Hi3510 移动侦测实现

Application Notes

文档版本 02

发布日期 2006-12-20

BOM编码 N/A

深圳市海思半导体有限公司

深圳市海思半导体有限公司为客户提供全方位的技术支持，用户可与就近的海思办事处联系，也可直接与公司总部联系。

深圳市海思半导体有限公司

地址：　深圳市龙岗区坂田华为基地华为电气生产中心　邮编：518129

网址：　http://www.hisilicon.com

客户服务电话：　0755-28788858

客户服务传真：　0755-28788838

客户服务邮箱：　support@hisilicon.com

商标声明

 、Hisilicon、海思，均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

由于产品版本升级或其它原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

前 言

概述

本节介绍本文档的内容、对应的产品版本、适用的读者对象、行文表达约定、历史修订记录等。

产品版本

与本文档相对应的产品版本如下所示。

产品名称	产品版本
Hi3510	V100R001B01

读者对象

本文档描述了基于 Hi3510 的移动侦测方案以及 API 接口函数说明。使用本文档的程序员应该：

- 熟练掌握 Linux 文件读写操作和内存映射编程方法
- 熟练使用 C 语言
- 具备在 Linux 环境编译，调试程序的知识

内容简介

本文档介绍了基于 Hi3510 的移动侦测方案在区域锁定模式下的应用，针对实际应用场景，详细阐述了相应的方案细节。

章节	内容
1 概述	介绍了基于 Hi3510 的移动侦测的适用领域及其文档的内容组织。
2 移动侦测解决方案	重点介绍区域锁定模式下的单个和多个侦测区域的解决方案。
3 如何实现移动侦测	以区域锁定模式下的普通场景应用为例，通过实现的部分代码和界面实例说明如何实现移动侦测。

章节	内容
附录 A　数据类型定义	列出文档中部分数据类型定义。

约定

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	以本标志开始的文本表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	以本标志开始的文本表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	以本标志开始的文本表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
窍门	以本标志开始的文本能帮助您解决某个问题或节省您的时间。
 说明	以本标志开始的文本是正文的附加信息，是对正文的强调和补充。

通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。
楷体	警告、提示等内容一律用楷体，并且在内容前后增加线条与正文隔离。
“Terminal Display”格式	“Terminal Display”格式表示屏幕输出信息。此外，屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。

修改记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修改日期	版本	修改说明
2006-12-20	02	删除了“区域锁定模式移动侦测的实现思路”、“触发门限”等内容。
2006-8-31	01	第一次版本发布。

1 概 述

关于本章

本章描述内容如下表所示。

标题	内容
1.1 应用领域	介绍移动侦测的应用领域。
1.2 主要内容	说明本文档涉及的主要内容。

1.1 应用领域

基于 Hi3510 芯片开发的 DVS 产品的移动侦测系统，可广泛应用于金融、电信、交通、电力、水利、教育、旅游、文物保护等领域，以及党政机关、政法、军警、监狱等部门，还适用于智能大厦、智能小区、宾馆饭店、超市、工矿企业等环境的安全技术防范系统及管理。

此文档面向基于 Hi3510 芯片进行 DVS 产品开发的人员，提供移动侦测的基本解决方案。针对不同的实际应用场景，开发者可以调用为其提供的软件开发包很便利的进行开发，也可以在 Hi3510 移动侦测 API 接口函数的基础上进行开发。

1.2 主要内容

本文档介绍了基于 Hi3510 芯片的移动侦测的实现机制，针对区域锁定模式下的单个和多个侦测区域的应用，给出了几个典型的用户场景，为每个用户场景设计了详细的解决方案，供开发者参考。Hi3510 移动侦测模块的 API 接口函数提供了移动侦测功能的启用和禁止、移动侦测数据获取等功能，本文档对 API 接口函数进行了详细介绍。最后以区域锁定模式下的普通场景应用为例，通过部分代码和界面实例说明如何实现移动侦测。

2 移动侦测解决方案

关于本章

本章描述内容如下表所示。

标题	内容
2.1 底层数据的处理	介绍 SAD 的获取及处理方法。
2.2 区域锁定模式	描述区域锁定侦测模式的移动侦测解决方案。
2.3 API 接口函数	介绍移动侦测模块提供的部分 API 接口函数。

2.1 底层数据的处理

调用底层提供的 API 接口函数得到所有宏块的 SAD 值（Sum of Absolute Differences），再通过应用层进行处理。如果移动侦测的时间间隔为 f 帧（缺省为 5 帧），即每 5 帧进行一次移动侦测处理，在提供的样例中，每 5 帧对 SAD 进行一次累加，把累加得到的和 SAD_Sum 进行后续的移动侦测处理。

2.2 区域锁定模式

2.2.1 实现思路

区域锁定模式即在固定区域内侦测的方式。在该模式下用户对摄像镜头的视频窗口设定侦测区域，当一个或多个选定的侦测区域内的图像发生变化，导致整个摄像镜头的视频窗口的侦测参数值超出预设值，则触发报警。

产生报警的摄像镜头的视频界面自动放大，视频界面上红色报警区域不停闪烁，同时发出报警声音，启动录像和抓拍，发出报警短信或邮件。

该模式下的侦测方案涉及的参数包括侦测区域、侦测灵敏度、触发门限值和场景。

移动侦测区域由用户设定，每个区域的侦测灵敏度和触发门限也可由用户修改。输入 f 帧的 SAD_Sum，如果在选定的侦测区域内的图像发生变化，一旦区域内的图像侦测参数超出预设值，则触发区域报警。

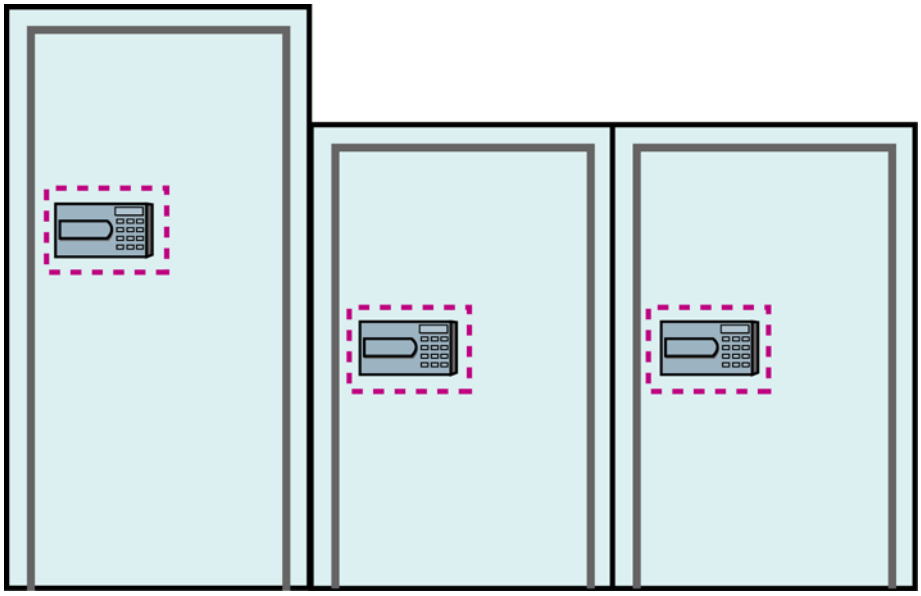
根据场景的算法，区域报警引发此摄像镜头窗口的报警，从而启动联动报警。

以下通过介绍参数设置的方法对侦测方案作详细介绍。

2.2.2 侦测区域的选定

用户可以通过在视频窗口中用鼠标拖动画出矩形框来设置侦测区域，每个窗口最多可以设置 4 个侦测区域。如图 2-1 所示，矩形框的选取最好恰好包围待侦测的区域，以节省系统运算时间同时提高准确率。

图2-1 选定侦测区域



2.2.3 单个侦测区域的解决方案

在选定了侦测区域后，如果只有单个侦测区域，用户只需要设定侦测灵敏度和触发门限值这两个参数。

侦测灵敏度的定义

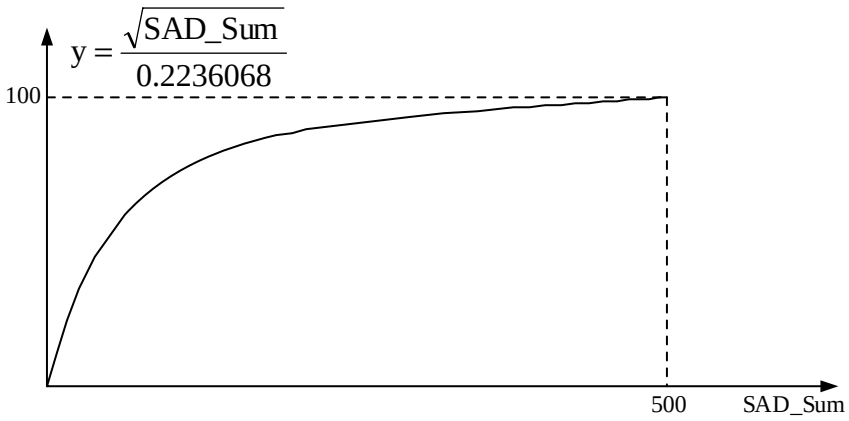
侦测灵敏度是指对每个宏块图像变化的敏感程度，取值范围为 0～100，值越大代表对图像变化的敏感程度越高。将宏块图像变化的度量值通过一定的函数变换，与侦测灵敏度相比较，就可以判断宏块是否报警。当侦测灵敏度取 100 时，该宏块总是报警；取 0 时总不报警。

在 Hi3510 DVS 系统中，NTSC 制每秒有 30 帧图像（PAL 制每秒 25 帧），采取每 5 帧（f=5）做一次移动侦测。5 帧图像的 SAD 的和为 SAD_Sum，SAD_Sum 值的变化范围为 0～500，而系统的侦测灵敏度的范围是 0～100，因此需要对 SAD_Sum 进行标准化处理。

在标准化处理的过程中，经过多次试验发现，如果简单把 SAD_Sum 除以 5，则 SAD_Sum 值的分布集中在 2～5 范围内。再对 SAD_Sum 取 100 的补，则 SAD_Sum 值的分布集中在 95～98。系统的灵敏度值范围为 0～100，因此 SAD_Sum 分布明显不均匀且不合常理。

为了使 SAD_Sum 值均匀地分布在 0～100 之间，可采用图 2-2 所示的转换，把 SAD_Sum 作均匀分布处理，再以 100 减去处理结果，即得到直接与所设灵敏度比较的值。

图2-2 SAD_Sum 的标准化



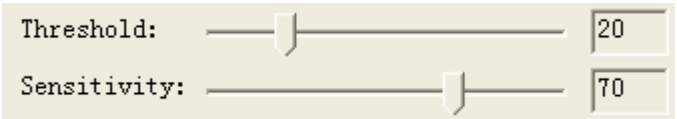
触发门限的定义

触发门限值越大，则触发一个侦测区域报警所需要的报警宏块数就越大，触发门限的范围为 0～100，取 100 时要求该侦测区域内所有宏块都报警该区域才报警，取 0 时总是报警。

参考设置界面

灵敏度和触发门限的参考设置的界面如图 2-3 所示。

图2-3 单个区域移动侦测参数的设置



2.2.4 多个侦测区域的解决方案

当一个视频窗口内划分有多个侦测区域时，首先要为各个侦测区域设定各自的侦测灵敏度和触发门限值，此外还要根据实际需要，选择恰当的侦测场景模式，即为各侦测区域间设置某种关联。

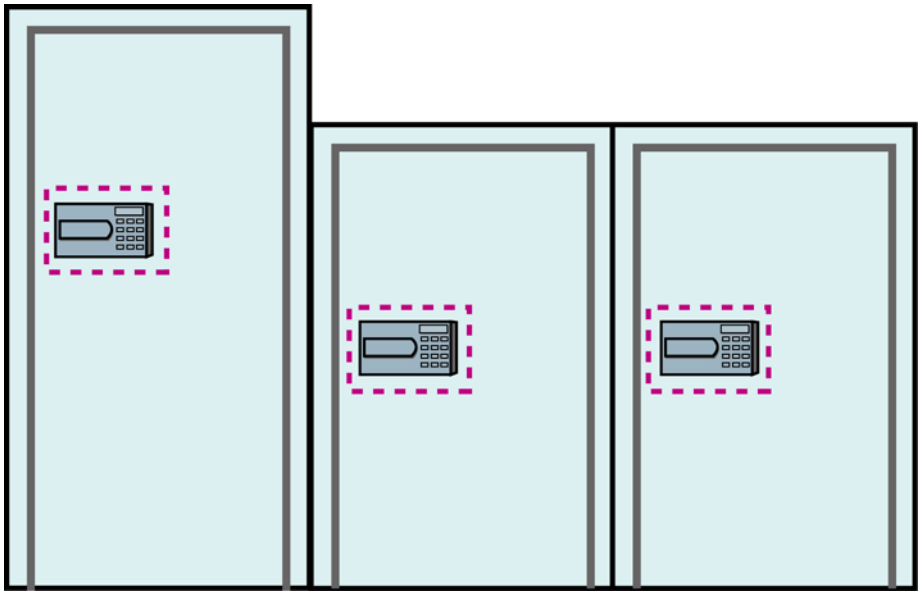
本节给出了几个典型的应用场景，由开发者根据具体情况参考。其他场景的情况，开发者可以灵活地应用 Hi3510 的移动侦测 API 接口函数。

普通场景

首先在视频窗口界面上根据实际需要选定多个侦测区域（最多 4 个），为每个侦测区域设定侦测灵敏度和触发门限值。普通场景下，各个侦测区域之间的相互联系设为“或”关系，即任何侦测区域报警都会使得整个系统产生报警。

如图 2-4 所示从左到右有三个密码锁，中间位置的密码锁需要重点监控。因此分别设定三个侦测区域，再将中间位置的侦察区域设置为高侦测灵敏度、低触发门限，即可完成布防。

图2-4 普通场景

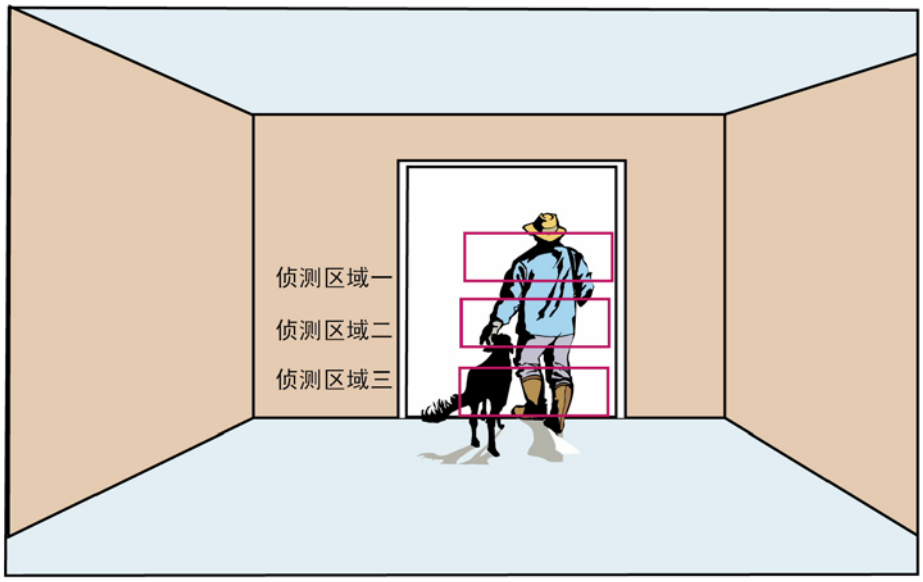


走廊场景

对门窗、走廊实施移动侦测时，建议使用走廊模式。

如图 2-5 所示，某小区保安需要对小区进出口进行移动侦测，又要避免各种小动物、昆虫、鸟类可能产生的误报。可在进出口的监视画面上划分出多个侦测区域，各侦测区域间的相互联系设置“与”关系，即当所有的侦测区域都同时触发报警时，整个系统才会报警。由于小动物、昆虫的体积有限，不可能同时侵入所有的侦测区域，减少了系统误报的可能。当有人经过进出口时，必然会同时进入所有侦测区域，立即触发系统报警。

图2-5 走廊场景



室外场景

由于室外环境存在雨、云、雾等干扰因素，为减少侦测系统误报，需要对这些外界干扰作判断处理。为了提高处理的速度，不必对图像的每一个宏块进行判断，只需在图像上找出几个代表性的宏块进行判断。如果每个宏块的侦测值都超过预设值，就认为此图像的变化是由于外界干扰的原因，无需进行后续处理。

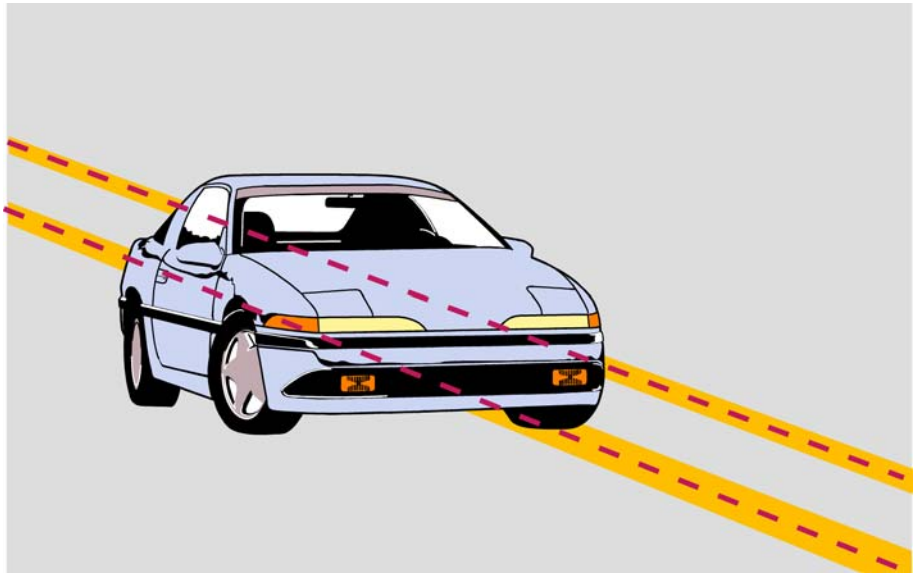
边界场景

在边界场景中，侦测系统对区域内的运动不做处理，当物体的运动侵入区域的边界时，才会触发报警。

边界场景也有着比较广泛的应用，如公路的压黄线（如图 2-6 所示）、斑马线、建筑物大门的入口、监狱的围墙等。

在边界场景中，有的用户可能对物体运动方向也有需求，这时除了需要图像移动的度量值 SAD 以外，还需要运动矢量（Motion Vector）。

图2-6 边界场景



2.3 API 接口函数

Hi3510 移动侦测模块主要提供移动侦测功能的启用和禁止、移动侦测数据获取等功能，该功能模块主要提供以下 API：

- HI_MD_Enable
  使能移动侦测功能接口
- HI_MD_Disable
  禁用移动侦测功能接口
- HI_MD_GetEnable

获取移动侦测功能是否启用接口

- HI_MD_GetStatus
  获取移动侦测功能数据接口

【需求】

- 头文件：hi_common_api.h
- 库文件：libhiapi.a

HI_MD_Enable

【目的】

使能指定通道的移动侦测功能

【语法】

```
HI_S32 HI_MD_Enable (HI_S32 ChnID );
```

【参数】

参数名称	描述	输入/输出
ChnID	表示视频编码通道号	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	描述
HI_SUCCESS (0x00)	函数执行成功。
HI_ERR_MD_INVALID_CHNNUM (0x02)	无效的通道号。
HI_ERR_MD_CHN_NOT_CREATE (0x0E)	未创建逻辑编码通道。
HI_ERR_MD_CHN_NOT_STARTVENC (0x0D)	未启动逻辑编码通道。
HI_ERR_MD_CHN_FAILED_SETTYPE (0x06)	设置移动侦测类型失败。
HI_ERR_MD_CHN_FAILED_ENABLE (0x07)	使能通道的移动侦测功能失败。

HI_MD_Disable

【目的】

禁用指定通道的移动侦测功能

【语法】

```
HI_S32 HI_MD_Disable (HI_S32 ChnID );
```

【参数】

参数名称	描述	输入/输出
ChnID	表示视频编码通道号	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	描述
HI_SUCCESS (0x00)	函数执行成功。
HI_ERR_MD_INVALID_CHNNUM (0x02)	无效的通道号。
HI_ERR_MD_CHN_NOT_CREATE (0x0E)	未创建逻辑编码通道。
HI_ERR_MD_CHN_NOT_STARTMD (0x0C)	未使能通道的移动侦测功能。
HI_ERR_MD_CHN_FAILED_DISABLE (0x08)	禁止通道的移动侦测功能失败。

HI_MD_GetEnable

【目的】

获取当前该视频编码通道的移动侦测功能是否打开

【语法】

```
HI_S32 HI_MD_GetEnable (HI_S32 ChnID, HI_BOOL *pbEnable);
```

【参数】

参数名称	描述	输入/输出
ChnID	表示视频编码通道号。	输入
pbEnable	指针类型，表示输出 TRUE 或 FALSE 标示移动侦测功能是否打开。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	描述
HI_SUCCESS (0x00)	函数执行成功。
HI_ERR_MD_INVALID_PTR (0x03)	无效的参数指针。
HI_ERR_MD_INVALID_CHNNUM (0x02)	无效的通道号。
HI_ERR_MD_CHN_NOT_CREATE (0x0E)	未创建逻辑编码通道。
HI_ERR_MD_CHN_FAILED_GETSTATUS (0x04)	查询通道的移动侦测功能是否使能失败。

HI_MD_GetStatus

【目的】

获取移动侦测状态

【语法】

```
HI_S32 HI_MD_GetStatus (HI_S32 ChnID, MD_STATUS_S *pStatus);
```

【参数】

参数名称	描述	输入/输出
ChnID	视频编码通道号。	输入
pStatus	移动侦测状态指针，内存由调用者分配，在调用成功后，本函数会将当前的移动侦测状态拷贝至 pStatus 所指的内存。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	描述
HI_SUCCESS　(0x00)	函数执行成功。
HI_ERR_MD_INVALID_PTR　(0x03)	无效的参数指针。
HI_ERR_MD_INVALID_CHNNUM　(0x02)	无效的通道号。
HI_ERR_MD_CHN_NOT_STARTMD　(0x0C)	通道的移动侦测功能未使能。
HI_ERR_MD_CHN_NOT_UPDATE　(0x0A)	没有更新的数据。

3 如何实现移动侦测

关于本章

本章描述内容如下表所示。

标题	内容
3.1 实现单个侦测区域的移动侦测	详细介绍单个区域移动侦测的实现源码。
3.2 实现多个侦测区域的移动侦测	给出普通场景下多个侦测区域移动侦测的实现源码。
3.3 调用 API 得到 SAD	详细介绍如何调用 API 得到 SAD。
3.4 应用界面示例	给出了一个实际的应用界面。

3.1 实现单个侦测区域的移动侦测

判断单个侦测区域是否报警的源代码如下。

```
/* 判断单个侦测区域是否报警 */
HI_BOOL Should_Alarm
(
    HI_S32 chno,
    HI_U16 avg_SADValue[MAX_MACROCELL_NUM][VIDEOINPUT_MAX_CHANNEL_NUM],
    MD_AREA_CFG_S*  pmd
)
{
    HI_S32 i;
```

```
    HI_U32 Area_SAD_Sum = 0;                /*侦测区域中宏块的平均SAD累计值*/
    HI_U32 Area_macrocell_num = 0;          /*侦测区域中的宏块个数*/
    HI_S32 Area_Alarm_macrocell_num = 0;    /*侦测区域中报警宏块个数*/

/*把侦测区域内每个宏块的avg_SADValue[i][chno]与SAD做比较*/
    for (i=0; i<MAX_MACROCELL_NUM; i++)
    {
        if (1 == pmd->mask[i])
        {
            Area_macrocell_num++;
            if (avg_SADValue[i][chno] >= pmd->Macro_threshold)
            {
                Area_Alarm_macrocell_num++;
            }
        }
    }

    /*如果区域中报警宏块比例超过触发门限就触发报警 */
    if (Area_Alarm_macrocell_num * pmd->>=Macro_ratio >=
  Area_macrocell_num )
    {
        return HI_TRUE;
    }
    else
    {
        return HI_FALSE;
    }
}
```

其中，MD_AREA_CFG_S 结构体定义如下：

```
typedef struct hiMD_AREA_CFG_S
{
    HI_S32   chno;                  /* 通道号*/
    HI_CHAR  name[32];              /* 单个移动侦测区域名 */
    HI_BOOL  enable_flag;           /* 用户指定是否使能该区域进行移动侦测*/
    HI_BOOL  run_flag;              /*该区域正在进行移动侦测标志*/
    RECT_S   md_area;               /* 单个移动侦测区域 */
    HI_S32   Macro_threshold;       /* 宏块阈值 */
    HI_S32   Macro_ratio;           /* 区域中报警宏块比例 */
    HI_S32   FrameInterval;         /* 帧间隔，每隔几帧做一次移动侦测处理 */
    HI_S8     mask[MAX_MACROCELL_NUM];  /* 单个移动侦测区域对应的宏块号 */
    struct hiMD_AREA_CFG_S *pNext;      /* 指向下一个区域的指针 */
```

```
}MD_AREA_CFG_S;
```

3.2 实现多个侦测区域的移动侦测

在普通场景下判断多个侦测区域是否报警的源代码如下。

```
/* 在视频窗口内如果有一个侦测区域报警，此窗口就输出报警 */
HI_S32 HI_CHN_MD_Algorithm(HI_S32 chno,HI_U8 status[])
{
    STATIC HI_U16  frameCount[VIDEOINPUT_MAX_CHANNEL_NUM]={0,0,0,0};   /*
帧计数*/
    STATIC HI_U16  SAD_Sum[MAX_MACROCELL_NUM][VIDEOINPUT_MAX_CHANNEL_NUM];
/*宏块SAD值的累计值(共4路通道的图像)*/
    MD_AREA_CFG_S *temp_pmd;       /*指向移动侦测区域的临时指针 */
    HI_U32 i;
    STATIC HI_U64 temp_frame_num = 0;

    /*在帧间隔时间之内累加宏块的绝对差值*/
    if( frameCount[chno] < temp_pmd->FrameInterval)
    {
        for (i=0;i<MAX_MACROCELL_NUM;i++)
        {
            SAD_Sum[i][chno] += status[i];
        }
        frameCount[chno]++;
    }

/*每几帧进行一次运动侦测处理*/
    else if ( frameCount[chno] >= temp_pmd->FrameInterval) {
        /*判定每个移动侦测区域是否应该报警*/
        temp_pmd = pmd_head;
        while( NULL != temp_pmd )
        {
            if( (temp_pmd->chno == chno) && (HI_TRUE == temp_pmd-
>enable_flag))
            {
                if(HI_TRUE == Should_Alarm(chno,SAD_Sum,temp_pmd))
                {
                    HI_MCTP_Send_MDAlert(chno,temp_pmd->name);
                    HI_OUT_Printf(0,"\r\n!!!Alarm!!! chn %d md %s
num:%d\r\n\r\n",chno,temp_pmd->name,temp_frame_num++);
                }
```

```
            }
            temp_pmd = temp_pmd->pNext;
        }

        /*帧计数和累计值归零*/
        frameCount[chno] = 0;
        for( i = 0;i<MAX_MACROCELL_NUM;i++)
        {
            SAD_Sum[i][chno] = 0;
        }
    }

    return HI_SUCCESS;
}
```

3.3 调用 API 得到 SAD

调用 API 得到 SAD 的源码如下。

```
HI_VOID* ProcessMoveDetect(HI_VOID* arg)
{
    HI_S32 chno;
    HI_S32 ret;
    MD_STATUS_S md_status;

    chno = *((HI_S32*)arg);  /*正在移动侦测的通道号*/

    ret = HI_MD_GetStatus(encode_attr.av_chno[chno].vechno,&md_status);

    /*进行报警算法*/
    if( HI_SUCCESS == ret)
    {
        HI_CHN_MD_Algorithm(chno,md_status.u8MDValue);
    }

    return HI_SUCCESS;
}
```

其中，结构体 MD_STATUS_S 结构定义如下。

```
typedef struct hiMD_STATUS_S
{
    HI_U8  u8Mask[MAX_MACROCELL_NUM];     /* 移动侦测宏块侦测标识 */
```

```
    HI_U8  u8MDValue[MAX_MACROCELL_NUM];  /* SAD，取值0～100表示 */
} MD_STATUS_S;
```

3.4 应用界面示例

基于 Hi3510 的移动侦测解决方案可以在板端实现。在 PC 机上，移动侦测界面如图 3-1 所示，用户在界面上可以设置联动报警功能、移动侦测时间计划。开发者可以很方便的调用 Hi3510 移动侦测模块提供的 API 接口函数，根据需求设计出界面整洁美观、功能丰富方便的移动侦测应用方案。

图3-1 应用界面

A 数据类型定义

```
typedef unsigned char          HI_U8;
typedef unsigned char          HI_UCHAR;
typedef unsigned short         HI_U16;
typedef unsigned int           HI_U32;
typedef char                  HI_S8;
typedef short                 HI_S16;
typedef int                   HI_S32;
typedef char                  HI_CHAR;
typedef void                  HI_VOID;
```