will.v

H.264 学习笔记（三）

Inter Prediction

1、 H.264 帧间预测是利用已编码视频帧/场和基于块的运动补偿的预测模式。

2、 搜索算法用来找到合适的运动矢量。

3、 首先根据当前帧和参考帧求出宏快（或者子块）的MV,MV本来足够说明最佳匹配块的位置，但是对MV的编码需要较大的数据量；根据相邻块之间的较强相关性，相邻块的MV之间的残差较小，这样，就可以利用以前块的MV来预测当前的MV，二者差值为MVD=MV-MVp，MVD相对来讲数据量较小。通过MV找到当前块的最佳匹配块，这些最佳匹配块重建成为预测帧，把当前帧和预测帧的差值进行变换和量化。

4、 mvA: left块；mvB: up块；mvC： up-right块；mvD： up-left块。

5、 参考帧的颗粒度只到 8*8,即：8*8以下的分割所用的参考帧相同。

6、 MV预测过程：

1) 确定相邻块 MV： 预测以宏块分割（或亚宏块分割，如果宏块存在亚分割）为单位，同一个宏块分割（或亚宏块分割）内所有 4*4 块 MV 预测值相同。以每个宏块分割（或亚宏块分割）的左上角像素 pixel1 和右上角像素 pixel2 为参考点来确定相邻块则：

pixel1 左侧相邻像素所在 4*4 块为当前宏块分割（或亚宏块分割）的相邻块 A

pixel1 上方相邻像素所在 4*4 块为当前宏块分割（或亚宏块分割）的相邻块 B

pixel2 右上对角线像素所在 4*4 块为当前宏块分割（或亚宏块分割）的相邻块 C

pixel1 左上对角线像素所在 4*4 块为当前宏块分割（或亚宏块分割）的相邻块 D

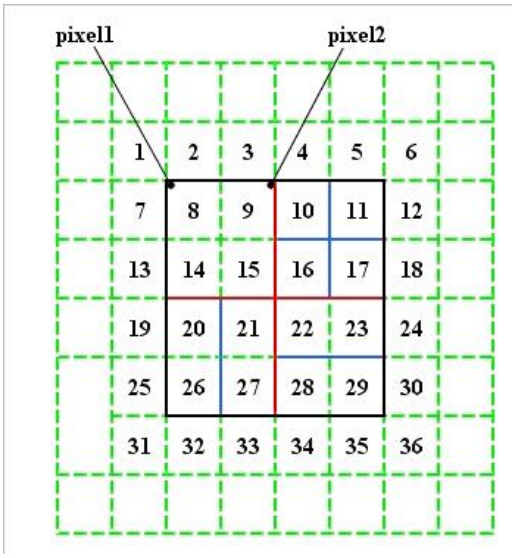

图2-1

以最复杂的 8*8 宏块分割类型为例（此时只存在亚宏块分割），分析如下：

假设图中黑色框表示宏块、每个绿色框表示一个 4*4 块、每个红色框表示一个 8*8 块。当前宏块的宏块分割模式为 8*8（如图中红色线），其亚宏块分割模式分别为：第一个 8*8 块为 8*8，第二个 8*8 块为 4*4（如图中蓝色线），第三个 8*8 块为 4*8（如图中蓝色线），第四个 8*8 块为 8*4（如图中蓝色线）。则按照上述方法来确定相邻块的方法如下：

第一个预测对象为第一个 8*8 块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 7 号 4*4 块，B 为 2 号 4*4 块，C 为 4 号 4*4 块，D 为 1 号 4*4 块。9、14、15 与 8 具有相同 MV 预测值

第二个预测对象为第二个 8*8 块的第一个 4*4 块，即 10 号块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 9 号 4*4块，B 为 4 号4*4块，C 为 5 号 4*4 块， D 为 3 号 4*4 块

第三个预测对象为第二个 8*8 块的第二个 4*4 块，即 11 号块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 10 号 4*4块，B 为 5 号4*4块，C 为 6 号 4*4 块，D 为 4 号 4*4 块

第四个预测对象为第二个 8*8 块的第三个 4*4 块，即 16 号块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 15 号 4*4块，B 为 10 号4*4块，C 为 11 号 4*4 块，D 为 9 号 4*4 块

第五个预测对象为第二个 8*8 块的第四个 4*4 块，即 17 号块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 16 号 4*4块，B 为 11 号4*4块，C 为 12 号 4*4 块，D 为 10 号 4*4 块

第六个预测对象为第三个 8*8 块的第一个 4*8 块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 19 号 4*4 块，B 为 14 号 4*4 块，C 为 15 号 4*4 块，D 为 13 号 4*4 块。26 与 20 具有相同 MV 预测值

第七个预测对象为第三个 8*8 块的第二个 4*8 块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 20 号 4*4 块，B 为 15 号 4*4 块，C 为 16 号 4*4 块，D 为 14 号 4*4 块。27 与 21 具有相同 MV 预测值

第八个预测对象为第四个 8*8 块的第一个 8*4 块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 21 号 4*4 块，B 为 16 号 4*4 块，C 为 18 号 4*4 块，D 为 15 号 4*4 块。23 与 22 具有相同 MV 预测值

第九个预测对象为第四个 8*8 块的第二个 8*4 块，以其左上角像素 pixel1 和右上角像素 pixel2 为参考点，则：A 为 27 号 4*4 块，B 为

22 号 4*4 块，C 为 24 号 4*4 块，D 为 21 号 4*4 块。29 与 28 具有相同 MV 预测值

2) 确定 A、B、C 的可用性：根据 A、B、C 所在宏块是否存在或者是否允许参与预测来判断。如果 C 不可用，采用 D 代替 C。

3) 预测 MV：

(1)、如果 A、B、C 三个参考块中只有一个与当前预测对象为同一参考帧， 则选取该参考块的 MV 作为最终 MV 预测值。

(2)、当前宏块是否为 8*16 或者 16*8 分割：

①、如果当前宏块为 8*16 分割类型：对于左边 8*16 分割，如果 A 与当前分割为同一参考帧，则采用A的 MV 为该分割的最终 MV 预测值。对于右边 8*16 分割，如果 C 与当前分割为同一参考帧，则采用 C 的 MV 为该分割的最终 MV 预测值。

②、如果当前宏块为 16*8 分割类型：对于上边 16*8 分割，如果 B 与当前分割为同一参考帧，则采用 B 的 MV 为该分割的最终 MV 预测值。对于下边 16*8 分割，如果 A 与当前分割为同一参考帧，则采用 A 的 MV 为该分割的最终 MV 预测值。

(3)、其余情况并且 B、C 中有一个可用或者两者都可用，则采用中值预测（取 A、B、C 三者中MV的中值为最终 MV 预测值）。

(4)、其余情况并且 B、C 皆不可用，则采用 A 的 MV 为最终 MV 预测值。

4) 注意：

(1)、对于不可用的相邻块，其 MV 仍然可能参与 MV 预测，但其值为 0。例如：A 不可用，B、C 可用，则最终可能仍然是在 A、B、C 中取中值，但此时 A 的 MV 为 0；

(2)、对于不可用的相邻块，其参考帧索引被设置为 -1，即必然与当前预测对象非同一参考帧；

7、 菱形算法（DS）：DS算法采用两种两种搜索模板，分别是有9个监测点的大模板和有5个监测点的小模板。大菱形搜索模式由一个中心点和它周围的8个搜索点构成，步长为2。这9个点组成一个菱形，小菱形搜索模式由5个搜索点构成，步长为1。搜索步骤：第1步，最初的大菱形搜索模式以搜索窗口中心为中心点，计算大菱形的9个搜索点的误差函数。若误差函数最小的搜索点位于中心，则跳到第3步；否则跳到第2步。第2步，以第1步搜索的9个点中最小点为大菱形的中心点，计算大菱形的9个搜索点的误差函数。若误差函数最小的搜索点位于中心，则跳到第3步；否则，跳到第2步。第3步，以上一步搜索的9个点中的最小点为中心点，计算小菱形的5个点的误差函数。误差函数最小的块为最佳匹配快。

8、 EPZS算法：

① 使用多个预测运动矢量，其中包括中值运动矢量、空域邻块运动矢量、时域（前一帧）对应块运动矢量...并记录此时的最优点和次优点。

② 从最优点开始，使用菱形模板反复搜索。

③ 如果有必要的话，在第一步中的次优点开始搜索，反复搜索。

以上每一步中都采用门限值进行判断，如果小于某门限，则停止搜索以 减小运算量。

9、 代码中，mv值是实际值的四倍，后两位分别用来标示是否使用1/2精度或1/4精度。

10、 亚像素位置的亮度和色度像素并不存在于参考图像中，需利用邻近已编码点进行内插而得。如果MV 的垂直和水平分量为整数，参考块相应像素实际存在。如果其中一个或两个为分数，预测像素通过参考帧中相应像素内插获得。

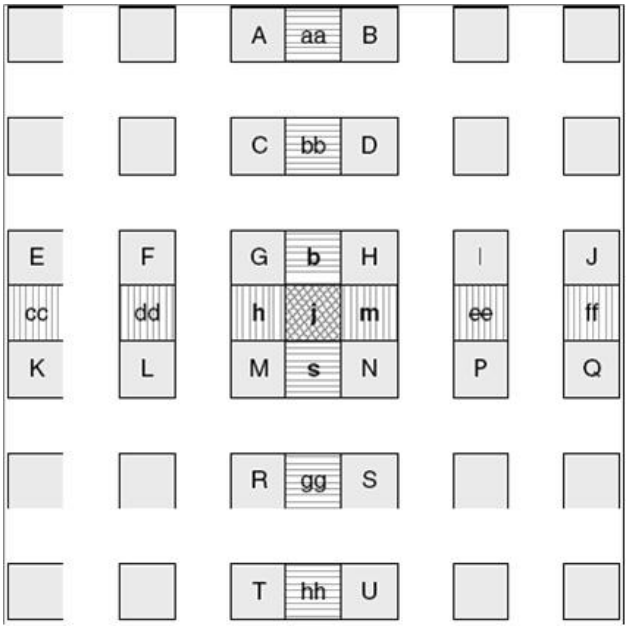

图2-2

内插像素生成：

首先生成参考图像亮度成分半像素像素。半像素点（如b,h,m）通过对相应整像素点进行6 抽头滤波得出，权重为（1/32 ,-5/32 ,5/8, 5/8, -5/32, 1/32）。b 计算如下：

类似的，h 由A、C、G、M、R、T 滤波得出。一旦邻近（垂直或水平方向）整像素点的所有像素都计算出，剩余的半像素点便可以通过对 6 个垂直或水平方向的半像素点滤波而得。例如，j 由cc, dd, h,m,ee,ff 滤波得出。

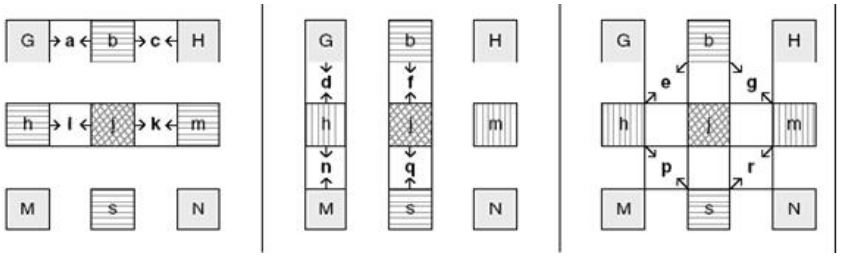

图2-3

半像素点计算出来以后，1/4 像素点就可通过线性内插得出，如图6.23 所示。1/4 像素点（如a,c, i, k, d, f, n, q）由邻近像素内插而得，如:

剩余1/4 像素点（p, r）由一对对角半像素点线性内插得出。如，e 由b 和h 获得。

11、 色度像素需要1/8 精度地MV，也同样通过整像素地线性内插得出，如图2-4所示：

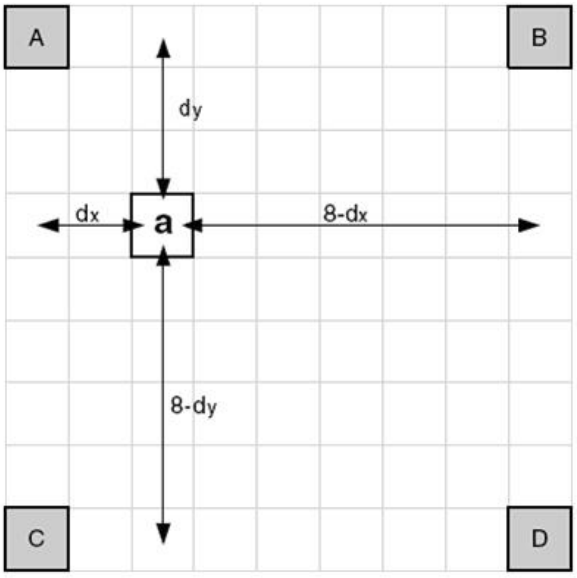

图2-4

其中，

12、 选最佳MV时，用SAD与阈值比较，来判断是否提前退出，cost=SAD值+ m_lambda_me *mvd bits的cost；mvd为当前mv-mvp。

13、 选出最佳MV后，对其进行菱形搜索，此时cost算法为cost1=SAD值+ lambda*mv的cost。返回的cost2=cost1- m_lambda_me * mvd bits的cost。

Cost3=dist（预测块与原块）+ m_lambda_motion * costMbMotion。最终cost取cost2与cost3中的最小值。

标签: H.264 笔记 inter 帧间预测

will.v
关注 - 3
粉丝 - 5
+加关注

« 上一篇：H.264 学习笔记（二）
» 下一篇：H.264 学习笔记（四）

posted @ 2012-02-24 20:45 will.v 阅读(413) 评论(0) 编辑 收藏