    print(file)

filenames=sorted((fn for fn in os.listdir('./airport2') if fn.endswith('.jpg')))
outPutDirName='airport3/'
if not os.path.exists(outPutDirName):
    os.makedirs(outPutDirName)


# CLAHE 有限对比适应性直方图均衡化(airport3)
for file in filenames:
    img = cv2.imread('airport2/'+file)
    b, g, r = cv2.split(img)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    cl1_b = clahe.apply(b)
    cl1_g = clahe.apply(g)
    cl1_r = clahe.apply(r)
    cl1 = cv2.merge([cl1_r, cl1_g, cl1_b])

    cv2.imwrite('airport3/'+file,cl1)
    print(file)

```

```
40.
41. # 提高对比度
42. filenames=sorted((fn for fn in os.listdir('./airport3') if fn.endswith('.jpg')))
43. outPutDirName='airport4/'
44. if not os.path.exists(outPutDirName):
45.     os.makedirs(outPutDirName)
46.
47. def Contrast_and_Brightness(alpha,beta,img):
48.     """使用公式 f(x)=α.g(x)+β"""
49.     #α 调节对比度，β 调节亮度
50.     blank = np.zeros(img.shape,img.dtype)  #创建图片类型的零矩阵
51.     dst = cv2.addWeighted(img,alpha,blank,1-alpha,beta)  #图像混合加权
52.     return dst
53.
54. for file in filenames:
55.     img = cv2.imread('airport3/'+file)
56.     img_new = Contrast_and_Brightness(1.5,15,img)
57.     cv2.imwrite('airport4/'+file,img_new)
58.     print(file)
59.
60.
61. # mask
62. filenames = sorted((fn for fn in os.listdir('./airport4') if fn.endswith('.jpg')))
63. outPutDirName = 'airport5/'
64. if not os.path.exists(outPutDirName):
65.     os.makedirs(outPutDirName)
66.
67. for file in filenames:
68.     img = cv2.imread('airport4/' + file)
69.     img[50:105, 35:670, :] = np.nan
70.     img[590:655, 890:1135, :] = np.nan
71.     cv2.imwrite('airport5/' + file, img)
72.     print(file)
```

8.3 引导滤波.py

```
1.  import numpy as np
2.  import cv2
3.  import os
4.
5.
6.  def zmMinFilterGray(src, r=5):
7.      '''''最小值滤波，r 是滤波器半径'''
8.      '''''if r <= 0:
9.          return src
```

```
    h, w = src.shape[:2]
    I = src
    res = np.minimum(I  , I[[0]+range(h-1)  , :])
    res = np.minimum(res, I[range(1,h)+[h-1], :])
    I = res
    res = np.minimum(I  , I[:, [0]+range(w-1)])
    res = np.minimum(res, I[:, range(1,w)+[w-1]])
    return zmMinFilterGray(res, r-1)'''
    return cv2.erode(src, np.ones((2 * r + 1, 2 * r + 1)))  # 使用 opencv 的 erode 函数更高效


def guidedfilter(I, p, r, eps):
    height, width = I.shape
    m_I = cv2.boxFilter(I, -1, (r, r))
    m_p = cv2.boxFilter(p, -1, (r, r))
    m_Ip = cv2.boxFilter(I * p, -1, (r, r))
    cov_Ip = m_Ip - m_I * m_p

    m_II = cv2.boxFilter(I * I, -1, (r, r))
    var_I = m_II - m_I * m_I

    a = cov_Ip / (var_I + eps)
    b = m_p - a * m_I

    m_a = cv2.boxFilter(a, -1, (r, r))
    m_b = cv2.boxFilter(b, -1, (r, r))
    return m_a * I + m_b


def getV1(m, r, eps, w, maxV1):  # 输入 rgb 图像，值范围[0,1]
    '''''计算大气遮罩图像 V1 和光照值 A, V1 = 1-t/A'''
    V1 = np.min(m, 2)  # 得到暗通道图像
    V1 = guidedfilter(V1, zmMinFilterGray(V1, 7), r, eps)  # 使用引导滤波优化
    bins = 2000
    ht = np.histogram(V1, bins)  # 计算大气光照 A
    d = np.cumsum(ht[0]) / float(V1.size)
    for lmax in range(bins - 1, 0, -1):
        if d[lmax] <= 0.999:
            break
    A = np.mean(m, 2)[V1 >= ht[1][lmax]].max()

    V1 = np.minimum(V1 * w, maxV1)  # 对值范围进行限制

```

```
    return V1, A


def deHaze(m, r=81, eps=0.001, w=1, maxV1=0.80, bGamma=False):
    Y = np.zeros(m.shape)
    V1, A = getV1(m, r, eps, w, maxV1)  # 得到遮罩图像和大气光照
    print(A)
    for k in range(3):
        Y[:, :, k] = (m[:, :, k] - V1) / (1 - V1 / A)  # 颜色校正
    Y = np.clip(Y, 0, 1)
    if bGamma:
        Y = Y ** (np.log(0.5) / np.log(Y.mean()))  # gamma 校正,默认不进行该操作
    return Y


filenames = sorted((fn for fn in os.listdir('./gaosu3') if fn.endswith('.bmp')))
filenames.sort(key=lambda x: int(x[14:-4]))
outPutDirName = 'gaosu5/'
if not os.path.exists(outPutDirName):
    os.makedirs(outPutDirName)

for file in filenames:
    m = deHaze(cv2.imread('gaosu3/' + file) / 255.0) * 255
    cv2.imwrite('gaosu5/' + file, m)
    print(file)
```

8.4 透视变化.py

```
# -*- coding: utf-8 -*-
import cv2
import numpy as np
from matplotlib import pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

img = cv2.imread('gaosu5/original_frame1.bmp')
rows, cols, ch = img.shape

pts1 = np.float32([[73, 177], [223, 177], [173, 420], [498, 420]])
pts2 = np.float32([[0, 0], [200, 0], [0, 300], [200, 300]])

M = cv2.getPerspectiveTransform(pts1, pts2)
dst = cv2.warpPerspective(img, M, (200, 300))

```

```
# for i in range(len(pts1)-1):
#     cv2.line(img, (pts1[i][0], pts1[i][1]), (pts1[i+1][0], pts1[i+1][1]), color=[0, 255, 0])


plt.figure(figsize=(8, 8))
p1 = plt.subplot(121)
p1.imshow(img)
p1.set_title('原始图像', fontsize=14)

p2 = plt.subplot(122)
p2.imshow(dst)
p2.set_title('透射变换', fontsize=14)
plt.savefig('透射变换.jpg', dpi=600, bbox_inches='tight')
plt.savefig('透射变换.pdf', bbox_inches='tight')
plt.show()
```

8.5 时间序列分析与建模.py

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from matplotlib.dates import datestr2num
import time
import scipy.stats as stats
from statsmodels.tsa.stattools import adfuller
from statsmodels.stats.diagnostic import acorr_ljungbox
import prettytable
from statsmodels.tsa.arima_model import ARMA

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


def cox_staut(list_c):
    lst = list_c.copy()
    n0 = len(lst)
    if n0 % 2 == 1:
        del lst[int((n0 - 1) / 2)]
    c = int(len(lst) / 2)
    pos = neg = 0
    for i in range(c):
        diff = lst[i + c] - lst[i]
        if diff > 0:
```

```
            pos += 1
        elif diff < 0:
            neg += 1
        else:
            continue
    n1 = pos + neg
    k = min(pos, neg)
    p = 2 * stats.binom.cdf(k, n1, 0.5)
    print('fall:%i  rise:%i  p-value:%f' % (neg, pos, p))


df = pd.read_excel('vis.xlsx')
timelist = [str(i)[0:11] + str(j) for i, j in zip(df['date'], df['time'])]
# df = df.set_index('time')
df['index'] = timelist
df['index'] = pd.to_datetime(df['index'])
df = df.set_index('index')

mor1 = df['MOR_1']
mor2 = df['MOR_2']

# 绘制 MOR 变化曲线(最小二乘法)
fig = plt.figure(figsize=(8, 4), facecolor='white')
ax = fig.add_subplot(111)
ax.plot_date(datestr2num(timelist), mor1, 'o-', color='red',
             linewidth=1, markersize=1, label='气象光学视程(m)')

z1 = np.polyfit(datestr2num(timelist), mor1, 1)
p1 = np.poly1d(z1)
plt.plot(datestr2num(timelist), p1(datestr2num(timelist)),
         "--", color='k', linewidth=1, label='变化趋势')
plt.legend(fontsize=12)
plt.xlabel('2016-04-14', fontsize=12)
plt.ylabel('气象光学视程(m)', fontsize=12)
plt.savefig('变化曲线趋势.jpg', dpi=600, bbox_inches='tight')
plt.show()

dftest = adfuller(mor1, autolag='AIC')

# Multiplicative Decomposition
result_mul = sm.tsa.seasonal_decompose(df['MOR_1'], model='multiplicative', extrapolate
_trend='freq', freq=30)

# Plot
```

```
plt.rcParams.update({'figure.figsize': (10, 10)})
result_mul.plot().suptitle('Multiplicative Decompose', fontsize=10)
plt.show()

# Extract the Components ----# Actual Values = Product of (Seasonal * Trend * Resid)
df_reconstructed = pd.concat([result_mul.seasonal, result_mul.trend, result_mul.resid,
result_mul.observed], axis=1)
df_reconstructed.columns = ['seas', 'trend', 'resid', 'actual_values']
df_reconstructed.head()

sm.tsa.seasonal_decompose(mor1).plot()
result = sm.tsa.stattools.adfuller(mor1)
plt.show()


# 趋势检验
cox_staut(list(mor1))

# 移动平均图
def draw_trend(timeseries, size):
    f = plt.figure(figsize=(6, 4),facecolor='white')
    ax = f.add_subplot(111)
    ax.plot_date(datestr2num(timelist), timeseries, 'o-', color = 'blue',
             linewidth = 1, markersize=1, label = '原始数据')
    # 对 size 个数据进行移动平均
    rol_mean = timeseries.rolling(window=size).mean()
    # 对 size 个数据移动平均的方差
    rol_std = timeseries.rolling(window=size).std()
    #timeseries.plot(color='blue', label='原始数据')
    #rol_mean.plot(color='red', label='滑动平均')
    #rol_std.plot(color='black', label='滑动标准差')
     ax.plot_date(datestr2num(timelist), rol_mean, 'o-', color = 'red',
              linewidth = 1, markersize=1, label = '滑动平均')
     ax.plot_date(datestr2num(timelist), rol_std, 'o-', color = 'black',
              linewidth = 1, markersize=1, label = '滑动标准差')
     plt.legend(loc='best',fontsize=12)
     plt.xlabel('2016-04-14',fontsize=12)
     plt.ylabel('气象光学视程(m)',fontsize=12)
     #plt.title('滑动平均与标准差',fontsize=12)
     plt.savefig('滑动平均与标准差.jpg', dpi=600, bbox_inches ='tight')
     plt.show()

draw_trend(mor1, 3)

```

```
#Dickey-Fuller test:
def teststationarity(ts):
    dftest = adfuller(ts)
    # 对上述函数求得的值进行语义描述
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    return dfoutput

teststationarity(mor1)

# 自相关和偏相关图，默认阶数为31阶
def draw_acf_pacf(ts, lags=30):
    f = plt.figure(figsize=(9, 7),facecolor='white')
    ax1 = f.add_subplot(211)
    plot_acf(mor1, lags=30, ax=ax1, title='自相关性')
    ax2 = f.add_subplot(212)
    plot_pacf(mor1, lags=30, ax=ax2, title='偏相关性')
    plt.savefig('自相关性和偏相关.jpg', dpi=600, bbox_inches ='tight')
    plt.show()
draw_acf_pacf(mor1, lags=30)
```

8.6 指数平滑法.py

```
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.dates import datestr2num, DateFormatter
from statsmodels.tsa.holtwinters import ExponentialSmoothing, Holt
import matplotlib.dates as mdates

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

df = pd.read_excel('vis.xlsx')
timelist = [str(i)[0:11] + str(j) for i, j in zip(df['date'], df['time'])]

date_new = pd.date_range(start='2016-04-14 7:39:11', periods=161, freq='40S', closed='right')
timelist2 = timelist
for i in date_new:
    timelist2.append(str(i))

mor1 = df['MOR_1']

```

```
fig = plt.figure(figsize=(6, 4))
ax = fig.add_subplot(111)

timelist = [str(i)[0:11] + str(j) for i, j in zip(df['date'], df['time'])]

# rol_mean = mor1.rolling(window=2).mean()
fit1 = ExponentialSmoothing(mor1, seasonal_periods=20, trend='add', seasonal='mul').fit(use_boxcox=True)
l0, = plt.plot_date(datestr2num(timelist), mor1,
                    marker='^', c='b', linestyle='solid', markersize=1)
# l1, = plt.plot_date(datestr2num(timelist2),list(fit1.fittedvalues) + list(fit1.forecast(180)),
#                     marker='o', c='orange', linestyle='solid', markersize=1)

fit2 = Holt(mor1, exponential=True).fit(smoothing_level=0.4, smoothing_slope=0.3, optimized=False)
l2, = plt.plot_date(datestr2num(timelist2), list(fit2.fittedvalues) + list(fit2.forecast(160)),
                    marker='o', c='orange', linestyle='solid', markersize=1)

ax.xaxis.set_major_locator(mdates.MinuteLocator(interval=10))
ax.xaxis.set_major_formatter(DateFormatter('%H:%M'))
plt.xticks(rotation=45)
# plt.ylim(100,160)

plt.xlabel('2016-04-14', fontsize=12)
plt.ylabel('气象光学视程(m)', fontsize=12)
plt.axhline(y=150, c='r', ls='--', lw=2)
plt.savefig('能见度变化预测结果.jpg', dpi=600, bbox_inches='tight')
plt.savefig('能见度变化预测结果.pdf', bbox_inches='tight')
plt.show()
```

8.7 大气光照强度计算.py

```
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 20 00:18:58 2020

@author: JoyceSong
"""

import numpy as np
import cv2
import os
import pandas as pd
```

```
12.
13.
14. def zmMinFilterGray(src, r=5):
15.     '''''最小值滤波，r 是滤波器半径'''
16.     '''''if r <= 0:
17.         return src
18.     h, w = src.shape[:2]
19.     I = src
20.     res = np.minimum(I  , I[[0]+range(h-1)  , :])
21.     res = np.minimum(res, I[range(1,h)+[h-1], :])
22.     I = res
23.     res = np.minimum(I  , I[:, [0]+range(w-1)])
24.     res = np.minimum(res, I[:, range(1,w)+[w-1]])
25.     return zmMinFilterGray(res, r-1)'''
26.     return cv2.erode(src, np.ones((2*r+1, 2*r+1)))
27.
28. def guidedfilter(I, p, r, eps):
29.     height, width = I.shape
30.     m_I = cv2.boxFilter(I, -1, (r,r))
31.     m_p = cv2.boxFilter(p, -1, (r,r))
32.     m_Ip = cv2.boxFilter(I*p, -1, (r,r))
33.     cov_Ip = m_Ip-m_I*m_p
34.
35.     m_II = cv2.boxFilter(I*I, -1, (r,r))
36.     var_I = m_II-m_I*m_I
37.
38.     a = cov_Ip/(var_I+eps)
39.     b = m_p-a*m_I
40.
41.     m_a = cv2.boxFilter(a, -1, (r,r))
42.     m_b = cv2.boxFilter(b, -1, (r,r))
43.     return m_a*I+m_b
44.
45. def getV1(m, r, eps, w, maxV1):  #输入 rgb 图像，值范围[0,1]
46.     '''''计算大气遮罩图像 V1 和光照值 A, V1 = 1-t/A'''
47.     V1 = np.min(m,2)                                    #得到暗通道图像
48.     V1 = guidedfilter(V1, zmMinFilterGray(V1,7), r, eps)     #使用引导滤波优化
49.     bins = 2000
50.     ht = np.histogram(V1, bins)                         #计算大气光照 A
51.     d = np.cumsum(ht[0])/float(V1.size)
52.     for lmax in range(bins-1, 0, -1):
53.         if d[lmax]<=0.999:
54.             break
55.     A  = np.mean(m,2)[V1>=ht[1][lmax]].max()
```

```
56.
57.     V1 = np.minimum(V1*w, maxV1)                    #对值范围进行限制
58.
59.     return V1,A
60.
61. filenames=sorted((fn for fn in os.listdir('../Extract') if fn.endswith('.bmp')))
62. filenames.sort(key = lambda x: int(x[14:-4]))
63.
64. # 计算得到每一幅图像的大气光照强度
65. A_list = []
66. for file in filenames:
67.     _, A = getV1(cv2.imread('../Extract/'+file)/255.0, r=81, eps=0.001, w=1, maxV1=0.80
    )
68.     A_list.append(A)
69.
70. A_list = pd.DataFrame(A_list)
71. A_list.to_csv('A2.csv', index=False, header=None)
```

8.8 能见度估算代码.py

```
1.  # -*- coding: utf-8 -*-
2.  """
3.  Created on Sat Sep 19 23:48:45 2020
4.
5.  @author: JoyceSong
6.  """
7.
8.  import cv2
9.  import numpy as np
10. from PIL import Image
11. import matplotlib.pyplot as plt
12. import os
13. import pandas as pd
14.
15. # 加载之前算好的大气光照强度
16. a = pd.read_csv('A.csv',header=None)
17.
18. filenames=sorted((fn for fn in os.listdir('../DCP_Mask') if fn.endswith('.bmp')))
19. filenames.sort(key = lambda x: int(x[19:-4]))
20.
21.
22. Vis = []
23.
24. for i, file in enumerate(filenames):
25.
```

```
26.     img_path = os.path.join('../DCP_Mask', file)
27.     img = cv2.imread(img_path)
28.     # plt.imshow(img)
29.
30.     # 300:720, 300:880
31.
32.     # 提取 ROI 的范围
33.
34.     # 1280, 720
35.     # 526, 572
36.     # 500, 545
37.
38.     # 272, 226 POI1 near 277 230 350, 284
39.     # 245, 200 POI2 far 277 230
40.
41.     # POI1 = img[347:350, 281:284, 0]
42.     # POI2 = img[272:275, 226:229, 0]
43.
44.     # POI1 = img[345:350, 279:284, 0]
45.     # POI2 = img[272:277, 226:231, 0]
46.
47.     POI1 = img[345:352, 279:282, 0]
48.     POI2 = img[272:279, 226:233, 0]
49.
50.     # plt.imshow(img)
51.
52.     A = a.iloc[i,0]
53.
54.     POI1 = POI1 / (A * 255)
55.     POI2 = POI1 / (A * 255)
56.
57.     POI1 = 1 - 0.95 * POI1
58.     POI2 = 1 - 0.95 * POI2
59.
60.     POI1 = np.log(POI1)
61.     POI2 = np.log(POI2)
62.
63.     sum_POI1 = POI1.sum()
64.     sum_POI2 = POI2.sum()
65.
66.     # 估算大气散射系数 beta
67.
68.     # beta_d1 = - sum_POI1 / 9
69.     # beta_d2 = - sum_POI2 / 9
```

```
70.
71.     # beta_d1 = - sum_POI1 / 25
72.     # beta_d2 = - sum_POI2 / 25
73.
74.     beta_d1 = - sum_POI1 / 49
75.     beta_d2 = - sum_POI2 / 49
76.
77.     beta = np.abs(beta_d1 - beta_d2) / 9
78.     # beta = np.abs(beta_d1 - beta_d2) / 9
79.
80.     # 能见度估算
81.
82.     visibility = - np.log(0.05) / beta
83.     Vis.append(visibility)
84.
85. Vis = np.array(Vis)
86. plt.plot(Vis)
87. plt.show()
88.
89. Vis = pd.DataFrame(Vis)
90. Vis.to_csv('vis4.csv', header=None)
```