运动侦测代码说明

1. 变量说明：

|  |  |  |
| --- | --- | --- |
| 变量名 | 类型 | 说明 |
| ctx | 输入 结构体IMG\_INFO | 运动侦测用的参数和malloc的数据指针 |
| ctx.img\_down\_sam | 输入 unsigned char\* | Yuv下缩放3倍左右的亮度数据指针，必须保证运动侦测执行过程中buf内数据不被覆盖。 |
| is\_single\_ref | 输入 int | 运动侦测工作模式，单参考帧或者多参考帧。多参考帧要比单参考帧多开辟3帧下缩放的亮度数据。 |
| width | 输入 int | 原始图像宽 |
| height | 输入 int | 原始图像高 |
| width\_ds | 输入 int | 下采样图像宽 |
| height\_ds | 输入 int | 下采样图像高 |
| roi\_in | 输入输出 结构体ROI\_INFO | 感兴趣区域，输入感兴趣区域有效性及其左上角和右下角原始坐标，输出该区域内是否运动的标志。 |
| info\_list | 输出 结构体INFO\_LIST | 运动侦测输出，每个运动区域的左上角和右下角的原始坐标。以及该坐标有效性。注意是原始图像位置的坐标，而不是下采样图像的坐标。 |

1. 函数说明：

运动侦测一共包括3个函数

Md\_instance move\_detection\_init(int width, int height, int width\_ds, int height\_ds, int is\_single\_ref)

初始化函数，包括malloc运动侦测所需要的数据空间，变量初始化。

void move\_detection\_deinit(Md\_instance ins)

反初始化函数，释放malloc运动侦测所需要的数据空间。

void move\_detection(Md\_instance ins, void \*sampledownptr, roi\_in \*in, INFO\_LIST \*info\_list)

运动侦测函数主体，一帧需要调用一次。在调用该函数之前，必须保证下采样数据已经存储在ctx.img\_down\_sam中，还需要把输出结构体info\_list清0。

1. 参考代码说明：

可以参照test.cpp文件中的函数 int hw\_simu\_top() 来调用运动侦测模块。该函数主要流程如下说明：

1. 初始化变量。
2. 调用函数move\_detection\_init，初始化运动侦测模块。
3. 读取一帧的yuv数据，调用rga\_ds函数进行亮度数据的下采样。
4. 初始化info\_list，清0。
5. 调用move\_detection函数，进行运动侦测，输出存放在变量info\_list中。
6. 重复步骤3~5，直到yuv结束。
7. 反初始化运动侦测move\_detection\_deinit。
8. 结束。
9. 其他说明：
10. 运动侦测工作模式说明：

变量is\_single\_ref用于配置运动侦测工作模式，采用单参考帧或者多参考帧。当采用单参考帧时，需要的数据较少，计算较快。采用多参考帧计算量比单参考帧大致增加40%。如果系统一开始配置的是多参考帧，允许在某一帧开始时候切换为单参考帧模式，但是以后不能再切换回来，因为参考帧列表无法重新建立。如果系统配置的是单参考帧模式或者已经切换为单参考帧模式，想要切换为多参考帧模式，需要重新初始化。

1. 感兴趣区域说明：

运动侦测会根据外部配置的感兴趣区域进行区域内运动的检测和报警。这部分功能还未设计完善。接口暂定如下：

typedef struct {

unsigned short flag; // 1 有效， 0无效

unsigned short is\_move; // 运动标志，每帧roi检测前必须初始化为0

unsigned short up\_left[2]; //下采样前左上角坐标，0是y坐标，1是x坐标

unsigned short down\_right[2]; //下采样前右下角坐标，0是y坐标，1是x坐标

} ROI\_INFO;

暂定只支持一个区域。已经实现。

1. 工作不正常时候的容错处理：

如果时间不够，可以切换为单帧模式，如果还不够，可以在外部控制为2帧跑1帧或者3帧跑1帧甚至更多。但是输出给mpp的信号必须每帧都输出，可以2帧或者3帧输出同样的结果。

1. 宏的说明：

#define HW 0 软仿时候用0，上板必须改为1

#define TIME 0 上板跑的时候打印各个子函数时间

#define debug 1 只影响软仿时候的文件打印

#define LIGHT 1 暂时不影响整个流程

#define compare 0 只影响软仿时候的文件打印

#define USE\_NEON 1 是否用neon代码，软仿或者上板都可以用或者不用neon

1. 遗留问题：
2. 支持小数倍的下采样，不过还未得到有效验证。 小数倍下采样的数据无法提供。
3. 下采样后的图像边界，不是4的倍数的时候still\_value\_set无法计算剩余点数，所以这部分点数不参与计算（没有做扩边的处理）。不是16的倍数的时候，add等函数没有对不是16倍数的块进行处理。因此实际上，下采样后的图像，不是16的整数倍的块都是没有处理的（最坏情况是15个点，折合原始图像的像素大概是45个点）。出现在原始图像最右边和最下边的区域有可能是不检测的。解决方法，可以考虑下采样的时候就按照下采样后是16的倍数进行。例如2560x1440，可以下缩放为848x480。