1. **将一幅彩色图片，变成灰度图。比较变换前后Bitmap格式和JPG格式图像大小**

**解：**

1. **技术思路、主要计算公式**
2. 利用python的opencv2库，imread读取老师给的JPG图片，imwrite写入BMP图片
3. 利用cvtcolor 和 imwrite函数将两幅JPG和BMP图片转换为灰度图，比较转换前后大小
4. **核心代码**

import cv2 as cv

import random

import time

#将jpg转换为bmp

img = cv.imread('E:/2019HW\_1.jpg', 1)

cv.imwrite('E:/2019HW\_1.bmp', img)

#jpg和bmp转为灰度图

img1 = cv.imread('E:/2019HW\_1.jpg', 1)

img1 = cv.cvtColor(img1, cv.COLOR\_BGR2GRAY)

cv.imwrite('E:/2019HW\_1\_gray.jpg', img1)

img2 = cv.imread('E:/2019HW\_1.bmp', 1)

img2 = cv.cvtColor(img2, cv.COLOR\_BGR2GRAY)

cv.imwrite('E:/2019HW\_1\_gray.bmp', img2)

1. **运行结果**

主机配置: CPU :intel core i5-7300 2.50GHZ RAM :8.0GB

运行环境：win10 64位操作系统

开发环境：python3.7

执行时间：2019/9/25

输入文件名：E:/2019HW\_1.jpg

输出文件名：E:/2019HW\_1.bmp E:/2019HW\_1\_gray.jpg E:/2019HW\_1\_gray.bmp

运行时间：

（包含imread读取原始矩阵时间）

JPG to BMP use 0.56701 s

JPG to GRAY use 1.62922 s

BMP to GRAY use 0.16060 s

变换前JPG大小：5921kb 5.78mb

变换前BMP大小：70313kb 68.6mb

JPG灰度图大小：6270kb 6.12mb

BMP灰度图大小：23439kb 22.8mb

图片展示：

|  |  |
| --- | --- |
| 2019HW_1 | 2019HW_1 |
| 2019HW\_1.jpg | 2019HW\_1.bmp |
| 2019HW_1_gray | 2019HW_1_gray |
| 2019HW\_1\_gray.jpg | 2019HW\_1\_gray.bmp |

1. **压缩BMP图片，从BMP到JPG，比较不同压缩率下的计算时间，主观评价容忍度(1到5级)**

**解：**

1. **技术思路、主要计算公式**

利用imwrite函数及其质量压缩参数，将1中生成的BMP图片压缩，质量分别为100%，90%，80%，...20%

1. **核心代码**

for num in range(0,9):

img = cv.imread('E:/2019HW\_1.bmp', 1)

start = time.time()

cv.imwrite('E:/lossy'+str(num)+'.jpg',img,[int(cv.IMWRITE\_JPEG\_QUALITY),100-num\*10])

end = time.time()

print('lossy %d use %.5f s' %(100-num\*10,end-start))

1. **运行结果**

主机配置: CPU :intel core i5-7300 2.50GHZ RAM :8.0GB

运行环境：win10 64位操作系统

开发环境：python3.7

执行时间：2019/9/25

运行时间：

lossy 100 use 1.24622 s

lossy 90 use 0.87377 s

lossy 80 use 0.81230 s

lossy 70 use 0.77153 s

lossy 60 use 0.72500 s

lossy 50 use 0.68075 s

lossy 40 use 0.66757 s

lossy 30 use 0.64432 s

输入文件名：E:/2019HW\_1.bmp

输出文件名：E:/lossy100.jpg E:/lossy90.jpg ... E:/lossy30.jpg

变换前JPG大小：5921kb 5.78mb

变换前BMP大小：70313kb 68.6mb

JPG灰度图大小：6270kb 6.12mb

BMP灰度图大小：23439kb 22.8mb

图片展示：

|  |
| --- |
| lossy100 |
| lossy100.jpg |
| lossy90 |
| lossy90.jpg |
| lossy80 |
| lossy80.jpg |
| lossy70 |
| lossy70.jpg |
| lossy60 |
| Lossy60.jpg |
| lossy50 |
| Lossy50.jpg |
| lossy40 |
| lossy40.jpg |
| lossy30 |
| lossy30.jpg |

视觉容忍度：

压缩质量为100 90 80 70 60时为5级

压缩质量为 50 40 时为4级

压缩质量为 30 时为3级

运行时间：

lossy 100 use 1.24622 s

lossy 90 use 0.87377 s

lossy 80 use 0.81230 s

lossy 70 use 0.77153 s

lossy 60 use 0.72500 s

lossy 50 use 0.68075 s

lossy 40 use 0.66757 s

lossy 30 use 0.64432 s

观察分析可知压缩质量越好，耗费时间越长

1. **在BMP图片内嵌入一段文字，并正确提取**

**解：**

1. **技术思路、主要计算公式**

1.编写两个函数embed和extract，embed函数的输入为BMP文件的文件名，嵌入的字符串及输出文件名，extract函数的输入为包含隐藏信息的BMP文件，执行后在屏幕输出嵌入的字符串

为了提高隐藏能力，采用随机选取像素点存储信息的方式

2.embed函数

embed(text,src,output)

return x,y

#text为包含输入字符串的txt文件，src为原始bmp，output为隐藏信息的bmp文件名

#return list x,y为隐藏了信息的pixels坐标

用open()读文件中字符串 ord()将字符转ascii bin()转二进制 不足8bits的补全8bits。之后存储到str，相当于把信息转化为二进制

利用imread读取原始BMP矩阵，size()获取长宽高后用random.sample获取随机pixels的x,y坐标

之后cv2.split函数分离出R通道，将信息分bit写到选取好的pixels的最后一位(每bit选取一个像素点)

之后cv2.mrge函数合并写入了信息的R通道和原始BG通道

3.extract函数

extract(hidebmp,x,y):

hidebmp为隐藏了信息的文件名 list x，y为隐藏了信息的pixels坐标

利用imread读取hidebmp的R通道，将选取像素每8个为一组恢复原始ascii字符，合并即为原始信息

1. **核心代码**

def embed(text,src,output):

#open()读文件中字符串 ord()将字符转ascii bin()转二进制 不足8bits的补全8bits。之后存储到strbin

#strbin存储转换为8位二进制ascii的信息，lenth为strbin长度，也即信息总bit数

f=open(text,'r',encoding='utf-8')

res=f.read()

#print(res)

strbin=''

lenth=0

for c in res:

k=bin(ord(c)).replace('0b', '')

if len(k)<8:

k='0'\*(8-len(k))+k

strbin=strbin+k

lenth=lenth+8

#strbin=strbin+'s'+(bin(ord(c)).replace('0b', ''))

f.close()

#读图并获得高度宽度

img1 = cv.imread(src, 1)

size=img1.shape

width=size[1]

height=size[0]

#随机选取lenth个像素用于隐藏信息

x=random.sample(range(1,width-1),lenth); #选取了pixels的x,y坐标,不重复;

y=random.sample(range(1,height-1),lenth);

#print(type(x)) #隐藏了信息的像素点坐标存在list x,y中

# 通道分离，注意顺序BGR不是RGB

#(B, G, R) = cv2.split(image)

#提取R通道，在所选像素最后一位写入信息

hideimg=cv.split(img1)[2]

for i in range(0,lenth):

#print(hideimg[y[i],x[i]])

#print(bin(hideimg[y[i],x[i]]))

tmp=bin(hideimg[y[i],x[i]])

tmp=tmp[0:-1]+strbin[i]

#print(tmp)

#print(strbin[i])

rep=int(tmp,2)

#print(rep)

hideimg[y[i],x[i]]=rep

bimg=cv.split(img1)[0] #src的B通道

gimg=cv.split(img1)[1]

# 将写入信息的R通道和原始BG通道合并

merged=cv.merge([bimg,gimg,hideimg])

cv.imwrite(output, merged)

return x,y

def extract(hidebmp,x,y):

img1 = cv.imread('E:/merged.bmp', 1)

readimg=cv.split(img1)[2]

output=''

rang=int(len(x)/8) #隐藏信息字符个数

#恢复信息

for i in range(0,rang):

result=''

for j in range(0,8):

k=i\*8+j

result=result+bin(readimg[y[k],x[k]])[-1]

output=output+chr(int(result,2))

#final=result.decode('utf-8')

print(output)

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

message='E:/names.txt' #包含信息的TXT文件

srcbmp='E:/2019HW\_1.bmp' #原始bmp文件

outputbmp='E:/merged.bmp' #隐藏了信息的bmp文件名

x,y=embed(message,srcbmp,outputbmp)

extract(outputbmp,x,y)

**（3）运行结果**

主机配置: CPU :intel core i5-7300 2.50GHZ RAM :8.0GB

运行环境：win10 64位操作系统

开发环境：python3.7

执行时间：2019/9/26

输入文件名：'E:/names.txt' 'E:/2019HW\_1.bmp'

输出文件名：'E:/merged.bmp'

运行时间：

Embed隐藏信息use 0.86765 s

Extract读取信息use 0.08378 s

图片展示：



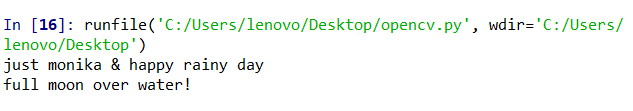
包含信息的txt文件 'E:/names.txt'



原始bmp图片 E:/2019HW\_1.bmp



隐藏了信息的bmp图片 'E:/merged.bmp'



控制台输出信息