2017-04-28

1.TComMV.h(在该文件中加入了相应的变量)

Position=0时，为在相同面上；

Position=1时，在左边面上；

postion=2时，在上边面上；

Position=3时，在右边面上；

Position=4时，在下边面上；

2.在构建mergecandidate列表时，需要考虑到，如果加入列表中的MV和其它MV虽然相同，但是来自于不同的面，那么就不需要在进行判断了，直接加入即可。

3.需要保证加入后的MV使得其为当前面上的mv了，需要明确标识出。

4.需要找到所有可能使用MV的地方，然后将其改掉，最好的办法就是找motioncompentation，这是最好的办法，将其全部改掉，即可。同时需要注意的是，在块划分决策的时候确实使用了这个，但是在重建帧的时候以及在真正编码到比特流的时候有没有又再次用到了这个东西？？？

5. TEncCu.cpp 被修改了

6. TComMotionInfo.h被修改了

7. setAll存储的时候是按照Z扫描顺序来存储的。所以自己在存储时，可能要用到数组转换功能。iPartAddr为setALL中的一个参数，其代表的是一种Z扫描顺序，但具体应该是一个PU在CU内部的绝对偏差，而不是在一个CTU内部的绝对偏差（编码端）。

8. 在解码端的xDecodeCU中pcCU指的是一个CTU，所以每次传到setAllMvField中的是一个绝对地址。

9. 在编码端进行运动矢量的存储时，每个CU的pcMV都是独立的，在进行从小的CU到大的CU进行转移时，会进行copyFrom或者是copyTo函数中进行。

10. xWeightedPredictionUni对于该函数来说，输入进去的uiPartAddr, iWidth, iHeight应该就是实际的Addr还有宽和高，但是对于xPredInterUni输入进去的uiPartAddr, iWidth, iHeight应该要变掉。

11．在修改时首先令左上角的MV为一个有标记的，等到运动补偿完之后，再将这个有标记的MV给去除标记，一定要去除标记才可以。

12. m\_uiCUPelX = ( ctuRsAddr % pcPic->getFrameWidthInCtus() ) \* maxCUWidth;得到的是一个CTU的左上角位置的坐标。

总结：在找一个相关的函数中，最好的办法是去找到这个类，在这个类中会有相关的函数，通过名字可以发现其相应的含义。

在下方的那个点按照左侧的进行计算了

2017-05-06

1. deriveLeftBottomIdx( partIdx, partIdxLB )

其中partIdxLB得到的是当前PU内部左下角的4\*4小块在一个CTU内部的Z地址。

2017-05-10

1. xGetTemplateCost中使用了xPredInterBlk这一函数，而xGetTemplateCost在判决最好的搜索起点时使用了。
2. 一个CU中只有一个AMVPInfo，每次使用时都会被覆盖掉。

AMVPInfo\* pcAMVPInfo = pcCU->getCUMvField(eRefPicList)->getAMVPInfo();

3. aaiMvpIdx[iRefList][iRefIdxTemp] = pcCU->getMVPIdx(eRefPicList, uiPartAddr);//作为起点的MV的idx；

aaiMvpNum[iRefList][iRefIdxTemp] = pcCU->getMVPNum(eRefPicList, uiPartAddr);//iN的值，即列表中有几个可用的MV；

2017-05-17

typedef Distortion (\*FpDistFunc) (DistParam\*) 其中FpDistFunc为一个函数指针，该函数返回的类型是Distortion，其中参数为DistParam\*

在merge时使用的是SATD，Hadamard变换获得SATD的值来代替DCT或DST。 在判决AMVP和merge时使用的cost不是RDcost，不是经过实际变换量化后的值。

在整像素用的是SAD在分像素用的是SATD？？？

2017-05-19

1. 在运动搜索中，首先进行单向列表的搜索，即首先进行每个列表的搜索，从中选取每个列表中最好的一个运动矢量以及参考帧，然后比较两个列表中最小的cost，然后cost最小的一个将其固定住，然后继续遍历cost较大的list，从中选取最好的作为双向预测的模式。
2. xSetSearchRange ( pcCU, cMvPred, iSrchRng, cMvSrchRngLT, cMvSrchRngRB );//

cMvSrchRngLT保存的是一个MV类型的变量，其中hor分量表示的是cMvPred的水平分量减去iSrchRng，同时还有一定的缩放，由于给的是64或者96，还需要左移两位，而竖直分量为cMvPred的竖直分量减去iSrchRng。

cMvSrchRngRB的水平和竖直分量则是加上对应的iSrchRng。

1. pcPatternKey 的作用就是保存当前块当前的像素地址（左上角）；当前块的宽度和高度，亮度分量一帧的宽度；当前视频的比特数（8或10），之后再赋给后面的m\_cDistParam来计算SAD
2. xPatternSearchFast中只用了case MESEARCH\_DIAMOND目前的配置中只有xTZSearch这个选项。pIntegerMv2Nx2NPred这个变量是之前就有的，可能在之后有用。

xTZSearch( pcCU, pcPatternKey, piRefY, iRefStride, pcMvSrchRngLT, pcMvSrchRngRB, rcMv, ruiSAD, pIntegerMv2Nx2NPred, false );

1. 在xTZSearch函数中，首先在进行搜索之前将MV减半，作为搜索的起点
   1. if ( bTestOtherPredictedMV )满足该条件时则进行其他的预测MV的尝试，在当前配置下，该过程不会进入。
   2. xTZSearchHelp
   3. 在xTZSearch中第一次调用xTZSearchHelp函数的目的就是set rcMv (Median predictor) as start point and as best point ；第二次调用xTZSearchHelp函数的目的就是test whether zero Mv is better start point than Median predictor，并没有搜索的任务在里面只是计算一下cost而已
   4. xTZSearchHelp在该函数下，首先进行的是参考块在预测MV减半时的像素位置的获取；

setDistParam（每次修改都应当修改这个位置，因为给其块可能不一样）设置的是将参考像素的信息和当前像素的信息都给m\_cDistParam

* 1. 无论是xTZ8PointDiamondSearch还是xTZ2PointSearch都是调用xTZSearchHelp
  2. **最重要的结构体就是cStruct里面包含最好的MV的两个分量；但是是否是要划子块不能再该结构体中看出；**

1. 进行完整像素的搜索之后进行分像素搜索，首先将参考帧的信息给一个TComPattern类的对象cPatternRoi，然后将参考帧中的1/2像素插值出来，其中dstStride = m\_filteredBlock[0][0].getStride(COMPONENT\_Y)基本为0x50。搜索的任务在xPatternRefinement（）中完成的
2. PredMV 一定是在列表中没有修改过的MV，并且搜索范围也应当是在没有修改过的MV之内。
3. 从上到下一次想一下如何修改这个整体的流程，先进行自己的修改，然后再是其它的修改；