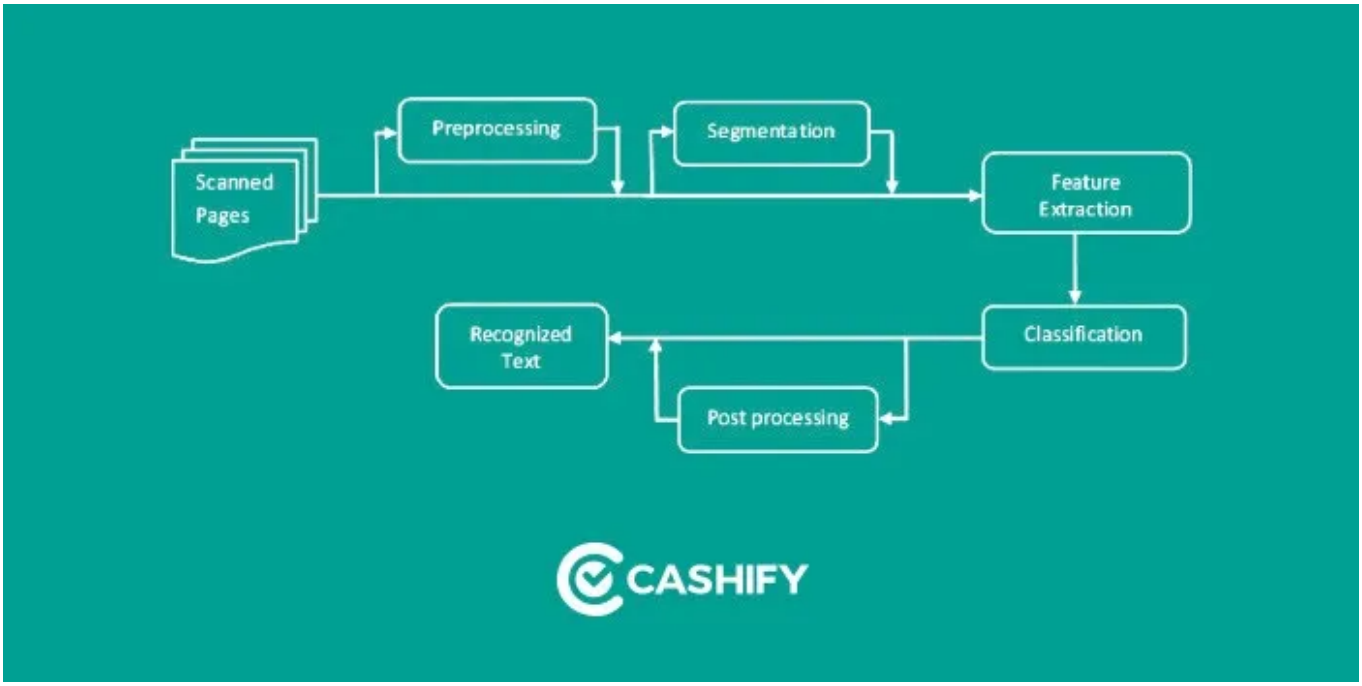
如何利用圖像預處理提高OCR的準確性?

小白 3D視覺初學者 2021-12-27 10:05



OCR代表光學字符識別，將文檔照片或場景照片轉換為機器編碼的文本。有很多工具可以在你們的系統中實現OCR，例如Tesseract OCR和Cloud Vision。他們使用AI和機器學習以及經過訓練的自定義模型。文本識別取決於多種因素，以產生高質量的輸出。OCR輸出在很大程度上取決於輸入圖像的質量，這就是每個OCR引擎都提供有關輸入圖像質量及其大小的準則的原因，這些準則可幫助OCR引擎產生準確的結果。

圖像預處理功能可以提高輸入圖像的質量，以便OCR引擎為我們提供準確的輸出，使用以下圖像處理操作可以改善輸入圖像的質量。

圖像縮放

圖像縮放比例對於圖像分析很重要。通常，OCR引擎會準確輸出300 DPI的圖像。DPI描述了圖像的分辨率，換句話說，它表示每英寸的打印點數。

```
def set_image_dpi(file_path):
    im = Image.open(file_path)
    length_x, width_y = im.size
```

```
    factor = min(1, float(1024.0 / length_x))
    size = int(factor * length_x), int(factor * width_y)
    im_resized = im.resize(size, Image.ANTIALIAS)
    temp_file = tempfile.NamedTemporaryFile(delete=False,   suffix='.png')
    temp_filename = temp_file.name
    im_resized.save(temp_filename, dpi=(300, 300))
    return temp_filenam
```

偏斜矯正

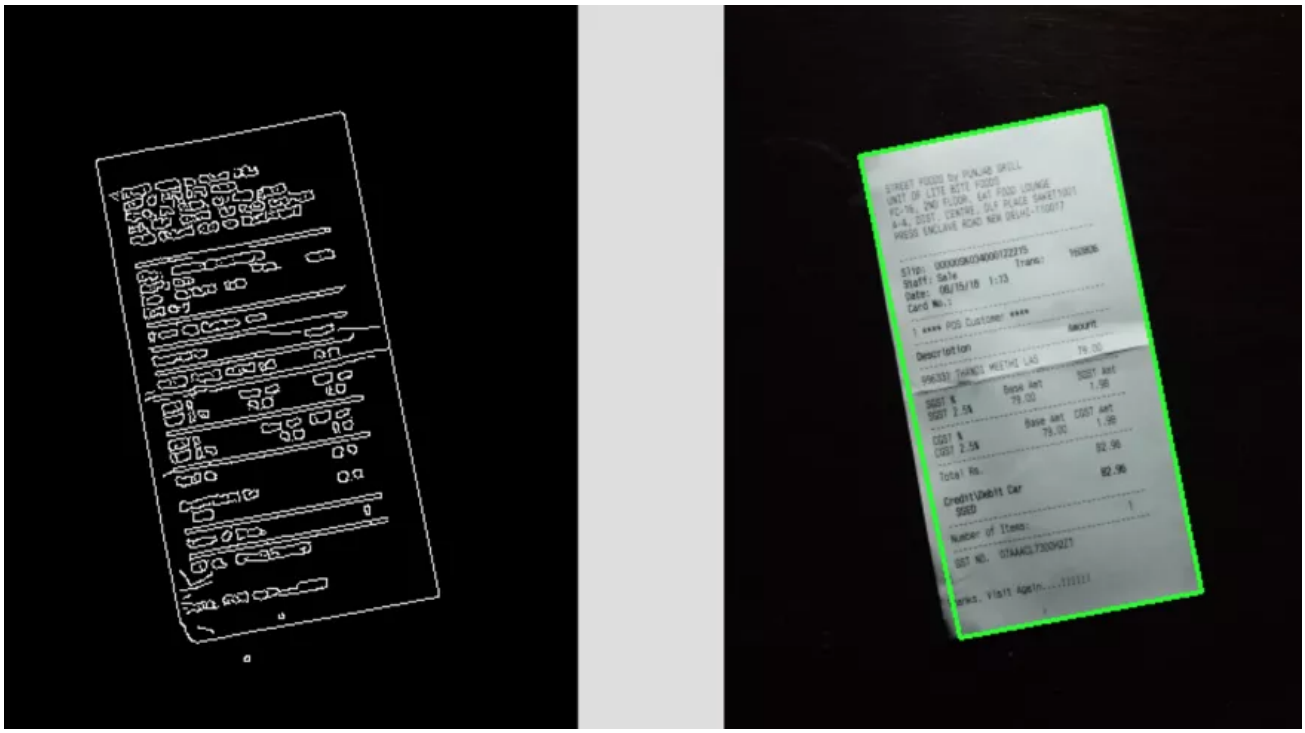
歪斜圖像定義為不直的文檔圖像。歪斜的圖像會直接影響OCR引擎的行分割，從而降低其準確性。我們需要執行以下步驟來更正文本傾斜。

1.檢測圖像中歪斜的文本塊

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (5, 5), 0)
edged = cv2.Canny(gray, 10, 50)
cnts = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIM
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:5]
screenCnt = None
for c in cnts:
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.02 * peri, True)
    if len(approx) == 4:
        screenCnt = approx
```

```
        break

cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 2)
```



2.計算旋轉角度

3.旋轉圖像以校正歪斜

```
pts = np.array(screenCnt.reshape(4, 2) * ratio)
warped = four_point_transform(orig, pts)
def order_points(pts):
    # initialzie a list of coordinates that will be ordered
    # such that the first entry in the list is the top-left,
    # the second entry is the top-right, the third is the
    # bottom-right, and the fourth is the bottom-left
    rect = np.zeros((4, 2), dtype="float32")

    # the top-left point will have the smallest sum, whereas
    # the bottom-right point will have the largest sum
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]

    # now, compute the difference between the points, the
```

```
    # top-right point will have the smallest difference,
    # whereas the bottom-left will have the largest difference
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]

    # return the ordered coordinates
    return rect


def four_point_transform(image, pts):
    # obtain a consistent order of the points and unpack them
    # individually
    rect = order_points(pts)
    (tl, tr, br, bl) = rect

    # compute the width of the new image, which will be the
    # maximum distance between bottom-right and bottom-left
    # x-coordiates or the top-right and top-left x-coordinates
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))

    # compute the height of the new image, which will be the
    # maximum distance between the top-right and bottom-right
    # y-coordinates or the top-left and bottom-left y-coordinates
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

    # now that we have the dimensions of the new image, construct
    # the set of destination points to obtain a "birds eye view",
    # (i.e. top-down view) of the image, again specifying points
    # in the top-left, top-right, bottom-right, and bottom-left
    # order
    dst = np.array([
        [0, 0],
        [maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],
        [0, maxHeight - 1]], dtype="float32")
```

```
57
58      # compute the perspective transform matrix and then apply it
59      M = cv2.getPerspectiveTransform(rect, dst)
60      warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
61      return warped
```

二值化

通常，OCR引擎會在內部進行二值化處理，因為它們可以處理黑白圖像。最簡單的方法是計算閾值，然後將所有像素轉換為白色，且其值高於閾值，其餘像素轉換為黑色。

除噪或降噪

噪點是圖像像素之間顏色或亮度的隨機變化。噪聲會降低圖像中文本的可讀性。噪聲有兩種主要類型：鹽椒噪聲和高斯噪聲。

```
def remove_noise_and_smooth(file_name):
    img = cv2.imread(file_name, 0)
    filtered = cv2.adaptiveThreshold(img.astype(np.uint8), 255, cv2.ADAPTIVE_TH
    kernel = np.ones((1, 1), np.uint8)
    opening = cv2.morphologyEx(filtered, cv2.MORPH_OPEN, kernel)
    closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
    img = image_smoothening(img)
    or_image = cv2.bitwise_or(img, closing)
    return or_image
```