江蘇理工學院
JIANGSU UNIVERSITY OF TECHNOLOGY

机器人集成应用综合实践

垃圾识别与分拣系统设计-实验报告

学院名称：	机械工程学院
专　　业：	机器人工程
班　　级：	20 机器人工程 2
学　　号：	2020326204
姓　　名：	张庭灿
指导老师：	刘文汇、强红宾
实训地点：	45-2-213

二〇二三 年 十 月

1、任务书

江苏理工学院 机器人集成应用综合实践 任务书

<table>
<tr><td>学　院</td><td colspan="2">机械工程学院</td><td>班 级</td><td>20 机器人工程 2</td><td>姓　名</td><td>张庭灿</td></tr>
<tr><td rowspan="2">题　目</td><td>设 计</td><td colspan="3">垃圾识别与分拣系统设计</td><td rowspan="2">指导教师</td><td rowspan="2">刘文汇
强红宾</td></tr>
<tr><td>论 文</td><td colspan="3">基于 Resnet18 的垃圾识别与分拣系统设计</td></tr>
<tr><td>主要内容及基本要求</td><td colspan="6">主要内容及要求：
1. 运用 Python 语言与 PyCharm 软件编写程序；
2. 运用 Opencv 库与深度学习算法设计垃圾识别程序；
3. 采集两类垃圾数据并对其进行训练；
4. 调整参数优化算法检测精度；
5. 设计机械臂运动控制程序；
6. 设计机器人界面显示流程与检测结果；
7. 撰写实训报告。</td></tr>
<tr><td>参考文献</td><td colspan="6">[1]姚峰林，詹海英，李元宗．机器视觉中的边缘检测技术研究［J］．机械工程与自动化，2005(1) : 108－110．
[2]刘志刚，鲍加贞，汤时虎．基于 VC 的最小二乘拟合圆在 LAMOST 中的应用［J］．现代制造工程，2008(1) : 94－96．
[3]吴文琪，孙增圻． 机器视觉中的摄像机定标方法综述［J］．计算机应用研究，2004，21(2) : 4－6．
[4]龚中良,杨张鹏,梁力.基于机器视觉的柑橘表面缺陷检测[J].江苏农业科学,2019,236-237．
[5]黄登未，汪西原，王胜男．基于阈值标记的分水岭算法道路提取[J].图像与信号处理,2017,198-203.</td></tr>
</table>

2 任务背景

智能是知识和智力的总和，通过人与智能机器的合作共事，可以去扩大、延伸和部分地取代人类专家在制造过程中的脑力劳动。

当今，制造系统正在由原先的能量驱动型转变为信息驱动型，这就要求制造系统更加智能，否则难以处理如此大量而复杂的信息工作量的。其次，瞬息万变的市场需求和激烈竞争的复杂环境，也要求制造系统表现出更高的灵活、敏捷和智能。基于此，智能制造受到越来越高度的重视，也得到了更广度的使用。

图 2.1 机械臂在智能制造中的应用

从身边的例子来看， 如今的年轻一代普遍不愿进厂工作使得工业分拣工作这一耗费精力，而技术含量不高的工作处境艰辛。面对这一问题，AI 智能分拣带来的人力成本的下降和工作效率的提高成为物流业中一个划时代突破。企业采用这种基于视觉识别的形状识别技术使工作效率不断提高，不仅可以节省空间，也可以提高商品向外配送的速度。而各种优势之下，由于这一工作中影像、高速率传输对网

路频宽的需求，对控制系统网路化也形成了一定的要求。工业生产流程与网路传递的封包资讯的严谨性，网路上传递资讯多样化，都要靠控制系统的智慧能力，进行适当的判断才能得以实现。这都是等待着我们逐步去实现与完善的。

3、方案搭建思路

方案流程图如图 3.1 所示。

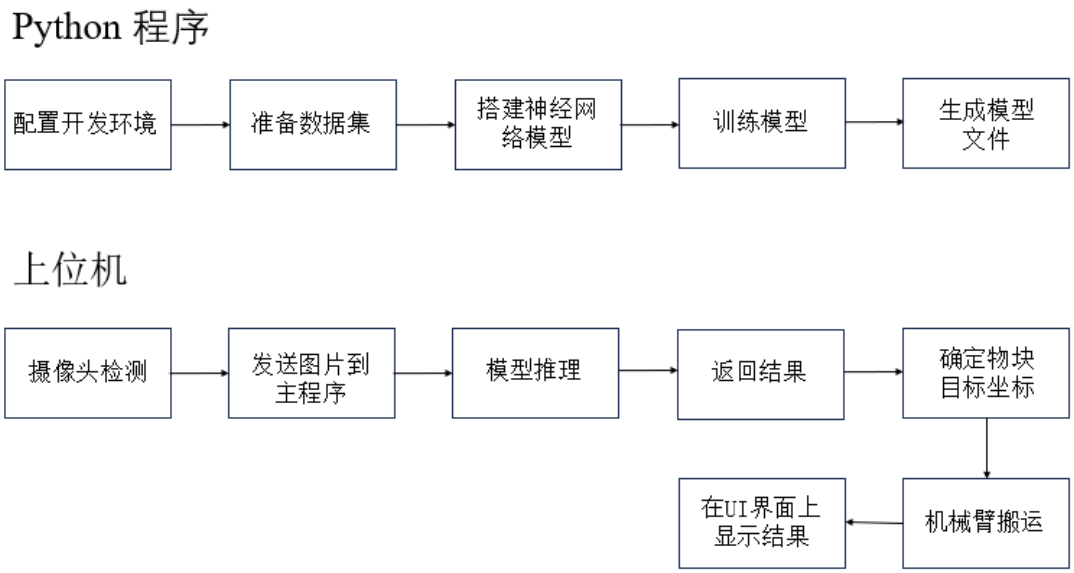

图 3.1 方案流程图

4、垃圾分类程序设计

4.1 硬件设计

4.1.1 机械臂功能简述

Dobot Magician机械臂是一款桌面级智能机械臂，支持示教再现、

脚本控制、Blockly 图形化编程、写字画画、激光雕刻、3D 打印、视觉识别等功能，还具有丰富的 I/O 扩展接口，供用户二次开发时使用。具备 3D 打印、激光雕刻、写字画画等多种功能，预留 13 个拓展接口支持二次开发，用户可以通过软件编程结合硬件拓展来开发更多的应用场景。

4.1.2 机械臂结构与外观

Dobot Magician 机械臂由底座、大臂、小臂、末端工具等组成，外观如下图 4.1 所示。

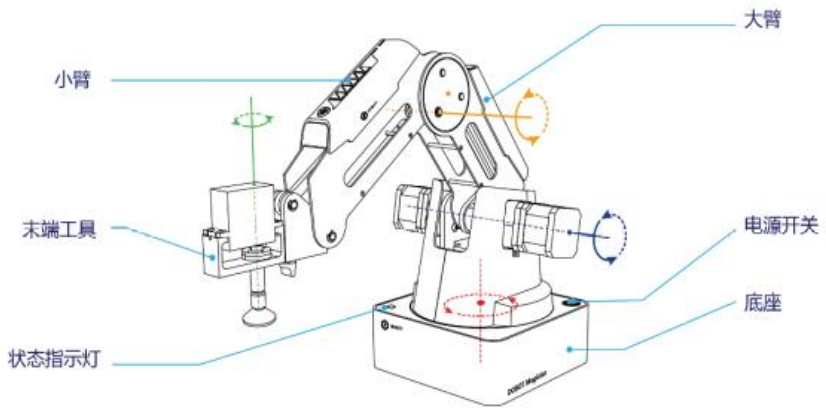

图 4.1 Dobot Magician 外观示意图

Dobot Magician 机械臂的标准三视图如图 4.2 所示。

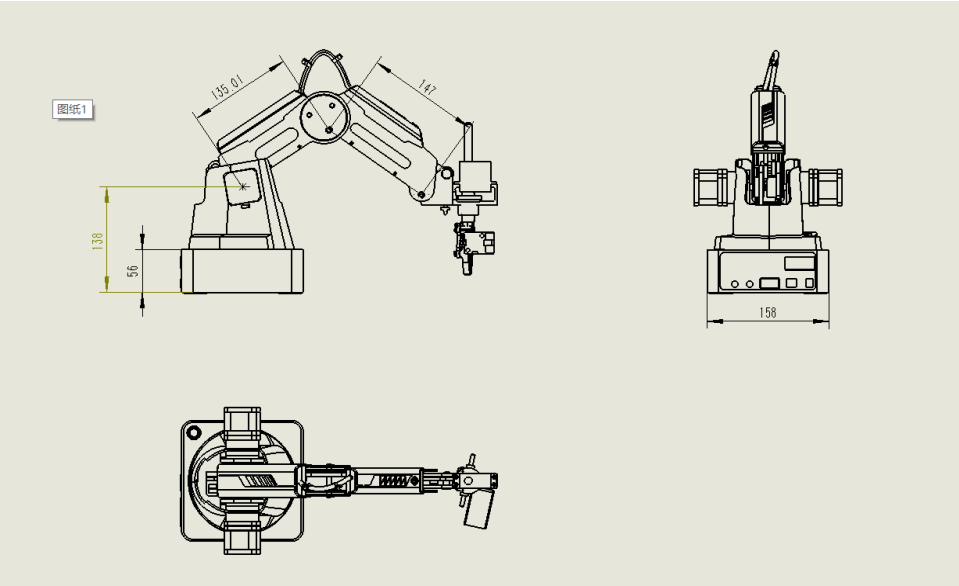

图 4.2 Dobot Magician 机械臂的标准三视图

Dobot Magician 的尺寸参数如图 4.3，其末端安装孔尺寸参数如图 4.4 所示。

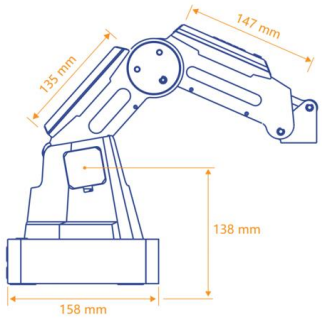

图 4.3 Dobot Magician 机械臂的尺寸参数

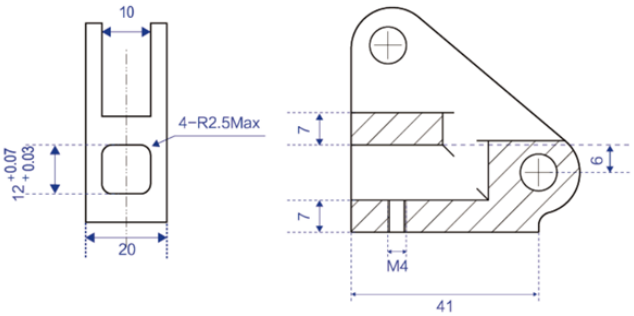

图 4.4 Dobot Magician 机械臂末端安装孔尺寸参数

Dobot Magician 机械臂的工作空间如图 4.5 所示。

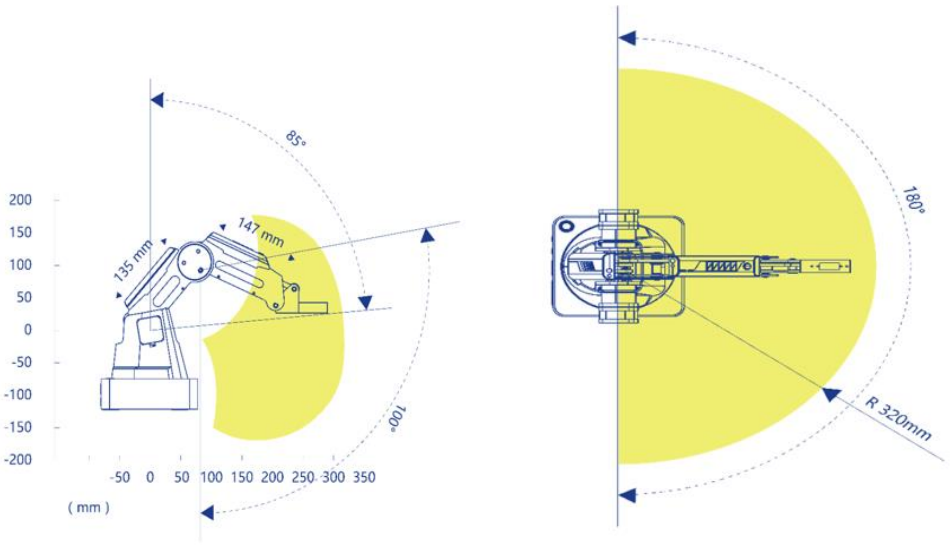

图 4.5 Dobot Magician 机械臂的工作空间

4.1.3 摄像头

摄像头采用海康威视 MV-CE050-30UC，500 万像素 USB3.0 面阵相机，AR0521，彩色。支持自动或手动调节增益、曝光时间、白平衡、Gamma 校正、LUT 等。支持自定义 ROI，支持水平镜像和垂直镜像。支持硬触发、软触发以及自由运行模式。结构紧凑，外形尺寸 29 mm×29 mm×30 mm，适用于较小的安装要求。兼容 USB3 Vision 协议及 GenlCam 标准，可接入第三方软件平台。最大帧率 44.7fps，动态范围 60dB，信噪比 40dB. 像素格式: Mono 8/10/12、Bayer GR 8/10/10p/12/12p、YUV422_YUYV_Packed，YUV 422 Packed、RGB 8，BGR 8。

图 4.6 海康威视 MV-CE050-30UC

4.2 软件设计

4.2.1 信息通讯原理

通讯协议又叫通讯规程，指的是通讯的双方为了传输一个信息而制定的一种规定。规定中包含了数据的格式，同步的方式，传输的速度以及检查，纠错的方式，加密方式等等。现在已经发展出了足够成熟的通讯协议，常见的有 TCP/IP 协议，HTTP 协议，USB 协议等等。

本项目采用 TCP/IP 协议，通过创建 socket 客户端来实现机械臂 A、B 和光源之间的信息传递。

4.2.2 TCP/IP 传输协议介绍

TCP/IP 传输协议，即传输控制/网络协议，即网络通讯协议，它是在网络的使用中最基本的通讯协议，其原理也较为简单。TCO/IP 传输协议对互联网中各部分进行通信的标准和方法进行了规定。并且，TCP/IP 传输协议是保证网络数据信息及时、完整的两个重要的协议。TCP/IP 传输协议严格来说是一个四层的体系结构，包含应用层、传输层、网络层和数据链路层。其大体结构如图 4.7 所示。

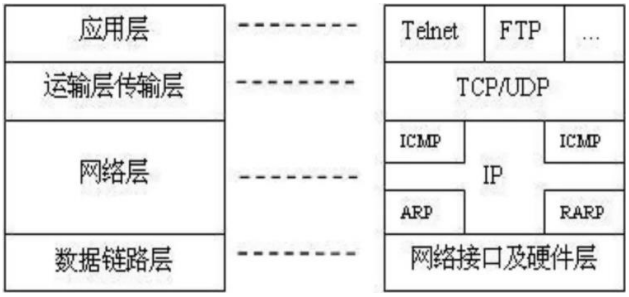

图 4.7 TCP/IP 传输协议

在两个进程相互通讯时，最基本的一个前提是能够唯一地表示一个进程。在本地通讯中，我们可以使用 PID 来唯一标示一个进程，而在网络中，两个进程 PID 冲突几率很大，使其不具有唯一性。此时便显示出 TCP/IP 的优越性了。IP 层的 IP 地址可以唯一标示主机，而 TCP 层协议端口号可以唯一标示主机的一个进程，这样就可以利用 ip 地址+协议+端口号唯一标示网络中的一个进程。在此基础上创建一个 socket，两个进程便可以通讯了。

Socket，常翻译为套接字，是在应用层和传输层之间的一个抽象层，它把 TCP/IP 层复杂的操作抽象为几个简单的接口供应用层使用以实现进程在网络中的通信。其作用后的整体通讯流程如图 4.8 所示。

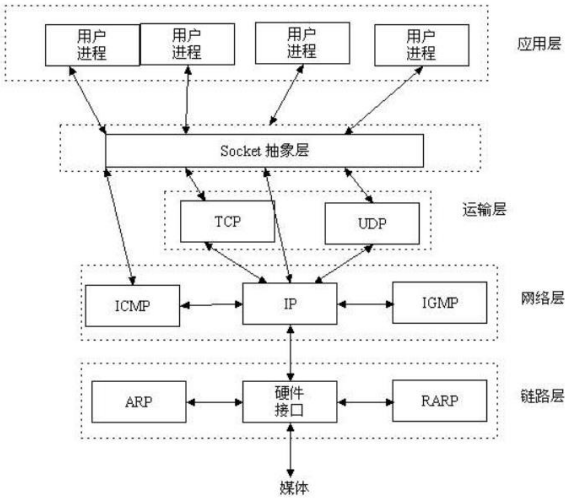

图 4.8 通讯流程图

具体到本项目中，先在机械臂上创建一个 socket 客户端变量，将其 IP 设定为 127.0.0.1（本机地址），端口设为 4000，并连接。在此过程中，需新增一个端口 7890，用于发送与接收识别精度。

同时，需在 PC 端上创建一个 TCP 服务端，IP 与机械臂的一致，通过发送”ok”指令使主程序开始运行，当相机检测到目标后，将图像传送给 PC 端，PC 端进行视觉识别，将最后的结果发送给机械臂，机械臂将其吸起并运送到指定位置。

4.2.3 视觉识别

（1）模板匹配

在系统使用前，应对其进行标定，通过模板匹配实现摄像头对点位的识别。

（2）数据集准备及模型训练

对于已给的材料（需被搬运的物块），使用 MV-CE050-30UC 相机对其在不同光照条件下进行多角度、多姿态的拍摄，将拍摄的图片以前后顺序作为文件名，将拍摄的图片保存到本地文件夹。

本项目规定使用 Pytorch 平台进行神经网络的搭建及训练。在 Pytorch 平台中，首先读取数据集内的全部图片，抽取其中的 20%作为验证集单独放入一个新文件夹并进行随机排序。根据目标物将图片分到四个文件夹里，并贴上相应的标签。为保证数据的多样性，在将其用作训练数据前，使用数据增强对原始训练集进行处理。

接下来是对数据的预处理：创建训练集列表、设置图片大小、一次训练抓取的样本数量（batch_size），并判断 GPU 是否可用；若不

可用，则调用 CPU 进行模型训练。完成后，使用 plt 库函数展示一些经过预处理后的图片，观察是否符合我们的预期。若符合，使用 numpy 函数将预处理过后的图片转化为矩阵形式。

接下来创建训练模型函数：定义优化器、损失函数、学习周期等一系列参数，通过梯度下降、迭代等方式实现模型的高精度。网络结构选用传统的 resnet18 网络，通过可视化模型来判断是否存在过拟合或其他影响模型精度的情况。最后，导出模型并存储。

4.2.4 程序设计流程图

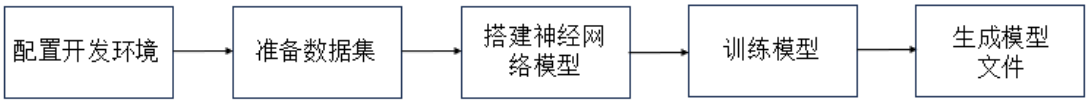

图 4.9 程序设计流程图

4.2.5 核心代码与参数

准备数据集时，使用如图 4.10 所示程序进行图片的分类、裁剪。

```
import cv2
import os

input_folder = "./train2/box1"
output_folder = "./train2/boxCut"

if not os.path.exists(output_folder):
    os.makedirs(output_folder)

for filename in os.listdir(input_folder):
    if filename.endswith(".bmp"):
        image = cv2.imread(os.path.join(input_folder, filename))

        height, width = image.shape[:2]
        crop_image = image[500:height//2 + 600, 800:width//2 + 600]

        cv2.imwrite(os.path.join(output_folder, filename), crop_image)
```

图 4.10 准备数据集

```python
# 数据集存放路径
path = 'C:/Users/Croon/PycharmProjects/Robotic_Arm/Object_Recognition/data'

# 遍历数据集
for folder in ["banana", "plate", "cola", "water"]:
    # 图片格式为.jpg或.png
    jpg_files = glob.glob(os.path.join(path, "train", folder, "*.bmp"))
    print(jpg_files)
    png_files = glob.glob(os.path.join(path, "train", folder, "*.png"))
    files = jpg_files + png_files
    # 统计训练集数据
    num_of_img = len(files)
    print("Total number of {} image is {}".format(folder, num_of_img))
    # 从训练集里面抽取20%作为验证集
    shuffle = np.random.permutation(num_of_img)
    percent = int(num_of_img * 0.2)
    print("Select {} img as valid image".format(percent))
    # 新建val文件夹存放验证集数据
    path_val = os.path.join(path, "val", folder)
    if not os.path.exists(path_val):
        os.makedirs(path_val)
    # 把训练集里面抽取20%的数据复制到val文件夹
    # shuffle()方法将序列的所有元素随机排序
    for i in shuffle[:percent]:
        print("copy file {} ing".format(files[i].split('\\')[-1]))
        shutil.copy(files[i], path_val)
```

图 4.11 遍历数据集

```python
# 数据增强与变换
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.Grayscale(3),
        transforms.RandomRotation(5),
        transforms.RandomVerticalFlip(0.5),
        transforms.RandomHorizontalFlip(0.5),
        transforms.ToTensor(),
        transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.Grayscale(3),
        transforms.RandomRotation(5),
        transforms.RandomVerticalFlip(0.5),
        transforms.RandomHorizontalFlip(0.5),
        transforms.ToTensor(),
        transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])
    ])
}
```

图 4.12 数据增强

```python
# 加载resnet18网络结构
model_ft = torchvision.models.resnet18(pretrained=False)
# 加载resnet18网络参数
model_ft.load_state_dict(torch.load('./model/resnet18.pth'))
# 提取fc层中固定的参数
num_ftrs = model_ft.fc.in_features
# 重写全连接层 4种分类
model_ft.fc = nn.Linear(num_ftrs, 4)
model_ft = model_ft.to(device)
# 定义一个损失函数和优化器
# 这里使用分类交叉熵Cross-Entropy作为损失函数，动量SGD作为优化器
criterion = nn.CrossEntropyLoss()
# 初始化优化器
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)
# 每7个epochs衰减LR通过设置gamma = 0.1
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
# 开始训练模型 默认迭代100次
model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler, num_epochs=50)
# 可视化预测模型
visualize_model(model_ft)
# 保存模型
torch.save(model_ft.state_dict(), './model/Model-face.pkl')
```

图 4.13 设置神经网络参数

4.2.6 实验结果与分析

模型精度如图 4.14 所示。

```
Epoch 49/49
----------
train Loss: 0.0107 Acc: 0.9967
val Loss: 0.1280 Acc: 0.9395

Training complete in 10m 5s
Best val Acc: 0.996139
```

图 4.14 模型训练结果

5、机械臂控制程序设计

5.1 程序设计流程图

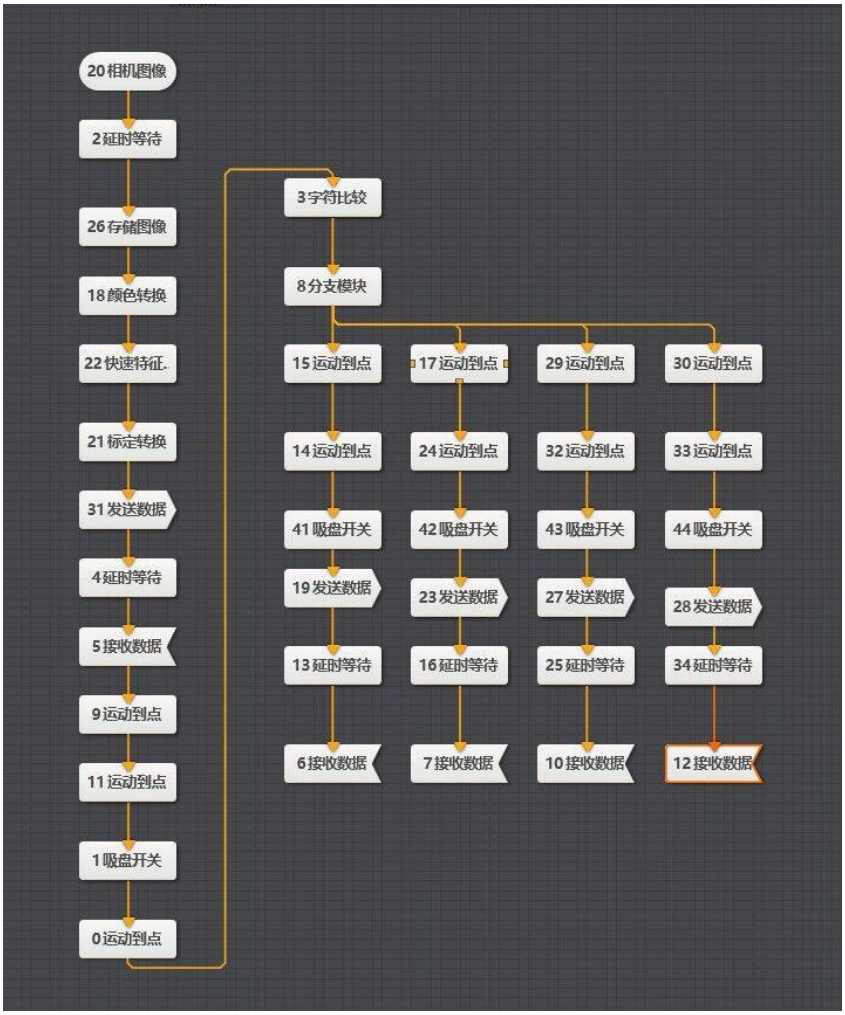

图 5.1 Dobot 程序流程图

5.2 核心设置与参数

本次实验需要将两个软件进行 TCP 通讯，所有需要在 Dobot 中创建好 TCP 通讯的服务端即如下图 4.2 所示 TCP 服务端_上料机器人，设定好通信参数并且确保 PyCharm 代码所创建的 TCP 客户端 IP 与端口是否一致。通过 TCP 助手测试确保通讯正常，双端能够互相

发送和接受数据。

图 5.2 通讯管理

创建好整个流程中所需要使用到的全局变量，如图 5.3 所示

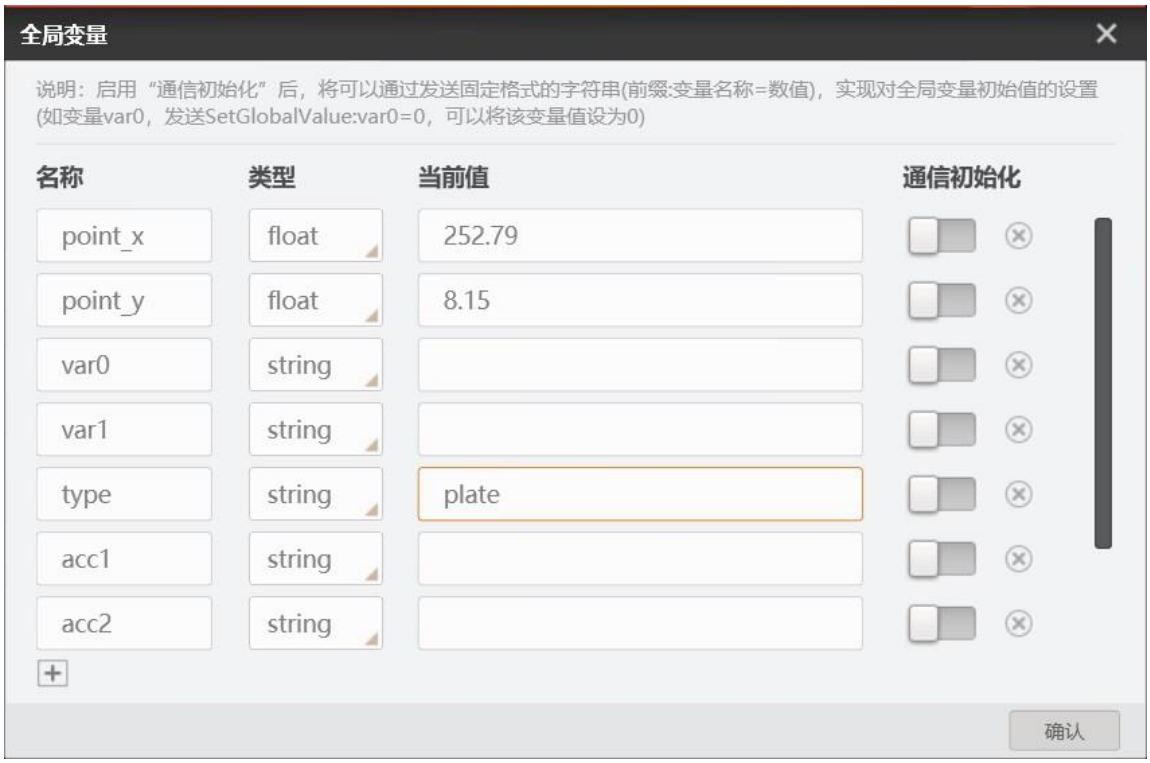

图 5.3 全局变量

根据实验目标，首先需要通过相机采集 RGB24 的垃圾图像，存

储图像到 PyCharm 识别程序读所取的路径，在对图像进行颜色处理，查找快速特征，对垃圾图像位置进行查找，通过标定转换得到垃圾图像所在位置坐标，此时再发送数据 ok 给 PyCharm 端如图 5.4，这时 PyCharm 收到 ok 执行识别程序，得到结果发送数据，把发送数据发送给 Dobot 端，将数据输入到全局变量 var1 中如图 5.5 所示。然后在进行垃圾的抓取。通过字符比较图 5.8 与分支模块图 5.9，将所对全局变量 var1 所写入的数据进行比较，将四种垃圾依次对应一种引索数字，分支模块根据字符比较所得到的引索数字运行对应的流程，从而实现垃圾分类的功能，从而将垃圾抓取归类在不同的区域，实现实验目标。而图 5.6 与图 5.7 的数据发送接收则是作用在运行界面，用来反馈识别结果与识别精度

图 5.4 发送数据 1

图 5.5 接收数据 1

图 5.6 发送数据 2

图 5.7 接收数据 2

图 5.8 字符比较

图 5.9 分支模块

5.3 系统运行的界面

搭建如下图 5.10 所示的运行界面，清晰反应出所示的物品，以及四种垃圾所对应的识别精度。识别精度的显示则是创建了一个新的

TCP 端口用来传输识别精度，如图 5.11 所示。然后通过图 5.7，将接收到的数据写入全局变量 acc1，从而在运行界面实现反馈，以此类推创建 4 个全局变量，从而达成图 5.10 的效果

图 5.10 运行界面

图 5.11 TCP 端口 2

5.4 实验结果与分析

实验结果如下图所示，识别精度较高，并且能够分类放置，从而实现垃圾分类的目的。

图 5.11 水瓶识别结果及精度

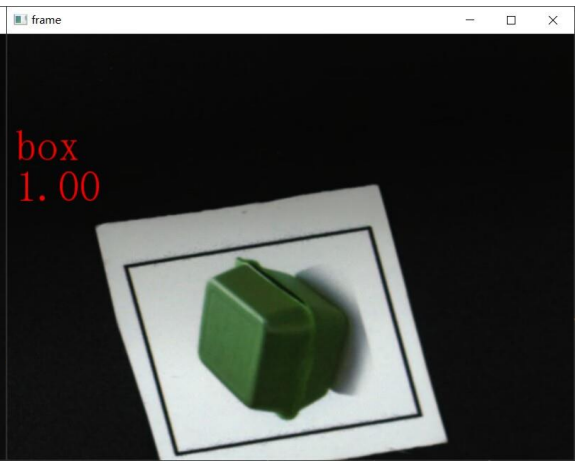

图 5.12 盒子识别结果及精度

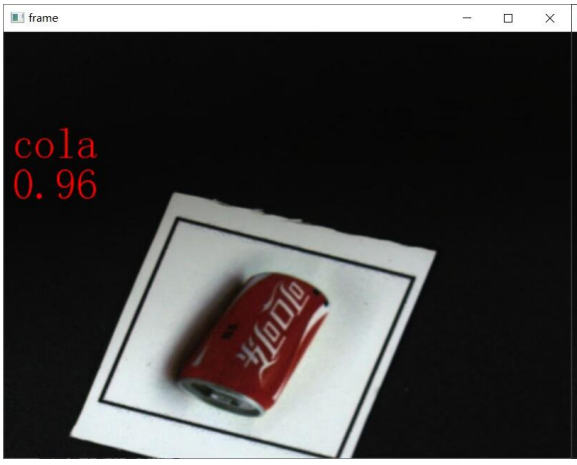

图 5.13 可乐识别结果及精度

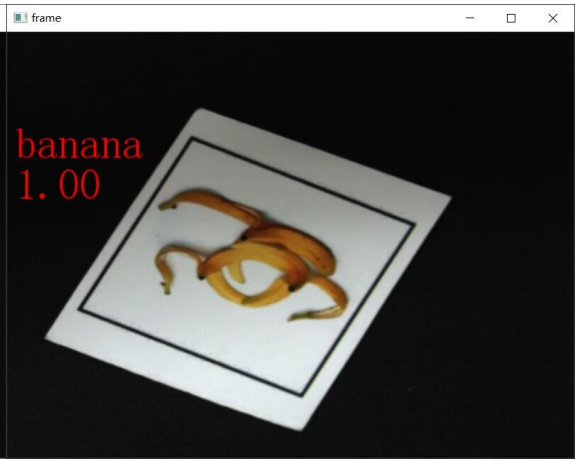

图 5.14 香蕉识别结果及精度

6、发现问题与总结

本次实验遇到以下一些问题：

1. 在程序第一次运行时候会运行到一半断开，重新继续运行，

随后就会产生每次程序执行都是上次接受到的识别结果，但是在TCP 接收端口则是正常的接收到了数据。从而导致程序会将使用上一次运行结果进行流程，这也是为什么程序每次打开后第一次执行会发生错误重新拍摄运行，也就是因为没有接收到数据。解决办法则是在接收数据前添加一个 5s 的延时等待；

2. 运行程序时候会发现突然卡顿，或者各种各样停顿，吸盘卡顿等等，将多余流程删除，只保留主要的流程，可以解决大部分的程序问题，因为 Dobot 的程序设计上会运行所有的程序，所有会存在问题冲突；

3. 分支模块理解问题，分之模块作用就是从这个模块分支出去，并不需要设置 4 个分支模块，设置四个分支模块只会产生冲突；

4. 图像存在路径错误，导致识别程序无法运行；

5. 相机设置错误，并没有使用 RGB24 模式。

6. 使用 Cuda-GPU 训练模型的速度是使用 CPU 的至少五倍。