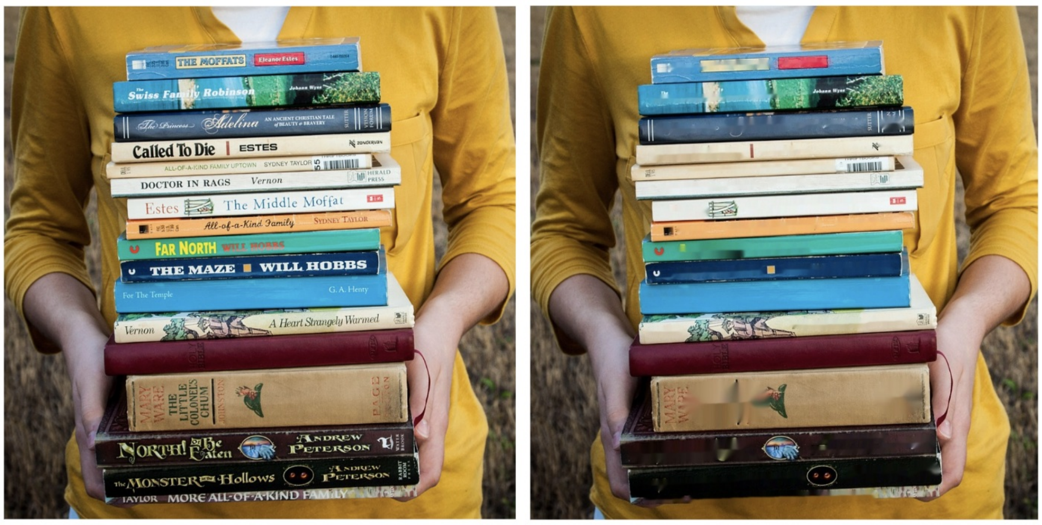
Remove Text from Images using CV2 and Keras-OCR

How to automatically modify images to make them text-free using Python

Carlo Borella 1 day ago · 7 min read



An example of before and after removing text using Cv2 and Keras. Source: image by the author processing an image by morningbirdphoto from Pixabay.

Introduction

In this article I will discuss how to quickly remove text from images as a pre-processing step for an image classifier or a multi-modal text and image classifier involving images with text such as memes (for instance the Hateful Memes Challenge by Facebook).

Removing text can be useful for a variety or reasons, for example we can use the text-free images for data augmentation as we can now pair the text-free image with a new text.

For this tutorial we will use OCR (Optical Character Recognition) to detect text inside images, and inpainting - the process where missing parts of a photo are filled in to produce a complete image - to remove the text we detected.

The Process

In order to erase text from images we will go through three steps:

1. Identify text in the image and obtain the bounding box coordinates of

each text, using Keras-ocr.

2. For each bounding box, apply a mask to tell the algorithm which part of the image we should inpaint.
3. Finally, apply an inpainting algorithm to inpaint the masked areas of the image, resulting in a text-free image, using cv2.

A representation of the process from an image with text to a text-free image. Source: Image by Author.

The Implementation

Brief overview of Keras-ocr

`Keras-ocr` provides out-of-the-box OCR models and an end-to-end training pipeline to build new OCR models (see: https://keras-ocr.readthedocs.io/en/latest/).

In this case we will use the pre-trained model, which works fairly well for our task.

Keras-ocr would automatically download the pre-trained weights for the detector and recognizer.

When passing an image through Keras-orc it will return a (word, box) tuple, where the box contains the coordinates (x, y) of the four box corners of the word.

Here is a quick example:

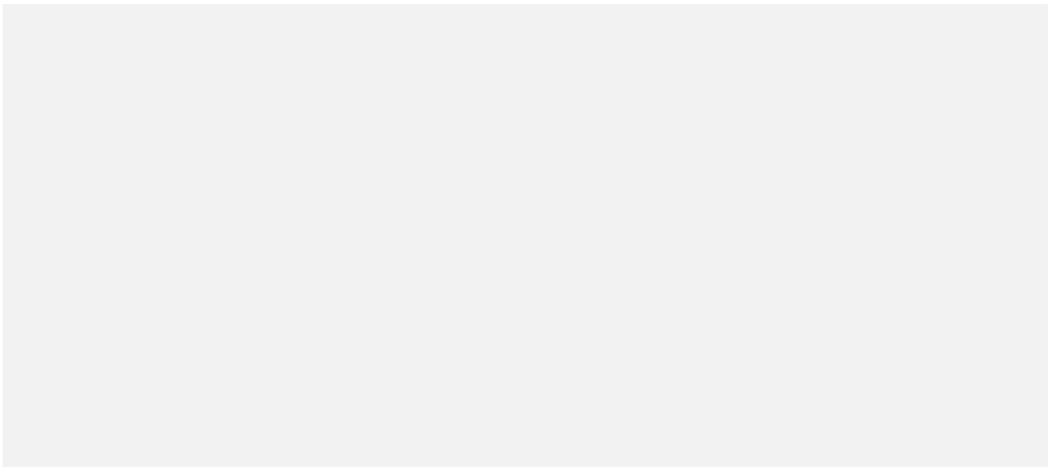
```
import matplotlib.pyplot as plt
import keras_ocr

pipeline = keras_ocr.pipeline.Pipeline()

#read image from the an image path (a jpg/png file or an image url)
img = keras_ocr.tools.read(image_path)

# Prediction_groups is a list of (word, box) tuples
prediction_groups = pipeline.recognize([img])

#print image with annotation and boxes
keras_ocr.tools.drawAnnotations(image=img,
predictions=prediction_groups[0])
```



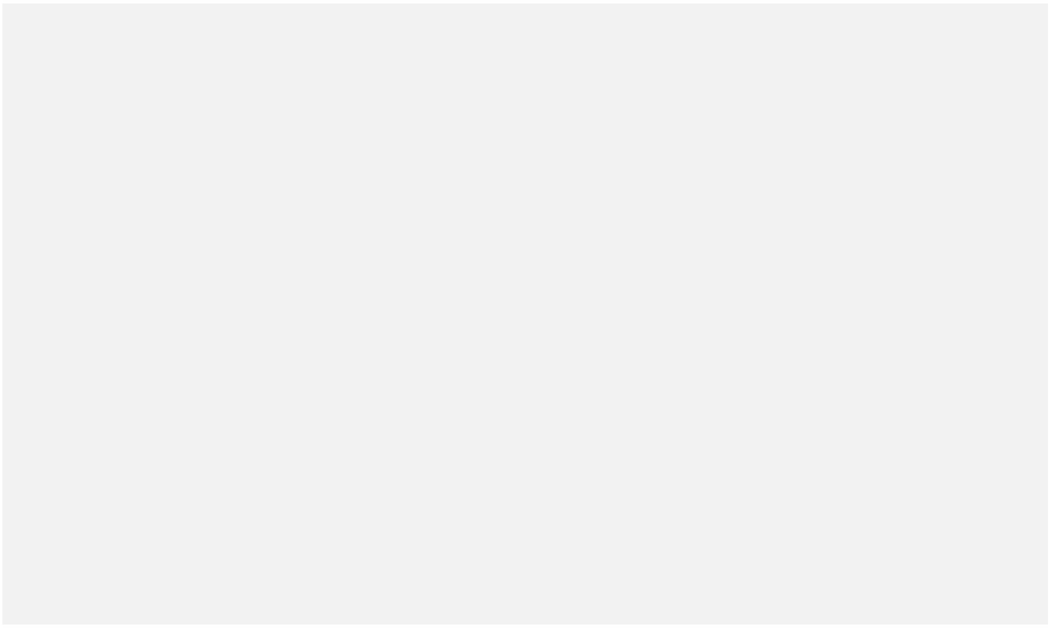
Source: Image by Paul Steuber from Pixabay.

If we take a look at `prediction_groups` we will see that each element corresponds to a pair of word-box coordinates.

For example, `prediction_groups[0][10]` would look like:

```
('tuesday',
 array([[ 986.2778 ,  625.07764],
        [1192.3856 ,  622.7086 ],
        [1192.8888 ,  666.4836 ],
        [ 986.78094,  668.8526 ]], dtype=float32))
```

The first element of the array corresponds to the coordinates of the top-left corner, the second element corresponds to the bottom-right corner, the third elements is the top-right corner, while the fourth element is the bottom-left corner.



A representation of a text bounding box, and its coordinates. Source: Image by Author.

Brief overview of cv2 inpaint functions

When applying an inpainting algorithm using OpenCV we need to provide two images:

1. **The input image with the text we want to remove.**

2. **The mask image, which shows where in the image the text that we want to remove is.** This second image should have the same dimensions as the input. The mask will display non-zero pixels corresponding to those areas of the input image that contain text and would be inpainted, while the areas where we have zero pixels won't be modified.

Cv2 features two possible inpainting algorithms and allows to apply rectangular, circular or line masks (see: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_photo/py_inpainting/py_inpainting.html)

In this case I decided to use line masks, as they are more flexible to cover text with different orientations (rectangular masks would only work well for words parallel or perpendicular to the x-axis and circular masks would cover an area larger than necessary).

In order to apply the mask we need to provide the coordinates of the starting and the ending points of the line, and the thickness of the line:

The start point will be the mid-point between the top-left corner and the bottom-left corner of the box while the end point will be the mid-point between the top-right corner and the bottom-right corner.

For the thickness we will calculate the length of the line between the top-left corner and the bottom-left corner.

```
import math
import numpy as np

def midpoint(x1, y1, x2, y2):
    x_mid = int((x1 + x2)/2)
    y_mid = int((y1 + y2)/2)
    return (x_mid, y_mid)

#example of a line mask for the word "Tuesday"

box = prediction_groups[0][10]

x0, y0 = box[1][0]
x1, y1 = box[1][1]
x2, y2 = box[1][2]
x3, y3 = box[1][3]

x_mid0, y_mid0 = midpoint(x1, y1, x2, y2)
x_mid1, y_mi1 = midpoint(x0, y0, x3, y3)
thickness = int(math.sqrt( (x2 - x1)**2 + (y2 - y1)**2 ))
```

Now we can create our mask:

```
mask = np.zeros(img.shape[:2], dtype="uint8")
cv2.line(mask, (x_mid0, y_mid0), (x_mid1, y_mi1), 255, thickness)
```

We can also check the masked area to make sure it is working properly.

```
masked = cv2.bitwise_and(img, img, mask=mask)
```

```
plt.imshow(masked)
```

The masked area corresponding to the word “Tuesday”.

Finally, we can inpaint the image. In this case we will be using `cv2.INPAINT_NS` which refers to the inpainting algorithm described in the paper **“Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting”.**

```
img_inpainted = cv2.inpaint(img, mask, 7, cv2.INPAINT_NS)
plt.imshow(img_inpainted)
```

As you can see the work “Tuesday” was removed from the image.

The Implementation

Now let’s wrap it up altogether and create a function to inpaint text from any image.

We will just need to generate the list of boxes and iterate masking and inpainting each text box.

```
import matplotlib.pyplot as plt
import keras_ocr
import cv2
import math
import numpy as np

def midpoint(x1, y1, x2, y2):
    x_mid = int((x1 + x2)/2)
    y_mid = int((y1 + y2)/2)
    return (x_mid, y_mid)

pipeline = keras_ocr.pipeline.Pipeline()

def inpaint_text(img_path, pipeline):
    # read image
    img = keras_ocr.tools.read(img_path)
    # generate (word, box) tuples
    prediction_groups = pipeline.recognize([img])
    mask = np.zeros(img.shape[:2], dtype="uint8")
    for box in prediction_groups[0]:
        x0, y0 = box[1][0]
        x1, y1 = box[1][1]
        x2, y2 = box[1][2]
        x3, y3 = box[1][3]

        x_mid0, y_mid0 = midpoint(x1, y1, x2, y2)
        x_mid1, y_mi1 = midpoint(x0, y0, x3, y3)

        thickness = int(math.sqrt( (x2 - x1)**2 + (y2 - y1)**2 ))

        cv2.line(mask, (x_mid0, y_mid0), (x_mid1, y_mi1), 255,
        thickness)
        img = cv2.inpaint(img, mask, 7, cv2.INPAINT_NS)

    return(img)
```

and here is the final result (before vs after):

I also included another couple of examples:

Source: image by the author generated by processing a meme from HackerNoon.

Source: image by the author generated by processing an image by Alfred Derks from Pixabay.

Note that if you want to save the image you will need to convert it to the RGB format, otherwise the colours will be inverted!

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

cv2.imwrite(‘text_free_image.jpg’,img_rgb)
```

In case you were interested in removing certain words only, an if-condition can be included as follows:

Given a list of words to remove

```
remove_list = [‘tuesday’, ‘monday’]
```

We can include the `if` condition in the for-loop

```
def inpaint_text(img_path, remove_list, pipeline):
    # read image
    img = keras_ocr.tools.read(img_path)
    # generate (word, box) tuples
    prediction_groups = pipeline.recognize([img])
    mask = np.zeros(img.shape[:2], dtype="uint8")
    for box in prediction_groups[0]:
        if box[0] in remove_list:

           x0, y0 = box[1][0]
           x1, y1 = box[1][1]
           x2, y2 = box[1][2]
           x3, y3 = box[1][3]
```

```
            x_mid0, y_mid0 = midpoint(x1, y1, x2, y2)
            x_mid1, y_mi1 = midpoint(x0, y0, x3, y3)

            thickness = int(math.sqrt( (x2 - x1)**2 + (y2 - y1)**2
))

            cv2.line(mask, (x_mid0, y_mid0), (x_mid1, y_mi1), 255,
            thickness)
            img = cv2.inpaint(img, mask, 7, cv2.INPAINT_NS)

      return(img)
```

This of course is just a quick case-sensitive example on how to apply the inpainting to just a certain list of words.

End Notes

In this article, we discussed how to implement an algorithm to automatically remove text from images with a pre-trained OCR model using Keras and an inpainting algorithm using cv2. The algorithm seems to work fairly well to quickly remove text from images without the need to train a model for this specific task. It generally performs not as well when a text box is close to other objects as it may distort the surroundings.

I appreciate any feedback and constructive criticism! My email is carbor100@gmail.com

Thanks to Elliot Gunn.

Computer Vision Ocr Python Machine Learning Image Processing