基于动态阈值的帧复制粘贴篡改检测①

卢贺楠[1,2,4], 黄添强[2,3,4], 林 晶[1,2,4]

[1](福建师范大学 数学与计算机科学学院, 福州 350007)
[2](福建省公共服务大数据挖掘与应用工程技术研究中心, 福州 350007)
[3](福建师范大学 软件学院, 福州 350007)
[4](大数据分析与应用福建省高校工程研究中心, 福州 350007)

摘 要: 帧复制粘贴是一种常见的时域篡改方式, 篡改者采用这种方式来移除视频中某段内容, 如犯罪现场、犯罪证据等. 针对这种篡改, 已经有不少方案被提出, 但是它们有两个缺点: 一采用固定的阈值; 二时间复杂度很高. 大部分被篡改视频会经过再压缩处理, 由于视频压缩基本是有损压缩, 可能会导致固定阈值失去应有作用. 因此, 本文提出一种基于动态阈值的被动取证算法, 来增强算法的鲁棒性, 并且通过引入字典排序算法来缩小帧匹配时的搜索范围, 成功降低了时间复杂度. 文章采用精确度、召回率和平均每帧计算时间来对提出算法的性能进行评估, 结果表明本算法在这三方面都优于其他算法, 而且具有更好的鲁棒性.

关键词: 帧复制粘贴; 视频取证; 时域篡改; 动态阈值; 结构相识性系数

Frame Copy and Paste Tampering Detection Based on Dynamic Threshold

LU He-Nan[1,2,4], HUANG Tian-Qiang[2,3,4], LIN Jing[1,2,4]

[1](School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350007, China)
[2](Fujian Engineering Research Center of Public Service Big Data Mining and Application, Fuzhou 350007, China)
[3](School of Software, Fujian Normal University, Fuzhou 350007, China)
[4](Fujian Provincial University Engineering Research Center of Big Data Analysis and Application, Fuzhou 350007, China)

Abstract: Frame duplication forgery is a very common operation for video tampering in the temporal domain. By removing some frames which contain a crime scene or crime evidence, the forger can change the video content. A lot of solutions have been proposed for detecting this type of tempering operation. However, there are two disadvantages. The first one is fixed threshold; the other is huge computation. In general, frame duplication forgery and re-compression would be performed on a video at the same time. Since re-compression will cause data lost, the fixed thresholds may lose their effects. Therefore, an algorithm based on dynamic threshold is proposed in this paper, which can improve the robustness. Additionally, the proposed method adopts dictionary order algorithm to reduce the search scope of frame matching and the time complexity. Three performance indices: precision, recall and average computation time per frame are employed to evaluate our algorithm. The results demonstrate that the proposed method outperforms the existing methods in terms of precision, recall and computation time.

Key words: frame duplication; digital video forensics; temporal forgery; dynamic threshold; structural similarity index

视频已经广泛应用于新闻报道、安全监控、法庭证据等各个领域, 与图片相比, 视频传递信息更直观、生动. 然而, 近年来视频编辑技术发展迅速, 编辑软件种类多样并且容易使用, 非专业人员也能实现视频篡改, 而且不遗留任何人眼可见的篡改痕迹. 数字媒体作为当今信息传播的主要媒介, 如果视频传递的信息不真实, 有可能导致人们做出错误的决定, 甚至会给社会带来不好的影响. 例如虚假的产品商业宣传就会给购物者带来经济损失. 更危险的是当一段篡改后的视频被用于法庭证据时, 会照成严重的后果. 因此,

① 收稿时间:2016-03-14;收到修改稿时间:2016-05-12 [doi:10.15888/j.cnki.csa.005499]

最近十年, 视频内容完整性取证逐渐成为学者们热门领域之一, 旨在研究探索视频的处理历史[1].

随着研究的深入, 许多视频篡改取证方案被提出. Wang 和 Farid 发现一种删帧指纹[2], 并利用它对采用固定 GOP(Group of picture)组的 MPEG 格式视频进行双重压缩检测; 文献[3]改进了 Wang 等人的特征, 构建两种删除指纹, 分别用来对采用固定 GOP 组的视频和动态 GOP 组的视频进行删帧和插帧篡改检测; Stamm M C 等人发现如果视频采用可变的 GOP, 则可以利用 P 帧预测误差序列和其自身平滑处理后做差结果来进行检测[4]; Su 等人采用运动补偿边缘效应(motion compensated edge artifact, MCEA)来检测删帧篡改[5]; 而 Dong 等人探索相邻两个 P 帧 MCEA 的差异, 检测视频帧序列经过双重压缩后其傅里叶变换域中是否存在波峰, 进而判断视频是否经过篡改[6]. 林等人提出一种基于纹理谱的时域篡改检测算法, 通过计算纹理谱序列之间的相关系数, 根据给定阈值定位视频篡改位置[7]. 为实现视频局部区域复制粘贴篡改检测, Hsu Chih-Chung 等人提出一种基于模式噪声的取证方案[8].

帧复制粘贴篡改是一种时域篡改方式, 针对这种篡改取证的研究已经有不少成果[9-12], Wang 等人[9]把帧序列划分成若干个重叠的子序列, 利用存在复制粘贴关系的子序列的空间域和时间域的相似性, 来进行篡改取证, 但是由于帧的数量, 特征提取时计算量很大. Subramanyam 等人[10]利用 HOG(Histogram of Oriented Gradients)特征进行匹配, 可是高维的 HOG 数据导致算法很复杂. Lin 等人[11]先利用时域相似性来进行粗检测, 找到可疑的帧; 然后利用空间的相似性从可疑帧查找存在复制粘贴关系的帧序列. 文献[9-11]采用的特征比较复杂, 导致算法计算量很大, 而且精确度不够理想. 因此, Li 等人[12]提出一种基于结构相似度算法, 以测量的帧间相似度系数为指标来进行篡改检测, 该方法提高了检测精度, 同时降低了时间复杂度. 但 Li 采用的是固定阈值, 对再压缩操作比较敏感.

本文提出的算法为每一帧设定一个动态阈值, 该阈值随着当前帧和它前后两个相邻帧相似度的变化而变化, 因此算法对再压缩有很好的鲁棒性. 此外算法引入字典排序算法, 缩小帧匹配过程中搜索范围, 有效的降低时间复杂度.

1 结构相似度系数

在传输和解压过程中, 数字图像不可避免的发生不同程度的扭曲、失真现象, 不仅会影响视觉欣赏而且影响图像传递信息的真实性. 为了得到较好质量的图片, 需要对接收的图像进行质量评估, 来优化传输系统. 图像的质量评估工作过于繁重, 由观察员来进行主观评价代价高而且浪费时间. Wang Zhou 等人[13]根据人眼视觉特征提出结构相似度系数(structural similarity (SSIM) index)作为图像质量评估的指标. 该指标性与主观评价有很好的一致性, 而且计算简单, 已经广泛应用于图像处理等领域.

SSIM 系数是衡量两幅图像相似度的指标, 其值越大表示图像越相似, 最大值为 1. SSIM 系数的计算分为三个部分: 亮度比较函数 $l(x,y)$ 的计算, 如式(1), 对比度比较函数 $c(x,y)$ 的计算, 如式(2), 以及结构比较函数 $s(x,y)$ 的计算, 如式(3). 三者合并得到两幅图像全局相似度 SSIM 系数. 假设 x、y 分别代表原始图像和接收的图像, 两者的 SSIM 系数计算如下:

$$l(x,y)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1} \tag{1}$$

$$c(x,y)=\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2} \tag{2}$$

$$s(x,y)=\frac{\delta_{xy}+C_3}{\sigma_x\sigma_y+C_3} \tag{3}$$

$$SSIM(x,y)=[l(x,y)]^{\alpha}\bullet[c(x,y)]^{\beta}\bullet[s(x,y)]^{\gamma} \tag{4}$$

其中, μ_x、σ_x 分别是 x 图像的均值和方差, μ_y、σ_y 分别是 y 图像的均值和方差, δ_{xy} 表示两幅图像的协方差. C_1、C_2、C_3 是用来防止分母约等于 0 的情况下产生不可预知的事件的发生, 即使计算结果更稳定. 一般情况, $C_1=(K_1L)^2$, $C_2=(K_2L)^2$, $C_3=C_2/2$, K_1、K_2 是非常小的常数, L 为像素最大值. α、β、γ, 是用来调整三个分量相对重要性的参数. 为使计算更简单, Wang Zhou 等人设定 $\alpha=\beta=\gamma=1$, 得到一个特殊的 SSIM 系数计算式(5).

$$SSIM(x,y)=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \tag{5}$$

在实际应用中, 图片被分割成若干个 11╳11 像素大小的窗口, 用式(5)计算两幅图像中所有对应窗口的

SSIM 系数, 取平均得到平均的 SSIM 系数, 定义如式(6), 其中 N 是局部窗口总数, i 是局部窗口序号.

$$MSSIM = \frac{1}{N}\sum_{i=1}^{N} SSIM(x_i, y_i) \quad (6)$$

2 篡改检测算法

一个视频经过帧复制粘贴篡改, 它的帧序列中至少包含一个复制粘贴帧序列片段, 而两个帧序列片段之间如果存在复制粘贴关系, 那么它们的帧之间相似度会很高. 基于这个思想, 所提算法用式(6)计算两帧之间的 MSSIM 系数, 并以此结果作为判定两帧之间存在复制粘贴关系的主要依据. 算法的第一步是为每一帧设定一个阈值, 与帧匹配过程中计算的 MSSIM 系数作比较, 进而确定两帧是否存在复制粘贴关系. 本文算法与其他算法相比, 主要优势在于设定的阈值是动态的, 对视频再压缩有很强的鲁棒性. 阈值设定后, 通过与计算得到的两帧之间的 MSSIM 系数, 把帧分为存在复制粘贴关系的和不存在复制粘贴关系的帧对. 算法最后一步是剔除误检的帧对, 同时找到漏检的帧对, 然后把它们按视频中的顺序排列, 并利用视频内容连续性的特点, 给出原始序列和复制粘贴序列. 由上述可知, 本文算法可以分成三部分, 其框图如图 1 所示.

图 1　算法框图

2.1 阈值设定

根据视像编码标准[14]可知, 视频中每一帧在编码过程中都会经过 DCT(Discrete Cosine Transform, DCT)变换和量化操作, 量化操作使得 DCT 系数由浮点数转变为整数, 两者之间的差值被称为量化误差[15,16]. 重构帧图像时又有反 DCT 变换和反量化操作, 反量化使得反 DCT 变换系数转变为整数, 同时把不在[0,255]范围内的系数强制转为 0 或 255[16]. 帧图像编码和重构过程中都有数据丢失, 整个过程不可逆, 而且这种现象在全部视像编码过程中都存在, 从而导致两帧间的相似度随着两帧间隔距离的增加而降低. 为了展现这个结论, 我们来做一个实验. 首先, 选取一个原始视频, 视频背景静止并且没有任何运动目标, 测量第 90 帧和视频中所有帧之间的 MSSIM 系数, 结果如图 2. 然后对该视频进行帧复制粘贴篡改, 为方便对比, 我们把 83~120 的帧序列复制粘贴到 196~233 的位置, 篡改后的视频帧序列中第 90 帧和第 203 帧之间存在复制粘贴关系, 同样测量第 90 帧和篡改后视频的所有帧之间的 MSSIM 系数, 结果如图 3.

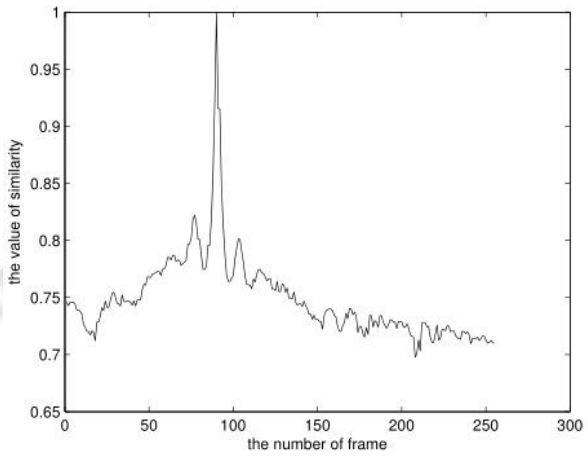

图 2　原始视频帧序列第 90 帧和所有帧之间的 MSSIM 系数变换趋势

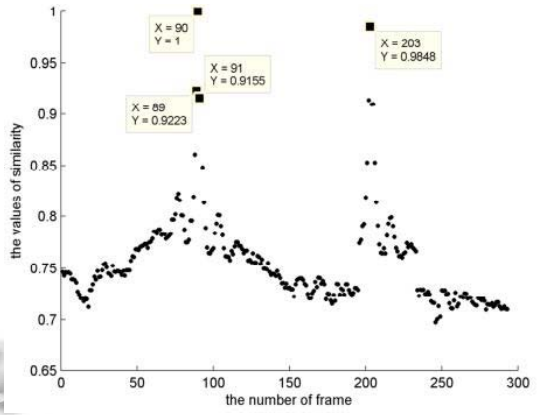

图 3　篡改视频帧序列第 90 帧和所有帧之间的 MSSIM 系数变换趋势

图 2 展示的是 MSSIM 的值随着两帧之间距离的增加的变换趋势, 我们发现随着间隔距离的增加, 两帧间相似度整体表现为下降的趋势. 虽然有一些轻微的波动, 但总体是下降的趋势, 并且下降速度先快后慢. 由图 3 可知存在复制粘贴关系的帧之间的相似度很高, 同时从中可以发现第 90 帧和第 203 帧之间的 MSSIM 系数并没有达到 1, 这是因为视频篡改后进行了二次压缩, 使得两帧之间的相似度降低了.

此外, 第 90 帧和第 203 帧之间的 MSSIM 系数高于第 90 帧和它的两个相邻帧(89 帧和 91 帧)的 MSSIM

系数. 为证明这一现象在视频帧序列中普遍存在, 50 段不同视频的帧序列被用来做进一步的实验. 假设帧 P_i 和帧 P_j 存在复制粘贴关系, 帧 P_i 是原始帧 V_1 是帧 P_i 和帧 P_j 之间的 MSSIM 系数, V_2 是帧 P_i 和帧 P_{i+1} 之间的 MSSIM 系数, V_3 是帧 P_i 和帧 P_{i-1} 之间的 MSSIM 系数. 分别计算每个视频相应的 V_1、V_2、V_3, 部分结果如表 1. 用事件 A 表示: V_1 是最大的; 用事件 B 表示: V_1 介于 V_2 和 V_3; 用事件 C 表示: V_1 是最小的.

表 1 部分实验结果

	视频 1	视频 2	视频 3	视频 4	视频 5	视频 6	视频 7	……
V_3	0.9889	0.9328	0.9819	0.9887	0.9951	0.9481	0.9741	……
V_2	0.9882	0.9281	0.9218	0.9895	0.9953	0.9442	0.9787	……
V_1	0.9986	0.9796	0.9779	1	0.9975	0.9951	0.9902	……
结果	$V_1 > V_3$ $V_1 > V_2$	$V_1 > V_3$ $V_1 > V_2$	$V_1 < V_3$ $V_1 > V_2$	$V_1 > V_3$ $V_1 > V_2$	$V_1 > V_3$ $V_1 > V_2$	$V_1 > V_3$ $V_1 > V_2$	$V_1 > V_3$ $V_1 > V_2$	

从表中数据可知: 事件 A 经常发生, 事件 B 偶尔发生, 事件 C 一次也没发生. 因为我们做实验所用视频数目很少, 不能因此而得出事件 C 不可能发生这一结论, 但可以得出事件 C 发生的概率很小. 因此可以取 V_2、V_3 中较小的一个作为帧 P_i 的阈值来寻找与它有复制粘贴关系的帧.

在许多篡改取证算法中, 设定阈值是一个关键问题, 也是一个难点. Lin 等人 [11]和 Li 等人[12]采用固定参数作为判定阈值. 由上文分析知: 二次压缩操作会降低复制粘贴帧之间的相似度, 继而影响设定的固定阈值的效果, 使得算法的精确度降低. 图 4 展示一个篡改视频经过不同比率的二次压缩后, 所有相邻帧的 MSSIM 系数平均值的变化趋势, 二次压缩率为 0%、10%、20%、……、90%, 在图中的编号依次为 1, 2, 3, ……, 10. 图 4 表明相邻帧相似度值会随着压缩率的增加而降低. 因此, 采用固定阈值的算法针对二次压缩的鲁棒性较弱.

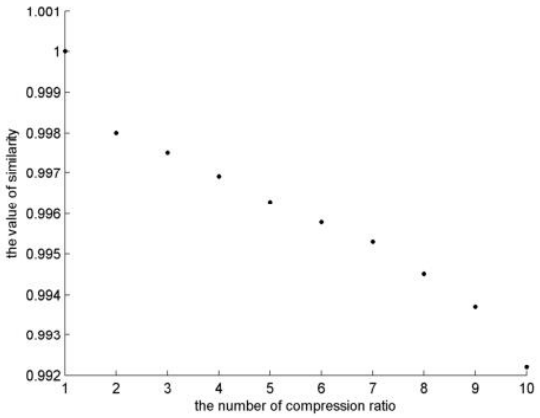

图 4 帧间相似度随着压缩率增加的变化趋势

针对这个问题, 本文给出一种动态阈值设定方法. 根据上述实验可知: 可以取当前帧与其前后两个相邻帧之间 MSSIM 系数中较小的一个为当前帧的判定阈值, 即为每一帧设定一个阈值, 且阈值由该帧本身和它相邻的帧决定. 详细步骤如下.

T1. 第一步, 视频转换为帧序列: F_1、F_2、……、F_i、……、F_n, n 是帧的总数, 然后把每一帧转换为灰度图像, 得到灰度图像序列: P_1、P_2、……、P_i、……、P_n.

T2. 第二步, 为每一帧设定一个判定阈值. 用 T_i 表示帧 P_i 的阈值. 分别计算 P_i 和 P_{i+1} 之间的 MSSIM 系数, P_i 和 P_{i-1} 之间的 MSSIM 系数, 取两者中较小的为 P_i 的判定阈值.

T3. 最后, 计算所有阈值的平均值 E 和方差 D, 根据式(7)更新每一帧的阈值.

$$T_i = \begin{cases} T_i & T_i \geq E - D \\ E & T_i < E - D \end{cases} \tag{7}$$

步骤 T3 中更新阈值操作很关键, 不能省略. 因为对某一段视频进行检测之前, 我们无法预测视频经过什么方式的处理, 也不清楚视频内容以及场景的变换规律, 但是我们要尽可能的把对算法有影响的因素都考虑到. 例如, 视频中的场景快速变换; 一个快速移动目标进入视频; 有复制粘贴篡改操作等. 我们用图 5 来解释刚给出的三种状况带来的问题. 假设第一种或第二种状况发生在图中的第三帧, 那么从第二帧图片到第三帧图片, 两者的内容发生了明显的变化, 两帧的相似度会急剧下降, 即两帧的 MSSIM 系数很小. 如果 3、4、5 帧是从视频中其他位置复制粘贴得到的, 那么第二帧和第三帧的 MSSIM 系数也很小, 因为这时

两者之间的内容也存在明显的差异. 考虑到这些状况可能会发生, 为剔除步骤 T2 中可能存在的较小阈值, 必须用式(7)对阈值进行更新. 因为某帧阈值越小, 则找到的与该帧存在复制粘贴关系的可疑帧会越多(原因将在下一节说明), 从而增加算法计算量, 而且使得后处理更加困难, 降低了算法的效率.

图 5　部分视频帧序列

2.2 帧匹配

在 Lin 等人和 Li 等人提出的方案中, 都先把整个视频帧序列分割成若干个重叠的子序列, 再进行帧间相似度测量. 子序列分割方式如图 6 所示. 假设帧总数为 n, 子序列的长度为 m, 则得到的子序列共有(n-m+1)个. 每一个子序列都和其他子序列进行相似度测量, 即使每两个子序列只进行一次计算, 也需要进行(n-m-1)(n-m)/2 次, 随着帧数目的增加, 计算量会快速增加.

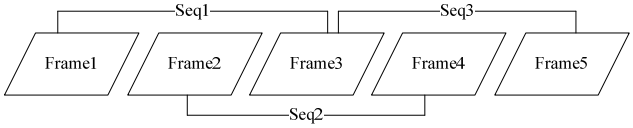

图 6　子序列分割方式

为了降低计算量, 本算法引入字典排序算法. 字典排序能有效的把相似度较高图片排序放在距离较近的地方, 而且简洁迅速. 字典排序对提取的特征较为敏感, 本方案采用图像 DCT 变换后系数作为特征, 为减少特征向量的维数, 只取 DCT 系数 Zig-Zag 排序后的前十个系数作为每一帧的特征向量. 详细步骤如下:

T1. 提取每一帧图像的特征向量. 对灰度图像序列: P_1, P_2 , …, P_i, …, P_n 中的每一帧进行 DCT 变换, 得到 n 个 DCT 系数矩阵, 每个矩阵中的系数经过 Zig-zag 排序后取前十个系数作为对应帧的特征向量. 定义 M=[V1,V2,…,Vi,…,Vn], 其中 Vi 表示帧 P_i 的特征向量, n 是帧的总数.

T2. 对矩阵 M 进行字典排序得到矩阵 M_{order}.

T3. 根据字典排序的结果, 来搜索可能存在复制粘贴关系的帧对, 这个过程称为帧匹配. 假设帧 P_i 在矩阵 M_{order} 中的第 j 行, 那么寻找与帧 P_i 存在复制粘贴关系的帧时, 只需要检测在 $[j-10, j)\cup(j, j+10]$ 范围内的帧. 计算帧 P_i 和在 $[j-10, j)\cup(j, j+10]$ 范围内所有帧之间的 MSSIM 系数, 并与 P_i 的阈值 T_i 进行比较. 如果帧 P_k 在 $[j-10, j)\cup(j, j+10]$ 范围内, 且 P_i 和 P_k 两者之间的 MSSIM 系数 T_{ik} 大于阈值 T_i, 则把帧对 (P_i, P_k) 放入集合 C 中.

引入字典排序后, 缩小了帧匹配时搜寻的范围, 每一帧在匹配的过程中只需要进行 20 次相似度测量, n 帧需要计算 $20n$ 次, 与 Lin 和 Li 等人的方案相比, 所提算法有效的减少计算次数, 提高了方案的效率.

2.3 后处理

在 2.2 节, 匹配结果中可能会发现有一帧对应多帧的情况, 这是问题(一), 导致这种现象的原因有三个, 一是这一帧确实多次复制粘贴, 二是上文中讨论的事件 B 和事件 C 至少有一个出现了, 三是特殊情况, 这两帧相似度确实很高, 但它们之间不存在复制粘贴关系; 此外, 在 2.2 节中采用字典排序算法来缩小搜索范围, 虽然提高了方案效率, 但是同时带来一个问题. 因为字典排序不能保证把所有相似的帧都排序放在相距很近的位置, 即有可能存在一帧与 P_i 有复制粘贴关系, 但是经过字典排序后, 这一帧不在 $[j-10, j)\cup(j, j+10]$, 导致漏检, 这是问题(二).

为了解决问题(一), 我们设计了一个分类器, 分类器由两部分组成, 第一部分是用来消除由事件 B 和事件 C 发生造成的误检测, 第二部分是用来排除特殊情况造成的误检测. 第二部分的设计原理是利用了人眼视觉特征, 因为必须连续复制粘贴 15 帧, 人眼才能感觉到视频内容的变化, 所以可以认为复制粘贴的帧序列长度至少为 15, 如果检测到的帧序列长度小于 15, 则判定是特殊现象带来的误差. 问题(二)是通过检测到的帧, 按帧序号一帧一帧向外扩展解决的, 具体步骤如下.

T1. 分类器第一部分定义如式(8), 其中 i 、j 表示帧序号. 集合 C 中所以帧对都要经分类器处理, 如果 $\phi(P_i, P_j)=0$, 则把帧对 (P_i, P_j) 从集合 C 中移除.

$$\phi(P_i,P_j)=\begin{cases}1 & |\mathrm{i}-\mathrm{j}|\geq 15\\ 0 & |\mathrm{i}-\mathrm{j}|<15\end{cases}\quad (P_i,P_j)\in \mathrm{C} \tag{8}$$

T2. 集合 C 中的帧对经过分类器处理后, 从 C 中任取一帧对 (P_p, P_q), 把它放入集合 S 中, 并从 C 中移除. 记录下当前这两帧的序号, 令 $a=p$, $b=q$.

T3. 检测帧对 $(\mathrm{P}_{p+1}, \mathrm{P}_{q+1})$ 是否存在与集合 C 中, 如

果存在, 则把它从 C 中移除, 放入 S 中, 令 $p=p+1$, $q=q+1$, 跳转至 T3; 如果不存在, 跳转至 T4.

T4. 计算帧 P_{p+1} 和帧 P_{q+1} 两者之间的 SSIM 系数, 如果它们的 SSIM 系数大于阈值 T_{p+1} (P_{p+1} 的阈值)或者 T_{q+1} (P_{q+1} 的阈值)其中一个, 则令 $p=p+1$, $q=q+1$, 并跳转至 T3; 否则, 把帧对 (P_{p+1},P_{q+1}) 从集合 S 中删除, 令 $p=a$, $q=b$, 然后跳转至 T5.

T5. 检测帧对 (P_{p-1},P_{q-1}) 是否存在于集合 C 中, 如果存在, 则把它从 C 中移除, 放入 S 中, 令 $p=p-1$, $q=q-1$, 跳转至 T5; 如果不存在, 跳转至 T6.

T6. 计算帧 P_{p-1} 和帧 P_{q-1} 之间的 SSIM 系数, 如果它们的 SSIM 系数大于阈值 T_{p-1} 或者 T_{q-1}, 则令 $p=p-1$, $q=q-1$, 跳转至 T5; 否则, 跳转至 T7.

T7. 统计集合 S 中帧对的总数, 记 *Num*, 分类器的第二部分定义如公式(9). 如果 $U(\mathrm{S})=1$, 则认为当前集合 S 中可以组成两段存在复制粘贴关系的帧序列. 假设帧两段帧序列分别为 P_i, P_{i+1}, … , P_{i+m} 和 P_j,P_{j+1}, …, P_{j+m}. 根据复制粘贴序列和其粘贴位置连续性较弱的特点来判定原始帧序列和复制粘贴帧序列, 计算公式如 (10)~(13). 如果 $MSSIM_i+MSSIM_{i+m}>MSSIM_j+MSSIM_{j+m}$, 则 P_i,P_{i+1}, …, P_{i+m} 是原始序列, P_j,P_{j+1}, …, P_{j+m} 是复制粘贴帧序列; 否则, $P_j,P_{j+1},\ldots,P_{j+m}$ 是原始帧序列, P_i, P_{i+1}, …, P_{i+m} 是复制粘贴序列. 如果 $U(\mathrm{S})=0$, 则清空集合 S, 转至 T8.

$$U(S)=\begin{cases}1 & Num\geq 15\\ 0 & Num<15\end{cases} \tag{9}$$

$$MSSIM_i=SSIM(P_i,P_{i-1}) \tag{10}$$

$$MSSIM_{i+m}=SSIM(P_{i+m},P_{i+m+1}) \tag{11}$$

$$MSSIM_j=SSIM(P_j,P_{j-1}) \tag{12}$$

$$MSSIM_{j+m}=SSIM(P_{j+m},P_{j+m+1}) \tag{13}$$

T8. 重复步骤 T2~T7, 直到 C 为空.

3 实验结果及分析

为完成实验, 我们选取 15 段视频, 其中包括 5 段分辨率为 640×480 的视频、5 段分辨率为 720×576 的视频以及 5 段分辨率为 1920×1080 的视频. 篡改视频时, 复制的帧序列长度从 15~150 不等, 使用 Adobe Premiere Pro CS5 来篡改视频, 使用 Free Video compression 来对视频进行压缩, 电脑配置了 3.20GHz 的 CPU, 运行软件为 Matlab2012a.

3.1 算法评估参数

为评估本方案的性能, 我们统计了算法的精确度(precision)、召回率(recall)和计算时间三种参数. 精确度和召回率是比较传统的评估参数, 已经广泛应用于图片相关检查、分类研究等领域. 精确度越高表明算法的检测越准确; 召回率越高表明算法的误检率很低. 精确度和召回率的定义分别如式(14)(15), N_c 表示正确检测结果的数目, N_f 是错误检测结果的数目, N_m 是漏检帧的数目.

$$precision=\frac{N_c}{N_c+N_f} \tag{14}$$

$$recall=\frac{N_c}{N_c+N_m} \tag{15}$$

此外, 计算时间也被用来作为算法评估的一个重要参数, 本文采用是平均每一帧所花费的时间.

3.2 实验结果

表 2 给出用于实验的 15 段视频的详细信息. 表 3 给出本文算法检测的结果. 为了证明算法对二次压缩有很强的鲁棒性, 我们对篡改视频进行压缩, 压缩率分别为 20%、50%以及 90%, 然后用所提算法对压缩后的视频依次检测, 结果分别在表 4、表 5 和表 6 中展示, 从实验结果可以知: 无论篡改视频是否经过压缩, 本文算法都能精确定位篡改帧序列.

表 3 结果显示当篡改者没有对篡改视频进行二次压缩时, 算法检测正确率达到 100%, 无一帧漏检. 原因是如果视频篡改后不经过压缩, 那么存在复制粘贴关系的两帧之间的相似度值是 1, 而根据给出的阈值自适应设定方法知, 阈值一定小于 1, 这样自然就不会出现漏检现象.

表 2 测试视频

测试视频	帧总数	分辨率	原始帧	复制粘贴帧
视频 1	443	640×480	没有篡改	
视频 2	851	640×480	510-659	224-373
视频 3	403	640×480	65-128	258-321
视频 4	264	640×480	52-99	162-209
视频 5	549	640×480	102-194	358-450
视频 6	659	720×576	没有篡改	
视频 7	721	720×576	28-88	414-474
视频 8	1169	720×576	860-936	317-393
视频 9	621	720×576	164-220	342-398
视频 10	1163	720×576	662-788	245-371

视频 11	321	1920×1080	没有篡改	
视频 12	300	1920×1080	217-256	101-140
视频 13	283	1920×1080	31-45	150-164
视频 14	575	1920×1080	123-201	363-441
视频 15	293	1920×1080	83-120	196-233

表 3　本文算法对没有经过二次压缩视频检查结果

测试视频	分辨率	原始帧	复制粘贴帧	计算时间(s/每帧)
视频 1	640×480	没有篡改		0.305
视频 2		510-659	224-373	0.361
视频 3		65-128	258-321	0.358
视频 4		52-99	162-209	0.318
视频 5		102-194	358-450	0.386
视频 6	720×576	没有篡改		0.468
视频 7		28-88	414-474	0.486
视频 8		860-936	317-393	0.551
视频 9		164-220	342-398	0.476
视频 10		662-788	245-371	0.507
视频 11	1920×1080	没有篡改		1.835
视频 12		217-256	101-140	1.970
视频 13		31-45	150-164	1.982
视频 14		123-201	363-441	1.798
视频 15		83-120	196-233	1.979

对比表 3 和表 4 发现, 当视频二次压缩率小于 20%时, 所提算法几乎不受影响. 而由表 5、表 6 可得, 当压缩率达到 50%时, 结果偶尔出现漏检现象, 如视频 9 的帧对(164,342)、视频 14 的帧对(123,363)未检测到. 当压缩率达到 90%时, 漏检现象就很普遍了, 如视频 3、5、7、9、10、13、14、15 都出现漏检情况, 但是我们发现漏检的都是篡改帧序列的开始 1-2 帧或者后面 1-2 帧, 这不影响算法对复制粘贴序列的定位. 此外, 本文算法对未篡改的视频判定完全正确, 而且只有表 6 中视频 5 的帧对(195,451)是误检测的, 说明所提算法的误检率很低.

表 4　算法对经过二次压缩率为 20%的视频检测结果

测试视频	分辨率	原始帧	复制粘贴帧	计算时间(s/每帧)
视频 1	640×480	没有篡改		0.303
视频 2		510-659	224-373	0.362
视频 3		65-128	258-321	0.355
视频 4		52-99	162-209	0.311
视频 5		102-194	358-450	0.378
视频 6	720×576	没有篡改		0.468
视频 7		28-88	414-474	0.482
视频 8		860-936	317-393	0.559
视频 9		164-220	342-398	0.477
视频 10		662-788	245-371	0.502
视频 11	1920×1080	没有篡改		1.825
视频 12		217-256	101-140	1.971
视频 13		31-45	150-164	1.972
视频 14		123-201	363-441	1.788
视频 15		83-120	196-233	1.984

表 5　本文算法对经过二次压缩率为 50%的视频检测结果

测试视频	分辨率	原始帧	复制粘贴帧	计算时间(s/每帧)
视频 1	640×480	没有篡改		0.315
视频 2		510-659	224-373	0.351
视频 3		65-128	258-321	0.348
视频 4		52-99	162-209	0.321
视频 5		102-194	358-450	0.385
视频 6	720×576	没有篡改		0.454
视频 7		28-88	414-474	0.481
视频 8		860-936	317-393	0.546
视频 9		165-220	343-398	0.469
视频 10		662-788	245-371	0.509
视频 11	1920×1080	没有篡改		1.825
视频 12		217-256	101-140	1.968
视频 13		31-45	150-164	1.980
视频 14		124-200	364-440	1.788
视频 15		83-120	196-233	1.969

表 6　本文算法对经过二次压缩率为 90%的视频检测结果

测试视频	分辨率	原始帧	复制粘贴帧	计算时间(s/每帧)
视频 1	640×480	没有篡改		0.298
视频 2		510-659	224-373	0.354
视频 3		67-128	260-321	0.360
视频 4		52-99	162-209	0.314
视频 5		103-195	359-451	0.376
视频 6	720×576	没有篡改		0.464
视频 7		29-88	415-474	0.479
视频 8		860-936	317-393	0.548
视频 9		165-220	343-398	0.477
视频 10		663-788	246-371	0.506
视频 11	1920×1080	没有篡改		1.834
视频 12		217-256	101-140	1.967
视频 13		32-45	151-164	1.988
视频 14		124-201	364-441	1.791
视频 15		84-120	197-233	1.969

3.3 算法对比

综上所述, 本文提出的算法动态设定阈值, 增强了算法的鲁棒性, 同时采用字典排序算法, 有效的降低算法的计算量. 为说明算法的有效性, 将从准确度、召回率和计算时间三方面与文献[12]所提算法进行比较, 结果如表 7.

表 7 准确度、召回率和计算时间比较

	算法	准确度	召回率	计算时间(s/每帧)
视频没有经过	文献[12]	0.981	1	6.218
二次压缩	本文算法	1	1	0.919
视频二次	文献[12]	0.921	0.981	6.337
压缩率为 20%	本文算法	1	1	0.920
视频二次	文献[12]	0.204	0.284	6.511
压缩率为 50%	本文算法	1	0.993	0.924
视频二次	文献[12]	0	0	6.251
压缩率为 90%	本文算法	0.991	0.984	0.899

由表 7 可得, 文献[12]给出的算法随着二次压缩率的增加, 算法准确度和召回率都快速降低, 而无论视频是否经过二次压缩, 本文所提算法的准确度和召回率都没有太大的波动, 并且都高于文献[12]的算法. 结果证明本文算法对压缩有很强的鲁棒性.

表 7 最后一列给出每帧平均所花费的时间, 平均时间计算方法: 首先, 分别计算分辨率为 640×480, 720×576, 1920×1080 的视频每帧所用时间, 然后三者取平均. 相比文献[12]的方法, 所提算法有效的降低了时间复杂度.

4 结语

本文提出一种新的帧复制粘贴篡改盲取证算法, 该算法为每一帧设定一个阈值, 而且阈值是由当前帧本身和它前后相邻帧 MSSIM 系数决定, 随着 MSSIM 系数的变化而动态变化. 动态阈值使得算法能很好的应对不同强度的压缩, 增强了算法的鲁棒性. 而且所提算法引入了字典排序, 字典排序简单有效, 通过缩小帧匹配过程中的搜索范围, 减少相似度计算的次数, 提高算法效率. 实验结果表明所提算法优于文献[12]中的算法.

相比已经存在的算法, 所提算法时间效率有了较大提升, 但是随着视频分辨率的增加, 平均每帧所用时间也大幅增加, 为应对视频分辨率的增长, 研究并发现一个更简单、更有效的新特征来进行视频篡改检测是我们未来工作的主要目标.

参考文献

1 Milani S, Fontani M, Bestagini P, et al. An overview on video forensics. APSIPA Trans. on Signal and Information Processing, 2012, 1: e2.

2 Wang W, Farid H. Exposing digital forgeries in video by detecting double quantization. Proc. of the 11th ACM Workshop on Multimedia and Security, ACM MM&SEC. New York. ACM. 2009. 39–48.

3 Stamm MC, Lin W, Liu KJ. Temporal forensics and anti-forensics for motion compensated video. IEEE Trans. on Information Forensics and Security, 2012, 7(4): 1315–1329.

4 Stamm MC, Wu M, Liu KJR. Information forensics: An overview of the first decade. Access, IEEE, 2013,1: 167–200.

5 Su Y, Zhang J, Liu J. Exposing digital video forgery by detecting motion-compensated edge artifact. International Conference on Computational Intelligence and Software Engineering. Wuhan, China. IEEE. 2009.

6 Dong Q, Yang G, Zhu N. A MCEA based passive forensics scheme for detecting frame-based video tampering. Digital Investigation, 2012, 9(2): 151–159.

7 林新棋,林云枚,林志新,等.基于纹理谱的视频帧间篡改检测.计算工程,2015,41(11):314–321.

8 Chih C, Htzu YH, Lin C, et al. Video forgery detection using correlation of noise residue. Multimedia Signal Processing, 2008 IEEE 10th Workshop on. Cairns. IEEE. 2008. 170–174.

9 Wang W, Hany F. Exposing digital forgeries in video by

detecting duplication. Proc. of the 9th Workshop on Multimedia & Security. New York. ACM. 2007. 35–42.

10 Subramanyam AV, Emmanuel S. Video forgery detection using HOG features and compression properties. 14th International Workshop on Multimedia Signal Processing (MMSP). Banff. IEEE. 2012. 89–94.

11 Lin G, Chang J. Detection of frame duplication forgery in videos based on spatial and temporal analysis. International Journal of Pattern Recognition and Artificial Intelligence, 2012, 26(7): 1250017.

12 Li F, Huang T. Video copy-move forgery detection and localization based on structural similarity. Proc. of the 3rd International Conference on Multimedia Technology, ICMT 2013. Springer Berlin Heidelberg. 2014. 63–76.

13 Wang Z, Bovik AC, Sheikh HR, et al. Image quality assessment: From error visibility to structural similarity. IEEE Trans. on Image Processing, 2004, 13(4): 600–612.

14 林福宗.多媒体技术基础.第 3 版.北京:清华大学出版社,2011.

15 Huang F, Huang J, Shi Y. Detecting double JPEG compression with the same quantization matrix. IEEE Trans. on Information Forensics and Security, 2010, 5(4): 848–856.

16 Yang J, Jin X, Zhu G, et al. An effective method for detecting double JPEG compression with the same quantization matrix. IEEE Trans. on Information Forensics and Security, 2014, 9(11): 1933–1942.