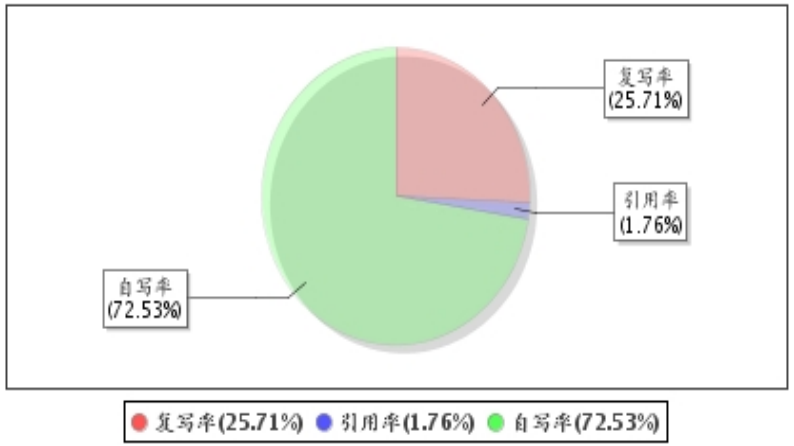
论文检测报告

报告编号： 4923BACCD2884A558802183D74B5EA64
送检文档： 课程设计报告
论文作者： 刁肥宅

文档字数： 26972
检测时间： 2019-02-11 19:25:45
检测范围： 互联网，中文期刊库（涵盖中国期刊论文网络数据库、中文科技期刊数据库、中文重要学术期刊库、中国重要社科期刊库、中国重要文科期刊库、中国中文报刊报纸数据库等），学位论文库（涵盖中国学位论文数据库、中国优秀硕博论文数据库、部分高校特色论文库、重要外文期刊数据库如Emerald、HeinOnline、JSTOR等）。

一、检测结果：

总相似比： 27.47% [即复写率与引用率之和]
检测指标： 自写率 72.53%复写率 25.71%引用率 1.76%
相 似 比： 互联网 18.23% 学术期刊 3.48% 学位论文 4%
其他指标： 表格 0 个 脚注 0 个 尾注 0 个



章节抄袭比
24.97% 课程编号：0521172B课程性质：必修

二、相似文献汇总：

序号	标题	文献来源	作者	出处	发表时间
1	基于小波变换的数字图像水印算法的研究	学位论文	王艳玲	硕博学位论文	2006
2	基于linux的停车场管理系统软件的设计	学位论文	王元伟	硕博学位论文	2010

3	bmp图片格式详解-yun_hen的博客-CSDN博客	互联网		互联网	
4	QImage和QPixmap-壁立千仞无欲则刚的博客-CSDN博客	互联网		互联网	
5	CSDN论坛	互联网		互联网	
6	图像数字水印的设计与实现_百度文库	互联网		互联网	
7	C++编写的读取一幅位图的类,在输入位图路径时有有关问题,如果在...	互联网		互联网	
8	基于图像特征和霍夫曼编码的图像水印算法IMAGE	互联网		互联网	
9	基于数字水印的多媒体教学资源保护	学术期刊	王冰	海南广播电视大学学报	2012
10	用VC编程实现BMP图像裁切	学术期刊	钱方明	电脑编程技巧与维护	2008
11	76数字水印的嵌入和提取数字水印的实现可以分为空间域数字水印和...	互联网		互联网	
12	图像处理的研究方面,目前有哪些突破性的进展,有哪些新方法?_已...	互联网		互联网	
13	开关位置图像识别及其在电力系统中的应用	学位论文	范杰清	硕博学位论文	2005
14	自然场景下基于边界先验的图像显著性检测-豆丁网	互联网		互联网	
15	Windowsc++界面开发初学(一)-unirrrrr的博客-CSDN博客	互联网		互联网	
16	激光条纹图像处理算法的研究	学术期刊	张德保 刘普吉 姜寿山	微型机与应用	2011
17	OpenCV&Qt学习之一——打开图片文件并显示-emouse-博客园	互联网		互联网	

18	QT编写TCP入门+简单的实际项目(附源程序)-寂寞的小乞丐-博客园	互联网		互联网	
19	C++Qt笔记004:从helloworld说起	互联网		互联网	
20	大地win8系统安装盘_大地GHOST_WIN8_64位旗舰版v2014.06-系统城	互联网		互联网	
21	百度快照	互联网		互联网	
22	linux系统下的QT应用程序开发	学术期刊	兰佳卉 余鑫	中国科技博览	2012

三、全文相似详情：

（红色字体为相似片段、浅蓝色字体为引用片段、深蓝色字体为可能遗漏的但被系统识别到与参考文献列表对应的引用片段、黑色字体为自写片段）

课程编号：0521172B课程性质：必修
数据结构课程设计报告
2019年1月20日 至 2019年2月11日
目录

一 实验概述：

1.1课程设计题目：

题目要求对给定的一种简单的二值图像的数字水印算法编程实现。

1.2课程设计目的：

对数字水印技术建立一定的认识，能建立位矩阵、位向量等ADT，并能用这些ADT 给定二值图像数字水印的嵌入和抽取。

1.3系统主要内容与功能：

1.3.1 设计内容：

具体包括以下内容：

1）图像的读取与保存，及相应的矩阵和向量的运算；

2）二值图像水印算法的实现；

3）软件的界面和接口设计，信号发送和槽位的设计；
4）软件鲁棒性分析、算法鲁棒性分析和相关总结。
1.3.2设计功能：
1）设计合理的数据结构，编程实现算法；
2）给定测试图片，按照指定的bmp格式，保存于外存中。
1.3.3实验环境与工具：
1）操作系统：Windows企业版2016长期服务版；
2）开发工具：Qt、Qt Creator与Windows图形API；
3）实现语言：C++。
二 实验原理：
2.1 图像水印技术简述：
随着互联网和信息技术的快速发展，近年来数字内容的未授权获取，传输，操纵和分发的问题变得越来越严重。信息安全研究引起了人们的广泛关注。除了一般采用的数字加密算法之外，近年来用于信息安全的影像视觉算法包括光学图像加密、认证和水印算法被广泛地研究和应用起来。影响视觉信息安全算法通常拥有并行告诉处理和多维能力的优势。信息隐藏技术，即图像水印技术，也称隐写技术是一种隐蔽性地改变载波信号以嵌入隐藏消息，即水印信号，的技术。可以对各种各样类别的信号执行信息隐藏，其中包括但不限于：音频信号、图像信号和视频信号。信息隐藏技术（图像水印技术）允许将特定的信息加入到需要保护的媒体信息中，加入的信息一般为具有特定意义的内容，如版权所有者信息、发行标志、特定代码等。
而图像水印技术也因而成为了数字图像处理专业当下或未来重要的研究领域，在知识产权的保护等方面有着广发的应用前景。为了确保大规模在线分发的多媒体内容的版权和知识产权，我们需要通过有效的保护来控制分发和传播，控制来自盗版用户或未经授权普通用户的恶意操纵和恶意拷贝传播。
为了提高效率，水印需要良好的隐蔽性，并且拥有嵌入高容量和有效载荷，能够在确保有效载荷的安全传输的同时，对最常见的图像处理（恶意或非恶意）进行鲁棒性处理。此外水印技术还有以下的特点：
（1）水印后的主图像不应存在显著地信息损坏和分辨率降低；
（2）水印应具有在主图像中良好的隐蔽性；

（3）水印应具有良好的鲁棒性，不易从主图像中被损坏。水印应可以承受不同类型的信息损坏打击，例如JPEG压缩、裁剪、旋转、缩放、噪声、滤波运算和模糊运算。应该特别指出，该特点仅适用于一部分鲁棒性极好的水印算法中；

（4）未经授权的用户/盗版用户应很难非法访问到该水印信息。

2.2 图像处理技术简述：

在图像水印技术实现时，需要大量运用到图像处理知识和技术。数字图像处理(Digital Image Processing)是通过计算机对图像进行去除噪声、增强、复原、分割、提取特征等处理的方法和技术。数字图像处理（Digital Image Processing）又称为计算机图像处理，它是指将图像信号转换成数字信号并利用计算机对其进行处理的过程。数字图像处理的产生和迅速发展主要受三个因素的影响：一是计算机的发展；二是数学的发展（特别是离散数学理论的创立和完善）;三是广泛的农牧业、林业、环境、军事、工业和医学等方面的应用需求的增长。

在图像水印技术中，本着不损坏原有图像质量的原则，该算法需要对图像快速的读写能力、对分块图像稳定和鲁棒的运算能力，以及优秀的边缘提取能力。

2.3 Qt及Qt Creator：

Qt是一个跨平台的C++图形用户界面应用程序框架。它为应用程序开发者提供建立艺术级图形用户界面所需的所有功能。它是完全面向对象的，很容易扩展，并且允许真正的组件编程。

作为一个优秀的C++框架，由于其优秀的信号与槽机制，Qt被广泛地应用于软件编写中。本次课程设计将采用Qt和Qt Creator作为程序界面的编写工具。

2.4 相关算法简介：

数字图像水印算法是将一段信息附加在图像上的算法，其中被附加信息的图像被称为主机图像(host image)，简称主图像。根据水印算法的鲁棒性，水印算法可以分为鲁棒/稳健的水印算法(robust watermarking)和脆弱的水印算法(fragile watermarking)，其中绝大多数的数字水印算法都属于前者。如上文所提到，鲁棒的水印在主图像受到攻击和扭曲时，仍能保持完整。由于鲁棒的水印很难从主机图像中移除，因此常常用于保护版权；另一方面，脆弱的水印一般仅用于验证，即主机图像的完整性检查。完整的脆弱水印表示主机图像处于其原始形式，并没有收到编辑、损坏或更改。在本次课程设计中，为了简洁起见，我们采用脆弱的水印算法。

经典的数字图像水印算法构建的系统包括双随机相位编码（DRPE）系统，离轴全息系统，相移全息系统，优化的仅相位掩模结构，联合变换相关器（JTC），重影成像系统和ptychography系统，等等。它们在各自的领域都具有相当不错的效果。在本次课程设计中，为了兼顾效率和效果，我们采用一种简单的二值图像数字水印算法，其基本思路如下：

由于水印算法需要改变主机图像的像素值，从而一定程度上改变主机图像的信息。而被改变像素值，因而保存着水印信息的像素点被称为隐藏点，它决定了隐藏了怎

样的信息。而简单的处理方法容易导致图像质量的下降，例如，在全白的图像块中插入一个黑色的噪声点。另一方面，隐藏点也应避免选取在图像中的细线区域、直线边的中间像素、孤立像素等图像信息熵较大的点。由此，二值图像的数字水印嵌入算法的关键是隐藏点的选取，以下是隐藏点的选取规则：

二值图像数据应隐藏域图像中黑/白色区域的边界上，但上述诸如细线区域等仍不适于隐藏数据的边界。因此隐藏点应具有以下的特点：它是边界像素，并且不同时是左边界和右边界/上边界和下边界，以确保避免将信息隐藏在细线区域等。这需要一个优秀的边界提取算法。

在隐藏点被确定后，水印信息应转换为二进制序列，并保存在隐藏点中。同理，因此所有可被转换为二进制序列的诸如文字信号、音频信号、图像信号和视频信号，并可进行反转换的信号都可以作为该算法的水印信息。在本次课程设计中，我们简单地以字符串信息和二进制序列信息为例，来保存相关的水印。

值得一提的是，为了保证水印信息更好地隐蔽，水印信息被加入前，需要用一段密钥进行加密，并在提取时，使用该密钥进行解密。

2.5 技术流程：

本次课程设计采用的主要框架包括C++、Qt、Windows图形API等，其中C++作为程序实现语言，Qt作为软件实现框架，使用C++中的指针工具和 ” window.h “ 库作为图像处理工具。

在该环境下，常用的图像解码工具包括C++语句借助自带的 ” windows.h ” 库、Qt的QPixMap模块以及OpenCV库。其中使用C++进行bmp文件解码过程较为复杂，需要鲁棒的文件读写设计以及接口设计，同时，对于不同类型的文件需要不同的解码方式，不广泛适用于.png，.img，.jpg，.giff等等图像格式的使用，但减少了第三方库的利用。而QPixMap框架的引入对于文件数据对象的空间开辟和释放过于频繁，降低了程序的效率，并需要引入标准模板库框架。作为对比，OpenCV库拥有极高的鲁棒性、延展性和高效性，但需要引入第三方库，软件打包较为复杂。在 “ 尽量引入较少的第三方库 ” 的原则下，我们采用了纯C++语言，借助标准库自带的windows.h进行编写，以提高本软件的实用性。

本次课程设计的技术流程如下：首先设计并实现边缘信息提取的算法edgeExtract() 函数，以确定隐藏点，再设计数字隐藏的算法encodeImage() 函数。在软件流程中，首先点击 “ 浏览 ” 按钮，选取二值图像，再输入需要加入的水印，最后点击 “ 编码 ” 按钮，则生成并显示了附有水印信息的图像。该图像被保存在.pro的Qt项目文件同名文件夹下，被命名为encode.bmp。此外，对于已编码的encode.bmp，点击 “ 解码 ” 按钮，会以对话框形式弹出被解码出的水印信息。

三 实验结果：

本次课程设计生成的软件结果如下：

软件打开后的界面如下，为避免用户每次打开软件后都需要输入水印，过于麻烦，本软件内置了水印二进制序列，如图1所示。

图1 软件开启界面

输入二值图像后，会被展示在软件中，如图2所示。
图2 显示所输入二值图像
点击“ 编码 ”按钮后，生成编码图像，如图3所示。
图3 生成编码图像
点击“ 解码 ”按钮，弹出水印信息，如图4所示。
图4 显示水印信息
本程序也支持字符串的水印编码，将水印附带的comboBox选取为“ QString ”，然后编码并解码后的结果，如图5所示。
图 5 字符串水印编码图像
四 算法分析：
本程序经过第三方测试发现并无bug存在，鲁棒性优秀，可以完整的实现简单的二进制序列的二值数字图像水印算法功能。
在第二章节中提到，优秀的水印算法拥有上述的四个特点：a.不改变原图信息；b.隐蔽性；c.鲁棒性；d.不可访问性。本报告将从这四个角度评判该算法的性能。由第三章节的内容易见，该算法具有从肉眼中几乎无法分辨与原图像的区别，因而完美地符合了不改变原图信息和隐蔽性的原则。该算法经过轻微地噪声扰动后就无法保存水印信息，因此鲁棒性极低，然而这符合脆弱的水印算法的特点，可以用于验证图像的完整性。从encode.bmp和原图之间的差分影像可以轻易地获得加密后的二进制序列。但是由于有密钥的存在，无密钥的非授权用户和盗版用户无法访问到原始水印的信息，具有不可访问性。关于不可访问性，在附录中有所展示。综上所述，本算法在效率极高的同时，完美地满足了优秀的脆弱水印算法的四个特点，同时保证了效果和效率。
五 总结：
在本次课程设计之后，我对数字水印技术建立一定的认识，能建立位矩阵、位向量等ADT，并能用这些ADT 完成上述二值图像数字水印的嵌入和抽取，完成了该次课程设计的各个要求。
附录（源代码与其他）：
源代码：
头文件additional_utility.h：

```
#ifndef ADDITIONAL_UTILITY_H
```

```
#define ADDITIONAL_UTILITY_H
#include 〈windows.h〉
#include 〈iostream〉
#include 〈fstream〉
#include 〈QtCore〉
#include 〈QMessageBox〉
using byteArray = QVector〈bool〉;
#endif // ADDITIONAL_UTILITY_H
```

头文件bmputil.h：

```
#ifndef BMPUTIL_H
#define BMPUTIL_H
#include "additional_utility.h"
//#include 〈windows.h〉
class watermark
{
public:
QString array2byte(byteArray &array);
QString array2str(byteArray &array);
byteArray byte2Array(QString &number);
byteArray decodeImg(uchar* buffer, uchar* dst, const int width, const int height, const int length);
uchar* edgeExtract(uchar* buffer, const int width, const int height);
byteArray encode(byteArray src, byteArray key);
```

```
byteArray generateKey(const int length);
byteArray img2Array(QString &dir);
uchar* readBmp(const char *bmpName, int& bmpWidth, int& bmpHeight);
bool savebmp(const char* filename, uchar* buffer, const int height, const int width);
byteArray str2Array(QString &str);
uchar* substract(uchar* buffer1, uchar* buffer2, const int size);
uchar* translation(uchar* buffer, const int width, const int height, int x_off, int y_off);
uchar* watermarkImg(uchar* buffer, uchar* edge, const int size, byteArray code);
private:
BITMAPINFOHEADER BMIH;
BITMAPFILEHEADER BMFH;
int biWidth;
int biHeight;
int biBitCount;
unsigned char *pBmpBuf;
int lineByte;
RGBQUAD *pColorTable;
protected:
};
#endif // BMPUTIL_H
```

头文件mainwindow.h：

```
#ifndef MAINWINDOW_H
```

```
#define MAINWINDOW_H
#include 〈QMainWindow〉
#include 〈QPixmap〉
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
Q_OBJECT
explicit MainWindow(QWidget *parent = 0);
~MainWindow() ;
private slots:
void on_pushButtonBrowse_clicked() ;
void on_lineEdit_textChanged(const QString &arg1);
void on_pushButtonEncode_clicked() ;
void on_pushButtonDecode_clicked() ;
Ui::MainWindow *ui;
QPixmap image;
byteArray key;
uchar* dst;
#endif // MAINWINDOW_H
```

源文件additional_utility.cpp：

源文件bmputil.cpp：

```
#include "bmputil.h"
uchar* watermark::readBmp(const char *bmpName, int& bmpWidth, int& bmpHeight)
FILE *fp = fopen(bmpName, "rb");
if(fp == Q_NULLPTR)
QMessageBox::warning(Q_NULLPTR, "Error", "Error in Open File!");
return Q_NULLPTR;
// skip the fileheader
fseek(fp, sizeof(BITMAPFILEHEADER), 0);
// read the infoheader
BITMAPINFOHEADER* head = new BITMAPINFOHEADER;
fread(head, sizeof(BITMAPINFOHEADER), 1, fp);
bmpWidth = head-〉biWidth;
bmpHeight = head-〉biHeight;
int bitCount = head-〉biBitCount;
if(bitCount == 8)
//if(bitCount)
pColorTable = new RGBQUAD[256];
fread(pColorTable, sizeof(RGBQUAD), 256, fp);
uchar* pBmpBuf = new uchar[ bmpWidth * bmpHeight ];
fread(pBmpBuf, sizeof(uchar), bmpWidth * bmpHeight, fp);
fclose(fp);
uchar* buffer = new uchar[bmpWidth * bmpHeight];
```

```
for(int i = 0; i ＜ bmpHeight; i++)
for(int j = 0; j ＜bmpWidth; j++)
if(pBmpBuf[(bmpHeight- i - 1)*bmpWidth + j] != 255 && pBmpBuf[(bmpHeight- i - 1)*bmpWidth + j] != 0)
QMessageBox::warning(Q_NULLPTR, "Error", "This is not a binary image!");
buffer[i*bmpWidth + j] = pBmpBuf[(bmpHeight- i - 1)*bmpWidth + j];
return buffer;
else
QMessageBox::warning(Q_NULLPTR, "Error", "Our program can only deal with 8-bit image!");
bool watermark::savebmp(const char* filename, uchar* buffer, const int height, const int width)
if(buffer == Q_NULLPTR)
QMessageBox::warning(Q_NULLPTR, "Error", "The Buffer is nullptr!");
return false;
uchar* data = new uchar[height*width];
for(int i = 0; i ＜ height; i++)
for(int j = 0; j ＜width; j++)
data[i*width + j] = buffer[(height- i - 1)*width + j];
int colorTableSize = 1024;
BITMAPFILEHEADER fileHeader;
fileHeader.bfType = 0x4D42;
fileHeader.bfReserved1 = 0;
fileHeader.bfReserved2 = 0;
fileHeader.bfSize = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + colorTableSize + height*width;
```

```
fileHeader.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + colorTableSize;
BITMAPINFOHEADER bitmapHeader = { 0 };
bitmapHeader.biSize = sizeof(BITMAPINFOHEADER);
bitmapHeader.biHeight = height;
bitmapHeader.biWidth = width;
bitmapHeader.biPlanes = 1;
bitmapHeader.biBitCount = 8;
bitmapHeader.biSizeImage = height*width;
bitmapHeader.biCompression = 0;
FILE *fp = fopen(filename, "wb");
QMessageBox::warning(Q_NULLPTR, "Error", "Error in Save File!");
fwrite(&fileHeader, sizeof(BITMAPFILEHEADER), 1, fp);
fwrite(&bitmapHeader, sizeof(BITMAPINFOHEADER), 1, fp);
fwrite(pColorTable, sizeof(RGBQUAD), 256, fp);
fwrite(data, height*width, 1, fp);
return true;
// generate keyArray
byteArray watermark::generateKey(const int length)
byteArray res;
```

四、指标说明：

1. 总相似比即类似于重合率。总相似比即送检论文中与检测范围所有文献相似的部分（包括参考引用部分）占整个送检论文的比重，总相似比=复写率+引用率。
2. 引用率即送检论文中被系统识别为引用的部分占整个送检论文的比重（引用部分一般指正确标示引用的部分）。
3. 自写率即送检论文中剔除雷同片段和引用片段后占整个送检论文的比重，一般可用于论文的原创性和新颖性评价，自写率=1-复写率-引用率。
4. 复写率即送检论文中与检测范围所有文献相似的部分（不包括参考引用部分）占整个送检论文的比重。
5. 红色字体代表相似片段；浅蓝色字体代表引用片段、深蓝色字体代表可能遗漏的但被系统识别到与参考文献列表对应的引用片段；黑色字体代表自写片段。

五、免责声明：