java图片隐写_利用24位BMP图实现信息隐写（java语言）

原创

我本废柴 于 2021-02-25 08:25:40 发布
文章标签： java图片隐写

这礼拜，教我RESTful框架的老师布置了一份有趣的作业。如下图

分析**

需求极简单。老师提供了一个txt文档，一个bmp图片。需要将txt文档信息隐写在bmp图片中，并生成一张新图片。但这张图片用肉眼是看不出来与原图有什么差别的。

一开始解决这个问题确实有些困难。但看完老师提供的原理说明，思路如泉涌。这里限于篇幅，详细原理说明将会在本文最下方提供下载链接。

2.代码实现

具体操作还涉及文件上传等知识，这里只给出实现隐写的代码。

思路：

1.获取隐写文件的字节信息，并转化成二进制bit值；

2.将获取载体图片的像素值，并按R、G、B分量展开；

3.将bit值依次填入R、G、B分量的最低位。

4.保存新生成的图片。

Steganalysis类

(1)成员变量

```
//需要隐写的文件路径
private String sourceFilePath;
//载体bmp的文件路径
private String bmpFilePath;
```

(2)成员方法

(a)datasourceToBMP()方法

```
//将信息隐写在图片中，并保存为新图片
public void datasourceToBMP(String newbmpFilePath) {
BufferedImage image = this.getBufferedImage();
```

```
//当前像素点坐标
int curX,curY;
//计数器
int count = 0;
try {
int[] bitContainer = this.getBit();
//比较隐写信息和载体的大小，如果大于隐写失败
if(bitContainer.length > image.getWidth()*image.getHeight()*3) {
System.out.println("隐写信息过大");
return;
}
//先行后列
for(curY=0;curY
for(curX=0;curX
//获取像素值
int rgb = image.getRGB(curX, curY);
//当前像素值分量
int iRgb = 0;
//分解像素值
int R =(rgb & 0xff0000 ) >> 16 ;
int G= (rgb & 0xff00 ) >> 8 ;
int B= (rgb & 0xff );
//判断信息是否隐写完毕
if(count >= bitContainer.length) {
//将隐写文件的字节大小作为文件名
this.saveToBmp(image, newbmpFilePath + bitContainer.length/8 + ".bmp");
return;
}
//将隐写文件位信息放入像素分量最低位
while(true) {
if(count >= bitContainer.length) {
```

```
break;
}
switch (iRgb) {
case 0:
R = this.swapBit(R, bitContainer[count]);
break;
case 1:
G = this.swapBit(G, bitContainer[count]);
break;
case 2:
B = this.swapBit(B, bitContainer[count]);
break;
default:
break;
}
if(iRgb ==2 ) {
break;
}
count++;
iRgb++;
}
//重组rgb像素值
rgb = (R << 16) | (G << 8) | B;
//重设rgb像素值
image.setRGB(curX, curY, rgb);
}
}
} catch (IOException e) {
}
}
```

以下方法都是datasourceToBMP()方法内调用的方法。

(b)getBufferedImage()方法

```
//将图片图片转换为可操作对象
public BufferedImage getBufferedImage() {
BufferedImage image = null;
try {
image = ImageIO.read(new FileInputStream(this.bmpFilePath));
} catch (FileNotFoundException e) {
System.out.println("载体图片未找到");
} catch (IOException e) {
System.out.println("获取流失败");
}
return image;
}
```

(c)getBit()方法

```
/*获取文件信息 并将信息以二进制形式存储 */
public int[] getBit() throws IOException {
int[] bitContainer = null;
try {
FileInputStream fileInputStream = new FileInputStream(this.sourceFilePath);
//获取文件字节大小
int bytesize = fileInputStream.available();
//创建存储bit的容器
bitContainer = new int[bytesize * 8];
//计数器
int count = 0;
for(int i=0; i
//获取字节值
int curbyte = fileInputStream.read();
//从低到高依次获取bit
for(int j=0; j<8; j++) {
//获取当前bit
```

```
bitContainer[count] = (curbyte & 1);
//右移一位
curbyte >>= 1;
count++;
}
}
} catch (FileNotFoundException e) {
System.out.println("读取隐写文件失败");
}
return bitContainer;
}
```

(d)swapBit()方法

```
//bit值(0或1)放在old的最低位
private int swapBit(int old,int bit) {
old = (old & 0xFE);
old |= bit;
return old;
}
```

(e)saveToBmp()方法

```
//将图片保存为bmp格式
private void saveToBmp(BufferedImage image,String newbmpFilePath) {
Iterator writers = ImageIO.getImageWritersByFormatName("bmp");
ImageWriter writer = (ImageWriter) writers.next();
ImageOutputStream imageOutputStream = null;
try {
imageOutputStream = ImageIO.createImageOutputStream(new FileOutputStream(new File(newbmpFilePath)));
} catch (IOException ioe) {
}
writer.setOutput(imageOutputStream);
try {
writer.write(image);
```

} catch (IOException e) {

}

}

源码及相关原理、素材下载链接

用Tomcat启用服务后：

1.访问:localhost:8080/steganalysis进入文件上传界面；

2.访问:localhost:8080/steganalysis/a/readImage?fileMainName=197024查看读取隐写的信息