第 18 卷 第 1 期
2013 年 1 月

中国图象图形学报
Journal of Image and Graphics

Vol. 18, No. 1
Jan., 2013

中图法分类号: TP309.3 文献标识码: A 文章编号: 1006-8961(2013)01-0024-12
论文引用格式:徐明,黄立,张海平,徐建,郑宁. 头部缺失的 JPEG 文件碎片恢复[J]. 中国图象图形学报,2013,18(1):24-35.

头部缺失的 JPEG 文件碎片恢复

徐明, 黄立, 张海平, 徐建, 郑宁
杭州电子科技大学计算机学院, 杭州 310018

摘 要: JPEG 碎片恢复和修复一直是数字取证领域中的一个难点和热点问题,特别是在遇到 JPEG 文件的头部缺失或者损坏而导致其数据无法正常解码和显示时。提出一种头部缺失且无重启标记的 JPEG 文件碎片解码与显示问题的解决方法。首先提出了基于 JPEG 碎片数据的各种解码参数(哈夫曼表、图像宽度、量化表和采样因子等)的估算方法;其次对显示时出现的位置错位和颜色偏移问题提出了相应的调整算法。实验结果表明,本文方法能够正确有效地解码并显示没有头部与重启标记的 JPEG 文件碎片。

关键词: 数据恢复; JPEG 碎片; 重启标记; 重构; 解码

Recovery method for JPEG file fragments with missing headers

Xu Ming, Huang Li, Zhang Haiping, Xu Jian, Zheng Ning
College of Computer, Hangzhou Dianzi University, Hangzhou 310018, *China*

Abstract: Recovering the incomplete fragments of a JPEG file is a challenge in the digital forensics field, especially in the case of where JPEG header is lost or damaged. In this paper, we present a method to decode and display JPEG file fragments whose header has been lost or damaged and its data doesn't include any restart marker. First, the decoding parameters (Huffman tables, image width, quantization tables, sampling factor) are estimated based on the fragment content; second, corresponding adjustment algorithms for the display problems about mismatched positions and colors are proposed. Experiments show that the proposed method can correctly decode and display the JPEG file fragments without header and RST marker.

Key words: data recovery; JPEG fragments; restart marker; reconstruction; decoding

0 引 言

数字图像以其能够直观反映大量的信息而备受计算机取证者的"青睐",如反映犯罪嫌疑人的面貌、非法照片等。因此它经常作为证据材料出现在法庭之上,尤其是在日益增长的涉及计算机与网络犯罪的案例(如儿童色情犯罪[1-2])当中。但是在取证的过程中经常会遇到恢复与雕复出看似无用的图像文件碎片的难题。在文献[3]中举了这样一个实例,犯罪嫌疑人被怀疑传播儿童色情照片,取证者控制其电脑准备搜集证据,但是发现里面的数据已经被删除。随后计算机取证者利用文件雕复等技术获取了大量的图像文件碎片,遗憾的是大部分恢复出来的碎片却并不能正常解码和显示出来。究其原因是因为恢复出来的 JPEG 文件大部分已经损坏。

JPEG 文件被广泛应用的同时,也自然而然地成为了数字取证中重点调查的对象。但是磁盘上的 JPEG 文件数据不一定连续存储。Garfikel 等人调查了 350 个硬盘后发现至少有 16% 的 JPEG 文件在磁

收稿日期:2012-02-16; 修回日期:2012-06-28
基金项目:国家自然科学基金项目(61070212,61003195);浙江省自然科学基金项目(Y1090114, LY12F02006)
第一作者简介:徐明(1970—),男,教授,2004 年于浙江大学获计算机科学与技术专业工学博士学位,主要研究方向为网络信息安全和数字取证等。E-mail: mxu@hdu.edu.cn

盘上发生了分片[4]。此外,由于已删除的 JPEG 文件数据可能会被其他文件数据覆盖或是被恶意的损坏等原因,导致在 JPEG 文件恢复和雕复的过程中经常会遇到没有头部的 JPEG 文件碎片的情形[5]。因此,研究如何让头部缺失的 JPEG 文件碎片能够正确解码与显示的问题是非常有价值的。

1　相关研究

文件雕复技术作为一种不依赖于文件系统元信息的数据恢复技术,是计算机取证过程中的一种重要技术手段。而针对 JPEG 文件的雕复技术研究是其中的一个热点和难点问题。JPEG 文件雕复技术研究大致经历了完整 JPEG 文件雕复[6-7]、分片 JPEG 文件雕复[4,8-10]、残缺 JPEG 文件雕复[11-13] 3 个阶段。而针对残缺 JPEG 文件雕复技术研究目前还处于起步阶段,尤其是对于头部缺失的 JPEG 文件。解码与显示 JPEG 文件所需的信息大都存储在头部字段当中,当头部损坏或是缺失则很难将数据正确解码并显示出来。

对于头部缺失的 JPEG 文件恢复,首先是要估算其解码所需要的参数信息。文献[12]提出一种通过构造伪头部来正确解码 JPEG 数据的方法。其认为解码一幅 JPEG 图像所需的信息包括 4 个主要组成部分: 哈夫曼表、图像的宽度、最小编码单元(MCU)和量化表。但该文献并没有详细介绍怎样去重构这些解码信息。Karresand 等人[11]提出一种通过编码的数据来估计图像宽度的方法。但前提条件是编码的数据中含有重启标记(restart Marker)并保证图像中至少存在一条垂直竖线才能成功。文献[13]在 Karresand 基础上,提出一种通过计算邻近量化直流系数(DC)之间相似性来估计图像宽度的方法。其算法能较好地估算图像的宽度,但是缺点同样需要重启标记来进行同步。

由于重启标记具有很好的解码定位作用,可以用来对 JPEG 编码数据进行正确解码。但是并不是所有 JPEG 文件都具有重启标记,因为重启标记并不是 JFIF 格式标准必要的组成部分。所以针对已有的 JPEG 碎片恢复研究都假设碎片数据中包含重启标记的前提,提出一种无重启标记的 JPEG 碎片数据的解码和显示方法,并对碎片数据解码参数如量化表等也提出相应新的估计算法。

2　Huffman 编码的自同步能力分析

在头部缺失的 JPEG 文件恢复过程中,先前的研究都假设数据碎片中包含重启标记(RST)用来进行解码同步。通过分析 Huffman 编码数据的特点,发现编码的数据具有很强的自同步能力,利用其自动同步特性可以实现在没有重启标记时仍可正确解码的功能。

2.1　Huffman 编码算法及其范式编码

Huffman 编码是一种最优的不等长前缀编码技术[14],在数据压缩中得到广泛的应用,如 GZIP 和 JPEG。Huffman 编码首先对所有待编码符号扫描一遍,统计各个符号出现的概率,然后按出现概率的大小指定不同长度的码字,最后建立一棵与之对应的哈夫曼树。Huffman 编码大致可以分为以下 5 个主要步骤[14]:

1) 扫描数据,统计符号出现的概率;

2) 建立一个以二叉树构成的森林,每个叶子节点保存一个符号及其概率;

3) 对符号出现概率按从小到大排序;

4) 将排名前二的二叉树合并成新的二叉树,且置新的二叉树的根节点的权值为左右子树根节点的权值之和;

5) 合并完成后再重新对所有根节点进行排序,重复步骤 4),直到合并为一棵二叉树;

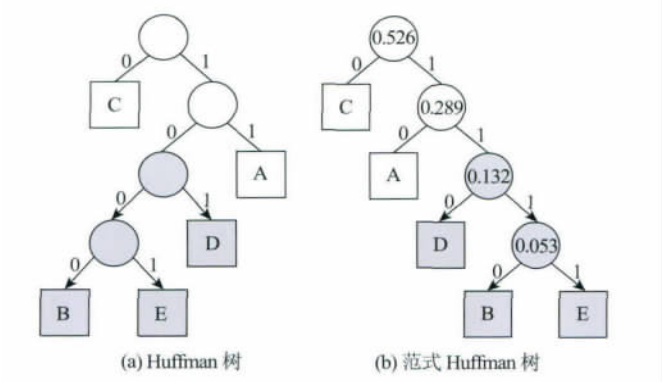
表 1 所示为一份统计的英文字符频度表,对其进行 Huffman 编码得到的哈夫曼树如图 1(a)所示。

表 1　字符符号频度

Table 1　The character symbol frequency

	A	B	C	D	E
频度	6	1	9	3	1

Huffman 编码是完全按照字符出现的概率来构造字符平均长度最短的编码,因此又叫最优的前缀编码。但是它的缺点是建立哈夫曼表必须扫描数据两次,且码树占用的空间较大,编码具有不唯一性。因此,Schwartz 在 1964 年提出范式 Huffman 编码的概念,它是 Huffman 编码的一个子集,特点是可以根据编码位长来推算出码字。其中心思想是采用一些强制的约定去重构哈夫曼树的结构。范式编码具有 3 个重要的约定: 1) 码字长度最小的第 1 个编码从 0 开始; 2) 相同码字长度具有数字序列属性,即要求



(a) Huffman 树 (b) 范式 Huffman 树

图 1 哈夫曼树和范式树

Fig. 1 Huffman Tree and Canonical Tree

相同长度的码字是连续整数的二进制描述;3) 长度为 i 的第 1 个码字 $f(i)$ 能够根据公式 $f(i)=2f(i-1)+1$ 从长度为 $i-1$ 的最后一个码字求出。范式哈夫曼算法采用一个巧妙的方法对哈夫曼树进行规范调整。首先,对于同一层的节点,所有的叶子节点都调整到左边。然后,对于同一层的叶子节点按照符号顺序从小到大调整。最后,进行左 0 右 1 分配编码。经过调整之后的范式树如上图 1(b) 所示。

对表 1 分别进行 Huffman 编码和范式编码得到的码字如表 2 所示。

表 2 编码表

Table 2 The coding table

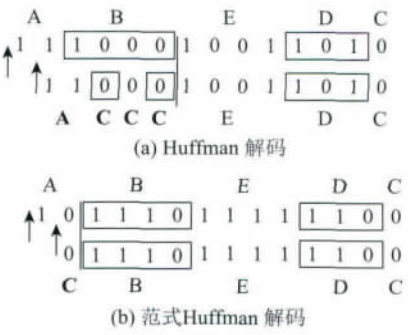
符号	编码长度	Huffman 编码	范式 Huffman 编码	概率
C	1	0	0	0.45
A	2	11	10	0.30
D	3	101	110	0.15
B	4	1000	1110	0.05
E	4	1001	1111	0.05

范式 Huffman 编码的优点是不用保存整个哈夫曼树,只需要存储编码的长度就能够重构出整个树的结构。JPEG 图像中就采用范式 Huffman 编码表进行快速解码。

2.2 Huffman 编码的自同步属性

Huffman 编码采用变长编码,因此解码时需要确定解码的起始位置,否则就会出现解码错误。哈夫曼树是完全前缀编码树[14],即任意的比特序列都能从中间某点解码出来。

对于表 2 中的码表,字符序列{A,B,E,D,C} 分别进行 Huffman 编码和范式编码得到的二进制序列为{11100010011010} 和{10111011111100}。如图 2 所示,在解码的过程中如果起始位置不是从二进制序列的第 1 个比特而是第 2 个比特时,解码的符号分别为{A,C,C,C,E,D,C} 和{C,B,E,D,C}(加粗表示解码的符号并非原始符号)。从图 2 可以发现当错误解码一段比特流之后,解码能够重新回到某个码字的边界,使得后面的比特流能够正确解码,这即是哈夫曼码流的自同步属性[15],如图 3(a) 所示下图 3(b) 显示的是错误解码并达到同步的 Lena 图像。范式编码的自同步能力一般要强于没有调整过的 Huffman 编码。从范式码流的任何非码字边界的位置开始解码,最多不超过 1 个码字的错误解码就能同步,如图 2(b) 所示。这也是 JPEG 采用范式 Huffman 编码的原因之一。



(a) Huffman 解码

(b) 范式Huffman 解码

图 2 哈夫曼码流的不正确解码

Fig. 2 The inaccuracy decoding of encoded bits stream

(a) 自同步哈夫曼解码 (b) Lena 图像自同步哈夫曼解码

图 3 哈夫曼解码的自同步能力

Fig. 3 Syschronization of Huffman decoding

2.3 估计同步点的位置

为了估计需要多少比特的错误解码能够达到同步,文献[16] 从数学上分析了经过特定的 Huffman 编码的数据在出现解码错误时达到同步平均需要经过的比特数。为了能够适应于更加复杂的 JPEG 数据流,本节延伸其方法。以图 1 的哈夫曼树为例,首先定义如下的符号。用 T,I,L 分别表示整个哈夫曼树、中间节点集合和叶子节点集合。$p_1,\cdots,p_n$ 表示对应叶子节点即编码符号的统计概率。用 $l_1,\cdots,l_n$ 分别表示对应符号的编码长度。$\forall i\in I$,用 L_i 表

示以中间节点 i 为根节点的子树的叶子节点。中间节点 $\boldsymbol{I}$ 表示码字可能被切断的位置。特别地,根节点表示位于两个码字之间的位置。

一段经 Huffman 编码的二进制数据流,随机地从其中某个位置开始解码,现在需要估计平均可能需要经过多少比特数据的解码才能达到同步。对于一棵确定的哈夫曼树,任意长度的码字在确定的区间上具有相同的概率,因此可以用统计的方法大致估计同步点的位置。由于切断点的位置随机,切断的位置可能是在码字的边界也可能在码字的中间。首先第一步计算从不是码字的边界位置开始解码的概率 $P(i)$,也就是从图 1(a) 中取某个中间节点的概率。对于一个给定的中间节点 $i \in \boldsymbol{I}$,概率 $P(i)$ 表示中间节点 i 被作为解码起始点的位置。对于确定的哈夫曼树,其平均码字长度

$$W = \sum_{x=1}^{n} p_x l_x \tag{1}$$

定义 $\boldsymbol{T}_i$ 表示以中间节点 i 为根节点的子树。$\boldsymbol{L}_i = \boldsymbol{L} \cap \boldsymbol{T}_i$ 表示子树的叶子节点。对于某个中间节点,如图 1(a) 中标记的中间节点,以其作为根节点构成的子树如标记所示。以其作为起始位置有 3 条分支可以到达的叶子节点 B,D,E。则概率 $P(i)$ 则可认为是三者的权重概率之和除以平均码字长度,即

$$P(i) = \left(\sum_{y \in I_x} p_y\right) / W \tag{2}$$

对于经过 Huffman 编码的比特流,如果从某个码字 B 的中间比特 i 开始解码,用符号 E 表示同步点在解码到码字 B 的边界时就已经达到同步的事件。则 $P(E)$ 根据全概率公式可以表示为

$$P(E) = \sum_{i \in I} P(E \mid i) P(i) \tag{3}$$

而 $P(E \mid i)$ 的概率则可表示为

$$P(E \mid i) = \frac{\sum_{j \in L_i} p_j Q(i,j)}{\sum_{j \in L_i} p_j} \tag{4}$$

其中对于任意 $i \in \boldsymbol{I}$ 和 $j \in \boldsymbol{L}$,定义权重因子

$$Q(i,j) = \begin{cases} 1 & \text{从 } i \text{ 到 } j \text{ 的路径上有一个或者多个码字} \\ 0 & \text{其他} \end{cases}$$

则最终得到

$$P(E) = \left[\sum_{i \in I} \sum_{j \in L_i} P_j Q(i,j)\right] \Big/ W \tag{5}$$

从推导出来的式(5) 可以看出概率 $P(E)$ 只与给定的分布和哈夫曼树的形状有关。权重 $Q(i,j)$ 越大和平均码字长度越小,得到的概率 $P(E)$ 越大。从图 1 的哈夫曼树和范式哈夫曼树的比较可知,范式树更能在某中间节点更快地达到同步。如对于经过图 1(a) 码字 B(1000),从第一个 0 开始解码,查找表 2 需经过 3 个 0 后才能到达码字的边界,相应的权重 $Q(i,j) = 1$。而对于图 1(b) 中码字 B(1110),从任何中间位置,都只需一个码字的错误解码便能在码字的边界达到同步。

互补事件 $\overline{E}$ 表示解码点在某个码字的中间,但是没有在码字的边界达到同步的事件。可以将经过多少错误解码能够达到同步的数学模型看成是一个几何分布,则达到同步的期望值为 $1/P(E)$,平均的码字长度为

$$E = W/P(E) \tag{6}$$

对图 1 中的哈夫曼树和范式树通过式(6) 估计的码字分别约为 3.00 和 2.06。而平均码字根据式(1) 约为 1.9,即对于图 1(a) 中的 Huffman 树需要 1.5 个码字才能达到解码同步,而图 1(b) 的范式树大约只需要一个码字就能同步,从理论上分析了 Huffman 解码的自同步能力。

3　JPEG 碎片恢复方法

3.1　估计解码参数

根据 JPEG 标准,JPEG 解码流程大致如图 4 所示。

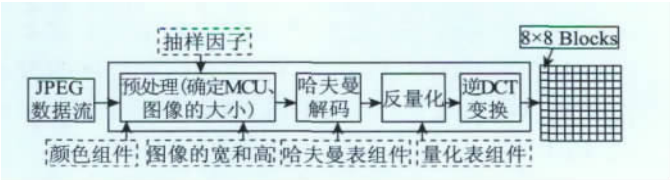


图 4　JPEG 解码流程框图

Fig. 4　The approximately flow chart of the JPEG decoding

JPEG 文件解码需要经过以下 4 个关键步骤: 1) 预处理,确定解码所需最小编码单元(MCU) 以及图像的宽度和高度; 2) 哈夫曼解码,读取哈夫曼表,将码流解码为量化 DCT 系数; 3) 反量化,量化的 DCT 系数乘以量化表得到重构的 DCT 系数; 4) 逆 DCT 变换,逆 DCT 变换返回图像的空域。图 4 虚线框中的信息是 JPEG 码流能够正确解码与显示必须的信息,相应地存储在文件头中。JPEG 解码器是根据文件头部保存的这些信息来正确解码和显示 JPEG 文

件。因此,在文件头部缺失的JPEG数据碎片恢复问题上,首先是估算解码所需的各种参数[12](MCU的组成和大小、哈夫曼表、图像的宽度和量化表),使其能够正确地解码。

3.1.1 估计MCU的组成与大小

MCU的大小由采样因子决定。JPEG标准中采用的采样因子通常有以下3种情况{4:1:1,2:1:1,1:1:1}[12],对应的MCU的大小分别为16×16、16×8、8×8。灰度图像只有一个通道,其MCU即为8×8的数据块大小。由于采样因子的选取较少,可以采用穷举的方法来获取正确的采样因子。这是因为当采样因子不匹配时解码会发生错误。可能的错误有以下两种:

1) 溢出错误

当解码数据块时,通过中间符号(n,l)计算前面0的数目n和解码的AC系数的总数超过了63则认为发生了溢出错误。

2) 码字错误

JPEG编码采用的哈夫曼树并非完全的[17],所以当解码的DC系数出现(111111111)等码字时则认为发生了码字错误。

根据解码中溢出错误和码字错误的出现与否,可以用来估计图像先前的采样因子。

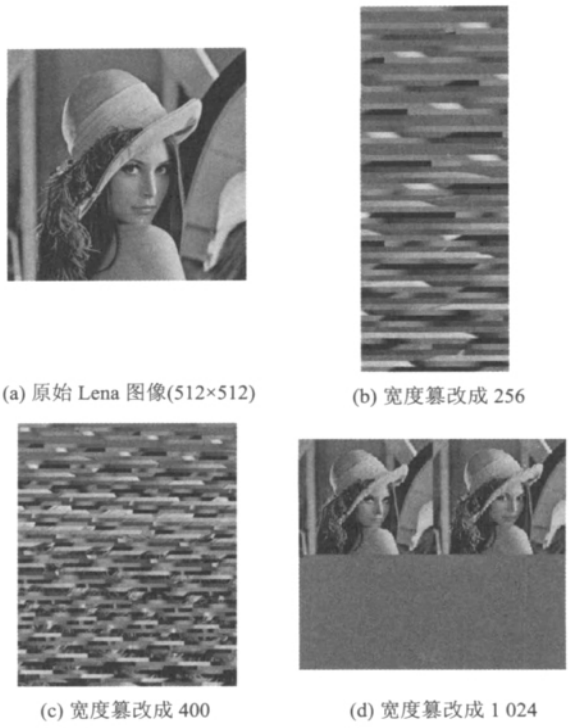
3.1.2 估计图像的宽度

图像的宽度不正确,即使MCU的大小选取正确,解码出来的图像也会出现显示错位。

图5是对Lena图像的宽度进行人为篡改显示的图像。当宽度不匹配时明显影响图像的正常显示。从图5(d)还可以发现当宽度为原始宽度的两倍时可以看到两个相似的小图,其高度相应也变为原来的一半。

亮度分量的DC系数(也即直流分量)保存了图像的大体轮廓信息[18-19],因此对于JPEG图像显示至关重要。并且图像像素之间具有很强的连续性,表现在DCT域也即相邻DC值之间具有很强的关联性[20]。文献[13]提出一种通过计算相邻DC值之间相似性来估计图像宽度的方法,但是该方法的前提条件是图像数据中具有重启标记。

即使在不确定解码起始位置的情况下,根据2.3节的理论分析,错误解码一小段数据流之后能够同步,并通过实验(详见第4.1节)发现一般错误解码的DC系数只占整个DC链中很少的一部分,因此在本节宽度估计算法中将其影响忽略。



(a) 原始Lena图像(512×512) (b) 宽度篡改成256

(c) 宽度篡改成400 (d) 宽度篡改成1 024

图5 不同宽度显示的Lena图像

Fig. 5 The displaying of Lena images with different width

假定解码的差分亮度DC值的序列为{Dif_0,Dif_1,…,Dif_{n-1}},图像的宽度为w,将DC链排列为2维数组,如表3所示。

表3 宽为w的差分DC值的排列

Table 3 The ranking of differential DC values of width w

Dif_0	Dif_1	Dif_3	…	Dif_{w-1}
Dif_w	Dif_{w+1}	Dif_{w+2}	…	Dif_{2w-1}
…	…	…	…	…
Dif_{hw}	Dif_{hw+1}	Dif_{hw+2}		

为了比较相邻DC值之间的相关性,定义

$$S(w)=\frac{1}{4n}\sum_{x=1}^{\lfloor nb/w\rfloor-1}\sum_{y=1}^{w-2}(|Dif(x,y)-Dif(x,y-1)|+|Dif(x,y)-Dif(x-1,y)|+|Dif(x,y)-Dif(x,y+1)|+|Dif(x,y)-Dif(x+1,y)|) \tag{7}$$

$Dif(x,y)$表示在假定宽情况下(x,y)位置像素的差分DC值,$S(w)$表示在假定宽度为w情况下所有相邻DC值的差值权重和。当$S(w)$达到极小值时,表明整幅图像像素越平滑,此时对应的w值最有可能对应着图像实际的宽度大小。

宽度估算的详细算法如下:

输入: 解码出来的亮度 *DC* 系数序列。

输出: 估算的宽度 w。

1) 输入差分 *DC* 序列 $DIF[\,] = (Dif_0, Dif_1, \cdots Dif_{N-1})$;

2) 初始化数组 $S[\,]$; //$S[\,]$ 储存指定假定宽 w 时的函数 $S(w)$ 值;

初始化数组 *Intervrals*[ ] = ( ); //数组 *Intervrals*[ ] 储存数组 *Sim*[ ] 中的极小值;

3) for $i = 1$ to *Length* of *DIF*[ ]

$S[\,] = S(i)$;

end for

4) /* 返回 $S[\,]$ 数组中极小值之间的间距* /

//计算 DC 值数组的宽 w 值

SLIDING_SIZE = 20 //初始化 *SLIDING_SIZE* 值为的实验/经验值;

Times = 0;

//取得局部区域的极小值并保存到数组 *Intervrals*[ ] 中

for *j* = *SLIDING_SIZE* to *sizeOf*(*DC* [ ]) -*SLIDING_SIZE*

for *k* = 0 to *SLIDING_SIZE*

Left += *Sim*[*j* - *k* - 1] - *Sim*[*j* - *k*];

Abs_Left += |*Sim*[*j* - *k* - 1] - *Sim*[*j* - *k*]|;

Right += *Sim*[*j* + *k* + 1] - *Sim*[*j* + *k*];

Abs_Right += |*Sim*[*j* + *k* + 1] - *Sim*[*j* + *k*]|;

end for

end for

if (*Left* > *Abs_Left*/2) and(*Right* > *Abs_Right*/2) and(*Sim*[*j*-1] > *Sim*[*j*] < *Sim*[*j*+1])

then

Intervrals[*Times*++] = *j*;

Left = *Abs_Left* = *Right* = *Abs_Right* = 0;

end for

//通过计算极小值之间的序号间隔来得到宽 w 值(单位: 8 × 8 数据块)

for $i = 1$ to *Times*-1

w += *Intervrals*[*i*] - *Intervrals*[*i* - 1];

end for

$w = w / (Times - 1)$;

5) //输出估算的宽度结果

return w。

3.1.3　哈夫曼表的确定

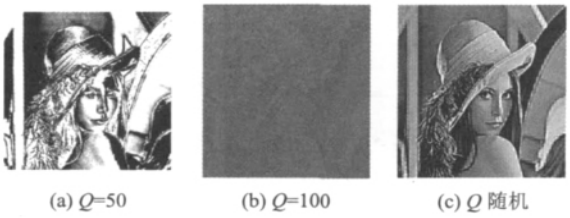
哈夫曼表的选取正确与否直接影响解码 DCT 系数是否正确,因此是解码最为重要的参数。由于哈夫曼表的种类很多,从编码的数据流中估计原始哈夫曼表是 JPEG 文件恢复中的一个难点问题。文献[12]通过分析 Huffman 编码的比特流中特定码字出现的频率来估计是否采用某种哈夫曼表。在一个随机的 m-bit 二进制流中对于给定的 n-bit 序列 ($m >> n$) 出现的频率期望值是 $m/2^n$。而在哈夫曼码流中,某些特定的比特序列频繁出现,如在标准 Huffman 表编码中数据流中,"1010"出现的频率远远大于随机的一段二进制数据流。

通过统计发现大部分数码相机或是 JPEG 生成软件为了减少建立哈夫曼表的时间,大都默认采用 JPEG 标准提供的哈夫曼表。标准哈夫曼表是 JPEG 专家组对几十万幅图像统计得出[17],具有很好的通用性,因此被大部分相机生产商采用。文献[11]对 76 种流行的数码相机进行调查发现,其中有 69 种数码相机(91%) 采用 JPEG 标准默认的哈夫曼表。因此,在本文实验中主要针对标准哈夫曼表编码的 JPEG 数据碎片进行实验。对于其他类型的哈夫曼表,如某些相机生产厂商自定义的哈夫曼表,本文不作考虑。

3.1.4　估计量化表

在 JPEG 编码过程中量化表控制着图像的压缩比率,但是标准中并没有明确规定采用怎样的量化表。因此出现了各种各样的量化表。文献[21]列举了常见相机如 Canon、Sony、Nikon 以及 Adobe Photoshop 等图像处理软件采用的量化表。Kornblum 对数码相机和软件程序生成的 JPEG 图像中量化表进行了分类统计,发现大部分量化表能够分为以下 4 类[22]: Standard tables, Extended standard tables, Custom fixed tables 和 Custom adaptive tables。

为了让解码过程顺利进行,文献[12]通过随机选取常用的量化表来近似代替原始量化表,结果并不影响图像的显示。但是这种方法很多情况下并不合适。这是因为不匹配的量化表会使图像的显示变得模糊和扭曲。图 6 显示了以不同量化表解码的 Lena 图像。图 6(a) 是用质量因子为 50 的量化表取代原始质量因子为 90 的量化表解码显示后的结果; 图 6(b) 是将质量因子为 90 的量化表篡改质量因子为 100 之后的图像; 而图 6(c) 则是随机选取的量化表显示后的结果。



(a) Q=50　(b) Q=100　(c) Q 随机

图 6　不同量化表显示的 Lena 图像

Fig. 6　The Lena images with different quantization tables

为了能够近似重构最合适的量化表，以下分两种情况分别进行讨论。第1种是仅仅头部缺失，但是数据完整；第2种是头部和前面小部分数据丢失情况。对于两种情况的区分可以通过验证文件结构字段信息，当缺少时则按第2种情况来处理。

对于第1种情况，采用的方法是将量化表按比例进行缩放（如标准量化表的100个等级）。然后以不同级别的量化表分别解压缩这幅图像，根据解码过程中的IDCT变换会发生截断[23-24]（即因解码值超出[0,255]范围而进行的截断操作）情况，在截断溢出数达到最小值时对应的量化表即为最合适的量化表。通过大量的统计实验发现几乎所有图像在解码过程中都会存在像素截断的现象，并且量化表越不匹配，发生截断现象也越明显。图7描述的是原始DCT系数直方图与不匹配量化表重压缩后的DCT系数直方图之间的差异。图7(a)(c)是正确量化表对应的DC系数和第3个通道AC系数统计直方图。图7(b)(d)是以不匹配量化表重压缩后显示的DC系数和第3个通道AC系数的直方图。通过比较重压缩前后DCT系数直方图的差异，可以近似重构出最为可能的量化表。

为了度量重压缩前后DCT系数之间的差异，定义相似度量

$$Sim(dCoef,dCoef') = \frac{\sum_{i=1}^{N} |\hat{D}(i) - \check{D}(i)|}{\frac{1}{2}\sum_{i=1}^{N} |\hat{D}(i) + \check{D}(i)|} \tag{8}$$

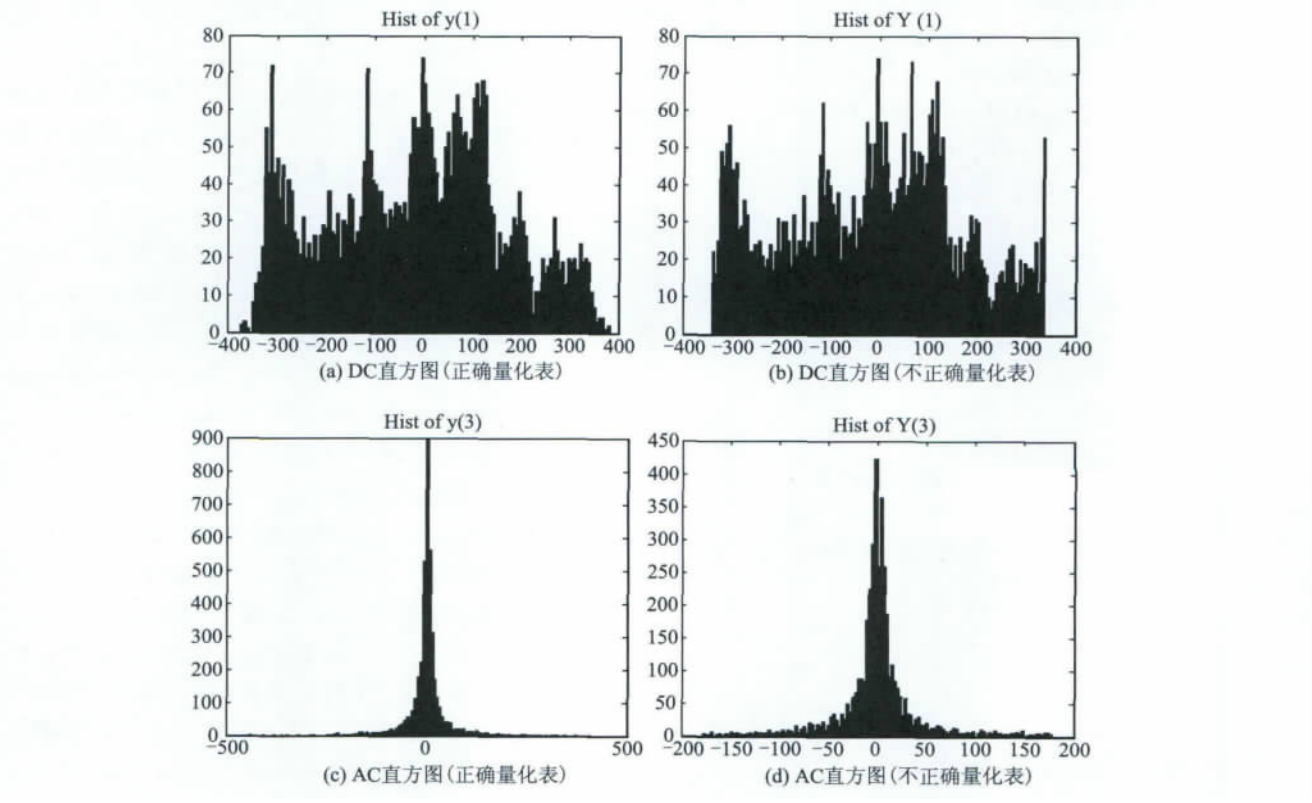


图7 不同的量化表对应的DCT系数直方图

Fig. 7 The DCT coefficient histograms of different quantization tables

$\hat{D}(i)$表示第i个8×8的数据块的量化DCT系数，N是总的8×8数据块数目。$\check{D}(i)$表示通过以式(9)计算所得，它表示以不同量化表压缩的DCT系数。$dCoef$表示原始量化DCT系数，$dCoef'$表示通过可能的量化表重压缩后的量化DCT系数。Sim值越小，说明重压缩后的DCT系数越接近原始值。

$$\check{D}(i) = round(DCT([DCT^{-1}(\hat{D}(i)\times Q_x)]))/Q_x \tag{9}$$

$round$表示取整操作，Q_x表示候选的量化表。

对于第2种情况，可以利用JPEG码流的自同步能力先重构出近似的原始DCT系数，然后采用第一种情况中的方法。但是由于DC值采用差分编码，并且初始

DC 值已经丢失,因此得到的 DC 值为差分值而并非原始值。此时需要先正确估算出近似原始 DC 链。

3.2　显示错误纠正

3.2.1　错位纠正

通过以上估算出的解码参数就可以对 JPEG 数据碎片进行解码。当数据碎片正确解码之后,但由于解码的图像碎片数据往往不是原始图像的左边缘位置开始的,所以解码出来的内容可能会出现显示错位问题。图 8(a) 是在数据完整之下数据块的排列情况; 图 8(b) 是图像数据前面丢失了 4 个数据块; 图 8(c) 是对残缺数据重新解码排列情况; 图 8(d) 是通过固定块填充使数据块排列正确示意图。

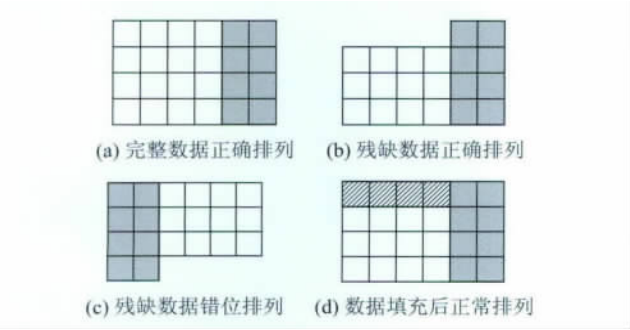


图 8　显示错位及固定块填充

Fig. 8　The mismatched displaying and fixed blocks padding

如图 8(c) 所示,在第 2 列和第 3 列的边缘会出现不连续的一条竖线,所以为了估算 JPEG 数据碎片前面需要填充的数据块的数目,可以先估算这条竖线的位置,然后通过先前宽度估计结果得到需要填充的固定数据块的数目。而一个数据块的 DC 值表示图像大部分信息,所以采用计算相邻两列数据块 DC 值之间的平滑度的方法。如图 8(c) 中由于数据丢失,使得应该在(4,0) 位置的数据块移位到(0,0) 的位置,使得整体在竖线处发生错位。将边缘两列除外,竖线相邻两列的差值应该达到最大值。为了计算这种差异定义平滑度

$$N(w,y)=\begin{cases}\dfrac{\sum_{x=1}^{\lfloor nb/w\rfloor}|Dif(x,y)-Dif(x,y+1)|}{\lfloor nb/w\rfloor} & 1\leqslant y\leqslant w-1\\ \dfrac{\sum_{x=1}^{\lfloor nb/w\rfloor}|Dif(x,1)-Dif(x,w)|}{\lfloor nb/w\rfloor} & y=w\end{cases}\tag{10}$$

$Dif(x,y)$ 表示在(x,y) 位置的差分 DCT 值,w 表示图像的宽度。通过比较相邻列之间差分权重的大小,可以估计出需要填充的数据块数目。估计首行需要填充的数据块数目算法如下:

输入: 解码出来的差分 DC 系数序列和图像的宽度。

输出: 需填充的数据块数目。

1) 输入差分 DC 序列 $DIF[\,]=(Dif_0,Dif_1,\cdots Dif_{N-1})$ 和图像的宽 w;

2) 初始化数组 $Diffrence[\,]$; //$Diffrence[\,]$ 储存指定宽 w 值时的函数 $N(Dif,w,y)$ 的值;

3) for $y=1$ to w

　　$Diffrence[y]=N(Dif,w,y)$;

end for

4) //返回 $Diffrence[\,]$ 数组中最大值的序号;

for $i=1$ to w

　　if $max<Diffrence[i]$ then $max=Diffrence[i]$;

　　$Index=i$;

end for

/* 变量 $Index$ 表示 JPEG 碎片数据块第 1 行块的数量; * /

5) $Miss\_Block\_Number=w-Index$;

/* 变量 $Miss\_Block\_Number$ 表示 JPEG 碎片数据块第 1 行丢失块的数量; * /

6) //输出 JPEG 碎片数据块首行丢失数据块数:

return　$Miss\_Block\_Number$。

3.2.2　彩色偏移的调整

由于 DC 值不是对自身而是相邻像素点之间差值进行编码的,因此当数据不是从首块开始解码时需要确定碎片第 1 个 DC 初始值值。为了让解码能顺利进行,可以先假定其初始值取中间值 0,然后根据显示结果再对其进行调整。对于亮度分量,错误的初始值可能使图像太亮或是太暗; 对于色度分量,可能会太红或是太绿。图 9 所示为彩色图像中 Y,Cb,Cr 分量错误估计的各种示例。当太暗时,可以相应将 Y 分量的初始 DC 值设置大一点,而太亮则相应向相反的方向调整。其他分量的调整和 Y 分量类似,直至显示达到正常的状态。

4　实验结果与分析

4.1　JPEG 数据流的自同步能力实验

2.3 节从理论上说明经过 Huffman 编码,尤其是范式调整的码流具有很强的自同步能力,并给出 Huffman 编码流的同步点近似估计公式。但是

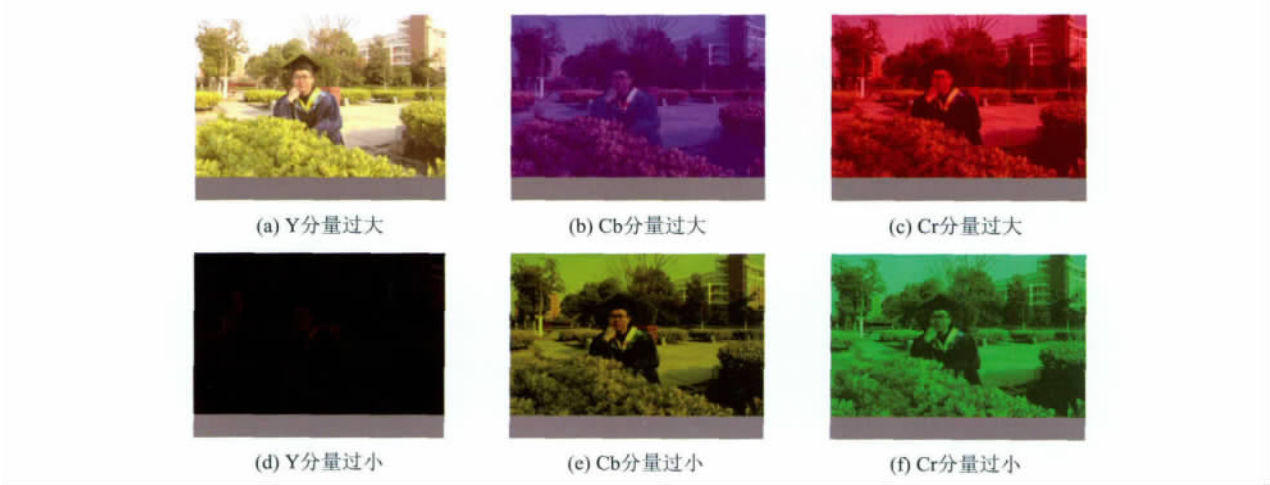
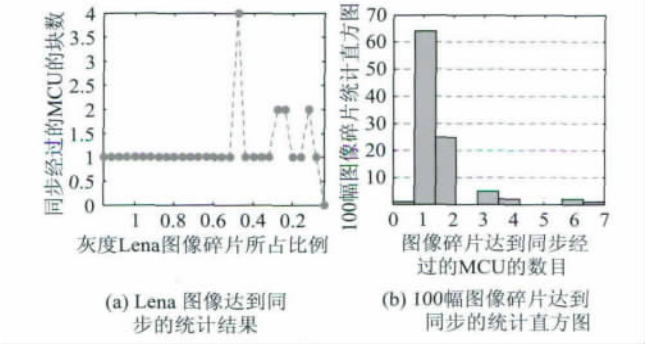


图9 Y、Cb、Cr 错误极限显示情况

Fig. 9 The false limit display of components of Y、Cb、Cr

JPEG 编码则更为复杂，不仅采用 4 个不同的哈夫曼表同时编码，而且分块处理。为了验证 JPEG 数据的自同步能力，首先对图像通过人工删除不同比例的数据来模拟不同解码位置对同步点的影响。实验数据来自标准的图像如 Lena 图像、常用的图像处理库 UCID 以及生活中数码相机拍摄的图像。图 10(a) 是对 Lena 图像的统计结果。从图 10(a) 可以发现大部分数据碎片在经过一个 MCU 的错误解码之后能够达到同步，最多也只是经过4 个MCU 的大小。为了验证不同 JPEG 文件碎片的自同步能力，图 10(b) 统计了 100 幅图像碎片达到同步的统计直方图。其结果是大约有 63% 的图像碎片在错误解码一个 MCU 之后达到同步点。错误解码的 MCU 区间范围为 [0,7]。0 说明在经过一个 MCU 之后就达到了同步。实验结果验证了即使在没有重启标记的情况下，JPEG 数据流的自同步能力可以使 JPEG 碎片到达正确解码的目的。



(a) Lena 图像达到同步的统计结果 (b) 100幅图像碎片达到同步的统计直方图

图10 JPEG 图像碎片达到同步经过 MCU 块数

Fig. 10 The number of MCU when reach synchronization

4.2 图像宽度的估算实验

在图像宽度的估算实验中，为了验证 3.1.2 节改进方法在没有重启标记情况下的有效性，对完整的数据部分和不完整的数据部分图像宽度的估算结果进行比较，如图 11 所示。图 11(a)(c) 分别是对宽度都为 512 的 Lena 图像和图 9 中的图像在数据完整情况下的估算结果，发现在 64 的整数倍周期性地达到极小值。图 11(b)(d) 分别是对 Lena 图像和图 9 中的图像在数据缺失情况下的估算结果。同样也是在 64 的倍数周期性地达到极小值，只是幅度相较于在数据完整情况下偏小一些。而图像的原始宽度为 $64\times8=512$，验证了在没有重启标记情况下图像宽度估计算法的有效性。

4.3 量化表的近似重构实验

为了近似估计 JPEG 数据采用的量化表，图 12 显示了 Lena 图像量化表的估算结果。图 12 是原始质量因子分别为 75、80、85 和 90 的估计结果。图 12(a) 在靠近 75 的附近 73、74、76 同样得到最小值，并且为 0，说明原始量化表可能是其中的一个。而图 12(d) 在质量因子为 90 时只有一个极小值，说明原始量化表最接近质量因子为 90 的量化表。从图中还可发现，当量化表趋于原始量化表时，其相似性匹配权重达到最小值。并且算法估计的准确性直接与图像的质量有关。当质量越低时，估计的量化表可能会有多个选择，但是选取的量化表对于人眼来说很难觉察。

4.4 恢复实例

这里列举了几个恢复实例。图 13(a) 是原始完

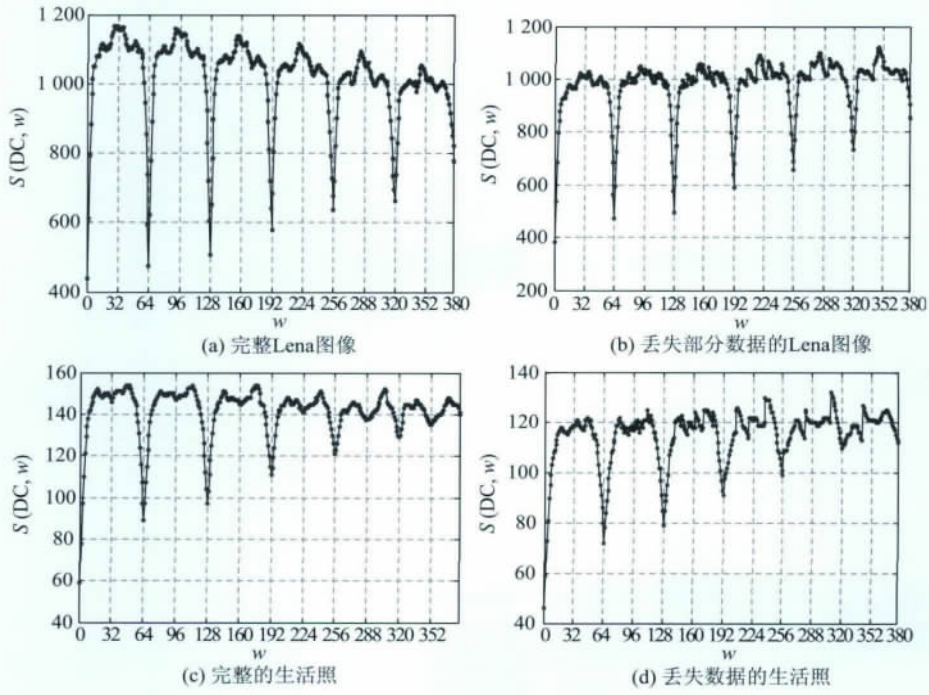


图 11　JPEG 图像宽度估计结果

Fig. 11　JPEG images width estimation result

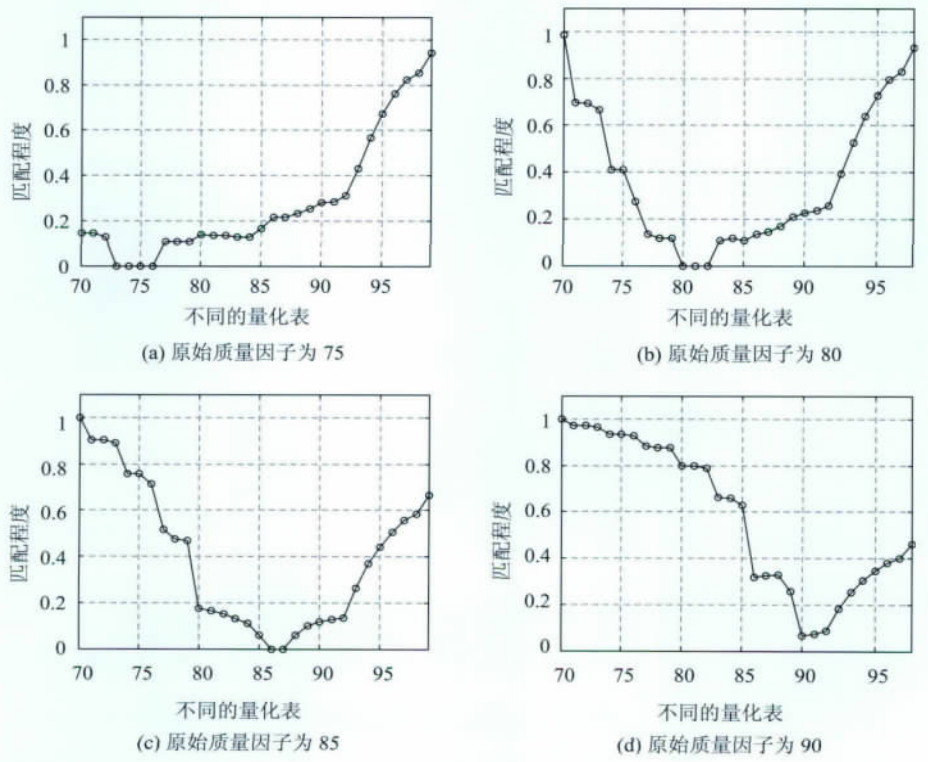


图 12　Lena 图像量化表估算结果

Fig. 12　Quantization tables estimation result for Lena image

好的一幅图像，图13(b)是将头部及数据的前半部分删除后，通过本文方法解码并显示的图像。可以看出不仅显示较暗，人和物不易识别，而且数据块的显示从中间出现了错位现象。图13(c)是采用固定块填充算法填充后显示的图像，仍然存在颜色偏移的问题。适度调高亮度分量的初始DC值，使显示更清晰，图13(d)是对初始DC值进行调节后显示的图像。

图13 受损图像恢复结果

Fig. 13 The recovery result for one of damaged images

为了验证算法的普遍性，图14列举了3个来源于常用的数字图像(如Lena图像)、常用的图像库以及生活照片JPEG碎片的恢复结果。

5 结 论

本文提出一种用于解码和显示没有头部和重启标记的JPEG文件碎片的方法。

分析了经过Huffman编码的数据流的特点，即使在没有重启标记的情况下数据流本身就具有很强的自同步能力。这是解决本文问题的基础。首先是估计解码所需的重要参数，对哈夫曼表、采样因子、图像的宽度和量化表等重要的解码参数提出了有效的估算方法；其次是给出了解码后数据显示出现的图像错位和色彩偏移等问题的解决方法。这些技术能够帮助计算机取证调查人员恢复磁盘上损坏的JPEG文件。实际上，利用本文估算出来的参数可以很方便地给JPEG碎片重构一个伪头部结构，从而实现把碎片恢复成一个“完整的”JPEG文件。

在图14量化表的估计结果可能存在细微的偏差，以及在图13的宽度估计过程中估计的宽度也会和原始的宽度存在细微的差别。所以在以后的研究中将提高估算解码参数算法的准确性，并通过更多的实验来验证本文算法的有效性。同时将注意力转移到如何鉴别磁盘上可能的JPEG碎片，并改进算法以适应真实而复杂的磁盘环境。

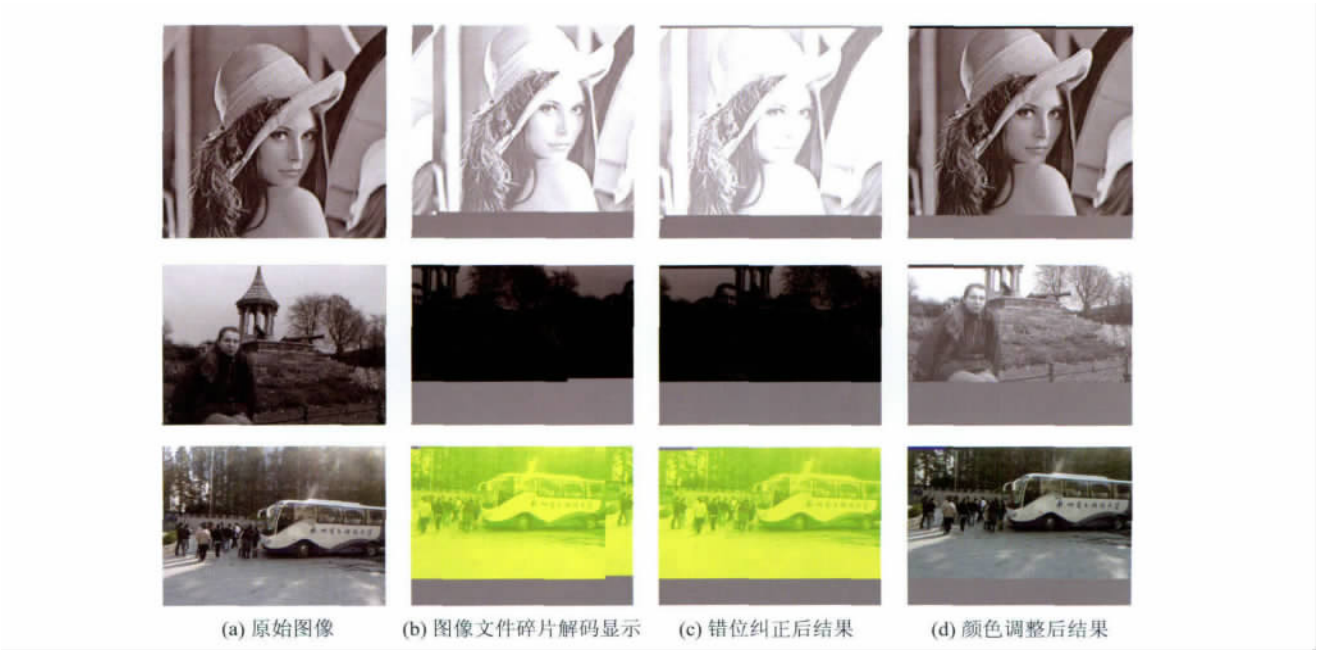


图14 JPEG碎片的恢复结果3个实例

Fig. 14 The final estimation results for proposed date sets

参考文献(References)

[1] Wortley R, Smallbone S. Child pornography on the internet [EB/OL]. (2006-05-01) [2011-12-30]. http://www.cops.usdoj.gov/Publications/e04062000.pdf.

[2] Ruscitti J. Child porn fight merits police resources [EB/OL]. (2008-03-14) [2011-12-30]. http://www.crime-research.org/news/16.02.2008/3201/.

[3] Karresand M, Shahmehri N. Oscar-file type identification of binary data in disk clusters and RAM pages [C]//Proceedings of the IFIP International Information Security Conference: Security and Privacy in Dynamic Environments. Berlin: Springer Verlag, 2006: 413-424.

[4] Garfinkel S. Carving contiguous and fragmented files with fast object validation [J]. Digital Investigation, 2007, 4(s): 2-12.

[5] Digital Assembly LLC. SmartCarving [EB/OL]. [2010-06-30]. http://digital-assembly.com/products/adroit-photo-forensics/features/smartcarving.html.

[6] Mikus N. An analysis of disc carving techniques [D]. Monterey, California, USA: Naval Postgraduate School, 2005.

[7] Golden G R Ⅲ, Vassil R. Scalpel: A frugal, high performance file carver [C]//Proceedings of the 2005 Digital Forensic Research Workshop. New Orleans: Elsevier Press, 2005: 1-10.

[8] Pal A, Shanmugasundaram K, Menmon N. Automated reassembling images fragments [C]//Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. New York: IEEE Press, 2003: 732-735.

[9] Pal A, Sencar H T, Memon N. Detecting file fragmentation point using sequential hypothesis testing [J]. Digital Investigation, 2008, 5(s): 2-13.

[10] Memon N, Pal A. Automated reassembly of file fragmented images using greedy algorithms [J]. IEEE Transactions on Image Processing, 2006, 15(2): 385-393.

[11] Karresand M, Shahmehri N. Reassembly of fragmented JPEG images containing restart markers [C]//Proceedings of the European Conference on Computer Network Defense. Washington DC: IEEE Press, 2008: 25-32.

[12] Sencar H T, Memon N. Identification and recovery of JPEG files with missing fragments [J]. Digital Investigation, 2009, 6(s): 88-98.

[13] Xu Y, Xu M. Width extraction of JPEG fragment via frequency coefficients scale similarity measuring [C]//Proceedings of the 2nd International Conference on Future Computer and Communication. Wuhan, China: IEEE Press, 2010: 513-517.

[14] Huffman D A. A method for the construction of minimum redundancy codes [J]. Proceedings of the IRE, 1952, 40(9): 1098-1101.

[15] Ferguson T, Rabinowitz J H. Self-synchronizing Huffman codes [J]. IEEE Transactions on Information Theory, 1984, 30(4): 687-693.

[16] Klein S T, Wiseman Y. Parallel Huffman decoding with applications to JPEG files [J]. The Computer Journal, 2003, 46(5): 487-497.

[17] Wallace G K. The JPEG still picture compression standard [J]. IEEE Transactions on Consumer Electronics, 1991, 34(4): 30-44.

[18] Lu Y, Wong T T, Heng P A. Digital photo similarity analysis in frequency domain and photo album compression [C]//Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia. New York: ACM, 2004: 237-244.

[19] Popovici I, Withers W D. Locating edges and removing ringing artifacts in JPEG images by frequency-domain analysis [J]. IEEE Transactions on Image Processing, 2007, 16(5): 1470-1474.

[20] Lin Z C, He J F, Tang X O, et al. Fast, automatic and fine-grained tampered JPEG iamge detection via coefficient analysis [J]. Pattern Recognition, 2009, 42(11): 2492-2501.

[21] Calvin H. JPEG compression quality from quantization tables [EB/OL]. (2006-01-01) [2011-12-30]. http://www.impulseadventure.com/photo/jpeg-quantization.html.

[22] Kornblum J D. Using JPEG quantization tables to identify imagery processed by software [J]. Digital Investigation, 2008, 5(s): 21-25.

[23] Fridrich J, Doljan M, Du R. Steganalysis based on JPEG compatibility [C]//Proceedings of the SPIE Multimedia Systems and Applications. Washington DC: SPIE Press, 2001: 275-280.

[24] Ye S, Sun Q, Chang E C. Detecting digital image forgeries by measuring inconsistencies of blocking artifact [C]//Proceedings of the IEEE International Conference on Multimedia and Expo. Beijing, China: IEEE Press, 2007: 12-15.