全国大学生嵌入式芯片与系统设计竞赛

基于机器视觉的机场场面积水检测与预警系统

小组成员

曾志成

樊卓铭

刘绎妍

二〇二三年七月

目　录

第一章 设计概述

1.1 设计目的

本研究致力于搭建机场场面积水检测与预警平台，旨在减轻机场场面(跑道、滑行道、停机坪) 积水引发的严重安全事件。通过对机场路面的实时拍摄，并由Hi3516DV300 进行图像的裁剪、分类网和检测网识别得到场面积水信息，进一步完成路面积水探测分析。搭配 Hi3861V100 发送采集数据上传预警平台完成现场勘探和预警，实现对机场场面环境风险的总体性识别判断，为机场环境风险预警及综合管理提供决策支撑。

综合应用 ResNet 分类网和 YOLOV2 检测网等神经网络图像识别技术，结合物联网传感技术实现场面检测与边缘设备控制；以数据融合、模式识别技术，基于多对象的图像数据实现对检测结果的计算与展示和对场面状况的数字重建。

1.2 应用领域

机场作为航空交通的关键节点，其正常运行和航班安全直接受到环境因素的影响。其中，积水是一个常见而严重的问题，可能导致飞机及地面保障车辆的制动和操控受限，甚至引发飞机滑出跑道、地面车辆失控等严重后果。[1]

通过引入机器视觉和图像处理技术，系统能够实时监测机场场面的积水情况，并识别出积水区域。通过预警平台的支持，系统可以快速将积水信息传送给机场管理部门，使其能够及时采取排水和疏导等措施，有效减轻积水带来的安全风险和运营延误，为“平安机场”、“智慧机场”项目发展提供支撑。

1.3 主要技术特点

- **类数字孪生虚实联动监控平台**

利用部署在机场的监测模块，可实现对机场场地的实时监控，完成对积水情况的判断；通过返还不同拍摄角度的积水图样，绘制实际积水区域图；通过 RTSP 流协议回传视频，使用透视算法将视频变换为俯视图，实现对积水区域的实景监控

- **泛在操作系统驱动多功能模式**

通过中央积水预警平台的多类型终端指令发布，实现终端设备匹配不同种模式；通过多线程实现连接 TCP 服务器、RTSP 视频流、用户输入、检测算法、场景生成等，做到一个终端对整个系统软硬件的全面控制，对场面的全方面监测与预警。后续进一步拓展，可加入更多边缘设备实现联合检测，构建控制与监测场面的全方位操作终端系统。

- **自训练神经网络机器视觉识别**

根据应用场景，贴合实际视觉识别任务，完成训练数据集的收集和标记，构建分类检测双网络，提高识别效率。后续可进一步拓展其功能，增加更多的神经网络和识别程序，实现对机场场面 FOD，人员侵限的检测与预警，做到对机场场面情况的全方位监控。

- **云边协同的算力均衡**

将部分计算工作下放至边缘设备，使上位机专注于收发数据和显示处理。均衡算力的意义在于在未来增加更多边缘设备时减少对上位机算力的需求

1.4 关键性能指标

1.系统应答响应时间

网络连接（局域网）所需时间约 3s，指令传达所需时间约 1.8s，实现路面情况检测周期约 5.4s，数据传输返回约 0.5s，预警平台实现路面情况分析处理约 0.8s，实现路段监控画面传回延时约 1.2s，监控画面处理和视角转换展示约 2.0s，全流程 14.7s，实时性较好。

2.神经网络推理正确率

使用 Restnet 分类模型 mmclassfication 框架完成训练，在 epoch100 时分类网络的正确率为 96.08%，损失 loss 为 0.1101；YOLOv2 模型，利用 darenet-master 框架，在 Avg IOU 能达到 92.1%，obj 识别率为 0.9138，平均召回率接近 1.0。训练效果良好，部署检测性能优良。

3.积水区域图转换误差率

系统部署后完成积水识别，将道路信息传输给预警平台，预警平台可处理道路积水信息，返回积水图例，通过多次实验，区域图和实际误差率小于 8.9%。

1.5 主要创新点

- **利用机器视觉来完成对场面的实时监控，拥有更高的实时性和时效性**

区别于难以在机场场景部署的传统水位电尺和无法查看场面细小水面信息的遥感检测法。机器视觉检测能够在仅依靠视觉的情况下短时间对大范围区域进行检测，且容易被潮湿环境侵蚀。

- **设备间通信协议多样，应用层协议设计巧妙，抗干扰能力强。**

在无线传输使用无线局域网进行通信，在协议层包含 TCP 协议，Telnet 协议，RTSP 视频流协议等。在 TCP 协议传输中，本系统在应用层采用了一个先导字符来区分传输字段的内容，用于正确引导数据流向。该设计增加整个无线通信的鲁棒性和容错性。

所有 tcp 通信协议所传输的命令均采用询问应答方式进行通信。终端设备客户端会不断在合适的时机执行中断向上位机服务端发送对应字段的先导字符进行询问，服务端收到后会把对应的标志位使用相同的字段回答客户端，这样客户端就能不断同步服务端的指令，从而实现控制。由边缘设备来决定通信时机，做到云边协同。

有线传输协议中，我们采用了 uart 串口来进行通信。用于使能抓拍与回传检测结果，回传的检测结果再经由 TCP 协议进行转发。

- **设备存在休眠状态，功耗较低。**

  查看场面积水情况由中心发送指令，在无任务需求时无需唤醒终端设备。休眠时边缘设备控制终端会每 14 秒向服务端发出“是否启动”的询问，若回答中的对应字节为‘0’则继续休眠。积水检测模块会等待远程 telnet 终端主动连接自己，连接后等待执行程序的嵌入式 linux 终端命令

- **预警平台查看积水情况，操作界面智能简洁。**

  在预警平台程序上，有解码检测数据、检测结果展示、全景监控生成、积水图例生成四种模式，可以清晰的了解路面情况，确定积水状态。

- **人工+机器双验证，容错性高**

  航行安全始终是机场人员的最高追求，积水检测任务也需要确保其安全性。本产品同时训练分类和检测网络，相互提供佐证，识别更为精确；同时，通过 RTSP 流传输现场视频，可以实现对路面监控，供工作人员进一步确认。

第二部分 系统组成及功能说明

2.1 整体方案介绍

本作品为基于机器视觉的机场场面积水检测与预警系统，系统目标是获得机场跑道的积水区域，并能够完成对积水路段的监控和预警。以下为技术路线图。

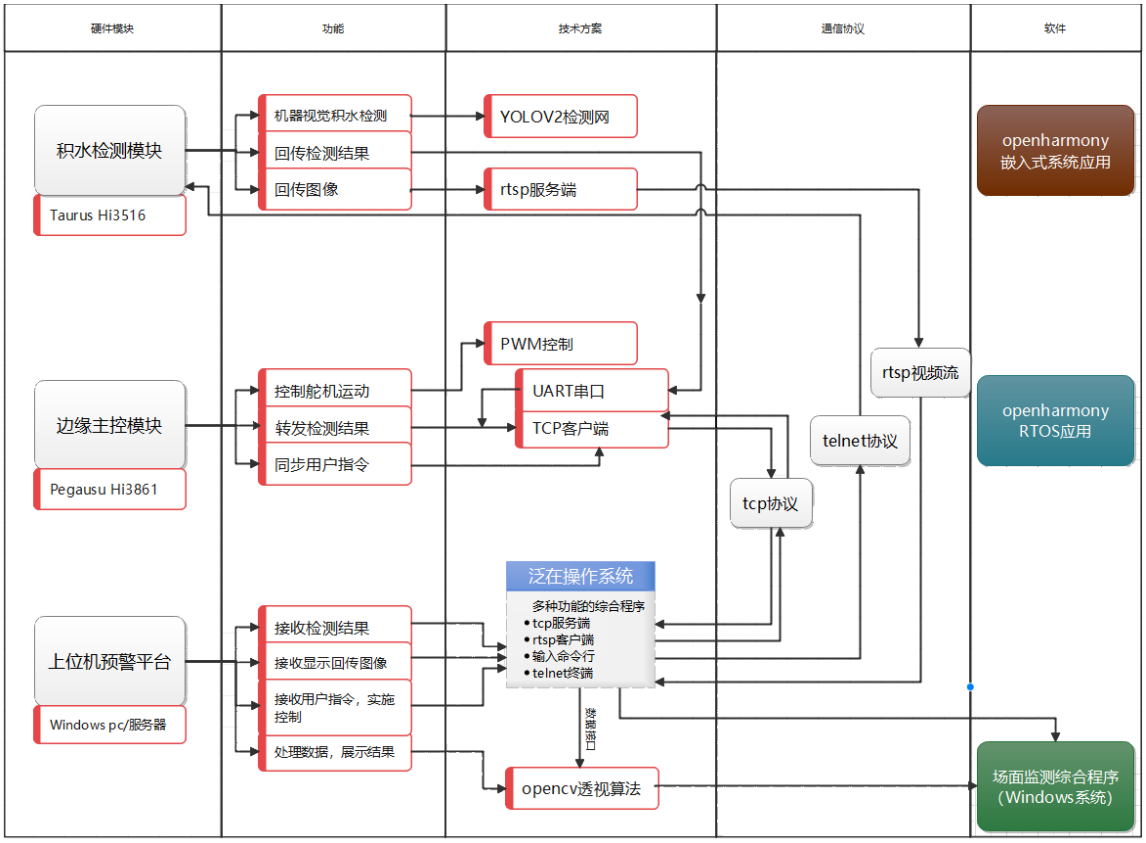


图 1 技术路线图

通过分析每个硬件模块要承担的功能，我们选择了如上的技术方案和通信协议，并分别开发了三套软件分别部署在两个板端和 pc 端。

2.2 功能模块设计

该作品采用模块化设计思想，主要分为三个部分，分别是积水预警平台、Pegaus 控制模块以及 Taurus 和舵机组成的积水检测模块。三个模块之间能够实现不同协议的通信，完成整个积水检测和预警系统的搭建。

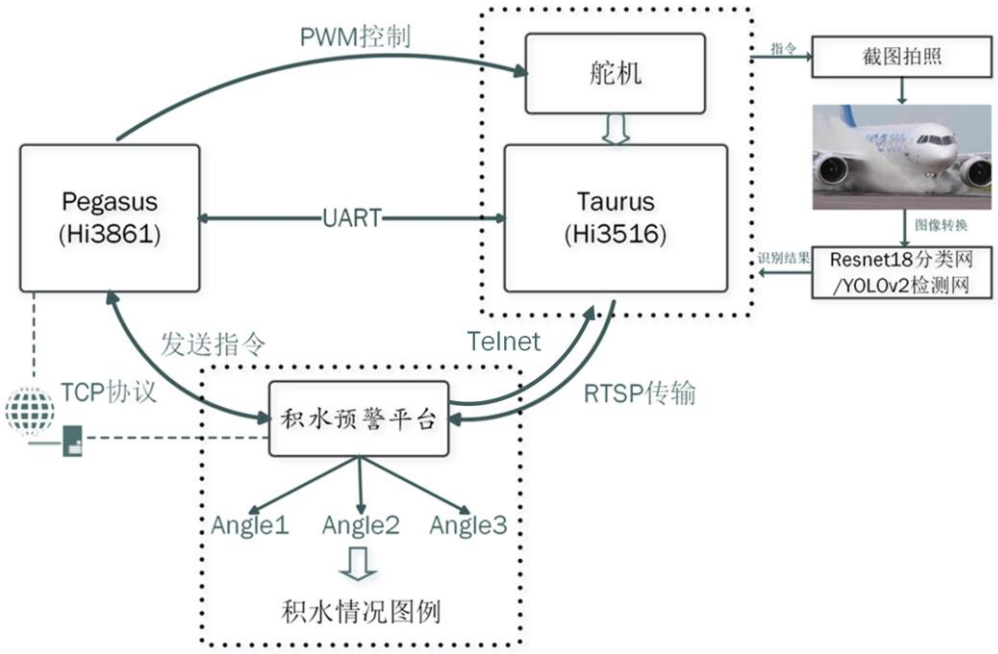


图 2 功能模块系统整体框图

①操作操作人员发起指令，要求对积水路段进行检测；

②Pegasus 组成的终端控制模块通过 TCP 协议发起询问，收到任务后可以发生 PWM 驱动舵机，到位后便能够通过 UART 给 Taurus 发送指令使能拍照，随后执行检测/分类网程序；

③Taurus 和舵机组成的积水检测模块，首先通过 Telnet 完成启动，将拍摄画面在其板端显示，在被告知舵机转向到位后，分类与检测网络被执行，得到此时画面的分类和检测数据信息，经过打包后经 UART 由 TCP 协议返还预警平台；

④同时，利用 RTSP 流传输三个角度的照片发送给积水预警平台；

⑤积水预警平台收到路面信息后，通过透射变换关系，获取积水区域坐标，并能够由图像完成路面情况的监控，供操作人员确认。

2.3.1 积水预警平台

积水预警平台服务于机场塔楼工作人员，当区域飞机发起着陆申请时，便能够通过发起指令来完成对机场场面的检测，由此来判断积水情况是否严重，飞机是否适合降落。同时，为了能够保存场面积水的情况，方便进一步的完成分析，能够由检测网返还的三种角度下的积水分布坐标，通过坐标变换算法大致生成积水分布区域图。

2.3.2 终端控制模块

终端控制模块由 Hi3861V100 组成。首先，作为终端接收模块，在同一局域网下可以通过 TCP 协议接收来自预警平台发送的指令；其次，其能够通过产生 PWM 波驱动舵机，来完成对视频监控模块的旋转角度控制；另外，作为控制模块，能够通过 UART 串口的方式，来实现和视频检测模块之间的通信，完成指令的传达。

2.3.3 积水检测模块

积水检测模块主体由舵机和 Hi3516DV300 组成，通过 Telnet 实现启动。首先，能够实现实时画面的视频流在板载的显示；其次，在每转动一个角度之后，将进行拍照截取，并输入自训练的积水检测 Restnet18 网络进行分类，同时通过 YOLOv2 检测网络给出积水检测区域然后将分类和检测信息通过 UART 的方式经过终端控制模块传回预警平台；另外，能够同时通过 RTSP 将图片传输给积水平台，在平台上可以对路面信息进行监控。

2.4 系统布置说明

构建一个完整的积水监控和预警系统，显然一组设备是不行的。借助无人机组网的思想，期望能够在机场跑道的每一个航行标记灯的位置，布置下一个由终端控制模块和积水检测模块的单元，这样就能实现完全的覆盖。

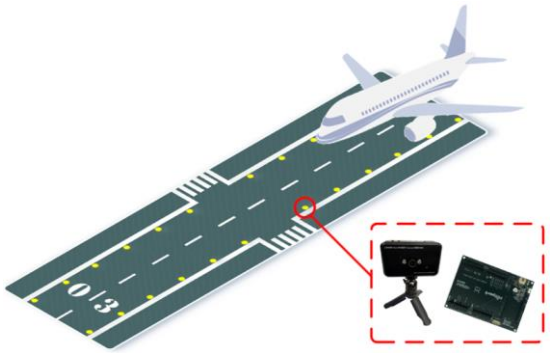


图 3 系统布置

考虑由于经济和时间成本原因，为了方便展示，这里只考虑预警平台和其中一个单元组成的系统，通过该系统的任务，完成对整个系统的搭建思路描述。

第三部分 系统搭建

3.1 硬件系统搭建

➢ 模块连接情况

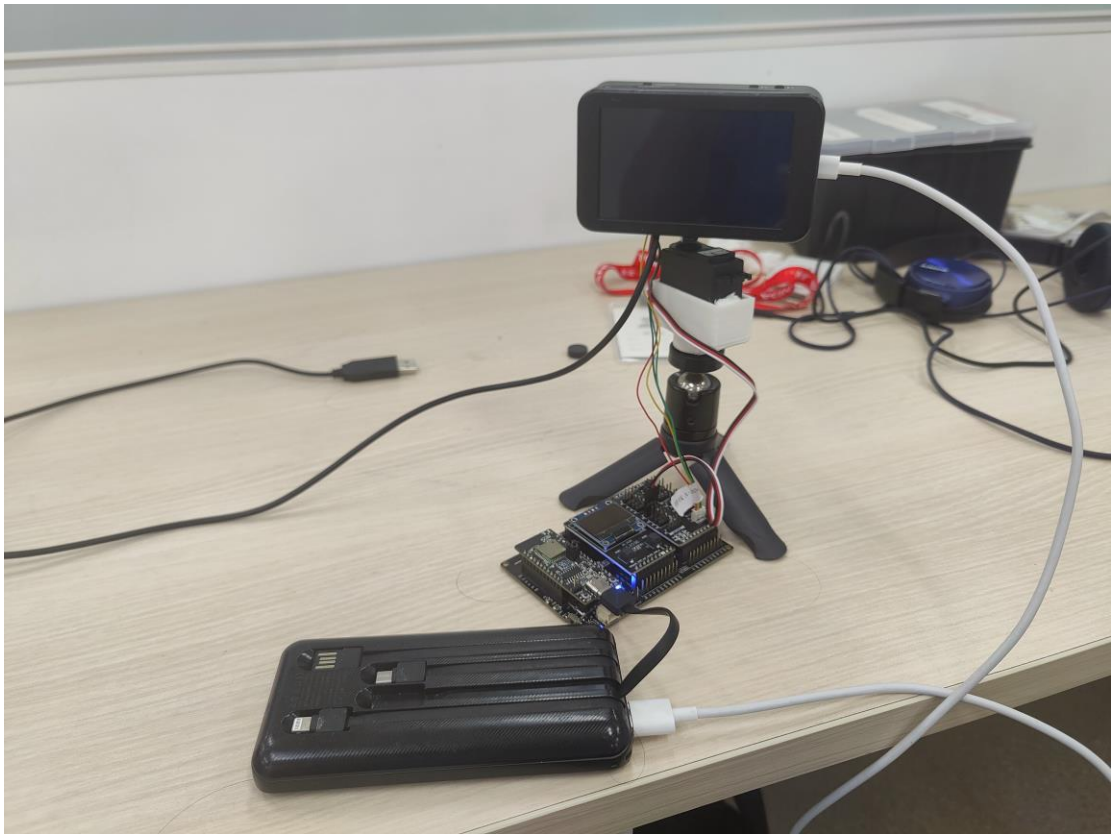


图 4 模块连接情况

积水预警平台、Pegaus 控制模块以及 Taurus 和舵机组成的积水检测模块三个模块之间的布置如上图所示，硬件部分的连接，舵机通过 gpio 口的方式与控制模块连接，Taurus 和控制模块模块之间由 UART 排线的方式进行连接。预警平台可以由共享网络的方式通过 TCP 协议和 Telnet 实现终端互联。

➢ 系统布置情况

(a)左侧视图

(b)俯视图

(c)右侧视图

图 5 系统布置情况

终端控制模块和积水检测模块组成的积水预警单元被安装在道路的一侧，可以完成对道路情况的收集、处理和传输。预警系统可以在可连接共享网络的室内布置。

3.2 软件系统搭建

➢ 上位机软件平台展示

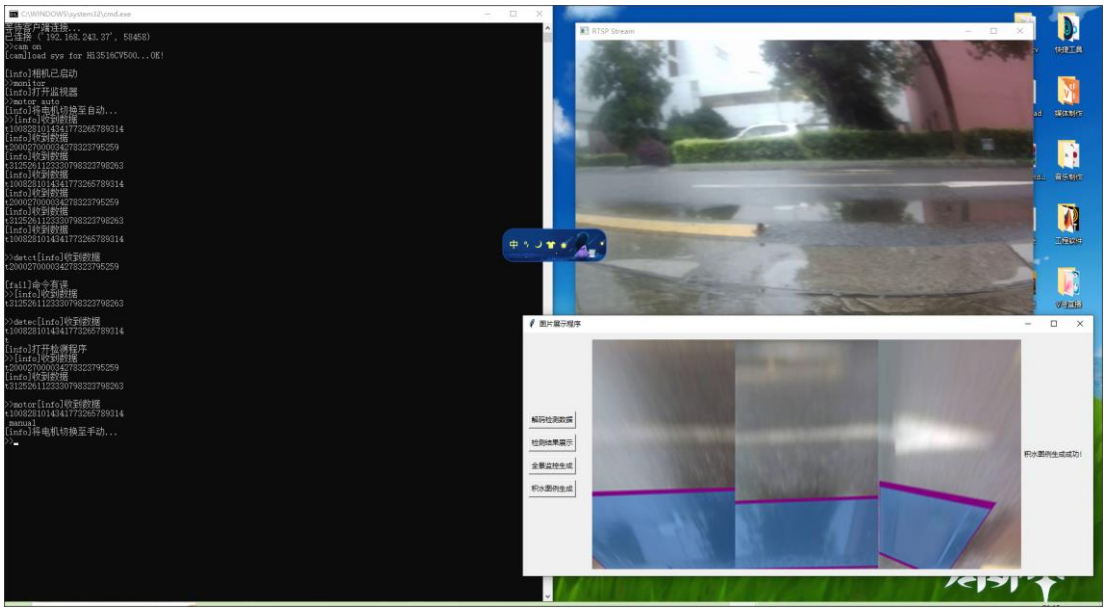


图 6 上位机软件平台

（左为上位机泛在终端，右上为 rtsp 实时画面，右下为类数字孪生展示）

上位机软件由泛在操作终端和各个检测和显示程序组成。目前主要的程序为 rtsp 实时串流与自动截图程序和检测结果显示程序。

上位机操作终端为其他拓展程序提供了相应的数据接口文件，并能调用 Windows 的 os 命令，实现与多个上位机软件的通信与控制。通过 tcp 服务端线程和 telnet 线程以及 rtsp 服务端线程实现从边缘设备收集数据和控制边缘设备。

Pegasus 和 Taurus 板端均有对应的应用程序：Pegasus 上搭载有包含控制舵机，收发 uart 串口数据以及 tcp 客户端的程序；Taurus 上搭载有 rtsp 串流服务端程序，抓拍程序及分类网/检测网积水检测程序。Pegasus 的程序通过单一线程适时阻塞中断来完成数据收发；Taurus 的程序分开使用，由上位机的指令决定启动哪一个程序来实现不同功能。

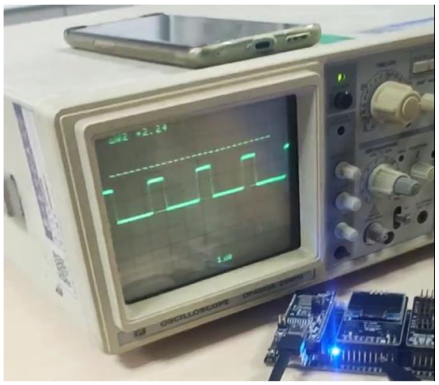
使用时先启动泛在操作终端，通过指令来控制软件和硬件，具体指令详见后文详细介绍。

第四部分 技术方案详解及其性能参数

将系统按功能，大致可以划分为如下部分：

4.1、PWM 波和舵机控制

PWM（Pulse Width Modulation）脉冲宽度调制，通过高频率控制半导体开关器件的通断时间，从而实现控制输出电压的大小。市面上多为模拟型舵机，通过输入一个脉冲，使得电机转到相应位置。所以本质上由输入脉冲波占空比来完成对角度的控制。



(a)PWM 波形

(b)舵机控制

图 7 舵机控制部分功能实现

在本作品中，舵机采用 Futaba S3003 标准舵机，控制 PWM 波发生采用 GPIO 模拟方案，能输出高电平为 3.3V 的 PWM 波，舵机在该电平下大致扭矩为 2kg·cm，能够满足转向要求，由舵机尺寸和套件底座完成 3D 打印连接支架，能够由程序精准控制角度。

以正前方为 0°，舵机在自动模式下会分别转到-30°，0°和+30°执行积水检测。此外，我们的系统还包含手动模式，可以在上位机的终端手动输入角度控制舵机从 0-180°进行转向，由于 tcp 应用层协议统一的缘故，在上位机终端控制时，本设计只能旋转 20 的倍数的角度，但对于监控工作而言已经完全足够了。

4.2、TCP 通信

TCP 是一种可靠的、面向连接的通信协议，用于在网络中进行数据传输。它通过提供可靠性、流量控制和拥塞控制等机制，确保数据能够准确地传输并适应不同的网络环境。由于其可靠性和广泛应用，TCP 成为互联网通信中最常用的协议之一。

在本作品中，上位机通过 TCP 向终端设备传输指令。TCP 服务端始终保持对各个边缘客户端的监听。边缘客户端会在自己合适的时机向 TCP 服务端发起询问， TCP 服务端线程根据用户的指令，回答边缘客户端的询问，从而控制和监测边缘中枢设备（Pegasus）的运行。

客户端询问、传输和回答，服务端问答、询问方式的数组格式如下所示：

表 1 TCP 传输数据格式

格式	字符串格式	含义
客户端询问格式	[m/c]	[舵机/相机]
客户端传输格式	[t]+[角度编号]	rtsp 模式检测
	[t]+[角度编号]+[识别框四点坐标（连续）]	检测网模式
	[t]+[角度编号]+[1/0]	分类网模式
服务端回答格式	[m/c]+[1/0]+[1/0]	[舵机/相机]+[1/0]+[1/0]
服务端询问格式	[a]	\
客户端回答格式	[a]+[1/0]+[1/0]+[1/0]	[a]+[舵机状态]+[执行状态]+[相机状态]
请求异常	[x]	\

（回传的字符串均采用字段字母+数据的格式）

Telnet 协议是 TCP/IP 协议族中的一员，是 Internet 远程登录服务的标准协议和主要方式。它为用户提供了在本地计算机上完成远程主机工作的能力。在终端使用者的电脑上使用 Telnet 程序，用它连接到服务器。终端使用者可以在 Telnet 程序中输入命令，这些命令会在服务器上运行，就像直接在服务器的控制台上输入一样，可以在本地就能控制服务器，本次设计就是使用 python 程序控制 Telnet 的方式唤起 Taurus。

4.3、UART 串口通信连接

通用异步收发传输器(Universal Asynchronous Receiver/Transmitter)，通常称作 UART，是一种异步收发传输器。在 UART 通讯协议中信号线上的状态位高电平代表‘1’，低电平代表‘0’。其特点是通信线路简单，只要一对传输线就可以实现双向通信，大大降低了成本，但是传送速度较慢，因此仅用于使能抓拍和最终检测结果的传输。

作品中，Hi3861V100 和 Hi3516DV300 之间的通信使用 UART 串口通信，利用排线将其连接起来，就实现对检测数据的传输。两者之间波特率配置为 9600，在使能拍照时，3861 向 3516 随意发送一个 ASCII 码，只要 3516 收到任意数据就使能抓照，并进行检测，完成后将回答连续 24 个数字字符的四个坐标（分类网发送一个字节的 0/1），若没检测到将发送 x。3861 收到后会将其在 TCP 发送的 t 字段转发。

4.4、神经网络部署和检测

YOLOv2 采用了一个新的基础模型(特征提取器),称为 Darknet-19,包括 19 个卷积层和 5 个 maxpooling 层，其被广泛应用于目标检测任务当中；残差网络 Resnet18 采用 18 层网络结构，是最为常见的分类单元网络，采用卷积、池化、线性和激活多层的连接构建，可以用来完成视觉分类任务。且这两种框架训练模型能够较为匹配的部署在 Hi3516V300 当中。

本作品采用的是 Resnet18 分类网和 YOLOv2 检测网络，分类网络进行道路信息初步的判断识别，随后输入检测网完成道路积水的检测，随后将返回积水的坐标信息和置信度。

训练过程采用自制数据集，对实地道路中的 500 张积水照片进行拍照存储，

随后通过坐标变换、亮度调节等方式扩充到 1500 张，并使用 LabelImg 工具对积水体进行标注。标注完成将样本图像按照数据比例 8：2 分为训练集和测试集，每幅图像的分辨率均为 1280 像素×720 像素。

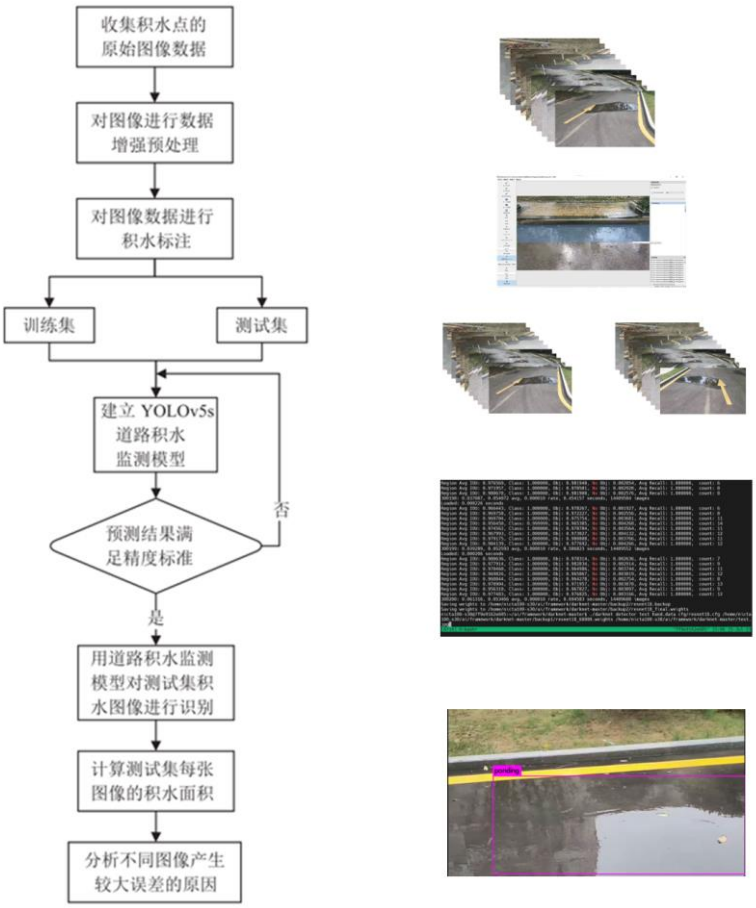


图 6 神经网络训练过程（以检测网为例）

使用 Restnet 分类模型 mmclassfication 框架完成训练，在 epoch100 时分类网络的正确率为 96.08%，损失 loss 为 0.1101；YOLOv2 模型，利用 darenet-master 框架，在 Avg IOU 能达到 92.1%，obj 为 0.9138，平均召回率接近 1.0。可见训练效果良好。

随后，分别将分类和检测网模型文件转换为 caffee model 和适配文件 Prototxt 网路检查，随后利用海思提供的量化工具 RuyiStudio 量化为板载的.wk 模型。

```
   14 conv    128  3 x 3 / 1    80 x  48 x 128   →    80 x  48 x 128  1.132 BFLOPs
   15 conv    128  1 x 1 / 1    80 x  48 x 128   →    80 x  48 x 128  0.126 BFLOPs
   16 res   13                  80 x  48 x 128   →    80 x  48 x 128
   17 max         2 x 2 / 2    80 x  48 x 128   →    40 x  24 x 128
   18 conv    256  1 x 1 / 1    40 x  24 x 128   →    40 x  24 x 256  0.063 BFLOPs
   19 conv    256  3 x 3 / 1    40 x  24 x 256   →    40 x  24 x 256  1.132 BFLOPs
   20 conv    256  1 x 1 / 1    40 x  24 x 256   →    40 x  24 x 256  0.126 BFLOPs
   21 res   18                  40 x  24 x 256   →    40 x  24 x 256
   22 conv    256  1 x 1 / 1    40 x  24 x 256   →    40 x  24 x 256  0.126 BFLOPs
   23 conv    256  3 x 3 / 1    40 x  24 x 256   →    40 x  24 x 256  1.132 BFLOPs
   24 conv    256  1 x 1 / 1    40 x  24 x 256   →    40 x  24 x 256  0.126 BFLOPs
   25 res   22                  40 x  24 x 256   →    40 x  24 x 256
   26 max         2 x 2 / 2    40 x  24 x 256   →    20 x  12 x 256
   27 conv    512  1 x 1 / 1    20 x  12 x 256   →    20 x  12 x 512  0.063 BFLOPs
   28 conv    512  3 x 3 / 1    20 x  12 x 512   →    20 x  12 x 512  1.132 BFLOPs
   29 conv    512  1 x 1 / 1    20 x  12 x 512   →    20 x  12 x 512  0.126 BFLOPs
   30 res   27                  20 x  12 x 512   →    20 x  12 x 512
   31 conv    512  1 x 1 / 1    20 x  12 x 512   →    20 x  12 x 512  0.126 BFLOPs
   32 conv    512  3 x 3 / 1    20 x  12 x 512   →    20 x  12 x 512  1.132 BFLOPs
   33 conv    512  1 x 1 / 1    20 x  12 x 512   →    20 x  12 x 512  0.126 BFLOPs
   34 res   31                  20 x  12 x 512   →    20 x  12 x 512
   35 conv     30  1 x 1 / 1    20 x  12 x 512   →    20 x  12 x  30  0.007 BFLOPs
   36 detection
mask_scale: Using default '1.000000'
Unused field: 'max_boxes = 300'
Loading weights from /home/nicta100-s30/ai/framework/darknet-master/backup2/resnet18_300000.weights...Done!
/home/nicta100-s30/ai/framework/darknet-master/test_o.png: Predicted in 0.004231 seconds.
ponding: 98%
ponding: 98%
nicta100-s30@7f9e9162e605:~/ai/framework/darknet-master$
```

图 8 检测网测试结果

4.5、RTSP 视频流传输

RTSP 全称实时流协议（Real Time Streaming Protocol），RSTP 是一个网络控制协议，设计用于娱乐、会议系统中控制流媒体服务器。RTSP 用于在希望通讯的两端建立并控制媒体会话（session），客户端通过发出 VCR-style 命令如 play、record 和 pause 等来实时控制媒体流。

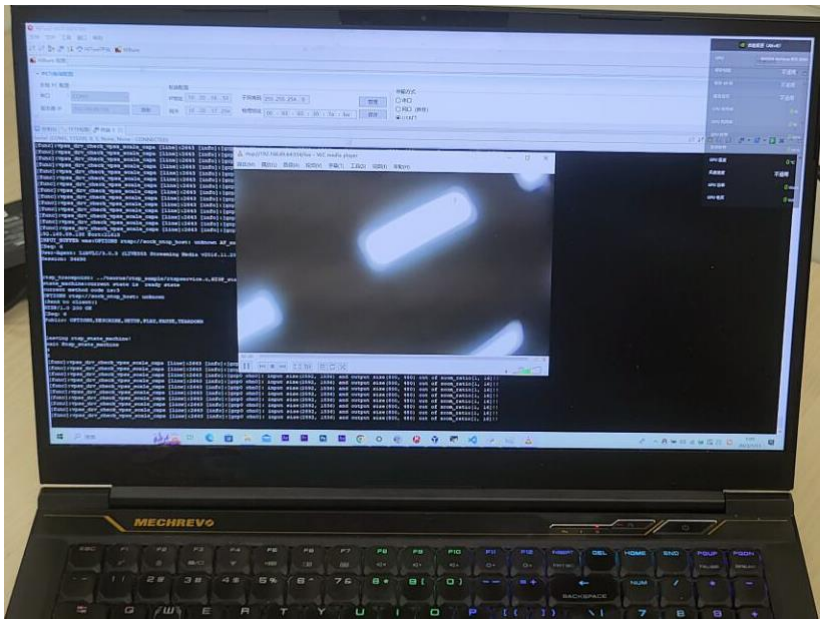


图 9 RSTP 实现 Hi3516DV300 向 PC 的传输

通过 RSTP 协议，不仅能够完成媒体流的控制，在本作品中，通过泛在操作终端提供的数据接口文件，其能够结合指令发送的具体信息，通过 opencv 的读取和保存，实现某一状态下单张图片的截图保存，方便后续的处理显示过程。视频流传输的视频为 800*480，25fps 的视频，平均码率为 7.15Mbps。

4.6、畸变矫正与透视投影算法

对于 Hi3516DV300 拍摄的图片，首先需要完成对其的预处理，预处理包括相机的广角畸变矫正、感兴趣 region 的裁剪和图像增强。

对于相机畸变矫正，本作品中的相机畸变属于“桶形失真”，由相机坐标系到图像坐标系的转换，可以通过张正友标定法结合 Brown 算法来测量畸变参数，随后使用 cv.calibrateCamera( )和 cv.getOptimalNewCameraMatrix( )便能获得相机参数，接下来就能完成图像的校准和矫正。畸变坐标对应关系如下

$$\begin{cases} x' = (u - c_x)/f_x \\ y' = (v - c_y)/f_y \end{cases} \rightarrow \begin{cases} x'' = x' \cdot (1 + k_1\gamma^2 + k_2\gamma^4) + p_2 \cdot (\gamma^2 + 2 \cdot x'^2) + 2 \cdot p_2 x' \cdot y' \\ y'' = y' \cdot (1 + k_1\gamma^2 + k_2\gamma^4) + p_1 \cdot (\gamma^2 + 2 \cdot y'^2) + 2 \cdot p_2 x' \cdot y' \end{cases}$$

通过标定，本相机的畸变系数大致为

$$[k1,\ k2;\ p1,\ p2] = [-0.2121,\ \ 0.1043;\ \ 0.0035,\ 0.0009]$$

其中，k1,k2 为径向畸变系数，p1,p2 为切向畸变系数。得到畸变系数后便能够进行转换和裁剪。

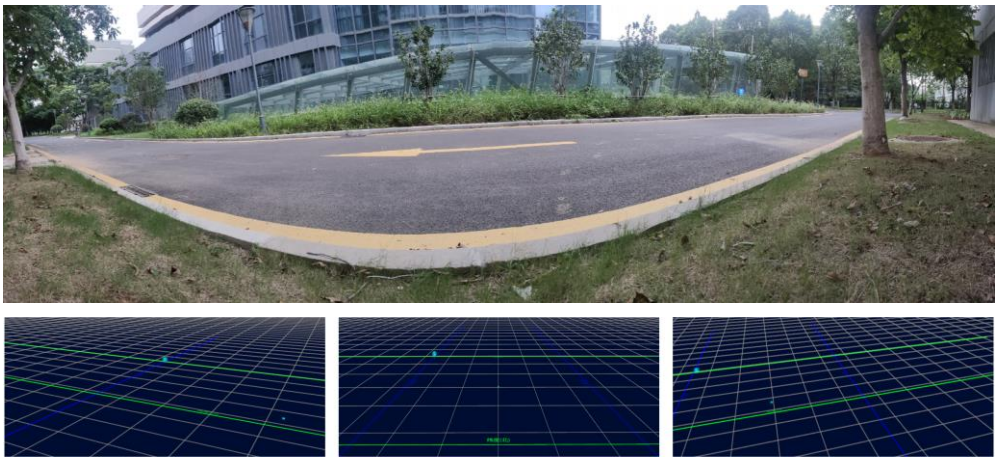


图 10 透视算法原理示意图

通过坐标变换程序调用，能够实现透视投影。侧视坐标系与俯视坐标系之间的变换关系为

$$H_{12} = KR_{12}K^{-1}$$

其中，已知 PnP 计算得出的旋转矩阵为 R_{12}，该旋转矩阵可以分解为 $R_{12} = R_z R_x R_z$，即俯视相机坐标系先绕 x 轴旋转变成斜视，再绕旋转后的坐标轴的 z 轴旋转（对应安装误差），最后绕初始的 z 轴旋转（对应标定纸摆放倾斜）。在完成分解之后，只需令 R_z=I，重新计算 $R'_{12} = R_z R_x$，即可得到矫正后的相对旋转。

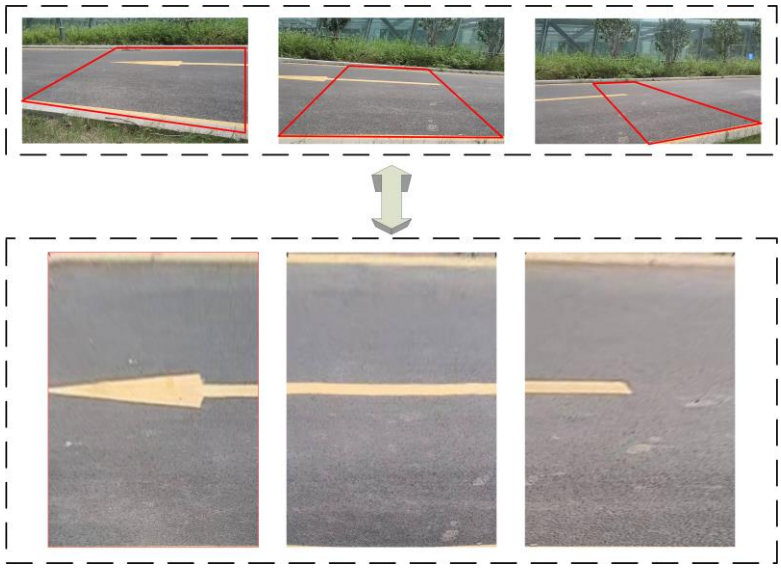


图 11 道路检测目标透视变换效果展示

根据这样的变换关系，由边缘计算完成检测网返回检测目标的位置信息，通过变化，便能够完成积水区域的转换，结合实际路面尺寸，便能够得到积水区域的基本情况。利用 canny 边缘检测道路边界，通过透视投影的变换原理，便可以利用 RTSP 回传的图片完成积水图例的构建。上面为实际过程中的效果展示。

4.7、上位机泛在操作系统

泛在操作系统作为所有数据流的中枢，负责在上位机统筹和分配回传的数据和下达用户输入的命令。

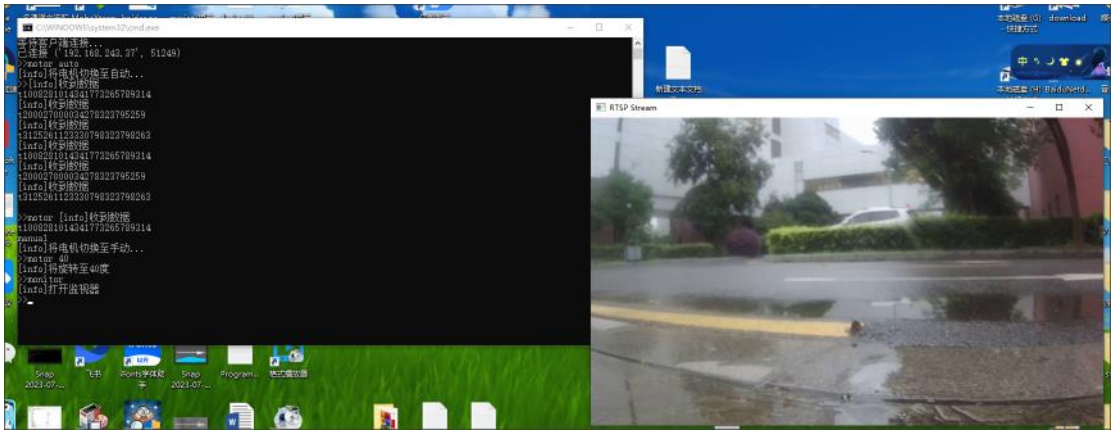


图 12 上位机泛在操作系统

TCP 服务端线程根据用户的指令，回答边缘客户端的询问，从而控制和监测边缘中枢设备（Pegasus）的运行。

表 2 上位机用户命令对照表

命令	指令含义	命令	指令含义
motor auto	自动舵机	motor [angel]	转动角度
motor manual	手动舵机	detect	启动检测程序
monitor	启动监控器	cam on	启动相机
cam off	关闭相机	state	查看工作状态

Telnet 命令线程和 RTSP 客户端线程用于直接与 Taurus 通信，Telnet 线程负责根据用户指令给其发送 Linux 命令，RTSP 客户端负责从 Taurus 拉流视频用于实时监控和积水俯视图显示。

用户输入线程负责接收用户输入的命令，并根据命令操作程序中的公共变量，TCP 线程将直接使用这些公共变量回答边缘客户端的询问。对于一些指令，该线程也直接调用 Windows OS 命令启动检测展示等窗口程序。

此外，该系统还提供数据接口文件(目前直接使用文本文档)来为诸如检测结果计算显示的拓展程序提供数据。以该程序为例，系统将接收到的最新数据（t 字段 tcp 通信）保存在 tcp.txt 的文件中，该检测结果计算展示程序就能通过该文件中保存的三方位的积水检测结果（检测网所返回的检测结果四点坐标），结合 rtsp 客户端程序所截下的三方位照片，将结果以虚映实，实现类数字孪生的方式展示积水检测结果。

4.8、积水预警平台

图 13 积水预警平台操作界面

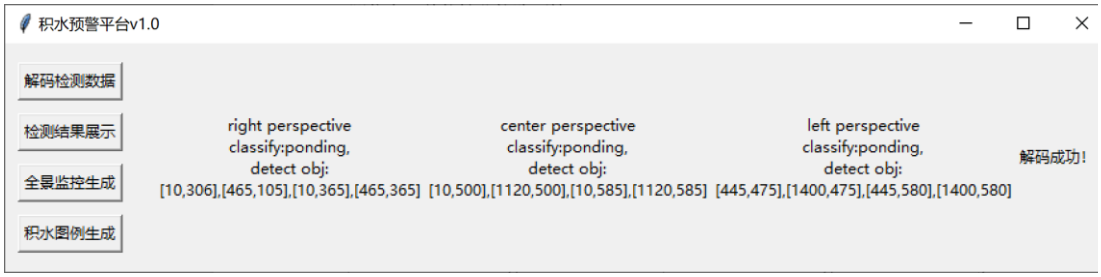
积水预警平台对接由边缘系统传输的路面信息，供操作人员了解路面积水检测信息，并能够监控路面情况：

• 解码检测数据：对积水监控设备传输回来的道路信息“tcp.txt“进行解码，初步判断是否有积水；

• 检测结果展示：获取分类网和检测网探测数据，并能由检测数据框定积水区域；

• 全景监控生成：对拍摄情况完成全景图像生成，供操作人员监控现场情况；

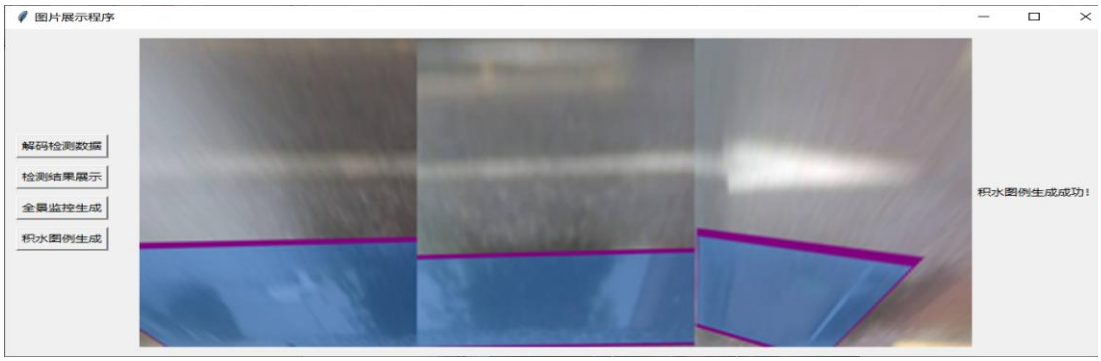
• 积水图例生成：智能视角转换，对路面情况进俯视角度检测，进一步确保机场安全。



(a)解码检测数据

(b)检测结果展示

(c)全景监控生成



(d)积水图例生成

图 14 积水预警平台功能展示

第五部分 总结

5.1 可拓展部分

1. 通过对积水数据的收集和分析，我们可以建立积水数据库，并进行趋势分析和预测。这有助于机场管理者进行规划和优化排水系统，提高机场的抗洪能力和航空安全水平。
2.增加手动控制精度，由于当前应用层协议的缘故，对于舵机角度只分配了一个字节且使用数字字符进行通信，能传递的信息过少，通过改进应用层协议，可以使终端对远端的控制更精确。
3. 通过更多边缘设备协同检测，能够检测更广的区域；通过加入更多不同的神经网络能够实现更多如人员侵限，场面 FOD 等；最后还能进一步给上位机的泛在操作系统增加 GUI，增加更多功能性拓展程序，真正实现数字化机场场面监测中心。

5.2 当前作品存在的问题及反思

1. 图像透视算法效果欠佳

由于检测摄像头的高度过低，镜头俯仰角度设置不合理，导致透视变换十分困难。透视算法即使经过了镜头畸变修正，权重化透视变化，还原出的俯视效果也并不令人满意。

2. 操作终端仍需改进

虽然该终端已经能完成协调控制软硬件和数据流的任务，但命令行式的操作终端终究离真正的“泛在操作系统”还有不小的距离，能控制的设备数量也有限。今后得把命令行换成一个长得像“电脑桌面”的窗口。给 TCP 服务端线程添加更多监听，让其能够真正完成“泛在操作”。

5.3 心得体会

本作品为基于机器视觉的机场场面积水检测与预警系统。积水预警平台、Pegaus 控制模块以及 Taurus 和舵机组成的积水检测模块。三个模块之间能够实现同协议的通信，即 Pegasus 通过 pwm 控制舵机的角度转动，Taurus（Hi3516）粘附在舵机上随之转动，而 Taurus 通过 RTSP 传输将识别结果传送给积水预警平台（即在电脑上显示图像），完成整个积水检测和预警系统的搭建。
我们的工作模式是：

首先，由摄像头采集有积水的路面，图像积水检测模块主体由舵机和 Hi3516DV300 组成，能够实现实时画面的视频流在板载的显示；其次，摄像头将转动三个角度，在每转动一个角度之后，进行拍照截取，将截取的图片输入已训练好的积水检测 Restnet18 网络进行分类，然后将分类信息通过 UART 的方式经

过终端控制模块传回预警平台；而后，通过 RTSP 将图片传输给积水平台，平台通过 YOLOv2 检测网络给出积水检测区域。

综上，我们得到三种角度下的积水分布坐标，通过特定的坐标变换大致生成积水分布区域图在预警平台呈现，积水预警平台服务于机场塔楼工作人员，当区域飞机发起着陆申请时，便能够通过发起指令来完成对机场场面的检测，由此来判断积水情况是否严重，飞机适合在哪条跑道进行降落。

参考文献

[1] http://finance.people.com.cn/n1/2021/0901/c1004-32214863.html-民航局通报华夏航空航班滑出跑道原因：机组使用刹车时机偏晚--经济·科技--人民网
[2] https://blog.csdn.net/shyjhyp11/article/details/109506149-相机畸变校正详解
[3] 白岗岗, 侯精明, 韩浩, 等. 基于深度学习的道路积水智能监测方法[J]. 水资源保护, 2021, 37(5): 75-80.
[4] 王海, 蔡柏湘, 蔡英凤, 等. 基于语义分割网络的路面积水与湿滑区域检测[J]. 汽车工程, 2021, 43(4): 485-491.
[5] 钱永军, 姜仕军, 臧勐佳, 等. 基于深度学习的轨道车辆线阵相机图像畸变校正方法[J]. 智慧轨道交通, 2021.
[6] 卢鹏. 数字孪生技术在智慧机场建设中的应用综述[J]. 电子通信与计算机科学, 2022, 4(4): 44-45.

附录

A.嵌入式系统代码

A.1 pwm 波发生和舵机控制

```
static void ToDeg(int angle)
{
    printf("%d\n",angle);
    int angle_time;
    int angle_anti_time;
    int conut;
    angle*=20;
    printf("%d\n",angle);
    angle_time=angle*11+500;

    angle_anti_time=20000-angle_time;

    printf("%d\n",angle_time);
    printf("%d\n",angle_anti_time);

        for (conut=0;conut<50;conut++)
        {
            // set GPIO_2 output high levels to turn on LED(servo)
            IoTGpioSetOutputVal(LED_GPIO, 1);

            // delay 500+angle*1000/90us(angle/90+0.5ms,rotate to 30degree)
            hi_udelay(angle_time);
```

```
            // set GPIO_2 output low levels to turn off LED
            IoTGpioSetOutputVal(LED_GPIO, 0);

            // delay 20000-angle_timeus(20ms-~us)
            hi_udelay(angle_anti_time);
        }
          // wait for 2s
        IoTGpioSetOutputVal(LED_GPIO)
```

A.2 TCP 协议传输

```
int TcpClientTest(const char* host, unsigned short port)
{
    int sockfd = socket(AF_INET, SOCK_STREAM, 0); // TCP socket
    int reclen=0,flag;
    char i='1';
    t_request[0]='t';
    int deg=60;
    int degR=-1;

    struct sockaddr_in serverAddr = {0};
    serverAddr.sin_family = AF_INET;  // AF_INET 表示 IPv4 协议
    serverAddr.sin_port = htons(port);  // 端口号，从主机字节序转为网络字节序
    if (inet_pton(AF_INET, host, &serverAddr.sin_addr) <= 0) {  // 将主机 IP 地址从
“点分十进制”字符串 转化为 标准格式（32 位整数）
        printf("inet_pton failed!\r\n");
        lwip_close(sockfd);
    }
     // 尝试和目标主机建立连接，连接成功会返回 0 ，失败返回 -1
    if (connect(sockfd, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) < 0)
{
        printf("connect failed!\r\n");
        lwip_close(sockfd);
        return 0;
    }
    printf("connect to server %s success!\r\n", host);

    while (1)
    {
        for(i='1';i<='3';i++)
        {   t_request[1]=i;t_request[2]='\0';
            ssize_t retval = send(sockfd, m_request, sizeof(m_request), 0);
            if (retval < 0) {
                printf("send m_request failed!\r\n");
                return 0;
```

```
            }
       printf("send m_request{%s} %ld to server done!\r\n", m_request, retval);

            while (1)
            {
                retval = recv(sockfd, response, sizeof(response), 0);
                if (retval <= 0) {
                    printf("send g_response from server failed or done, %ld!\r\n"
, retval);
                    return 0;
                }
                response[retval] = '\0';
                printf("recv g_response{%s} %ld from server done!\r\n", response,
 retval);
                reclen=(int)retval;

                if (response[0]!='a')
                {
                    break;
                }
                retval = send(sockfd, a_send, sizeof(a_send), 0);
                if (retval < 0)
                {
                    printf("send a_send failed!\r\n");
                    return 0;
                }
            }

            if(response[0]=='m')
            {
                if(response[1]=='0')
                {   if (response[2]==degR)
                    {
                    break;
                    }
                    degR=response[2];
                    deg=(int)response[2];
                    deg-=48;
                    ToDeg(deg);
                    break;
                }
                if (response[1]=='1')
                {
                    distinguish123(i);
```

```
                }
            }
            retval=UartTask(10);
            if(retval==1)
                strcpy(t_request+2,uartReadBuff);

            retval = send(sockfd, t_request, sizeof(t_request), 0);
            if (retval < 0) {
                printf("send t_request failed!\r\n");
                break;
            }
            printf("send t_request{%s} %ld to server done!\r\n", t_request, retva
l);

            osDelay(300);

        }
```

B.预警平台代码

B.1 相机畸变矫正代码

```
import cv2
import numpy as np
    datadir = dataroot + 'test'
    path = os.path.join(datadir)
    img_list = os.listdir(path)

    for i, name in enumerate(img_list):
        img = cv2.imread(os.path.join(path, name))
        height, width = img.shape[:2]
        p = cv2.fisheye.estimateNewCameraMatrixForUndistortRectify(K, D, (width,
height), None)
        mapx2, mapy2 = cv2.fisheye.initUndistortRectifyMap(K, D, None, p, (width,
 height), cv2.CV_32F)
        img_rectified=cv2.remap(
img, mapx2, mapy2, interpolation = cv2.INTER_LINEAR,borderMode = cv2.BORDER_CONST
 ANT)     # 边界的填充方式
        cv2.imwrite(dataroot + 'output/' + name, img_rectified)
```

B.2 GUI 界面关键代码 platform_gui.py

```
def __init__(self, root):
        self.root = root
        self.root.title("图片展示程序")
        var = tk.StringVar()
        # 创建左边的控制栏
        control_frame = tk.Frame(root)
        control_frame.pack(side=tk.LEFT, padx=10, pady=10)
```

```
        self.txt_frame = tk.Frame(root)
        self.txtlabel = tk.Label(self.txt_frame,textvariable=var,justify='left')
        self.txtlabel.pack(side='top')

        self.button1 = tk.Button(control_frame, text="解码检测数据
", command=self.decode)
        self.button1.pack(pady=5)

        self.button1 = tk.Button(control_frame, text="检测结果展
", command=self.display_photos_1_2_3)
        self.button1.pack(pady=5)

        self.button2 = tk.Button(control_frame, text="全景监控生成
", command=self.generate_panorama)
        self.button2.pack(pady=5)

        self.button2 = tk.Button(control_frame, text="积水图例生成
", command=self.vertical_view)
        self.button2.pack(pady=5)

        # 创建右边的展示区
        self.display_frame = tk.Frame(root)
        self.display_frame.pack(side=tk.RIGHT, padx=10, pady=10)

        self.image_label = tk.Label(self.display_frame)
        self.image_label.pack()

        self.display_label = tk.Label(self.display_frame)
        self.display_label.pack()
        bg_image = Image.open("bg.png")  # 替换为背景图片的文件路径
        bg_image = bg_image.resize((700, 400))  # 调整图片尺寸
        bg_photo = ImageTk.PhotoImage(bg_image)
        self.display_label.configure(image=bg_photo)
        self.display_label.image = bg_photo
```

```
def decode_file(self):
    second_chars = []
    third_chars = []
    axis = []
    a=[]
    filename="tcp.txt"
    # 打开文件并读取前三行数据
    with open(filename, 'r') as file:
        lines = file.readlines()[:6]
```

```
    # 遍历每行数据
    for line in lines:
        line = line.strip()  # 去除换行符和空白字符
        words = [char for char in line]
        if len(words) == 3:
            # second_char.append(words[1])  # 将第二个字符添加到第二个数组中
            second_char = words[1][0]
            if second_char == "1":
                second_chars.append("left perspective")
            elif second_char == "2":
                second_chars.append("center perspective")
            elif second_char == "3":
                second_chars.append("right perspective")

            third_char = words[2][0]
            if third_char == "0":
                third_chars.append("classify:noponding")
            elif third_char == "1":
                third_chars.append("classify:ponding")
            else:
                third_chars.append(third_char)
        else:
            axis.append(line[2:])
    return second_chars, third_chars, axis
def decode(self):
    self.clear_display()
    self.state_display("解码成功！")
    second_chars,third_chars, axis=self.decode_file()
    for i in range(len(second_chars)):
        a=(second_chars[i],'\n',third_chars[i],",\ndetect obj:\n",axis[i])
        a=''.join(a)
        self.state_display(a)
```

```
def generate_panorama(self):
    self.clear_display()  # 清空展示区
    # 图片路径列表
    imgs=["left_pre.jpg","center_pre.jpg"]
    stitcher = stitching.Stitcher()
    panorama = stitcher.stitch(imgs)
    cv2.imwrite("cat.png", panorama)

    imgs=["center_pre.jpg","right_pre.jpg"]
    stitcher = stitching.Stitcher()
    panorama = stitcher.stitch(imgs)
    cv2.imwrite("cat2.png", panorama)
```

```
    if os.path.exists("cat.png") and os.path.exists("cat2.png"):
        imgs=["cat.png","cat2.png"]
        stitcher = stitching.Stitcher()
        panorama = stitcher.stitch(imgs)
        cv2.imwrite("panorama.png", panorama)
    else:
        print("panorama generate fail")
    image2 = Image.open("panorama.png")  # 替换为照片 2 的文件路径
    image2 = image2.resize((1078, 320))  # 调整照片大小
    photo2 = ImageTk.PhotoImage(image2)
    label2 = tk.Label(self.display_frame, image=photo2)
    label2.image = photo2
    label2.pack(side=tk.LEFT)
    self.state_display('panorama 生成成功!')
    os.remove('cat.png')
    os.remove('cat2.png')
```