基于计算机视觉的钢筋数量检测

张家瑞　王建锋
西京学院机械工程学院　陕西西安　710123

摘　要：本文介绍了如何使用 Python 和 OpenCV 库检测一张图片中钢筋的数量。讨论了小市场中一些个体户在清点螺纹钢数量时，浪费大量时间的问题，并提出了解决方案。具体步骤包括：导入必要的库、读取图像、将彩色图像转换为灰度图像、进行边缘检测以查找钢筋轮廓、绘制钢筋轮廓，并统计钢筋数量。

关键词：计算机视觉；OpenCV；数量检测

一、问题介绍

在日常工作中，螺纹钢的数量确定是一件枯燥且麻烦的任务，准确性还不能得到保证，一般情况下都是要数两遍甚至更多，这就导致了大量的时间浪费。

Python是一种开源系统，用户可自由开发和共享源代码。它易于学习和使用，广泛应用于数据处理、自动化技术、人工智能和机器视觉等 [1]。Python 软件开发包中的算法可用于图像预处理，例如灰度算法、高斯滤波算法和 Canny 边缘检测算法。合理利用 Python 这些优点来帮助我们完成这样的任务。要解决上述提到的问题，对拍摄的图像依次进行图像处理、边缘检测、特征提取和计数。

二、图像获取和处理

首先，我们需要导入一些必要的库。 在这里，我们使用的是 Python 中的 cv2 模块，它是 OpenCV 库的接口。接下来我们需要读取一张图片，这张图片我们在获取的时候要尽量保证它的周围环境单一且干扰比较少。否则实验可能会变得更加困难，并且可能遇到很多处理起来很麻烦的意外，这样做有利于实验平稳，高效的进行下去。

在 OpenCV 中有许多种进行颜色空间转换的方法。但是一般情况下我们用到的也就两种。分别是 BGR 转化为 Gray 和 BGR 转化为 HSV。在图像处理中，将 BGR 图像转换为灰度图像是一个常见的操作。这种转换将彩色图像转换为黑白图像，使图像只有单一的亮度级别。在 BGR 到 Gray 的转换中，使用的是加权平均法。具体的计算公式为：

Gray = 0.1140 * B + 0.5870 * G + 0.2989 * R

其中，B、G、R 分别代表红、绿、蓝三个通道的像素值，Gray 表示转换后的灰度值。通过这种加权平均的方法，可以有效地将彩色图像转换为灰度图像，方便后续的处理和操作。

HSV 色彩空间是一种与人类视觉感知更为接近的色彩空间，它由色调（H）、饱和度（S）和亮度（V）三个分量组成。在 HSV 色彩空间中，色调表示颜色的种类，饱和度表示颜色的纯度或强度，亮度表示颜色的明暗程度。

在这里我们在处理图像时，将其转换为灰度图像。这是因为灰度图像只有一个颜色通道，相对于彩色图像而言具有更高的处理速度 [2]。灰度处理代码如下：

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.cvtColor() 函数表示转换颜色空间，cv2.COLOR_BGR2GRAY 表示将 BGR 转化为 Gray，这是将 img 图片灰度处理后存到 gray 中。

在图像上进行边缘检测可以查找轮廓，这对于钢筋数量检测非常重要。 OpenCV 库中有许多种边缘检测算法，其中最流行的是 Canny 边缘检测算法。

Canny 边缘检测算法的原理有三步：

噪声滤波：使用高斯滤波器来平滑图像，以减少噪声干扰。

计算梯度：计算图像梯度，得到可能边缘；

非极大值抑制和双阈值检测：针对每个象素点，检测它是否为可能的边界点。假设某一个点的梯度值大于其相邻像素，则此点就可以为边界点。接着，通过二个预设的强阈值（低阈值和高阈值）进一步确定是否为边界点。当一个点的强梯度值大于了高阈值时，可将它标记为强边界点。若一个点的梯度值在低阈值与高阈值之间，可将它标记为弱边缘点。如果一个点的梯度值小于最低阈值，则将它标识为非边

缘点。Canny 代码如下：

edged = cv2.Canny(gray, 30, 200)

在上面的代码行中，我们使用 cv2.Canny() 函数对灰度图像进行边缘检测，并将结果保存在名为 edged 的变量中。参数 30 和 200 是边缘梯度的阈值，低于 30 的梯度被认为是不相关的，而高于 200 的梯度被认为是强的边缘。

1. 查找并计数

查找并计数有许多种方法，例如模板匹配法，它的基本思想是将一幅已知的需要匹配的小图像，在一幅大图像中搜寻目标，已知该图中有要找的目标，且该目标同模板有相同的尺寸、方向和图像元素。通过统计计算图像的均值、梯度、距离、方差等特征可以在图中找到目标，确定其坐标位置。但是，这种方法对于图像或模板的旋转和缩放很敏感，因此，其匹配能力有限。

轮廓检测方法，他主要是为了从图像中提取出物体的边缘信息，即轮廓。它通过分析图像的灰度变化和梯度等信息，来识别图像中的物体边缘，并形成连续的轮廓线。轮廓检测可以用于形状分析、图像分割、物体识别等应用领域。

还有一种新兴的图像识别方法，即基于深度学习的方法，它通过训练深度神经网络来实现图像识别。深度学习技术能够自动地从大量训练数据中学习到有效的特征，这些特征能够更好地捕捉图像中的关键信息，从而提高了图像识别的准确性。

考虑到本例中的实际情况，这里我们使用 cv2.findContours() 函数查找轮廓，返回一个由轮廓组成的列表和层次结构。在本例中，我们使用 edged 变量中包含的边缘检测图像来查找钢筋轮廓，并将结果保存在名为 contours 和 hierarchy 的两个变量中，contours 变量包含了所有检测到的轮廓信息，每个轮廓由一组点坐标表示。而 hierarchy 变量则包含了各个轮廓之间的层级关系信息，如某一轮廓是否为另一个轮廓的子轮廓等。

第二个参数是轮廓的检索模式，指定为 cv2.RETR_EXTERNAL 以仅检测最外层轮廓。

第三个参数表示逼近方法，cv2.CHAIN_APPROX_SIMPLE 表示压缩水平方向，垂直方向，对角线方向的元素，只保留该方向的终点坐标 [5]；代码如下：

contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

然而，就这样还不足以满足检测螺纹钢数量的功能，螺纹钢有一个不同于普通钢筋的特性，那就是它的表面并不是光滑的，而是由一段又一段的螺纹线所布满，如果我们就在这里结束，那么必然导致最后检测出来的钢筋数远远大于正常的数量，因为螺纹的轮廓也被计算到了钢筋数里面，这样测量与技术是极为不准确的，产生了极大的误差，必须把图像进行进一步的处理。

2. 解决计数问题

使用 cv2.arcLength() 函数计算每个轮廓的周长，并将其与阈值进行比较。如果轮廓的周长大于阈值（例如 300），则将其保留，并使用 cv2.drawContours() 函数绘制钢筋轮廓。

if cv2.arcLength(contour, True) > 300:

cv2.drawContours(img, [contour], −1, (0, 255, 0), 2)

这行代码会将名为 img 的图像上以绿色 (0, 255, 0) 绘制一个名为 contour 的轮廓，线宽为 2 个像素。其中，[contour] 是一个包含了单个要绘制的轮廓的列表，−1 表示绘制所有的轮廓点，而不仅仅是轮廓边界。

然后使用一个名为 num_steel_reinforcements 的变量来记录符合长度条件的轮廓数量。对于每个轮廓，我们使用 cv2.arcLength() 函数计算其周长，并将其与阈值进行比较。如果轮廓的周长大于阈值（例如 300），则 num_steel_reinforcements 变量的值加 1。最后，我们使用 num_steel_reinforcements 变量的值打印符合长度条件的钢筋数量 [6]。如下：

if cv2.arcLength(contour, True) > 300:

num_steel_detection += 1

最后得到处理后的图片，如图 1。

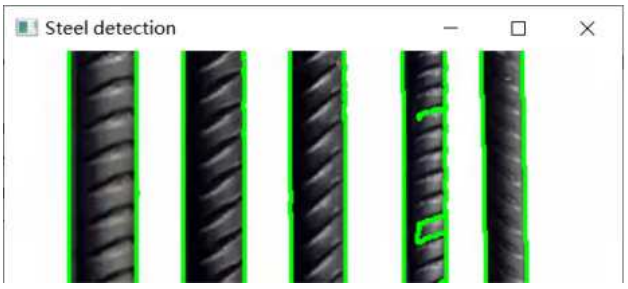

图 1、处理后的轮廓检测图

然而每根钢筋能检测到两个符合条件的轮廓，所以我们要在最后的结果里将得到的轮廓数除以 2，如下：

print(“Number of steel detection:”, num_steel_detection /2)

得到这个例子的结果为 5。

三、总结

这篇文章主要介绍了使用 OpenCV 进行基本计算机视觉应用的操作，其中最主要的是图像轮廓检测，通过这个例子，我们学到一些知识，例如如何读取和显示图像，对相应的环境选择采用合适的方法对图像进行二值化处理。还有函数的使用，例如使用 cv2.findContours() 函数查找图像中的轮廓；使用 cv2.drawContours() 函数绘制轮廓并在图像上显示结果。除此之外，我们还可以学到一些常见的图像处理技巧，例如阈值化、轮廓检测等。这些技巧是数字图像处理领域中的基础知识，掌握它们对于深入理解更高级别的图像处理算法和应用至关重要。但是在实际应用中，我们还需要考虑的更多，不同的图像往往需要不同的参数或修改不同的阈值才能得到较好的结果。因此，我们需要对不同的图像进行测试和优化以获得最佳的效果。

参考文献

[1] 向艳芳，潘跃亮，刘苗 . 基于 Python 工业机器人视觉检测与抓取研究 [J]. 现代农机 ,2022(05):46-48.

[2] 吴黎，解文欢，张有智等 . 黑龙江省水稻种植面积遥感提取研究 [J]. 现代农机 ,2022(05):44-46.

[3] 刘晓波，宋成明，曹凯源 . 基于图像识别技术的垃圾智能分类 [J]. 自动化应用 ,2022(02):77-79+84.DOI:10.19769/j.zdhy.2022.02.021.

[4] 郭董菊 . 基于图像处理的苹果表面及叶面病害检测技术研究 [D]. 浙江科技学院 ,2022.DOI:10.27840/d.cnki.gzjkj.2022.000233.

[5] 雷寰宇 . 基于图像的表格识别问题研究 [J]. 科技视界 ,2021(13):32-34.DOI:10.19694/j.cnki.issn2095-2457.2021.13.13.

[6] 白金华 . 自然环境下基于计算机视觉的果园苹果检测 [D]. 贵州民族大学 ,2022.DOI:10.27807/d.cnki.cgzmz.2022.000306.