适用于Android智能手机的灰度均值水印算法

彭 乐 金 聪
(华中师范大学计算机学院 武汉 430079)

摘 要 提出了一种适用于Android智能手机的灰度均值数字水印算法。在嵌入阶段,将经过像素重新组织的水印图像嵌入载体图像中;在提取阶段,先提取出初始水印,再判断初始水印是否需要区域替换,如果需要,则进行区域替换,否则不进行;最后,将初始水印还原成所需的水印图像。实验表明,该算法在具有良好水印透明性的同时,对JPEG压缩、加噪声、裁剪等常见的攻击具有鲁棒性。

关键词 灰度均值,Android智能手机,数字水印,重新组织,区域替换
中图法分类号 TP309.2 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.7.018

Digital Watermarking Algorithm for Android Smart Phones

PENG Le JIN Cong
(School of Computer Science, Central China Normal University, Wuhan 430079, China)

Abstract In this paper, a digital watermarking algorithm based on mean gray value was proposed for the Android smart phones. In the embedding phase, the watermark after the pixel reorganization is embedded into the carrier image. In the extraction phase, the initial watermark is extracted at first, and then whether the initial watermark needs replacing its areas is determined. If necessary, the areas with a better one are replaced. Otherwise, any areas won't be replaced. Finally, the initial watermark is transformed into the desired watermark. Experimental results show that the algorithm has a fine transparence of embedded watermark and is robust to attacks such as noise adding, compressing and cropping.

Keywords Mean gray value, Android smart phones, Digital watermarking, Reorganization, Regional replacement

1 引言

目前,Android平台手机的全球市场份额稳居第一。显然,Android智能手机的出现丰富了我们的日常生活,给我们带来了极大的便利。然而Android智能手机强大的多媒体和网络功能使得图像版权保护问题越来越突出,已成为迫切需要解决的问题。数字水印技术是解决版权保护问题的有力工具。目前,图像数字水印算法[1]很多,但针对Android智能手机的有效图像水印算法却很少。智能手机的运算能力远不能与个人电脑相比,存储空间也较小,这使得许多优秀的图像水印算法由于空间或时间复杂度大而在Android平台上无法正常运行,因此,Android智能手机的图像水印算法设计是一个挑战。

数字水印算法大致可分为空间域和变换域两类。前者直接修改载体图像的像素位,常见的有LSB算法[2]、Patchwork算法、基于直方图的算法[3]等。变换域算法先对图像进行变换,然后在变换域实现水印嵌入。与变换域算法相比,空间域算法具有运算速度上的优势及设计简单的优点。因此,空间域上的图像水印算法更适合Android手机环境。

文献[4]提出了一种基于灰度均值的图像水印算法,它是利用奇偶量化[6]实现的,而且该算法对JPEG压缩、添加噪声及滤波等常见攻击具有良好的鲁棒性。然而,该算法不能抵抗裁剪等作用在局部范围内的攻击,主要原因是分区均值对分区内部的像素点突变过度敏感,而裁剪会造成分区内的所有像素值突变为0或1。同时,由于安全性要求,算法在嵌入水印前会对水印图像进行置乱[7],这使得水印像素值均匀地分散在图像的各个区域。针对上述不足,本文在水印嵌入前将水印像素重新组织,此过程类似于隔行、隔列采样,最终使水印图像变成由4个相似区域构成的图像。这样做不仅将各水印像素均匀分散,而且更重要的是可以使用受损程度小的区域代替受损程度大的区域,使提取出的水印相关值大大提高。

2 所提出的图像水印算法

2.1 图像分区

为了描述方便,假设载体图像I的大小为$M\times N$,水印W的大小为$m\times n$,W为二值水印。$M=km$,$N=ln$。

将I均匀地不重叠地分成$m\times n$个大小为$k\times l$的区域B_{ij},$i=1,2,\cdots,m$;$j=1,2,\cdots,n$。

2.2 水印嵌入

2.2.1 二值水印图像的预处理

如果直接将大小为$m\times n$的水印图像嵌入到载体图像的

到稿日期:2014-07-08 返修日期:2014-10-13 本文受武汉市科技攻关计划(201210121023)资助。
彭 乐(1989—),男,硕士生,主要研究方向为数字水印,E-mail:369629308@qq.com;金 聪(1960—),女,博士,教授,主要研究方向为数字水印、图像处理、信息安全等。

$m\times n$ 个均匀不重叠的分区中得到隐秘图像，那么隐秘图像受到局部区域上的攻击尤其是裁剪攻击时，受攻击区域中的水印信息可能会部分甚至全部丢失。

为了使算法具有抵抗上述攻击的能力，本文将水印图像中的像素按图 1 所示的方式重新组织。

(a)原始水印图像　　(b)像素重新组织后的水印图像

图 1

图 1(a)是原始水印图像，图中的每个方格代表图像的一个像素，方格中的 1,2,1′,…表示的是像素值(0 或 1)。一般情况下，经过像素重新组织，原始水印图像会变成由 4 个相似区域(图 1(b)中 1、2、3、4 组成的左上区域，1′、2′、3′、4′组成的右上区域，1″、2″、3″、4″组成的左下区域，1‴、2‴、3‴、4‴组成的右下区域)组成的水印图，而且每个区域的内容均与原始水印图相似。如图 2 所示，将水印图像的 4 个相似区域依次编号为 1、2、3、4。经过像素重新组织，原始水印图像的局部区域内的像素得到均匀分散。

图 2　水印图像的区域编号

图 3(a)是一幅原始水印图，图 3(b)为图 3(a)像素重新组织得到的水印图。

(a)原始水印图像　　(b)重新组织像素后的水印图像

图 3

水印图像重新组织像素后，其 1,2,3,4 区域中的值为 1 的像素数(一共 4 个)及后面嵌入过程中的量化步长将一起作为密钥，在水印提取时使用。

2.2.2　嵌入过程

由于本文算法要对彩色图像的区域灰度均值进行量化以嵌入水印信息，因此先将彩色图像转化成灰度图，然后在灰度图中嵌入水印，再把嵌入水印的灰度图转化为彩色图像，这种方案是可行的。但在 Android 智能手机上，彩色图像与其对应灰度图之间的转换不仅耗时，而且需要用一定的存储空间来存放灰度图。

设彩色图像像素值的 3 个分量分别为 R,G,B，则利用式(1)可得如下值：

$$Gray=0.299R+0.587G+0.114B \tag{1}$$

本文中 $Gray$ 代表灰度值。嵌入水印时需要将 $Gray$ 修改为 $Gray+\Delta$，可以直接将 3 个分量 R,G,B 分别加上 Δ 来将 $Gray$ 值修改为 $Gray+\Delta$，如式(2)所示。

$$Gray+\Delta=0.299(R+\Delta)+0.587(G+\Delta)+0.114(B+\Delta) \tag{2}$$

由式(2)可直接修改载体彩色图像的像素值来达到量化区域灰度均值的目的。由于不用进行彩色图像与对应灰度图之间的相互转换，因此节省了嵌入时间和存储空间。

本文的均值量化方法与文献[4]的方法类似。此处简单描述如下：

1)将载体图像划分成大小相等且互不重叠的区域，分别求出各区域的灰度均值。

2)使用奇偶量化方法量化各区域灰度均值以嵌入对应的水印信息。

3)当所有区域的灰度均值量化完成，便得到隐秘图像。

上述嵌入过程不需要在载体图像对应的灰度图上进行。另外，对于步骤 2)，本文修改区域灰度均值的方法与文献[4]不同。设区域 B_{ij} 的大小为 $w\times h$，原区域灰度均值为 I_{ij}，修改后的区域灰度均值为 I'_{ij}，对 B_{ij} 中的每一像素使用式(3)修改：

$$\begin{cases} r'_{xy}=round(r_{xy}+I'_{ij}-I_{ij}) \\ g'_{xy}=round(g_{xy}+I'_{ij}-I_{ij}) \\ b'_{xy}=round(b_{xy}+I'_{ij}-I_{ij}) \end{cases} \tag{3}$$

其中，$x=1,2,\cdots,w;y=1,2,\cdots,h;r_{xy},g_{xy},b_{xy}$ 代表位于分区 B_{ij} 的第 x 行第 y 列的像素值的 3 个分量，r'_{xy},g'_{xy},b'_{xy} 为 r_{xy},g_{xy},b_{xy} 修改后的值。$round$ 表示四舍五入运算。

本文使用加法来修改区域灰度均值。由于像素灰度值是整数，故区域中各个像素灰度值修改量约为 $I'_{ij}-I_{ij}$(误差不超过 0.5)。与文献[4]的乘法修改相比，本方法的明显优势是运算速度的提高。表 1 显示了使用两种不同修改方法时的 PSNR 值(单位 dB)。

表 1　使用两种不同修改方法时的 PSNR 值

修改方法	Lena	Baboon	Sailboat	Airplane
加法	40.7038	40.7343	40.6862	40.6822
乘法	40.5368	40.4609	40.3460	40.5731

表 1 使用大小为 512×512 的 Lena, Baboon, Sailboat, Airplane 的彩色图像测试，步长 δ 取 4，分区大小为 8×8。本文将修改均摊给区域中的各个像素，而文献[4]中修改集中在灰度值大的像素上，这使得加法修改的水印透明性(以 PNSR 度量)在上述测试条件下略显优势，测试结果也说明了这点。图 4 显示了分别使用两种不同修改方法时的效果。

(a)使用加法修改　　(b)使用乘法修改

图 4

2.3　水印提取

2.3.1　经过像素重新组织的水印图像提取

本文算法在提取像素重新组织的水印图像时所使用的方法与文献[4]的提取方法类似。具体步骤如下：

1)对载体图像进行区域划分后，分别求出各区域的灰度均值。

2)设某个区域的灰度均值为 $\bar{r}$，按式(4)确定该区域对应的水印图像像素值：

$$w'=\left\lfloor \frac{\bar{r}}{\delta} \right\rfloor \bmod 2 \tag{4}$$

其中，δ 表示步长，$\lfloor\ \rfloor$ 表示向下取整。

3)取出所有水印图像像素值，从而得到要提取的水印 W'。

需要说明的是，量化步长 δ 的选取是在水印透明性和鲁

棒性之间的折衷，δ 大则鲁棒性好而透明性差，否则反之。由于不同的载体图像可能差异较大，故无法用统一的公式来确定 δ 值，实际中一般用实验方法来确定。本文通过多次实验观察水印透明性与鲁棒性，认为 δ 取 4 较好。

在步骤 2)中，本文计算水印像素值 w' 的方法与文献[4]不同。本文没有使用扰动 E 的均值 $mean(E)$。原因是 $mean(E)$ 在受到 JPEG 压缩、零均值高斯噪声与椒盐噪声、平滑等攻击时相对于所取的步长 δ 来说非常小，而且当隐秘图像受到局部裁剪攻击时，$mean(E)$ 的引入有时(如 $mean(E)=\delta$ 时)会造成未裁剪区域提取出的水印像素值 w' 与原始水印中的值全部相反(即 w' 为 0 时原始水印中的值为 1，w' 为 1 时原始水印中的值为 0)。

2.3.2 水印图像的恢复

W' 对应于像素重新组织后的水印图像，应该用上述水印图像预处理的逆过程来重新组织 W' 的像素，以生成对应于原始水印图像的水印 W''。这里组织像素的方式如图 5 所示。

1	2	1′	2′
3	4	3′	4′
1″	2″	1‴	2‴
3″	4″	3‴	4‴

(a)水印 W'

1	1′	2	2′
1″	1‴	2″	2‴
3	3′	4	4′
3″	3‴	4″	4‴

(b)水印 W''

图 5 W' 的像素组织方式

2.3.3 区域之间的替换

为了便于说明，这里使用还原之前的水印图像 W'。2.2.1节提到，如果没有受到攻击，一般区域 1、2、3、4 是相似的。因此，如果其中的某个区域因受到攻击受损严重，则可以用其它受损较轻的区域代替受损严重的区域。基于这种思想，本文在满足一定条件下用受损最轻的区域来取代受损程度超过一定阈值的区域。

设原始水印图像为 W，经过像素重新组织的水印图像为 $\hat{W}$，W' 与 $\hat{W}$ 的 1，2，3，4 区域的像素值为 1 的像素数分别为 n_1'，n_2'，n_3'，n_4' 与 $\hat{n}_1$，$\hat{n}_2$，$\hat{n}_3$，$\hat{n}_4$。区域 i 受损的程度 d_i 用式(5)表示：

$$d_i=\left|\frac{n_i'-\hat{n}_i}{n_i'}\right|,\ i=1,2,3,4 \tag{5}$$

虽然 d_i 不能完全准确地反映区域 i 的受损情况，但只要 d_i 不为 0，则一定可以说明区域 i 受损，且 d_i 越大受损越严重。

区域之间的替换不是在每次提取水印时都进行，而是在满足式(6)时才进行。

$$\exists(d_i-d_j)>T_1,\ i\neq j \text{ 且 } i,j=1,2,3,4 \tag{6}$$

每个被替换的区域应满足式(7)：

$$d_i>T_2,\ i=1,2,3,4 \tag{7}$$

其中，T_1，T_2 是两个阈值，$\exists$ 表示存在。替换时使用受损程度最轻的区域(其受损程度为 d_1，d_2，d_3，d_4 的最小值)去替换受损程度超过 T_2 的区域。

T_1 较大说明 1，2，3，4 区域受损程度之间差别较大，此时才可能进行区域之间的替换，否则不会替换。T_2 取较大值说明仅对受损较严重的区域进行区域之间的替换。

T_1 与 T_2 的取值范围均为(0，1)。T_2 只是说明要进行区域之间的替换时区域受损应该满足的条件，其取值与人的主观意愿有关。在本文实验中，取 $T_2=0.5$，$T_1=0.52$。

3 实验结果

实验中取大小为 64×64 的有意义二值图像作为水印，载体图像取大小为 512×512 的 Lena，Baboon，Sailboat，Airplane 的彩色图像，步长 $\delta=4$，如图 6(a)－图 6(e)所示。

(a)Lena

(b)Baboon

(c)Sailboat

(d)原始水印图像

(e)Airplane

图 6

本文用峰值信噪比(PSNR)评价嵌入水印后的水印透明性，用归一化相关系数(NC)衡量提取出的水印质量。

下面将依次介绍本文算法的时间复杂度、水印透明性及鲁棒性，并与文献[4,5]的算法进行性能比较。

3.1 算法的时间复杂度

本文算法时间复杂度测试的平台是三星 S7572 手机，其 CPU 的频率是 1228MHz，RAM 容量为 768MB，操作系统为 Android OS 4.1。测试方法为每张图像重复运行本文算法 10 次，取时间的平均值。一般变换域算法在嵌入水印前需进行空间域到变换域的转换，因此其时间复杂度高于本文算法。而文献[4]算法和本文算法同属空域水印算法，故这里与之对比运行时间。测试结果如表 2 所列。

本文算法在嵌入水印计算分区均值和修改分区均值时需分别遍历一次载体图像，提取水印时需遍历一次隐秘图像。若忽略对水印像素的重新组织，则本文嵌入算法与提取算法的时间复杂度均为 $O(M \cdot N)$，容易看出文献[4]算法的嵌入与提取时间复杂度也为 $O(M \cdot N)$，本文算法比文献[4]算法耗时稍长，其原因为本文算法的处理对象是彩色图像，并且需要对水印图像重新组织像素。这说明本文算法时间复杂度很低，适合运算能力低的 Android 智能手机。

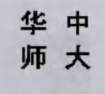
表 2　本文算法与文献[4]算法的运行时间对比

图像	嵌入时间(s)		提取时间(s)	
	本文	文献[4]	本文	文献[4]
Lena	约 2.5368	约 2.3600	约 0.9081	约 0.7935
Baboon	约 2.5548	约 2.3393	约 0.8834	约 0.7974
Sailboat	约 2.5379	约 2.3127	约 0.8735	约 0.7934
Airplane	约 2.5247	约 2.3500	约 0.8825	约 0.8003

3.2　水印透明性

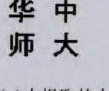
图 7(a)－图 7(d)是隐秘图像，图 7(e)－图 7(h)是图 7(a)－图 7(d)隐秘图像未受攻击时提取出的水印。由图 7 可知，隐秘图像的 PSNR 值均超过 40dB，这说明本文算法具有较好的水印透明性。NC 值均为 1，表明在载体图像未遭受攻击时，提取出的水印与原始水印没有差别，这进一步说明本文算法能够完全正确地提取出嵌入载体图像的水印。文献[4]为空间域算法，量化步长 δ 取 4，其隐秘图像 Lena 的 PSNR 为 40.6699dB，文献[5]为变换域算法，在量化步长 $Q=70$ 时 PSNR 也在 40 左右，均与本文的相当。

(a)隐秘图像 Lena
(PSNR=40.7038dB)



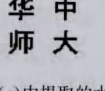
(b)从(a)中提取的水印
(NC=1.0000)

(c)隐秘图像 Baboon
(PSNR=40.7343dB)



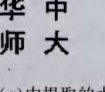
(d)从(c)中提取的水印
(NC=1.0000)

(e)隐秘图像 Airplane
(PSNR=40.6822dB)



(f)从(e)中提取的水印
(NC=1.0000)

(g)隐秘图像 Sailboat
(PSNR=40.6822dB)



(h)从(g)中提取的水印
(NC=1.0000)

图 7

3.3　水印鲁棒性

对嵌入水印的 Lena，Baboon，Airplane，Sailboat 彩色图进行鲁棒性测试。模拟攻击有 JPEG 压缩、加噪声、平滑及裁剪。表 3 是上述 4 幅图像在进行不同质量因子的 JPEG 压缩后提取出水印图像的 NC 值。从表 3 可见，本算法对于 JPEG 压缩攻击的鲁棒性比较好，在质量因子大于 50 的情况下，JPEG 压缩对本算法的水印提取基本没有影响。

当质量因子小于 30 时，NC 值开始急剧下降，导致水印提取失效。原因是质量因子低于 30 以后，压缩后的图像相对于隐秘图像而言在嵌入水印的分区中已经发生了较大的改变。比如许多原来相邻且差别不是很大的分区，压缩后变得完全相同。

表 3　隐秘图像经过 JPEG 压缩后提取的水印 NC 值

质量因子	60	50	30	25
Lena	1.0000	1.0000	0.9719	0.5313
Baboon	1.0000	1.0000	0.9824	0.5245
Airplane	1.0000	1.0000	0.9574	0.5194
Sailboat	0.9997	0.9997	0.9841	0.5018

为测试本文算法对噪声攻击的鲁棒性，先对 4 幅隐秘图像添加零均值且方差不同的高斯噪声。表 4 是添加零均值高斯噪声后的图像提取出的水印 NC 值。

表 4　隐秘图像添加零均值高斯噪声后提取出的水印 NC 值

方差	0.001	0.002	0.003	0.004
Lena	0.9923	0.9602	0.9012	0.8510
Baboon	0.9946	0.9560	0.9131	0.8516
Airplane	0.9943	0.9534	0.9004	0.8422
Sailboat	0.9946	0.9514	0.9012	0.8414

表 5 给出了 4 幅嵌入水印图像后再添加密度不同的椒盐噪声得到的结果。容易看出，添加椒盐噪声后图像中提取的水印均具有较高的 NC 值。这表明，本文算法对椒盐噪声具

有比较高的鲁棒性。

表5　隐秘图像添加不同密度椒盐噪声后提取的水印NC值

噪声密度	0.001	0.002	0.003	0.005
Lena	华中师大 0.9920	华中师大 0.9849	华中师大 0.9690	华中师大 0.9395
Baboon	华中师大 0.9951	华中师大 0.9875	华中师大 0.9781	华中师大 0.9582
Airplane	华中师大 0.9889	华中师大 0.9795	华中师大 0.9648	华中师大 0.9483
Sailboat	华中师大 0.9826	华中师大 0.9801	华中师大 0.9608	华中师大 0.9293

本文采用模板大小为3×3、标准差为1的高斯滤波和模板大小为3×3的中值滤波进行对平滑攻击的鲁棒性实验。实验结果如表6所列。

表6　隐秘图像经过平滑攻击后提取的水印NC值

平滑方式	Lena	Baboon	Airplane	Sailboat
高斯滤波 模板3×3 标准差为1	华中师大 0.9356	华中师大 0.8970	华中师大 0.9219	华中师大 0.9268
中值滤波 模板3×3	华中师大 0.9730	华中师大 0.8181	华中师大 0.9455	华中师大 0.8828

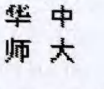
表6说明了本文算法对高斯滤波和中值滤波两种平滑攻击也表现出良好的鲁棒性。从表中对Baboon滤波攻击的结果来看，本算法对Baboon这类边缘丰富的彩色图像平滑攻击的鲁棒性还有待提高。

为测试本文算法对裁剪攻击的鲁棒性，以隐秘图像Lena为例，从隐秘图像上分别裁去部分区域，然后分别提取对应的水印，如图8—图10所示。

(a)从隐秘图像上半部分裁去328×200的区域

(b)从(a)中提取的水印(NC=0.9767)

(c)从隐秘图像下半部分裁去512×128的区域



(d)从(c)中提取的水印(NC=0.9758)

图8

(a)从隐秘图像左半部分裁去128×512的区域

(b)从(a)中提取的水印(NC=0.9685)

(c)从隐秘图像右半部分与下半部分同时裁去全图1/4区域

(d)从(c)中提取的水印(NC=0.9673)

图9

(a)从隐秘图像中间裁去256×256区域

(b)从(a)中提取的水印(NC=0.7648)

(c)从隐秘图像中间裁去512×128区域

(d)从(c)中提取的水印(NC=0.7424)

图10

从图8—图10可见，本文算法具有比较好的抗裁剪攻击能力。若将隐秘图像均匀而不重叠地分成4个大小相同的区域，则各区域因裁剪攻击造成的破坏程度越不均匀，本文算法优势越明显，其原因为本文算法中水印图像的区域替换策略是用受损轻的区域替换受损严重的区域。实验结果表明，水印图像区域1,2,3,4是相似的，且与原始水印图像相似。图10(b)与图10(d)的水印图像提取过程并没有进行区域替换，虽然丢失了部分水印信息，但水印图像中的文字内容依然可见。

将本文算法与文献[4]比较，选择的攻击方式有JPEG压缩、添加高斯噪声和椒盐噪声，因文献[4]使用的是灰度图像，故先将彩色图像转化为灰度图像。将式(3)改为式(8)：

$$Gray'_{xy}=round(Gray_{xy}+I'_{ij}-I_{ij}) \tag{8}$$

对比结果如表7所列。

表7　本文算法与文献[4]的比较结果

攻击类型	文献[4]的NC值	本文算法NC值
JPEG压缩(质量因子50)	1.0000	1.0000
JPEG压缩(质量因子25)	1.0000	0.5160
高斯噪声(方差0.001)	0.9571	0.9492
椒盐噪声(密度0.002)	0.9579	0.9387

当JPEG压缩质量因子为30时，本文算法的NC值为0.9645。由表7知，除了质量因子为25的JPEG压缩，其它3种情况下本文算法的NC值与文献[4]的结果相差不大，这表

明本文算法能够保持灰度均值算法的优势。

因变换域算法在现今被广泛使用，而且与空域图像水印相比，基于 DCT 的数字水印对压缩、滤波和其他一些攻击具有更强的鲁棒性，故将本文算法与文献[5]进行比较。实验中本文步长 $\delta=4$，这与文献[5]中量化步长 $Q=70$ 时的 PSNR 相当。

由于文献[5]使用的水印图像大小为 32×32，而嵌入的信息量直接影响水印的鲁棒性，因此本文采用相同的水印与文献[5]进行鲁棒性比较。根据本文的分区方法，分区大小变为 16×16。同样，先将 Lena 彩色图像转换成 256 级灰度图像并使用式(8)来修改载体图像。水印图像如图 11 所示，实验结果如表 8 所列。

图 11　与文献[5]比较所用的水印

表 8　本文算法与文献[5]的对比结果

攻击类型	本文算法 NC 值	文献[5]的 NC 值
裁剪 1/8	最坏 0.875*	0.9477
裁剪 1/4	最坏 0.75*	0.7908
3×3 均值滤波	0.9331	0.9656
5×5 均值滤波	0.8317	0.8765
椒盐噪声 0.2%	0.9978	0.9774
椒盐噪声 0.5%	0.9859	0.9570
JPEG 压缩因子:40	1.0000	0.9720
JPEG 压缩因子:20	0.9913	0.8840

因为算法的抗裁剪攻击能力与被裁剪区域的位置有关，所以此处表 8 中测定的是没有发生区域替换的情况下的最坏 NC 值(以"*"标记)。在有区域替换的情况下最坏 NC 值由受损最轻的区域决定。最坏情况是指被裁剪区域嵌入的水印信息全部丢失。由表 8 可以看出本文算法在对抗噪声攻击与压缩攻击方面优于文献[5]，这是分区的大小增加了 4 倍而使均值更加稳定的缘故。因此，在分区较大的条件下本文算法对于噪声与压缩攻击效果更好。

结束语　本文提出了一种适用于 Android 智能手机的灰度均值图像水印算法。该算法实现简单、时间复杂度低，适合 Android 平台。实验结果表明，本算法不仅具有良好的透明性，而且对 JPEG 压缩、加噪等攻击具有较强的鲁棒性，同时有较好的抵抗裁剪攻击能力。由于水印的 1,2,3,4 区域一般是相似的，因此可以考虑存储区域之间的差异。由于相似区域的差异很少，甚至有时完全没有或者少得可忽略，此时便可将整个区域当作一个整体来量化以提高区域均值的稳定性，这为今后的研究提供了方向。

参 考 文 献

[1] 张晓强，王蒙蒙，朱贵良. 图像水印算法研究新进展[J]. 计算机工程与科学，2012，34(4)：17-22
Zhang Xiao-qiang, Wang Meng-meng, Zhu Gui-liang. A Novel Survey on the Image Watermarking Algorithms[J]. Computer Engineering & Science, 2012, 34(4): 17-22

[2] Zhu Shao-min, Liu Jian-ming. A novel fragile watermarking scheme for image tamper detection and recovery[J]. Chinese Optics Letters, 2010, 8(7): 661-665

[3] Xiang S, Kim H, Huang J. Invariant image watermarking based on statistical features in the low-frequency domain [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(6): 777-790

[4] 李旭东. 基于图像灰度平均值的数字水印算法[J]. 武汉大学学报，2007，32(6)：556-559
Li Xu-dong. Mean Gray Value Based Image Watermarking Algorithm[J]. Geomatics and Information Science of Wuhan University, 2007, 32(6): 556-559

[5] 王友卫，申铉京，吕颖达，等. 基于 DCT 和 SVD 变换的盲数字水印算法[J]. 计算机工程与应用，2011，41(21)：157-161
Wang You-wei, Shen Xuan-jing, Lv Ying-da, et al. Blind digital watermarking algorithm based on DCT and SVD transform[J]. Computer Engineering and Applications, 2011, 47(21): 157-161

[6] 陈利，何选森，赵天娇. 一种自适应的奇偶量化水印算法[J]. 计算机工程与应用，2011，47(35)：203-205
Chen Li, He Xuan-sen, Zhao Tian-jiao. Adaptive odd-even quantization watermarking algorithm[J]. Computer Engineering and Applications, 2011, 47(35): 203-205

[7] 陈益飞，曹瑞. 混沌理论在水印置乱中的应用[J]. 电子设计工程，2011，19(1)：157-160
Chen Yi-fei, CAO Rui. Application of the chaos theory in the watermarking scrambling[J]. Electronic Design Engineering, 2011, 19(1): 157-160

(上接第 77 页)

[10] Dell'Amico M, Martello S. Bounds for the cardinality constrained $P|C_{max}$ problem [J]. Journal of Scheduling, 2001, 4(3): 123-138

[11] Dell'Amico M, Iori M, Martello S. Heuristic algorithms and scatter search for the cardinality constrained $P|C_{max}$ problem [J]. Journal of Heuristics, 2004, 10(2): 169-204

[12] Dell'Amico M, Iori M, Martello S, et al. Lower bound and heuristic algorithms for the k_i partitioning problem [J]. European Journal of Operational Research, 2006, 171(3): 725-742

[13] He Y, Tan Z, Zhu J, et al. k-Partitioning problems for maximizing the minimum load [J]. Computers and Mathematics with Applications, 2003, 46(10/11): 1671-1681

[14] Bruglieri M, Ehrgott M, Hamacher H W, et al. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints [J]. Discrete Applied Mathematics, 2006, 154(9): 1344-1357

[15] Kellerer H, Woeginger G. A tight bound for 3-partitioning [J]. Discrete Applied Mathematics, 1993, 45(3): 249-259

[16] Kellerer H, Kotov V A. 7/6-approximation algorithm for 3-partitioning and its application to multiprocessor scheduling [J]. INFOR: Information Systems and Operational Research, 1999, 37(1): 48-56

[17] Chen S P, He Y, Lin G H. 3-partitioning for maximizing the minimum load [J]. Journal of Combinatorial Optimization, 2002, 6(1): 67-80

[18] Garey M R, Johnson D S. Computer and Intractability: A Guide to The Theory of NP-Completeness [D]. San Francisco: W. H. Freeman and Company, 1979

[19] Ahuja R K, Magnanti T L, Orlin J B. Network Flows: Theory, Algorithms, and Applications [M]. Prentice Hall, NJ, 1993

[20] Petrank E. The hardness of approximation: gap location [J]. Computational Complexity, 1994, 4(2): 133-157