HM 代码解释

Jinze Chen

HM 代码

```
//Pic->Slice->CU

class TComSlice{
  TComPic* m_pcPic;
  UInt m_sliceCurStartCtuTsAddr;
};

class TComPic{
  TComPicYuv* m_apcPicYuv[3]; // picRec, picOrg
  TComPicSym m_picSym;
};

class TComPicSym{
  TComDataCU** m_pictureCtuArray;
  std::deque<TComSlice*> m_apSlices;
};

class TComPicYuv{
  Pel* m_piPicOrg[3];
  Void dump(string filename, Bitdepth bitdepth);
};//存储 PicYuv, 算地址, copy, dump

class TComDataCU{
  TComPic* m_pcPic;
  TComPic* m_pcSlice;
  Void copyToPic(UChar uiDepth);

  UInt          m_ctuRsAddr;                                ///< CTU (also known as LCU) address
  UInt          m_absZIdxInCtu;                             ///< absolute address in a CTU. It's
  UInt          m_uiCUPelX;                                 ///< CU position in a pixel (X)
  UInt          m_uiCUPelY;                                 ///< CU position in a pixel (Y)
  UChar*        m_puhWidth;                                 ///< array of widths
  UChar*        m_puhHeight;                                ///< array of heights
```

```
  UChar*        m_puhDepth;                              ///< array of depths
  // TsAddr => Tile scan addr
  // RsAddr => Raster scan addr


  // 编码相关:
  Double& getTotalCost(); // RD cost
  Distortion& getTotalDistrotion(); // get D
  UInt& getTotalBits(); // get R
  Void initSubCU(TComDataCU* pcCU, UInt uiPartUnitIdx, UInt uiDepth, Int qp); // 初始化 subc

  // 写码流相关
  UChar* m_puhCbf[MAX_NUM_COMPONENT]; // intra mode assume rqt_root_cbf==1
  UChar getQtRootCbf(UInt uiIdx); // return getCbf(Y) || getCbf(Cb) || getCbf(Cr)
  UChar* m_puhTransformSkip[MAX_NUM_COMPONENT];
  TCoeff* m_pcTrCoeff[MAX_NUM_COMPONENT]; // 写码流时获取该信息
};

class TComTU{
  UInt mCuDepth; //<CU 是 TU 的根节点, 记录 CU 深度
  UInt mTrDepthRelCU[MAX_NUM_COMPONENT]; //< 记录 TU 相对于 CU 的深度
  UInt mSection;//< 如果进行劈分, 那么劈分后处理 SubCU 的 index
  TU_SPLIT_MODE mSplitMode;//< the split mode
  TComRectangle mRect[MAX_NUM_COMPONENT]; //< 当前 TU 的位置信息 Bool mCodeAll[MAX_NUM_COMPON
  UInt mOrigWidth[MAX_NUM_COMPONENT];
  UInt mOffsets[MAX_NUM_COMPONENT];//<cur TU 与 CU 左上角的偏移量, 比如 TU 为 8*8, 那么处理第二个
  UInt mAbsPartIdxCU;//<the abs index of CU in LCU
  UInt mAbsPartIdxTURelCU;//<the abs index of TU in cur CU
  UInt mAbsPartIdxStep;//<if split, the added index
  TComDataCU *mpcCU;
  UInt mLog2TrLumaSize;//< relative to the size of Cur TU
  TComTU *mpParent; //< 如果劈分进行递归, 对应父节点
};

class TComPrediction{
  Void initIntraPatternChType(TComTU &rTu, const ComponentID compID, const Bool bFilterRefSa
  fillReferenceSamples(); // fill reference sample from pcPicYuvRec
  // then filterReferenceSample
  }
};

class TEncCU{
// TComDataCU** 含义: 不同 Depth 申请对应的内存空间 (heap), 所有 depth 的 bestCU 只有一个
  TComDataCU**          m_ppcBestCU;      ///< Best CUs in each depth
  TComDataCU**          m_ppcTempCU;      ///< Temporary CUs in each depth
  UChar                 m_uhTotalDepth;
```

```
// TComYuv** 含义：不同 Depth 申请合适的内存空间，所有 depth 的 bestPred 只有一个
  TComYuv**                m_ppcPredYuvBest; ///< Best Prediction Yuv for each depth
  TComYuv**                m_ppcResiYuvBest; ///< Best Residual Yuv for each depth
  TComYuv**                m_ppcRecoYuvBest; ///< Best Reconstruction Yuv for each depth
  TComYuv**                m_ppcPredYuvTemp; ///< Temporary Prediction Yuv for each depth
  TComYuv**                m_ppcResiYuvTemp; ///< Temporary Residual Yuv for each depth
  TComYuv**                m_ppcRecoYuvTemp; ///< Temporary Reconstruction Yuv for each depth
  TComYuv**                m_ppcOrigYuv;     ///< Original Yuv for each depth
}
```

Intra fast mode decision

1. SATD ⇒ 8/3 + MPM[3] ⇒ 8~11 种 mode
2. max Transform RD ⇒ 确定 1 个 mode
3. full RQT ⇒ 对该 mode 进行划分，决定 RQT

4×4 块 Intra mode ⇒ 仅应用在 YUV420 情况下的 UV 处理?

周围块为 inter 块时认为周围块 not available

intra 以 TU 为单位

intra reference pixels

constrained intra: pps 选项

bool[4*16+1]: is_available array

copy 左侧像素

pcCU->getPic()->getPicYuvRec(): reference pixels

ROI(上方或左侧参考像素长度 (129,65...ext)) EXT(intra 模式下的参考像素)

filtered prediction

目前很大的问题

如何编码/解码 ⇒ 填充像素值 128

编码流程

编码一个 CU: 满足条件 (luma TU width == 4 && 其他条件) 填充 DC(或角度)
进行编码 ⇒ 其他条件可以类似为每个 TU 的 RD< 原始 RD/4

理论基础: intra 预测情况下 CTU 间参考只体现在 reference sample

仅下一个 CTU 在预测时可受到 AI 的加成

xCompressCU: 递归获得 CU mode decision 与 RQT 结果 xCheckRDCostIntra{ estIntraPredLumaQT; estIntraPredChromaQT; }: 划分 RQT

SIZE_NxN = 3; SIZE_2Nx2N = 0;

compressGOP->compressSlice->compressCtu->xCompressCU(递归)->xCheckRDCostIntra(初始化)->estIntraPredLumaQT(fast mode decision)->xRecurIntraCodingLumaQT(递归选择 RQT)

chroma 预测模式: planar(1), vertical(26), horizontal(10), DC(0), derived mode(36)

pcCU->getPic()->getPicYuvRec()->getAddr()⇒ 重构像素信息

mode selection: 先 check 再 recur

为什么 rec 与 str 匹配?

TComTrQuant:: transformNxN()->xQuant()->xRateDistOptQuant()=>estBits

TEncSearch:: xIntraCodingTUBlock()->transformNxN()

TEncEntropyIf 作用-> 虚类，代指实际编码的熵编码器：cavlc 或 cabac

m_entropyBits ContextModel3DBuffer