浙江大学

课程名称：数字媒体资源管理

姓　　名：刘佳润

学　　院：计算机科学与技术学院

专　　业：数字媒体技术

学　　号：3180105640

指导教师：张宏鑫

2020　年　10　月　29　日

浙江大学实验报告

课程名称：数字媒体资源管理 实验类型：综合

实验项目名称：作业 4：数字水印

学生姓名：刘佳润 专业：数字媒体技术 学号：3180105640

同组学生姓名： 指导老师：张宏鑫

实验地点： 实验日期：2020 年 10 月 29 日

一、 实验目的和要求

实现 Stenography 数字水印添加，制作一个网页端应用。

要求：基于 PIL 库或者 OpenCV

二、 实验内容和原理

图片的存储：每位像素是 8 个 bit。

Stenography 隐写术：

水印添加：提取水印图片的每位像素最高两位，除以 85 取整，与目标图片每位像素除去最低两位相叠加。这样得到的图片包含了大多数原图片的信息，并且隐含了水印图片的部分信息。

水印提取：将添加水印后的图片的每个像素最低两位提取，乘以 85，复现处水印图片。

三、 实验器材

开发工具：Python 3.7 Flask PIL 库

四、 实验步骤

1. 添加水印

针对单个像素字节的处理——利用位运算

```
def enco(s, w):
    # s为原图，w为水印
    # X & 1100 0000 可以得到X的最高2位
    # X & 1111 1100 可以得到X除最低2位外其他部分
    return int((w & 192) / 85) + (s & 252)
```

添加水印的函数：

```
def add_watermark(src, watermark=WM_PATH):
    filename = src.split("\\")[-1].split(".")[0]
    path_dst = os.path.join(DST_PATH, filename + "_res.png")

    s_img = Image.open(src).convert("RGB")
    s_p = s_img.load()
    width, height = s_img.size
    w_img = Image.open(watermark)
    w_p = w_img.resize((width, height)).load()
    d_img = Image.new("RGB", (width, height))
    d_p = d_img.load()

    for x in range(width):
        for y in range(height):
            (rw, gw, bw) = w_p[x, y]
            (rs, gs, bs) = s_p[x, y]
            rd = enco(rs, rw)
            gd = enco(gs, gw)
            bd = enco(bs, bw)
            d_p[x, y] = (rd, gd, bd)

    d_img.save(path_dst)

    s_img.close()
    w_img.close()
```

通过 PIL 的 Image 类，可以用 open 函数打开源图片，或用 new 函数创建目标图片。首先对水印图片进行 resize，使之与目标图片像素匹配。逐一像素进行 encode，将得到的结果存入新的图片中。

2. 提取水印

针对单个像素字节的处理——利用位运算

```
def deco(s):
    # X & 0000 0011 可以得到X的最低2位
    return (s & 3) * 85
```

提取水印的函数：

```
def read_watermark(src):
    filename = src.split("\\")[-1].split(".")[0]
    path_dst = os.path.join(DST_PATH, filename + "_extract.png")

    s_img = Image.open(src)
    s_p = s_img.load()
    w, h = s_img.size

    d_img = Image.new("RGB", (w, h))
    d_p = d_img.load()
    for x in range(w):
        for y in range(h):
            (rs, gs, bs) = s_p[x, y]
            rd = deco(rs)
            gd = deco(gs)
            bd = deco(bs)
            d_p[x, y] = (rd, gd, bd)

    d_img.save(path_dst)

    s_img.close()
    d_img.close()
```

原理同上，略。

3. 网页端实现技术部分说明

- Flask 框架说明

  路由设置：

```python
@app.route('/')
def homepage():...

@app.route('/convert/')
def convert():...

src_filename = 'kanata.jpg'
wm_filename = 'watermark.jpg'
wmget_filename = 'watermark.jpg'

@app.route('/watermark/', methods=['POST', 'GET'])
def watermark():...

@app.route('/extractor/', methods=['POST', 'GET'])
def extract():...
```

注意 watermark 和 extract 路由设置了允许 POST 模式访问，即可以进行文件上传下载等。

在页面里通过 JavaScript 脚本设置了实时读取文件路径并显示图片的脚本函数：

```javascript
function fileChange(){
    file.addEventListener("change",function()
    {
        var files = file.files[0];
        var reader = new FileReader();
        reader.onload = function(e){
            e = e || event;
            sourceImg.src = e.target.result;
        };
        reader.readAsDataURL(files);
    },false);
```

- 图片上传技术

需要用到 HTML 的表单（form）类。method 属性设置为 POST，即允许较大量数据（如文件）的传输。enctype 属性设置为“multipart/firm-data”，可以传输文件。前端代码如下：

```
<form action='#' method="post" enctype="multipart/form-data">
    <div class="img_box"><img src="{{url_for('static', filename='')}}{{src}}"
    <img hidden src="{{url_for('static', filename='')}}{{res}}" id="res_img">
    <div class="to">+</div>
    <div class="img_box"><img src="{{url_for('static', filename='')}}{{wm}}" i
        <div class="select_box">
        <section class="file_choose">
            <span>Please select your image:</span>
            <label for="src">+</label>
                <input style="..." type="file" id="src" name="src" />
            <br><br>Please select an image as watermark:
            <label for="wm">+</label>
                <input style="..." type="file" id="wm" name="wm" />
        </section>
        </div>
    <br>
    <div class="tips"...>
    <p></p>
    <button for="upload" id="turnTo">Add Watermark
        <input style="display: none" type="submit" id="upload"/>
    </button>
</form>
```

在 form 标签下的三个 input 内容会被记录：type 为 file 的是文件，分别用两个 id 来区分。最后一个 type 为 submit，提交，标签被隐藏在了按钮里。提交后，表单里的内容就会被传递到后台。通过 Flask 运行脚本的函数里可以读取到，并调用我们前面编写的编码或解码函数来处理：

```
@app.route('/watermark/', methods=['POST', 'GET'])
def watermark():
    canDownload = False
    global src_filename, wm_filename
    if request.method == 'POST':
        f = request.files['src']
        g = request.files['wm']
        basepath = os.path.dirname(__file__)  # 当前文件所在路径
        if f.filename == '':
            flash('Please select your image!')
            canDownload = False
        else:
            if g.filename != '':
                src_filename = f.filename
                wm_filename = g.filename
                src_upload_path = os.path.join(basepath, 'static/img/upload', src_filename)
                wm_upload_path = os.path.join(basepath, 'static/img/upload', wm_filename)
                f.save(src_upload_path)
                g.save(wm_upload_path)
                add_watermark(src_upload_path, wm_upload_path)
            else:
                src_filename = f.filename
                src_upload_path = os.path.join(basepath, 'static/img/upload', src_filename)
                f.save(src_upload_path)
                add_watermark(src_upload_path)
```

flask 库的 request 模块可以读取到 method 是 GET 还是 POST。表单内的内容是 POST 形式，所以可以调取到这些文件内容，并且保存到某个目录，然后调用 add_watermark 函数就可以实现添加水印。

另外，flask 的 flash 函数可以将信息传递到前端网页里显示。需要先设置 SECRET_KEY。配合函数 get_flashed_messages 可以读取到 flash 内容：

```
{% with messages = get_flashed_messages() %}
    {% if messages %}
        <script>
            var messages = {{ messages | safe }};
            for (var i=0; i<messages.length; i++) {
                alert(messages[i]);
            }
        </script>
    {% endif %}
{% endwith %}
```

- 一键下载技术

由于转换函数创建的图片是在 static/img/res 文件夹下，如果真的将网站部署到服务器的话，用户是无法得到结果的。因此编写了一个非常简单的一键下载，将文件以 png 模式保存的桌面。相关的 JS 代码如下：

```
function downloadPic() {
//    console.log(canDownload)
    if(canDownload == 'True')
    {
        var imgData = wmImg.src;
        var down = document.getElementById("downIMG");
        down.href = imgData;
        down.download = (new Date()).toLocaleString()+"."+(sType?sType:"jpg");
        var mouseEv = document.createEvent("MouseEvents");
        mouseEv.initMouseEvent("click",false,false,window,0,0,0,0,0,false,false,false,false,0,null);
        down.dispatchEvent(mouseEv);
    }
    else
    {
        alert('No result image!');
    }

}
```

五、 实验结果分析

实验结果请参见录屏文件。

六、 感想与反思

本次实验对于 Jinja2 语法、JavaScript 脚本编写，Flask 框架下前后端数据传输和 PIL 库的使用都有了更深入的了解，收获很大。

但是从目的出发来看，Stenography 的隐写术非常低效，且很容易受到攻击。一方面当图片大小比较大（比如 2M 左右），添加水印就会非常耗时；另一方面，这种方式得到的水印信息会有所损失，并且非常容易破解。我们在“媒体信号处理基础”这门课上学习的 DCT、DWT 等编码方式在数字水印方面应用更加广泛，且效果更好。一方面是 FFT（快速傅里叶变换）大大加速了编码的速度，另一方面保留了更多信息，且信息被隐写在每一个像素里。因为时间关系（疯狂 Prj 推

进中），这次并没有在实验里实现这种方法。下次一定。