第 4 课 图像处理方法介绍

1.图像预处理

在很多时候我们需要对原始图像进行特殊处理，比如做人脸识别时，往往需要将图像转换为灰度图像。在正式学习图像处理前，我们先来了解下 OpenCV 中不同色彩空间的转换。

OpenCV 中有数百种关于不同颜色空间之间转换的方法。目前在计算机视觉领域常见的颜色空间有三种：灰度、BGR、HSV（色调 Hue，饱和度 Saturation，亮度 Value）。在我们的玩法课程中有用到 YCrCba 颜色空间，其中 Y 指的是明亮度，Cr 指的是 RGB 输入信号的红色部分与 RGB 信号亮度值之间的差异，Cb 指的是 RGB 输入信号蓝色部分与 RGB 信号亮度值之间的差异。这种色彩空间的特点是将亮度和色度分离开，从而适合于图像处理领域，经常用在人体肤色检测中。

颜色转换：cv2.cvtColor(Img,Transform_model)。

Img——图片对象。

Transform_model——指定色彩转换模式。

另外需要注意的是这里的色光混合与颜料混合产生颜色不一样。颜料有颜色是因为它反射那种颜色的光，多种颜色混合时遵循的是减色混合，而屏幕上的光是屏幕发出的光，遵循的是增色混合原理。

2.图像处理原理

在 OpenCV 中，对图像和视频的大多数处理都会涉及到傅里叶变换的概念。傅里叶观察到数学中所有的波形都可以由一系列简单而且频率不同的正弦曲线叠加得到。由此推断并证明了人们所看到的任何波形都是由其他波形叠加得到的。

在我们处理图像时，可以通过去掉一部分波形来得到感兴趣的区域。

除了傅里叶变换外，在图像处理过程常常会提到高通滤波器和低通滤波器。

其中高通滤波器是检测图像的某个区域，然后根据像素与周围像素的亮度差值来提升该像素亮度的滤波器。

高通滤波器会增大像素点与周围像素点亮度差距，一般用在边缘检测上。

低通滤波是在像素与周围像素的亮度差值小于一个特定值时，平滑该像素的亮度的滤波器。它主要用于去噪和模糊化，比如最常用的的模糊滤波器（平滑滤波器）：高斯模糊，它的实质上就是一个削弱高频信号强度的低通滤波器。

3.边缘检测

每张图片内都有很多内容，在处理图像时往往需要把不同的内容分离开。

两个具有不同灰度值的相邻区域之间总存在边缘，而边缘是灰度值不连续的表现。边缘检测目的通俗来说就是检测图像的边缘，即图像差异较大的地方，为后续进一步图像处理做准备。

例如我们要从一张同时有苹果、橘子、梨的图片里找到橘子，那么我们需要先判断图片中哪些是物体，哪些是背景。在这里我们感兴趣的是水果，边缘检测可以检测图片里水果的轮廓边缘，帮助计算机判断图片中哪些地方是物体，哪些地方是背景，为后续进一步判断哪些是橘子，哪些不是橘子做准备。

OpenCV 提供了许多边缘检测滤波函数，包括 Laplacian()、Sobel()以及 Scharr()。这些滤波函数都会将非边缘区域转换为黑色，将边缘区域转换为白色或其他饱和的颜色。

但是在实际处理中，这些函数都很容易将噪声错误地识别为边缘，所以需要在找边缘之前对图像进行模糊处理，去除噪声。OpenCV 同样提供了诸多模糊滤波函数，常用的有 blur()（简单的算术平均）、medianBlur()（中值滤波）、GaussianBlur()。

在实际使用中还有一个非常方便的 Canny 函数，它只需要用一行代码就能轻松实现边缘检测。

cv2.Canny(Pic,Minval,Maxval)。

Pic——要检测边缘的图像对象。

Minval——最小灰度梯度阈值，当图像灰度梯度低于 Minval 时，会被舍去。

Maxval——最大梯度值阈值，当图像灰度梯度高于 Maxval 时，会被认定为边缘，当图像灰度梯度在 Minval 和 Maxval 之间时，如果与边缘相连是边界的一部分，否则被舍弃。

Canny 边缘检测算法内部实际上使用如下 5 个步骤对图像进行了处理：

1）使用高斯滤波器，平滑图像，滤除噪声。

2）计算图像中每个像素点的梯度强度和方向。

3）在边缘上使用非最大值抑制（Non-Maximum Suppression）,以消除边缘检测带来的杂散响应。

4）在检测到的边缘上使用双阈值去除假阳性（false positive），也就是判断梯度值在 Minval 和 Maxval 之间的部分哪些是边缘哪些要舍弃。

5）分析所有边缘及其之间的连接，抑制孤立的弱边缘，以保证留下真正的边缘并消除不明显的边缘。

示例：新建一个 py 文件，并在该 py 文件同一个文件夹放一张全名为“camera.png”的图片，输入如下代码，运行后可看到画面中出现该图像白色轮廓，其他地方变为黑色，按下任意键后图像隐藏。

```
import cv2
import numpy as np
img=cv2.imread('camera.png',0)
cv2.imwrite('camera_2.png',cv2.Canny(img,200,300))
cv2.imshow('camera_2',cv2.imread('camera_2.png'))
cv2.waitKey()
cv2.destroyALLWindows()
```

4.轮廓检测

在计算机视觉中，常常需要检测图像或视频中物体的轮廓，并在此基础上计算多边形边

界、形状逼近、计算感兴趣区域等等。

在 OpenCV 中可以调用 findContours()函数来检测轮廓，函数具体参数如下：

findContours(Img,Model,Method)。

Img——要检测轮廓的图片对象，注意该图片需要是灰度图片。

Model——表示轮廓的检测模式：

cv2.RETR_EXTERNAL：表示只检测外轮廓。

cv2.RETR_LIST：检测的轮廓不建立等级关系。

cv2.RETR_CCOMP：建立两个等级的轮廓，上面的一层为外边界，里面的一层为内孔的边界信息。如果内孔内还有一个连通物体，这个物体的边界也在顶层。

cv2.RETR_TREE：建立一个等级树结构的轮廓。

Method——表示轮廓的逼近方法，检测策略，具体为如下三种：

cv2.CHAIN_APPROX_NONE 存储所有的轮廓点，相邻的两个点的像素位置差不超过 1，即 max（abs（x1-x2），abs（y2-y1））==1。

cv2.CHAIN_APPROX_SIMPLE 压缩水平方向，垂直方向，对角线方向的元素，只保留该方向的终点坐标，例如一个矩形轮廓只需 4 个点来保存轮廓信息。

cv2.CHAIN_APPROX_TC89_L1， CV_CHAIN_APPROX_TC89_KCOS 使用 teh-Chinl chain 近似算法。

findContours()函数有三个返回值：修改后的图像、图像的轮廓以及它们的层次。

5.直线和圆检测

霍夫（Hough）变换是直线和形状检测背后的理论基础，它由 Richard Duda 和 Peter Hart 在 20 世纪 60 年代提出。

直线检测可以通过 HougLines，HoughLinesP 两种函数中的任意一个完成。第一个函数

使用标准的霍夫变换，第二个函数使用概率霍夫变换。

概率霍夫变换是标准霍夫变换的优化版，它只通过分析点的子集并估计这些点都属于一条直线的概率来进行直线检测，这种算法计算量较少，执行速度快。

概率霍夫变换：

cv2.HougLinesP(img,rho,theta,threshod,minLineLength,maxLineGap)

img——需要处理的图像，必须是灰度图（二值图），一般使用 canny 边缘检测的结果图像。

rho——线段以像素为单位的距离精度，double 类型的，推荐用 1.0。

theta——线段以弧度为单位的角度精度，推荐用 numpy.pi/180。

threshod——阈值，低于该阈值的直线都会被删除。

minLineLength——线段以像素为单位的最小长度，根据应用场景设置。

maxLineGap——同一方向上两条线段判定为一条线段的最大允许间隔（断裂），小于设定值，则把两条线段当成一条线段，值越大，允许线段上的断裂越大，越有可能检出潜在的直线段。

OpenCV 中可以使用 HoughCircles 函数来检测圆，它与使用 HougLines 函数类似。

cv2.HoughCircles(image, method, dp, minDist, circles, param1, param2, minRadius, maxRadius)

image——输入图像，需要灰度图（二值图）。

method——检测方法，常用 CV_HOUGH_GRADIENT。

dp——为检测内侧圆心的累加器图像的分辨率于输入图像之比的倒数，如 dp=1，累加器和输入图像具有相同的分辨率，如果 dp=2，累计器便有输入图像一半那么大的宽度和高度。

minDist——表示两个圆之间圆心的最小距离。

param1——默认值 100，它是 method 设置的检测方法的对应的参数，对当前唯一的方法霍夫梯度法 cv2.HOUGH_GRADIENT，它表示传递给 canny 边缘检测算子的高阈值，而低阈值为高阈值的一半

param2——默认值 100，它是 method 设置的检测方法的对应的参数，对当前唯一的方法霍夫梯度法 cv2.HOUGH_GRADIENT，它表示在检测阶段圆心的累加器阈值，它越小，就越可以检测到更多根本不存在的圆，而它越大的话，能通过检测的圆就更加接近完美的圆形了。

minRadius——默认值 0，圆半径的最小值。

maxRadius——默认值 0，圆半径的最大值。

示例：新建一个 py 文件，并在该 py 文件同一个文件夹放一张全名为“camera.png”的图片，输入如下代码，运行后可看到画面中用绿色的线把检测到的圆形轮廓圈出来，并且把圆心标出了，按下任意键后关闭图像窗口。

```
import cv2
import numpy as np
planets=cv2.imread('camera.png',1)
gray_img=cv2.cvtColor(planets,cv2.COLOR_BGR2GRAY)
img=cv2.medianBlur(gray_img,5)
cimg=cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)
#find circles
circles=cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,120,param1=100,
                         param2=80,minRadius=0,maxRadius=0)
#save circle data in np'array
circles=np.uint16(np.around(circles))
for i in circles[0,:]:
    #draw the outer circle
    cv2.circle(planets,(i[0],i[1]),i[2],(0,255,0),2)
    #draw the center of the circle
    cv2.circle(planets,(i[0],i[1]),2,(0,0,255),3)

cv2.imwrite('planets_circles.jpg',planets)
cv2.imshow('houghCircles',planets)
cv2.waitKey()
cv2.destroyALLWindows()
```

如果要检测多边形可以结合 cv2.findContours 函数和 cv2.approxPloyDP 函数。