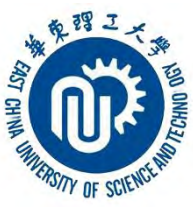



中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生

数学建模竞赛

学　　校	中国石油大学（华东）
参赛队号	20104250114
队员姓名	1.夏文辉
	2.陈昱行
	3.袁晶贤

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 基于视频数据的能见度估计与预测

摘 要：

低能见度灾害综合评价对人们的生活和交通运输管理部门的风险防范有着重要意义。视频数据由于包含了连续而丰富的地面气象数据，因此成为能见度估计与预测的重要手段。本文通过建立多元线性回归方程描述了能见度与地面气象观测数据的关系；利用 VGG16 卷积神经网络深度学习模型来估计能见度并对其精度进行评估；在不依赖观测数据的基础上，只根据视频数据的景深特征建立了能见度估计算法，并绘制了能见度随时间的变化曲线；最后建立模型预测大雾的变化，经过由深及浅、循序渐进的分析，能够准确对视频数据能见度估计与预测，为交通工作者提供技术参考。

针对问题一，使用相关性分析的方法建立了多元线性回归模型，得出能见度与地面观测数据呈线性相关的结论。对描述能见度与地面气象观测数据（温度、湿度和风速等）之间的关系这一问题，首先对给出的机场 AMOS 数据进行预处理，按照时间规律结合温度与湿度数据，将能见度数据和风数据的缺失值补全，增加了数据的完整性。之后将这三项数据按照时间规律合并，并对三项数据中共 12 个指标进行了相关性分析，计算得到相关系数矩阵表。查阅相关资料可知， RVR_1A（跑道视程）和 MOR_1A（气象光学视程）两项指标与能见度相关，前者需要考虑到灯光数据（LIGHT）的影响，后者则与灯光数据（LIGHT）无关。本文共选择温度与湿度三项指标（本站气压（PAINS）、温度（TENP）和相对湿度（RH）、）和风两项指标（2 分钟平均风速（WS2A）和 2 分钟平均垂直风速（CW2A））与两种定义下的能见度进行多元线性回归分析，经过计算得到两个回归方程的系数矩阵 β_V 和 β_M，分别用来描述 RVR_1A 和 MOR_1A 与地面气象观测数据的关系。分析表明，RVR_1A 与 1 分钟平均风速（WS2A）、本站气压（PAINS）、温度（TEMP）和相对湿度（RH）数据呈正相关，与 2 分钟平均垂直风速（CW2A）和灯光数据（LIGHT）呈负相关。对于 MOR_1A，与 1 分钟平均风速（WS2A）、2 分钟平均垂直风速（CW2A）、本站气压（PAINS）、温度（TEMP）和相对湿度（RH）这五项特征指标都呈正相关。最后，在统计分析的基础上，对回归方程系数进行验证与分析评价，本文利用可决系数 R^2 和均方根误差 MSE 对其分析与验证。从结果可以看出两个回归方程模型拟合效果较好，能够准确的描述出能见度与地面观测数据的关系。

针对问题二，本文建立了基于视频数据与能见度估计的 VGG 卷积神经网络深度学习模型，并进行精度评价，可以看出该模型能够准确的估计能见度。首先获得视频帧序列图像，然后根据时间建立图像与能见度数据之间的联系，形成一一对应的关系，之后将视频数据按照 MOR_1A 数据划分能见度阈值，形成四类描述能见度的标签数据作为训练样本。由于视频数据具有特征多，特征连续性强，计算量大的特点。VGG16 卷积神经网络具有传统卷积神经网络可以保留邻域的联系和空间的局部特点，还能够在获得更多图像特征的同时控制参数的个数，提高运算效率。所以本文建立 VGG16 卷积神经网络深度学习模型来评估能见度，并对模型算法做出改进，模型精度跟效率都得到提升,验证集精度达到 86.2%，并选取测试集对模型效果进行评估，精度达到 88.3%，最后利用混淆矩阵对能见度估计效果进行精度分析评价。

针对问题三，本文建立了基于监控视频数据的能见度估计算法，算法首先以截图中的车道线为参考物，利用 Canny 算子检测出车道线，再利用 Hough 变换完成车道线的提取，以此为依据来估算该视频路段车道线的消失点（晴天状况下的最远可视点）及灰度变化拐点，进一步利用视频图像中灰度变化拐点位置估算出能见度大小，最终得到能见度变化曲线。分析数据得知，随着时间的变化，能见度呈上升的趋势。

针对问题四，在问题三能见度数据估计的基础上，分别建立线性回归模型和灰度预测模型对大雾的变化趋势进行预测。两种方法都表明，随着时间的推移大雾慢慢减弱，在三小时左右大雾散去。

关键词：卷积神经网络；能见度估计；相关系数矩阵；景深；灰度预测模型

目录

1　问题重述

1.1　问题背景

根据飞行安全统计，机场低能见度是造成飞行事故的原因之一。机场周围的能见度差会通过减少跑道和滑行道的容量而影响飞行，这会导致航班延误，改道飞往其他机场甚至取消航班。因此低能见度灾害的综合评价对交通运输管理部门的风险防范有着重要的指导作用和现实意义。

激光能见度仪的出现，方便了本文对于能见度的检测。但由于其价格昂贵、对团雾检测度低，探测范围有限，维护成本不足等缺点，使得本文不得不另寻他路。近几年，视频能见度检测的方法备受关注，它能够在一定程度上弥补激光能见度仪的不足。由于现有的视频图像能见度检测方法计算复杂，能够准确的估计能见度比较困难。有些学者在这方面做出了许多工作，但并没有充分运用视频数据的连续信息，所以精度还有待提高。

在大雾发生的情况下，需要准确的得到当前的能见度甚至是能够预测到未来的能见度。因此本研究只关注大雾的演变规律。而视频数据涵盖资料丰富，有着涵盖近地层等气象因素的信息能够准确的描述大雾变化的过程，提高能见度预测精度。

图 1.1.1　合肥机场大雾影响飞机起飞

1.2　需要解决的问题

根据上述背景可知，不同的大雾情况使得对应的能见度不同，而大雾的变化也受诸多因素的影响，使得过程极其复杂。大雾与近地面的气象因素究竟有什么关系呢？视频数据涵盖资料丰富，那视频数据对能见度估计的精度如何？而在没有能见度仪观测数据的情况下，视频数据能否准确估计能见度呢？能见度随时间变化是否有可寻的规律呢？建立简化的模型来估计能见度，探究其中的规律，为交通运输业提供技术支持。

问题一：描述能见度与地面气象观测之间的关系

分析、统计并处理机场 AMOS 观测得到的温度湿度数据、风速数据和能见度数据，建

立合适的模型分析并描述能见度与地面观测数据的关系。

问题二：建立基于视频数据的能见度估计深度学习模型

根据能见度数据与处理得到的机场视频数据，建立两者的深度学习模型，并根据深度学习模型估计能见度，建立深度学习精度评价方法对估计的能见度进行精度评价。

问题三：建立基于视频数据的能见度估计算法，绘制能见度与时间的变化曲线

在仅利用监控视频数据不能利用能见度仪观测数据的情况下，以视频中不同物体的景深为依据，建立视频数据能见度估计算法。另外，根据题目提供的视频来找到能见度随时间的变化规律并绘制变化曲线。

问题四：预测大雾变化趋势

根据问题三绘制的能见度随时间的变化曲线图，建立预测模型，预测大雾的变化趋势。

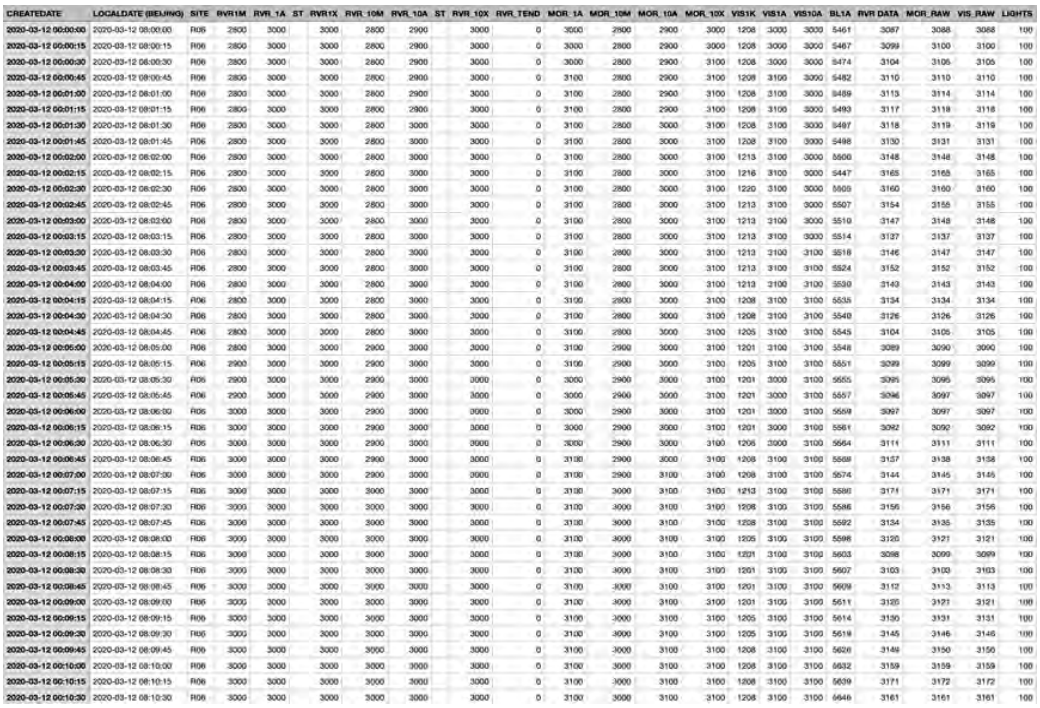
2 数据说明

问题 1 中给出了机场 AMOS 观测数据即地面气象观测数据。包含 2019 年 12 月 15 号与 2020 年 03 月 12 号两天的 AMOS 数据，分别涵盖了温度与湿度数据（PTU）、能见度数据（VIS）和风速的数据（WIND）。其中温度与湿度数据包含了 6 项描述信息分别为本站气压（PAINS（HPA））, 飞机着陆地区最高点气压（QFE 06），修正海平面气压（QNH），温度（TEMP），相对湿度（RH）， 露点温度（DEWPOINT）。每隔一分钟监测一次，共包含 1440 个时间点对应的数据。

	CREATEDATE	LOCALDATE (BEIJING)	SITE	PAINS (HPA)	QNH AERODROME (HPA)	ST	QFE R06 (HPA)	ST.1	QFE R24 (HPA)	ST.2	QFF AERODROME (HPA)	TREND	TENDENCY	TEMP (°C)	RH (%)	DEWPOINT (°C)	TU DATA
0	2020-03-12 00:00:00	2020-03-12 08:00:00	R06	1016.9	1018.44		1016.88		1017.05		1018.47	0.3	3	9.7	85.0	7.32	9.7 85 81F6
1	2020-03-12 00:01:00	2020-03-12 08:01:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	9.8	85.0	7.42	9.8 85 5834
2	2020-03-12 00:02:00	2020-03-12 08:02:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	9.8	84.0	7.24	9.8 84 6B05
3	2020-03-12 00:03:00	2020-03-12 08:03:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.3	3	9.9	84.0	7.34	9.9 84 2CD6
4	2020-03-12 00:04:00	2020-03-12 08:04:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.3	3	9.9	84.0	7.34	9.9 84 2CD6
5	2020-03-12 00:05:00	2020-03-12 08:05:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.4	3	10.0	84.0	7.44	10.0 84 DDA5
6	2020-03-12 00:06:00	2020-03-12 08:06:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.4	3	10.0	83.0	7.27	10.0 83 4432
7	2020-03-12 00:07:00	2020-03-12 08:07:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.4	3	10.1	84.0	7.54	10.1 84 9A76
8	2020-03-12 00:08:00	2020-03-12 08:08:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.4	3	10.2	84.0	7.64	10.2 84 5203
9	2020-03-12 00:09:00	2020-03-12 08:09:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.4	3	10.2	83.0	7.46	10.2 83 CB94
10	2020-03-12 00:10:00	2020-03-12 08:10:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.4	3	10.3	84.0	7.73	10.3 84 15D0
11	2020-03-12 00:11:00	2020-03-12 08:11:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.4	3	10.5	85.0	8.1	10.5 85 A62A
12	2020-03-12 00:12:00	2020-03-12 08:12:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.4	3	10.5	83.0	7.76	10.5 83 0C8C
13	2020-03-12 00:13:00	2020-03-12 08:13:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.4	3	10.5	83.0	7.76	10.5 83 0C8C
14	2020-03-12 00:14:00	2020-03-12 08:14:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.4	3	10.6	82.0	7.68	10.6 82 F7C8
15	2020-03-12 00:15:00	2020-03-12 08:15:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	10.6	82.0	7.68	10.6 82 F7C8
16	2020-03-12 00:16:00	2020-03-12 08:16:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.3	3	10.6	80.0	7.32	10.6 80 91AA
17	2020-03-12 00:17:00	2020-03-12 08:17:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.3	3	10.7	81.0	7.6	10.7 81 E548
18	2020-03-12 00:18:00	2020-03-12 08:18:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.3	3	10.7	80.0	7.42	10.7 80 D679
19	2020-03-12 00:19:00	2020-03-12 08:19:00	R06	1017.1	1018.64		1017.08		1017.25		1018.67	0.3	3	10.7	80.0	7.42	10.7 80 D679
20	2020-03-12 00:20:00	2020-03-12 08:20:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	10.8	80.0	7.51	10.8 80 0FBB
21	2020-03-12 00:21:00	2020-03-12 08:21:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	10.9	80.0	7.61	10.9 80 4868
22	2020-03-12 00:22:00	2020-03-12 08:22:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	10.9	79.0	7.43	10.9 79 DEC1
23	2020-03-12 00:23:00	2020-03-12 08:23:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	11.1	81.0	7.99	11.1 81 0AC6
24	2020-03-12 00:24:00	2020-03-12 08:24:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.3	3	11.2	81.0	8.09	11.2 81 C2B3
25	2020-03-12 00:25:00	2020-03-12 08:25:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.1	3	11.2	79.0	7.72	11.2 79 6726
26	2020-03-12 00:26:00	2020-03-12 08:26:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.1	3	11.3	78.0	7.63	11.3 78 13C9
27	2020-03-12 00:27:00	2020-03-12 08:27:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.1	3	11.2	78.0	7.54	11.2 78 541A
28	2020-03-12 00:28:00	2020-03-12 08:28:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.1	3	11.2	78.0	7.16	11.2 78 7715
29	2020-03-12 00:29:00	2020-03-12 08:29:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.1	3	11.5	79.0	8.01	11.5 79 A033
30	2020-03-12 00:30:00	2020-03-12 08:30:00	R06	1017.0	1018.54		1016.98		1017.15		1018.57	0.1	3	11.5	78.0	7.83	11.5 78 9302

图 2.1 温度与湿度资料信息

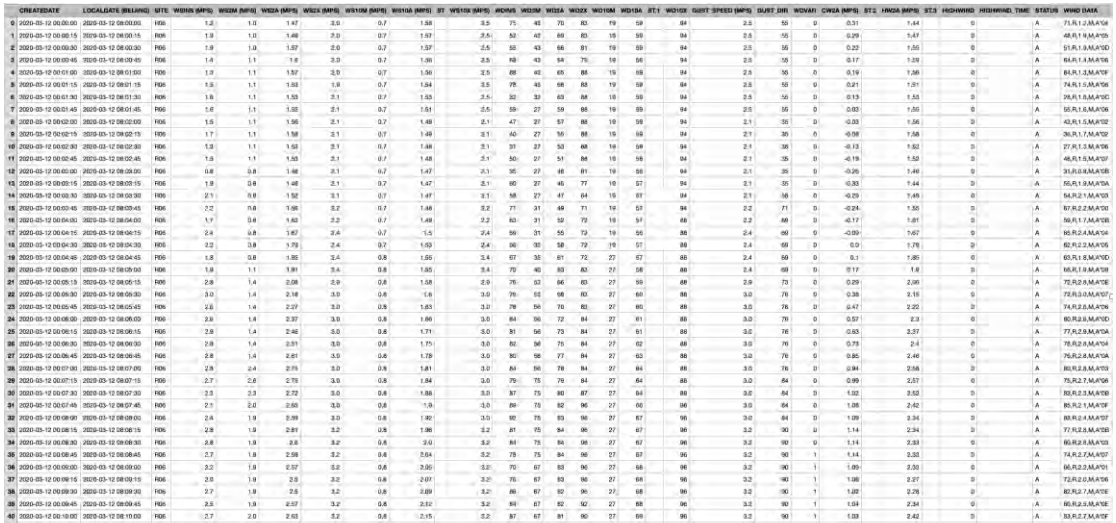
能见度数据包含了 4 项描述信息包括能见度（RVR≤2000 米，MOR≤10000 米），1 分钟平均 RVR 值（RVR_1A），1 分钟平均 MOR 值（MOR_1A），灯光数据（LIGHT_S）。从北京时间 8 点起，每隔 15 秒钟监测一次即每分钟有 4 次数据，共 5755 项数据。



CREATEDATE	LOCALDATE (BEIJING)	SITE	RVR11M	RVR_1A	ST	RVR1X	RVR_10M	RVR_10A	ST	RVR_10X	RVR_TEND	MOR_1A	MOR_10M	MOR_10A	MOR_10X	VIS1K	VIS1A	VIS10A	BL1A	RVR DATA	MOR_RAW	VIS_RAW	LIGHTS
2020-03-12 00:00:00	2020-03-12 08:00:00	R05	2800	3000		3000	2800	2900		3000	0	3000	2800	2900	3000	1208	3000	3000	5461	3087	3088	3068	100
2020-03-12 00:00:15	2020-03-12 08:00:15	R06	2800	3000		3000	2800	2900		3000	0	3000	2800	2900	3000	1208	3000	3000	5467	3099	3100	3100	100
2020-03-12 00:00:30	2020-03-12 08:00:30	R06	2800	3000		3000	2800	2900		3000	0	3000	2800	2900	3100	1208	3000	3000	5474	3104	3105	3105	100
2020-03-12 00:00:45	2020-03-12 08:00:45	R06	2800	3000		3000	2800	2900		3000	0	3100	2800	2900	3100	1208	3100	3000	5482	3110	3110	3110	100
2020-03-12 00:01:00	2020-03-12 08:01:00	R06	2800	3000		3000	2800	2900		3000	0	3100	2800	2900	3100	1208	3100	3000	5489	3113	3114	3114	100
2020-03-12 00:01:15	2020-03-12 08:01:15	R06	2800	3000		3000	2800	2900		3000	0	3100	2800	2900	3100	1208	3100	3000	5493	3117	3118	3118	100
2020-03-12 00:01:30	2020-03-12 08:01:30	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1208	3100	3000	5497	3118	3119	3119	100
2020-03-12 00:01:45	2020-03-12 08:01:45	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1208	3100	3000	5498	3130	3131	3131	100
2020-03-12 00:02:00	2020-03-12 08:02:00	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1213	3100	3000	5500	3148	3148	3148	100
2020-03-12 00:02:15	2020-03-12 08:02:15	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1216	3100	3000	5447	3165	3165	3165	100
2020-03-12 00:02:30	2020-03-12 08:02:30	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1220	3100	3000	5505	3160	3160	3160	100
2020-03-12 00:02:45	2020-03-12 08:02:45	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1213	3100	3000	5507	3154	3155	3155	100
2020-03-12 00:03:00	2020-03-12 08:03:00	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1213	3100	3000	5510	3147	3148	3148	100
2020-03-12 00:03:15	2020-03-12 08:03:15	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1213	3100	3000	5514	3137	3137	3137	100
2020-03-12 00:03:30	2020-03-12 08:03:30	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1213	3100	3100	5518	3146	3147	3147	100
2020-03-12 00:03:45	2020-03-12 08:03:45	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1213	3100	3100	5524	3152	3152	3152	100
2020-03-12 00:04:00	2020-03-12 08:04:00	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1213	3100	3100	5530	3143	3143	3143	100
2020-03-12 00:04:15	2020-03-12 08:04:15	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1208	3100	3100	5535	3134	3134	3134	100
2020-03-12 00:04:30	2020-03-12 08:04:30	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1208	3100	3100	5540	3126	3126	3126	100
2020-03-12 00:04:45	2020-03-12 08:04:45	R06	2800	3000		3000	2800	3000		3000	0	3100	2800	3000	3100	1205	3100	3100	5545	3104	3105	3105	100
2020-03-12 00:05:00	2020-03-12 08:05:00	R06	2800	3000		3000	2900	3000		3000	0	3100	2900	3000	3100	1201	3100	3100	5548	3089	3090	3090	100
2020-03-12 00:05:15	2020-03-12 08:05:15	R06	2900	3000		3000	2900	3000		3000	0	3100	2900	3000	3100	1205	3100	3100	5551	3099	3099	3099	100
2020-03-12 00:05:30	2020-03-12 08:05:30	R05	2900	3000		3000	2900	3000		3000	0	3000	2900	3000	3100	1201	3000	3100	5555	3095	3095	3095	100
2020-03-12 00:05:45	2020-03-12 08:05:45	R06	2900	3000		3000	2900	3000		3000	0	3000	2900	3000	3100	1201	3000	3100	5557	3096	3097	3097	100
2020-03-12 00:06:00	2020-03-12 08:06:00	R05	3000	3000		3000	2900	3000		3000	0	3000	2900	3000	3100	1201	3000	3100	5559	3097	3097	3097	100
2020-03-12 00:06:15	2020-03-12 08:06:15	R05	3000	3000		3000	2900	3000		3000	0	3000	2900	3000	3100	1201	3000	3100	5561	3092	3092	3092	100
2020-03-12 00:06:30	2020-03-12 08:06:30	R05	3000	3000		3000	2900	3000		3000	0	3000	2900	3000	3100	1205	3000	3100	5564	3111	3111	3111	100
2020-03-12 00:06:45	2020-03-12 08:06:45	R05	3000	3000		3000	2900	3000		3000	0	3138	2900	3000	3100	1208	3100	3100	5568	3157	3138	3158	100
2020-03-12 00:07:00	2020-03-12 08:07:00	R06	3000	3000		3000	2900	3000		3000	0	3138	2900	3100	3100	1208	3100	3100	5574	3144	3145	3145	100
2020-03-12 00:07:15	2020-03-12 08:07:15	R05	3000	3000		3000	3000	3000		3000	0	3138	3000	3100	3100	1213	3100	3100	5580	3171	3171	3171	100
2020-03-12 00:07:30	2020-03-12 08:07:30	R05	3000	3000		3000	3000	3000		3000	0	3120	3000	3100	3100	1208	3100	3100	5586	3156	3156	3156	100
2020-03-12 00:07:45	2020-03-12 08:07:45	R05	3000	3000		3000	3000	3000		3000	0	3138	3000	3100	3100	1208	3100	3100	5592	3134	3135	3135	100
2020-03-12 00:08:00	2020-03-12 08:08:00	R05	3000	3000		3000	3000	3000		3000	0	3120	3000	3100	3100	1205	3100	3100	5598	3120	3121	3121	100
2020-03-12 00:08:15	2020-03-12 08:08:15	R05	3000	3000		3000	3000	3000		3000	0	3138	3000	3100	3100	1201	3100	3100	5603	3098	3099	3099	100
2020-03-12 00:08:30	2020-03-12 08:08:30	R05	3000	3000		3000	3000	3000		3000	0	3120	3000	3100	3100	1201	3100	3100	5607	3103	3103	3103	100
2020-03-12 00:08:45	2020-03-12 08:08:45	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1201	3100	3100	5609	3112	3113	3113	100
2020-03-12 00:09:00	2020-03-12 08:09:00	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1201	3100	3100	5611	3120	3121	3121	100
2020-03-12 00:09:15	2020-03-12 08:09:15	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1205	3100	3100	5614	3130	3131	3131	100
2020-03-12 00:09:30	2020-03-12 08:09:30	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1205	3100	3100	5618	3145	3146	3146	100
2020-03-12 00:09:45	2020-03-12 08:09:45	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1208	3100	3100	5626	3149	3150	3150	100
2020-03-12 00:10:00	2020-03-12 08:10:00	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1208	3100	3100	5632	3159	3159	3159	100
2020-03-12 00:10:15	2020-03-12 08:10:15	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1208	3100	3100	5639	3171	3172	3172	100
2020-03-12 00:10:30	2020-03-12 08:10:30	R05	3000	3000		3000	3000	3000		3000	0	3100	3000	3100	3100	1208	3100	3100	5646	3161	3161	3161	100

图 2.2 能见度资料信息

风速数据包含了 3 项描述风速的信息，包括 2 分钟平均风速（WS2A），2 分钟平均风向（WD2A），2 分钟平均垂直风速（CW2A）。与温度与湿度数据相同，也是从北京时间 8 点起，每隔 15 秒钟监测一次即每分钟有 4 次数据，共 5755 项数据。



	CREATEDATE	LOCALDATE (BEIJING)	SITE	WSHA (MPS)	WSZM (MPS)	WS2A (MPS)	WS2X (MPS)	WS10M (MPS)	WS10A (MPS)	ST	WS10X (MPS)	WDHS	WD2M	WD2A	WD2X	WD10M	WD10A	ST 1	WD10X	GUST SPEED (MPS)	GUST DIR	WDVAR	CW2A (MPS)	ST 2	HWOA (MPS)	ST 3	HIGHWIND	HIGHWIND_TIME	STATUS	WIND DATA
0	2020-03-12 00:00:00	2020-03-12 08:00:00	R05	1.2	1.0	1.47	3.0	0.7	1.58		3.5	75	40	70	83	19	59		94	2.5	55	0	0.31		1.44		0		A	71,R,1.2,M,A'08
1	2020-03-12 00:00:15	2020-03-12 08:00:15	R06	1.9	1.0	1.48	3.0	0.7	1.57		3.5	52	42	69	83	19	59		94	2.5	55	0	0.29		1.47		0		A	48,R,1.9,M,A'08
2	2020-03-12 00:00:30	2020-03-12 08:00:30	R06	1.9	1.0	1.57	3.0	0.7	1.57		3.5	58	43	66	81	19	59		94	2.5	55	0	0.22		1.55		0		A	51,R,1.9,M,A'0D
3	2020-03-12 00:00:45	2020-03-12 08:00:45	R06	1.4	1.1	1.6	3.0	0.7	1.56		3.5	69	43	64	75	18	56		94	2.5	55	0	0.17		1.59		0		A	64,R,1.4,M,A'06
4	2020-03-12 00:01:00	2020-03-12 08:01:00	R06	1.0	1.1	1.57	3.0	0.7	1.56		3.5	68	43	65	88	19	59		94	2.5	55	0	0.19		1.56		0		A	64,R,1.3,M,A'0F
5	2020-03-12 00:01:15	2020-03-12 08:01:15	R06	1.5	1.1	1.53	3.0	0.7	1.54		3.5	78	45	66	83	19	59		94	2.5	55	0	0.21		1.51		0		A	74,R,1.5,M,A'08
6	2020-03-12 00:01:30	2020-03-12 08:01:30	R06	1.6	1.1	1.53	3.1	0.7	1.53		3.5	32	33	69	88	18	59		94	2.5	55	0	0.13		1.53		0		A	26,R,1.5,M,A'0C
7	2020-03-12 00:01:45	2020-03-12 08:01:45	R06	1.6	1.1	1.55	3.1	0.7	1.51		3.5	59	27	59	88	19	59		94	2.5	55	0	0.03		1.55		0		A	55,R,1.6,M,A'06
8	2020-03-12 00:02:00	2020-03-12 08:02:00	R05	1.5	1.1	1.56	3.1	0.7	1.49		3.1	47	27	57	88	18	59		94	3.1	55	0	-0.03		1.56		0		A	42,R,1.5,M,A'02
9	2020-03-12 00:02:15	2020-03-12 08:02:15	R05	1.7	1.1	1.58	3.1	0.7	1.49		3.1	40	27	55	88	19	59		94	2.1	55	0	-0.08		1.58		0		A	36,R,1.7,M,A'02
10	2020-03-12 00:02:30	2020-03-12 08:02:30	R06	1.5	1.1	1.53	3.1	0.7	1.48		3.1	31	27	53	88	19	59		94	2.1	55	0	-0.13		1.52		0		A	27,R,1.5,M,A'06
11	2020-03-12 00:02:45	2020-03-12 08:02:45	R06	1.5	1.1	1.53	3.1	0.7	1.48		3.1	50	27	51	88	18	58		94	2.1	55	0	-0.19		1.53		0		A	48,R,1.5,M,A'07
12	2020-03-12 00:03:00	2020-03-12 08:03:00	R06	0.8	0.8	1.48	3.1	0.7	1.47		3.1	35	27	46	81	19	58		94	2.1	55	0	-0.25		1.49		0		A	31,R,0.8,M,A'0B
13	2020-03-12 00:03:15	2020-03-12 08:03:15	R06	1.9	0.8	1.48	3.1	0.7	1.47		3.1	60	27	45	77	18	57		94	2.1	55	0	-0.33		1.44		0		A	55,R,1.9,M,A'0A
14	2020-03-12 00:03:30	2020-03-12 08:03:30	R05	2.1	0.8	1.52	3.1	0.7	1.47		3.1	58	27	47	64	19	57		94	2.1	55	0	-0.29		1.49		0		A	54,R,2.1,M,A'03
15	2020-03-12 00:03:45	2020-03-12 08:03:45	R05	2.0	0.8	1.56	3.2	0.7	1.48		3.2	71	31	49	71	19	57		94	2.2	71	0	-0.24		1.55		0		A	67,R,2.2,M,A'00
16	2020-03-12 00:04:00	2020-03-12 08:04:00	R05	1.7	0.8	1.63	3.2	0.7	1.48		3.2	63	31	52	72	19	57		88	2.2	68	0	-0.17		1.61		0		A	59,R,1.7,M,A'0B
17	2020-03-12 00:04:15	2020-03-12 08:04:15	R05	2.4	0.8	1.67	3.4	0.7	1.5		3.4	59	31	55	72	19	56		88	2.4	69	0	-0.09		1.67		0		A	65,R,2.4,M,A'04
18	2020-03-12 00:04:30	2020-03-12 08:04:30	R05	2.2	0.8	1.79	3.4	0.7	1.53		3.4	66	32	58	72	19	57		88	2.4	69	0	0.0		1.78		0		A	62,R,2.2,M,A'05
19	2020-03-12 00:04:45	2020-03-12 08:04:45	R05	1.8	0.8	1.85	3.4	0.8	1.55		3.4	67	33	61	72	22	57		88	2.4	69	0	0.1		1.85		0		A	63,R,1.8,M,A'0D
20	2020-03-12 00:05:00	2020-03-12 08:05:00	R06	1.9	1.1	1.91	3.4	0.8	1.55		3.4	70	40	63	83	22	58		88	2.4	69	0	0.17		1.9		0		A	68,R,1.9,M,A'0B
21	2020-03-12 00:05:15	2020-03-12 08:05:15	R06	2.8	1.4	2.08	2.9	0.8	1.58		2.9	76	63	66	83	27	59		88	2.9	73	0	0.29		2.06		0		A	72,R,2.8,M,A'0E
22	2020-03-12 00:05:30	2020-03-12 08:05:30	R06	3.0	1.4	2.18	3.0	0.8	1.6		3.0	76	68	68	83	27	60		88	3.0	76	0	0.38		2.15		0		A	72,R,3.0,M,A'07
23	2020-03-12 00:05:45	2020-03-12 08:05:45	R06	2.8	1.4	2.27	3.0	0.8	1.63		3.0	78	56	70	83	27	60		88	3.0	76	0	0.47		2.22		0		A	74,R,2.8,M,A'06
24	2020-03-12 00:06:00	2020-03-12 08:06:00	R06	2.8	1.4	2.37	3.0	0.8	1.66		3.0	84	66	72	84	27	61		88	3.0	79	0	0.57		2.3		0		A	80,R,2.8,M,A'0D
25	2020-03-12 00:06:15	2020-03-12 08:06:15	R06	2.9	1.4	2.46	3.0	0.8	1.71		3.0	81	66	73	84	27	61		88	3.0	76	0	0.63		2.37		0		A	77,R,2.9,M,A'0A
26	2020-03-12 00:06:30	2020-03-12 08:06:30	R06	2.9	1.4	2.51	3.0	0.8	1.71		3.0	82	66	75	84	27	62		88	3.0	76	0	0.75		2.4		0		A	78,R,2.9,M,A'0A
27	2020-03-12 00:06:45	2020-03-12 08:06:45	R06	2.8	1.4	2.61	3.0	0.8	1.79		3.0	80	66	77	84	27	63		88	3.0	76	0	0.85		2.49		0		A	76,R,2.8,M,A'0A
28	2020-03-12 00:07:00	2020-03-12 08:07:00	R06	2.8	2.4	2.75	3.0	0.8	1.81		3.0	84	66	78	84	27	64		88	3.0	76	0	0.94		2.56		0		A	80,R,2.8,M,A'03
29	2020-03-12 00:07:15	2020-03-12 08:07:15	R06	2.7	2.6	2.75	3.0	0.8	1.84		3.0	79	75	76	84	27	64		88	3.0	84	0	0.99		2.57		0		A	75,R,2.7,M,A'06
30	2020-03-12 00:07:30	2020-03-12 08:07:30	R06	2.3	2.3	2.72	3.0	0.8	1.88		3.0	87	75	80	87	27	64		88	3.0	84	0	1.02		2.62		0		A	80,R,2.3,M,A'0B
31	2020-03-12 00:07:45	2020-03-12 08:07:45	R06	2.1	2.0	2.60	3.0	0.8	1.9		3.0	89	75	82	96	27	66		96	3.0	84	0	1.06		2.62		0		A	85,R,2.1,M,A'0F
32	2020-03-12 00:08:00	2020-03-12 08:08:00	R06	2.4	1.9	2.59	3.0	0.8	1.92		3.0	92	75	83	96	27	67		96	3.0	84	0	1.09		2.64		0		A	88,R,2.4,M,A'07
33	2020-03-12 00:08:15	2020-03-12 08:08:15	R06	2.8	1.9	2.61	3.2	0.8	1.96		3.2	87	75	84	96	27	67		96	3.2	90	0	1.14		2.64		0		A	77,R,2.8,M,A'0B
34	2020-03-12 00:08:30	2020-03-12 08:08:30	R06	2.8	1.9	2.6	3.2	0.8	2.0		3.2	84	75	84	96	27	67		96	3.2	90	0	1.14		2.63		0		A	80,R,2.8,M,A'03
35	2020-03-12 00:08:45	2020-03-12 08:08:45	R06	2.7	1.9	2.59	3.2	0.8	2.04		3.2	78	75	84	96	27	67		96	3.2	90	1	1.14		2.63		0		A	74,R,2.7,M,A'07
36	2020-03-12 00:09:00	2020-03-12 08:09:00	R06	3.2	1.9	2.57	3.2	0.8	2.06		3.2	70	67	83	96	27	68		96	3.2	90	1	1.09		2.59		0		A	66,R,3.2,M,A'01
37	2020-03-12 00:09:15	2020-03-12 08:09:15	R06	2.8	1.9	2.6	3.2	0.8	2.07		3.2	76	67	83	96	27	68		96	3.2	90	1	1.09		2.27		0		A	72,R,2.0,M,A'06
38	2020-03-12 00:09:30	2020-03-12 08:09:30	R06	2.7	1.9	2.6	3.2	0.8	2.09		3.2	86	67	82	96	27	68		96	3.2	90	1	1.09		2.28		0		A	82,R,2.7,M,A'0E
39	2020-03-12 00:09:45	2020-03-12 08:09:45	R06	2.5	1.9	2.57	3.2	0.8	2.12		3.2	84	67	82	92	27	68		96	3.2	90	1	1.04		2.34		0		A	80,R,2.5,M,A'0E
40	2020-03-12 00:10:00	2020-03-12 08:10:00	R06	2.7	2.0	2.63	3.2	0.8	2.15		3.2	87	67	81	90	27	69		96	3.2	90	1	1.03		2.42		0		A	83,R,2.7,M,A'0F
41	2020-03-12 00:10:15	2020-03-12 08:10:15	R06	2.6	2.0	2.59	2.9	0.8	2.18		3.2	90	67	81	91	27	70		96	3.2	90	1	1.01		2.39		0		A	86,R,2.6,M,A'05
42	2020-03-12 00:10:30	2020-03-12 08:10:30	R06	3.2	2.0	2.61	3.2	0.8	2.21		3.2	75	67	82	91	27	70		96	3.2	75	1	1.00		2.39		0		A	71,R,3.2,M,A'06
43	2020-03-12 00:10:45	2020-03-12 08:10:45	R05	2.9	2.0	2.6	3.2	0.8	2.24		3.2	73	67	81	91	27	71		96	3.2	75	1	1.01		2.39		0		A	69,R,2.9,M,A'0F
44	2020-03-12 00:11:00	2020-03-12 08:11:00	R05	2.7	2.0	2.59	3.2	0.8	2.26		3.2	80	67	81	91	27	71		96	3.2	75	1	1.01		2.39		0		A	76,R,2.7,M,A'05
45	2020-03-12 00:11:15	2020-03-12 08:11:15	R05	2.5	2.2	2.63	3.2	0.8	2.29		3.2	82	67	82	91	27	71		96	3.2	75	1	1.07		2.4		0		A	78,R,2.5,M,A'0F

图 2.3 风速资料信息

问题 2 给出了在 2020 年 03 月 13 号的机场视频数据，共 11 小时 33 分 28 秒。描述了机场从雾的产生到消散的全部过程，形象直观。有助于我们对雾的整个形成与消散过程的理解，更好的建立能见度估计模型。

图 2.4 机场视频数据

问题 3 给出了 100 张高速公路的视频截图，时间是在 2016 年 04 月 14 号，从 6 点 30 分 26 秒到 7 点 39 分 11 秒，每间隔 42 秒钟截取一张视频图片。

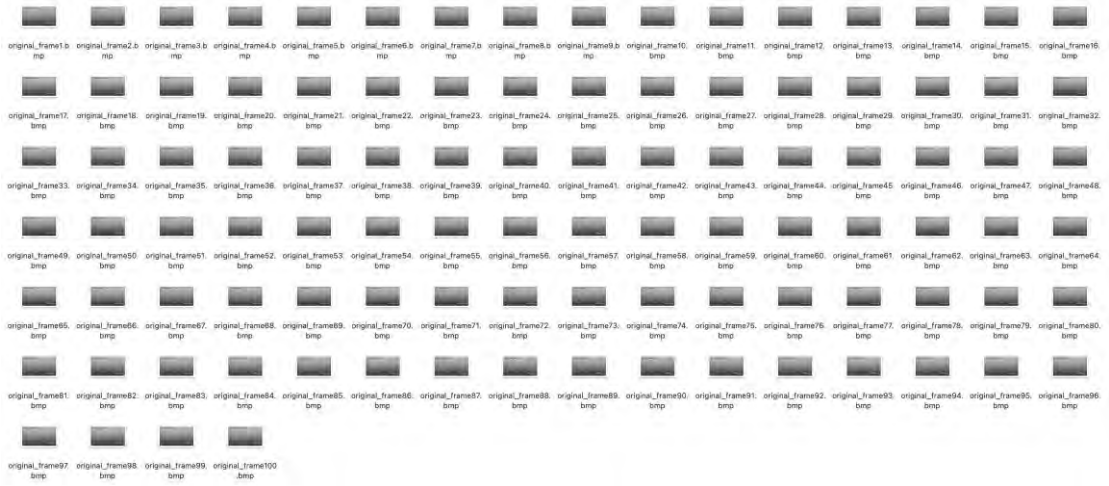


图 2.5 高速公路视频截图

3 模型基本假设与符号说明

3.1 模型基本假设

1. 假设，其他气象因素对能见度影响很小；
2. 假设，气象站点传感器精度在 2019-12 到 2020-03 之间没有变化；
3. 假设监控视频中均包含地平线（最大可视点）；
4. 假设监控区域不存在较大的海拔起伏（道路面为近似平面）；
5. 假设摄像头与人眼对能见度的感知相同；

3.2 模型基本符号说明

表 3.3.2 模型基本符号说明

符号	相关说明
V	RAR_1A 能见度
G_RVR	与 RAR_1A 能见度对应的地面气象观测数据（因变量）
β_V	RAR_1A 能见度与选择的地面气象观测数据关系的系数
M	MOR_1A 能见度
G_MOR	与 MOR_1A 能见度对应的地面气象观测数据（因变量）
β_M	MOR_1A 能见度与选择的地面气象观测数据关系的系数
R^2	可决系数
MSE	均方根
G_W	WS2A
G_C	CW2A
G_L	LIGHTS
G_P	PAINS
G_T	TEMP
G_H	RH

4 问题一

4.1 问题描述与分析

根据题目的描述，本问题要求根据题目提供的地面气象观测数据建模描述能见度与地面气象观测数据的关系，由问题可知，需要对已知数据进行处理分析，建立能见度与温度和湿度、能见度与风速之间的关系。

问题一属于多元线性回归问题，题目给出的数据中有 6 项指标来描述温度与湿度，有 3 项指标描述风速，4 项指标描述能见度信息，本文可以分别对各项指标进行相关性分析，根据相关性矩阵，得到相关性较高的几项指标，舍弃相关性较低的指标，最后建立回归模型来描述能见度与地面气象观测之间的关系，并推导出具体关系式。

第一步：数据预处理。首先读取机场 AMOS 数据，查看并分析数据的缺失或异常情况并对其进行处理，得到比较合理的数据。然后根据时间对应，将三项地面气象观测数据即

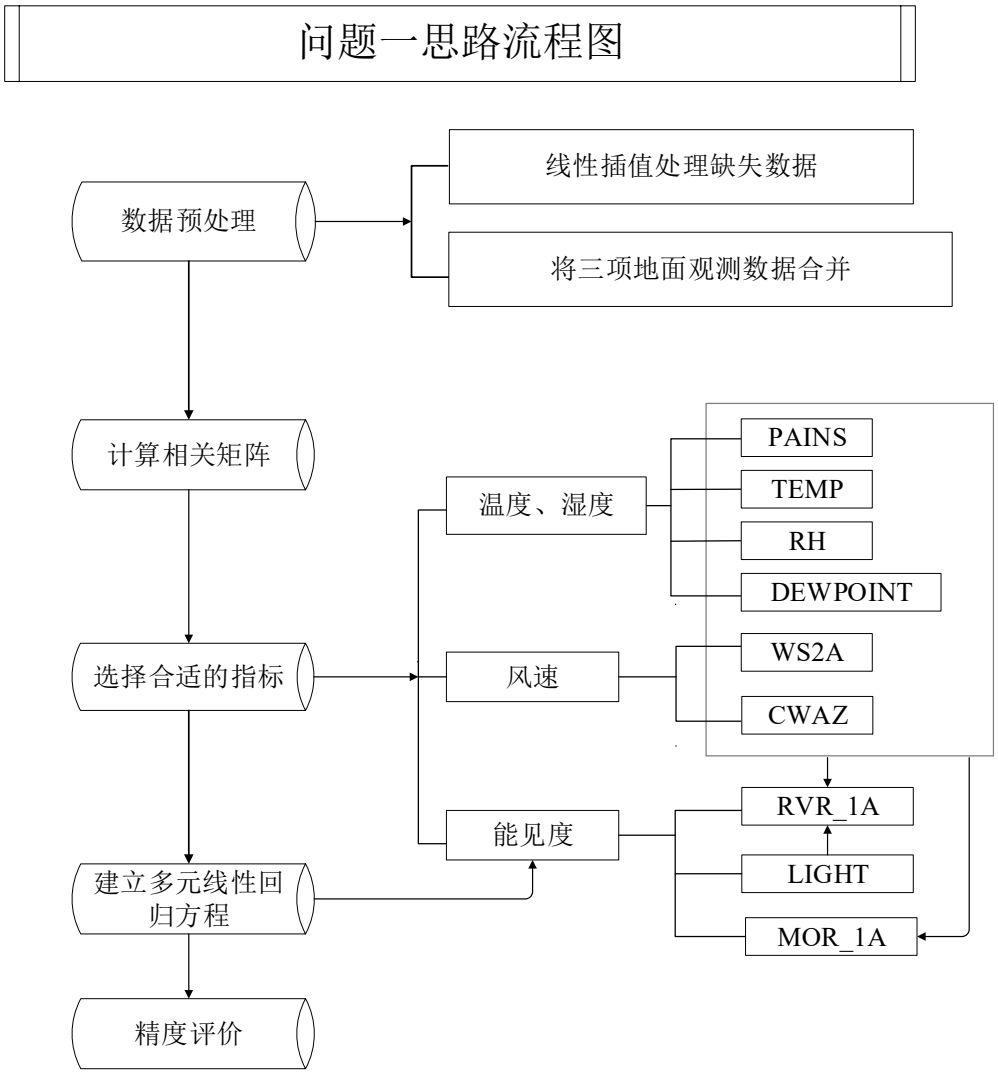
温度与湿度、能见度、风速数据进行合并处理。

第二步：计算相关性矩阵。对每项指标进行线性相关分析，得到相关性矩阵。

第三步：指标的选择。根据相关性大小，将相关性小的指标舍弃，在剩余的指标中选择合适的数据。

第四步：建立多元线性回归方程。找到能见度与地面观测数据之间的关系。

第五步：模型评价。对建立的多元线性回归方程评价分析，使结果能够更加合理地描述两者之间的关系，为以后数据预测做铺垫。问题一的总体任务如下所示：



问题一的思路流程图

4.2 问题求解与结果评价

依照上述分析，将问题分为四步求解：

4.2.1 数据预处理

首先读取能见度数据文件，由于大部分数据都是每隔 15 秒有一项数据，检查分析后发现有 5 个时间点没有对应的数据值，数据在时间上不连续，即数据有缺失情况，所以本文根据时间运用线性插值的方法对能见度资料缺失的数据进行填充。填充的数据如表 4.2.1 所示：

表 4.2.1 能见度补充的数据

CREATEDATE	SITE	RVR_1A	MOR_1A	LIGHTS
2020/3/12 1:12	R06	3000	3700	100
2020/3/12 10:05	R06	3000	4100	10
2020/3/12 13:29	R06	3000	2800	10
2020/3/12 17:39	R06	650	200	10
2020/3/12 20:11	R06	187	50	10
2019/12/15 1:26	R06	3000	7000	10
2019/12/15 8:00	R06	3000	6000	10
2019/12/15 10:36	R06	3000	5000	10
2019/12/15 12:21	R06	3000	6000	10
2019/12/15 23:18	R06	200	50	100

然后以同样的分析方法对风速数据进行缺失值的补充。

表 4.2.2 风补充的数据

CREATEDATE	SITE	WS2A	WD2A	CW2A
2020/3/12 1:12	R06	2.09	86	0.98
2020/3/12 10:05	R06	3.05	139	23.02
2020/3/12 13:29	R06	1.32	86	0.62
2020/3/12 17:39	R06	1.19	201	0.69
2020/3/12 20:11	R06	0.84	229	0.13
2019/12/15 1:26	R06	6.77	118	5.9
2019/12/15 8:00	R06	4.31	82	1.78
2019/12/15 10:36	R06	3.65	92	2.03
2019/12/15 12:21	R06	4.51	121	3.96
2019/12/15 23:18	R06	2.46	97	1.55

将处理得到的数据按照时间对应合并，合并后每隔 1 分钟对应 1 项温度与湿度数据，对应 4 项能见度数据和 4 项风速数据，形成一张表格后，方便以后的回归分析。

CREATEDATE	WS2A	WD2A	CW2A	RVR_1A	MOR_1A	LIGHTS	SITE	PAINS	QNH	QFE_R06	TEMP	RH	DEWPOINT
2020-03-12 00:00:00	1.47	70	0.31	3000	3000	100	R06	1016.9	1018.44	1016.88	9.7	85.0	7.32
2020-03-12 00:00:15	1.49	69	0.29	3000	3000	100	R06	1016.9	1018.44	1016.88	9.7	85.0	7.32
2020-03-12 00:00:30	1.57	66	0.22	3000	3000	100	R06	1016.9	1018.44	1016.88	9.7	85.0	7.32
2020-03-12 00:00:45	1.6	64	0.17	3000	3100	100	R06	1016.9	1018.44	1016.88	9.7	85.0	7.32
2020-03-12 00:01:00	1.57	65	0.19	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	85.0	7.42
2020-03-12 00:01:15	1.53	66	0.21	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	85.0	7.42
2020-03-12 00:01:30	1.53	63	0.13	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	85.0	7.42
2020-03-12 00:01:45	1.55	59	0.03	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	85.0	7.42
2020-03-12 00:02:00	1.56	57	-0.03	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	84.0	7.24
2020-03-12 00:02:15	1.58	55	-0.08	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	84.0	7.24
2020-03-12 00:02:30	1.53	53	-0.13	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	84.0	7.24
2020-03-12 00:02:45	1.53	51	-0.19	3000	3100	100	R06	1017.0	1018.54	1016.98	9.8	84.0	7.24
2020-03-12 00:03:00	1.48	48	-0.26	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:03:15	1.48	45	-0.33	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:03:30	1.52	47	-0.29	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:03:45	1.56	49	-0.24	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:04:00	1.62	52	-0.17	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:04:15	1.67	55	-0.09	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:04:30	1.79	58	0.0	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:04:45	1.85	61	0.1	3000	3100	100	R06	1017.1	1018.64	1017.08	9.9	84.0	7.34
2020-03-12 00:05:00	1.91	63	0.17	3000	3100	100	R06	1017.1	1018.64	1017.08	10.0	84.0	7.44
2020-03-12 00:05:15	2.08	66	0.29	3000	3100	100	R06	1017.1	1018.64	1017.08	10.0	84.0	7.44
2020-03-12 00:05:30	2.18	68	0.38	3000	3000	100	R06	1017.1	1018.64	1017.08	10.0	84.0	7.44

图 4.2.1 合并后总数据

4.2.2 计算相关性矩阵

总体相关性系数用 ρ 来表示：

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sqrt{\operatorname{var}(X)\operatorname{var}(Y)}} = \frac{E\left[(X-\mu x)(Y-\mu y)\right]}{\sqrt{\sigma^2{}_X \sigma^2{}_Y}} \quad (4\text{-}1)$$

其中，$\sigma^2{}_X$ 为 X 的总体方差，$\sigma^2{}_Y$ 是 Y 的总体方差，$\operatorname{cov}(X,Y)$ 是 x 与 y 的协方差。根据公式计算得到 12 项指标的协方差矩阵如下：

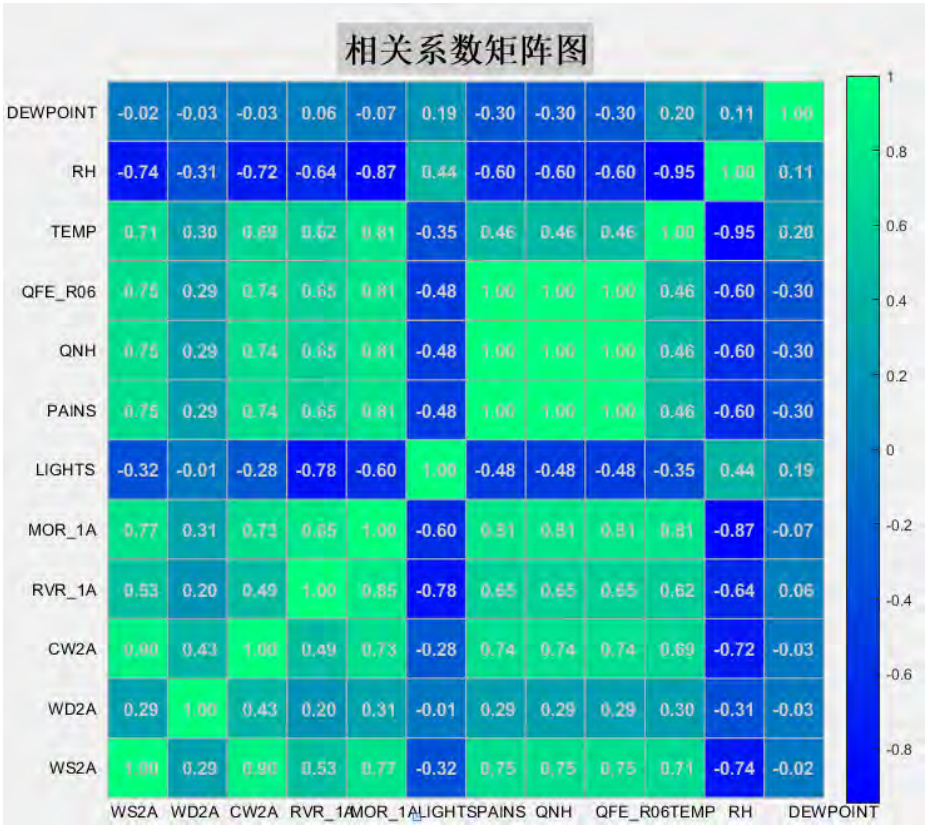


图 4.2.2 相关系数矩阵图

4.2.3 选择合适的指标

首先对于能见度的 3 个指标，查阅资料得知，RVR 不是直接测量的气象元素，它是经大气投射仪测量后考虑大气消光系数、视觉阈值和跑道灯强度而计算的数值，也可经前向散射仪,测量后计算得到。即 RVR 数值的大小与跑道灯光的强度有关。故对于 RVR 能见度定义，建立关系式时，关系式中需要考虑灯光数据的影响。而 MOR 能见度定义中与灯光因素无关，所以在建模分析是不考虑灯光对能见度的影响。

对于温度与湿度的 6 项指标，其中描述气压的指标有 3 项，包括本站气压（PAINS），修正海平面气压（QNH）及飞机着陆地区最高点气压（QFE），由（2）中得到的相关性表格与下图分析可知，对于 RVR 和 MOR 两种能见度定义，三种气压与能见度都是成正相关，也就是随气压的增大，能见度越大。并且三种气压的相关系数非常接近，其差值均<0.001，说明三者具有极强的相关性，故在本题中，选择更接近本站气压数据的本站气压（PAINS）作为唯一的气压影响因素。

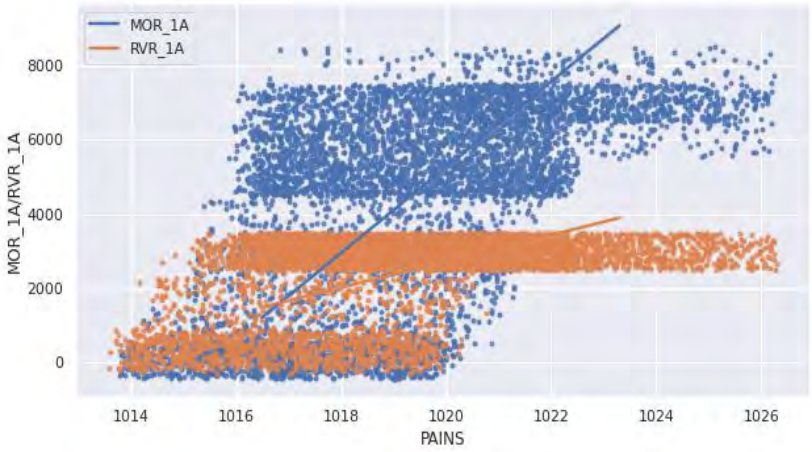


图 4.2.3 本站气压（PAINS）与 MOR_1A 和 RVR_1A 的关系

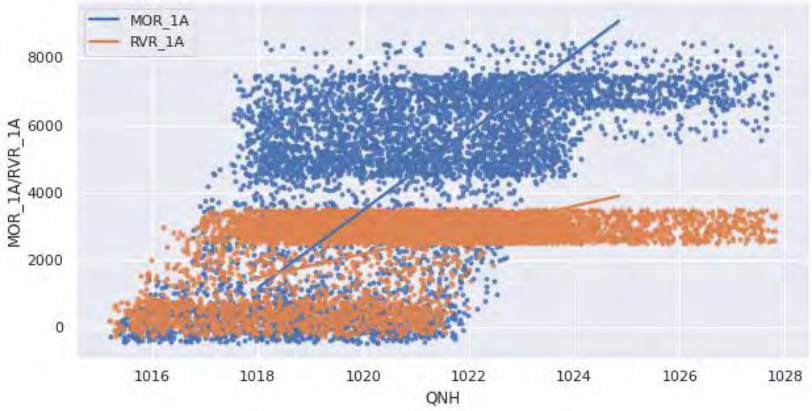


图 4.2.4 修正海平面气压（QNH）与 MOR_1A 和 RVR_1A 的关系

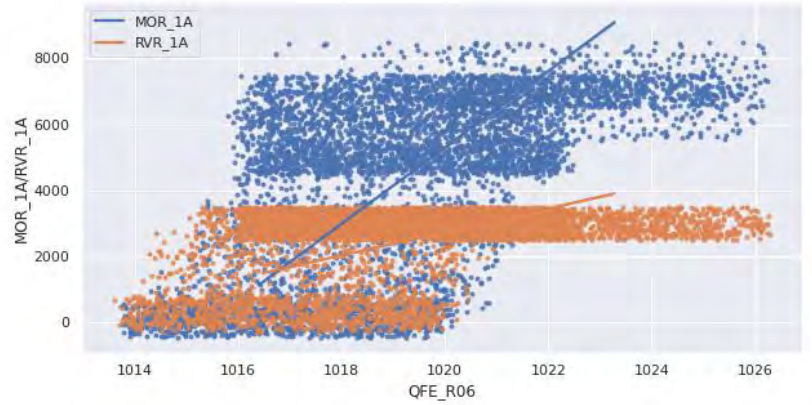


图 4.2.5 飞机着陆地区最高点气压（QFE）与 MOR_1A 和 RVR_1A 的关系

同上分析可知，描述湿度的指标有两项，包括相对湿度（RH）和露点湿度（DEWPOINT），经过相关性分析表与下图分析可知，两种湿度数据与能见度都是呈负相关关系，也就是随着湿度越大，能见度越小。但是相对湿度（RH）与结果数据的相关系数的绝对值均大于 0.60，而露点湿度（DEWPOINT）绝对值均小于 0.1，所以可认为露点湿度与能见度无强相关。

所以在模型中，使用相对湿度（RH）代表湿度作为影响因素。

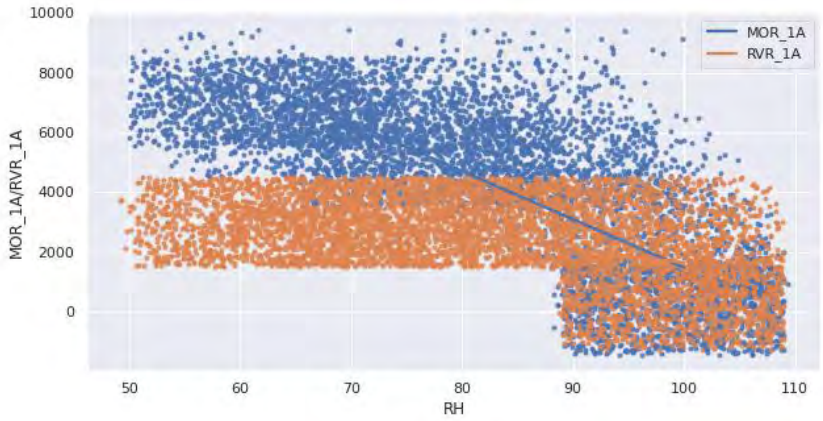


图 4.2.6 相对湿度（RH）与 MOR_1A 和 RVR_1A 的关系

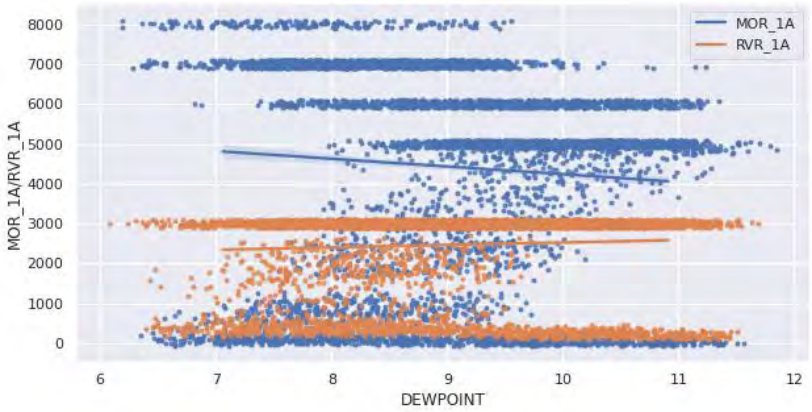


图 4.2.7 露点湿度（DEWPOINT）与 MOR_1A 和 RVR_1A 的关系

经过分析，温度（TEMP）与能见度（RVR_1A、MOR_1A）的线性关系如下图，可以看出温度与能见度呈正相关关系，随着温度的升高，两种能见度越大，反正越小。

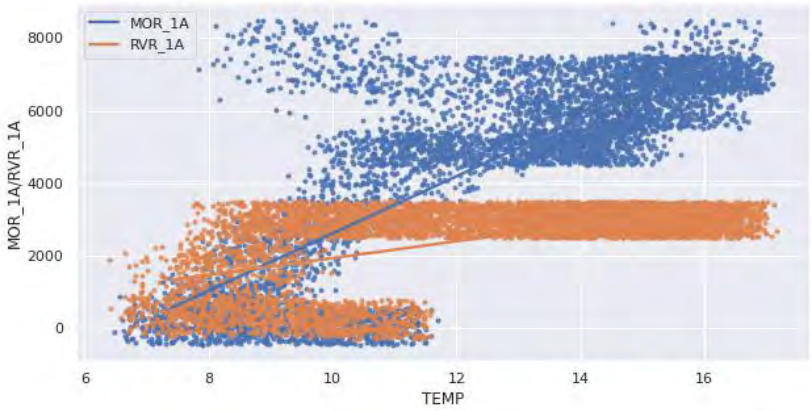


图 4.2.8 温度（TEMP）与 MOR_1A 和 RVR_1A 的关系

综上，本文共选择了温度与湿度 3 项指标包括本站气压（PAINS）、温度（TEMP）、相对湿度（RH）来描述温度与湿度和能见度之间的关系。

最后，对于风速数据，经过分析可知，风对雾的空间移动起关键作用，其中风速直接体现风力，所以水平风速和垂直风速都应考虑为模型影响因素。而对于风向，其影响雾的空间移动方向，运动方向对了解相邻区域的能见度变化有参考意义，但在本问题中，风向对采样站这一定点的能见度不会产生影响。由相关度分析表，风向（WD2A）与 RVR 和 MOR 数据的相关系数为 0.19676337 和 0.30543444，其分布如图，观察易得，其与能见度变化相关性较小，故不将其作为模型影响因素。两种风速与能见度的关系如下图所示，可见两者呈正相关，即随着风速的增大，能见度也越来越大，反之越小。

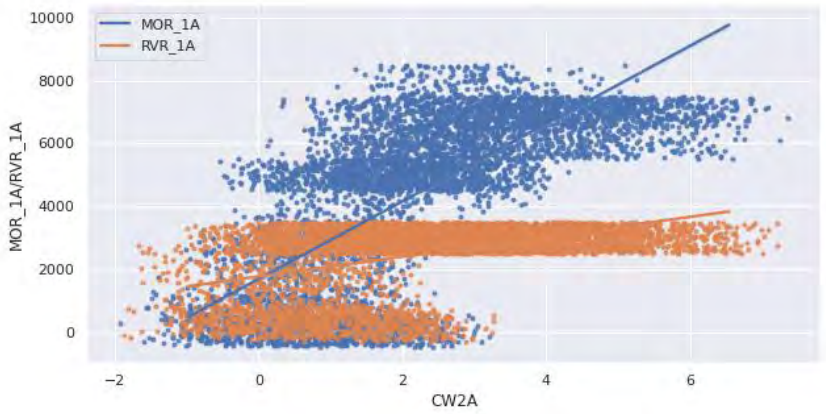


图 4.2.9　2 分钟平均垂直风速（CW2A）与 MOR_1A 和 RVR_1A 的关系

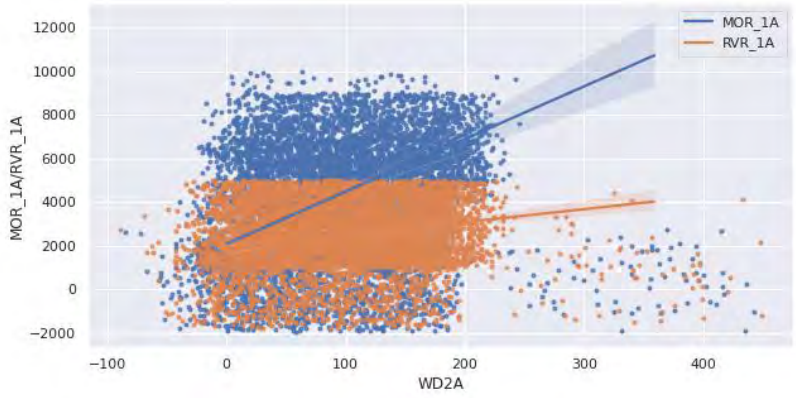


图 4.2.10 2 分钟平均风向（WD2A）与 MOR_1A 和 RVR_1A 的关系

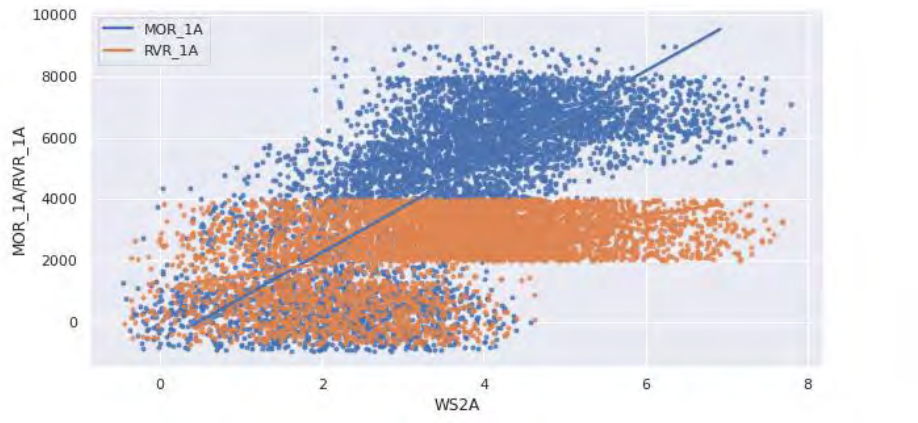


图 4.2.11　2 分钟平均风速（WS2A）与 MOR_1A 和 RVR_1A 的关系

综上，本文共选择了温度与湿度 3 项指标包括本站气压（PAINS）、温度（TEMP）、相对湿度（RH），风速 2 项指标包括 1 分钟平均风速（WS2A）和 2 分钟平均垂直风速（CW2A）来描述地面观测数据和能见度之间的关系。

4.2.4 建立多元线性回归方程

多元线性回归模型为

$$\begin{cases} y = \beta_0 + \beta_1 \chi_1 + \cdots + \beta_m \chi_m + \varepsilon \\ \varepsilon \sim N\left(0, \sigma^2\right) \end{cases} \tag{4-2}$$

其中，$\beta_0, \beta_1, \cdots, \beta_m, \sigma^2$ 都是与 $\chi_1, \chi_2, \cdots, \chi_m$ 无关的未知参数，其中 $\beta_0, \beta_1, \cdots, \beta_m$ 称为回归系数。

现得到 n 个独立观测数据 $\left(y_i, \chi_{i1}, \cdots, \chi_{im}\right), i = 1, \cdots, n, n > m$，由上式得

$$\begin{cases} y_i = \beta_0 + \beta_1 \chi_{i1} + \cdots + \beta_m \chi_{im} + \varepsilon_i \\ \varepsilon \sim N\left(0, \sigma^2\right), i = 1, \cdots, n \end{cases} \tag{4-3}$$

记

$$X = \begin{bmatrix} 1 & \chi_{11} & \cdots & \chi_{1m} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \chi_{n1} & \cdots & \chi_{mm} \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{4-4}$$

$$\varepsilon = \left[\varepsilon_1 \cdots \varepsilon_n\right]^T, \beta = \left[\beta_0 \ \ \beta_1 \cdots \beta_m\right]^T \tag{4-5}$$

表为

$$\begin{cases} Y = X\beta + \varepsilon \\ \varepsilon \sim N\left(0, \sigma^2 E_n\right) \end{cases} \tag{4-6}$$

其中 E_n 为 n 阶单位矩阵。

为了消除特征指标间单位和尺度差异的影响，使得每个特征指标的重要性更加均衡，对每个特征指标同等看待，需要对特征指标进行归一化。然后根据上述公式计算，可以得到 RVR_1A 能见度数据与选择的地面观测指标关系可以描述为：

$$V = G\_RVR * \beta_V + (-0.30233045048430185)$$

其中 $G\_RVR = \left[G_W, G_C, G_L, G_P, G_T, G_H\right]$，$\beta_V = \begin{bmatrix} 0.39915663 \\ -0.03183423 \\ -0.17340543 \\ 0.14826755 \\ ?.4237511 \\ 0.4039178 \end{bmatrix}$

综上，对于 RVR 能见度与地面观测数据表现线性相关的关系。具体来说，与 1 分钟平均风速（WS2A）、本站气压（PAINS）、温度（TEMP）和相对湿度（RH）成正相关，即随着这四项特征指标数据的增大，RVR 能见度增大，反之降低。与 2 分钟平均垂直风速（CW2A）和灯光数据（LIGHT）成负相关，即随着这两项特征指标的增大，RVR 能见度降低，反之增大。

MOR_1A 能见度数据与选择的地面观测指标关系可以描述为

$$M = G\_MOR * \beta_M + 0.3368816806579784$$

其中，$G\_MOR = \left[G_W, G_C, G_P, G_T, G_H\right]$，$\beta_M = \begin{bmatrix} 0.41775059 \\ 0.31520404 \\ 0.26087259 \\ 0.52221721 \\ 0.45588482 \end{bmatrix}$。

综上，对于 MOR 能见度与地面观测数据表现线性相关的关系。具体来说，与 1 分钟平均风速（WS2A）、2 分钟平均垂直风速（CW2A）、本站气压（PAINS）、温度（TEMP）和相对湿度（RH）这五项特征指标都成正相关，即随着这五项指标数据的增大，MOR 能见度增大，反之降低。

4.2.5 模型验证与分析评价

为了检验上述建立的回归方程的精度,本文利用可决系数和均方根误差两项精度评价指标来检验。其中，在模型参数估计时，使用的数据为 2019 和 2020 数据的合集共 11520 条数据，精度评价时，使用根据能见度数据升序排列后的 2020 年的数据。

决定系数 R^2 的计算公式为

$$R^2 = 1 - \frac{\sum_i \left(\hat{y}_i - y_i\right)^2}{\sum_i \left(\bar{y}_i - y_i\right)^2} \tag{4-7}$$

其中，分子部分表示真实值与预测值的平方之和，分母部分表示真实值与均值的平方

之和。结果越趋近于 0，说明拟合效果差；如果结果越接近于 1，说明模型拟合效果越好。

均方根误差的公式描述为：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - \hat{y}_i\right)^2 \qquad (4\text{-}8)$$

用来评价所建立的回归模型。

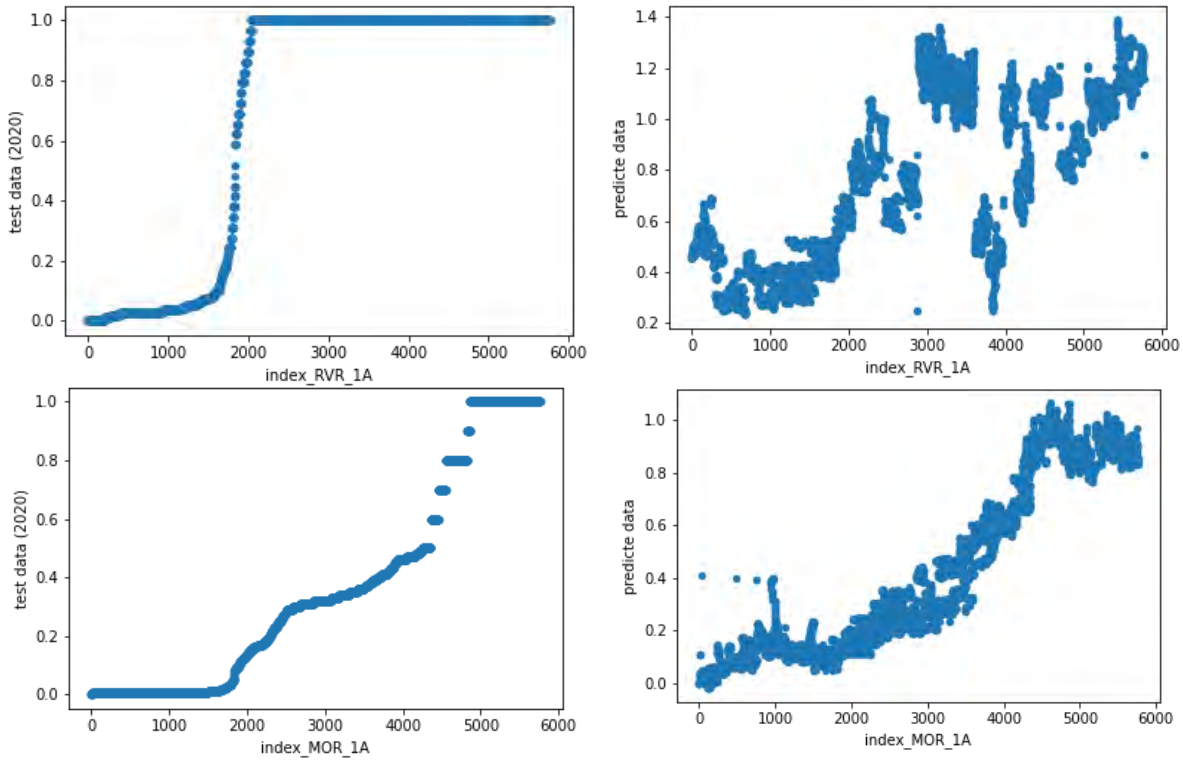


图 4.2.12 测试数据与预测数据对比图

由上述公式计算的得到的 RVR_1A 能见度与地面观测数据之间的回归方程模型的均方根误差为 0.07，可决系数为 0.56。MOR_1A 能见度与地面观测数据回归方程模型的均方根误差为 0.01，可决系数为 0.87。由得到的结果图可以看出，计算的线性回归模型效果较好，能够比较准确的得到能见度与地面观测数据的关系，从而为以后能见度预测做了技术支持，更加方便直接的预测能见度。

5 问题二

5.1 问题描述与分析

由问题背景可知，能见度的高低与大雾的形成消散过程密切相关。由于视频能见度检测方法利用密集监视摄像机捕获视频图像，能够检测到每个部分的可见度值，通过计算机图像处理和判断可以避免主观和任意性，能够在连续的时间范围内描述大雾整个变化过程，具有成本低、连续性强、操作简便和覆盖范围广的优点。题目要求实际上是利用给出的机场视频数据结合地面气象观测数据分析得到能见度的大小，要根据视频中的特征信息将其与能见度数据建立对应的关系，建立深度学习模型，分析得到能见度也就

是雾的浓厚程度，也就是将问题转化为成了分类问题，即利用深度学习模型将视频按照雾的浓度进行分类，评估能见度并进行精度验证。

第一步：数据预处理。按照机场 AMOS 数据的时间，获取对应时间内视频帧序列图像。

第二步：根据 MOR_1A 能见度等级划分图像。针对第一步挑选的图像按照能见度划分成四类，形成训练样本。

第三步：建立深度学习模型，利用处理后的视频数据估计能见度的大小。

第四步：展示模型估计能见度的结果，并对模型进行精度评价。

流程图如下所示：

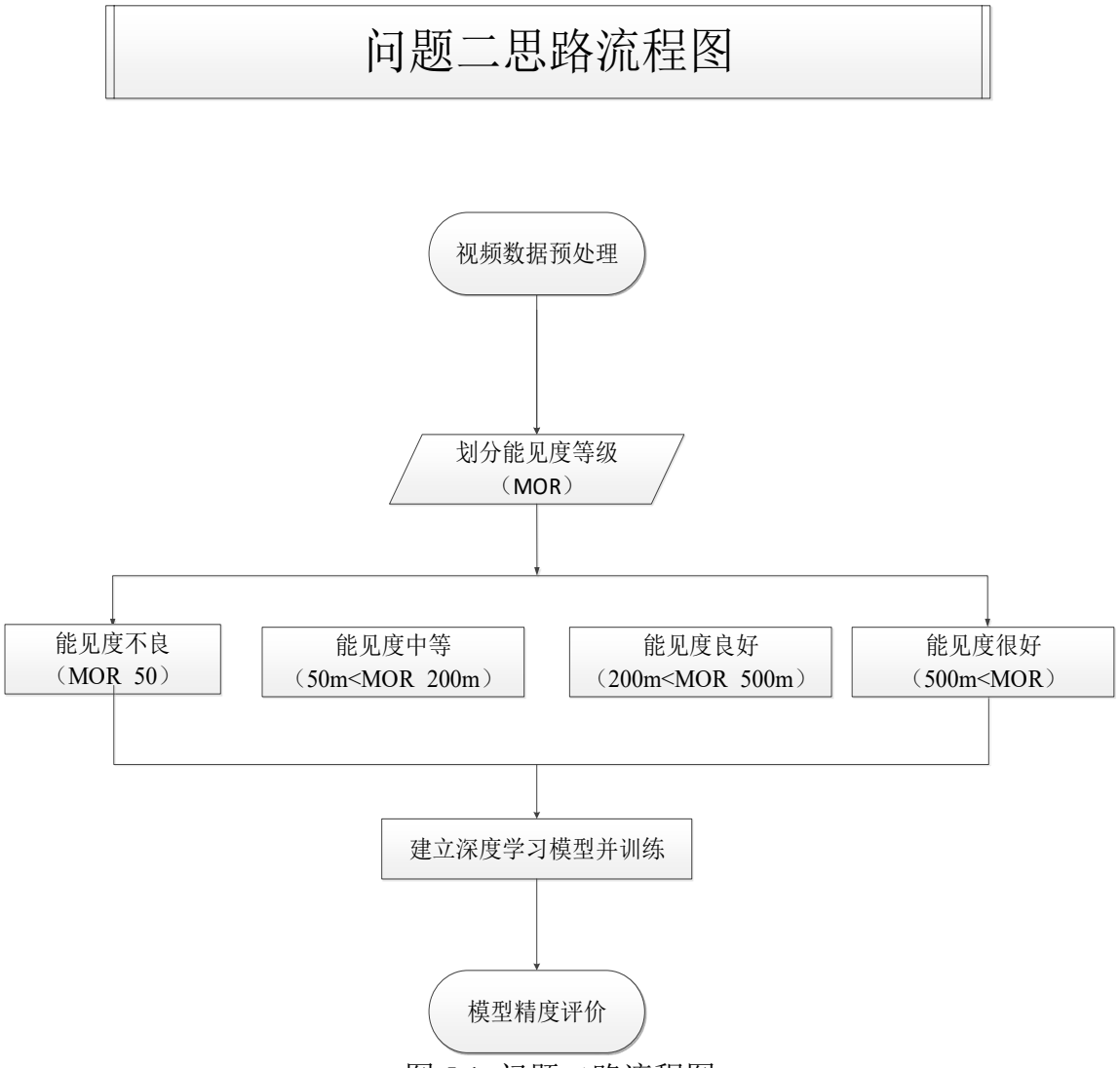


图 5.1 问题二路流程图

5.2 问题求解与结果评价

依照上述分析，将问题分为四步求解：

5.2.1 视频数据的预处理

根据题中所给机场视频数据，获取对应时间内视频帧序列图像，取样间隔为 22 帧，得到 2020 年 3 月 13 日 0 时至 7 时 59 分 00 秒时间内每一秒对应的能见度图像，共获取 27941 副影像，从中取出每一分钟内 15s、30s、45s、00s 四个时刻对应图像，共 1860 张图像构成能见度预测数据集，视频中的数据与提供的能见度数据形成一一对应关系。经过数据处理，本文可以得到比较准确的训练样本，为下面的工作做了充分的准备。

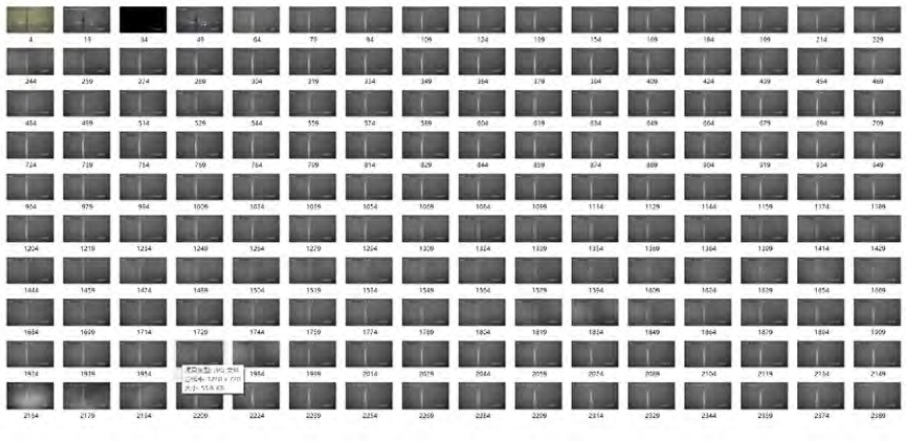


图 5.2.1 视频数据按帧序列图

5.2.2 视频数据能见度的划分

本文根据题目所给的能见度数据，结合能见度等级表，使用等级分析法对能见度进行分级，粗略地划分为：MOR≤50m，50m<MOR≤200m，200<MOR≤500m，500m<MOR≤1000m，MOR>1000m 五个等级（表 2.1）。将以上五个等级作为五种类别，建立基于视频数据的能见度估计深度学习模型。

表 2.1 能见度等级划分表

能见度范围（m）	能见度等级
MOR≤50	能见度不良
50m<MOR≤200m	能见度中等
200m<MOR≤500m	能见度良好
500m<MOR	能见度很好

通过能见度等级对样本数据集进行划分，数据总量为 1860，各类别数据量如图所示：

表 2.1 样本数量

能见度范围（m）	数据集数量
MOR≤50	1461
50m<MOR≤200m	223
200<MOR≤500m	92
500m<MOR≤1000m	84

将样本集中的 1620 张用于训练，180 张作为验证集用于调整分类模型参数并对分类效果进行初步评估，另外选取 60 张用于测试集，用于评估模型最终的分类效果。

5.2.3 建立深度学习模型

与传统的图像分类方法相比，卷积神经网络能够直接将图像数据作为输入，不需要人工的预处理及特征提取等操作，通过局部感受野、权值共享、池化层下采样减少了很多参数，尽可能保留了重要参数，同时具有一定程度平移、旋转、尺度和非线性形变稳定性，可以保留邻域的联系和空间的局部特点,因此被广泛应用于图像分类领域。

卷积神经网络训练算法包括 4 步，这 4 步分成两个阶段。首先是向前传播阶段。

选取样本(X, Y_p)，输入网络计算相应的实际输出

$$O_p = F_n(,,(F_2(F_1(X_p W^{(1)})W^{(2)}),,)W^{(n)}) \tag{5-1}$$

向后传播阶段，计算实际输出O_p与对应的理想输出Y_p的差，并按照极小化误差的方法调整权矩阵。精度描述为

$$E_P = \frac{1}{2}\sum_{j=1}^{\mathrm{m}}(y_{pj} - o_{pj})^2 \tag{5-2}$$

作为第 p 个样本的误差测度。整个样本的误差为

$$E = \sum E_p \tag{5-3}$$

首先假设输入层、中间层和输出层的单元数分别是N、L和M。$X = (x0, x1,,, \ ,xN)$是加到网络的输入矢量，$H = (h0, h1,,, \ ,hL)$是中间层输出矢量，$Y = (\mathrm{y}0, y1,,, \ ,yM)$是网络的实际输出矢量，并且用$D = (\mathrm{d}0, d1,,, \ ,dM)$来表示训练组中各模式的目标输出矢量输出单元$i$到隐单元$j$的权值是$V_{ij}$，而隐单元$j$到输出单元$k$的权值是$W_{jk}$。另外用$\theta_k$和$\varphi_j$来分别表示输出单元和隐含单元的阈值。

于是，中间层各个单元输出为：

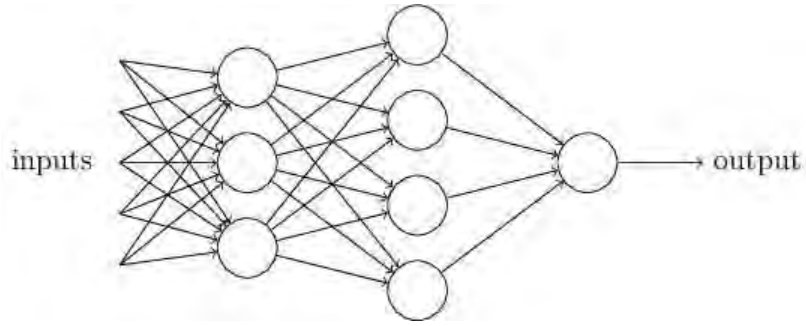
$$h_j = f(\sum_{i=0}^{N-1} V_{ij}x_i + \phi_j) \tag{5-4}$$

输出层各个单元输出为：

$$y_k = f(\sum_{j=0}^{L-1} W_{\mathrm{u}}h_j + \theta_k) \tag{5-5}$$

激励函数表示为：

$$f(x) = \frac{1}{1 + e^{-kx}} \tag{5-6}$$



卷积神经网络工作原理图

VGG16 作为经典卷积神经网络之一，探索了卷积神经网络层数对算法性能的提升程度，整个网络结构简洁，反复使用相同参数的 3×3 卷积核和 2×2 池化层进行堆叠，构筑了 16 层的网络，使用非常多的 3×3 卷积串联达到一个较大卷积核的效果，保持相同感受野的同时能够减少参数量，达到不错的性能。所以本文以 VGG16 卷积神经网络建立深度学习模型。建立过程如下：

如图所示，VGG16 总共有 16 层，其中包含 13 个卷积层和 3 个全连接层，有 6 个块结构，每个块结构中的通道数相同， 前五个卷积块（block1~5）通道数分别为 64、128、512、512、512，更多的信息随着通道数的增加被提取出来，输入图片调整为（224×224×3）进入网络，卷积层全部都是 3*3 的卷积核，其主要网络结构如图所示，概括如下：

1、block1：输入图像尺寸 224×224×3，经两次 64 个通道的 3×3 的卷积核卷积，再经 2X2 最大池化，输出 net 为(112,112,64)

2、block2：经两次 128 个通道 3×3 卷积网络，再 2×2 最大池化，输出 net 为(56,56,128)。

3、block3：经三次 256 个通道 3×3 卷积网络，再 2×2 最大池化，输出 net 为(28,28,256)。

4、block4：经三次 512 个通道 3×3 卷积网络，再 2×2 最大池化，输出 net 为(14,14,512)。

5、block5：经三次 512 个通道 3×3 卷积网络，再 2×2 最大池化，输出 net 为(7,7,512)。

6、最后做 3 层全连接层，前两个全连接层的神经元个数为 4096，最后的全连接层是 1000 个神经元，然后输出的就是每个类的预测。[8]

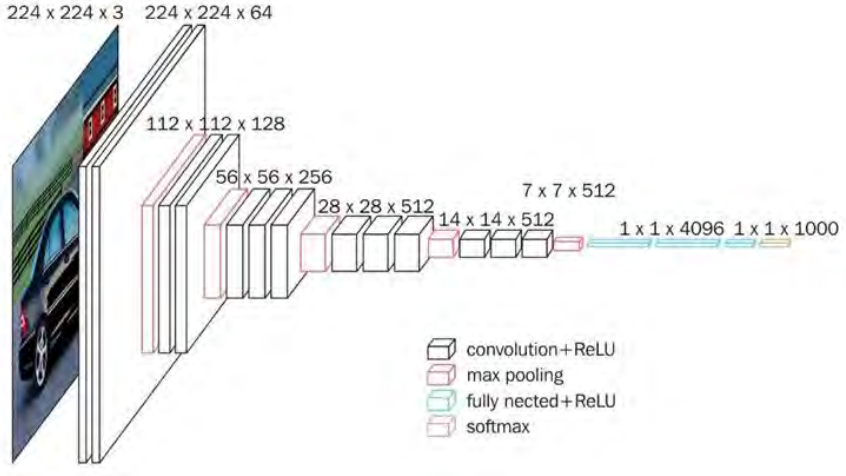


图 5.2.3 VGG16 卷积神经网络工作原理

本文在 VGG16 特征提取网络基础上对全连接层做出修改，原网络全连接层使用 4096 会造成算法复杂度与参数量的增加，而且不容易收敛，有限训练次数精度提升不明显，本文此处需要对三类图像进行分类，所需参数较少，因此将两个全连接层 4096 修改为 256 和 128，参数量得到减少，速度和精度都得到提升。同时，采用迁移学习的思想，已经训练好的大型网络的主干特征提取部分经过大量数据训练因而具有较好的鲁棒性和特征提取效果，因此我们直接使用 VGG16 特征提取网络的权重和参数，训练时对前 5 层卷积特征提取部分不训练，只训练全连接部分，具体步骤如下：

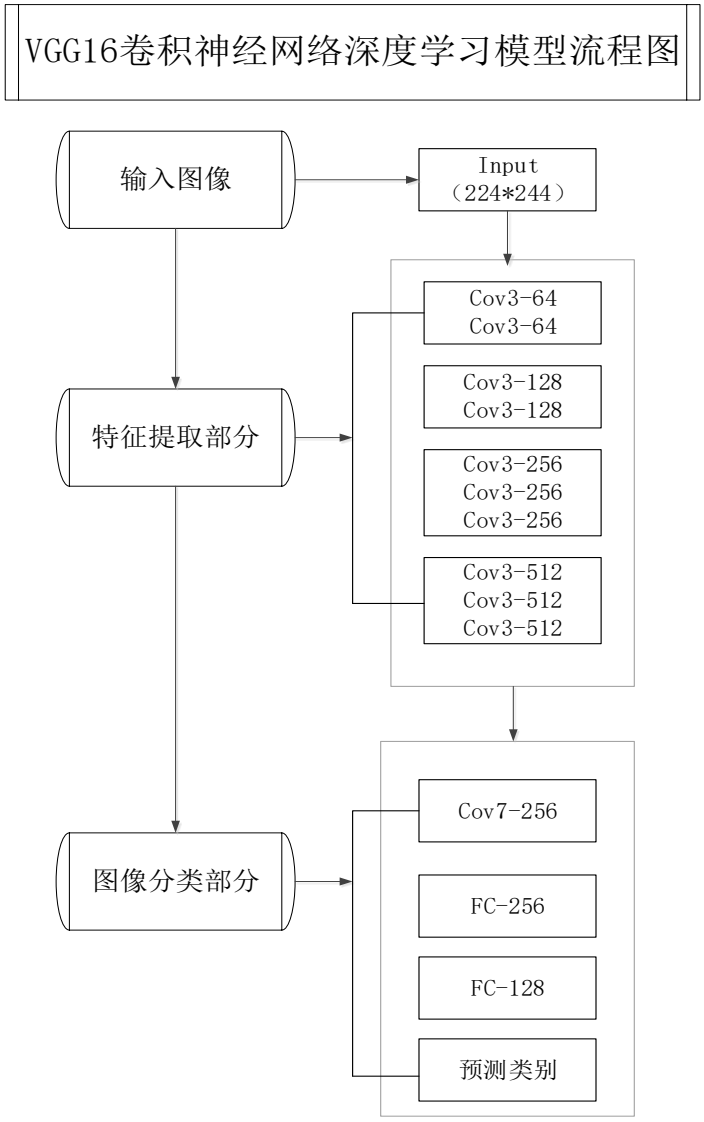


图 5.2.4 VGG16 卷积神经网络深度学习模型流程图

5.2.4 训练结果及精度评价

设置 epoch 为 50 进行训练，通过图 5.2.5 可以看出训练集跟验证集 loss 整体呈下降趋势，当 epoch 训练到 30 附近时，训练集损失不断下降，而测试集数据出现振荡趋于不变，说明模型在 epoch 训练到 30 之后，模型训练出现过拟合现象，对比训练集测试集精度图也可以验证这一点， 在 epoch 设置为 100 的训练图中，这一现象会更加明显。在 epoch 训练到 30 之后测试集精度趋于不变，而训练集精度仍在上升，出现该现象的原因可能与各类样本数量不均衡有关，MOR 值在 50 以下占样本集数量过大，与其它种类样本集数量存在较大差异，同时样本数据中个别图像存在的噪声严重、图像质量差等问题也是造成该现象的原因之一。

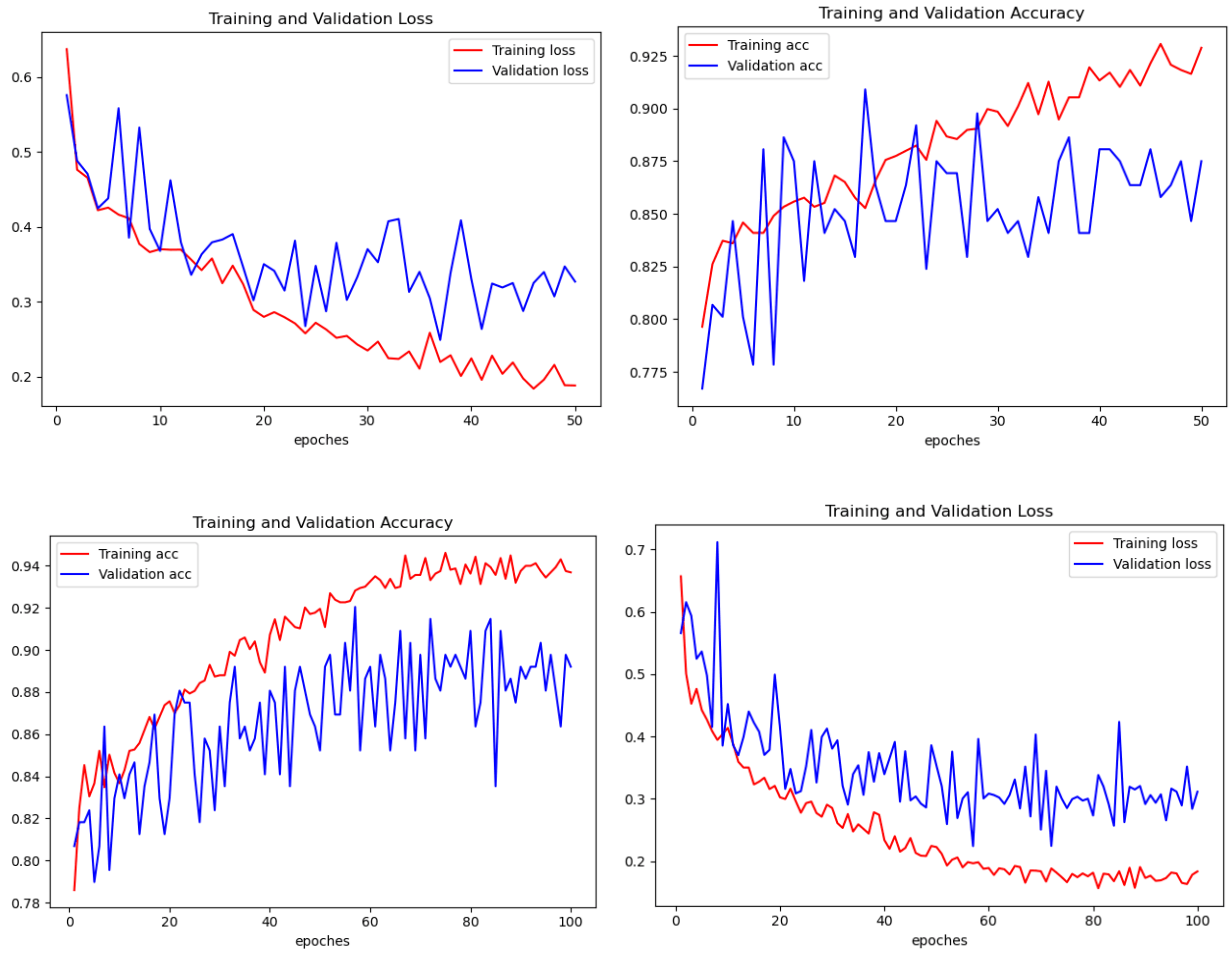


图 5.2.5 损失函数与精度图

为了防止该问题的出现，我们选取 epoch 为 30 时的模型训练权重作为模型训练结果，对测试集数据进行预测，其混淆矩阵如下图所示，预测精度为 88.33%，同时我们可以发现模型能够较好的区分能见度很好与能见度不良两种类别，因为这两类与其他类别从图像上观察还是有很大区分度的，比较容易辨别，但是能见度中等与良好两种类别容易出现错分现象，正是两者辨识度小而且训练集较少造成这种原因，误分最严重的是能见度良好类别，其样本集数量在四个类被中找的比重最小。

表 5.2.1 混淆矩阵表

实际 / 预测	能见度不良	能见度中等	能见度良好	能见度很好
能见度不良	40	1	0	0
能见度中等	1	7	0	0
能见度良好	0	5	1	0
能见度很好	0	0	0	5

6　问题三

6.1　问题分析与描述

在建立算法之前,首先要面对的问题就是，在不依赖观测数据的情况下，什么样的算法估计出的结果是比较好？通过分析单一视频帧估计能见度，得出两种思路：一种是在不同能见度的情况下，当前视频帧中可视物体的最大景深会发生变化，可用最大景深来估计当前视频帧中的能见度。另一种思路是，若一段视频中有可参照的位置固定的物体，可利用同个物体在不同视频帧中的亮度（反射率）来估计能见度。

分析本题提供的高速公路视频，视频时段交通流稀少，100 幅视频帧中，只有一幅有车辆出现，同时，该时段能见度相对较低，视野范围内也无树木，告示牌等参照物，故无法用行驶中的车辆或树木等来来估计能见度。在视频帧的近景区域，道路车道线显示清晰，且车道线的明显程度随能见度的变化出现肉眼可见的变化，因此，可选择道路车道线作为参照物。

第一步：利用 Canny 边缘检测算法检验出车道线 。

第二步：利用 Hough 变换提取车道线，并估计该视频路段的车道线消失点。

第三步：通过车道线被检测到的最远景深定性分析能见度的变化。结合相机模型，将图像坐标转换为真实世界的平面坐标，估计实际能见度，绘制能见度随时间的变化曲线。

具体算法的流程图如下所示：

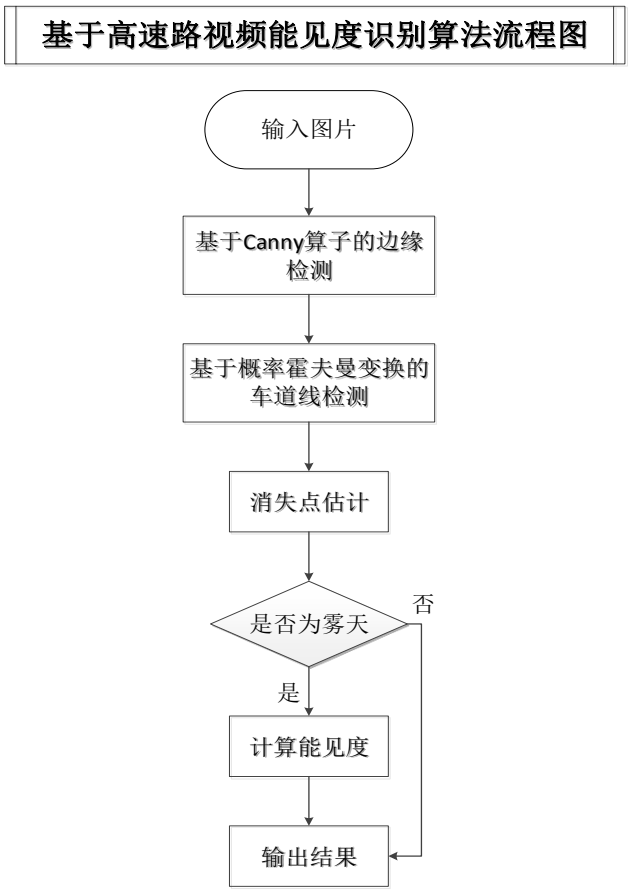


图 6.1.1 基于高速路视频能见度识别算法流程图

6.2 问题求解与结果评价

依照上述分析，将问题分为三步求解：

6.2.1 Canny 算子边缘检测

常用的边缘检测算子有 Sobel、Canny、Prewitt 等。本文采用 Canny 算子，它是于 1986 年由 JohnF.Canny 提出，具有低误码率、高定位精度和抑制虚假边缘等优点。原始彩色图像信息量大，处理速度慢， 所以首先进行灰度化。对灰度图进行边缘检测，能够提取图像中的灰度级突变、纹理结构变化、色彩变换等信息[1]。首先利用 Canny 算子边缘检测检验出图像中的车道线，此算法的原理及具体步骤如下

1. 噪声消除。因为边缘检测对图像中的噪声非常敏感，所以该算法第一步为使用一个 5X5 的高斯滤波器来做噪声移除。滤波器函数表达式为:

$$G\left(x\right) = \frac{1}{2\pi\sigma^2} cxp(\frac{-x^2}{2\sigma^2}) \qquad (6\text{-}1)$$

2. 寻找图像的强度梯度。该步骤使用已经平滑过得图像，使用 sobel kernel 在垂直和水平方向同时做滤波，获取垂直方向和水平方向上的一阶导数。从中本文可以得到每一个像素上的边缘梯度和方向。梯度的方向总是与边缘垂直，需将其归为水平竖直或者对角线方向

$$Edge\_Gradient(G) = \sqrt{G^2_x + G^2_y} \qquad (6\text{-}2)$$

$$Angle(\theta) = \tan^{-1}(\frac{G_y}{G_x}) \qquad (6\text{-}3)$$

3. 非最大抑制。在获取梯度大小和方向后，将其图像进行全面扫描，将不是边缘组成部分的像素移除。因此，扫描每个像素时，需检验该像素是否为梯度方向上局部极大值。点 a 在垂直方向的边缘上，梯度方向与边缘垂直。点 b 和点 c 也在梯度方向，所以点 a 需要与点 b 和点 c 一起检验，验证是否组成了一个局部极大值。如果存在局部极大值，则执行下一步，否则，终止执行。
4. 该步骤确定哪些边缘是真正的边缘。为此，需要设定两个阈值，一个较大的值和一个较小的值。本文规定，强度梯度大于较大的阈值的任何边缘必定是本文预期的边缘，而小于较小阈值的边缘，本文认为其不是预期边缘，不予考虑。在两个阈值之间的对象再根据其连通性再进一步划分为边缘或非边缘。如果它与“确认边缘”连通，将它们归类为边缘，否则也不予考虑。[3]

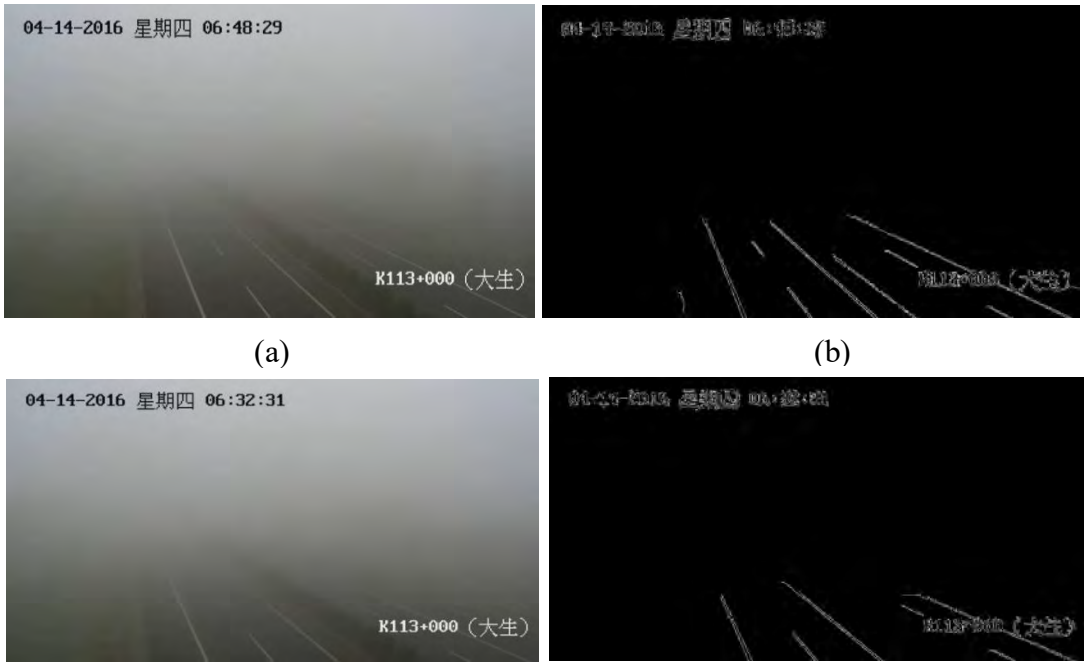
在边缘检测之前，为提高图像的处理速度，需对原始的 RGB 图像做灰度化处理，即将彩色空间的图像转换为灰度图像。 一般情形下，灰度化公式采取：

$$Gray = 0.299 \times R + 0.587 \times G + 0.144 \times B$$

其中，R,G,B 分别为红、绿、蓝三色通道的值，取值范围为［0，255］。但以上式灰度化的图像会丢失 大量色彩信息，为增强道路标识信息，又考虑到车道线 一般为白色(255，255，255)和黄色(255，255，0)，故本次图像灰度化采取为:

$$Gray = 0.5 \times R + 0.5 \times B$$

然后利用上述原理，对图像进行 Canny 边缘检测，即对车道线进行检测，检测结果如下图：



(a) (b)

(c) (d)

图 6.2.1 车道线检验图

观察两组对照图 a 和图 c，在 06:48:29 时刻，比 06:32:31 时刻能见度更高，在 Canny 边缘检测结果图 b 和图 d 中，对比则更加明显，在图 b 中，不管是近侧还是远侧，检测到的道路边缘，其车道线都更长，这符合原图像中的能见度差异。

6.2.2 概率 Hough 变换

经典 Hough 变换的实质是对图象进行坐标变换, 使变换的结果更易于识别和检测。最典型的例子就是将直线方程由直角坐标转换为极坐标形式, 转换后的结果:直角坐标系下的一个点在极坐标下成为一条曲线,及时图形有破损或扭曲，仍可以可以使用 Hough 变换检测出来。经典 Hough 变换主要针对直线的检测, 它将画面上所有前景点进行相同的变换, 将变换平面上各点看作一个个累加器 (点与点的距离取决于变换所取步长的大小) , 原图象平面上各点变换后所得曲线在变换平面上每经过的某点, 便将该点计数器值加 1, 最后结果是各曲线的交点处计数器值达到最大, 该点坐标值既直线的参数。借助各种几何图形的特点, 便可正确识别它们。[2]

Hough 变换是检测图形的常用方法。如果可以将图形用数学表达式表示出来。一条线可以用$y = mx + c$的形式，或者使用圆心坐标系。因此，如果线在原点下方，其ρ为整数，且角度小于 180.如果它在原点的上方，其角度小于 180，将ρ取为负，任何垂直线记为 0 度，水平线为 90 度。其中ρ的公式描述为

$$\rho = x\cos\theta + y\sin\theta \qquad (6\text{-}4)$$

对于线条，任何行都可以用两个量来表示：ρ和θ，因此有限需要一个容器或计数器分别用来统计这两个量，其初始值记为 0。在上述的 Hough 变换中，即使对于带有两个参

数的行，依然需要大量的计算。概率 Hough 变换是对 Hough 变换的优化。在计算时，不考虑所有的点，而是只考虑点的随机子集，就可以将线检测出来，但在这个过程中需要降低阈值。用行来表示ρ，列来表示θ，容器的大小取决于需要的精度。例如，精度需求为 1 时，容器需要有 180 个列。对于θ，最大距离为图像的对角线距离，行数最大值可以设置为对角线的长度取整。[4]

根据上述原理建立算法，利用 Hough 变换对检测得到的车道线提取，提取结果如下：

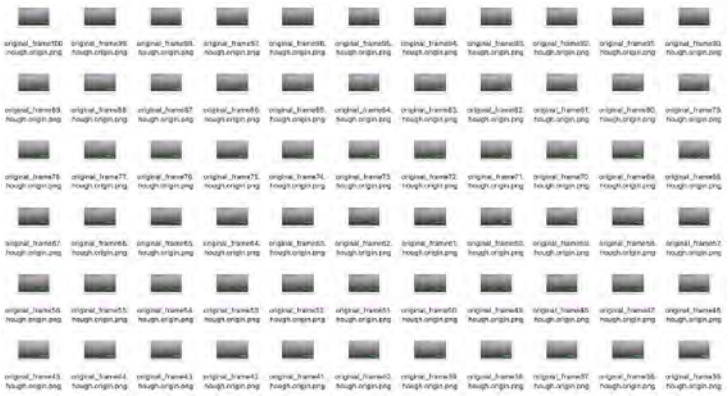


图 6.2.2 车道线提取图

根据对该时间段视频数据的分析和观察，车道线信息主要位于图像的下半部分部，上半部为比较均匀的雾气及时间水印等无效信息。所以在进行概率 Hough 变换时，只选取包含车道线的部分进行处理。这样一方面减少了信息的量，提高运算速度，另一方面，划定区域减少，有效减少了分辨率有限错将雾气垂直灰度方向发生变化视为直线的情况，提高了检测精度。此外，在该视频数据中，车道线在图像中呈稳定分布，在在图像批处理中，计算变换时限定了ρ和θ值，其中，θ的取值范围为:$40 \leq \rho \leq 120$，ρ的范围为：

$$-\omega 2+(2*\mathrm{h}3)2 \leq \rho \leq \omega 2+(2*\mathrm{h}3)2$$

其中ω，h 分别为图像的宽度。[5]

随着能见度的变化，检测得到的车道线长度也会相应的变化，能见度越低，视野越模糊，检测到的车道线越短，能见度越高，视野越清晰，检测到的车道线越长。

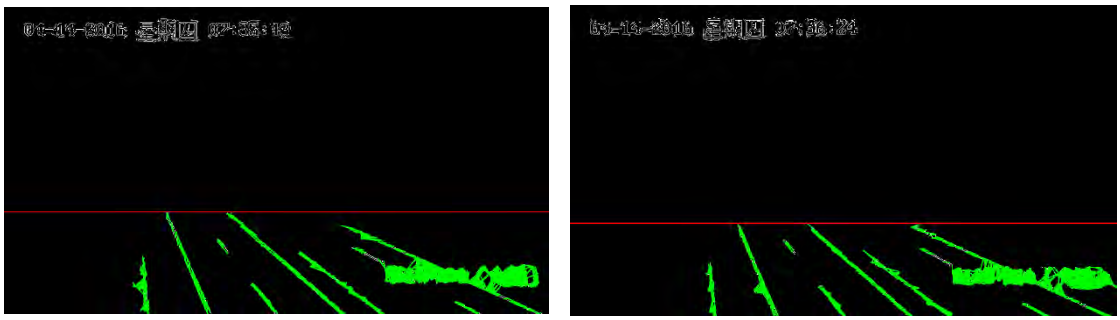


图 6.2.3 不同能见度下检测的车道长度变化图

根据检测到的车道线对消失点进行估计。在假设该视频路段无较大的海拔起伏的情况下，车道线消失点，可近似转化为车道线延长线的交点。如图 6.2.3，根据车道线延长线，估计出消失点的位置。

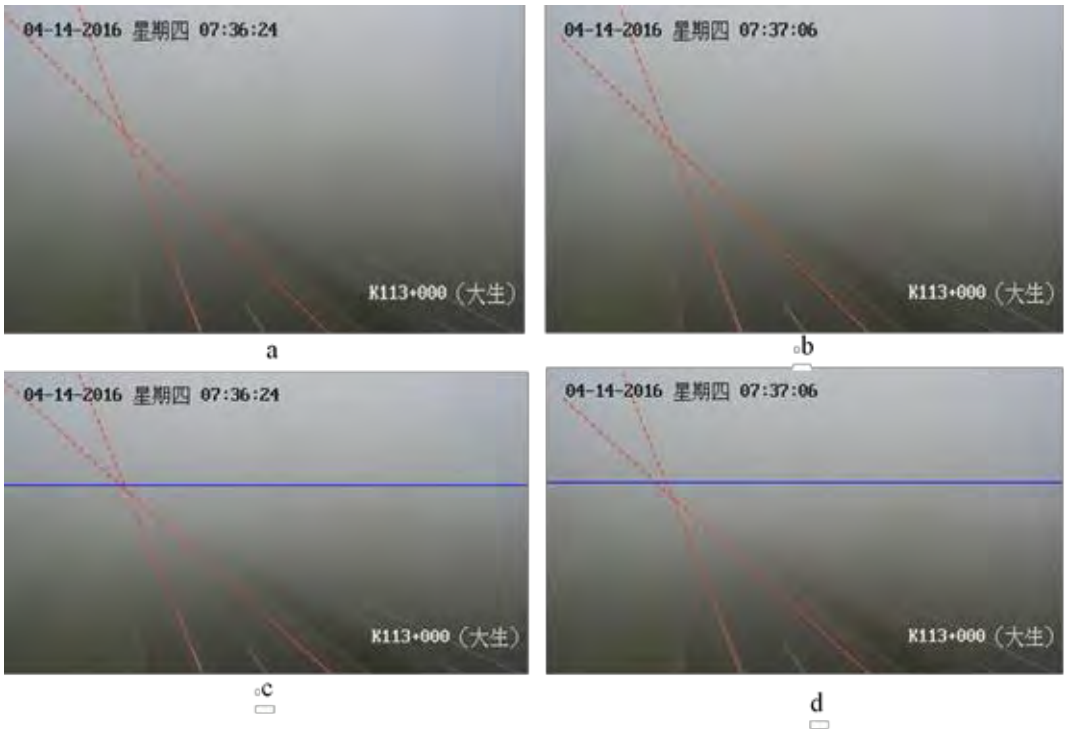


图 6.2.4　消失点估计图

由于视频数据中能见度很低，无样本数据可以提取出道路与天空的边界。因此，本问题中，本文将消失点，作为晴天状况下的可视点。过消失点做与 u 轴水平的延长线，以此替代道路与天空的边界。

因为视频数据的时间分辨率为一分钟左右，即使取连续的三帧图像，其灰度变化的拐点值也不相同，因此无法通过滑动平均来计算灰度变化的拐点值。因此本文取车道线在 v 轴最小值作为灰度变化拐点值的替代。

在边缘检测结果的基础上，使用概率 Hough 变换提取道路线，并与原图像进行叠加，结果有：

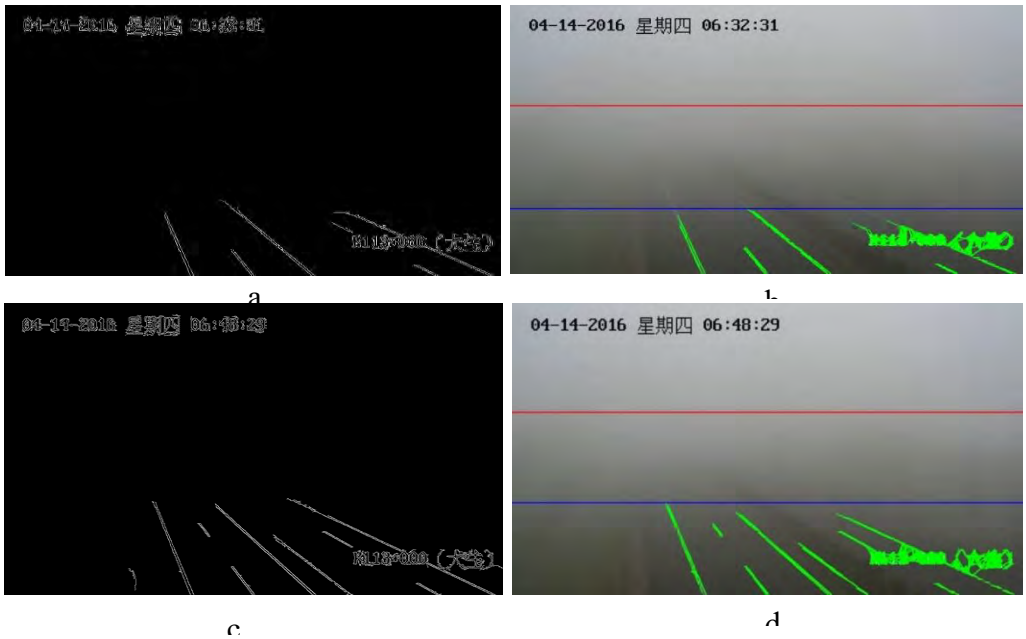


图 6.2.5 拐点值与消失点边界线图

图中，蓝色线为灰度变化拐点值，红色线为由消失点得出的道路与天空的边界的替代，观察 b，d 两图得出以下结论：灰度变化拐点值，随着能见度的提升而向图像上方移动，反之则下图像下方移动。

6.2.3 能见度随时间变化曲线

首先根据相机模型在其环境中建模，建立灰度变化拐点值最大景深线与能见度的定量关系。

相机模型具体可以根据下图描述，它位于（S，X，Y，Z）坐标系中相对于场景的 H 高度处。它的固有参数是焦距 f 和像素大小 t。θ是相机光轴与水平方向之间的夹角。 在图像坐标系内，（u，v）表示像素的位置，（μ_0，ν_0）表示光学中心 C 的位置，而ν_h是水平线的垂直位置。[6]

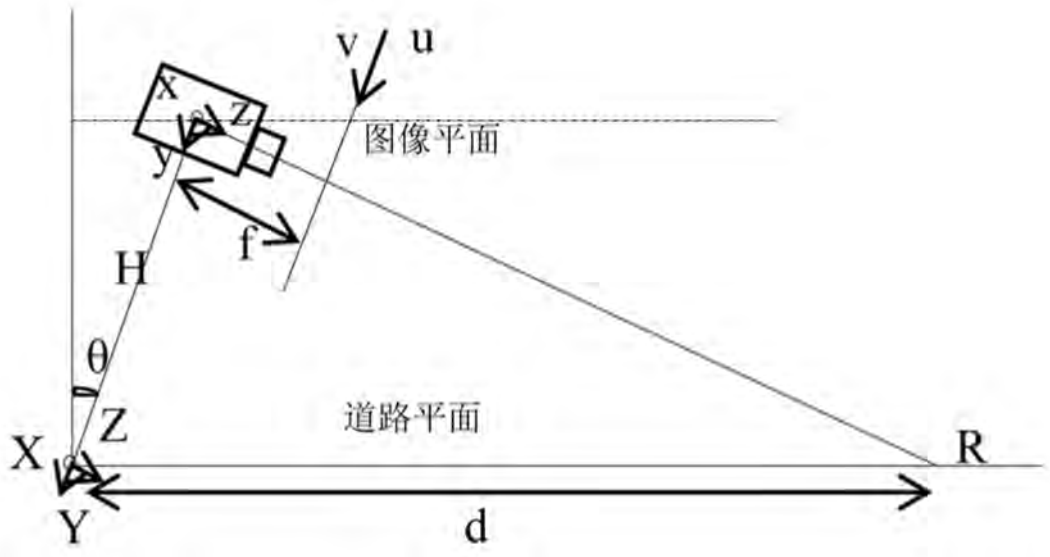


图 6.2.6 相机模型原理图

通过将相机模型应用于在本问题中，将相机参考系统内具有三维坐标（x，y，z）的点投影到图像平面上有以下表达式：

$$\begin{cases} \mu = \mu_0 + \alpha\frac{x}{z} \\ \nu = \nu_0 + \alpha\frac{y}{z} \end{cases} \tag{6-5}$$

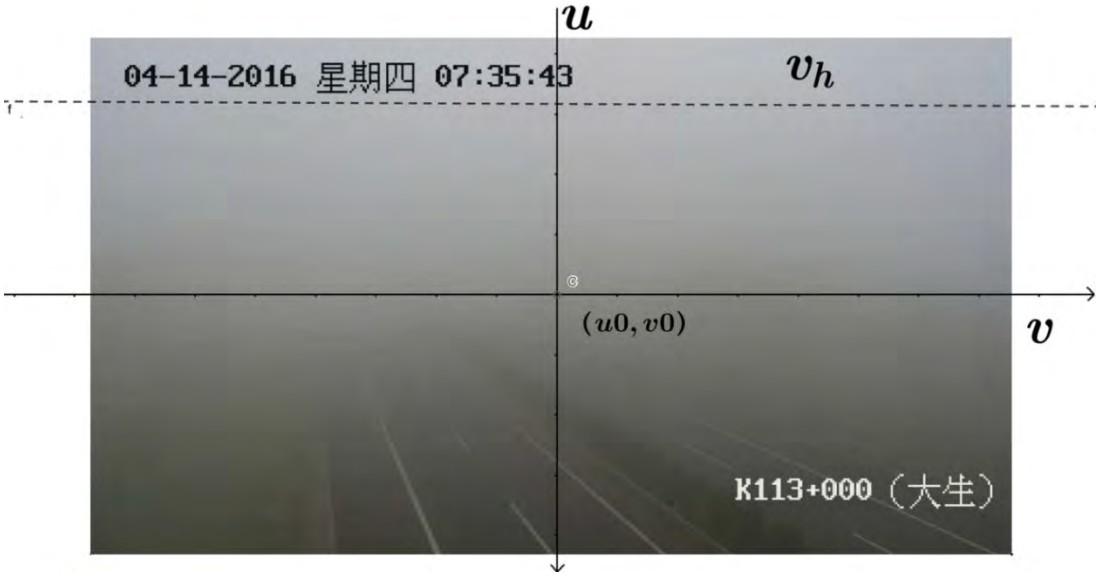


图 6.2.7 相机模型与图像结合图

根据相机模型示意图，穿过光学中心的水平线与 z 轴形成θ夹角，投影到图像平面上，该水平线满足如下表达式：

$$v_h = v_0 - \alpha \tan(\theta) \tag{6-6}$$

化简公式有：

$$\frac{v - v_h}{\alpha} = \frac{y}{z} + \tan(\theta) \tag{6-7}$$

将等式与相机的（X，Y，Z）空间结合起来，可推导出

$$\frac{v - v_h}{\alpha} = \frac{Y + H}{z} + \tan(\theta) \tag{6-8}$$

进而得到，在真实世界空间中的距离相机坐标系中原点距离为 d 的点，满足：

$$X, Y, Z = \begin{pmatrix} X \\ -d\sin(\theta) \\ d\cos(\theta) \end{pmatrix} \tag{6-9}$$

化简得：

$$\frac{v - v_h}{\alpha} = \frac{H}{\mathrm{d}\cos(\theta)} \tag{6-10}$$

最终得到距离 d 的表达式为

$$d = \begin{cases} \dfrac{\lambda}{v - v_h} & if \quad v > v_h \\ v = v_0 + \alpha^{\frac{y}{z}} & if \quad v > v_h \end{cases} \qquad where \quad \lambda = \frac{H\alpha}{\cos(\theta)} \tag{6-11}$$

其中，H、α 和θ为相机模型位置及相机参数相关，在此假定为常数。可知v与v_h的差值与d成反比，又因为可以把v_h作为一定值，故可认为，最大景深距离与可视距离成反

比。[7]

且上式中只有是λ未知的，因此只需估计相机参数就可以进一步估算出能见度的真实值。如果我们知道两点d_1，d_2间的实际距离以及图像平面中v_1和v_2间的真实距离，即可使用图中的相机模型，依据距离与纵轴坐标值关系公式对相机参数进行估算，如下式：

$$\lambda = \frac{d_1 - d_2}{\left(\frac{1}{v_1 - v_h} - \frac{1}{v_2 - v_h}\right)}$$

查阅资料得知，高速公路应急避险道标准长度为 3.5m。

图 6.2.8 原理坐标图

左侧应急避险道靠近摄像头，几乎没有因透视原因发生形变。设 GH 间距离与应急避险车道宽度相等，则 GH 间距离为 3.5m，又图像上两点 G，H 在图像平面上坐标分别为 G(331,629),H(331,720)，将数据代入上面公式，得到$\lambda = 15540$。

通过对 100 张图像进行处理，得出各幅图像的近似最大景深线。取最大景深线的 Y 值作为标识。由于图形学 Y 值自上向下递增，而能见度变化与之相反，为了更好地显示能见度变化的趋势，在此将图像高度 H 与 Y 值相见，得到趋势相反的趋势线，如下图所示。

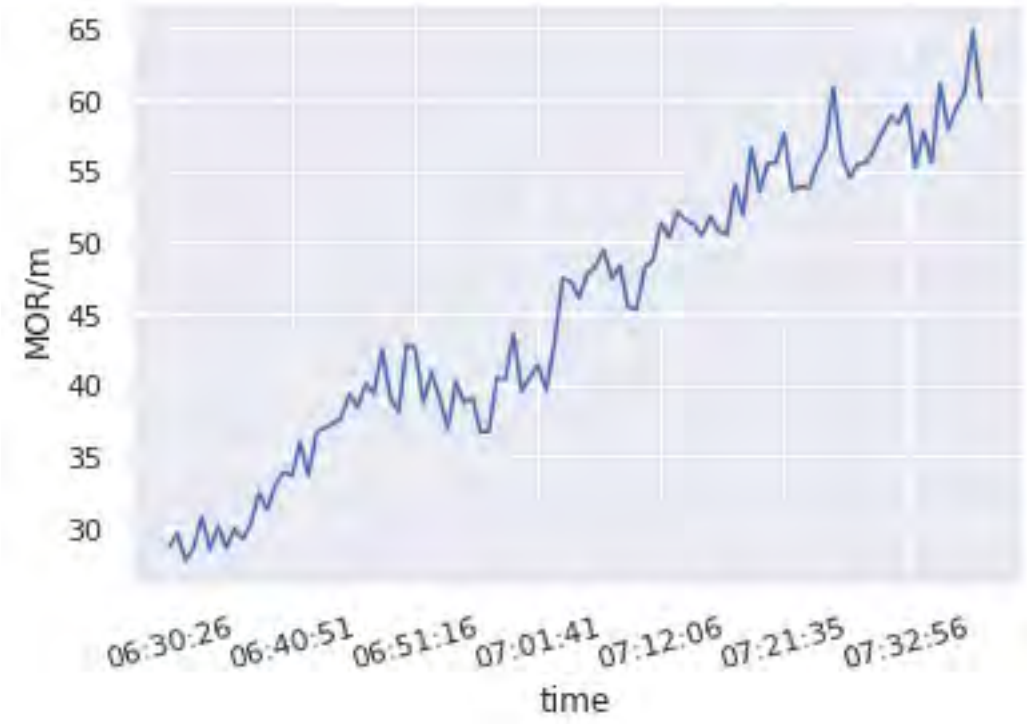


图 6.2.8 能见度随时间变化图

由上图可知，随着时间 0（2016-04-14 06:30:26）至时间 100（2016-04-14 07:39:11）过程中，能见度时间 26（2016-04-14 06:47:47）时有所提高，达到区域最大值 0.0045，在时间 39（2016-04-14 06:50:49）时有区域最小值 0.0024，其后，能见度出现逐渐增加的趋势。

7 问题四

7.1 问题描述与分析

根据问题三的能见度随时间变化曲线图，建立线性回归模型来预测大雾的变化趋势，并根据指定的能见度来定义得到大雾消散的时间，也就是找到具体的时间点能达到某个数值。

第一步：建立线性回归模型。

第二步：建立灰度预测模型。

第三步：分析两种模型预测到的变化趋势，确定消散时间。

7.2 模型建立与问题求解

7.2.1 建立线性回归模型模型

关于模型的原理详情见问题一。

在问题三中，得到了该时间段内能见度的变化曲线。又分析原始数据得出，视频帧在时间上的的分布是不均匀的，相互之间时间间隔各不相同。在变化曲线的表示中，可以忽略其影响，但在建立模型定量分析能见度变化时，这种不均匀的分布会影响模型的精度。故第一步需要对每帧数据做时间的标注。

本文借助 OCR 手段提取了图片中的时间，因为图片摄于同一天，所以只需提取时间的“时分秒”即可。为了提高精度，首先对视频帧做裁剪。裁剪后的数据如下图所示。

图 7.2.1 图上时间裁剪

将视频帧中日期数据被整理到同一个表中，经检查，有如下脏数据。

表 7.2.1 脏数据剔除表

识别字段	所在文件	真实时间
7:06:38	pic/original_frame56.bmp	7:08:38
0:17:39	pic/original_frame69.bmp	7:17:39
0242:351	pic/original_frame76.bmp	7:22:31
7:25:12	pic/original_frame77.bmp	7:23:12
7:26:04	pic/original_frame84.bmp	7:28:04
7:26:45	pic/original_frame85.bmp	7:26:45

因为时间为字符串，所以将其转化为时间戳的形式，既能作为 x 轴连续数据，又能表现准确表达时间变化量。

整理数据后得出，最早时间时间戳为 1460586625.0。为减小自变量的取值范围，将所有时间减去 1460586624.0，将其作为最终自变量。

建立时间和能见度的线性回归方程模型，得到两者的线性关系式为：

$$Y = 0.00793594\mathrm{t} + 29.1509186916548$$

由上式分析可以看出能见度与提高的趋势。当 MOR=150 时，计算得 x 约为 15247.0 转换为真实时间戳为 1,460,601,871，换算为时间为 2016-04-14 10-44-31。

7.2.2 建立灰度预测模型

本文利用 GM（1，1）建立灰度预测模型，其原理如下：

设时间序列 $X^{(0)}$ 有 n 个观察值， $X^{(0)} = \left\{X^{(0)}(1), X^{(0)}(2),..., X^{(0)}(n)\right\}$，通过累加生成新序列 $X^{(1)} = \left\{X^{(1)}(1), X^{(1)}(2),..., X^{(1)}(n)\right\}$， 则 GM（1，1）模型相应的微分方程为：

$$\frac{dX^{(1)}}{dt} + aX^{(1)} = \mu \tag{7-1}$$

其中：α 称为发展灰数；μ 称为内生控制灰数。

设 $\hat{\alpha}$ 为待估参数向量 $\hat{\alpha} = \begin{pmatrix} a \\ \mu \end{pmatrix}$，可利用最小二乘法求解。解得：

$$\hat{\alpha} = \left(B^T B\right)^{-1} B^T Y_n \tag{7-2}$$

求解微分方程，即可得预测模型：

$$\hat{X}^{(1)}(k+1) = \left[X^{(0)}(1) - \frac{\mu}{a} \right] e^{-ak} + \frac{\mu}{a} \quad , \quad k = 0,1,2...,n \tag{7-3}$$

模型检验，灰色预测检验一般有残差检验、关联度检验和后验差检验。

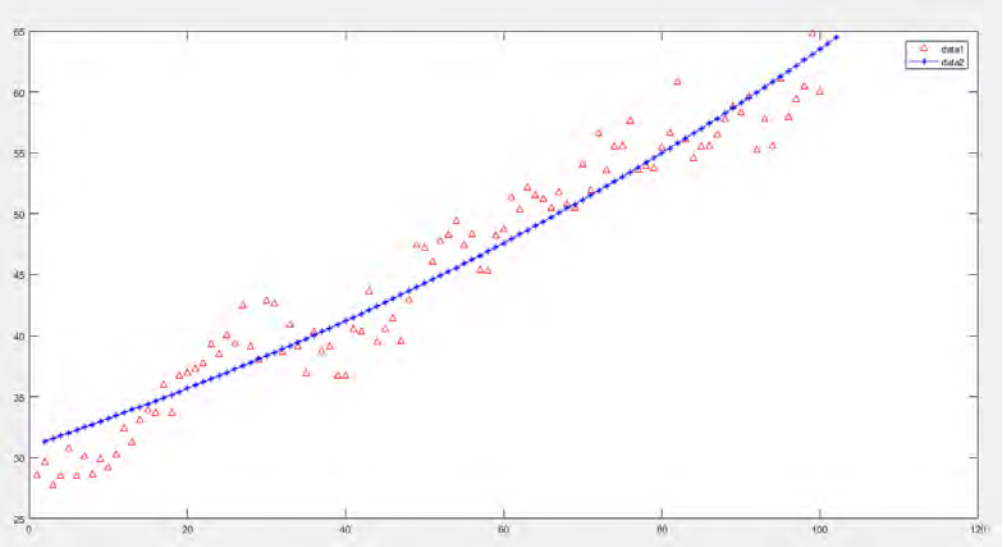


图 7-2 灰度模型预测图

逐步增加预测的步骤数，发现当增加至 200 时，MOR 的预测值达到 150，由分钟与步数比例为 0.69:1,可知，在 2 小时 30 分之后大雾散去。

7.2.3 结果分析

两种模型都能够比较准确的预测大雾的变化趋势。对两种预测模型结果对比分析，可以总结出，随着时间的推移，大雾呈现减弱的趋势，并且两种预测模型估计达到 150 米的 MOR 能见度的时间偏差在 30 分钟左右。

8 总结与评价

本文基于题目给出的地面观测数据与视频数据，利用多元线性回归模型、VGG 卷积神经网络深度学习模型、相机模型、模型，分析了地面观测数据与能见度的关系，并建立了关系式，定量直观的分析了两者之间的关系；建立深度学习模型，基于视频数据对能见度进行估计并进行精度评价；利用视频数据，以景深为依据，建立了一套不依赖观测数据的能见度估计算法，并绘制了能见度随时间的变化曲线。

针对问题一，经过相关性分析与降维分析，得到与 RVR 能见度相关性较大的 3 项温度与湿度指标，2 项风速指标，加上灯光数据的影响，以此建立多元线性回归模型，反映这些指标的关系。由线性回归图并结合实际可以看出，建立的模型精度较高，可以很好的反映特征指标之间的关系。

针对问题二，建立了 VGG 卷积神经网络深度学习模型，将视频数据分成三类训练样本导入模型，经过分析与训练，得到精度为

针对问题三，使用 Canny 边缘检测算子和 Hough 变换形成了一套能见度估计算法，该算法可以不参考观测数据，只考虑监控视频数据。然后绘制了能见度随时间变化曲线，可以看出，随着时间的推移，能见度呈总体上升的趋势。

针对问题四，经线性回归和灰度预测两种模型预测，可以看出随着时间的推移，两种模型预测得到的大雾都是慢慢减弱的趋势，

本文也存在着一些问题，首先是训练样本没有按照标准的能见度等级来划分，仅将能见度粗略的划分成了三类，所估计的能见度也只能按照三种等级划分，不够全面。其次，

所建立的视频数据能见度估计算法只针对图像中有线性参照物的情况，不具有普适性。

参考文献

[1] 朱舞雪, 宋春林. 基于视频的雾天驾驶场景及其能见度识别算法研究[J]. 图像与信号处理, 2015, 004(003):P.67-77.
[2] 杨治明,周齐国.基于 Hough 变换理论的图形识别[J].重庆工业高等专科学校学报,2002(04):16-17+20.
[3] 周晓明, 马秋禾, 肖蓉. 一种改进的 Canny 算子边缘检测算法[J]. 测绘工程, 2008, 17(1):28-31.
[4] 辛超, 刘扬. 基于概率 Hough 变换的车道线识别算法[J]. 测绘通报, 2019(S2).
[5] 邱东, 翁蒙, 杨宏韬. 基于改进概率 Hough 变换的车道线快速检测方法[J]. 计算机技术与发展, 2020, 030(005):43-48.
[6] 汪涛. 基于频谱分析和改进 Inception 的雾霾能见度检测研究[D]. 2019.
[7] 王金冕. 基于监控视频分析的高速公路能见度检测与预警系统研究[D].
[8] Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer ence, 2014.

附录

以下是用于数据处理与模型训练的 Python 代码。

```
# 高速视频截图处理代码

import cv2
import numpy as np
import pytesseract
from matplotlib import pyplot as plt
ans = []
times= []
paths = []
for i in range(1,101):
    img = cv2.imread(f'pic/original_frame{i}.bmp')
    paths.append(f'pic/original_frame{i}.bmp')
    img_cv = img[20:80, 461:720]
    cv2.imwrite(f'fog_head/original_frame{i}.clip.bmp',img_cv)

    # By default OpenCV stores images in BGR format and since pytesseract
assumes RGB format,
    # we need to convert from BGR to RGB format/mode:
    img_rgb = cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB)
```

```
    res = pytesseract.image_to_string(img_rgb, config='--psm 12 -c
tessedit_char_whitelist=0123456789:').strip()
    times.append(res)
    print(res)



    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray,25,50)
    #
    # # lines = cv2.HoughLines(edges)
    lines =
cv2.HoughLinesP(edges,1,np.pi/180,1,minLineLength=10,maxLineGap=20)
    # # lines = cv2.HoughLinesP(edges,1,np.pi/180,100)
    # # print(lines)
    # # print(lines)
    # # print(img.shape)
    h,w,_ = img.shape
    th = h * 0.5
    ymin = float("inf")
    for line in lines:
        x1 = line[0][0]
        y1 = line[0][1]
        x2 = line[0][2]
        y2 = line[0][3]
        if y1<th or y2<th:
            continue
        tmp_ymin = min(y1,y2)
        if tmp_ymin<ymin:
            ymin = tmp_ymin
        cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
    t = int(i - 50)
    ans.append(15540)/(ymin-269)

    # cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)
    cv2.line(img,(0,269),(w,269),(0,0,255),2)
    cv2.line(img,(0,ymin),(w,ymin),(255,0,0),2)
    cv2.imwrite(f'pic/original_frame{i}.hough.origin.png',img)
print(img.shape)
from matplotlib import pyplot as plt
import pandas as pd
s1 = pd.Series(ans)
s2 = pd.Series(times)
s3 = pd.Series(paths)
```

```
df2 = pd.concat([s1,s2,s3], axis=1)
df2.to_csv("with_time.vis.csv")
df = pd.DataFrame(ans)
df.to_csv("pingjia-fog.csv")
# print(df)
a = df.plot()
a.set_ylabel("distance (λ/Δv)")
a.set_xlabel("time")
a.legend(['distance (λ/Δv)'])
plt.show()
```

#边缘检测代码

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# img = cv2.imread(f'pic/original_frame30.bmp',0)
# edges = cv2.Canny(img,7,50)
# plt.subplot(121), plt.imshow(img, cmap='gray')
# plt.title('Original Image'), plt.xticks([]), plt.yticks([])
# plt.subplot(122), plt.imshow(edges, cmap='gray')
# plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
# plt.show()


for i in range(1,101):
    img = cv2.imread(f'pic/original_frame{i}.bmp',0)
    # kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]], np.float32)  #
锐化
    # edges = cv2.filter2D(img, -1, kernel=kernel)
    blur_img = cv2.GaussianBlur(img, (0, 0), 5)
    img = cv2.addWeighted(img, 1.5, blur_img, -0.5, 0)
    edges = cv2.Canny(img,15,70)
    # l.append((img,edges))
    cv2.imwrite(f'pic/original_frame{i}.canny.png', edges)
    plt.subplot(121), plt.imshow(img, cmap='gray')
    plt.title('Original Image'), plt.xticks([]), plt.yticks([])
```

```
    plt.subplot(122), plt.imshow(edges, cmap='gray')
    plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
    plt.show()

# for img,edge in l:
#     print(img)
#     plt.subplot(121), plt.imshow(img, cmap='gray')
#     plt.title('Original Image'), plt.xticks([]), plt.yticks([])
#     plt.subplot(122), plt.imshow(edges, cmap='gray')
#     plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
#     plt.show()



# 模型训练集生成代码

import cv2
import mxnet as mx
import pytesseract
def get_time(path):
    # print(path)
    # img = mx.image.imread(path, 1)[50:120, 461:727]
    img_cv = cv2.imread(path)[50:120, 461:720]
    cv2.imwrite("fog_head/let.png",img_cv)

    # By default OpenCV stores images in BGR format and since pytesseract
assumes RGB format,
    # we need to convert from BGR to RGB format/mode:
    img_rgb = cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB)
    res = pytesseract.image_to_string(img_rgb,config='--psm 7 -c
tessedit_char_whitelist=0123456789:').strip()
    # res = ocr.ocr_for_single_line(img)
    # # print(res)
    # res = "".join(res[-8:])
    # ch = ""
    try:
        hour = res[-8:-6]
        hour = int("".join(hour))
        mins = res[-5:-3]
        mins = int("".join(mins))
        sec = res[-2:]
        sec = int("".join(sec))
        # print(hour,mins,sec)
        if sec >0 and sec <7.5:
            sec = 0
```

```
        elif sec >7.5 and sec<22.5:
            sec = 15
        elif sec > 22.5 and sec<37.5:
            sec = 30
        elif sec > 37.5 and sec<52.5:
            sec = 30
        elif sec >52.5 and sec < 59.5 :
            sec = 0
            if mins <= 58:
                mins += 1
            else:
                mins = 0
                hour += 1
        time = f"2020-03-13 {'%02d'%hour}:{'%02d'%mins}:{'%02d'%sec}"
        # print(time)
        return time
    except :
        print(res,"err",path)
        return None

import pandas as pd
df=pd.read_csv("./VIS_R06_12.csv",sep="\t",usecols=['LOCALDATE
(BEIJING)','RVR_1A'])
df.to_csv("./vis_csv_new.csv")
def get_data(time):

    # print(df.head())
    # print(time,"2020-03-12 16:00:30")
    # print(time)
    row = df.loc[df['LOCALDATE (BEIJING)'] == time]
    # print(time,row)
    try:

        data = int(row.values[0][1])
        if data >=0 and data <200:
            return "LOW"
        # elif data >= 50 and data <100:
        #     return "LOW"
        # elif data >= 100 and data < 250:
        #     return "MID"
        elif data >= 200 and data < 350:
            return "MID"
        elif data >= 350 and data < 1000:
            return "HIGH"
```

```
        elif data >= 1000 :
            return "VERY_HIGHT"
    except:
        return None
    # print(a.head())


import os
import shutil
i = 1
for _,_,files in os.walk("train"):
    for name in files:
        if name == ".DS_Store":
            continue
        i+=1
        n = "train/"+name
        # print("fog/"+name)
        time = get_time(n)
        if not time:
            print("not time",n)
            continue

        data = get_data(time)
        if not data:
            print("not data",n)
        img = cv2.imread(n)
        # [:,450:661]
        cv2.imwrite(f"fog_clip/{data}.{i}.jpg",img)
        # shutil.copyfile(n,f"fog_clip/{data}.{i}.jpg")
# get_data(get_time("fog/4.jpg"))




# 能见度回归分析代码
import pandas as pd
import numpy as np
import seaborn as sns

# t = "RVR_1A"
t = "MOR_1A"
```

```
max_min_scaler = lambda x: (x - np.min(x)) / (np.max(x) - np.min(x))


def regularit(df):
    newDataFrame = pd.DataFrame(index=df.index)
    columns = df.columns.tolist()
    for c in columns:
        d = df[c]
        MAX = d.max()
        MIN = d.min()
        newDataFrame[c] = ((d - MIN) / (MAX - MIN)).tolist()
    return newDataFrame


df_2019 = pd.read_csv('merge.index.csv')
df_2020 = pd.read_csv('merge.2020.csv')

df = df_2019.append(df_2020, sort=False)
df = df.sort_values(t)
df = regularit(df[['WS2A', 'CW2A', 'PAINS', 'TEMP', 'RH', t]])
df_2019 = df_2019.sort_values(t)
df_2019 = regularit(df_2019[['WS2A', 'CW2A', 'PAINS', 'TEMP', 'RH', t]])
df_2020 = df_2020.sort_values(t)
df_2020 = regularit(df_2020[['WS2A', 'CW2A', 'PAINS', 'TEMP', 'RH', t]])
# feature_cols = ['WS2A', 'WD2A', 'CW2A',
#        'LIGHTS',  'PAINS', 'QNH', 'QFE_R06', 'TEMP', 'RH', 'DEWPOINT']
# df = regularit(df[['WS2A','CW2A', 'LIGHTS','PAINS','TEMP', 'RH' ,t]])
# df = df.sort_values(t)
feature_cols = ['WS2A', 'CW2A', 'PAINS', 'TEMP', 'RH']
X_19 = df_2019[feature_cols]
y_19 = df_2019[t]

X_20 = df_2020[feature_cols]
y_20 = df_2020[t]

X = df[feature_cols]
y = df[t]

# from sklearn.model_selection import train_test_split
# X_train, X_test, y_train, y_test = train_test_split(X, y)
# print(X_train.shape)
# print(X.shape)
X_train, X_test, y_train, y_test = X, X_20, y, y_20
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
from sklearn.linear_model import LinearRegression


##2 模型的加载
model = LinearRegression()


##3 模型的训练
model.fit(X_train, y_train)
model

# print(model.predict(data_X[:4,:]))
# print(data_y[:4])

# print(model.predict(X_train))
print(model.coef_)
print(model.intercept_)

# y_test=y_test.sort_values(t)[feature_cols]
y_pred = model.predict(X_test)
# y_pred=y_pred.sort()
from sklearn.metrics import mean_squared_error, r2_score

print('Mean squared error: %.2f'
      % mean_squared_error(y_test, y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'
      % r2_score(y_test, y_pred))

from matplotlib import pyplot as plt

test = y_test.tolist()
pred = y_pred
# print(test.tolist())
# print(y_pred)
test = pd.Series(test, name="test data (2020)")
pred = pd.Series(pred, name="predicte data")
tp = pd.concat([test, pred], axis=1)
tp['index_MOR_1A'] = range(1, len(tp) + 1)
print(tp)
# tp.plot(kind="hist",alpha=0.4)
tp.sort_values('predicte data')
tp = tp.reset_index(drop=True)
```

```
tp.to_csv('fenbu.csv', index=True)
tp.plot.scatter(x='index_MOR_1A', y='predicte data')
fig = plt.gcf()
fig.savefig(f'reg_imgs/mor_pred.png')
tp.plot.scatter(x='index_MOR_1A', y='test data (2020)')
fig = plt.gcf()
fig.savefig(f'reg_imgs/mor_test.png')

# MOR_1A

# 测试数据总数 5760

# ['WS2A','CW2A' ,'PAINS', 'TEMP', 'RH' ]
# [ 0.41448949 -0.31260577  0.26520221  0.5227623  -0.46001655]
# 0.3377940016168508
# Mean squared error: 0.02
# Coefficient of determination: 0.86

# 照片顺序


import os
path = r"E:\111\zhunque\low"
filelist = os.listdir(path)
count=1
for file in filelist:
    Olddir=os.path.join(path,file)
    filename=os.path.splitext(file)[0]
    filetype=os.path.splitext(file)[1]
    Newdir=os.path.join(path,'low.'+str(count)+filetype)
    # Newdir=os.path.join(path,str(count)+filetype)

    os.rename(Olddir,Newdir)
    count+=1

# 获取图片
from __future__ import division, print_function, absolute_import
import sys
sys.path.append('../../')
import shutil
import os
import random
import math

root_path= r'E:\111'
```

```
image_path = r'E:\111\Fog20200313000026'
image_output = r'E:\111\444'


image_list = os.listdir(image_path)
image_name = [n.split('.')[0] for n in image_list]
image=[]
a=0
for i in range(10000):
    a=a+1
    b = 7722 + a * 15
    print(b)
    if b<=27941:
        image.append(b)
    # else:
    #     break
# print(image)

for i in image:
    shutil.copy(os.path.join(image_path, str(i) + '.jpg'), image_output)
```

```
# 预测

import numpy as np
import utils
import cv2
from keras import backend as K
from model.VGG16 import VGG16
import os
import glob as gb
from sklearn.metrics import confusion_matrix


K.set_image_dim_ordering('tf')

if __name__ == "__main__":
    model = VGG16(4)
```

```
    model.load_weights(r"./logs/ep045-loss0.198-val_loss0.288.h5")
    test_path = './test/'
    # jpg_list = os.listdir(test_path)
    # sorted(jpg_list)

    y_pred = []
    y_true = []
    image_file = os.listdir(test_path)
    for image in image_file:
        print(image)
        label_true = image.split(".")[0]
        y_true.append(label_true)

    img_path = gb.glob(r'./test/*.jpg')
    for path in img_path:
        print(path)
        img = cv2.imread(path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = img/255
        img = np.expand_dims(img, axis=0)
        img = utils.resize_image(img, (224, 224))
        #utils.print_answer(np.argmax(model.predict(img)))
        result = utils.print_answer(np.argmax(model.predict(img)))
        y_pred.append(result)
        # print(utils.print_answer(np.argmax(model.predict(img))))
    print(y_pred)
    print(y_true)
    C = confusion_matrix(y_true, y_pred, labels=["dense", "low", "good",
"clea"])
    print(C)
```

#训练

```
from keras.callbacks import TensorBoard, ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping
from keras.utils import np_utils
from keras.optimizers import Adam
from model.VGG16 import VGG16
import numpy as np
import utils
import cv2
from keras import backend as K
import matplotlib.pyplot as plt
```

```
K.set_image_dim_ordering('tf')

def generate_arrays_from_file(lines,batch_size):
    # 获取总长度
    n = len(lines)
    i = 0
    while 1:
        X_train = []
        Y_train = []
        # 获取一个batch_size大小的数据
        for b in range(batch_size):
            if i==0:
                np.random.shuffle(lines)
            name = lines[i].split(';')[0]
            # 从文件中读取图像
            #img = cv2.imread(r".\data\image\train" + '/' + name)
            img = cv2.imread(r"E:\vgg16xwh\VGG16-Keras-
master\data\image\train" + '/' + name)
            img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
            img = img/255
            X_train.append(img)
            Y_train.append(lines[i].split(';')[1])
            # 读完一个周期后重新开始
            i = (i+1) % n
        # 处理图像
        X_train = utils.resize_image(X_train,(224,224))
        X_train = X_train.reshape(-1,224,224,3)
        #Y_train = np_utils.to_categorical(np.array(Y_train),num_classes= 2)
        Y_train = np_utils.to_categorical(np.array(Y_train), num_classes=4)
        yield (X_train, Y_train)


if __name__ == "__main__":
    # 模型保存的位置
    #log_dir = "./logs/"
    log_dir = "E:/vgg16xwh/VGG16-Keras-master/logs/"

    # 打开数据集的txt
```

```
    with open(r"E:\vgg16xwh\VGG16-Keras-master\data\train.txt","r") as f:
        lines = f.readlines()

    # 打乱行，这个txt主要用于帮助读取数据来训练

    # 打乱的数据更有利于训练
    np.random.seed(10101)
    np.random.shuffle(lines)
    np.random.seed(None)

    # 90%用于训练，10%用于估计。
    num_val = int(len(lines)*0.1)
    print(num_val)
    num_train = len(lines) - num_val
    print(num_train)

    # 建立AlexNet模型
    model = VGG16(4)

    # 注意要开启skip_mismatch和by_name

model.load_weights("vgg16_weights_tf_dim_ordering_tf_kernels.h5",by_name=True
,skip_mismatch=True)

    # 指定训练层
    for i in range(0,len(model.layers)-5):
        model.layers[i].trainable = False

    # 保存的方式，3世代保存一次
    checkpoint_period1 = ModelCheckpoint(
                                    log_dir + 'ep{epoch:03d}-loss{loss:.3f}-
val_loss{val_loss:.3f}.h5',
                                    monitor='acc',
                                    save_weights_only=False,
                                    save_best_only=True,
                                    period=3
                                )

    # 学习率下降的方式，acc三次不下降就下降学习率继续训练
```

```
    reduce_lr = ReduceLROnPlateau(
                            monitor='acc',
                            factor=0.5,
                            patience=3,
                            verbose=1
                        )

    # 是否需要早停，当val_loss一直不下降的时候意味着模型基本训练完毕，可以停止
    early_stopping = EarlyStopping(
                            monitor='val_loss',
                            min_delta=0,
                            patience=10,
                            verbose=1
                        )


    # 交叉熵
    model.compile(loss = 'categorical_crossentropy',
            optimizer = Adam(lr=1e-3),
            metrics = ['accuracy'])


    # 一次的训练集大小
    batch_size = 16

    print('Train on {} samples, val on {} samples, with batch size
{}.'.format(num_train, num_val, batch_size))


    # 开始训练
    hist = model.fit_generator(generate_arrays_from_file(lines[:num_train],
batch_size),
            steps_per_epoch=max(1, num_train//batch_size),
            validation_data=generate_arrays_from_file(lines[num_train:],
batch_size),
            validation_steps=max(1, num_val//batch_size),
            epochs=100,
            initial_epoch=0,
            callbacks=[checkpoint_period1, reduce_lr])

    model.save_weights(log_dir+'last1.h5')
    # acc = hist.history['acc']
    # loss = hist.history['loss']
    # epochs = range(1, len(acc) + 1)
```

```
    #
    # plt.title('Training and Validation Loss')
    # plt.plot(epochs, acc, 'red', label='Training acc')
    # plt.plot(epochs, loss, 'blue', label='Training loss')
    # plt.legend()
    # plt.show()
    #
    # val_acc = hist.history['val_acc']
    # val_loss = hist.history['val_loss']
    # epochs = range(1, len(val_acc) + 1)
    #
    # plt.title('Validation Accuracy and Loss')
    # plt.plot(epochs, val_acc, 'red', label='Validation acc')
    # plt.plot(epochs, val_loss, 'blue', label='Validation loss')
    # plt.legend()
    # plt.show()
    acc = hist.history['acc']
    val_acc = hist.history['val_acc']
    epochs = range(1, len(acc) + 1)

    plt.title('Training and Validation Accuracy')
    plt.plot(epochs, acc, 'red', label='Training acc')
    plt.plot(epochs, val_acc, 'blue', label='Validation acc')
    plt.xlabel('epoches')
    plt.legend()
    plt.show()

    loss = hist.history['loss']
    val_loss = hist.history['val_loss']
    epochs = range(1, len(val_acc) + 1)

    plt.title('Training and Validation Loss')
    plt.plot(epochs, loss, 'red', label='Training loss')
    plt.plot(epochs, val_loss, 'blue', label='Validation loss')
    plt.xlabel('epoches')
    plt.legend()
    plt.show()

# VGG16 神经网络

import tensorflow as tf
from tensorflow import keras
from keras import Model,Sequential
from keras.layers import Flatten, Dense, Conv2D, GlobalAveragePooling2D
from keras.layers import Input, MaxPooling2D, GlobalMaxPooling2D
```

```
def VGG16(num_classes):

    image_input = Input(shape = (224,224,3))

    # 第一个卷积部分

    x = Conv2D(64,(3,3),activation = 'relu',padding = 'same',name =
'block1_conv1')(image_input)
    x = Conv2D(64,(3,3),activation = 'relu',padding = 'same', name =
'block1_conv2')(x)
    x = MaxPooling2D((2,2), strides = (2,2), name = 'block1_pool')(x)


    # 第二个卷积部分

    x = Conv2D(128,(3,3),activation = 'relu',padding = 'same',name =
'block2_conv1')(x)
    x = Conv2D(128,(3,3),activation = 'relu',padding = 'same',name =
'block2_conv2')(x)
    x = MaxPooling2D((2,2),strides = (2,2),name = 'block2_pool')(x)


    # 第三个卷积部分

    x = Conv2D(256,(3,3),activation = 'relu',padding = 'same',name =
'block3_conv1')(x)
    x = Conv2D(256,(3,3),activation = 'relu',padding = 'same',name =
'block3_conv2')(x)
    x = Conv2D(256,(3,3),activation = 'relu',padding = 'same',name =
'block3_conv3')(x)
    x = MaxPooling2D((2,2),strides = (2,2),name = 'block3_pool')(x)


    # 第四个卷积部分

    x = Conv2D(512,(3,3),activation = 'relu',padding = 'same', name =
'block4_conv1')(x)
    x = Conv2D(512,(3,3),activation = 'relu',padding = 'same', name =
'block4_conv2')(x)
    x = Conv2D(512,(3,3),activation = 'relu',padding = 'same', name =
'block4_conv3')(x)
    x = MaxPooling2D((2,2),strides = (2,2),name = 'block4_pool')(x)


    # 第五个卷积部分

    x = Conv2D(512,(3,3),activation = 'relu',padding = 'same', name =
```

```
'block5_conv1')(x)
    x = Conv2D(512,(3,3),activation = 'relu',padding = 'same', name =
'block5_conv2')(x)
    x = Conv2D(512,(3,3),activation = 'relu',padding = 'same', name =
'block5_conv3')(x)
    x = MaxPooling2D((2,2),strides = (2,2),name = 'block5_pool')(x)


    # 分类部分
    x = Conv2D(256,(7,7),activation = 'relu',padding = 'valid', name =
'block6_conv4')(x)
    x = Flatten(name = 'flatten')(x)
    x = Dense(256,activation = 'relu',name = 'fullc1')(x)
    x = Dense(256,activation = 'relu',name = 'fullc2')(x)
    x = Dense(num_classes,activation = 'softmax',name = 'fullc3')(x)
    model = Model(image_input,x,name = 'vgg16')

    return model
```