第11章 综合实训案例解析

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 人工智能基础与应用 | | 章名 | | | 综合实训案例解析 | |
| 教学内容 | 综合实训案例解析2 | | | | 课时 | 2 | |
| 项目性质 | □演示性 □验证性 □设计性 √综合性 | | | | | | |
| 授课班级 |  | 授课日期 | |  | | 授课地点 |  |
| 教学目标 | 完成基于机械臂的工业分拣系统项目实现 | | | | | | |
| 教学内容 | 基于机械臂的工业分拣系统项目实现 | | | | | | |
| 教学重点 | 基于机械臂的工业分拣系统 | | | | | | |
| 教学难点 | 基于机械臂的工业分拣系统 | | | | | | |
| 教学准备 | 装有Python的计算机  教学课件PPT  教材：《人工智能基础与应用（微课版）》 | | | | | | |
| 作业设计 |  | | | | | | |

教学过程

|  |  |
| --- | --- |
| **教学环节** | **教学内容与过程**  **（教学内容、教学方法、组织形式、教学手段）** |
| **课前组织** | 做好上课前的各项准备工作（打开计算机、打开课件、打开软件、打开授课计划、教案等），吸引学生注意力。 |
| **课程说明** | 【课前说明】  回顾基于机械臂的工业分拣系统项目概要与项目设计。  【目的】  使学生从了解本节课的学习目标、学习重点、考评方式等方面明确课程学习的要求和目标。 |
| **课程内容描述** | 11.1.3 项目实现  这里主要针对嵌入式AI运算单元综合程序的Qt主线程以及神经网络识别前的图像预处理做详细描述，离线的图像识别网络是两层的CNN模型，在线的语音识别调用百度AI开放平台的API，这里不再赘述。  1．Qt主线程  在Qt主线程中需要实现界面初始化，界面的所有显示控制都在主线程中实现，打开串口实现嵌入式AI控制单元数据的收发，并开启人工智能线程、串口线程和UDP线程，同时对用户的控制做出应对和对各个线程传来的数据进行处理。  首先初始化界面，在打开该应用后需要对各个组件进行初始化。  void MainWindow::InitInterface()  {  //原标题隐藏  this->setWindowFlags(Qt::FramelessWindowHint);  //新建一个QListView，可以使comboBox下拉框变宽  ui->comboBoxSelectBox1->setView(new QListView());  ui->comboBoxSelectBox2->setView(new QListView());  ui->comboBoxStart->setView(new QListView());  ui->comboBoxEnd->setView(new QListView());  //图像加载，并显示  img0->load(":/picture/0.png");  img1->load(":/picture/01.png");  img2->load(":/picture/02.png");  img3->load(":/picture/03.png");  img4->load(":/picture/04.png");  img5->load(":/picture/05.png");  img6->load(":/picture/06.png");  img7->load(":/picture/07.png");  img8->load(":/picture/08.png");  img9->load(":/picture/09.png");  imgStart->load(":/picture/loading.png");  ui->label1\_1\_1->setPixmap(QPixmap::fromImage(\*img0));  ui->label1\_1\_2->setPixmap(QPixmap::fromImage(\*img1));  ui->label1\_1\_3->setPixmap(QPixmap::fromImage(\*img2));  ui->label1\_1\_4->setPixmap(QPixmap::fromImage(\*img3));  ui->label1\_2\_1->setPixmap(QPixmap::fromImage(\*img4));  ui->label1\_2\_2->setPixmap(QPixmap::fromImage(\*img5));  ui->label1\_2\_3->setPixmap(QPixmap::fromImage(\*img6));  ui->label1\_2\_4->setPixmap(QPixmap::fromImage(\*img7));  //图像加载，并显示  imgRight->load(":/picture/right.png");  ui->label7->setPixmap(QPixmap::fromImage(\*imgRight));  //图像加载，并显示  imgRight->load(":/picture/left.png");  ui->label8->setPixmap(QPixmap::fromImage(\*imgRight));  //设置spin的范围为0～250  ui->spinBox1\_1->setRange(0,250);  //设置spin不能手动输入  ui->spinBox1\_1->setFocusPolicy(Qt::NoFocus);  ui->spinBox1\_2->setRange(0,250);  ui->spinBox1\_2->setFocusPolicy(Qt::NoFocus);  ui->spinBox1\_3->setRange(0,250);  ui->spinBox1\_3->setFocusPolicy(Qt::NoFocus);  ui->spinBox1\_4->setRange(0,250);  ui->spinBox1\_4->setFocusPolicy(Qt::NoFocus);  ui->spinBox1\_5->setRange(0,250);  ui->spinBox1\_5->setFocusPolicy(Qt::NoFocus);  ui->spinBox1\_6->setRange(0,250);  ui->spinBox1\_6->setFocusPolicy(Qt::NoFocus);  //spin失能  SpinEnable(false);  //仓库一的显示内容为空  Box1DisplayImagePredict("1F2F3F4F");  //仓库二的显示内容为空  Box2DisplayImagePredict("1F2F3F4F");  //录音按键样式  ui->label2\_1->setStyleSheet("border:2px groove gray;\  border-radius:10px;background-color:rgb(255, 255, 0)");  //录音按键样式  ui->label2\_2->setStyleSheet("border:2px groove gray;\  border-radius:10px;background-color:rgb(255, 255, 0)");  }  在Qt中，信号槽机制是非常实用的一个技术。当某个事件发生之后，它就会发出一个信号（Signal），如果有对象对这个信号感兴趣，它就会使用连接（connect）函数将想要处理的信号和自己的一个函数（槽函数slot）绑定，以处理这个信号。  这里需要将主线程和各个子线程的数据传输用信号槽进行连接。  void MainWindow::InitSignal()  {  //摄像头显示图像的不同格式按键被按下时对应的不同颜色效果  connect(this, SIGNAL(TriggerDisplayStatus(char)), this,\  SLOT(DisplayStatus(char)));  qRegisterMetaType<MsgSerial>("MsgSerial");  qRegisterMetaType<MsgPython>("MsgPython");  qRegisterMetaType<MsgPython>("MsgUdp");  t\_serial = new UartThread();  t\_python = new PythonThread();  t\_udp = new UdpThread();  //主线程接收串口线程数据  connect(t\_serial, SIGNAL(SerialSendResult2Main(MsgSerial)), \  this, SLOT(GetSerialData(MsgSerial)));  //主线程往串口线程发送serialPort  connect(this, SIGNAL(TriggerPort(MsgSerial)), t\_serial, \  SLOT(RecvPort(MsgSerial)));  //主线程接收Python线程的数据  connect(t\_python, SIGNAL(PythonSendResult2Main(MsgPython)), \  this, SLOT(GetPythonData(MsgPython)));  //主线程接收Python线程模块加载成功信息  connect(t\_python, SIGNAL(StartProgrammer(bool)), this, \  SLOT(PythonModuleLoadSuccess(bool)));  //主线程往Python线程发送识别方式  connect(this, SIGNAL(TriggerPredict(MsgPython)), t\_python, \  SLOT(RecvPredict(MsgPython)));  //主线程接收UDP线程数据  connect(t\_udp, SIGNAL(UdpSendResult2Main(MsgUdp)), this, \  SLOT(GetUdpData(MsgUdp)));  //主线程往串口线程发送socket  connect(this, SIGNAL(TriggerSocket(MsgUdp)), t\_udp, \  SLOT(RecvSocket(MsgUdp)));  //开启串口线程  t\_serial->start();  //开启Python线程  t\_python->start();  //开启UDP线程  t\_udp->start();  }  当串口线程通过状态机接收到串口数据后，由于需要将数据传输到主线程以完成数据处理，所以需要实现一个处理串口线程数据的函数。  void MainWindow::GetSerialData(MsgSerial msgSerial)  {  // qDebug("recv uart : %02x",msgSerial.int\_SerialInfo);  switch (msgSerial.int\_SerialInfo) {  //串口接收到机械臂发送来的当前的6个参数内容  case 0x92:  //在spin上显示6个数据  DisplayStatusInSpinBox(msgSerial.str\_SerialInfo);  break;  //串口接收到停到仓库一（0x52）和仓库二（0x53）位置消息后开始1s计时  case 0x52:  case 0x53:  //计时器  timerSavePic = new QTimer(this);  //设置间隔：1000ms  timerSavePic->setInterval(1000);  //一旦超时则触发SavePic(),即保存一帧图像  //SIGNAL(timeout())：每当计时结束时，计时器归零并重新计时，并发送一个信号激活SLOT()函数  connect(timerSavePic,SIGNAL(timeout()),this,SLOT(SavePic()));  //开始计时  timerSavePic->start();  break;  //货物抓取成功后返回的标志位  case 0x72:  MoveGoodsSuccess();  break;  //在排序过程中机械臂返回的货物的位置号  case 0x71:  DisplaySortInBox(msgSerial.str\_SerialInfo);  break;  //货物排序完成后，标志位清零  case 0xb3:  predictImageBoxAndSort = DONOTSORT;  break;  //显示出单片机的地址  case 0xa1:  ui->label\_displayIP\_2->setText(msgSerial.str\_SerialInfo);  break;  default:  break;  }  }  人工智能线程在发送识别命令后，如果接收到返回的识别结果，需要将结果传输到主线程完成显示及控制。  void MainWindow::GetPythonData(MsgPython msgPython)  {  switch (msgPython.int\_PythonInfo2Main) {  case CANNOTLOADPYTHON: //提示未能成功加载Python模块  QMessageBox::warning(NULL, "警告", "未能加载Python模块!", \  QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  break;  case CANNOTLOADMAIN: //提示未能成功加载主函数模块  QMessageBox::warning(NULL, "警告", "未能加载主函数，请检查路径!", \  QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  break;  case CANNOTLOADIMAGEPREDICT: //提示未能成功加载图像识别函数  QMessageBox::warning(NULL, "警告", "未能成功加载图像识别函数!", \  QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  break;  case CANNOTLOADVOICESTART: //提示未能成功加载开始录音函数  QMessageBox::warning(NULL, "警告", "未能成功加载开始录音函数!", \  QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  break;  case CANNOTLOADVOICESHORTPREDICT: //提示未能成功加载简易控制命令识别函数  QMessageBox::warning(NULL, "警告", "未能成功加载简易控制命令识别函数!", \  QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  break;  case CANNOTLOADVOICELONGPREDICT: //提示未能成功加载搬运控制命令识别函数  QMessageBox::warning(NULL, "警告", "未能成功加载搬运控制命令识别函数!", \  QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  break;  case IMAGEPREDICT1: //仓库一（图像不旋转）图像识别成功标志位  //在仓库一的界面上显示结果  Box1DisplayImagePredict(msgPython.str\_PythonInfo2Main);  //如果是AR的识别  if(dealARFlag){  //以UDP单波的形式将识别后的结果发送出去  SendUniCast2AR(sendToARPlace + " " + box1OriginalPlace1.right(2) \  + " " + box1OriginalPlace2.right(2) + " " + \  box1OriginalPlace3.right(2) + " " + box1OriginalPlace4.right(2));  //将界面上的显示清空  Box1DisplayImagePredict("1F2F3F4F");  //清除标志位  dealARFlag = false;  }  if(predictImageBoxAndSort == ASCENDING){  //识别成功后，往串口发送以升序排序信号  WriteSerialData(sendDataToSort + "61" + msgPython.str\_ PythonInfo2Main);  }else if(predictImageBoxAndSort == DESCENDING){  //识别成功后，往串口发送以降序排序信号  WriteSerialData(sendDataToSort + "62" + msgPython.str\_ PythonInfo2Main);  }  break;  //仓库二（图像旋转180°）图像识别成功标志位  case IMAGEPREDICT2:  //在仓库二的界面上显示结果  Box2DisplayImagePredict(msgPython.str\_PythonInfo2Main);  break;  //短语音识别结束  case VOICEPREDICTSHORT:  //识别结果显示在左侧的label  if(msgPython.str\_PythonInfo2Main != "ab"){  ui->label2\_1->setText(" 成功");  WriteSerialData(sendDoAction + msgPython.str\_PythonInfo2Main);  }else{  ui->label2\_1->setText(" 失败");  }  break;  //长语音识别结束  case VOICEPREDICTLONG:  //识别结果显示在左侧的label  if(msgPython.str\_PythonInfo2Main != "ab"){  ui->label2\_2->setText(" 成功");  WriteSerialData(sendGoodsMoveData + msgPython.str\_ PythonInfo2Main);  }else{  ui->label2\_2->setText(" 失败");  }  break;  default:  break;  }  }  2．图像预处理  将拍摄到的原始图像预处理为识别的图像，这个阶段需要对原始图像中的数字物体进行定位、切割等操作。  import cv2  import numpy as np  import sys  import os  def position(pic):  arr = []  # 复制图像，防止图像被更改  img = pic.copy()  # 将RGB转换为HSV  img\_hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)  lower\_red = np.array([20, 43, 46])  upper\_red = np.array([40, 255, 255])  # 获取每一个像素点的HSV值，将黄色像素点转换为黑色像素点，其余为白色  img = cv2.inRange(img\_hsv, lower\_red, upper\_red)  ret, img=cv2.threshold(img,127,255,cv2.THRESH\_BINARY)  # 进行闭操作  kenel = cv2.getStructuringElement(cv2.MORPH\_RECT, (3, 5))  close = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kenel, iterations=1)  # 查找轮廓，只检查外轮廓  binary,contours,hierarchy = cv2.findContours(\  close, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  for i in range(len(contours)):  cnt = contours[i]  # 计算该轮廓的面积  area = cv2.contourArea(cnt)  # 面积小的都筛选掉  if (area < 1800):  continue  # 轮廓近似，作用很小  epsilon = 0.001 \* cv2.arcLength(cnt, True)  approx = cv2.approxPolyDP(cnt, epsilon, True)  # 找到最小的矩形  rect = cv2.minAreaRect(cnt)  # box是4个点的坐标  box = cv2.boxPoints(rect)  box = np.int0(box)  # 计算高和宽  height = abs(box[0][1] - box[2][1])  width = abs(box[0][0] - box[2][0])  # 物体正常情况下的长宽比在0.5～1.5之间  ratio =float(width) / float(height)  if (ratio > 1.5 or ratio < 0.5):  continue  # 得到上、下边的坐标  box = cv2.boxPoints(rect)  box = np.int0(box)  ys = [box[0, 1], box[1, 1], box[2, 1], box[3, 1]]  xs = [box[0, 0], box[1, 0], box[2, 0], box[3, 0]]  ys\_sorted\_index = np.argsort(ys)  xs\_sorted\_index = np.argsort(xs)  if box[xs\_sorted\_index[0], 0] > 0:  x1 = box[xs\_sorted\_index[0], 0]  else:  x1 = 0  if box[xs\_sorted\_index[3], 0] > 0:  x2 = box[xs\_sorted\_index[3], 0]  else:  x2 = 0  if box[ys\_sorted\_index[0], 1] > 0:  y1 = box[ys\_sorted\_index[0], 1]  else:  y1 = 0  if box[ys\_sorted\_index[3], 1] > 0:  y2 = box[ys\_sorted\_index[3], 1]  else:  y2 = 0  img\_plate = binary[y1:y2, x1:x2]  arr.append(y1)  arr.append(y2)  arr.append(x1)  arr.append(x2)  return arr  得到4个物体在原始图像上的坐标后，将切割后的图像放入神经网络中就可以开始预测过程。 |
| **总结评价** | 本节课完成基于机械臂的工业分拣系统项目概要与项目设计，主要讲解针对嵌入式AI运算单元综合程序的Qt主线程以及神经网络识别前的图像预处理。 |