[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04N 7/24

[12] 发明专利申请公开说明书

[21] 申请号 200410018508.0

[43] 公开日 2005 年 2 月 16 日

[11] 公开号 CN 1581972A

[22] 申请日 2004.5.20
[21] 申请号 200410018508.0
[71] 申请人 复旦大学
地址 200433 上海市邯郸路 220 号
[72] 发明人 肖友能 薛向阳

[74] 专利代理机构 上海正旦专利代理有限公司
代理人 陆 飞 沈 云

权利要求书 2 页 说明书 8 页 附图 2 页

[54] 发明名称 一种隐藏错误的视频译码方法

[57] 摘要

本发明是一种隐藏错误的视频译码方法。本发明利用视频解码器在解码时得到的副产品信息，包括运动矢量和 I 帧的直流(DC)图像，进行实时镜头分割，判断当前发生错误的帧是否是镜头边界。如果当前帧是镜头边界，则使用空域上基于块的分割－匹配的错误隐蔽方法进行错误隐藏；如果不是边界，则用时域上的前向－后向块匹配错误隐蔽方法进行错误隐藏。该方法具有速度快、系统开销小、隐藏效果好、鲁棒性好等优点，非常适合于各种类型的视频播放器。

ISSN1008-4274

1、一种隐藏错误的视频译码方法，其特征在于：首先对视频流中错误进行定位：检查宏块边界的像素值和它周围宏块边界像素值的差值，或者从 MPEG 码流的语法中实现错误检查；然后对有错帧，采用基于运动矢量或直流（DC）图像方法，判断它是否出现在镜头边界上；如果不在镜头边界上，就用时域上前向－后向块匹配错误隐蔽方法进行错误隐藏；如果在镜头边界上，就用空域上基于块的分割-匹配的错误隐蔽方法进行隐藏。

2、根据权利要求 1 所述的方法，其特征在于所述检查宏块边界的像素值和它周围宏块边界像素值的差值，是计算当前宏块的上边界和上方宏块下边界的 U 分量的差值、以及该宏块的下边界和下方宏块上边界的的 U 分量的差值；如果这两个差值都大于一个给定阈值 TH_1，则断定当前宏块有错，这里 U 分量为绿色差。

3、根据权利要求 1 所述的方法，其特征在于所述从 MPEG 码流的语法中检索错误，是当检测到一个宏块的运动矢量或离散余弦变换（DCT）系数的变字长码字（VLC）不在编码表中时，则断定当前宏块的码流有错。

4、根据权利要求 1 所述的方法，其特征在于在空域上基于块的分割-匹配的错误隐蔽方法，具体步骤如下：

(1)定位出错的宏块的位置；

(2)判断“出错宏块的正上方相邻宏块和正下方相邻宏块”是否都可以用；如果不可用，找到最近的可以用于错误隐藏的宏块；

(3)出错宏块中区域和可用宏块中的区域的相似度计算；

(4)完成分割-匹配过程；

(5)计算第四步中分割-匹配过程所用的两个区域是否是相似区域；如果不是，转（3），如果是，转（6）；

(6)完成错误隐藏；

(7)判断当前出错宏块的每个区域进行了错误隐藏；如果不是，转（3）；如果是，退出。

5、根据权利要求 4 所述的方法，其特征在于所述的两个宏块的相似度以绝对误差均值来衡量，大小为 $bx \times by$ 的图像块 B_T 和 B_B 之间绝对误差的均值通过下式计算：

$$MAD = \frac{1}{bx \times by} \sum_{x=0}^{bx} \sum_{y=0}^{by} \left| B_T(x,y) - B_B(x,y) \right|$$

6、根据权利要求 1 所述的方法，其特征在于镜头分割采用基于运动矢量的实时镜头分割方法后向块匹配错误隐蔽。

7、根据权利要求 6 所述的方法，其特征在于当进行实时镜头分割时候，对 P 帧，检测没有运动矢量的宏块和有前向运动矢量的宏块的比率是否异常；对 B 帧，检测有后向运动矢量的宏块和有前向运动矢量的宏块的比率是否异常；对 I 帧，检测 I 帧 DC 图像的差值是否出现异常。

8、根据权利要求 7 所述的方法，其特征在于检测 P 帧、B 帧、J 帧的异常，采用滑动窗口算法，具体步骤如下：

上述步骤中，FrameNo 为帧序号，ShotCount 为当前镜头计数器，W 为一个长度为 l 的滑动窗口，w_i 是窗口中的元素，$E(W)$ 为窗口的期望；X 表示 P 帧、B 帧、I 帧类型之一种。

9、根据权利要求 8 所述的方法，其特征在于对于 P 帧，X 取为 P，此时，InitialEle= 1000，$l \in [8,10]$， TH_P=4；对于 B 帧，X 取为 B，此时，InitialEle= 1000，$l \in [18,22]$， TH_B=6；对于 I 帧，X 取为 I， $InitialEle=\frac{width \times height \times 8}{256}$，$l=4$，TH_I=3。

一种隐藏错误的视频译码方法

技术领域

本发明属视频技术领域，具体涉及一种隐藏错误的视频译码方法。主要用于提高视频解码器的图像质量，其原理是隐藏（即尽量让人看不见）因视频码流错误而引起的译码图像中马赛克现象，降低人的视觉系统对图像质量恶化的敏感程度。

随着数字电视、会议电视、计算机网络等技术的不断发展和迅速普及，如何在 Internet 上传送视频流变得越来越重要。由于 Internet 没有为视频和音频等实时流媒体传输保留专门通道，使得音视频的传输很容易受到网络拥塞和延迟的影响。视频编码本身的一些基本压缩技术，如 MPEG-1/2/4 标准中的时域预测和变字长编码等，特别容易受到网络拥塞和丢包的影响，即使一个比特的误码或者丢失，也会造成视频图像中大片错误，造成主观视觉质量严重下降。

码流在网络传输过程中随时可能出现数据丢失。在 MPEG 视频数据中，大部分比特是用于编码运动矢量的哈夫曼码（Huffman）和离散余弦变换（DCT）系数的 Huffman 码，众所周知 Huffman 码是一种变字长码，一旦发生错误，很难立即发现，通常是错误蔓延到无法解码的时候才知道前面某个未知位置处的码流出现了错误。变长编码的特点使得译码算法不可能退回去定位出错的地点，只能重新查找开始码，从下一个同步点开始继续解码。从发现错误（实际的错误往往还在此之前）到下一个同步点之间的宏块的解码是没有定义的。而这一错误更由于具有很高压缩效率的时域预测而雪上加霜。如果这一错误发生在一个图象组中较前参考帧中，则该图象组中所有后续解码帧中相同位置宏块的解码都会引起失真，这种失真一般经常会持续半秒，后果是很严重的。

大量研究表明，可以从许多方面来克服这种错误扩散问题，例如数据分级，添加前向纠错（FEC）码，发送方流量控制等，在视频解码器中添加错误隐藏措施也被证明是一种很有效的技术手段，其代价只是增加一些译码算法的复杂度。

常用的错误隐藏方法包括：前向错误隐蔽、通过后处理的错误隐蔽、解码器和编码器交互的错误隐蔽三大类。（1）前向错误隐蔽：编码器起主要作用，通过增加了一定量的冗余信息在信息源编码器端或传送层编码器端，当数据在网络上发生错误时，接收端有可能从冗余信息中恢复出丢失的信息。通常增加的冗余信息数据量是不可忽视的，当网络中发生拥塞时，冗余信息会加重这种拥塞，引起视频质量更严重的下降。（2）在解码器和编码

器交互的错误隐蔽方法中，编码器端通过从解码器端获得反馈信息来调整自己的编码策略以获得更好的性能，这就要求编码器和解码器具有一定的交互性，因此编码器和解码器通常是同一个开发者。这样的编码器得到的视频流缺乏通用性。鉴于上述两种错误隐藏方法存在一些本质缺陷，通过后处理的错误隐蔽方法得到了广泛的应用。

通过后处理的错误隐蔽方法通常有下面几种：

(1)基于运动补偿的时域错误隐藏：一个简单的方法来利用视频信号的时域相关特性，即用错误宏块所在帧的前一帧中与错误宏块空间位置相同的宏块，来代替当前帧中的错误宏块。但是当视频中存在较剧烈的运动时，这种方法将对视觉效果产生很大影响。通过运动补偿来寻找相应的替代宏块时，对隐藏效果有很大提高。

(2)最大平滑恢复：利用大多数图象和视频信号的空间平滑特性，通过逐步能量最小化方法实现。

(3)在空域或频域上的错误隐藏：空域或频域上错误隐藏方法，是在一帧图像内，利用数据丢失块周围的图像，采用插值方法恢复丢失块。由于一个突发性的错误往往引起相邻数块数据的丢失，在这种情况下，插值的计算将更为复杂、效果也较差。

发明内容

本发明的目的在于提出一种计算较为简单，效果比较好的隐藏错误的视频译码方法。

本发明提出的隐藏错误的视频译码方法，是一种内嵌于视频解码器中进行错误隐藏的方法，它将压缩域快速镜头分割方法和空域错误隐藏、时域错误隐藏方法相结合，获得良好的错误隐藏效果。其步骤为：先对视频中的错误进行定位，具体可使用两种方法检查错误：（1）计算宏块边界的像素值和它周围宏块边界像素值差值；（2）检查 MPEG 码流的语法以发现码流中错误。然后对于有错误的帧，采用基于运动矢量（P 和 B 帧）和直流（DC）图像（I 帧）的方法检测发生错误的帧是否在镜头（shot）的边界上。如果不在镜头的边界，就用时域上的前向－后向块匹配错误隐蔽方法进行错误隐藏；如果在镜头的边界，那么前后相邻帧之间差异非常大，不能进行时间域的错误隐蔽，此时采用空域上基于块的分割—匹配的错误隐蔽方法进行错误隐藏。

附图说明

图 1 为本发明分割-匹配过程 4 个步骤的图示，其中(a)为初始匹配图示，(b)为首次分割图示，(c)为二次分割图示，(d)为不进行分割、增大搜索区域图示。

图 2 为时域上的错误隐藏图示。其中(A)为选择错误宏块上方的宏块，(B)为选择错误宏块下方的宏块，(C)为同时选择错误宏块上方和下方的宏块。

具体实施方式

本发明基于镜头分割的错误隐藏方法可以归纳成如下基本步骤：

1、解码第n帧。

2、检测第n帧是否发生错误，如果发生错误转 3，否则转 5。

3、检测第n帧是否是一个镜头的边界。

4、如果第n帧不是一个镜头的边界，进行时域错误隐藏。如果是镜头边界，进行空域错误隐藏。

5、如果第n帧是最后一帧，退出，否则$n=n+1$，转 1。

下面分别具体介绍错误发现机制和所采用的空域错误隐藏和时域错误隐藏方法，再给出压缩域快速镜头分割方法，最后将镜头分割方法和空域错误隐藏、时域错误隐藏结合起来。在讨论具体实施方法时，以 MPEG-2 标准的视频流为例介绍，但本发明方法并不仅限于 MPEG-2 标准，同样适用于 MPEG-1/4 标准或其它视频编码标准。

1、错误发现机制

为了做好错误隐蔽，最重要的是能否正确定位错误出现的位置。如果不能准确定位错误的位置，错误隐藏便不能开展。本发明错误发现主要用了两种方法：

(1)检查宏块边界的像素值和它周围宏块边界像素值的差值

当视频流中出现错误而在播放时产生“马赛克”现象时，通过直观可以感到出现错误的宏块和周围宏块有明显的差别。这里正是利用了出错宏块和周围宏块在色度、亮度上的差别来发现错误宏块的位置的。根据 MPEG-2 的编码方式，对于宏块采用了 VLC 编码，当某个宏块发生错误时，往往会影响到该片(slice)中后续宏块，直到下一个片的到来。又因为一个片是不会超越一行的，因此该宏块带来的错误最多只会影响本行宏块。

通过上面分析，可以知道：在错误宏块周围的宏块中，左右的宏块往往会受到影响，而其上下宏块一般不会受到影响，因此这里只计算宏块的上边界和上方宏块下边界的差值以及宏块的下边界和下方宏块上边界的差值。通过对许多帧的计算和比较发现在颜色 YUV 三个分量中(其中，Y 分量表示亮度，U 表示蓝色和绿色的色差，V 分量表示红色与绿色的色差)，U 分量值的差别是最显著的，因此在本方法中选择 U 分量的差别作为宏块和周围宏块差值大小的判别。选取少量像素值参与计算是为了降低算法的复杂度。

在本方法中选择计算当前宏块上边界的像素值和上方宏块下边界的像素值在 U 分量上的差值以及该宏块下边界的像素值和下方宏块上边界的像素在 U 分量上的差值同时作为判决条件，当这两个差值同时大于阈值（本方法中阈值范围为 25～35）时，可以判定该宏块发生了错误。

(2)当检测到的码值不在码表中时，说明码流中发生了错误

由于 MPEG 标准中所用的 VLC 码表并不是完全的，因此当码流发生错误时，很可能出现一个宏块的运动矢量或离散余弦变换(DCT)系数的变字长码字(VLC)不在 VLC 码表中，当这种情况出现时可以断定在码流中发生了错误。但是用这种方法往往不能精确定位错误的位置，因为当一个码字不在 VLC 码表中出现时，可能并不是这个码字出现了错误，而是在这个片中前面的码字就出现了错误，只是一个码字由于有比特错误变成了另外一个码字而且改变了码字的长度而引起的错误传播，导致在一段时间后才发现错误。不过，一旦出现这种情况，就可以断定该片中发生了传送错误，因此在本方法实现的错误隐蔽中，当发生这种情况时，我们将对整个片实施错误隐蔽。

2、空域上错误隐藏

空域上错误隐藏采用了空域上基于块的分割-匹配的错误隐蔽方法，下述算法 1 是对一个宏块进行错误隐藏的方法：

算法 1，具体步骤如下：

(1)定位出错的宏块的位置；

(2)判断“出错宏块的正上方相邻宏块和正下方相邻宏块”是否都可以用；如果不可用，找到最近的可以用于错误隐藏的宏块；

(3)出错宏块中区域和可用宏块中的区域的相似度计算；

(4)完成分割-匹配过程；

(5)计算第四步中分割-匹配过程所用的两个区域是否是相似区域；如果不是，转（3），如果是，转（6）；

(6)完成错误隐藏；

(7)判断当前出错宏块的每个区域进行了错误隐藏；如果不是，转（3）；如果是，退出。

下面将解释算法 1 所涉及的技术细节。

区域相似度估计：

在该方法中，大小为 $bx \times by$ 的相邻块 B_T 和 B_B 之间的绝对误差的均值通过下式估计：

$$MAD = \frac{1}{bx \times by} \sum_{x=0}^{bx} \sum_{y=0}^{by} \left| B_T(x,y) - B_B(x,y) \right|$$

当 MAD 值小于一个预设阈值时，或在指定搜索区域内找到了一个最小 MAD 时，用此时的两个匹配块来进行错误隐蔽。

分割-匹配过程

分割-匹配过程的 4 个步骤如图 1 所示。其中 MBc 表示要被错误隐蔽的宏块，

$MB_T, MB_{TL}, MB_{TR}, MB_B, MB_{BL}, MB_{BR}$ 分别表示被损宏块的上方、左上方、右上方、下方、左下方、右下方的可获得宏块。

在前 2 个步骤中，定义了较大的相邻块来进行匹配，而在最后 2 个步骤中都用了最小尺寸 4×4 的块进行匹配，但定义了不同的搜索区域。采用这个方法保证了最小的处理时间和尽可能好的隐藏效果。在过程的开始用较大块进行匹配保证了有较大一致性的区域可以被直接重构，避免了更多的计算和延迟。另一方面，有较多细节的区域需要更小的块来匹配。下面详细介绍 4 个步骤：

(1)初始匹配 — 最大的块尺寸

初始匹配试图匹配竖直方向上相邻的最大块 b_1 和 t_1，其尺寸为16×8像素。如图 1a 所示。如果这两个区域被认为是相似的，整个宏块的错误隐蔽可以通过拷贝来实现。否则算法继续执行第（2）步。

(2)首次分割 — 仅在竖直方向上

将第(1)步中初始块分割成两个更小的块，分别为 b_1、b_2 和 t_1、t_2，其大小为8×8。仅在竖直方向上为每一对进行匹配。如 b_1 与 t_1 和 b_2 与 t_2（见图 1(b)）。因此有较小纹理区域的隐藏被直接执行。

(3)二次分割 — 定义初始搜索区域—较少方向的集合

第(2)步中，块被进一步分割成所允许的最小的大小 4×4，用 b_i 和 t_i 来表示。在这一步骤中，如图 1(c)所示，定义了水平方向上的初始搜索区域，将要被匹配的最小块(b_i 及 t_i)在定义的搜索区域中滑动。所有它们的组合都用前面介绍过的相似度估计算法来进行是否是最佳匹配的决策。这一步骤被迭代执行，直到下一个最佳块组合不再满足相似度匹配或整个宏块的错误隐蔽已经实现。滑动的概念保证了任何方向的边界或线条可以被重构。

(4)不进行分割 — 增大搜索区域 — 较多方向的集合

如图 1(d)所示第(3)步中定义的搜索区域被进一步增大。允许重构的方向扩展到相邻的左上方、右上方或左下方、右下方的宏块，因此扩大了重构的方向。在该步骤中，用来匹配的块的大小保持不变，如图 1(d)所示。匹配在 b_k，t_k 间进行。这一步骤也是迭代进行，直到整个宏块都进行了错误隐蔽。

错误隐藏

在本方法的错误隐蔽方法中采用了拷贝的方法。如图 1 所示，上方和下方相邻最佳匹配块的图像内容按匹配方向分别被拷贝到丢失区域的上半部分和下半部分。（例如，在图 1a 中，如果 b_1 及 t_1 在第一步中被认为是匹配的，则 b_1 的内容被拷贝到错误宏块上半部分中

相等大小的区域，即图中颜色较浅的区域；t_1被拷贝到错误宏块的下半部分中相等大小的区域，即图中颜色较深的区域）。在分割-匹配过程的前两个步骤中，错误隐蔽被直接执行而不需要迭代过程，而在后两个步骤中需要迭代过程，对在前一步中还未进行错误隐蔽的区域进行错误隐蔽。因此在算法中为宏块中的每个像素设置了一个标志，用来指示该像素是否已经进行错误隐蔽。如果像素已经进行错误隐蔽，当他在下次最佳匹配的方向上出现时，不需要再对他进行错误隐蔽。

3、时域上的错误隐藏

和空域上的错误隐蔽相比，在运动较少或没有运动的区域，时域上的错误隐蔽可以取得近乎完美的效果。当发生错误的宏块周围宏块的运动向量不一致或由于其周围宏块为帧内编码的而无法获得它们的运动信息时，简单的块匹配方法无法获得令人满意的结果，而本方法中的时域上的前向－后向块匹配错误隐蔽方法可以减小这些缺点。该方法为错误宏块选择了邻域，在基于块匹配的原则上，在该邻域中可对错误宏块进行可能的最佳隐藏。

时域上的前向－后向块匹配错误隐蔽利用了视频序列的时域上的冗余。这里假设至少一些相邻的块有平滑、一致的运动向量。下面详细介绍该算法。

MB_C为当前发生错误的宏块，MB_T为错误宏块上方的宏块，MB_B为错误宏块下方的宏块。在参考帧中以与错误宏块相同位置的宏块为中心，预先定义$N \times M$大小的区域为搜索区域，同时选择错误宏块的周围的宏块作为候选者。这里我们可以只选择错误宏块上方的宏块，如图 2A 所示；也可以只选择错误宏块下方的宏块，如图 2(B)所示；也可以同时选择错误宏块上方和下方的宏块作为候选者，如图 2©所示。在最后一种情况下，上下宏块的竖直距离被限制为 16 象素，同时MB_T和MB_B在搜索区域中的运动是一致的。时域上的块匹配对所有候选者在选定的区域中搜索导致最小MAD的块来作为错误宏块内容的估计。错误隐蔽将找到的最佳匹配块的适当的周围块的内容拷贝到发生错误的宏块的位置上，比如如果我们采用图 2(A)的方式选择候选者，那么将匹配宏块下方的宏块拷贝到错误宏块的位置。

4、压缩域中快速镜头分割方法

在传统的错误隐藏策略中，一般来说时域错误隐藏具有计算开销小，隐藏效果好的特点，是在错误隐藏中被倾向使用的方法。但是当视频中相邻帧间的内容变化特别明显时（这种明显变化一般是由镜头切换引起的），时域错误隐藏的效果就很差。考虑到这一点，我们提出的新的错误隐藏方法将实时镜头分割和传统的错误隐藏方法相结合。

一个镜头是摄像机打开和关闭之间的时间内拍摄的视频片断。一个视频节目在语义上可以看作是一些镜头的组合，镜头分割就是找到这些语义的边界。用于镜头分割的算法很

多，考虑到我们的应用中，镜头分割的精度不是很重要，而镜头分割的速度却很关键，本发明中采用了基于运动矢量的实时镜头分割方法,基本思想如下：

在MPEG－2流中，有I、P、B三种类型的帧，在I帧中没有运动矢量，在P帧中有前向运动矢量，在B帧中可能存在前向、后向和双向运动矢量。但是，不管情况多么复杂，MPEG－1/2 编码器工作时，总会按照某一种依据来选择合理的编码模式。因此，编码器在处理变化镜头时候，仍遵循某种合理的策略。例如当遇到跳变镜头时，帧间预测必然会失效，此时比较倾向于选择帧内编码模式，而在镜头边界之前后两侧，则前向、后向和双向预测模式选择一定会符合某种合理的模式。

设 R_P 表示P帧中没有运动矢量的宏块和有前向运动矢量的宏块在帧中的比率：

$$R_P = \frac{MV_{non}}{MV_F}$$

MV_F 表示帧中有前向运动矢量的宏块的个数，MV_{non} 表示没有运动矢量的宏块的总数。在正常情况下，P帧中的大部分宏块会参考它前面的参考帧进行编码以降低编码比特，因此 R_P 很小(一般 $1<R_P$)；当在P帧处发生镜头切换的时候，由于当前P帧和前面的参考帧内容相差很大，参考前面的参考帧编码反而会用较多的比特，大部分P帧中的宏块会采用帧内编码方式（没有运动矢量），因此 R_P 很大。通过 R_P 的值可以判断是否在P帧处发生了镜头切换。

同理，设 R_B 表示 B 帧中有后向运动矢量的宏块和有前向运动矢量的宏块在帧中的比率

$$R_B = \frac{MV_B}{MV_F}$$

在正常情况下，B帧中的大部分宏块会参考它前面的参考帧和后面的参考帧进行编码以降低编码比特，R_B 是一个在1附近波动的值；当在B帧处发生镜头切换的时候，由于当前B帧和前面的参考帧内容相差很大。参考前面的参考帧编码反而会用较多的比特，大部分B帧中的宏块会采用后面的参考帧帧做预测，而不参考前面的参考帧做预测，因此 R_B 很大。通过 R_B 的值可以判断是否在B帧处发生了镜头切换。

采用滑动窗口，可以判断在P帧和在B帧出现的可能的镜头切换。在本方法中，为 R_P 和 R_B 要分别建立各自的滑动窗口，在这里仅以 R_P 的滑动窗口为例。

设 W 是一个长度为 l 的滑动窗口，w_i（$i \in [0,l)$）是窗口中的一个元素（即 R_P 的值），窗口的期望为

$$E(W)=\frac{\sum_{i=0}^{l-1}W_i}{l}$$

FrameNo表示当前进入到滑动窗口的P帧的帧序号，ShotCount表示当前的镜头计数器，用下面的算法可以找到可能P帧处发生的切换。

滑动窗口算法，具体步骤如下：

(1)FrameNo=0, ShotCount=0;

(2)对窗口中的每一个元素，令w_i=InitialEle，（$i\in[0,l)$），$i=0$，计算$E(W)$；

(3)如果$R_p(FrameNo)>TH_P\times E(W)$,转（4），否则转（5）；

(4)Shot[ShotCount]=FrameNo, ShotCount=ShotCount+1,FrameNo=FrameNo+1，转(3)，否则，转（5）；

(5)$W(i \bmod l)=R_p(FrameNo)$,重新计算$E(W)$,$i=i+1$，FrameNo=FrameNo+1,如果FrameNo>Max_X_Frame，退出，否则转（3）；

在本发明中，对于P帧(即X为P)，InitialEle= 1000，滑动窗口宽度$l\in[8,10]$，TH_P=4。对于B帧(X为B)，InitialEle= 1000，滑动窗口宽度$l\in[18,22]$，TH_B=6。

对于可能在I帧处发生的镜头切换，用下列方法检测：

对宏块的每个块（8×8像素）进行DCT变换，得到64个DCT系数，其中第一个系数为直流系数，即DC系数。设$c(0,0)$为每个块的DC系数，则

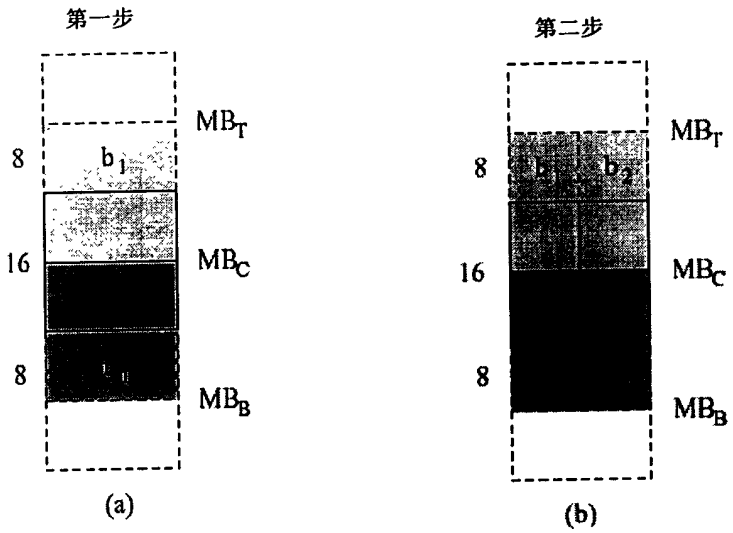
$$c(0,0)=\frac{\sum_{x=0}^{7}\sum_{y=0}^{7}c(x,y)}{64}$$

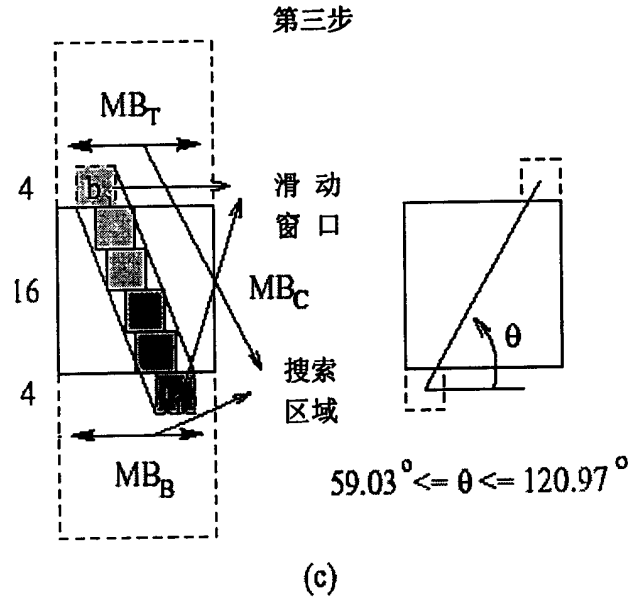
即DC系数为整个块的均值。由一帧中所有DC系数就构成了原始图像的DC图像，DC图像的分辨率是原始图像的1/64，却保留了原始图像的大部分信息。

为了进一步减少计算量，我们仅利用I帧亮度分量的DC图像来检测叠化。I帧的DC图像可以直接从压缩域中提取出来，设I_i和I_{i+1}为相邻的两个I帧，它们之间DC图像的差值定义为：

$$Diff\_I\_DC(I_i,I_{i+1})=\sum_{k=0}^{TotalBlock-1}|DC_k^i-DC_k^{i+1}|$$

在镜头发生叠化切换时，$Diff_I_DC(I_i,I_{i+1})$会出现一个明显峰值。同样，对$Diff_I_DC(I_i,I_{i+1})$也可以采用滑动窗口方法，在上述滑动窗口算法中，X为I，$InitialEle=\frac{width\times height\times 8}{256}$，滑动窗口宽度$l=4$，TH_I=3。


第一步
MB_T
8
b_1
16
MB_C
8
MB_B
(a)
第二步
MB_T
8
b_2
16
MB_C
8
MB_B
(b)


第三步
MB_T
4
b_1
滑 动
窗 口
16
MB_C
4
搜索
区域
MB_B
θ
$59.03^\circ <= \theta <= 120.97^\circ$
(c)

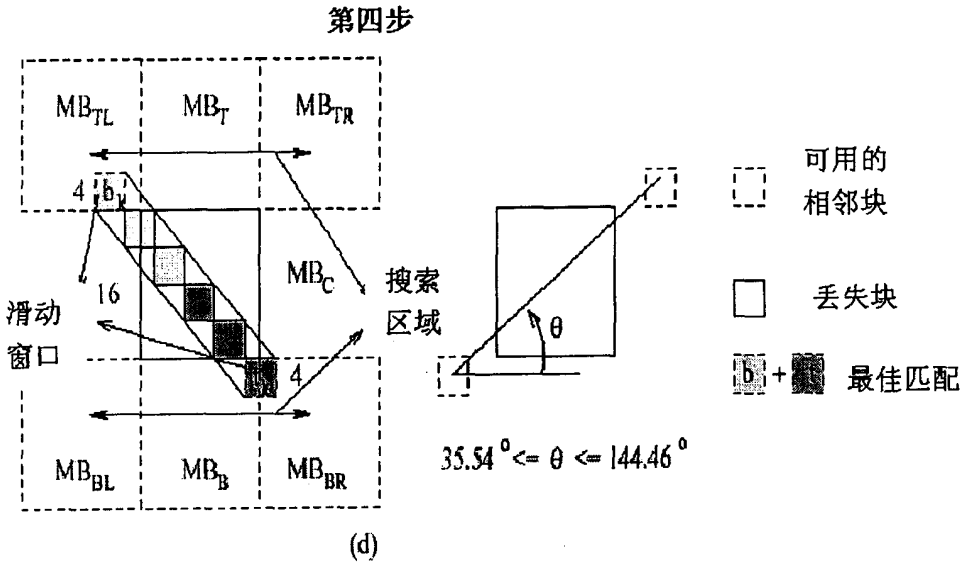

第四步
MB_TL
MB_T
MB_TR
4
16
MB_C
搜索区域
滑动窗口
4
MB_BL
MB_B
MB_BR
(d)
θ
35.54°<= θ <= 144.46°
可用的相邻块
丢失块
最佳匹配

图 1

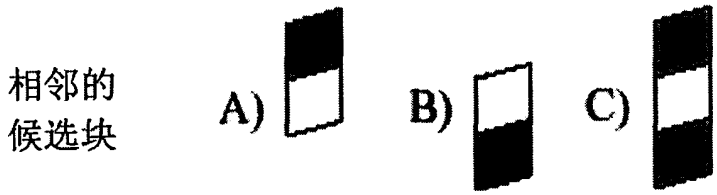

相邻的候选块
A)
B)
C)

图 2