

基于Python的图像隐写术--如何隐藏图像中的数据

原创

小白学视觉 于 2020-10-19 10:05:00 发布 815 收藏 22

文章标签: [算法](#) [python](#) [人工智能](#) [编程语言](#) [深度学习](#)

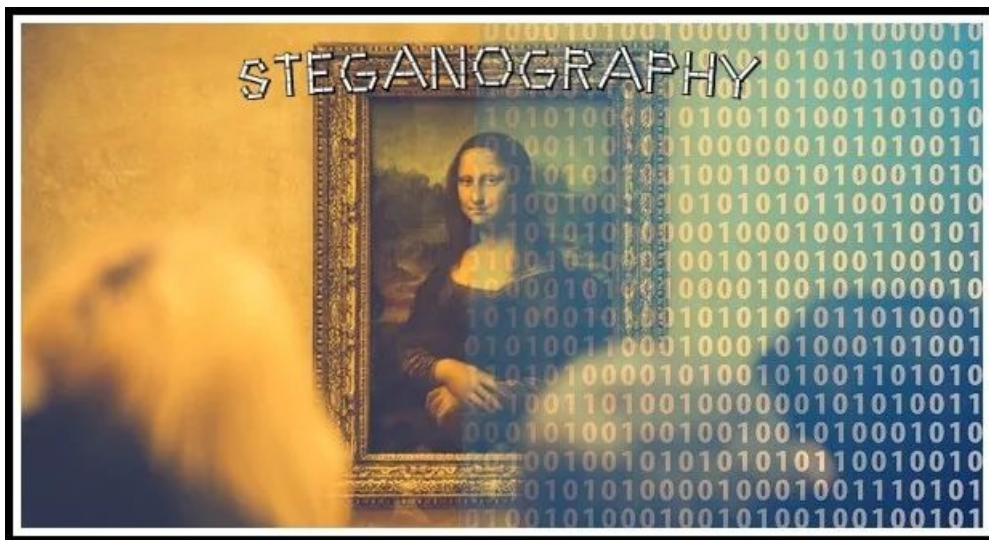
版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_42722197/article/details/109172975

版权

点击上方“小白学视觉”，选择加“星标”或“置顶”

重磅干货，第一时间送达



隐秘术是在任何文件中隐藏秘密数据的艺术。秘密数据可以是任何格式的数据，例如文本，甚至是文件。简而言之，隐写术的主要目的是在任何文件（通常是图像，音频或视频）中隐藏预期的信息，而无需实际更改文件的外观，即，其外观应与以前相同。

本期我们将一起学习基于图像的隐写术，即在图像中隐藏秘密数据。

但是在深入研究之前，让我们先看看图像的组成。

1. 像素是图像的基础。
2. 每个像素包含三个值：（红色，绿色，蓝色）也称为RGB值。
3. 每个RGB值的范围是0到255。

现在，让我们看看如何将数据编码和解码为图像形式。

01.编码方式

有很多算法可将数据编码到图像中，实际上，大家也可以自己制作。我们将使用的一个简单易于理解和实施的算法。具体步骤如下：

1. 对于数据中的每个字符，均采用其ASCII值并将其转换为8位二进制[1]。
2. 一次读取三个像素，总共具有 $3 * 3 = 9$ RGB值。前八个RGB值用于存储一个字符，该字符将转换为8位二进制数。

3. 比较相应的RGB值和二进制数据。如果二进制数字为1，则RGB值将转换为奇数，否则将转换为偶数。

4. 第九个值确定是否应读取更多像素。如果还有更多数据要读取（即编码或解码），则第九个像素变为偶数。否则，如果我们想停止进一步读取像素，则将其设为奇数。

重复此过程，直到所有数据都编码到图像中。

02.应用实例

假设要隐藏的消息是'Hi!'。

该消息为三个字节，因此，对数据进行编码所需的像素为 $3 \times 3 = 9$ 。请考虑一个 4×3 的图像，总共12个像素，足以对给定的数据进行编码。

[(27, 64, 164) , (248, 244, 194) , (174, 246, 250) , (149, 95, 232) ,
(188, 156, 169) , (71, 167, 127) , (132, 173, 97) , (113, 69, 206) ,
(255, 29, 213) , (53, 153, 220) , (246, 225, 229) , (142, 82, 175)]

第一步

的ASCII值为H72，其二进制等效值为01001000。

第二步

读取前三个像素。

(27, 64, 164) , (248, 244, 194) , (174, 246, 250)

第三步

现在，将像素值更改为1的奇数，甚至更改为0的奇数，如数据的二进制等价形式。

例如，第一个二进制数为0，第一个RGB值为27，则需要将其转换为偶数26。

类似地，由于下一个二进制数是1，因此64被转换为63，因此RGB值应设为奇数。

因此，修改后的像素为：

(26, 63, 164) , (248, 243, 194) , (174, 246, 250)

第四步

由于我们必须编码更多的数据，因此最后一个值应该是偶数。同样，i可以在此图像中进行编码。

通过+1或-1使像素值奇/偶时，应注意二进制条件。即，像素值应大于或等于0且小于或等于255。

新图像将如下所示：

[(26, 63, 164) , (248, 243, 194) , (174, 246, 250) , (148, 95, 231) ,
(188, 155, 168) , (70, 167, 126) , (132, 173, 97) , (112, 69, 206) ,
(254, 29, 213) , (53, 153, 220) , (246, 225, 229) , (142, 82, 175)]

03.解码

对于解码，我们将尝试找到如何逆转以前用于编码数据的算法。

1. 同样，一次读取三个像素。前8个RGB值向我们提供有关机密数据的信息，第9个值告诉我们是否继续前进。

2. 对于前八个值，如果该值为奇数，则二进制位为1，否则为0。

3. 这些位被连接成一个字符串，每三个像素，我们得到一个字节的秘密数据，这意味着一个字符。

4. 现在，如果第九个值是偶数，那么我们将一次读取三个像素，否则我们将停止。

让我们开始一次读取三个像素。考虑我们以前编码的图像。

```
[ (26, 63, 164) , (248, 243, 194) , (174, 246, 250) , (148, 95, 231) ,  
  (188, 155, 168) , (70, 167, 126) , (132, 173, 97) , (112, 69, 206) ,  
  (254, 29, 213) , (53, 153, 220) , (246, 225, 229) , (142, 82, 175) ]
```

第1步

我们首先阅读三个像素：

```
[ (26, 63, 164) , (248, 243, 194) , (174, 246, 250)
```

第2步

读取第一个值：26，它是偶数，因此二进制位是0。同样，对63，二进制位1和164它0。该过程一直持续到八个RGB值为止。

第三步

我们最终得到二进制值：01001000将所有单个二进制值连接在一起之后。最终的二进制数据对应于十进制值72，并以ASCII表示字符H。

第4步

由于第九个值是偶数，因此我们重复上述步骤。当遇到的第九个值是奇数时，我们停止。

结果，我们得到的原始消息是Hii。

用于上述算法的Python程序如下：

```
# Python program implementing Image Steganography  
  
# PIL module is used to extract  
# pixels of image and modify it  
from PIL import Image  
  
# Convert encoding data into 8-bit binary  
# form using ASCII value of characters  
def genData(data):  
  
    # list of binary codes  
    # of given data  
    newd = []  
  
    for i in data:  
        newd.append(format(ord(i), '08b'))  
    return newd  
  
# Pixels are modified according to the  
# 8-bit binary data and finally returned  
def modPixels(image, data):
```

```

def modpix(pix, data):

    datalist = genData(data)
    lendata = len(datalist)
    imdata = iter(pix)

    for i in range(lendata):

        # Extracting 3 pixels at a time
        pix = [value for value in imdata.__next__():3] +
              imdata.__next__():3] +
              imdata.__next__():3]]

        # Pixel value should be made
        # odd for 1 and even for 0
        for j in range(0, 8):
            if (datalist[i][j] == '0' and pix[j] % 2 != 0):
                pix[j] -= 1

            elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
                if(pix[j] != 0):
                    pix[j] -= 1
                else:
                    pix[j] += 1
                # pix[j] -= 1

        # Eighth pixel of every set tells
        # whether to stop or read further.
        # 0 means keep reading; 1 means the
        # message is over.
        if (i == lendata - 1):
            if (pix[-1] % 2 == 0):
                if(pix[-1] != 0):
                    pix[-1] -= 1
                else:
                    pix[-1] += 1

            else:
                if (pix[-1] % 2 != 0):
                    pix[-1] -= 1

        pix = tuple(pix)
        yield pix[0:3]
        yield pix[3:6]
        yield pix[6:9]

def encode_enc(newimg, data):
    w = newimg.size[0]
    (x, y) = (0, 0)

```

```

for pixel in modPix(newimg.getdata(), data):

    # Putting modified pixels in the new image
    newimg.putpixel((x, y), pixel)
    if (x == w - 1):
        x = 0
        y += 1
    else:
        x += 1

# Encode data into image
def encode():
    img = input("Enter image name(with extension) : ")
    image = Image.open(img, 'r')

    data = input("Enter data to be encoded : ")
    if (len(data) == 0):
        raise ValueError('Data is empty')

    newimg = image.copy()
    encode_enc(newimg, data)

    new_img_name = input("Enter the name of new image(with extension) : ")
    newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))

# Decode the data in the image
def decode():
    img = input("Enter image name(with extension) : ")
    image = Image.open(img, 'r')

    data = ''
    imgdata = iter(image.getdata())

    while (True):
        pixels = [value for value in imgdata.__next__()[ :3] +
                  imgdata.__next__()[ :3] +
                  imgdata.__next__()[ :3]]

        # string of binary data
        binstr = ''

        for i in pixels[:8]:
            if (i % 2 == 0):
                binstr += '0'
            else:
                binstr += '1'

        data += chr(int(binstr, 2))

```

```

        if (pixels[-1] % 2 != 0):
            return data

# Main Function
def main():
    a = int(input(":: Welcome to Steganography ::\n"
                 "1. Encode\n2. Decode\n"))

    if (a == 1):
        encode()

    elif (a == 2):
        print("Decoded Word : " + decode())
    else:
        raise Exception("Enter correct input")

# Driver Code
if __name__ == '__main__':

    # Calling main function
    main()

```

该程序中使用的模块是PIL，代表Python Imaging Library。它使我们能够在Python中对图像执行操作。

04.程序示例

```

ashwin@ash ~/Desktop
└─$ python3 steg.py
### Welcome to Steganography ###
1. Encode
2. Decode
1
Enter image name(with extension) : medium.png
Enter data to be encoded : Steganography
Enter the name of new image(with extension) : newImage.png
Data encoded successfully!!%

```

数据编码

```

ashwin@ash ~/Desktop
└─$ python3 steg.py
### Welcome to Steganography ###
1. Encode
2. Decode
2
Enter image name(with extension) : newImage.png
Decoded word- Steganography

```

数据解码

输入图像



medium.png

输出图像



newImage.png

05.局限性

对于JPEG 图像，此程序可能无法正常工作，因为JPEG使用有损压缩，这意味着修改了像素以压缩图像并降低质量，因此发生数据丢失。

参考文献

<https://www.geeksforgeeks.org/program-decimal-binary-conversion/>

<https://www.geeksforgeeks.org/working-images-python/>

<https://dev.to/erikwhiting88/let-s-hide-a-secret-message-in-an-image-with-python-and-opencv-1jf5>

A GUI version of the program can be found here: <https://github.com/goelashwin36/image-steganography>

交流群

欢迎加入公众号读者群一起和同行交流，目前有SLAM、三维视觉、传感器、自动驾驶、计算摄影、检测、分割、识别、医学影像、GAN、算法竞赛等微信群（以后会逐渐细分），请扫描下面微信号加群，备注：“昵称+学校/公司+研究方向”，例如：“张三 + 上海交大 + 视觉SLAM”。请按照格式备注，否则不予通过。添加成功后会根据研究方向邀请进入相关微信群。请勿在群内发送广告，否则会请出群，谢谢理解~



小白学视觉

计算机视觉

论文解读 求职感想

SLAM技术 深度学习 学习感受

距离我们只差一个
长按关注

聚集地
计算机视觉学者

