# python图像隐写_基于Python-PIL的隐写算法

原创

彼岸枫桥 于 2021-01-29 17:10:30 发布 179 收藏
文章标签： python图像隐写

本文链接：https://blog.csdn.net/weixin_33834241/article/details/113510015

我试图用Python-PIL为黑白图像编写基本的隐写算法。在

使用一个示例图像，我可以成功地提取其中隐藏的图像，并隐藏其他图像以随后提取它们。问题是隐藏文本然后提取它。在

代码如下：from PIL import Image

```
import matplotlib.pyplot as plt

import scipy.misc as sci

import numpy as np

import array

#CONVERTS IMAGE TO ARRAY OF BINARY 8-BIT NUMBER#

def getImgArray(image):

w,h = image.size

out = []

for x in range(w):

for y in range(h):

pixel = image.getpixel((y,x))

pixel = format(pixel, '08b')

out.append(pixel)

return out

def stringByteConverter(data, mode):

if (mode == "stringToByte"):

aux = map(ord,data.encode('utf8'))

aux = [format(char,'08b') for char in aux]

return aux

elif (mode == "byteToString"):

aux = [int(item,2) for item in data]

aux = "".join(map(chr, aux))
```

```
return aux
else:
print("Invalid mode. Use 'stringToByte' or 'byteToString'")
#GETS HIDDEN IMAGE AND RETURNS IT AS BYTE ARRAY REPRESENTING PIXELS#
def getHiddenImage(image):
buf = ""
width,height = image.size
img_aux = []
for x in range(width):
for y in range(height):
if(len(buf)<8):
pixel = image.getpixel((y,x))
pixel = format(pixel,'08b')
buf += pixel[-2:]
else:
img_aux.append(buf)
buf = ""
pixel = image.getpixel((y,x))
pixel = format(pixel,'08b')
buf += pixel[-2:]
return img_aux
#CONVERT ARRAY OF BYTES TO PNG IMG AND RETURNS PIL IMG OBJECT#
def saveImgArr(ImgArr, size, outputName):
pixels = np.empty(size)
iterator = 0
for i in range(size[0]):
for j in range(size[1]):
try:
pixels[i][j] = int(ImgArr[iterator],2)
iterator += 1
except IndexError:
```

```
break
aux = Image.fromarray(pixels)
aux = aux.convert("L")
aux.save(outputName+'.png', 'PNG')
return pixels
                                        #HIDE IMAGE IN OTHER IMAGE #
def hideImg(src, img, output):
iterator = 0
src = src.convert("L")
srcArr = getImgArray(src)
imgArr = getImgArray(img)
for i in range(len(srcArr)):
buf = []
buf.append(srcArr[i][:2])
buf.append(srcArr[i][2:4])
buf.append(srcArr[i][4:6])
buf.append(srcArr[i][6:])
for j in range(4):
imgArr[iterator] = imgArr[iterator][:-2] + buf[j]
iterator += 1
saveImgArr(imgArr,img.size,output)
#HIDE STRING INSIDE IMG#
def hideText(img, string, outputName):
imgArr = getImgArray(img)
stringBytes = stringByteConverter(string, "stringToByte")
iterator = 0
for i in range(len(string)):
buf = []
buf.append(stringBytes[i][:2])
buf.append(stringBytes[i][2:4])
buf.append(stringBytes[i][4:6])
```

```
buf.append(stringBytes[i][6:])
for j in range(4):
imgArr[iterator] = imgArr[iterator][:-2] + '00'
imgArr[iterator] = imgArr[iterator][:-2] + buf[j]
iterator += 1
print(imgArr[:len(string)*4]) #test print
saveImgArr(imgArr,img.size,outputName)
temp = Image.open(outputName+'.png')
tempArr = getImgArray(temp)
print(tempArr[:len(string)*4]) #test print
def getHiddenText(img, msgSize):
buf = ''
width,height = img.size
output = []
counter = 0
for x in range(width):
for y in range(height):
if(counter < msgSize*4):
pixel = img.getpixel((y,x))
pixel = format(pixel,'08b')
buf += pixel[-2:]
counter += 1
output = stringByteConverter(buf, "byteToString")
return output
```

通过在hideText()函数中打印数据数组，我可以获得以下结果：

```
^{pr2}$
['10100001', '10100011', '10100001', '10100000', '10100001',
'10011110', '10100001', '10100001', '10100101', '10100011',
'10100000', '10011111', '10011001', '10100011', '10011101',
'10011000']
['10011110', '10100000', '10011110', '10011101', '10011110',
```

'10011011', '10011110', '10011110', '10100011', '10100000',

'10011101', '10011100', '10010101', '10100000', '10011001',

'10010100']

hideText()调用获得的第一个向量与实际情况完全相同，但是在使用saveImgArr()保存图像并使用getImgArr()重新加载之后，将返回第二个向量，它完全不同。在

我一辈子都找不到问题。奇怪的是，使用图像提取隐藏数据或隐藏数据，这两个功能都能完美地工作。在

我只能猜测我在某种程度上处理了错误的文本字节。任何有见识的人都将不胜感激。在