(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111507260 B
(45)授权公告日 2022.08.05

(21)申请号 202010304043.4

(22)申请日 2020.04.17

(65)同一申请的已公布的文献号
申请公布号 CN 111507260 A

(43)申请公布日 2020.08.07

(73)专利权人 重庆邮电大学
地址 400000 重庆市南岸区南山街道崇文路2号
专利权人 重庆小犀智能科技有限公司

(72)发明人 雒江涛 吴悦 胡钢 刘伟

(74)专利代理机构 成都行之专利代理事务所(普通合伙) 51220
专利代理师 李朝虎

(51)Int.Cl.
G06V 20/40(2022.01)
G06V 10/74(2022.01)
G06K 9/62(2022.01)

(56)对比文件
CN 110427895 A,2019.11.08
CN 110889011 A,2020.03.17
CN 109445834 A,2019.03.08
CN 103699691 A,2014.04.02
CN 110083740 A,2019.08.02
CN 104050247 A,2014.09.17
CN 103257992 A,2013.08.21
CN 110309446 A,2019.10.08
US 2017185671 A1,2017.06.29
US 2018308019 A1,2018.10.25
US 2015186471 A1,2015.07.02
CN 101853486 A,2010.10.06
CN 110599486 A,2019.12.20
陈春玲 等.基于Simhash算法的重复数据删除技术的研究与改进.《南京邮电大学学报(自然科学版)》.2016,第36卷(第03期),85-91.
徐济惠.基于Simhash算法的海量文档反作弊技术研究.《计算机技术与发展》.2014,第24卷(第09期),103-107.
蒋洪.关于鸽巢原理和Ramsey定理的几个结论.《科教文汇(中旬刊)》.2008,(第11期),281.
(续)

审查员 吕亦昕

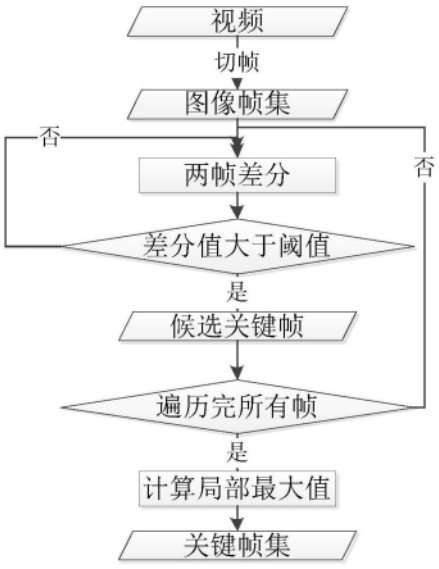
权利要求书3页 说明书9页 附图3页

(54)发明名称

一种视频相似度快速检测方法及检测装置

(57)摘要

本发明公开了在大量视频中鉴别相似视频的快速检测方及检测装置,包括输入视频库中的所有视频文件,提取视频的关键帧;通过计算DCT值得到每一关键帧的指纹;将每一指纹平均分为d块且组合为g个组,建立倒排索引;输入需要检测的视频文件并计算其关键帧的指纹,同样分为n块g个组,查找包含与某组值完全相同的完整指纹;通过计算图像帧相似度,判定视频是否相似;若不相似,则视频数据库中不存在与输入视频相似的视频。本方法利用一种基于分块的快速比较算法,通过视频最基础的关键帧图像的相似度比较进而判断两个视频的相似度,能够实现大多数情况下的大量视频相似度快速检测。



CN 111507260 B

[转续页]

[接上页]

(56)对比文件

Xiaomeng Wu 等.Ultrahigh-Speed TV Commercial Detection, Extraction, and Matching.《IEEE Trans. Circuits Syst. Video Techn》.2013,第23卷(第6期),1054-1069.

1.一种视频相似度快速检测方法,其特征在于,包括:

指纹集处理:

处理参考视频B得到对应于参考视频B的指纹集FB＝{fb_1,...,fb_j,…,fb_n};处理检测视频A得到对应于检测视频A的指纹集FA＝{fa_1,...,fa_i,…,fa_m};

指纹切块分组处理:

对指纹集FB和指纹集FA中的所有指纹做指纹切块分组处理,分别得到大小为n×g的分组指纹集合GB和大小为m×g的分组指纹集合GA;

指纹切块分组处理中,单个指纹做指纹切块分组处理具体为:将指纹集FB中任一指纹fb_j平均切分为d个等长的块,其中d>H,即fb_{j1},fb_{j2},…,fb_{jd},再随机取出其中(d-H)份构成一个新元素,共有$g = C_d^{d-H}$种可能的组合方式;将指纹集FA中任一指纹fa_i平均切分为d个等长的块,即fa_{i1},fa_{i2},…,fa_{id},再随机取出其中(d-H)份构成一个新元素,共有$g = C_d^{d-H}$种可能的组合方式;其中定义两个汉明距离在设定阈值H内的指纹为相似指纹;

建立倒排索引处理:

为分组指纹集合GB创建g个表,相同组合方式的元素存储在同一张表内,然后对该表内的每个元素建立倒排索引,指向包含它的完整指纹;

指纹比对处理:

从分组指纹集合GA中取一个元素C,并逐一在分组指纹集合GB的倒排索引中寻找与C值相等元素:若只存在一个相等的元素E,则直接计算对应指纹的汉明距离,若结果小于或等于H,则两指纹为相似指纹;若存在多个元素E,则选取汉明距离最小的作为相似指纹;若不存在,则跳到下一元素,继续比对;重复上述指纹比对处理,直至找到指纹集FA与指纹集FB中所有相似的指纹,定义相似的指纹对应的关键帧为相似关键帧,以此获得相似关键帧的个数;

相似判定处理:

根据相似关键帧个数计算相似度,设定相似度阈值T_2,如果检测视频A和参考视频B的相似度$S_{A,B}$大于等于阈值T_2,则认为检测视频A与参考视频B相似;否则,则认为两视频不相似。

2.根据权利要求1所述的一种视频相似度快速检测方法,其特征在于,

相似度按下式计算如下:

$$S_{A,B} = \frac{s_{num}}{\min(n,m)} \times 100\%$$

式中,s_{num}表示检测视频A和参考视频B中相似关键帧个数,n和m分别表示参考视频B的关键帧的个数和检测视频A的关键帧的个数。

3.根据权利要求2中所述的一种视频相似度快速检测方法,其特征在于,

所述指纹集处理的具体过程为:分别对参考视频B、检测视频A进行以下操作得到指纹集FB、指纹集FA,

S1、获取关键帧集合:分别获得参考视频B、检测视频A对应的关键帧集合,参考视频B为对应的关键帧集合KB＝{kb_1,kb_2,…,kb_j,…,kb_n},检测视频A为对应的关键帧集合KA＝{ka_1,ka_2,…,ka_i,…,ka_m};

S2、预处理:将每个关键帧图像缩小为预设M×M的尺寸,并将其转换为灰度图;

S3、离散余弦变换处理：对灰度图进行离散余弦变换DCT，只保留结果中左上角的N×N子矩阵$Z_{N\times N}$，N＝M/4；

S4、二值化处理：计算$Z_{N\times N}$所有元素的算数平均值；然后将大于或等于平均值的元素设为1，小于平均值的元素设为0；

S5、计算指纹处理：将二值化处理后的$Z_{N\times N}$按照先行后列的顺序得到一个二进制序列，作为该关键帧的指纹fb_j、指纹fa_i；

S6、重复S2至S5，分别得到指纹集FB＝{fb_1,...,fb_j…,fb_n}、指纹集FA＝{fa_1,...,fa_i…,fa_m}。

4.根据权利要求3所述的一种视频相似度快速检测方法，其特征在于，

所述S1、获取关键帧集合的具体过程为：

步骤S11：将参考视频B、检测视频A进行切帧处理，得到该视频的图像帧集；

步骤S12：遍历图像帧集中的所有图像帧，计算前后两帧差分得到帧间差分强度；

步骤S13：将帧间差分强度为局部最大值的候选关键帧作为关键帧，即得到该视频的关键帧集合为KB＝{kb_1,kb_2,…,kb_j,…,kb_n}，KA＝{ka_1,ka_2,…,ka_i,…,ka_m}。

5.根据权利要求4所述的一种视频相似度快速检测方法，其特征在于，

所述帧间差分强度，如下所示：

对于参考视频B有：Pj＝|b_{j+1}-b_j|；对于检测视频A有：Pi ＝|a_{i+1}-a_i|；

其中Pj表示该视频的第j帧的帧间差分强度，b_j表示参考视频B的图像帧集B＝{b_1,b_2,…,b_j,…,b_n}中的某一帧图像，b_{j+1}表示图像帧集B中b_j的后一帧图像，a_i表示检测视频A的图像帧集A＝{a_1,a_2,…,a_i,…,a_m}中的某一帧图像，a_{i+1}表示图像帧集A中a_i的后一帧图像；然后，将帧间差分强度大于阈值T_1的帧作为一个候选关键帧。

6.根据权利要求3所述的一种视频相似度快速检测方法，其特征在于，

所述离散余弦变换DCT计算如下所示：

$$F(u,v)=c(u)c(v)\sum_{w=0}^{M-1}\sum_{s=0}^{M-1}f(w,s)\cos\left[\frac{(w+0.5)\pi}{M}u\right]\cos\left[\frac{(s+0.5)\pi}{M}v\right]$$；

式中，(u,v)代表着变换变量，f(w,s)是输入的图像，其中(w,s)代表着输入图像中像素的空间位置，且补偿系数为$c(x)=\begin{cases}\sqrt{\frac{1}{M}}, & x=0\\ \sqrt{\frac{2}{M}}, & x\neq 0\end{cases}$，$x=u,v$。

7.视频相似度快速检测装置，其特征在于，包括：

指纹集获取单元：用于处理参考视频B得到对应于参考视频B的指纹集FB＝{fb_1,...,fb_j…,fb_n}；处理检测视频A得到对应于检测视频A的指纹集FA＝{fa_1,...,fa_i…,fa_m}；

指纹切块分组单元：用于对指纹集FB和指纹集FA中的所有指纹做指纹切块分组处理，分别得到大小为n×g的分组指纹集合GB和大小为m×g的分组指纹集合GA，用于指纹切块分组处理中，单个指纹做指纹切块分组处理具体为：将指纹集FB中任一指纹fb_j平均切分为d个等长的块，其中d>H，即fb_{j1},fb_{j2},…,fb_{jd}，再随机取出其中(d-H)份构成一个新元素，共有$g=C_d^{d-H}$种可能的组合方式；将指纹集FA中任一指纹fa_i平均切分为d个等长的块，即fa_{i1},fa_{i2},…,fa_{id}，再随机取出其中(d-H)份构成一个新元素，共有$g=C_d^{d-H}$种可能的组合方

式;其中定义两个汉明距离在设定阈值H内的指纹为相似指纹;

倒排索引建立单元:用于为分组指纹集合GB创建g个表,相同组合方式的元素存储在同一张表内,然后对该表内的每个元素建立倒排索引,指向包含它的完整指纹;

指纹比对单元:用于从分组指纹集合GA中取一个元素C,并逐一在分组指纹集合GB的倒排索引中寻找与C值相等元素:若只存在一个相等的元素E,则直接计算对应指纹的汉明距离,若结果小于或等于H,则两指纹为相似指纹;若存在多个元素E,则选取汉明距离最小的作为相似指纹;若不存在,则跳到下一元素,继续比对;重复上述指纹比对处理,直至找到指纹集FA与指纹集FB中所有相似的指纹,定义相似的指纹对应的关键帧为相似关键帧,以此获得相似关键帧的个数;

相似判定单元:用于根据相似关键帧个数计算相似度,设定相似度阈值T_2,如果检测视频A和参考视频B的相似度$S_{A,B}$大于等于阈值T_2,则认为检测视频A与参考视频B相似;否则,则认为两视频不相似。

8.根据权利要求7所述的视频相似度快速检测装置,其特征在于,

所述指纹集获取单元包括:

关键帧集合获取单元:用于分别获得参考视频B、检测视频A对应的关键帧集合,参考视频B为对应的关键帧集合KB=$\{kb_1,kb_2,\cdots,kb_j,\cdots,kb_n\}$,检测视频A为对应的关键帧集合KA=$\{ka_1,ka_2,\cdots,ka_i,\cdots,ka_m\}$;

指纹计算执行单元:用于对关键帧集合KB、关键帧集合KA执行程序X得到关键帧的指纹fb_j、指纹fa_i,

程序X为:

将每个关键帧图像缩小为预设M×M的尺寸,并将其转换为灰度图;

对灰度图进行离散余弦变换DCT,只保留结果中左上角的N×N子矩阵$Z_{N\times N}$,N=M/4;

计算$Z_{N\times N}$所有元素的算数平均值;然后将大于或等于平均值的元素设为1,小于平均值的元素设为0;

将二值化处理后的$Z_{N\times N}$按照先行后列的顺序得到一个二进制序列,作为该关键帧的指纹fb_j、指纹fa_j。

9.根据权利要求8所述的视频相似度快速检测装置,其特征在于,关键帧集合获取单元包括:

切帧模块:用于将参考视频B、检测视频A进行切帧处理,得到该视频的图像帧集;

帧间差分强度执行模块:遍历图像帧集中的所有图像帧,计算前后两帧差分得到帧间差分强度;

关键帧识别获取模块:用于将帧间差分强度为局部最大值的候选关键帧作为关键帧,即得到该视频的关键帧集合为KB=$\{kb_1,kb_2,\cdots,kb_j,\cdots,kb_n\}$,KA=$\{ka_1,ka_2,\cdots,ka_i,\cdots,ka_m\}$。

一种视频相似度快速检测方法及检测装置

技术领域

[0001] 本发明涉及视频相似识别技术领域，具体为一种视频相似度快速检测方法及检测装置。

背景技术

[0002] 视频相似度的检测是视频侵权鉴别的重要手段。在知识产权相关法规中，未经原创作者允许直接传播或者复制修改视频后嵌在自己的视频中进行网络传播属于侵权行为，但是在互联网高度发达的今天，这种侵权行为却层出不穷，且一直没有合适、高效的检测方法。鉴别视频侵权问题的关键是检测两个视频的相似度。只要能够准确检测两段视频的相似度，则能有效支持侵权的鉴别。

[0003] 现有的相似视频检测方法存在准确性不够、效率低的问题。例如，MD5计算法，能检测出直接转发的视频，但无法检测出篡改后的视频。最长公共子序列(LCS)算法能比较两个视频内嵌字幕的重复度，但仅限于有字幕的视频对比；基于标题、标签和描述内容的检测方式，需要大量的人工辅助才能完成；但由于每个人对视频的理解不同，产生的标签和描述也有所不同，导致最终的鉴别效果做不到客观一致。基于光流的视频复制移动伪造检测算法，虽然能够判断是否存在复制修改行为，但计算复杂度太大，效率太低。

发明内容

[0004] 本发明所要实现的目的是：寻求一种计算复杂度低、效率高的检测方法，从而得到适应大规模视频量的快速检测方法。

[0005] 本发明通过下述技术方案实现：

[0006] 一种视频相似度快速检测方法，包括：

[0007] 指纹集处理：

[0008] 处理参考视频B得到对应于参考视频B的指纹集FB＝{fb_1,...,fb_j,…,fb_n}；处理检测视频A得到对应于检测视频A的指纹集FA＝{fa_1,...,fa_i,…,fa_m}；

[0009] 指纹切块分组处理：

[0010] 对指纹集FB和指纹集FA中的所有指纹做指纹切块分组处理，分别得到大小为n×g的分组指纹集合GB和大小为m×g的分组指纹集合GA；

[0011] 建立倒排索引处理：

[0012] 为分组指纹集合GB创建g个表，相同组合方式的元素存储在同一张表内，然后对该表内的每个元素建立倒排索引，指向包含它的完整指纹；

[0013] 指纹比对处理：

[0014] 从分组指纹集合GA中取一个元素C，并逐一在分组指纹集合GB的倒排索引中寻找与C值相等元素：若只存在一个相等的元素E，则直接计算对应指纹的汉明距离，若结果小于或等于H，则两指纹为相似指纹；若存在多个元素E，则选取汉明距离最小的作为相似指纹；若不存在，则跳到下一元素，继续比对；重复上述指纹比对处理，直至找到指纹集FA与指纹

集FB中所有相似的指纹，定义相似的指纹对应的关键帧为相似关键帧，以此获得相似关键帧的个数；

[0015] 相似判定处理：

[0016] 根据相似关键帧个数计算相似度，设定相似度阈值T_2，如果检测视频A和参考视频B的相似度$S_{A,B}$大于等于阈值T_2，则认为检测视频A与参考视频B相似；否则，则认为两视频不相似。

[0017] 基于上述方案可以看出，本发明的构思是：基于对比视频最基础的图像特征，设计出一种基于分块的视频相似度快速检测方法，能大大提高对侵权视频的检测效率。

[0018] 基于分块的构思，定义两个汉明距离在设定阈值H内的指纹为相似指纹，将一个视频指纹平均切为d个等长的块。根据抽屉原理（例如，把9个苹果放进10个抽屉里，至少有一个空抽屉），两个相似的指纹至少有（d-H）块是完全相等的，可以得出：有（d-H）块完全相等的两指纹有可能相似，但（d-H）块不完全相等的一定不可能相似，从而排除一些不可能相似的指纹。

[0019] 对切分的d个指纹块随机取出（d-H）块，对这些指纹块形成的值构建倒排索引（从值即能找到包含该值的所有的完整指纹）。把被测视频的指纹同样切分、随机取出（d-H）块，去倒排索引里查找这些指纹块形成的值是否存在：存在即说明该完整指纹可能与包含其的完整指纹相似，就把两完整指纹进行对比，汉明距离小于H的即为相似指纹（关键帧）；不存在则说明这两组完整指纹一定不相似，比较下一指纹块形成的值。

[0020] 可以看出，基于上述指纹切块分组处理、建立倒排索引处理、指纹比对处理可以极大的减少计算量。其中，将相同组合方式的元素存储在同一张表内建立倒排索引，这样采用抽屉原理，可以确定一定不相似的数据，使得省略重复处理的过程，从而快速提升计算速度。总的使得本发明的处理速度可以很快。我们经过测试，可以得到以下参数，可以验证上述构思能达到预期提升检测速度的目的。

[0021] 表1测试数据：

[0022]

视频	未采用分块思想计算法用时	本发明采用分块思想

[0023]

	（s）	计算法用时（s）
视频 1	2.501906	0.428806
视频 2	7.477258	1.873162

[0024] 指纹切块分组处理中，单个指纹做指纹切块分组处理具体为：将指纹集FB中任一指纹fb_j平均切分为d个等长的块，，其中d>H，即fb_{j1}，fbj2，…，fb_{jd}，再随机取出其中（d-H）份构成一个新元素，共有$g = C_d^{d-H}$种可能的组合方式；将指纹集FA中任一指纹fa_i平均切分为d个等长的块，即fa_{i1}，fa_{i2}，…，fa_{id}，再随机取出其中（d-H）份构成一个新元素，共有$g = C_d^{d-H}$种可能的组合方式；其中定义两个汉明距离在设定阈值H内的指纹为相似指纹；

[0025] 优选的，相似度按下式计算如下：

[0026] $$S_{A,B} = \frac{s_{num}}{\min(n,m)} \times 100\%$$

[0027] 式中，s_{num}表示检测视频A和参考视频B中相似关键帧个数，n和m分别表示参考视频B的关键帧的个数和检测视频A的关键帧的个数。

[0028] 优选的，所述指纹集处理的具体过程为：分别对参考视频B、检测视频A进行以下操作得到指纹集FB、指纹集FA，

[0029] S1、获取关键帧集合：分别获得参考视频B、检测视频A对应的关键帧集合，参考视频B为对应的关键帧集合KB＝{kb_1,kb_2,…,kb_j,…,kb_n}，检测视频A为对应的关键帧集合KA＝{ka_1,ka_2,…,ka_i,…,ka_m}；

[0030] S2、预处理：将每个关键帧图像缩小为预设M×M的尺寸，并将其转换为灰度图；

[0031] S3、离散余弦变换处理：对灰度图进行离散余弦变换DCT，只保留结果中左上角的N×N子矩阵$Z_{N\times N}$，N＝M/4；

[0032] S4、二值化处理：计算$Z_{N\times N}$所有元素的算数平均值；然后将大于或等于平均值的元素设为1，小于平均值的元素设为0；

[0033] S5、计算指纹处理：将二值化处理后的$Z_{N\times N}$按照先行后列的顺序得到一个二进制序列，作为该关键帧的指纹fb_j、指纹fa_j；

[0034] S6、重复S2至S5，分别得到指纹集FB＝{fb_1,...,fb_j…,fb_n}、指纹集FA＝{fa_1,...,fa_i…,fa_m}。

[0035] 优选的，所述S1、获取关键帧集合的具体过程为：

[0036] 步骤S11：将参考视频B、检测视频A进行切帧处理，得到该视频的图像帧集；

[0037] 步骤S12：遍历图像帧集中的所有图像帧，计算前后两帧差分得到帧间差分强度；

[0038] 步骤S13：将帧间差分强度为局部最大值的候选关键帧作为关键帧，即得到该视频的关键帧集合为KB＝{kb_1,kb_2,…,kb_j,…,kb_n}，KA＝{ka_1,ka_2,…,ka_i,…,ka_m}。

[0039] 优选的，所述帧间差分强度，如下所示：

[0040] 对于参考视频B有：Pj＝|b_{j+1}-b_j|；对于检测视频A有：Pj＝|a_{i+1}-a_i|；

[0041] 其中Pj表示该视频的第j帧的帧间差分强度，b_j表示参考视频B的图像帧集B＝{b_1,b_2,…,b_j,…,b_n}中的某一帧图像，b_{j+1}表示图像帧集B中b_j的后一帧图像，a_i表示检测视频A的图像帧集A＝{a_1,a_2,…,a_i,…,a_m}中的某一帧图像，a_{i+1}表示图像帧集A中a_i的后一帧图像；然后，将帧间差分强度大于阈值T_1的帧作为一个候选关键帧。

[0042] 优选的，所述离散余弦变换DCT计算如下所示：

[0043] $$F(u,v)=c(u)c(v)\sum_{w=0}^{M-1}\sum_{s=0}^{M-1}f(w,s)\cos\left[\frac{(w+0.5)\pi}{M}u\right]\cos\left[\frac{(s+0.5)\pi}{M}v\right]$$；

[0044] 式中，(u,v)代表着变换变量，f(w,s)是输入的图像，其中(w,s)代表着输入图像中像素的空间位置，且补偿系数为$c(x)=\begin{cases}\sqrt{\frac{1}{M}}, & x=0\\ \sqrt{\frac{2}{M}}, & x\neq 0\end{cases}$，$x=u,v$。

[0045] 在大量视频中鉴别相似视频的检测装置，包括：

[0046] 指纹集获取单元：用于处理参考视频B得到对应于参考视频B的指纹集FB＝{fb_1,...,fb_j…,fb_n}；处理检测视频A得到对应于检测视频A的指纹集FA＝{fa_1,...,fa_i…,fa_m}；

[0047] 指纹切块分组单元:用于对指纹集FB和指纹集FA中的所有指纹做指纹切块分组处理,分别得到大小为n×g的分组指纹集合GB和大小为m×g的分组指纹集合GA;

[0048] 倒排索引建立单元:用于为分组指纹集合GB创建g个表,相同组合方式的元素存储在同一张表内,然后对该表内的每个元素建立倒排索引,指向包含它的完整指纹;

[0049] 指纹比对单元:用于从分组指纹集合GA中取一个元素C,并逐一在分组指纹集合GB的倒排索引中寻找与C值相等元素:若只存在一个相等的元素E,则直接计算对应指纹的汉明距离,若结果小于或等于H,则两指纹为相似指纹;若存在多个元素E,则选取汉明距离最小的作为相似指纹;若不存在,则跳到下一元素,继续比对;重复上述指纹比对处理,直至找到指纹集FA与指纹集FB中所有相似的指纹,定义相似的指纹对应的关键帧为相似关键帧,以此获得相似关键帧的个数;

[0050] 相似判定单元:用于根据相似关键帧个数计算相似度,设定相似度阈值T_2,如果检测视频A和参考视频B的相似度$S_{A,B}$大于等于阈值T_2,则认为检测视频A与参考视频B相似;否则,则认为两视频不相似。

[0051] 优选的,所述指纹集获取单元包括:

[0052] 关键帧集合获取单元:用于分别获得参考视频B、检测视频A对应的关键帧集合,参考视频B为对应的关键帧集合$KB=\{kb_1,kb_2,\cdots,kb_j,\cdots,kb_n\}$,检测视频A为对应的关键帧集合$KA=\{ka_1,ka_2,\cdots,ka_i,\cdots,ka_m\}$;

[0053] 指纹计算执行单元:用于对关键帧集合KB、关键帧集合KA执行程序X得到关键帧的指纹fb_j、指纹fa_j,

[0054] 程序X为:

[0055] 将每个关键帧图像缩小为预设M×M的尺寸,并将其转换为灰度图;

[0056] 对灰度图进行离散余弦变换DCT,只保留结果中左上角的N×N子矩阵$Z_{N\times N}$,N=M/4;

[0057] 计算$Z_{N\times N}$所有元素的算数平均值;然后将大于或等于平均值的元素设为1,小于平均值的元素设为0;

[0058] 将二值化处理后的$Z_{N\times N}$按照先行后列的顺序得到一个二进制序列,作为该关键帧的指纹fb_j、指纹fa_j。

[0059] 优选的,关键帧集合获取单元包括:

[0060] 切帧模块:用于将参考视频B、检测视频A进行切帧处理,得到该视频的图像帧集;

[0061] 帧间差分强度执行模块:遍历图像帧集中的所有图像帧,计算前后两帧差分得到帧间差分强度;

[0062] 关键帧识别获取模块:用于将帧间差分强度为局部最大值的候选关键帧作为关键帧,即得到该视频的关键帧集合为$KB=\{kb_1,kb_2,\cdots,kb_j,\cdots,kb_n\}$,$KA=\{ka_1,ka_2,\cdots,ka_i,\cdots,ka_m\}$。

[0063] 本发明可以达到以下效果:

[0064] 本发明包括以下步骤:输入视频库中的所有视频文件,提取视频的关键帧;通过计算DCT值得到每一关键帧的指纹;将每一指纹平均分为d块且组合为g个组,建立倒排索引;输入需要检测的视频文件并计算其关键帧的指纹,同样分为n块g个组,查找包含与某组值完全相同的完整指纹;通过计算图像帧相似度,判定视频是否相似;若不相似,则视频数据库中不存在与输入视频相似的视频。本方法利用一种基于分块的快速比较算法,通过视频

最基础的关键帧图像的相似度比较进而判断两个视频的相似度，能够实现大多数情况下的大量视频相似度快速检测。

附图说明

[0065] 此处所说明的附图用来提供对本发明实施例的进一步理解，构成本申请的一部分，并不构成对本发明实施例的限定。在附图中：

[0066] 图1为提取关键帧流程示意图。

[0067] 图2为计算关键帧指纹流程示意图。

[0068] 图3为指纹比较流程示意图。

[0069] 图4为视频1未采用分块思想计算法和分块思想计算法的用时输出结果图。

[0070] 图5为视频2未采用分块思想计算法和分块思想计算法的用时输出结果图。

具体实施方式

[0071] 在对本发明的任意实施例进行详细的描述之前，应该理解本发明的应用不局限于下面的说明或附图中所示的结构的细节。本发明可采用其它的实施例，并且可以以各种方式被实施或被执行。基于本发明中的实施例，本领域普通技术人员在没有做出创造性改进前提下所获得的所有其它实施例，均属于本发明保护的范围。

[0072] 实施例1

[0073] 如图3所示：

[0074] 一种视频相似度快速检测方法，包括：

[0075] 指纹集处理：

[0076] 理参考视频B得到对应于参考视频B的指纹集FB＝{fb_1,...,fb_j,…,fb_n}；处理检测视频A得到对应于检测视频A的指纹集FA＝{fa_1,...,fa_i,…,fa_m}；

[0077] 指纹切块分组处理：

[0078] 对指纹集FB和指纹集FA中的所有指纹做指纹切块分组处理，分别得到大小为n×g的分组指纹集合GB和大小为m×g的分组指纹集合GA；

[0079] 建立倒排索引处理：

[0080] 为分组指纹集合GB创建g个表，相同组合方式的元素存储在同一张表内，然后对该表内的每个元素建立倒排索引，指向包含它的完整指纹；

[0081] 指纹比对处理：

[0082] 从分组指纹集合GA中取一个元素C，并逐一在分组指纹集合GB的倒排索引中寻找与C值相等元素：若只存在一个相等的元素E，则直接计算对应指纹的汉明距离，若结果小于或等于H，则两指纹为相似指纹；若存在多个元素E，则选取汉明距离最小的作为相似指纹；若不存在，则跳到下一元素，继续比对；重复上述指纹比对处理，直至找到指纹集FA与指纹集FB中所有相似的指纹，定义相似的指纹对应的关键帧为相似关键帧，以此获得相似关键帧的个数；

[0083] 相似判定处理：

[0084] 根据相似关键帧个数计算相似度，设定相似度阈值T_2，如果检测视频A和参考视频B的相似度$S_{A,B}$大于等于阈值T_2，则认为检测视频A与参考视频B相似；否则，则认为两视频不

相似。

[0085] 基于上述方案可以看出，本发明的构思是：基于对比视频最基础的图像特征，设计出一种基于分块的视频相似度快速检测方法，能大大提高对侵权视频的检测效率。

[0086] 基于分块的构思，定义两个汉明距离在设定阈值H内的指纹为相似指纹，将一个视频指纹平均切为d个等长的块。根据抽屉原理(例如，把9个苹果放进10个抽屉里，至少有一个空抽屉)，两个相似的指纹至少有(d-H)块是完全相等的，可以得出：有(d-H)块完全相等的两指纹有可能相似，但(d-H)块不完全相等的一定不可能相似，从而排除一些不可能相似的指纹。

[0087] 对切分的d个指纹块随机取出(d-H)块，对这些指纹块形成的值构建倒排索引(从值即能找到包含该值的所有的完整指纹)。把被测视频的指纹同样切分、随机取出(d-H)块，去倒排索引里查找这些指纹块形成的值是否存在：存在即说明该完整指纹可能与包含其的完整指纹相似，就把两完整指纹进行对比，汉明距离小于H的即为相似指纹(关键帧)；不存在则说明这两组完整指纹一定不相似，比较下一指纹块形成的值。

[0088] 可以看出，基于上述指纹切块分组处理、建立倒排索引处理、指纹比对处理可以极大的减少计算量。其中，将相同组合方式的元素存储在同一张表内建立倒排索引，这样采用抽屉原理，可以确定一定不相似的数据，使得省略重复处理的过程，从而快速提升计算速度。总的使得本发明的处理速度可以很快。我们经过测试，H＝5，d＝8，M＝32可以得到以下参数，可以验证上述构思能达到预期提升检测速度的目的。

[0089] 表1测试数据：

[0090]

视频	未采用分块思想计算法用时（s）	本发明采用分块思想计算法用时（s）
视频 1	2.501906	0.428806
视频 2	7.477258	1.873162

[0091] 指纹切块分组处理中，单个指纹做指纹切块分组处理具体为：将指纹集FB中任一指纹fb_j平均切分为d个等长的块，其中d>H，即fb_{j1}，fb_{j2}，…，fb_{jd}，再随机取出其中(d-H)份构成一个新元素，共有$g = C_d^{d-H}$种可能的组合方式；将指纹集FA中任一指纹fa_i平均切分为d个等长的块，即fa_{i1}，fa_{i2}，…，fa_{id}，再随机取出其中(d-H)份构成一个新元素，共有$g = C_d^{d-H}$种可能的组合方式；其中定义两个汉明距离在设定阈值H内的指纹为相似指纹。

[0092] 优选的，相似度按下式计算如下：

[0093] $$S_{A,B} = \frac{s_{num}}{\min(n,m)} \times 100\%$$

[0094] 式中，s_{num}表示检测视频A和参考视频B中相似关键帧个数，n和m分别表示参考视频B的关键帧的个数和检测视频A的关键帧的个数。

[0095] 实施例2

[0096] 在上述实施例的基础上，如图2所示，其所述指纹集处理的具体过程可以是：

[0097] 分别对参考视频B、检测视频A进行以下操作得到指纹集FB、指纹集FA，

[0098] S1、获取关键帧集合：分别获得参考视频B、检测视频A对应的关键帧集合，参考视频B为对应的关键帧集合KB＝{kb_1,kb_2,…,kb_j,…,kb_n}，检测视频A为对应的关键帧集合KA＝{ka_1,ka_2,…,ka_i,…,ka_m}；

[0099] S2、预处理：将每个关键帧图像缩小为预设M×M的尺寸，并将其转换为灰度图；

[0100] S3、离散余弦变换处理：对灰度图进行离散余弦变换DCT，只保留结果中左上角的N×N子矩阵$Z_{N\times N}$，N＝M/4；

[0101] S4、二值化处理：计算$Z_{N\times N}$所有元素的算数平均值；然后将大于或等于平均值的元素设为1，小于平均值的元素设为0；

[0102] S5、计算指纹处理：将二值化处理后的$Z_{N\times N}$按照先行后列的顺序得到一个二进制序列，作为该关键帧的指纹fb_j、指纹fa_i；

[0103] S6、重复S2至S5，分别得到指纹集FB＝{fb_1,...,fb_j…,fb_n}、指纹集FA＝{fa_1,...,fa_i…,fa_m}。

[0104] 优选的，所述离散余弦变换DCT计算如下所示：

[0105] $$F(u,v)=c(u)c(v)\sum_{w=0}^{M-1}\sum_{s=0}^{M-1}f(w,s)\cos\left[\frac{(w+0.5)\pi}{M}u\right]\cos\left[\frac{(s+0.5)\pi}{M}v\right]$$；

[0106] 式中，(u,v)代表着变换变量，f(w,s)是输入的图像，其中(w,s)代表着输入图像中像素的空间位置，且补偿系数为$c(x)=\begin{cases}\sqrt{\frac{1}{M}}, & x=0\\ \sqrt{\frac{2}{M}}, & x\neq 0\end{cases}$，$x=u,v$。

[0107] 实施例3

[0108] 在上述实施例的基础上，如图1所示，关键帧集合可以采用以下具体过程：

[0109] 步骤S11：将参考视频B、检测视频A进行切帧处理，得到该视频的图像帧集；

[0110] 步骤S12：遍历图像帧集中的所有图像帧，计算前后两帧差分得到帧间差分强度；

[0111] 步骤S13：将帧间差分强度为局部最大值的候选关键帧作为关键帧，即得到该视频的关键帧集合为KB＝{kb_1,kb_2,…,kb_j,…,kb_n}，KA＝{ka_1,ka_2,…,ka_i,…,ka_m}。

[0112] 优选的，所述帧间差分强度，如下所示：

[0113] 对于参考视频B有：Pj＝|b_{j+1}-b_j|；对于检测视频A有：Pj＝|a_{i+1}-a_i|；

[0114] 其中Pj表示该视频的第j帧的帧间差分强度，b_j表示参考视频B的图像帧集B＝{b_1,b_2,…,b_j,…,b_n}中的某一帧图像，b_{j+1}表示图像帧集B中b_j的后一帧图像，a_i表示检测视频A的图像帧集A＝{a_1,a_2,…,a_i,…,a_m}中的某一帧图像，a_{i+1}表示图像帧集A中a_i的后一帧图像；然后，将帧间差分强度大于阈值T_1的帧作为一个候选关键帧。

[0115] 实施例4

[0116] 在大量视频中鉴别相似视频的检测装置，包括：

[0117] 指纹集获取单元：用于处理参考视频B得到对应于参考视频B的指纹集FB＝{fb_1,...,fb_j…,fb_n}；处理检测视频A得到对应于检测视频A的指纹集FA＝{fa_1,...,fa_i…,fa_m}；

[0118] 指纹切块分组单元：用于对指纹集FB和指纹集FA中的所有指纹做指纹切块分组处理，分别得到大小为n×g的分组指纹集合GB和大小为m×g的分组指纹集合GA；

[0119] 倒排索引建立单元:用于为分组指纹集合GB创建g个表,相同组合方式的元素存储在同一张表内,然后对该表内的每个元素建立倒排索引,指向包含它的完整指纹;

[0120] 指纹比对单元:用于从分组指纹集合GA中取一个元素C,并逐一在分组指纹集合GB的倒排索引中寻找与C值相等元素:若只存在一个相等的元素E,则直接计算对应指纹的汉明距离,若结果小于或等于H,则两指纹为相似指纹;若存在多个元素E,则选取汉明距离最小的作为相似指纹;若不存在,则跳到下一元素,继续比对;重复上述指纹比对处理,直至找到指纹集FA与指纹集FB中所有相似的指纹,定义相似的指纹对应的关键帧为相似关键帧,以此获得相似关键帧的个数;

[0121] 相似判定单元:用于根据相似关键帧个数计算相似度,设定相似度阈值T_2,如果检测视频A和参考视频B的相似度$S_{A,B}$大于等于阈值T_2,则认为检测视频A与参考视频B相似;否则,则认为两视频不相似。

[0122] 优选的,所述指纹集获取单元包括:

[0123] 关键帧集合获取单元:用于分别获得参考视频B、检测视频A对应的关键帧集合,参考视频B为对应的关键帧集合KB=$\{kb_1,kb_2,\cdots,kb_j,\cdots,kb_n\}$,检测视频A为对应的关键帧集合KA=$\{ka_1,ka_2,\cdots,ka_i,\cdots,ka_m\}$;

[0124] 指纹计算执行单元:用于对关键帧集合KB、关键帧集合KA执行程序X得到关键帧的指纹fb_j、指纹fa_i,

[0125] 程序X为:

[0126] 将每个关键帧图像缩小为预设M×M的尺寸,并将其转换为灰度图;

[0127] 对灰度图进行离散余弦变换DCT,只保留结果中左上角的N×N子矩阵$Z_{N\times N}$,N=M/4;

[0128] 计算$Z_{N\times N}$所有元素的算数平均值;然后将大于或等于平均值的元素设为1,小于平均值的元素设为0;

[0129] 将二值化处理后的$Z_{N\times N}$按照先行后列的顺序得到一个二进制序列,作为该关键帧的指纹fb_j、指纹fa_j。

[0130] 优选的,关键帧集合获取单元包括:

[0131] 切帧模块:用于将参考视频B、检测视频A进行切帧处理,得到该视频的图像帧集;

[0132] 帧间差分强度执行模块:遍历图像帧集中的所有图像帧,计算前后两帧差分得到帧间差分强度;

[0133] 关键帧识别获取模块:用于将帧间差分强度为局部最大值的候选关键帧作为关键帧,即得到该视频的关键帧集合为KB=$\{kb_1,kb_2,\cdots,kb_j,\cdots,kb_n\}$,KA=$\{ka_1,ka_2,\cdots,ka_i,\cdots,ka_m\}$。

[0134] 实施例5

[0135] 如图1、图2、图3结合所示:

[0136] 基于图像分块的相似视频快速检测方法,包括如下步骤:

[0137] 步骤1:将参考视频B进行切帧处理,得到该视频的图像帧集。

[0138] 步骤2:遍历图像帧集中的所有图像帧,计算前后两帧差分得到帧间差分强度,

[0139] 步骤3:将帧间差分强度为局部最大值的候选关键帧作为关键帧,即得到该视频的关键帧集合为KB=$\{kb_1,kb_2,\cdots,kb_j,\cdots,kb_n\}$。

[0140] 步骤4:将每个关键帧图像缩小为M×M的尺寸,并将其转换为灰度图。

[0141] 步骤5:对步骤S4中灰度图进行离散余弦变换(DCT),只保留结果中左上角的N×N子矩阵$Z_{N\times N}$,N=M/4。DCT计算如下所示:

[0142] $$F(u,v)=c(u)c(v)\sum_{w=0}^{M-1}\sum_{s=0}^{M-1}f(w,s)\cos\left[\frac{(w+0.5)\pi}{M}u\right]\cos\left[\frac{(s+0.5)\pi}{M}v\right]$$

[0143] 式中,(u,v)代表着变换变量,f(w,s)是输入的图像,其中(w,s)代表着输入图像中像素的空间位置,且补偿系数为$c(x)=\begin{cases}\sqrt{\frac{1}{M}}, & x=0\\ \sqrt{\frac{2}{M}}, & x\neq 0\end{cases}$, $x=u,v$。

[0144] 步骤6:二值化:计算$Z_{N\times N}$所有元素的算数平均值;然后将大于或等于平均值的元素设为1,小于平均值的元素设为0。

[0145] 步骤7:计算指纹:将二值化后的$Z_{N\times N}$按照先行后列的顺序得到一个二进制序列,作为该关键帧的指纹fb_j。

[0146] 步骤8:重复步骤4至7得到参考视频B的指纹集FB={fb_1,...,fb_j…,fb_n};

[0147] 步骤9:将检测视频A根据步骤S1至S8处理得到检测视频A的指纹集FA={fa_1,...,fa_i…,fa_m}。

[0148] 步骤10:指纹切块分组。定义两个汉明距离在设定阈值H内的指纹为相似指纹。将FB中任一指纹fb_j平均切分为d(d>H)个等长的块,即fb_{j1},fb_{j2},…,fb_{jd}。随机取出其中(d-H)份构成一个新元素,共有$g=C_d^{d-H}$种可能的组合方式。对FB集合和FA集合中的所有指纹做此操作,分别得到大小为n×g的分组指纹集合GB和大小为m×g的分组指纹集合GA。

[0149] 步骤11:建立倒排索引(inverted index):为GB创建g个表,相同组合方式的元素存储在同一张表内,然后对该表内的每个元素建立倒排索引,指向包含它的完整指纹。

[0150] 步骤12:指纹比对:定义相似指纹对应的关键帧为相似关键帧。从GA中取一个元素C,并逐一在GB的倒排索引中寻找与C值相等元素:若只存在一个相等的元素E,则直接计算对应指纹的汉明距离,若结果小于或等于H,则两指纹为相似指纹;若存在多个元素E,则选取汉明距离最小的作为相似指纹;若不存在,则跳到下一元素,继续比对。

[0151] 步骤13:重复步骤12,直至找到FA与FB中所有相似的指纹。

[0152] 步骤:14:计算相似度。

[0153] 步骤15:设定相似度阈值T_2。如果视频A和B的相似度$S_{A,B}$大于等于阈值T_2,则认为检测视频A与参考视频B相似;否则,则认为两视频不相似。

[0154] 以上所述的具体实施方式,对本发明的目的、技术方案和有益效果进行了进一步详细说明,所应理解的是,以上所述仅为本发明的具体实施方式而已,并不用于限定本发明的保护范围,凡在本发明的精神和原则之内,所做的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

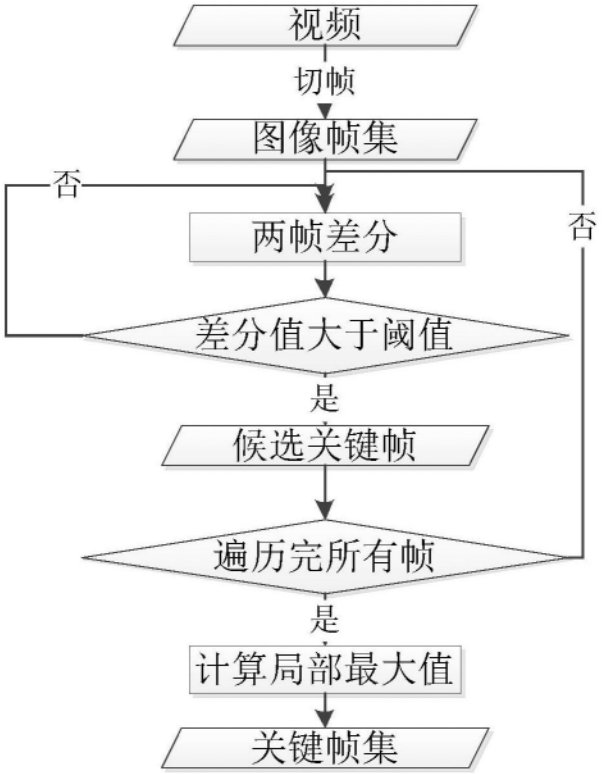

视频
切帧
图像帧集
否
两帧差分
否
差分值大于阈值
是
候选关键帧
遍历完所有帧
是
计算局部最大值
关键帧集

图1

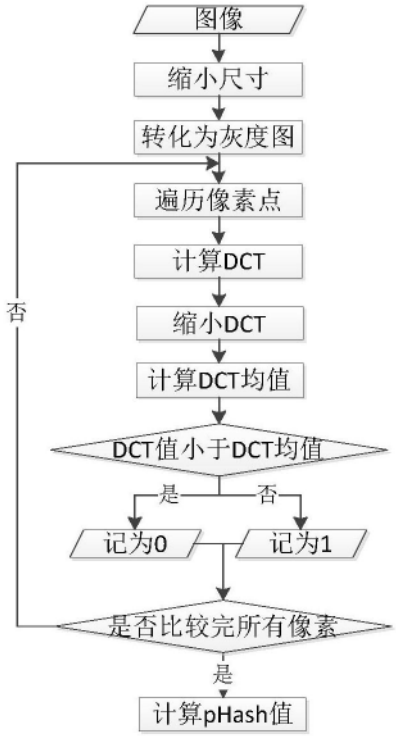

图像
缩小尺寸
转化为灰度图
遍历像素点
计算DCT
缩小DCT
计算DCT均值
DCT值小于DCT均值
是
否
记为0
记为1
否
是否比较完所有像素
是
计算pHash值

图2

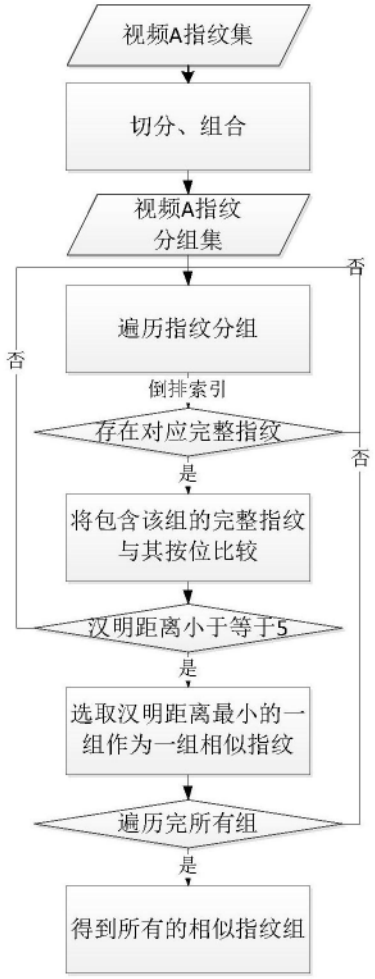

视频A指纹集
切分、组合
视频A指纹
分组集
遍历指纹分组
否
否
倒排索引
存在对应完整指纹
否
是
将包含该组的完整指纹
与其按位比较
汉明距离小于等于5
是
选取汉明距离最小的一
组作为一组相似指纹
遍历完所有组
是
得到所有的相似指纹组

图3

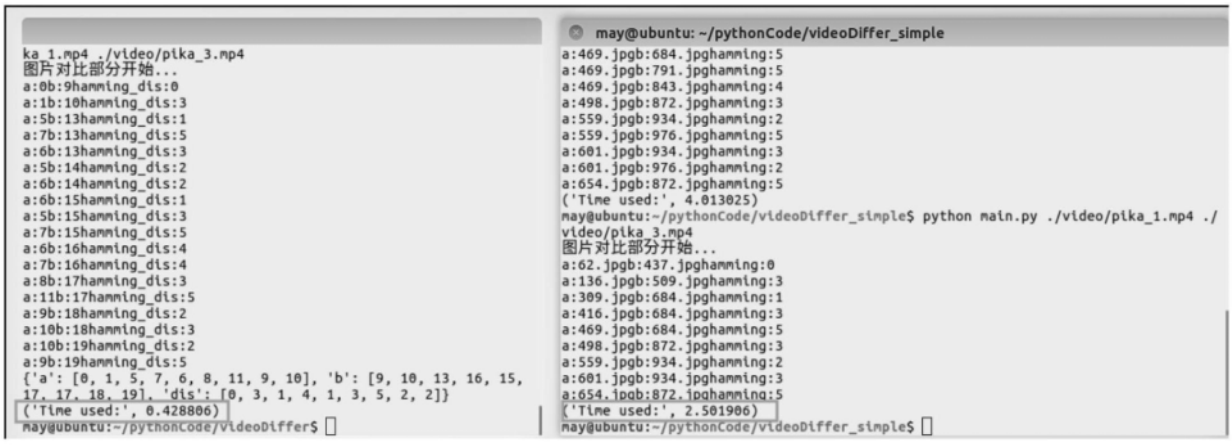

ka_1.mp4 ./video/pika_3.mp4
图片对比部分开始...
a:0b:9hamming_dis:0
a:1b:10hamming_dis:3
a:5b:13hamming_dis:1
a:7b:13hamming_dis:5
a:6b:13hamming_dis:3
a:5b:14hamming_dis:2
a:6b:14hamming_dis:2
a:6b:15hamming_dis:1
a:5b:15hamming_dis:3
a:7b:15hamming_dis:5
a:6b:16hamming_dis:4
a:7b:16hamming_dis:4
a:8b:17hamming_dis:3
a:11b:17hamming_dis:5
a:9b:18hamming_dis:2
a:10b:18hamming_dis:3
a:10b:19hamming_dis:2
a:9b:19hamming_dis:5
{'a': [0, 1, 5, 7, 6, 8, 11, 9, 10], 'b': [9, 10, 13, 16, 15, 17, 17, 18, 19], 'dis': [0, 3, 1, 4, 1, 3, 5, 2, 2]}
('Time used:', 0.428806)
may@ubuntu:~/pythonCode/videoDiffer$
may@ubuntu: ~/pythonCode/videoDiffer_simple
a:469.jpgb:684.jpghamming:5
a:469.jpgb:791.jpghamming:5
a:469.jpgb:843.jpghamming:4
a:498.jpgb:872.jpghamming:3
a:559.jpgb:934.jpghamming:2
a:559.jpgb:976.jpghamming:5
a:601.jpgb:934.jpghamming:3
a:601.jpgb:976.jpghamming:2
a:654.jpgb:872.jpghamming:5
('Time used:', 4.013025)
may@ubuntu:~/pythonCode/videoDiffer_simple$ python main.py ./video/pika_1.mp4 ./video/pika_3.mp4
图片对比部分开始...
a:62.jpgb:437.jpghamming:0
a:136.jpgb:509.jpghamming:3
a:309.jpgb:684.jpghamming:1
a:416.jpgb:684.jpghamming:3
a:469.jpgb:684.jpghamming:5
a:498.jpgb:872.jpghamming:3
a:559.jpgb:934.jpghamming:2
a:601.jpgb:934.jpghamming:3
a:654.jpgb:872.jpghamming:5
('Time used:', 2.501906)
may@ubuntu:~/pythonCode/videoDiffer_simple$

图4


may@ubuntu: ~/pythonCode/videoDiffer
ka_1.mp4 ./video/pika_3.mp4
图片对比部分开始...
Traceback (most recent call last):
File "main.py", line 91, in <module>
photolist1.sort(key=lambda x: int(x.split('.')[0]))#按照文件夹中文件的排列顺序
File "main.py", line 91, in <lambda>
photolist1.sort(key=lambda x: int(x.split('.')[0]))#按照文件夹中文件的排列顺序
ValueError: invalid literal for int() with base 10: 'keyframe_693'
may@ubuntu:~/pythonCode/videoDiffer$ python main.py ./video/pika_1.mp4 ./video/pika_3.mp4
图片对比部分开始...
a:0b:4hamming_dis:1
a:2b:6hamming_dis:1
a:3b:7hamming_dis:1
a:4b:8hamming_dis:0
a:5b:9hamming_dis:2
a:6b:10hamming_dis:0
{'a': [0, 2, 3, 4, 5, 6], 'b': [4, 6, 7, 8, 9, 10], 'dis': [1, 1, 1, 0, 2, 0]}
('Time used:', 1.8731619999999998)
may@ubuntu:~/pythonCode/videoDiffer$
may@ubuntu:~/pythonCode/videoDiffer_simple$ python main.py ./video/pika_1.mp4 ./video/pika_3.mp4
图片对比部分开始...
a:62.jpgb:437.jpghamming:0
a:136.jpgb:509.jpghamming:3
a:309.jpgb:684.jpghamming:1
a:416.jpgb:684.jpghamming:3
a:469.jpgb:684.jpghamming:5
a:498.jpgb:872.jpghamming:3
a:559.jpgb:934.jpghamming:2
a:601.jpgb:934.jpghamming:3
a:654.jpgb:872.jpghamming:5
('Time used:', 2.501906)
may@ubuntu:~/pythonCode/videoDiffer_simple$ python main.py ./video/pika_1.mp4 ./video/pika_3.mp4
图片对比部分开始...
a:39.jpgb:199.jpghamming:1
a:90.jpgb:250.jpghamming:1
a:180.jpgb:341.jpghamming:1
a:226.jpgb:386.jpghamming:0
a:272.jpgb:432.jpghamming:2
a:324.jpgb:484.jpghamming:0
('Time used:', 7.477258)
may@ubuntu:~/pythonCode/videoDiffer_simple$

图5