H.264学习笔记3——帧间预测 - DwyaneTalk - 博客园

帧间预测主要包括运动估计（运动搜索方法、运动估计准则、亚像素插值和运动矢量估计）和运动补偿。

对于H.264，是对16x16的亮度块和8x8的色度块进行帧间预测编码。

A、树状结构分块

H.264的宏块，对于16x16的亮度宏块，可以分成16x16、16x8、8x16和8x8的子块进行帧间预测。对于8x8的块（亚宏块，亮度和色度），往下又可以分成8x8、8x4、4x8、4x4的子块。在运动估计中，每一种分割都需要尝试，并计算出运动搜索结果的代价，选择最小代价的分割方式进行预测编码。

B、运动估计准则

运动估计就是在搜索范围内寻找最佳估计块，使得预测块与当前块的残差数据尽量小，这样就保证编码的代价尽量小。对于MxN的像素快，s(x,y)表示当前像素值，z(x,y)表示备选预测像素值，x=1……M,y=1……N。则有下面的运动估计准则：

1、SAD(sum of absolute difference)：绝对误差。MAE：平均绝对差值

SAD = sum(|s(x,y)-z(x,y)|)　　MAE = (1/MN)*SAD

2、SATD

3、SSD(sum of squared difference)：差值的平方和。MSE：平均平方误差

SSD = sum((s(x,y)-z(x,y)^2))　　MSE = (1/MN)*SSD

可以选择上面的估计标准，计算出的值越小，说明编码代价越小。

C、运动搜索方法

运动搜索就是在允许的搜索范围（一般是上下左右各一个子块大小）内查找最佳匹配块的过程，主要有全局搜索和快速搜索。

1、全局搜索：最简单的将所有可能进行搜索比较，总是能找到搜索范围内的最佳匹配块，但是效率太低。

2、快搜索算法：每种算法的过程各不相同，主要通过尽量避开不太可能是最佳匹配块的位置，从而提升搜索效率。过程就是：

》1、确定搜索起始点；

》2、判断该点有没有达到最佳匹配块的要求和能不能进一步继续搜索。

》3、按照搜索规则，以起始点周围点为新的起始点进行递归搜索。

常见的快速搜索算法有：三步法、二维对数法、交叉法、菱形法等，具体每种搜索方法可以参考书籍或者百度。

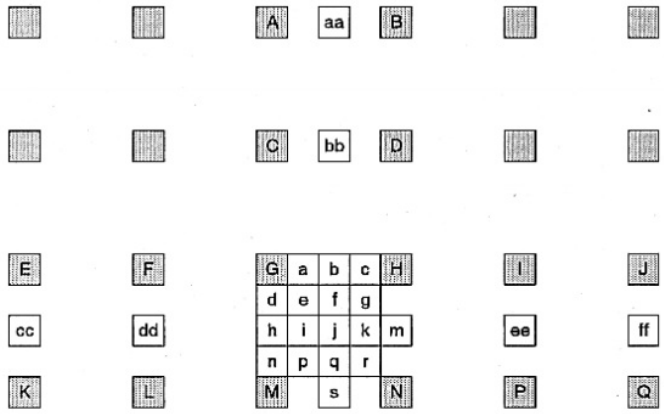
D、树状分级搜索和亚像素估计

在运动估计时，为了提高估计精度，通常会采用1/2、1/4和1/8的像素精度进行估计，但是全部估计都这样操作会产生很大的性能开销，所以一般采用树状分级搜索。

树状分级搜索就是在进行在进行运动估计时，先以整像素精度进行搜索，找到最佳匹配块之后，再在该位置周围进行1/2像素精度的搜索找到最佳匹配点。如果有需要的话，可以继续在1/2像素精度的最佳匹配点周围进行1/4像素精度的搜索，寻找最佳匹配点。

在H.264中，对亮度和色度块的估计，分别支持1/4和1/8像素精度的运动估计。

1、亮度的1/2和1/4像素插值：亮度亚像素差值如图一



图一：亮度块的亚像素插值　　图二：色度块的亚像素插值

如图：图中的灰色点是整数像素点，aa,bb,cc,dd,ee,ff,gg,hh和b,h,s,m,j是1/2像素，其他都是1/4像素。计算过程是先用（1,-5,20,20,-5,1）的六抽头滤波器进行1/2像素插值，然后再通过临近像素插值的方法计算1/4像素的插值。具体如下：

水平半像素：如 b = (E - 5F + 20G + 20H - 5I + J), b = Clip1((b+16) >> 5)；Clip1的作用是限制结果在0~255，右移5相当于除以32，加16是为了结果的四舍五入。

垂直半像素：如 h = (A - 5C + 20G + 20M - 5R + T), h = Clip1((h+16) >> 5)；

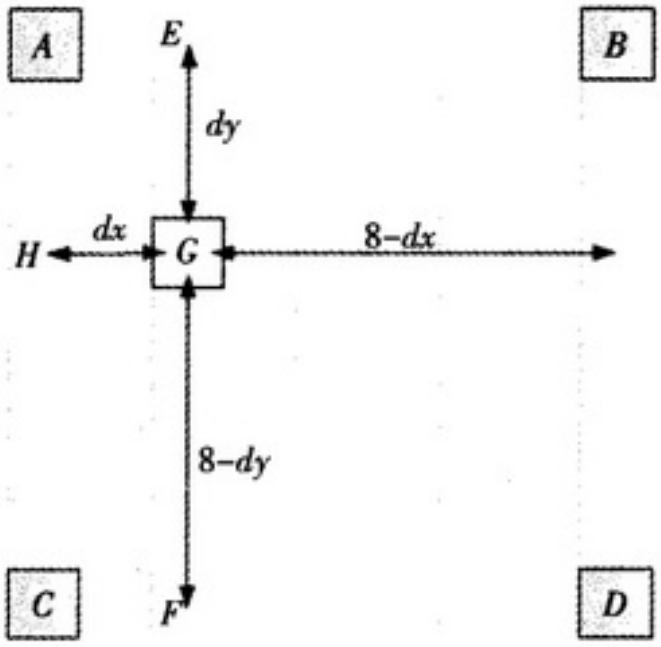
对角半像素：如 j = (cc - 5dd + 20h + 20m - 5ee + ff) = (aa - 5bb + 20b + 20q - 5gg + hh) ,j = Clip1((j+16) >> 5)； 即：水平和垂直方向计算结果相同（可以自行展开计算验证）。

水平1/4像素：如 a = (G + b +1) >> 1;
i = (h + j + 1) >> 1;

垂直1/4像素：如 d = (G + h +1) >> 1;
f = (b + j + 1) >> 1;

对角1/4像素：如 e = (h + b +1) >> 1;　g = (b + m + 1) >> 1;　p = (h + s + 1) >>1;　r = (s + m + 1) >>1;

2、色度的1/8像素插值：色度块是采用二次线型的1/8像素插值，如图二



如图：就是利用待查亚像素G周围的整数像素（A、B、C、D）来加权计算G的像素值。整像素点距离G越近，其权值越大。计算如下：

G = ((8-dx)*(8-dy)*A + (dx)*(8-dy)*B + (8-dx)*(dy)*C + (dx)*(dy)*D + 32) >> 6; 其中dx、dy分别为G相对A的水平和垂直距离（以1/8像素为单位1距离），取值范围是1~7。

E、B帧的预测

B帧是双向预测，分别参考List0和List1两个参考帧列表进行前向和后向预测，这里的前和后是针对播放顺序而不是编码顺序。H.264编码是以GOP分组处理，每个GOP的编码结构有III……、IPP……、IBB……P和分层B帧结构。其中第一种一般不使用，编码效率太低；第二种是属于H.264的基本档次。第三、四种属于主要档次和扩展档次。其中第三种（普通B帧）会产生较大的编码延时。

B宏块预测有4种模式：直接模式（Direct）、双向模式（Bipred）、List0和List1。其中16x8和8x16大小的块只能使用Direct、List0和List1，其他子块大小可以使用所有的模式。

双向预测：

从List0和List1两个参考帧列表中选择最佳匹配块进行预测。过程如下：

》在List0和List1中查找最佳匹配块，得到运动矢量MV（MV1，MV2）。

》利用临近块的同方向MV，估计当前块的两个方向的MV预测值MVp（MVp1，MVp2，方法见后面：MV的编码）。PS：该步骤可以和上一步对换，即：先估计得到MVp1和MVp2，然后以MVp1和MVp2为起点进行搜索得到运动矢量MV1和MV2。

》得到MV和MVp，就可以计算两个MVD（MVD1和MVD2），然后对MVD进行编码。

》得到两个预测参考块和运动矢量后，就可以计算像素块的像素预测值。

pred（i，j） = (P1 * pred_L0(i,j) + P2 * (pred_L1(i,j))) / (P1 + P2);　　// 其中P1和P2是两个方向的参考帧的权重。

List0和List1模式：这两种模式和双向模式类似，只不过只进行一个方向的预测 。

直接模式：直接模式应用于16x16、8x8以及8x8块中的所有子块。直接模式只编码预测残差，而不编码运动矢量（或MVD）和参考帧序列号，运动矢量和参考帧序列号在解码端通过计算前、后向MV，利用前后向MV得到像素预测值。

有两种计算MV的方法：时间模式和空间模式。

时间模式：基于假设“被预测的块在前后两个参考块间是均匀运动的”，所以场景切换和非均匀运动的块，预测效果不好。

运动估计时，假设当前B块在List0和List1中参考帧分别为F0和F1，对应的最佳匹配块（预测块）分别为B0和B1。如果在之前对B1块进行帧间预测时，B0是B1的前向最佳匹配块，即：存在当前块在List0中的参考块B0相对于当前块在List中的参考块B1的运动矢量MV，那么就可以根据这个运动矢量和当前帧与F0、F1之间的帧间隔（记为d0,d1）算出前向和后向的运动矢量MV0和MV1。

如：MV=（2.5,5），当前帧与F0和F1分别间隔2帧和1帧，即：d0=3,d1=2;于是MV0 = d0/(d0+d1) * MV = (1.5,3)，MV1 = -d1 / (d0+d1) * MV =(-1,-2)。

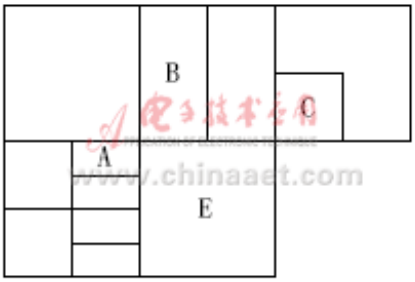
空间模式：利用当前帧的相邻块的运动信息估计当前块的MV和参考帧序号，主要过程包括“参考帧选择”和“运动矢量选择”。

》参考帧选择：利用当前块的邻近块A、B、C（位置见后面MV的编码）的参考帧中非负帧号最小的帧作为当前块的参考帧，这是为了选择离当前块最近的帧作为参考帧。如果有一个方向上A、B、C都没有参考帧，那么就采用单项帧间预测。如果A、B、C在两个方向都没有参考帧（即A、B、C都是帧内预测），那么当前块分别选择两个方向离当前帧最近的参考帧为当前块的参考帧。

》运动矢量选择：若List1中第一帧（当前帧的播放顺序后一帧）对应位置块的运动适量MV1和MV2小于1/4像素（即：当前块到下一帧运动的比较少），且该帧为短期参考帧，且当前块的某个方向的参考帧帧号为0，则该方向的MV为0。如果不满足，则按照后面MV的编码中描述的方法计算两个方向（A、B、C的运动适量的**中值**，不是均值）的MVP作为当前块的MV

F、MV的编码

通过上面的树状分块结构，针对各种分块大小都进行一次帧间预测。每种分块的帧间预测，通过树状像素精度分级搜索，先按照整像素精度找到最佳匹配块，然后在进一步按照1/2、1/4像素精度寻找更加准确的最佳匹配块。寻找最佳匹配块主要是通过快速搜索算法，按照某种搜索准则判断最佳匹配块。上述操作完成后，找到各种分块结构代价最小的最佳匹配块，从而根据最佳匹配块和当前块的位置，得到运动矢量（MV）。通过运动矢量和运动残差（参考块和当前块的像素差值）就可以在解码端还原图像数据。但是如果之间编码MV会产生很大的码流代价（16x16MB分成多个子块产生多个MV、采用亚像素、MV包含row和col两个参数），因为实际操作中往往是根据周围的块，对MV进行预测得到MVp，然后编码运动矢量的实际值与预测值的差值MVD=MV-MVp。



如上图：当前块（任意子块大小）的运动适量预测值有当前块的左、上、右上的块A、B、C（任意字块大小）进行预测。预测规则如下：

》对于非16x8和8x16的子块，运动矢量的预测值MVp 为A、B、C的运动矢量的中值。MVp = mid(MVA, MVB, MVC)。

》对于16x8的子块（即图中的E为上下两个16x8的子块），上面子块的MV预测值是块B的MVB，下面子块的MV预测值设计块A的MVA。

》对于8x16的子块（即图中的E为左右两个16x8的子块），左边子块的MV预测值是块A的MVA，右边子块的MV预测值设计块C的MVC。

》在进行上面的MV计算时，还要满足下面的限制条件：

1、只有当当前块E的参考帧和临近块（A、B、C）的参考帧为同一帧，那么才可以使用MVA、MVB、和MVC进行预测；

2、如果MVC不可用，则用当前块的左上边块的运动矢量MVD代替MVC；

3、如果MVA、MVB和MVC中没有可用的，那么不进行运动矢量预测，直接编码当前块的运动矢量MV；

4、如果MVA、MVB和MVC中只有一个可用的，那么MVP就是该可用的临近块运动矢量；

5、如果MVA、MVB和MVC中有两个可用的，那么将另一个不可用的当作0，然后按照3个都可用的策略计算MVP。

Skip模式：H.264中为了降低码率采用的特殊编码模式，是针对宏块（16x16）编码时，“既不传输运动矢量残差（MVD）、也不传输像素块残差”的编码方式。包括P_Skip和B_Skip。

P_Skip：

编码时：如果参考帧是List0中的第一帧，运动矢量和运动矢量的预测值相同（MVD为0），且残差系数通过变换量化后成为0或者通过某种策略（率失真优化等）舍弃。那么则只需要标记为P_Skip和传输运动矢量预测值MVP。解码时，就可以的MV=MVP，然后用预测像素值作为解码像素值。

B_Skip：是B块直接编码的一种特殊形式。

编码时：如果残差系数通过变化量化后成为0或者通过某种策略舍弃，那么则只需标记为B_Skip模式，也不需要传输MVP。解码时则按照B块的直接模式计算出参考帧号和相应的运动矢量，然后参考像素值作为当前像素的值。