第7章 OpenCV开发与应用

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 人工智能基础与应用 | | 章名 | | | OpenCV开发与应用 | |
| 教学内容 | OpenCV开发与应用 | | | | 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 | 熟悉OpenCV计算机视觉和机器学习软件库的使用  掌握OpenCV常见应用 | | | | | | |
| 教学内容 | 1. OpenCV介绍 2. OpenCV常见应用 | | | | | | |
| 教学重点 | OpenCV实现摄像头调用和图像处理 | | | | | | |
| 教学难点 | OpenCV实现摄像头调用和图像处理 | | | | | | |
| 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】  分别从OpenCV和特点、发展历史、安装等知识点进行初步的了解。  【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** | 7.1 OpenCV介绍  OpenCV用C++语言编写，它的主要接口也是C++语言接口，但是依然保留了大量的C语言接口。同时，OpenCV也提供了Python、Java、MATLAB等语言的接口，且OpenCV是跨平台的，可以在Windows、Linux、Mac OS、Android、iOS等操作系统上运行。  OpenCV有以下特点。  （1）开源：根据BSD 3条款许可发布。  （2）优化程度高：OpenCV是一个高度优化的库，专注于实时应用程序。  （3）跨平台：具有C++、Python和Java接口，支持Linux、Mac OS、Windows、iOS和Android。  OpenCV具有如下应用领域功能：运动估算、人脸识别、姿势识别、人机交互、运动理解、对象鉴别、分割与识别、立体视觉、运动跟踪、增强现实（AR技术）等。基于上述功能实现需要，OpenCV还包括以下基于统计学的机器学习库：Boosting算法、决策树算法、Gradient Boosting算法、EM算法（期望最大化）、KNN算法、朴素贝叶斯分类、人工神经网络、随机森林、支持向量机（SVM）。  本书统一采用OpenCV的Python接口来开发。要使用OpenCV的Python接口，必须对NumPy有足够的认识和了解，例如，Python接口的OpenCV在imread()获取到图像时返回的是一个NumPy类型的数据，这和C++获取图像后返回Mat类型数据不同。  OpenCV的发展历史如表所示。  OpenCV的发展历史   | 时间 | 发布内容 | | --- | --- | | 1999年 | OpenCV项目正式启动，旨在推进CPU密集型应用 | | 2000年 | 在IEEE计算机视觉和模式识别会议上向公众正式发布Alpha版本 | | 2001—2005年 | 发布了5个beta测试版本 | | 2006年 | 1.0版本正式发布 | | 2009年 | OpenCV 2.0正式发布，添加了C++的接口，对OpenCV中很多C语言的数据和API进行了优化，旨在实现更简单、更安全的模式。官方宣布以后每6个月发布一次新版本 | | 2012年 | OpenCV由一个非营利性基金会OpenCV.org接管，负责维护开发人员和用户网站 | | 2015年 | OpenCV 3.0正式发布，带来了更多的GPU加速功能 | | 2017年 | OpenCV 3.3（包含3.3.x版本）正式发布，为DNN模块添加了加速功能以及拓展功能 | | 2017—2018年 | OpenCV 3.4（包含3.4.x版本）正式发布，进一步拓展了DNN模块，增加了对量化TensorFlow网络的支持 | | 2018—2019年 | OpenCV 4.x正式发布，对DNN模块做了优化和改进，删除了很多OpenCV 1.x的C-API |   本书使用Python接口的OpenCV 3.4.x版本，使用前需要安装。和TensorFlow等第三方模块一样，使用pip工具进行安装，在Anaconda的TensorFlow环境下输入“pip install opencv-python== 3.4.0.12”，OpenCV安装过程如图所示。    OpenCV安装过程  安装完成后进入Anaconda的TensorFlow环境下，输入“python”进入Python解释器，输入“import cv2”，若无错误信息，则代表安装完成，如图所示。    OpenCV安装完成  如果在一些嵌入式设备（如树莓派）上安装Python接口的OpenCV，可以直接下载源码安装或者使用pip命令安装。需要注意的是，嵌入式设备在使用pip安装Python接口的OpenCV后，会缺少一些运行时必要的库，需要通过apt-get命令安装，包括libatlas3-base、libjasper1、libgst7、python3-gst-1.0、libqtgui4、libqt4-test、libilmbase12、openexr、libavcodec57、libavformat57、libswscale4。这些库安装完毕后，OpenCV才可以正常使用。  7.2 OpenCV常见应用  在一些项目上需要调用本地摄像头完成图像捕获，在捕获图像后需要对原图进行一系列的预处理，使得后续识别更为简单、准确。 7.2.1 摄像头调用 在Windows系统上或者Linux系统（如Ubuntu）上，都可以创建VideoCapture对象，调用摄像头，如cap = cv2.VideoCapture(0)，圆括号里的0指的是默认摄像头。如果笔记本电脑自带摄像头，就会开启自带的摄像头；如果是台式机或者嵌入式系统，不自带摄像头，那么就可以调用第一个插入的USB摄像头，而且0可以换为1，2，…，*n*，*n*为整数。圆括号里还可以是视频的路径加视频文件名，或者是网络视频流。  通过cap对象还可以设置视频参数，cap参数如表所示。  cap参数   |  |  |  | | --- | --- | --- | | 参数 | 值 | 功能 | | CV\_CAP\_PROP\_POS\_MSEC | 0 | 视频文件的当前位置（以毫秒为单位）或视频捕获时间戳 | | CV\_CAP\_PROP\_POS\_FRAMES | 1 | 从特定帧开始读取视频 | | CV\_CAP\_PROP\_POS\_AVI\_RATIO | 2 | 视频文件的相对位置：0—视频的开始，1—视频的结束 | | CV\_CAP\_PROP\_FRAME\_WIDTH | 3 | 视频每一帧的宽 | | CV\_CAP\_PROP\_FRAME\_HEIGHT | 4 | 视频每一帧的高 | | CV\_CAP\_PROP\_FPS | 5 | 视频的帧速 | | CV\_CAP\_PROP\_FOURCC | 6 | 4个字符表示的视频编码器格式 | | CV\_CAP\_PROP\_FRAME\_COUNT | 7 | 视频的帧数 | | CV\_CAP\_PROP\_FORMAT | 8 | Byretrieve()返回的Mat对象的格式 | | CV\_CAP\_PROP\_MODE | 9 | 指示当前捕获模式的后端特定值 | | CV\_CAP\_PROP\_BRIGHTNESS | 10 | 图像的亮度（仅适用于相机） | | CV\_CAP\_PROP\_CONTRAST | 11 | 图像的对比度（仅适用于相机） | | CV\_CAP\_PROP\_SATURATION | 12 | 图像的饱和度（仅适用于相机） | | CV\_CAP\_PROP\_HUE | 13 | 图像的色相（仅适用于相机） | | CV\_CAP\_PROP\_GAIN | 14 | 图像的增益（仅适用于相机） | | CV\_CAP\_PROP\_EXPOSURE | 15 | 曝光（仅适用于相机） | | CV\_CAP\_PROP\_CONVERT\_RGB | 16 | 表示图像是否应转换为RGB的布尔标志位 | | CV\_CAP\_PROP\_WHITE\_BALANCE | 17 | 目前不支持 | | CV\_CAP\_PROP\_RECTIFICATION | 18 | 立体摄像机的整流标志位 |   【例7-1】 新建OpenCV项目目录，在OpenCV目录下新建文件，命名为camera.py，在PyCharm中编写代码，实现摄像头调用并显示。  import cv2  cap = cv2.VideoCapture(0)  while True:  ret, frame = cap.read()  cv2.imshow('frame', frame)  if cv2.waitKey(1) & 0xFF == ord('q'):  break  cap.release() 7.2.2 OpenCV的图像简单处理 图像处理涉及很多方面，常见的包括图像显示、图像的变化等。  1．加载图像，显示并保存  【例7-2】 在OpenCV目录下新建文件，命名为load\_pic.py，在PyCharm中编写代码，实现图像加载、显示及保存的功能。  import cv2  img = cv2.imread('pic.jpg',cv2.IMREAD\_COLOR)  cv2.imshow('image',img)  cv2.waitKey(0)  cv2.destroyAllWindows()  cv2.imwrite('pic\_copy.png',img, [int(cv2.IMWRITE\_JPEG\_QUALITY), 95])  cv2.imread(filepath,flags)函数的作用是读入一幅图像，filepath参数表示读入图像的完整路径，flags是读入图像的标志位，flags参数意义如表所示。  flags参数意义   |  |  | | --- | --- | | 参数 | 意义 | | cv2.IMREAD\_COLOR | 默认参数，读入一幅彩色图像，忽略alpha通道 | | cv2.IMREAD\_GRAYSCALE | 读入灰度图像 | | cv2.IMREAD\_UNCHANGED | 读入完整图像，包括alpha通道 |   cv2.waitKey()函数表示等待键盘输入，参数为0表示一直等待。此处调用cv2.waitKey()的目的是让图像一直显示，而不会一闪而逝，看过结果后只需要关掉窗口即可。  cv2.imwrite(file,img,num)函数表示保存一幅图像。第一个参数是要保存的路径和文件名。第二个参数是要保存图像的数组。第三个参数是可选的，对于不同的图像存储格式，其意义不同：对于JPEG格式，其表示的是图像的质量，用0～100之间的整数表示，默认为95；对于PNG格式，第三个参数表示的是压缩级别，默认为3。此处将图像保存在与原图一样的路径下，名称为“pic\_copy.png”。  2．将图像处理为灰度图和二值化图  【例7-3】 在OpenCV目录下新建文件，命名为gray\_binary.py，在PyCharm中编写代码，将RGB图像转换为灰度图和二值化图并显示。  import cv2  img = cv2.imread('pic.jpg', cv2.IMREAD\_COLOR)  cv2.imshow('img', img)  cv2.waitKey(0)  gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  cv2.imshow('gray', gray)  cv2.waitKey(0)  ret, binary = cv2.threshold(gray, 127, 255, cv2.THRESH\_BINARY)  cv2.imshow('binary', binary)  cv2.waitKey(0)  cv2.destroyAllWindows()  将彩色RGB图像加载进来，需要注意的是，在OpenCV读取图片后，图像并非RGB顺序，而是采用BGR的顺序。使用cv2.cvtColor()函数将原始图像转换为灰度图。在灰度图中，像素值为0～255之间的某一个值，包含0和255。二值化后，图像中的像素值为0或255，cv2.THRESH\_BINARY参数表示如果当前像素点的值大于阈值，则将输出图像的对应位置像素值置为255，否则为0，CV\_THRESH\_BINARY\_INV参数功能正好相反。除了设置阈值二值化外还有自适应二值化等，本书不做详细介绍。图像的灰度图、二值化图如图所示。    灰度图、二值化图  3．图像绘制以及写字  【例7-4】 在OpenCV目录下新建文件，命名为draw.py，在PyCharm中编写代码进行图像绘制并显示。  import cv2  img = cv2.imread('draw.jpg', cv2.IMREAD\_COLOR)  cv2.line(img, (120, 200), (190, 180), (0, 0, 0), 15)  cv2.line(img, (340, 185), (410, 150), (0, 0, 0), 15)  cv2.line(img, (230, 350), (230, 450), (0, 0, 0), 15)  cv2.line(img, (300, 350), (300, 450), (0, 0, 0), 15)  cv2.rectangle(img, (190, 150), (340, 350), (255, 0, 0), 15)  cv2.circle(img, (265, 80), 70, (0, 0, 255), 15)  font = cv2.FONT\_HERSHEY\_SIMPLEX  cv2.putText(img, 'hello!!', (400,130), font, 1, (0,0,0), 2)  cv2.imshow('image', img)  cv2.waitKey(0)  cv2.imwrite('draw\_result.png',img, [int(cv2.IMWRITE\_JPEG\_QUALITY), 95])  cv2.line()函数的作用是在图中画直线，参数分别为图像、开始坐标、结束坐标、颜色（BGR）和线条粗细。cv2.rectangle()函数的作用是画矩形，参数分别为图像、左上角坐标、右下角坐标、颜色（BGR）和线条粗细。cv2.circle()函数的作用是画圆形，参数分别为图像、圆心、半径、颜色（BGR）和线条粗细。如果要画多边形，需要用到cv2.polylines()函数。cv2.putText()函数的作用是在图像上写字，不加字库的话只接收英文和标点，参数分别为图像、要写的字、坐标、字体、字体大小、颜色、字体粗细。绘制出的结果如图所示。    绘制出的结果  4．颜色过滤  【例7-5】 在OpenCV目录下新建文件，命名为color\_filtering.py，在PyCharm中编写代码来实现颜色过滤并显示。  import cv2  import numpy as np  lower\_red = np.array([0, 0, 0])  upper\_red = np.array([180, 255, 46])  img = cv2.imread('color\_filter.jpg', cv2.IMREAD\_COLOR)  hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)  mask = cv2.inRange(hsv, lower\_red, upper\_red)  cv2.imshow('img',img)  cv2.waitKey(0)  cv2.imshow('mask',mask)  cv2.waitKey(0)  cv2.destroyAllWindows()  将原图中的黑色像素点转换为二值化后的白色，将其余像素点转换为二值化后的黑色，将RGB的颜色转换为HSV（色调、饱和度、明度）颜色。HSV是一种将RGB色彩空间中的点在倒圆锥体中表示的方法，对用户来说是一种直观的颜色模型。lower\_red和upper\_red为黑色的HSV阈值，cv2.inRange()函数可设定阈值，将图像显示出来，原图和颜色过滤后的图如图所示。  7-5  原图和颜色过滤后的图  5．形态变换  【例7-6】 在OpenCV目录下新建文件，命名为transformation.py，在PyCharm中编写代码来实现形态变换并显示。  import cv2  import numpy as np  lower\_red = np.array([0, 0, 0])  upper\_red = np.array([180, 255, 46])  img = cv2.imread('color\_filter.jpg', cv2.IMREAD\_COLOR)  hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)  mask = cv2.inRange(hsv, lower\_red, upper\_red)  kernel = np.ones((5, 5), np.uint8)  erosion = cv2.erode(mask, kernel, iterations = 1) # 腐蚀  dilation = cv2.dilate(mask, kernel, iterations = 1) # 膨胀  opening = cv2.morphologyEx(mask, cv2.MORPH\_OPEN, kernel) # 开操作  closing = cv2.morphologyEx(mask, cv2.MORPH\_CLOSE, kernel) # 闭操作  cv2.imshow('erosion',erosion)  cv2.waitKey(0)  cv2.imshow('dilation',dilation)  cv2.waitKey(0)  cv2.imshow('opening',opening)  cv2.waitKey(0)  cv2.imshow('closing',closing)  cv2.waitKey(0)  cv2.destroyAllWindows()  先对原图的黑色进行过滤，过滤后分别进行腐蚀、膨胀、开操作、闭操作。  腐蚀：让滑块滑动（此处滑块的大小为5像素×5像素），如果滑块内所有的像素都是白色的，那么得到白色，否则是黑色。腐蚀有助于消除一些白色噪声。  膨胀：与腐蚀相反，如果整个滑块不全部是黑色的，就会转换成白色。  开操作：先腐蚀后膨胀，消除二值图像中小的白色干扰区域。  闭操作：先膨胀后腐蚀，消除二值图像中小的黑色干扰区域。  进行腐蚀、膨胀后的图像如图所示。    腐蚀、膨胀后的图像  开操作、闭操作后的图像如图所示。    开操作、闭操作后的图像 7.2.3 图像处理的意义及价值 图像处理一般指数字图像处理，人类的视觉感官和机器是不一样的，机器读入的是一个数组，该数组的元素称为像素。图像处理的常用方法如表所示。  图像处理的常用方法   |  |  | | --- | --- | | 方法 | 描述 | | 图像变换 | 由于图像阵列很大，直接在空间域中进行处理，涉及的计算量很大。因此，往往采用各种图像变换的方法，如傅里叶变换、沃尔什变换、离散余弦变换等间接处理技术，将空间域的处理转换为变换域处理，这样不仅可减少计算量，而且可获得更有效的处理（如傅里叶变换可在频域中进行数字滤波处理） | | 图像编码压缩 | 图像编码压缩技术可减少描述图像的数据量（即比特数），以便缩短图像传输、处理时间，以及减少所占用的存储器容量。压缩可以在不失真的前提下进行，也可以在允许的失真条件下进行 | | 图像增强和复原 | 图像增强和复原的目的是提高图像的质量，如去除噪声、提高图像的清晰度等。图像增强不考虑图像降质的因素，突出图像中所感兴趣的部分。如强化图像高频分量，可使图像中的物体轮廓清晰，细节明显；强化低频分量可减弱图像中噪声影响。图像复原要求对图像降质的原因有一定的了解，一般应先根据降质过程建立“降质模型”，再采用某种滤波方法恢复或重建原来的图像 | | 图像分割 | 图像分割是数字图像处理中的关键技术之一，是指将图像中有意义的特征部分提取出来。其有意义的特征包括图像中的边缘、区域等，图像分割是进一步进行图像识别、分析和理解的基础 | | 图像描述 | 图像描述是图像识别和理解的必要前提。二值图像可采用其几何特性描述物体的特性，一般图像采用二维形状描述。图像描述有边界描述和区域描述两类方法。对于特殊的纹理图像，可采用二维纹理特征描述 | | 图像分类（识别） | 图像分类（识别）属于模式识别的范畴，其主要内容是图像经过某些预处理（增强、复原、压缩）后，进行图像分割和特征提取，从而进行判决分类。图像分类常采用经典的模式识别方法，有统计模式分类和句法（结构）模式分类。近年来发展起来的模糊模式识别和人工神经网络模式分类在图像识别中也越来越受重视 |   OpenCV使得图像处理更加容易、方便，将处理之后的图像放在机器学习或深度学习中再做识别，可以使图像处理更准确。 |
| **总结评价** | 本节课主要学习OpenCV的开发和应用，针对摄像头的调用、图像的处理做了详细介绍。 |