**UAVS3E 软件基本架构 (v1.0)**

目录

[1 编码库API及调用流程 2](#_Toc43928852)

[1.1 数据结构说明 2](#_Toc43928853)

[1.2 函数接口说明 2](#_Toc43928854)

[1.3 调用过程 3](#_Toc43928855)

[2 数据结构分层描述 4](#_Toc43928856)

[2.1 顶层结构 4](#_Toc43928857)

[2.2 图像层结构 4](#_Toc43928858)

[2.3 LCU(ROW)层结构 5](#_Toc43928859)

[2.4 其他功能性结构体 7](#_Toc43928860)

[3 编码库调用框架 7](#_Toc43928861)

[3.1 主线程调用流程 7](#_Toc43928862)

[3.2 帧级编码线程调用流程 8](#_Toc43928863)

[3.3 LCU行编码流程 8](#_Toc43928864)

[3.4 LCU编码树递归 9](#_Toc43928865)

[3.5 CU帧间编码 9](#_Toc43928866)

[3.6 CU帧内编码 10](#_Toc43928867)

# 编码库API及调用流程

编码库接口头文件包含“com\_api.h”和“uavs3e.h”，编码库使用过程可参考“utest.c”。

## 数据结构说明

com\_img\_t：

存放待编码图像的结构体，结构体对象由编码库创建并提供调用者使用，用于输入。

enc\_stat\_t：

编码状态及编码信息输入输出结构体，由调用者创建。

enc\_cfg\_t：

编码配置结构体，由调用者创建。

## 函数接口说明

void uavs3e\_load\_default\_cfg(enc\_cfg\_t \*cfg);

功能：加载默认的编码配置。

参数： cfg – 指向编码配置结构体对象的指针，概述通过该指针给结构体对象赋值。

void\* uavs3e\_create(enc\_cfg\_t \*param, int \*err);

功能：创建编码库。

参数：param - 指向编码配置结构体对象的指针，以该配置初始化编码库。

err - 错误编号，若创建失败，返回相应错误信息。暂时无用，可以为NULL。

返回值： 编码库句柄。

void uavs3e\_free(void \*id);

功能：释放编码库。

参数：id – 编码库句柄。

int uavs3e\_get\_img(void \*id, com\_img\_t \*\*img);

功能：从编码库获取一个图像结构体的对象，用于存储下一帧待编码图像。

参数：id – 编码库句柄。

img – 指向图像指针的二级指针，编码库通过该指针对其指向的图像指针赋值，提供对象。

返回值： COM\_OK - 成功

COM\_ERR\_UNEXPECTED - 异常

int uavs3e\_enc(void \*id, enc\_stat\_t \*stat, com\_img\_t \*img\_enc);

功能：送入一帧图像进行编码。

参数：id – 编码库句柄。

stat – 指向编码状态结构体对象的指针，用于输入输出待编码图像之外的所有信息。

img\_enc – 指向待编码图像结构体的指针。该指针为NULL，表示该调用用于Flush码流。

返回值： COM\_OK – 编码输入成功，且有码流输出

COM\_OK\_OUT\_NOT\_AVAILABLE – 编码输入成功，码流输出还不可用

COM\_OK\_NO\_MORE\_FRM – 没有更多码流输出

其他异常值可以参考“com\_api.h”中的定义

void uavs3e\_find\_psnr(com\_img\_t \*org, com\_img\_t \*rec, double psnr[3], int bit\_depth);

功能：计算重建图像的PSNR。

参数：org – 指向原始图像的指针。

rec – 指向重建图像的指针。

psnr – 用于存放psnr计算结果的数组

bit\_depth – 图像的编码位宽

void uavs3e\_find\_ssim(com\_img\_t \*org, com\_img\_t \*rec, double ssim[3], int bit\_depth);

功能：计算重建图像的SSIM。

参数：org – 指向原始图像的指针。

rec – 指向重建图像的指针。

ssim – 用于存放ssim计算结果的数组

bit\_depth – 图像的编码位宽

## 调用过程

调用过程伪码描述如下：

enc\_cfg\_t cfg;

uavs3e\_load\_default\_cfg(&cfg);

void \*h = uavs3e\_create(&cfg, NULL);

while(循环处理每一帧图像){

com\_img\_t \*img\_enc;

enc\_stat\_t stat;

uavs3e\_get\_img(h, &img\_enc);

往 img\_enc 中填入待编码的图像数据;

ret = uavs3e\_enc(h, &stat, img\_enc);

if (ret == COM\_OK) {

输出码流;

}

}

Do { // flush未输出的码流

ret = uavs3e\_enc(h, &stat, NULL);

if (ret == COM\_OK) {

输出码流;

}

} while (ret == COM\_OK);

uavs3e\_free(h);

# 数据结构分层描述

## 顶层结构

|  |  |
| --- | --- |
| **enc\_ctrl\_t：主线程使用的结构体，主要负责输入输出、预处理、图像重排、参考帧管理、帧级编码线程池管理、RC等功能。** | |
| 主要成员变量 | 功能描述 |
| cfg | 全局的编码配置信息 |
| img\_lastIP | 输入链表、重排序链表，以及lookahead链表的管理 |
| img\_rlist |
| img\_rsize |
| node\_list |
| node\_size |
| lastI\_ptr |
| ilist\_size |
| ilist\_imgs |
| prev\_dtr | 用于ptr/dtr/pts/dts 的管理和计算 |
| dtr |
| ptr |
| prev\_pts |
| prev\_ptr |
| rpm | 参考配置集 |
| info | 存储图像级编码线程所需的信息，启动图像级编码线程时通过**pic\_thd\_param\_t**对象拷贝到工作线程的**enc\_pic\_t**对象中，用于图像编码 |
| pichdr |
| refp |
| top\_pic |
| frm\_threads\_pool | 图像级并行使用的线程池管理 |
| pic\_thd\_params |
| pic\_thd\_head |
| pic\_thd\_tail |
| pic\_thd\_active |
| rc | 码率控制模块 |
| pinter | 预处理阶段ME所需的ME对象 |
| tab\_mvbits | ME过程运动矢量比特数查表所需的表格 |
| tab\_mvbits\_offset |

|  |
| --- |
| **pic\_thd\_param\_t：用于向帧级线程传递当前编码帧的各种信息，例如图像头、参考关系，以及全局的编码配置信息等。** |

## 图像层结构

|  |  |
| --- | --- |
| **enc\_pic\_t：帧级编码线程中使用的结构体，线程池创建时进行创建。主要管理图像级的编码数据，包括各种编码信息的MAP、码流、帧级临时数据等，以及WPP线程池。** | |
| 主要成员变量 | 功能描述 |
| wpp\_threads\_pool | WPP并行的线程池 |
| map | 编码信息帧级map |
| info | 编码配置全局信息 |
| ip\_tmp\_buf | 帧级临时Buffer |
| bs\_buf\_demulate |
| linebuf\_intra |
| linebuf\_sao |
| map\_cu\_data | 存储编码决策结果，用于最后的熵编码 |
| pic\_alf\_on | 图像级的滤波信息及数据 |
| Enc\_ALF |
| pic\_alf\_Rec |
| alf\_var\_map |
| sao\_blk\_params |
| array\_row | LCU行的初始化信息，用于WPP并行 |
| main\_core | WPP关闭时使用的LCU层编码结构体对象，同时负责最后码流的熵编码 |
| pinter | 帧级预处理过程ME使用的结构体对象 |

|  |  |
| --- | --- |
| **enc\_lcu\_row\_t：用于启动WPP线程时传递所需信息。并存储WPP同步及熵编码所需的协调信息。** | |
| 主要成员变量 | 功能描述 |
| lcu\_y | 标识当前的LCU行的编号 |
| lbac\_row | 用于熵编码及协调LCU行之间的熵编码 |
| lbac\_row\_next |
| sem | WPP同步所需的信号量 |
| sem\_up |
| sem\_curr |
| pic\_info | 指向**enc\_pic\_param\_t**结构体对象的指针，用于LCU行编码过程。该对象被该帧所有WPP线程共享 |
| total\_qp | 统计编码QP |

|  |
| --- |
| **enc\_pic\_param\_t：存储指向各种帧级信息指针的结构体。在帧级编码线程中被初始化，后传入LCU编码线程，该结构体对象被当前帧所有LCU编码线程共享。** |

## LCU(ROW)层结构

|  |  |
| --- | --- |
| **core\_t：LCU编码过程使用的数据结构，贯穿整个编码底层。存储上层传递的序列级、图像级信息，存储CU决策的各级信息等。** | |
| 主要成员变量 | 功能描述 |
| Param/info/pichdr/  Map/map\_cu\_data/  pic\_org/pic\_rec  sao\_blk\_params  linebuf\_sao/  linebuf\_intra/  refp/ptr/  num\_refp/slice\_type  wq | 存储上层传递的序列级/图像级信息和map |
| motion\_cands | HMVP工具所需的历史mvp信息 |
| cnt\_hmvp\_cands |
| sao\_src\_buf | SAO滤波所需的临时buffer |
| lcu\_x/lcu\_y/  lcu\_pix\_x/lcu\_pix\_y | LCU坐标信息 |
| lcu\_qp\_y/u/v | LCU的QP和lambda信息 |
| Lambda/sqrt\_lambda  sqrt\_lambda |
| cu\_data\_best | CU Depth递归决策需要的数据栈 |
| cu\_data\_temp |
| history\_data | Save/load快速决策机制需要的历史信息 |
| tree\_status | 亮色度分离及const模式标志 |
| cons\_pred\_mode |
| cu\_width  —  cu\_scup\_in\_lcu | 该组变量存储CU的各种坐标宽高信息 |
| skip\_flag  —  ctmp | CU Depth决策和模式决策需要的临时变量 |
| lbac\_rdo  —  lbac\_tree\_c | RDO过程记录的熵编码状态信息 |
| rdoq\_bin\_est\_ctp  —  rdoq\_bin\_est\_lst | RDOQ过程快速bit数计算需要的表格，在每次RDOQ前进行设置 |
| mod\_info\_best | 模式决策存储的CU编码信息 |
| mod\_info\_curr |
| pinter | ME需要的数据结构对象 |
| intra\_pred\_all | 存放intra预测各个模式预测块的临时buffer |
| cost\_best | CU Depth决策过程记录的当前最优cost |
| inter\_satd  —  skip\_emvr\_mode | 各种快速算法需要的变量 |
|  |  |

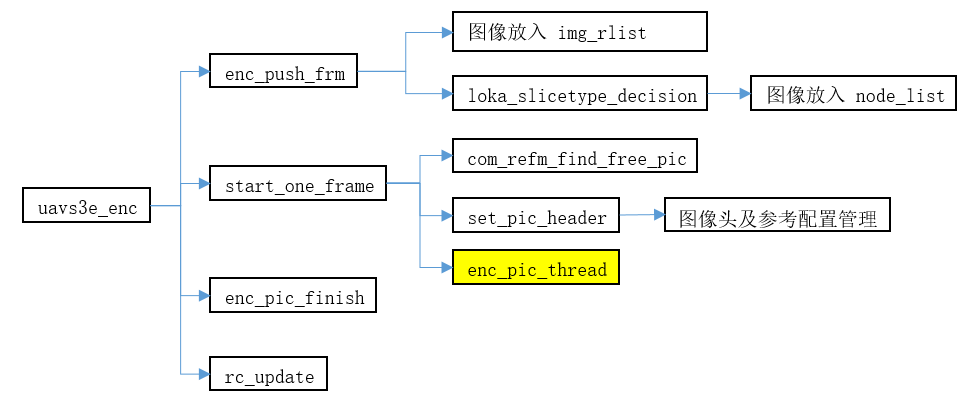
## 其他功能性结构体

|  |  |
| --- | --- |
| 结构体 | 功能描述 |
| app\_cfg\_t | 编码配置，在编码库外由调用者设置，可使用uavs3e\_load\_default\_cfg()载入默认初始化配置 |
| com\_info\_t | 编码全局信息，跟据编码配置生成的一些常用的全局信息，例如图像宽高、以LCU计算的宽高、LCU/SCU总数等常规信息，以及跟据配置中speed\_level设置的各种快速算法开关 |
| com\_seqh\_t | 序列头信息 |
| com\_pic\_header\_t | 图像头信息 |
| com\_patch\_header\_t | Slice头信息 |
| com\_pic\_t | 图像帧数据结构，包含**com\_img\_t**结构体对象指针、**com\_subpel\_t**分像素图像、图像ptr/dtr、参考关系、用于co-local预测的mv/ref信息，以及帧间同步需要的数据 |
| com\_subpel\_t | 存储分像素插值图像，包含16个**com\_img\_t**结构体对象指针 |
| com\_ref\_pic\_t | 参考帧结构体，包含指向**com\_pic\_t**结构体对象及其内部一些变量的若干指针。 |
| com\_map\_t | 图像级全局map，包含各种编码信息的图像级存储buffer |
| enc\_cu\_t | LCU编码信息存储单位，存储LCU内所有CU的编码信息。**enc\_pic\_t**结构体中map\_cu\_data指向的空间，存储了每一个LCU的**enc\_cu\_t**结构体对象 |
| com\_mode\_t | CU模式决策过程中用于记录CU编码信息的结构体 |
| enc\_history\_t | Save/load机制快速算法需要的结构体 |
| threadpool\_t | 线程池 |
| enc\_rc\_t | RC模块 |
| inter\_search\_t | ME模块 |
| lbac\_t | 熵编码引擎及上下文 |
| bs\_t | 用于码流管理的结构体 |
| enc\_alf\_var\_t | ALF模块 |
| com\_sao\_param\_t | SAO模块 |

# 编码库调用框架

## 主线程调用流程

主线程在enc\_ctrl\_t中维护两个链表，第一链表为输入lookahead链表(img\_rlist)，第二链表为重排序后的待编码链表(node\_list)。调用流程框图如下：

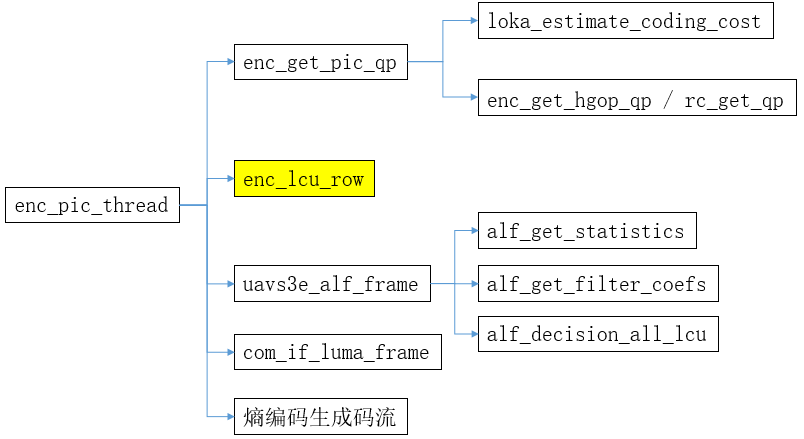


其中，loka\_slicetype\_decision()函数执行的条件是img\_rlist链表中图像数超过预设的lookahead帧数；start\_one\_frame()函数执行的条件是node\_list链表不为空。

图中黄色模块为通过线程池启动的帧级编码线程函数。

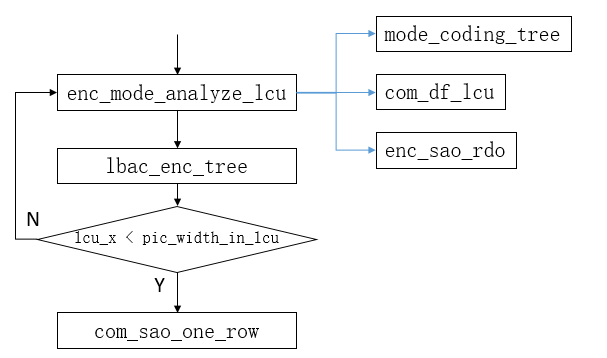
## 帧级编码线程调用流程

帧级编码线程首先获取图像QP，然后启动WPP线程对LCU行进行编码决策。所有LCU行决策完成后，进行ALF滤波和插值。最后，跟据决策结果对图像进行熵编码，生成码流。



## LCU行编码流程

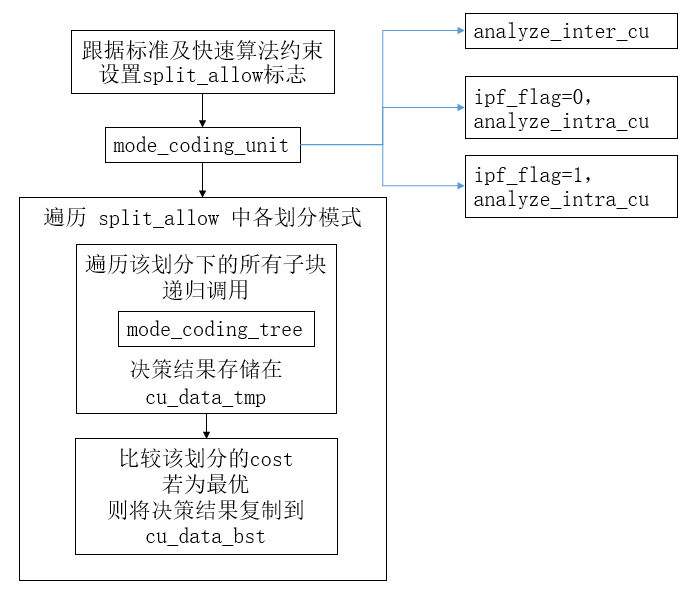
enc\_lcu\_row 函数对LCU行中的LCU逐个进行编码决策，并进行熵编码，更新算术编码的上下文模型。决策过程，包含了调用mode\_coding\_tree()进行编码树的决策，以及执行Deblocking和SAO决策。在决策完当前行所有LCU之后，再统一跟据SAO决策结果对LCU行进行SAO滤波。



## LCU编码树递归

mode\_coding\_tree()函数为编码树递归函数，其内部基本过程为：1. 确定可用的划分和不划分方式；2. 不划分如果可用，调用mode\_coding\_unit()进行CU的模式决策；3. 遍历其他可用的划分方式，对每一个划分方式，遍历每一个子块，递归调用mode\_coding\_tree()。

mode\_coding\_unit()函数为编码一个CU，内部尝试决策inter以及ipf标志为0和1时的intra模式。如果当前模式为最优，则将当前CU对应com\_mode\_t对象中存储的编码信息拷贝到enc\_cu\_t对象中。

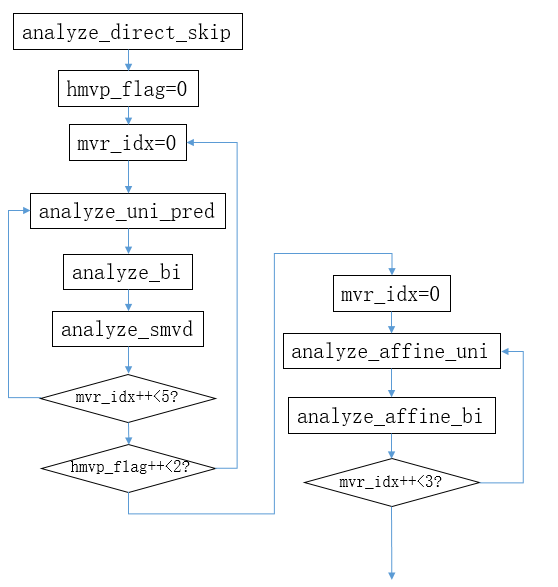


## CU帧间编码

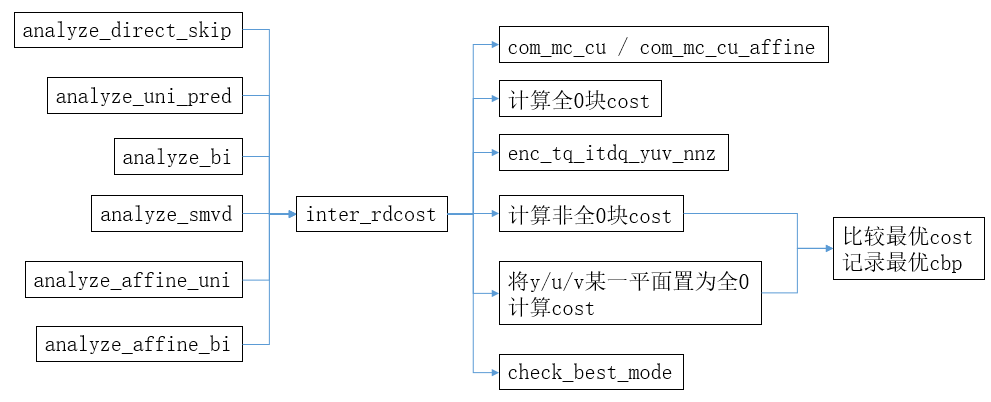
帧间编码函数ananlyze\_inter\_cu() 基本过程如下图。首先尝试Direct和Skip模式，然后跟据hmvp\_flag和mvr\_idx不同的值，多次尝试单向、双向及SMVD模式编码。最后跟据mvr\_idx不同的值，多次尝试affine的单向和双向编码。

其中，hmvp\_flag为0时，不同的mvr\_idx对应AMVR工具不同的运动矢量精度；hmvp\_flag为1时对应EMVR编码工具，此时mvr\_idx同时指定了对应的运动矢量精度及hmvp历史运动矢量。

此外，analyze\_direct\_skip()函数中，会将所有可能的运动矢量导出方式均尝试一遍，通过satd选出N个最优的候选，最后对挑选出的候选进行RDO决策。



对于所有的帧间模式，最后均调用inter\_rdcost()进行RDO，计算率失真代价。Inter\_rdcost()函数的基本流程如下图所示。首先进行MC得到预测块，然后计算得到全0块的RDCost。再进行变换量化，反量化反变换得到重建块，若y/u/v三个非全为0，则计算RDCost，并同全0块RDCost比较，记录最优RDCost和CBP。进一步将y/u/v逐一置为全0，计算RDCost，并比较和记录最优RDCost和CBP。最后调用check\_best\_mode()，将该模式最优RDCost同先前尝试的模式进行比较，记录最优RDCost和编码信息。



## CU帧内编码

帧内编码函数analyze\_intra\_cu()基本过程如下图。该函数对intra的决策分为亮度和色度两步。第一步决策亮度的模式，亮度可以采用不同的DT划分，所以首先检测可用的DT划分，然后遍历各个划分。对每个DT划分，遍历每个子块(PU)。对每个PU，需要决策其最优的预测模式。PU的RDCost通过intra\_pu\_rdcost()得到。PU预测模式决策过程需记录PU的最优RDCost和模式。所有子块决策完成后可得到DT划分的RDCost，再比较并记录最优的DT划分的RDCost和模式信息。

色度在亮度完成之后独立进行，色度没有DT划分，直接遍历可用的色度模式进行决策。

