基於OpenCV的銲件缺陷檢測

原創 努比 小白學視覺 2022-02-18 10:05

收錄於話題
#OpenCV視覺實戰項目

66個

點擊上方“**小白學視覺**”，選擇加"**星標**"或“**置頂**”
重磅乾貨，第一時間送達

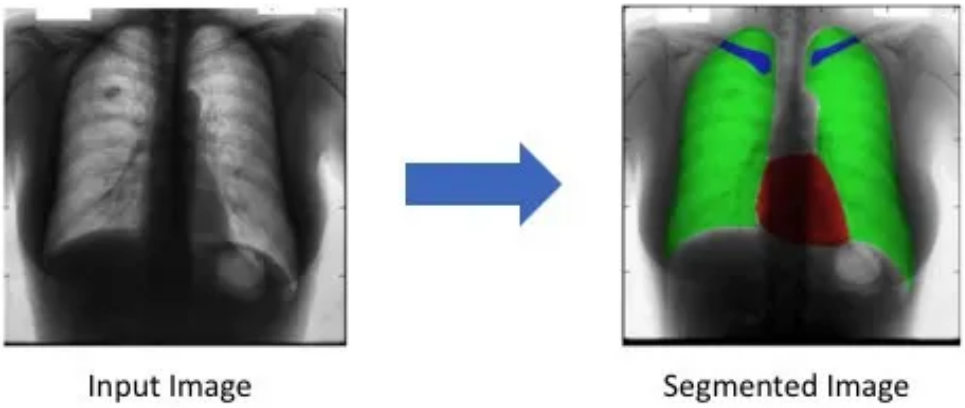
01. 簡介

焊接缺陷是指焊接零件表面出現不規則、不連續的現象。焊接接頭的缺陷可能會導致組件報廢、維修成本高昂，在工作條件下的組件的性能顯著下降，在極端情況下還會導致災難性故障，並造成財產和生命損失。此外，由於焊接技術固有的弱點和金屬特性，在焊接中總是存在某些缺陷。不可能獲得完美的焊接，因此評估焊接質量非常重要。

可以通過圖像來檢測焊接中的缺陷，並精確測量每個缺陷的嚴重性，這將有助於並避免上述危險情況的出現。使用卷積神經網絡算法和U-Net架構可提高檢測的效率，精度也能達到98.3%。

02. 圖像分割

圖像分割是指將圖像劃分為包含相似屬性的不同像素區域。為了對圖像分析和解釋，劃分的區域應與對象特徵密切相關。圖像分析的成功取決於分割的可靠性，但是圖像的正確分割通常是一個非常具有挑戰性的問題。



對心臟（紅色），肺部（綠色）和鎖骨（藍色）的胸部X光進行了分割

03. 圖像中心距

图像中心距是图像像素强度的某个特定加权平均值。图像矩可用于描述分割后的对象。通过图像瞬间发现的图像简单属性包括：

1. 面积（或总强度）
2. 质心
3. 有关其方向的信息

04. 数据

该数据集包含两个目录。原始图像存储在“图像”目录中，分割后的图像存储在“标签”目录中。让我们来看看这些数据：原始图像是RGB图像，用于训练模型和测试模型。这些图片的尺寸各不相同。直观地，较暗的部分是焊接缺陷。模型需要对这些图像执行图像分割。

来自“图像”的原始图像

“标签”目录的图像是二进制图像或地面真相标签。这是我们的模型必须针对给定的原始图像进行预测。在二进制图像中，像素具有“高”值或“低”值。白色区域或“高”值表示缺陷区域，而黑色区域或“低”值表示无缺陷。

来自“标签”的二进制图像

05. 算法

我们将使用U-Net来解决这个问题，通过以下三个主要步骤来检测缺陷及其严重性：

- 图像分割
- 使用颜色显示严重性
- 使用图像矩测量严重性

训练模型

使用的U-Net架构

注意事项：

- 每个蓝色框对应一个多通道特征图
- 通道数显示在框的顶部。
- （x，y）尺寸位于框的左下边缘。
- 箭头表示不同的操作。
- 图层名称位于图层下方。
- C1，C2，...。C7是卷积运算后的输出层
- P1，P2，P3是最大池化操作的输出层
- U1，U2，U3是上采样操作的输出层
- A1，A2，A3是跳过连接。
- 左侧是收缩路径，其中应用了常规卷积和最大池化操作
- 图像尺寸逐渐减小，而深度逐渐增大。
- 右侧是扩展路径，在其中应用了（向上采样）转置卷积和常规卷积运算
- 在扩展路径中，图像尺寸逐渐增大，深度逐渐减小
- 为了获得更好的精确位置，在扩展的每个步骤中，我们都使用跳过连接，方法是将转置卷积层的输出与来自编码器的特征图在同一级别上连接：
  A1 = U1 + C3
  A2 = U2 + C2
  A3 = U3 + C1
  每次串联后，我们再次应用规则卷积，以便模型可以学习组装更精确的输出。

```
import numpy as np
import cv2
import os
import random
import tensorflow as tf

h,w = 512,512

def create_model():

    inputs = tf.keras.layers.Input(shape=(h,w,3))

    conv1 = tf.keras.layers.Conv2D(16,(3,3),activation='relu',padding='sa
    pool1 = tf.keras.layers.MaxPool2D()(conv1)

    conv2 = tf.keras.layers.Conv2D(32,(3,3),activation='relu',padding='sa
    pool2 = tf.keras.layers.MaxPool2D()(conv2)

    conv3 = tf.keras.layers.Conv2D(64,(3,3),activation='relu',padding='sa
    pool3 = tf.keras.layers.MaxPool2D()(conv3)

```

```
    conv4 = tf.keras.layers.Conv2D(64,(3,3),activation='relu',padding='sa

    upsm5 = tf.keras.layers.UpSampling2D()(conv4)
    upad5 = tf.keras.layers.Add()([conv3,upsm5])
    conv5 = tf.keras.layers.Conv2D(32,(3,3),activation='relu',padding='sa

    upsm6 = tf.keras.layers.UpSampling2D()(conv5)
    upad6 = tf.keras.layers.Add()([conv2,upsm6])
    conv6 = tf.keras.layers.Conv2D(16,(3,3),activation='relu',padding='sa

    upsm7 = tf.keras.layers.UpSampling2D()(conv6)
    upad7 = tf.keras.layers.Add()([conv1,upsm7])
    conv7 = tf.keras.layers.Conv2D(1,(3,3),activation='relu',padding='sam

    model = tf.keras.models.Model(inputs=inputs, outputs=conv7)

    return model

images = []
labels = []

files = os.listdir('./dataset/images/')
random.shuffle(files)

for f in files:
    img = cv2.imread('./dataset/images/' + f)
    parts = f.split('_')
    label_name = './dataset/labels/' + 'W0002_' + parts[1]
    label = cv2.imread(label_name,2)

    img = cv2.resize(img,(w,h))
    label = cv2.resize(label,(w,h))

    images.append(img)
    labels.append(label)

images = np.array(images)
labels = np.array(labels)
labels = np.reshape(labels,
```

```
    (labels.shape[0],labels.shape[1],labels.shape[2],1))

print(images.shape)
print(labels.shape)

images = images/255
labels = labels/255

model = tf.keras.models.load_model('my_model')

#model = create_model()  # uncomment this to create a new model
print(model.summary())

model.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accu
model.fit(images,labels,epochs=100,batch_size=10)
model.evaluate(images,labels)

model.save('my_model')
```

该模型使用Adam优化器编译，由于只有两类（缺陷或没有缺陷），因此我们使用二进制交叉熵损失函数。我们使用10批次、100个epochs（在所有输入上运行模型的次数）。调整这些参数，模型性能可能会有很大的改善可能。

测试模型

由于模型采用的尺寸为512x512x3，因此我们将输入的尺寸调整为该尺寸。接下来，我们通过将图像除以255进行归一化以加快计算速度。图像进入模型后以预测二进制输出，为了放大像素的强度，二进制输出已乘以1000。

然后将图像转换为16位整数以便于图像操作。之后，算法将检测缺陷并通过颜色分级在视觉上标记缺陷的严重性，并根据缺陷的严重性为具有缺陷的像素分配权重。然后考虑加权像素，在此图像上计算图像力矩。最终将图像转换回8位整数，并以颜色分级及其严重性值显示输出图像。

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
import os
import random
import tensorflow as tf

```

```
h,w = 512,512
num_cases = 10

images = []
labels = []

files = os.listdir('./dataset/images/')
random.shuffle(files)

model = tf.keras.models.load_model('my_model')

lowSevere = 1
midSevere = 2
highSevere = 4

for f in files[0:num_cases]:
    test_img = cv2.imread('./dataset/images/' + f)
    resized_img = cv2.resize(test_img,(w,h))
    resized_img = resized_img/255
    cropped_img = np.reshape(resized_img,
          (1,resized_img.shape[0],resized_img.shape[1],resized_img.shape[

    test_out = model.predict(cropped_img)

    test_out = test_out[0,:,:,0]*1000
    test_out = np.clip(test_out,0,255)

    resized_test_out = cv2.resize(test_out,(test_img.shape[1],test_img.sh
    resized_test_out = resized_test_out.astype(np.uint16)

    test_img = test_img.astype(np.uint16)

    grey = cv2.cvtColor(test_img, cv2.COLOR_BGR2GRAY)

    for i in range(test_img.shape[0]):
     for j in range(test_img.shape[1]):
          if(grey[i,j]>150 & resized_test_out[i,j]>40):
            test_img[i,j,1]=test_img[i,j,1] + resized_test_out[i,j]
            resized_test_out[i,j] = lowSevere
```

```
            elif(grey[i,j]<100 & resized_test_out[i,j]>40):
              test_img[i,j,2]=test_img[i,j,2] + resized_test_out[i,j]
              resized_test_out[i,j] = highSevere
            elif(resized_test_out[i,j]>40):
              test_img[i,j,0]=test_img[i,j,0] + resized_test_out[i,j]
              resized_test_out[i,j] = midSevere
            else:
              resized_test_out[i,j] = 0

    M = cv2.moments(resized_test_out)
    maxMomentArea = resized_test_out.shape[1]*resized_test_out.shape[0]*h
    print("0th Moment = " , (M["m00"]*100/maxMomentArea), "%")

    test_img = np.clip(test_img,0,255)
    test_img = test_img.astype(np.uint8)

    cv2_imshow(test_img)
    cv2.waitKey(0)
```

07. 结果

我们使用颜色来表示缺陷的严重程度：

1. 绿色表示存在严重缺陷的区域。
2. 蓝色表示缺陷更严重的区域。
3. 红色区域显示出最严重的缺陷。

零阶矩将以百分比形式显示在输出图像旁边，作为严重程度的经验指标。

以下是三个随机样本，它们显示了原始输入，地面真实情况以及由我们的模型生成的输出。

范例1：

原始图像

二进制图像（地面真相）

具有严重性的预测输出

范例2：

原始图像

二进制图像（地面真相）

具有严重性的预测输出

范例3：

原始图像

二进制图像（地面真相）

具有严重性的预测输出

参考文献：

https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm#adaptive
https://medium.com/r/?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FImage_moment
https://medium.com/r/?url=https%3A%2F%2Ftowardsdatascience.com%2Funderstanding-semantic-segmentation-with-unet-6be4f42d4b47
https://www.sciencedirect.com/topics/materials-science/welding-defect

代码链接：https://github.com/malakar-soham/cnn-in-welding

下载1：OpenCV-Contrib扩展模块中文版教程

在「**小白学视觉**」公众号后台回复：**扩展模块中文教程，**即可下载全网第一份OpenCV扩展模块教程中文版，涵盖**扩展模块安装、SFM算法、立体视觉、目标跟踪、生物视觉、超分辨率处理**等二十多章内容。

下载2：Python视觉实战项目52讲

在「**小白学视觉**」公众号后台回复：**Python视觉实战项目，**即可下载包括**图像分割、口罩检测、车道线检测、车辆计数、添加眼线、车牌识别、字符识别、情绪检测、文本内容提取、面部识别**等31个视觉实战项目，助力快速学校计算机视觉。

下载3：OpenCV实战项目20讲

在「**小白学视觉**」公众号后台回复：**OpenCV实战项目20讲，**即可下载含有**20**个基于**OpenCV**实现20个**实战项目**，实现OpenCV学习进阶。

交流群

歡迎加入公眾號讀者群一起和同行交流，目前有**SLAM 、三維視覺、傳感器、自動駕駛、計算攝影**、檢測、分割、識別、**醫學影像、GAN 、算法競賽**等微信群（以後會逐漸細分），請掃描下面微信號加群，備註：”暱稱+學校/公司+研究方向“，例如：”張三 + 上海交大 + 視覺SLAM“。**請按照格式備註，否則不予通過。**添加成功後會根據研究方向邀請進入相關微信群。**請勿**在群內發送**廣告**，否則會請出群，謝謝理解~

收錄於話題#OpenCV视觉实战项目 66

下一篇 · 使用OpenCV校准鱼眼镜头

喜欢此内容的人还喜欢

基於Opencv實現眼睛控制鼠標

小白學視覺