浙江大学

数字媒体资源管理实验报告

Digital Assert Management

(Experiments)

姓　　名	李沛瑶
指导老师	张宏鑫
学　　号	3180101940
专业班级	数媒 1801

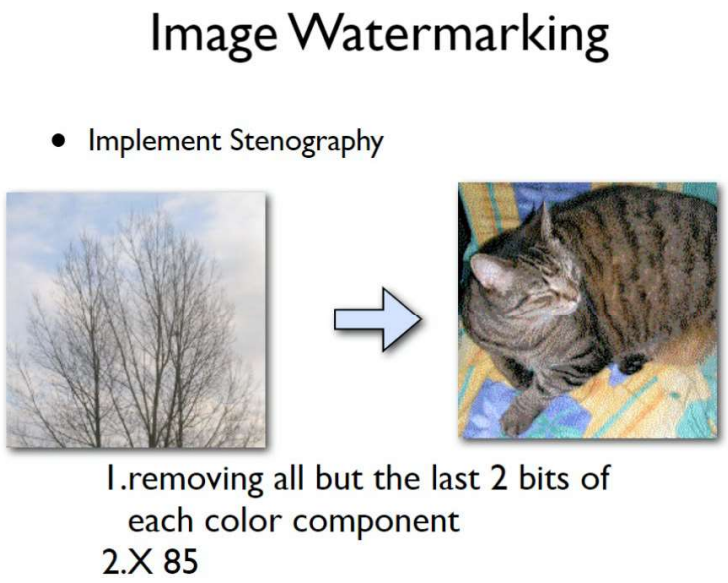
二〇二〇年

秋冬学期

实验四

实验名称： 数字水印 **指导老师：** 张宏鑫 **成绩：**

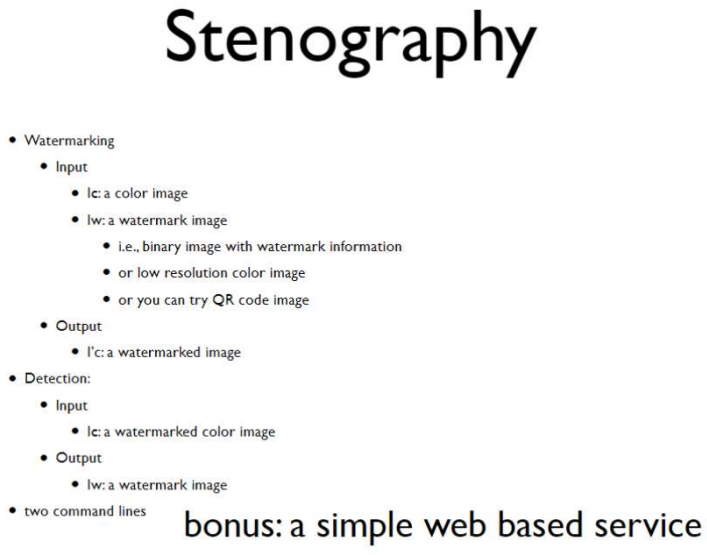
一、实验目的和要求



1. 通过命令行实现图像水印的添加和检测
2. 实现基于网页服务的水印添加和检测

二、实验内容和原理

1、实验内容



2、实验原理及实验过程

嵌入水印原理：使用 Python Image Library 库，读入一张载体图像以及水印图像，首先使用 resize 函数将水印图像缩放成和载体图像一样的大小。之后将载体图像的像素和水印图像的像素点一一对应，将载体图像的像素值（类型为 unsigned char，一共八位）抹去最后两位，将水印图像的像素值除以 85 作为最低两位，加入到载体图像的像素值中。

提取水印原理：使用 Python Image Library 库，读入一张已经嵌入水印的载体图像，对每一个像素点，提取出最后两位的数据，转化成 unsigned char 类型并乘 85，得到提取出的水印图像。

制作水印嵌入网页交互：使用 flask 框架，编写 html 网页，实现从网页选取载体图像以及水印图像，并且使用调用 python 文件的形式对图像进行嵌入水印、提取水印、比对水印操作，并将结果展示在网页中。

三、源代码与分析

1、main.py

```
1.  from flask import Flask, request
2.  from flask import render_template
3.  import os
4.  from PIL import Image
5.  import watermark
6.
7.
8.  app = Flask(__name__)
9.
10.
11. # 主页
12. @app.route('/', methods=['GET', 'POST'], strict_slashes=False)
13. def home():
14.     return render_template('home.html')
15.
16.
17. # 显示水印结果
18. @app.route('/upload', methods=['POST'], strict_slashes=False)
19. def Watermark():
```

```
    pic_addr=request.files['pic_addr']
    pic_path="./static/images/"+pic_addr.filename
    name, category = os.path.splitext(pic_addr.filename)  # 分解文件扩展名
    marked_path = "./static/images/" + name+"_marked.png"
    decode_path = "./static/images/" + name + "_decode.png"
    mark_addr=request.files['mark_addr']
    mark_path="./static/images/"+mark_addr.filename
    img_mark = Image.open(mark_path).convert("RGB")
    mark_path = "./static/images/" + name + "_mark.png"
    img_mark.save(mark_path)
    watermark.addWaterMark(pic_path, mark_path, marked_path)
    watermark.testWaterMark(marked_path, mark_path, decode_path)

    return render_template('watermark.html', pic_addr=pic_path, mark_addr=mark_path, marked_addr=marked_path, decode_addr=decode_path)


if __name__ == '__main__':
    app.run(debug=False, use_reloader=False)
```

2、watermark.py

```
# coding:utf-8
from PIL import Image, ImageChops
import os


# 为图像嵌入水印
def addWaterMark(pic, mark, marked):
    # 读取载体图像
    img = Image.open(pic).convert("RGB")
    width, height = img.size  # 载体图像大小

    # 读取水印图像
    img_mark = Image.open(mark).convert("RGB")
    img_mark = img_mark.resize((width, height))  # 缩放水印图像大小

    # 处理图像中的源数据
    img = img.point(lambda i: (int(i >> 2)) << 2)
    img_mark = img_mark.point(lambda i: round(i / 85))

    # 将图像数据转化成 list
    img_pixels = list(img.getdata())
```

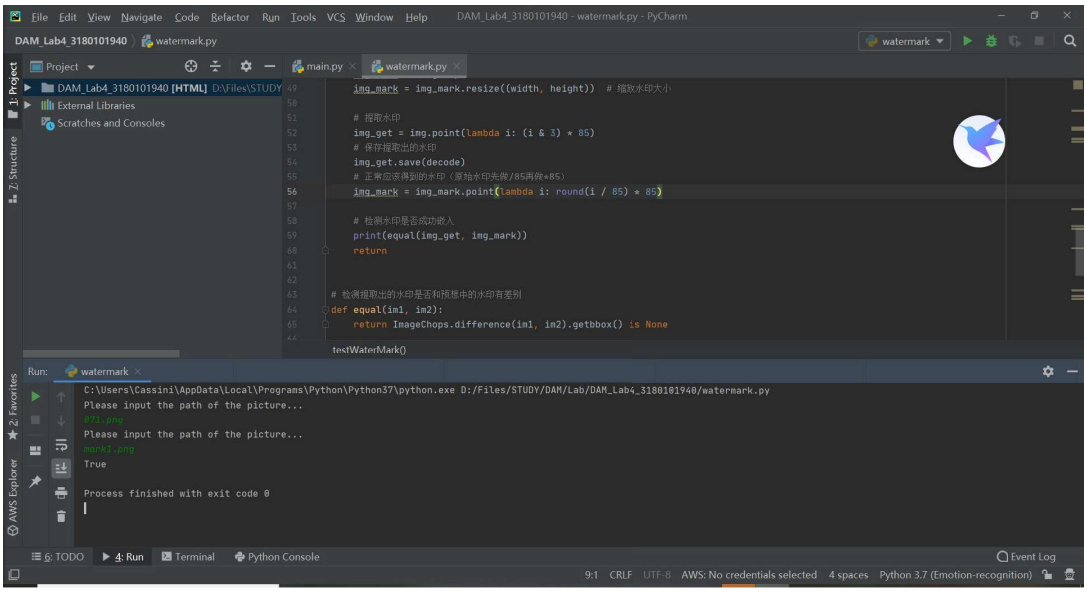
```
    mark_pixels = list(img_mark.getdata())

    # 嵌入水印信息
    new_pixels = []
    for index in range(len(img_pixels)):
        # 处理 RGB 三个通道的分量
        pixel_temp = []
        for i in range(3):
            pixel_temp.append(img_pixels[index][i] + mark_pixels[index][i])
        new_pixels.append(tuple(pixel_temp))

    # 创建新图像
    image_new = Image.new("RGB", (width, height))
    image_new.putdata(data=new_pixels)
    # 保存加水印后的图像
    image_new.save(marked)
    return


# 提取并检测水印，检测用于测试水印是否成功嵌入
def testWaterMark(pic, mark, decode):
    # 读取已经嵌入水印的图像
    img = Image.open(pic).convert("RGB")
    width, height = img.size  # 读取图像大小

    # 读取原本的水印图像
    img_mark = Image.open(mark).convert("RGB")
    img_mark = img_mark.resize((width, height))  # 缩放水印大小

    # 提取水印
    img_get = img.point(lambda i: (i & 3) * 85)
    # 保存提取出的水印
    img_get.save(decode)
    # 正常应该得到的水印（原始水印先做/85 再做*85）
    img_mark = img_mark.point(lambda i: round(i / 85) * 85)

    # 检测水印是否成功嵌入
    print(equal(img_get, img_mark))
    return


# 检测提取出的水印是否和预想中的水印有差别
def equal(im1, im2):
    return ImageChops.difference(im1, im2).getbbox() is None
```

```
66.
67.
68. # 测试用
69. if __name__ == '__main__':
70.     print("Please input the path of the picture...")
71.     pic_path=input()
72.     name, category = os.path.splitext(pic_path)  # 分解文件扩展名
73.     pic_path="./static/images/"+pic_path
74.     marked_path="./static/images/"+name+"_marked.png"
75.     decode_path = "./static/images/" + name + "_decode.png"
76.     print("Please input the path of the picture...")
77.     mark_path = "./static/images/" + input()
78.     img_mark = Image.open(mark_path).convert("RGB")
79.     mark_path = "./static/images/" + name + "_mark.png"
80.     img_mark.save(mark_path)
81.     addWaterMark(pic_path, mark_path, marked_path)
82.     testWaterMark(marked_path, mark_path, decode_path)
```

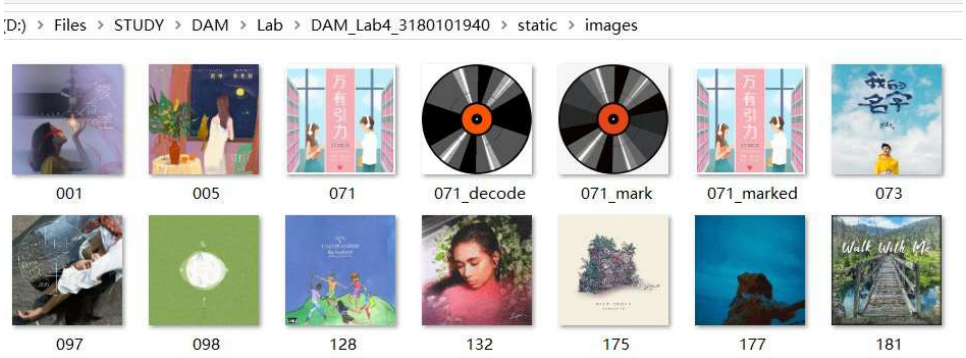
四、效果展示

1、 请将需要用到的载体图像和水印图像放置到 'static/images' 目录下。如图，Chinese和Others分别为第一次作业时收集的资源文件的两个分类。

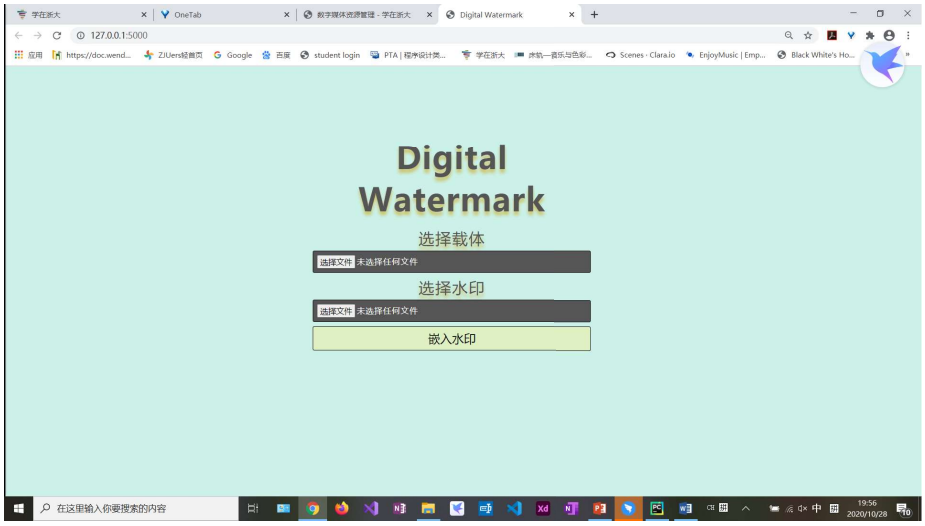
2、 将文件夹作为python工程打开（请注意将工程文件移入没有中文的路径下，否则可能会运行失败），运行“watermark.py”，即可在控制台通过输入文件名来嵌入水印，注意请提前将需要用到的图像文件放入 'static/images' 目录下，在控制台只需输入文件名（包括扩展名），效果如下图：



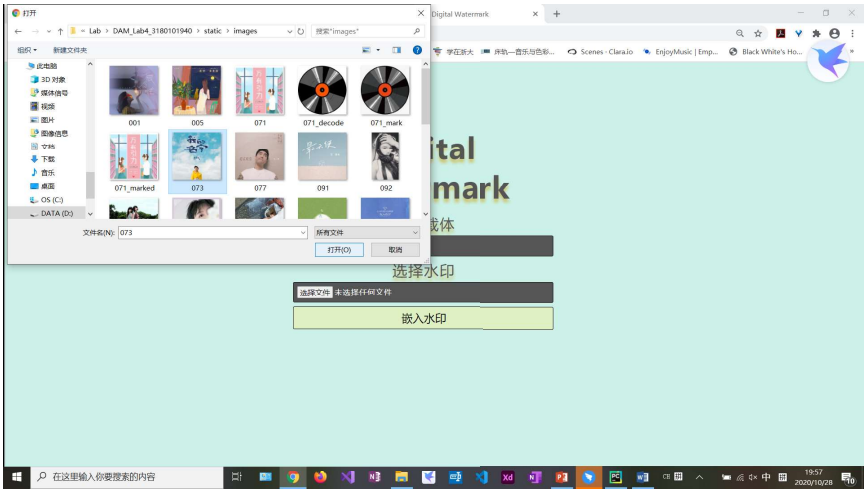
运行后，'static/images' 目录下，会对应生成并存储对应的图像文件。带有后缀“_mark”的图像是嵌入的水印图像文件，带有后缀“_marked”的图像是已经嵌入水印的图像文件，带有后缀“_decode”的图像是从图像中提取出的水印图像文件。



3、运行“main.py”，即可打开选择图像并嵌入水印的网页，效果如图：



可以点击选择文件自行选择载体图像和水印图像（请在'static/images' 目录下选择）。



点击嵌入水印，网页显示嵌入水印、提取水印的结果：

点击返回首页可以回到选择图像的页面，重新选择载体图像和水印图像并进行嵌入。