**《数字内容安全》实验报告**

**姓 名：** 丁昭旭  **学 号：** 16020031011

**年 级：** 2016级 **专 业：** 计算机科学与技术

1. **实验名称**

实验二 数字水印实验

1. **实验要求**

* 利用LSB算法将自己姓名隐藏到一张彩色图片（R通道）中。
* 可以从图片中提取出隐藏的信息。

1. **实验结果**
2. 根据实验要求，我选用方便且可扩展性强的Python语言进行编写。
3. 首先，创建一个文件insert.py用于实现向彩色位图中通过LSB算法插入图片信息，首先编写一个函数get\_hidden\_data用于将隐藏的数据以比特流的形式得到：

*def get\_hidden\_data():*

*hide = Image.open(r"hidden.bmp")*

*c, d = hide.size*

*print(c, d)*

*m = ''*

*for i in range(c):*

*for j in range(d):*

*m = m + str(bin(hide.getpixel((i, j))).lstrip('0b')).zfill(8)*

*return m*

1. 在主函数中调用之前定义的函数提取需要插入的信息，以hidden\_data这个变量存储之。之后使用python的图像处理库PIL的自带函数getpixel和putpixel十分方便地实现插入和更改像素的操作，具体代码如下：

*if \_\_name\_\_ == '\_\_main\_\_':*

*hidden\_data = get\_hidden\_data()*

*num = 0*

*image = Image.open("lenna\_ori.bmp")*

*a, b = image.size*

*print('Original Pic Size:', a, b)*

*print('Length of Hidden Data:', len(hidden\_data))*

*if a \* b < len(hidden\_data):*

*print("Hidden Pic is Too Big!")*

*for i in range(a):*

*for j in range(b):*

*if (num == len(hidden\_data)):*

*break*

*a = image.getpixel((i, j))*

*a1 = a[0] - mod(a[0], 2) + int(hidden\_data[num])*

*image.putpixel((i, j), (a1, a[1], a[2]))*

*num += 1*

*image.save(r"lenna.bmp")*

在这段代码中，首先以两个图片的大小能否实现LSB算法，然后在一个二层循环中对隐藏信息的每8bit进行操作，对原本图像的每一个像素进行循环，若隐藏的数据数组还没有循环完毕，则修改R通道的最后一位，将其修改成隐藏数据的一个信息，直到所有隐藏信息被插入到位图数据中。

1. 再编写一个extract.py文件用于实现提取隐藏信息，其具体代码如下：

*if \_\_name\_\_ == '\_\_main\_\_':*

*img = Image.new("L", (67, 60))*

*inserted\_image = Image.open("lenna.bmp")*

*c, d = img.size*

*a, b = inserted\_image.size*

*print('Size of Hidden Pic:', c, d)*

*print('Size of Inserted Pic:', a, b)*

*num = 0*

*m = ""*

*s = []*

*for q in range(a):*

*for p in range(b):*

*m = m + str(mod(inserted\_image.getpixel((q, p))[0], 2))*

*num += 1*

*if (num == 8):*

*m = "0b" + m*

*s.append(eval(m))*

*num = 0*

*m = ""*

*count = 0*

*for i in range(c):*

*for j in range(d):*

*if (count == len(s)):*

*break*

*img.putpixel((i, j), s[count])*

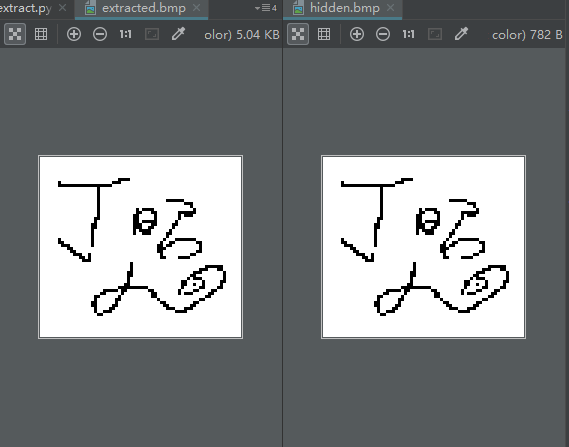
*count += 1*

*img.save(r"extracted.bmp")*

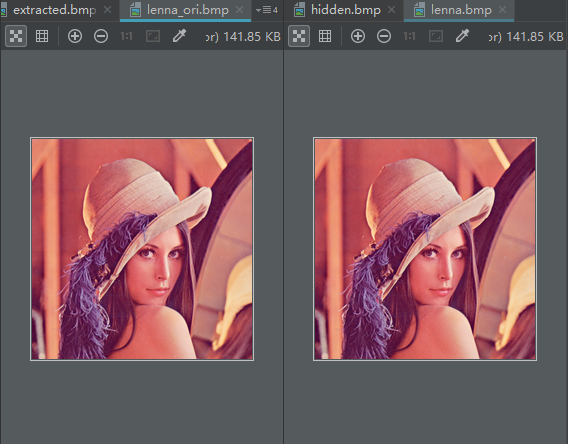
在这段代码中，首先通过PIL库建立了一个空的Image用于书写隐藏位图数据。之后通过两层循环对于插入隐藏信息后的图片进行隐藏信息提取，每提取8Bit则保存一次，将保存的数据存储在一个数组s中。

两次循环之后，所有隐藏信息已经存储在了s数组中，之后则通过另一个双层循环将s数组中的数据绘制在之前创建的图像中，将隐藏数据保存。

1. 下面是插入的隐藏信息和提取出的信息对比：

可以看出除了图片大小之外，隐藏的信息已经完全还原出来。

下面是未插入前的图片与插入数据后的图片对比：



可以看出插入隐藏数据之后原图片并未失真，算法成功！

1. **实验总结**

通过此次试验，我对LSB算法有了一定程度的认识，同时对于BMP位图的实现有了更深刻的理解，特别是RGB色彩通道相关的知识，也对比特的操作有了更熟练的掌握。

通过PIL库强大的功能，我可以省去对BMP文件的复杂操作，而直接对像素点的RGB值进行设置，极大提升了效率，也认识到了Python语言广阔的库世界。希望以后能够更多地了解PIL库更多的高级功能，能够充分挖掘它的优点。

希望以后的实验能够再接再厉，更出色地完成任务！