

# H.264至HEVC畫面間預測之轉換編碼研究

ChiTeng Tseng  
2012.07.20

# 大綱

- 所需檔案
- 修改方式
  - Fast Motion Estimation
  - Fast Mode Decision
- 統計分析



# 所需檔案

- Fast ME用
  - fast\_CT\_mv.cpp & .h
- Fast MD用
  - fast\_CT\_mode2.cpp & .h

 fast_CT_mode2.cpp	2012/6/12 上午 11...	C++ Source	13 KB
 fast_CT_mode2.h	2012/6/11 下午 10...	C/C++ Header	4 KB
 fast_CT_mv.cpp	2012/5/23 下午 10...	C++ Source	23 KB
 fast_CT_mv.h	2012/7/2 下午 04:43	C/C++ Header	3 KB

# 修改方式(FME)

- 放到HM程式碼資料夾中
- Ex. \HM-6.1rc1\source\App\TAppEncoder\

 encmain.cpp	2012/7/12 下午 05...	C++ Source	4 KB
 fast_CT_mode2.cpp	2012/6/12 上午 11...	C++ Source	13 KB
 fast_CT_mode2.h	2012/6/11 下午 10...	C/C++ Header	4 KB
 fast_CT_mv.cpp	2012/5/23 下午 10...	C++ Source	23 KB
 fast_CT_mv.h	2012/7/2 下午 04:43	C/C++ Header	3 KB
 TAppEncCfg.cpp	2012/4/20 下午 03...	C++ Source	53 KB
 TAppEncCfg.h	2012/4/20 下午 03...	C/C++ Header	16 KB
 TAppEncTop.cpp	2012/4/20 下午 03...	C++ Source	19 KB
 TAppEncTop.h	2012/4/30 下午 02...	C/C++ Header	5 KB

# 修改方式(FME)

- 將論文程式include到下列.cpp內
  - TAppEncoder/encmain.cpp (main所在處)
  - TLibEncoder/TEncSearch.cpp (ME相關函式)
  - TLibEncoder/TEncTop.cpp (encoder class)

```
#include "../App/TAppEncoder/fast_CT_mv.h"
```

由於前面將論文程式放在 TAppEncoder\  
如果不是同資料夾的程式碼要include要記得加上其相對位置

# encmain.cpp

- 在#include之後加入一個全域變數Mvmap\_CT MVmap

```
#if fast_CT_mv
#include "fast_CT_mv.h"
Mvmap_CT MVmap;
#endif
```

- 此變數將儲存reference MV資訊
- 在main()中呼叫Mvmap並且執行其功能

MVmap.Analysis\_data(); -讀入H.264解碼資訊(dec\_264\_mv.txt)

MVmap.Cell\_analysis\_data\_64(cTAppEncTop.get\_width(),cTAppEncTop.get\_height());

-將H.264解碼資訊依照address分區成CU的尺寸64x64

MVmap.Enhance\_cell(cTAppEncTop.get\_width(),cTAppEncTop.get\_height(),1,1);

-合成大區塊的reference MV,後面兩個參數為:改變64x64與32x32合成MV的方式

0:不合成, 1:中位數, 2:平均數 (平均數程式沒寫完整)

cTAppEncTop.get\_width(),cTAppEncTop.get\_height() 需自行編寫，取  
目前編碼畫面的高寬值

# TEncTop.cpp

- 在#include之後加入一個全域變數

`extern MVmap_CT MVmap;`

- `Void TEncTop::encode()`

- 在第一行加入

- `MVmap.Count_frame();`

為了計算目前為第幾張frame

# TEncSearch.cpp

- 在#include之後加入一個全域變數  
`extern MVmap_CT MVmap;`
- 在Void TEncSearch::xMotionEstimation ()做任何函式之前加入

```
MV_CT MV_264;  
static int temp_X,temp_Y,temp_SR;  
int X=pcCU->getCUPelX()+(iRoiWidth/2);  
int Y=pcCU->getCUPelY()+(iRoiHeight/2);  
if(X==temp_X&&Y==temp_Y){  
    iSrchrng=temp_SR;  
}else{  
    MV_264=MVmap.Get_nearestMV_64(MVmap.Get_currframe(),X,Y);  
    iSrchrng=MV_264.Get_MVdiff(cMvPred.getHor(),cMvPred.getAbsVer());  
    temp_X=X;  
    temp_Y=Y;  
    temp_SR=iSrchrng;  
}
```



# TEncSearch.cpp 續

- 接續前面

```
iSrchRng>>=2; (從Q pixel降到int pixel)
```

```
if(iSrchRng>=64)iSrchRng=64;(最小值的限制)
```

```
if(iSrchRng<=2)iSrchRng=2;
```

# 修改方式(FMD)

- 將論文程式include到下列.cpp內
  - TAppEncoder/encmain.cpp (main所在處)
  - TLibEncoder/TEncCU.cpp (CU相關函式)

```
#include "../App/TAppEncoder/fast_CT_mode2.h"
```

由於前面將論文程式放在 TAppEncoder\  
如果不是同資料夾的程式碼要include要記得加上其相對位置

# encmain.cpp

- 在#include之後加入一個全域變數Transcoder\_CT\_mode2  
Tcoder\_CTmode2

```
#if fast_CT_mode
#include "fast_CT_mode2.h"
Transcoder_CT_mode2 Tcoder_CTmode2;
#endif
```

- 此變數將儲存mode資訊
- 在main()中呼叫Tcoder\_CTmode2並且執行其功能

Tcoder\_CTmode2.init(cTAppEncTop.get\_width(),cTAppEncTop.get\_height(),cTAppEncTop.get\_frameToEnco(),cTAppEncTop.get\_QP(),0);

-初始化，輸入編碼畫面資訊，最後一個參數已經沒有用，預設為零

Tcoder\_CTmode2.create();

-讀入H.264解碼資訊(dec\_264\_mode2.txt)

Tcoder\_CTmode2.destroy(); -放在最後，釋出記憶體

cTAppEncTop.get\_width(),cTAppEncTop.get\_height()  
get\_frameToEnco(),cTAppEncTop.get\_QP()需自行編寫，取目前編碼  
畫面的高寬值，張數，QP

# TEncCU.cpp

- 在#include之後加入一個全域變數延伸

`extern Transcoder_CT_mode2 Tcoder_CTmode2;`

- 在Void TEncCu::compressCU ()做xCompressCU()之前加入

`if(m_ppcBestCU[0]->getSlice()->getSliceType() != I_SLICE) -針對P frame`  
`Tcoder_CTmode2.set_Tcoder_CU_AMBA(rpcCU->getPic()->getPOC(), rpcCU->getAddr());`  
`-設定Dct coeff參數`

- Void TEncCu::xCompressCU

- 這邊就是做CU中PU切割的遞回函式
- uiDepth 代表目前切割的深度
- 在這邊利用設置flag，當條件滿足利用flag開關快速決策 (flag的說明在後面)

## 跟論文的演算法流程一樣寫法

```
switch(uiDepth)
{
case 0:
    if(Tcoder_CTmode2.dec264_subMB>8)                bfastCT_skip=true;
    if(Tcoder_CTmode2.dec264_skip<2)                bfastCT_skip=true;
    break;

case 1:
    if(Tcoder_CTmode2.dec264_subMB>8)                bfastCT_skip=true;
    if(Tcoder_CTmode2.dec264_skip<2)                bfastCT_skip=true;

    if(Tcoder_CTmode2.dec264_AMBA_d0==0&&Tcoder_CTmode2.dec264_totalWeight<=25)
        bfastCT=false;
    break;

case 2:
    if(Tcoder_CTmode2.dec264_part[rpcBestCU->getZorderIdxInCU()/16]>2)
        bfastCT=true;
    else if(Tcoder_CTmode2.dec264_AMBA_d2[rpcBestCU->getZorderIdxInCU()/16]>=1.5)
        bfastCT=true;

    if(Tcoder_CTmode2.dec264_isskip[rpcBestCU->getZorderIdxInCU()/16]==1)
        bfastCT_d2inskip=true;
    break;

default:
    break;
}
```

# TEncCU.cpp Flag設定

- bfastCT (不再繼續往下切)
  - 作用同等於原程式碼中的bTrySplit
  - 加在程式何處請參考程式碼
  - ★ if(bBoundary)bfastCT=true;  
bBoundary指出當前CU是否為frame邊界
- bfastCT\_skip (簡單查找)
  - 作用同等於原程式碼中的bEarlySkip
- bfastCT\_d2inskip (不做intra)

# 統計分析

# 統計分析

- 論文中的實驗如果是**ME**的分析就修改
  - Void TEncSearch::xMotionEstimation ()
  - 此為**ME**過程的遞迴函式
- 如果是**MD**的分析
  - Void TEncCu::compressCU ()
  - 在xCompressCU()執行完之後分析
  - 此時**PU**最佳分割已經決定完成
  - 參考程式碼在  
\\論文中的實驗執行檔&程式 (HM 6.1)\\分析\\code



# 統計分析

- 算時間

```
#include <time.h>
```

```
{
```

```
double dResult;
```

```
long lBefore = clock();
```

```
dResult = (double)(clock()-lBefore) / CLOCKS_PER_SEC;
```

```
}
```