



HiIVE

## API 参考

文档版本 05

发布日期 2015-12-15

**版权所有 © 深圳市海思半导体有限公司 2014-2015。保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## **商标声明**



**HISILICON**、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

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**

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，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。

## **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：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：[support@hisilicon.com](mailto:support@hisilicon.com)



# 前 言

## 概述

本文档为使用海思媒体处理芯片的 IVE 协处理器进行智能分析方案开发的程序员而写，目的是供您在开发过程中查阅 IVE 协处理器支持的各种参考信息，包括 API、头文件、错误码、Proc 信息等。



### 说明

- 本文未做特殊说明，Hi3516D 与 Hi3516A 完全一致。
- 本文未做特殊说明，Hi3520DV300 与 Hi3521A 完全一致。
- 本文未做特殊说明，Hi3518EV201, Hi3516CV200 与 Hi3518EV200 完全一致。

## 产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3516A	V100
Hi3516D	V100
Hi3536	V100
Hi3521A	V100
Hi3531A	V100
Hi3520D	V300
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3519	V100








## 读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

## 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	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	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	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	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

## 修订记录

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。

日期	版本	修改描述
2015-12-15	05	第 3 章，修改 IVE_CCBLOB_S 的【成员】
2015-09-20	04	新增 FAQ 章节，添加 Hi3519 的相关内容
2015-07-29	03	新增 Hi3521A/Hi3520D V300、Hi3531A、Hi3518E V200/V201 和 Hi3516C V200 相关内容
2015-01-19	02	第 2 章 HI_MPI_IVE_Query 中的【举例】有更改
2014-12-20	01	添加 Hi3516D 及 Hi3536 的相关内容 第 3 章 新增 IVE_SRC_DATA_S 和 IVE_DST_DATA_S



日期	版本	修改描述
2014-09-25	00B03	1.2.1 章节增加“这样可以与后续任务组链执行，减少中断次数，提升性能”
2014-09-14	00B02	修改 5.2 章节调试信息及 IVE RUN-TIME INFO IVE 参数说明 第二次版本发布
2014-07-15	00B01	第一次版本发布。



## 目 录

<b>1 概述.....</b>	<b>1</b>
1.1 概述.....	1
1.2 功能描述.....	1
1.2.1 重要概念 .....	1
1.2.2 使用示意 .....	11
<b>2 API 参考 .....</b>	<b>12</b>
<b>3 数据类型和数据结构 .....</b>	<b>127</b>
<b>4 错误码.....</b>	<b>213</b>
4.1 IVE 错误码 .....	213
<b>5 Proc 调试信息.....</b>	<b>215</b>
5.1 概述.....	215
5.2 Proc 信息说明 .....	215
<b>6 FAQ.....</b>	<b>219</b>
6.1 使用 PC 端 IVE Clib 与板端 IVE SDK 开发算法的差异 .....	219
6.2 使用 IVE 与 OpenCV 开发算法的区别 .....	220
6.3 ANN/SVM 查找表的建立.....	220
6.4 Cache 内存的使用 .....	220



## 插图目录

图 1-1 跨度 (stride) 示意图.....	2
图 1-2 IVE_IMAGE_TYPE_U8C1 \IVE_IMAGE_TYPE_S8C1 \IVE_IMAGE_TYPE_S16C1 \IVE_IMAGE_TYPE_U16C1 \IVE_IMAGE_TYPE_S32C1 \IVE_IMAGE_TYPE_U32C1 \IVE_IMAGE_TYPE_S64C1 \IVE_IMAGE_TYPE_U64C1 类型的 IVE_IMAGE_S 图像 .....	5
图 1-3 IVE_IMAGE_TYPE_YUV420SP 类型的 IVE_IMAGE_S 图像 .....	5
图 1-4 IVE_IMAGE_TYPE_YUV422SP 类型的 IVE_IMAGE_S 图像 .....	6
图 1-5 IVE_IMAGE_TYPE_YUV420P 类型的 IVE_IMAGE_S 图像 .....	6
图 1-6 IVE_IMAGE_TYPE_YUV422P 类型的 IVE_IMAGE_S 图像 .....	7
图 1-7 IVE_IMAGE_TYPE_S8C2_PACKAGE 类型的 IVE_IMAGE_S 图像.....	7
图 1-8 IVE_IMAGE_TYPE_S8C2_PLANAR 类型的 IVE_IMAGE_S 图像 .....	8
图 1-9 IVE_IMAGE_TYPE_U8C3_PACKAGE 类型的 IVE_IMAGE_S 图像 .....	8
图 1-10 IVE_IMAGE_TYPE_U8C3_PLANAR 类型的 IVE_IMAGE_S 图像.....	9
图 1-11 IVE_DATA_S 类型的数据内存示意.....	10
图 1-12 IVE_MEM_INFO_S 类型的数据内存示意 .....	10
图 1-13 积分图 (IVE_IMAGE_TYPE_U64C1) 组合输出示意.....	10
图 1-14 直方图输出格式示意.....	10
图 2-1 快速拷贝示意图 .....	15
图 2-2 间隔拷贝示意图 .....	16
图 2-3 Filter 计算公式示意图 .....	18
图 2-4 Sobel 计算公式示意图 .....	24
图 2-5 MagAndAng 计算示意图.....	28
图 2-6 Dilate 计算公式示意图.....	31
图 2-7 Erode 计算公式示意图 .....	34
图 2-8 Thresh 8 种阈值化模式示意图.....	37
图 2-9 Thresh_S16 4 种阈值化模式示意图 .....	47
图 2-10 Thresh_U16 2 种阈值化模式示意图 .....	49



图 2-11 16BitTo8Bit 4 种转换模式示意图 .....	52
图 2-12 灰度图像 GMM 模型的内存配置示意图 .....	67
图 2-13 RGB 图像 GMM 模型的内存配置示意图 .....	68
图 2-14 灰度图像 GMM2 模型的内存配置示意图 .....	71
图 2-15 RGB 图像 GMM2 模型的内存配置示意图 .....	71
图 2-16 LBP 计算公式示意图 .....	76
图 2-17 NormGrad 计算公式示意图 .....	79
图 2-18 3 层金字塔 LK 光流计算示意图 .....	82
图 2-19 3 层金字塔 LK 光流计算示意图 .....	85
图 2-20 前景状态标志图形单个像素各比特位示意图 .....	97
图 2-21 ANN_MLP 输入样本向量示意图 .....	105
图 2-22 ANN_MLP 输出预测结果示意图 .....	105
图 2-23 ANN_MLP 输入样本向量数组示意图 .....	108
图 2-24 ANN_MLP 输出预测结果示意图 .....	108
图 2-25 SVM 输入样本向量示意图 .....	112
图 2-26 SVM 预测结果示意图 .....	113
图 2-27 SVM 输入样本向量数组示意图 .....	115
图 2-28 SVM 预测结果示意图 .....	116
图 2-29 CNN 输出特征向量数组示意图 .....	120
图 2-30 CNN 网络模型示意图 .....	121
图 2-31 CNN 各样本预测结果示意图 .....	123





# 1 概述

## 1.1 概述

IVE (Intelligent Video Engine) 是海思媒体处理芯片智能分析系统中的硬件加速模块。用户基于 IVE 开发智能分析方案可以加速智能分析，降低 CPU 占用。当前 IVE 提供的算子可以支撑开发视频诊断、周界防范等智能分析方案。

## 1.2 功能描述

### 1.2.1 重要概念

- 句柄(handle)  
用户在调用算子创建任务时，系统会为每个任务分配一个 handle，用于标识不同的任务。
- 及时返回结果标志 bInstant  
用户在创建某个任务后，希望及时得到该任务完成的信息，则需要在创建该任务时，将 bInstant 设置为 HI\_TRUE。否则，如果用户不关心该任务是否完成，建议将 bInstant 设置为 HI\_FALSE，这样可以与后续任务组链执行，减少中断次数，提升性能。
- 查询(query)  
用户根据系统返回的 handle，调用 [HI\\_MPI\\_IVE\\_Query](#) 可以查询对应算子任务是否完成。
- 及时刷 cache  
IVE 硬件只能从 DDR 中获取数据。如果用户在调用 IVE 任务时，访问空间可 cache 而且 CPU 曾经访问，为了保证 IVE 输入输出数据不被 CPU cache 干扰，此时用户需要调用 [HI\\_MPI\\_SYS\\_MmzFlushCache](#) 接口刷 cache（详细信息请参见《HiMPP Vx.y 媒体处理软件开发参考》），将数据从 cache 刷到 DDR，以供 IVE 使用。
- 跨度 (stride): 与图像或二维数据的 width 度量一致的量，如[图 1-1](#) 所示。
  - IVE\_IMAGE\_S 图像数据跨度，表示图像一行以“像素”计算的单元个数，“像素”位宽可以是 8bit, 16bit 等。



类型	图像描述	内存地址	跨度
IVE_IMAGE_TYPE_S8C1	8bit 有符号单通道图像，如图 1-2 所示	仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]	仅用到 u16Stride[0]
IVE_IMAGE_TYPE_YUV420SP	YCbCr420 SemiPlannar 数据格式图像，如图 1-3 所示	内存地址仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]（亮度 Y）和 u32PhyAddr[1]、pu8VirAddr[1]（色度，VU 间隔排列）；亮度和色度内存可以不连续，但不推荐	跨度仅用到 u16Stride[0]（亮度跨度）和 u16Stride[1]（色度跨度）
IVE_IMAGE_TYPE_YUV422SP	YcbCr422 SemiPlannar 数据格式图像，如图 1-4 所示	内存地址仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]（亮度 Y）和 u32PhyAddr[1]、pu8VirAddr[1]（色度，VU 间隔存储）；亮度和色度内存可以不连续，但不推荐	跨度仅用到 u16Stride[0]（亮度跨度）和 u16Stride[1]（色度跨度）
IVE_IMAGE_TYPE_YUV420P	YCbCr420 Planar 数据格式图像，如图 1-5 所示	内存地址用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]（亮度 Y），u32PhyAddr[1]、pu8VirAddr[1]（色度 U）和 u32PhyAddr[2]、pu8VirAddr[2]（色度 V）；Y、U、V 内存可不连续，但不推荐	跨度用到 u16Stride[0]（亮度 Y 跨度）、u16Stride[1]（色度 U 跨度）和 u16Stride[2]（色度 V 跨度）
IVE_IMAGE_TYPE_YUV422P	YCbCr422 Planar 数据格式图像，如图 1-6 所示	内存地址用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]（亮度 Y），u32PhyAddr[1]、pu8VirAddr[1]（色度 U）和 u32PhyAddr[2]、pu8VirAddr[2]（色度 V）	跨度用到 u16Stride[0]（亮度跨度）、u16Stride[1]（色度 U 跨度）和 u16Stride[2]（色度 V 跨度）
IVE_IMAGE_TYPE_S8C2_PACKAGE	8bit 有符号双通道且以 Package 格式存储的图像，如图 1-7 所示	内存地址仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]	跨度仅用到 u16Stride[0]



类型	图像描述	内存地址	跨度
IVE_IMAGE_TYPE_S8C2_PLANAR	8bit 有符号双通道且以 Planar 格式存储的图像，如图 1-8 所示	内存地址仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]和 u32PhyAddr[1]、pu8VirAddr[1]	跨度仅用到 u16Stride[0]和 u16Stride[1]
IVE_IMAGE_TYPE_S16C1	16bit 有符号单通道图像，如图 1-2 所示	内存地址仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]	跨度仅用到 u16Stride[0]
IVE_IMAGE_TYPE_U16C1	16bit 无符号单通道图像，如图 1-2 所示	内存地址仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]	跨度仅用到 u16Stride[0]
IVE_IMAGE_TYPE_U8C3_PACKAGE	8bit 无符号三通道且以 Package 格式存储的图像，如图 1-9 所示	内存地址仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]	跨度仅用到 u16Stride[0]
IVE_IMAGE_TYPE_U8C3_PLANAR	8bit 无符号三通道且以 Planar 格式存储的图像，如图 1-10 所示；	内存地址用到了 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]，u32PhyAddr[1]、pu8VirAddr[1]和 u32PhyAddr[2]、pu8VirAddr[2]；	跨度用到了 u16Stride[0]、u16Stride[1]和 u16Stride[2]；
IVE_IAMGE_TYPE_S32C1	32bit 有符号单通道图像，如图 1-2 所示；	内存地址仅用到了 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]；	跨度仅用到了 u16Stride[0]；
IVE_IMAGE_TYPE_U32C1	32bit 无符号单通道图像，如图 1-2 所示；	内存地址仅用到了 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]；	跨度仅用到了 u16Stride[0]；
IVE_IMAGE_TYPE_S64C1	64bit 有符号单通道图像，如图 1-2 所示；	内存地址仅用到了 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]；	跨度仅用到了 u16Stride[0]；
IVE_IMAGE_TYPE_U64C1	64bit 无符号单通道图像，如图 1-2 所示；	内存地址仅用到了 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]；	跨度仅用到了 u16Stride[0]；

- 特殊输出数据类型
  - Integ 组合输出 (IVE\_INTEG\_OUT\_CTRL\_COMBINE)
  - 用 IVE\_IMAGE\_TYPE\_U64C1 类型的 IVE\_IMAGE\_S, S (图像和) 占低 28bit, SQ (图像平方和) 占高 36bit。格式如图 1-13 所示。
  - 直方图输出如图 1-14 所示。

图1-2 IVE\_IMAGE\_TYPE\_U8C1 \ IVE\_IMAGE\_TYPE\_S8C1 \ IVE\_IMAGE\_TYPE\_S16C1 \ IVE\_IMAGE\_TYPE\_U16C1 \ IVE\_IMAGE\_TYPE\_S32C1 \ IVE\_IMAGE\_TYPE\_U32C1 \ IVE\_IMAGE\_TYPE\_S64C1 \ IVE\_IMAGE\_TYPE\_U64C1 类型的 IVE\_IMAGE\_S 图像

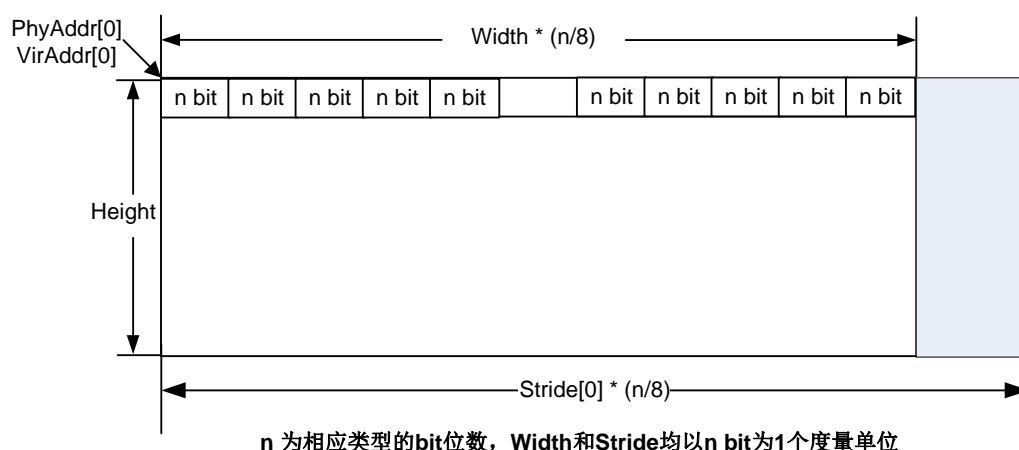


图1-3 IVE\_IMAGE\_TYPE\_YUV420SP 类型的 IVE\_IMAGE\_S 图像

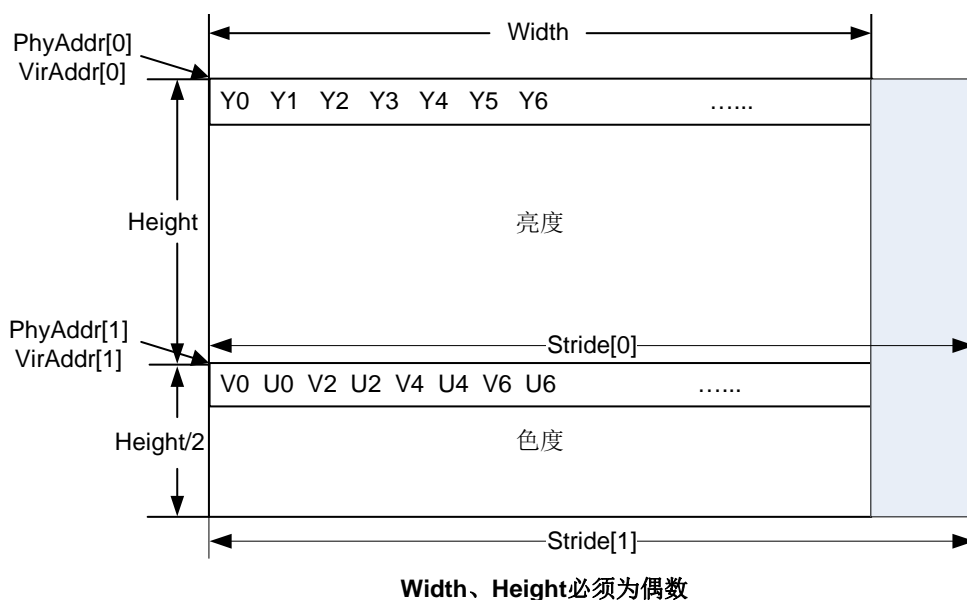
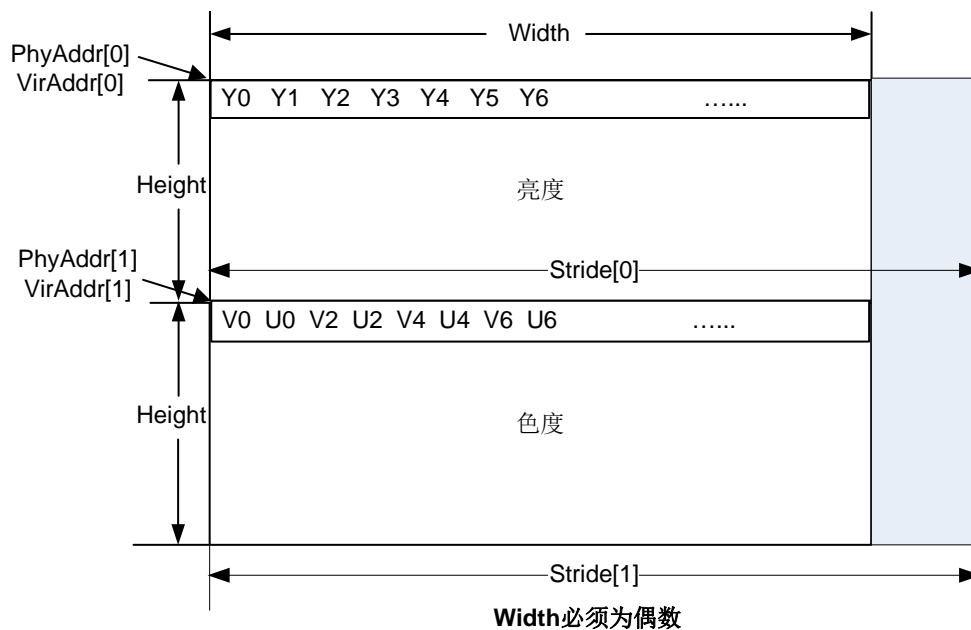


图1-4 IVE\_IMAGE\_TYPE\_YUV422SP 类型的 IVE\_IMAGE\_S 图像



注：这里 V 在前，U 在后，u32PhyAddr[2]和 pu8VirAddr[2]可配置为 U 的首地址，即 u32PhyAddr[1]+1 和 pu8VirAddr[1]+1。

图1-5 IVE\_IMAGE\_TYPE\_YUV420P 类型的 IVE\_IMAGE\_S 图像

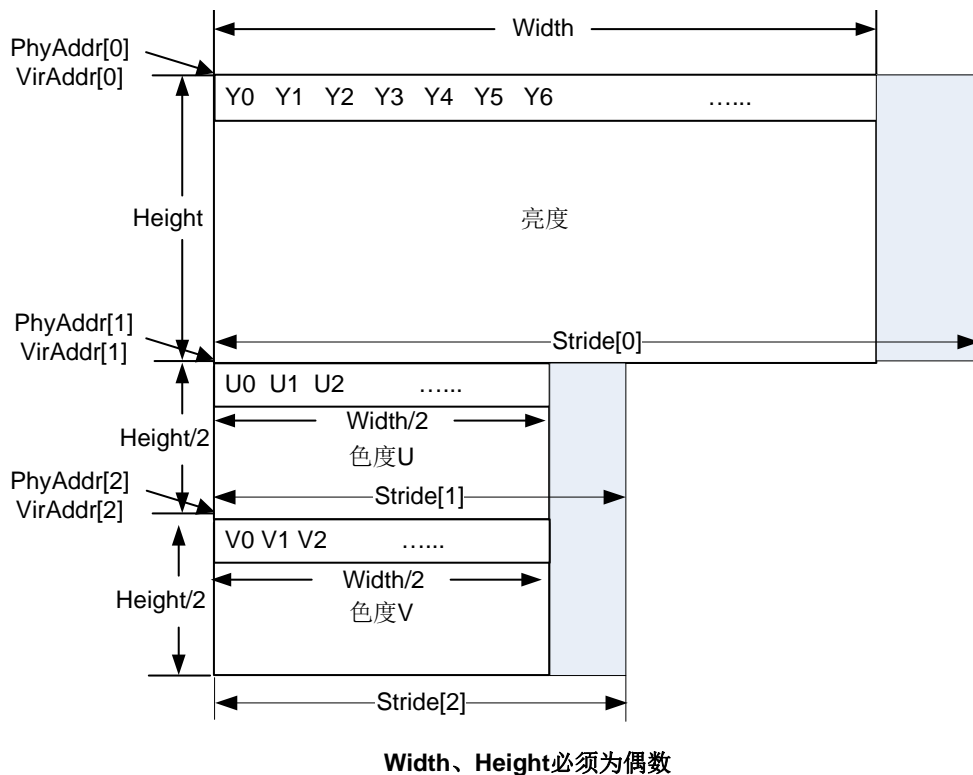


图1-6 IVE\_IMAGE\_TYPE\_YUV422P 类型的 IVE\_IMAGE\_S 图像

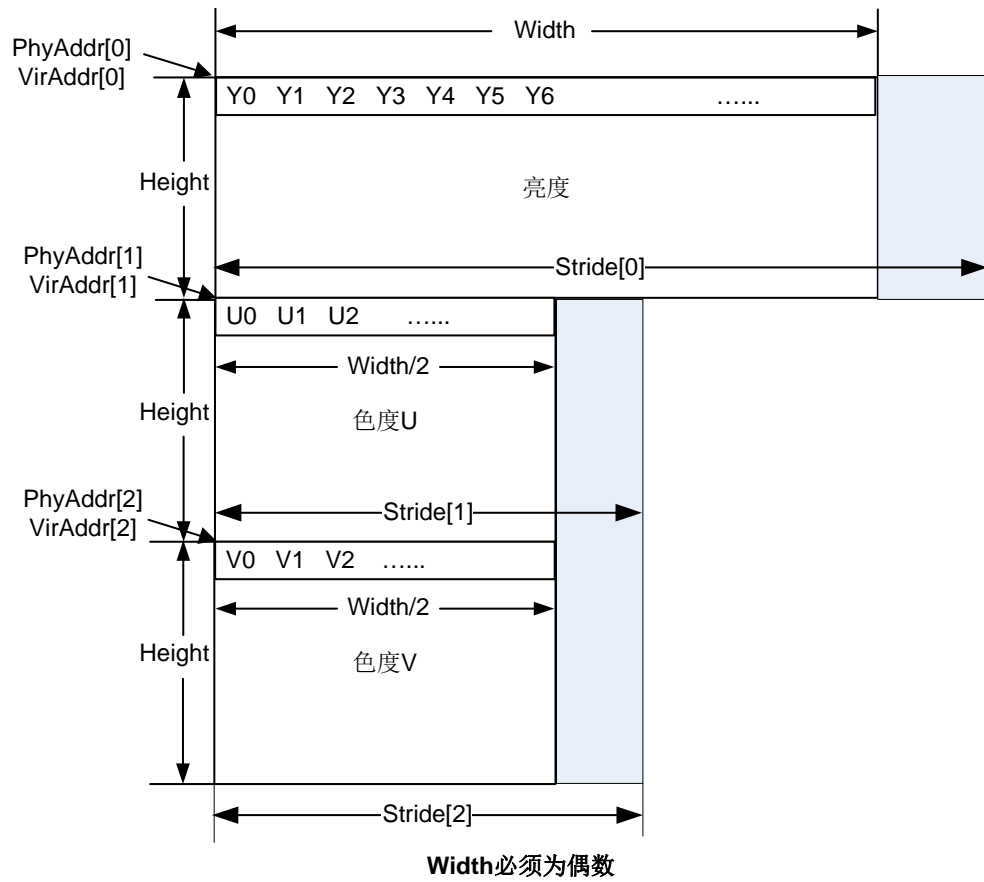


图1-7 IVE\_IMAGE\_TYPE\_S8C2\_PACKAGE 类型的 IVE\_IMAGE\_S 图像

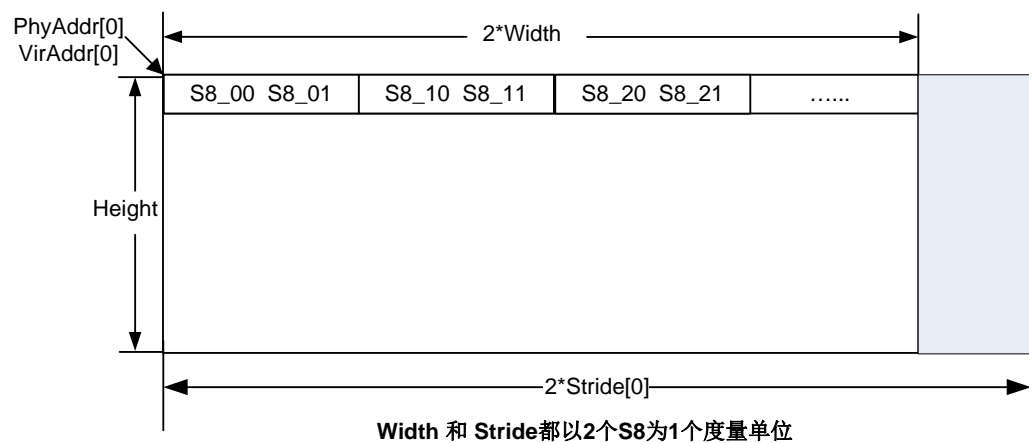


图1-8 IVE\_IMAGE\_TYPE\_S8C2\_PLANAR 类型的 IVE\_IMAGE\_S 图像

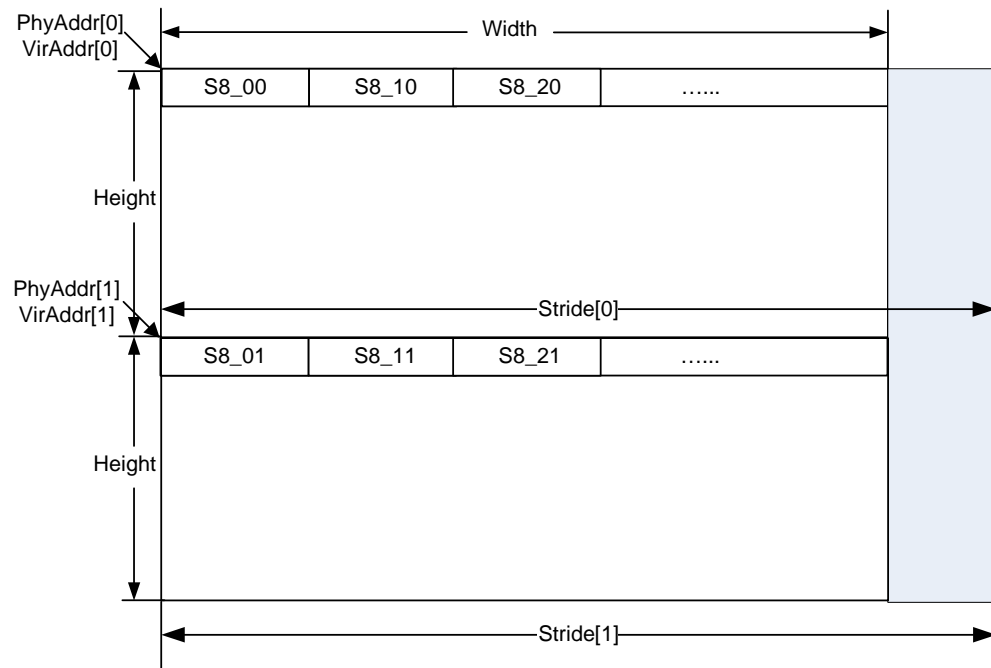
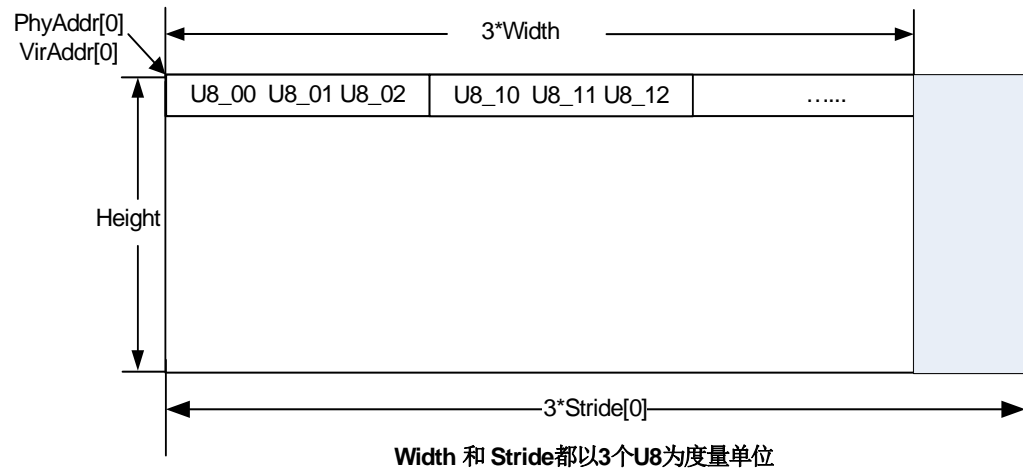
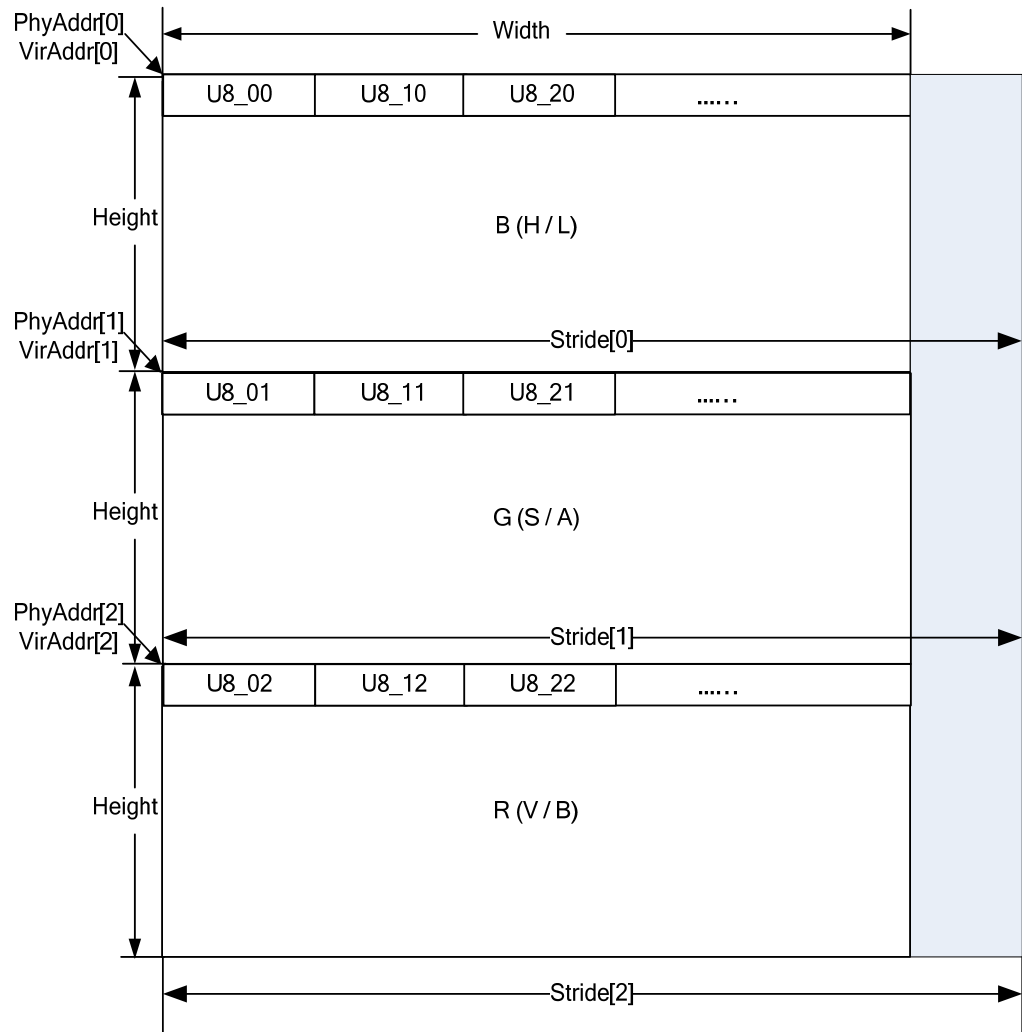


图1-9 IVE\_IMAGE\_TYPE\_U8C3\_PACKAGE 类型的 IVE\_IMAGE\_S 图像



注：对于 RGB\_PACKAGE 图像，是以 “B0G0R0B1G1R1...” 形式存储，B 在最前面；  
对于 HSV\_PACKAGE 图像，是以 “H0S0V0H1S1V1...” 形式存储，H 在最前面；  
对于 LAB\_PACKAGE 图像，是以 “L0A0B0L1A1B1...” 形式存储，L 在最前面。

图1-10 IVE\_IMAGE\_TYPE\_U8C3\_PLANAR 类型的 IVE\_IMAGE\_S 图像



注：对于 RGB\_PLANAR 图像，指针数组 `VirAddr[3]` 按顺序分别存储 B、G、R 的指针，而数组 `Stride[3]` 分别为 B、G、R 的跨度；  
 对于 HSV\_PLANAR 图像，指针数组 `VirAddr[3]` 按顺序分别存储 H、S、V 的指针，而数组 `Stride[3]` 分别为 H、S、V 的跨度；  
 对于 LAB\_PLANAR 图像，指针数组 `VirAddr[3]` 按顺序分别存储 L、A、B 的指针，而数组 `Stride[3]` 分别为 L、A、B 的跨度；



图1-11 IVE\_DATA\_S 类型的数据内存示意

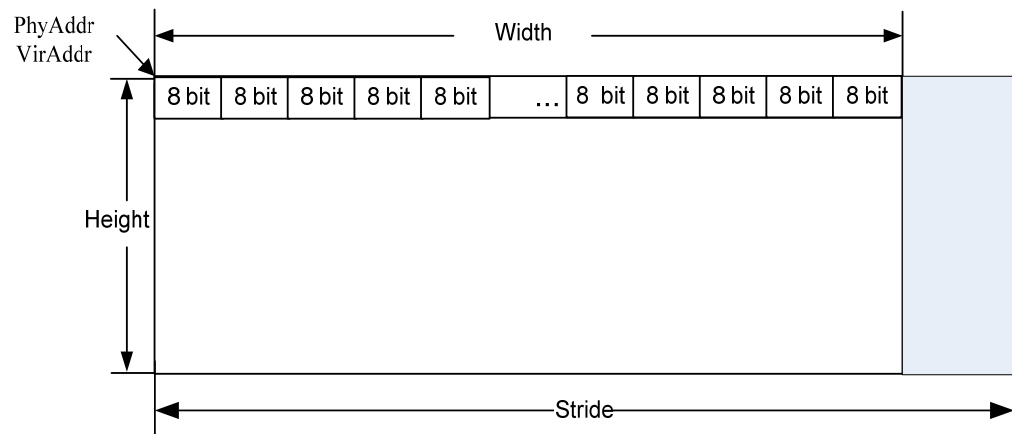


图1-12 IVE\_MEM\_INFO\_S 类型的数据内存示意

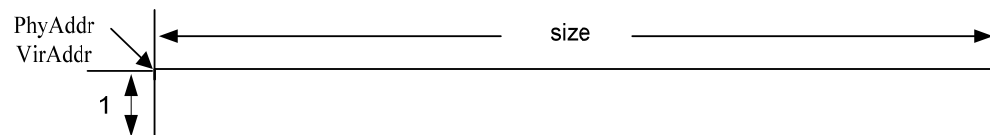


图1-13 积分图 (IVE\_IMAGE\_TYPE\_U64C1) 组合输出示意

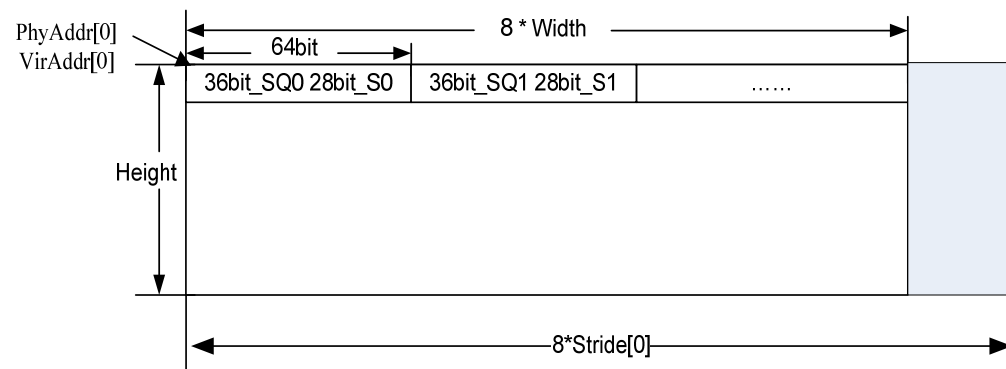
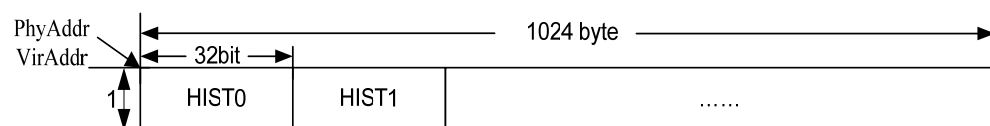


图1-14 直方图输出格式示意





## 1.2.2 使用示意

- 用户根据需求调用相应的算子接口创建任务，指定 `bInstant` 类型，并记录该任务返回的 `handle` 号。
- 根据返回的 `handle` 号，指定阻塞方式，可以查询到该任务的完成状态。  
具体可参见 [HI\\_MPI\\_IVE\\_Query](#) 中的【举例】。



## 2 API 参考

IVE 模块提供了创建任务和查询任务的基本接口。

该功能模块提供以下 MPI:

- [HI\\_MPI\\_IVE\\_DMA](#): 创建直接内存访问任务。
- [HI\\_MPI\\_IVE\\_Filter](#): 创建 5x5 模板滤波任务。
- [HI\\_MPI\\_IVE\\_CSC](#): 创建色彩空间转换任务。
- [HI\\_MPI\\_IVE\\_FilterAndCSC](#): 创建模板滤波加色彩空间转换复合任务。
- [HI\\_MPI\\_IVE\\_Sobel](#): 创建 5x5 模板 Sobel-like 梯度计算任务。
- [HI\\_MPI\\_IVE\\_MagAndAng](#): 创建 5x5 模板计算梯度幅值与幅角任务。
- [HI\\_MPI\\_IVE\\_Dilate](#): 创建膨胀任务。
- [HI\\_MPI\\_IVE\\_Erode](#): 创建腐蚀任务。
- [HI\\_MPI\\_IVE\\_Thresh](#): 创建图像二值化任务。
- [HI\\_MPI\\_IVE\\_And](#): 创建两图像相与任务。
- [HI\\_MPI\\_IVE\\_Sub](#): 创建两图像相减任务。
- [HI\\_MPI\\_IVE\\_Or](#): 创建两图像相或任务。
- [HI\\_MPI\\_IVE\\_Integ](#): 创建积分图统计任务。
- [HI\\_MPI\\_IVE\\_Hist](#): 创建直方图统计任务。
- [HI\\_MPI\\_IVE\\_Thresh\\_S16](#): 创建 S16 数据到 8bit 数据阈值化任务。
- [HI\\_MPI\\_IVE\\_Thresh\\_U16](#): 创建 U16 数据到 U8 数据阈值化任务。
- [HI\\_MPI\\_IVE\\_16BitTo8Bit](#): 创建 16bit 数据到 8bit 数据线性转化任务。
- [HI\\_MPI\\_IVE\\_OrdStatFilter](#): 创建 3x3 模板顺序统计量滤波任务。
- [HI\\_MPI\\_IVE\\_Map](#): 创建 Map (映射 U8->U8 赋值) 任务。
- [HI\\_MPI\\_IVE\\_Map](#): 创建 Map (映射 U8->U8\U8->U16\U8->S16 赋值) 任务。
- [HI\\_MPI\\_IVE\\_EqualizeHist](#): 创建灰度图像的直方图均衡化计算任务。
- [HI\\_MPI\\_IVE\\_Add](#): 创建两灰度图像的加权加计算任务。
- [HI\\_MPI\\_IVE\\_Xor](#): 创建两二值图的异或计算任务。
- [HI\\_MPI\\_IVE\\_NCC](#): 创建两相同分辨率图像的归一化互相关系数计算任务。
- [HI\\_MPI\\_IVE\\_CCL](#): 创建二值图像的连通区域标记任务。



- [HI\\_MPI\\_IVE\\_GMM](#): 创建 GMM 背景建模任务。
- [HI\\_MPI\\_IVE\\_GMM2](#): 创建 GMM2 背景建模任务。
- [HI\\_MPI\\_IVE\\_CannyHysEdge](#): 创建灰度图的 Canny 强弱边缘提取任务。
- [HI\\_MPI\\_IVE\\_CannyEdge](#): 灰度图的 Canny 边缘提取的后半部: 连接边缘点, 形成 Canny 边缘图。
- [HI\\_MPI\\_IVE\\_LBP](#): 创建 LBP 计算任务。
- [HI\\_MPI\\_IVE\\_NormGrad](#): 创建归一化梯度计算任务, 梯度均分量均归一化到 S8。
- [HI\\_MPI\\_IVE\\_LKOpticalFlow](#): 创建单层 LK 光流计算任务。
- [HI\\_MPI\\_IVE\\_LKOpticalFlowPyr](#): 创建多层金字塔 LK 光流计算任务。
- [HI\\_MPI\\_IVE\\_STCandiCorner](#): 灰度图像 Shi-Tomasi-like 角点计算的前半部: 计算候选角点。
- [HI\\_MPI\\_IVE\\_STCorner](#): 灰度图像 Shi-Tomasi-like 角点计算的后半部: 按规则挑选角点。
- [HI\\_MPI\\_IVE\\_SAD](#): 计算两幅图像按 4x4\8x8\16x16 分块的 16 bit\8 bit SAD 图像, 以及对 SAD 进行阈值化输出。
- [HI\\_MPI\\_IVE\\_Resize](#): 创建图像缩放任务。
- [HI\\_MPI\\_IVE\\_GradFg](#): 根据背景图像和当前帧图像的梯度信息计算梯度前景图像。
- [HI\\_MPI\\_IVE\\_MatchBgModel](#): 基于 CodeBook 演进的背景模型匹配。
- [HI\\_MPI\\_IVE\\_UpdateBgModel](#): 基于 CodeBook 演进的背景模型更新。
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_LoadModel](#): 读取 ANN\_MLP 模型文件, 初始化模型数据。
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_UnloadModel](#): 去初始化 ANN 模型数据。
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_Predict](#): 创建单个样本 ANN\_MLP 预测任务。
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_Predict](#): 创建同一模型多个样本 ANN\_MLP 预测任务。
- [HI\\_MPI\\_IVE\\_SVM\\_LoadModel](#): 读取 SVM 模型文件, 初始化模型数据。
- [HI\\_MPI\\_IVE\\_SVM\\_UnloadModel](#): 去初始化 SVM 模型数据。
- [HI\\_MPI\\_IVE\\_SVM\\_Predict](#): 创建单个样本 SVM 预测任务。
- [HI\\_MPI\\_IVE\\_SVM\\_Predict](#): 创建同一模型的多个样本 SVM 预测任务。
- [HI\\_MPI\\_IVE\\_CNN\\_LoadModel](#): 读取 CNN 模型文件, 生成 CNN 网络模型。
- [HI\\_MPI\\_IVE\\_CNN\\_UnloadModel](#): 卸载 CNN 网络模型, 释放内存。
- [HI\\_MPI\\_IVE\\_CNN\\_Predict](#): 用已有模型对一个或多个输入样本进行预测, 并输出预测结果。
- [HI\\_MPI\\_IVE\\_CNN\\_GetResult](#): 通过 CNN 全连接最后一层输出结果, 执行 Softmax 运算来预测每个样本图像类别及置信度。
- [HI\\_MPI\\_IVE\\_Query](#): 查询已创建任务完成情况。

## HI\_MPI\_IVE\_DMA

### 【描述】



创建直接内存访问任务，支持快速拷贝、间隔拷贝、内存填充：可实现数据从一块内存快速拷贝到另一块内存，或者从一块内存有规律的拷贝一些数据到另一块内存，或者对一块内存进行填充操作。

### 【语法】

```
HI_S32 HI_MPI_IVE_DMA(IVE_HANDLE *pIveHandle, IVE_DATA_S *pstSrc,
IVE_DST_DATA_S *pstDst, IVE_DMA_CTRL_S *pstDmaCtrl, HI_BOOL bInstant);
```

### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源数据指针。 不能为空。	输入(set 模式下同时也是输出)
pstDst	输出数据指针。 copy 模式下不能为空。	输出
pstDmaCtrl	DMA 控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

注：Copy 模式是指 IVE\_DMA\_MODE\_DIRECT\_COPY 和 IVE\_DMA\_MODE\_INTERVAL\_COPY 模式；

Set 模式是指 IVE\_DMA\_MODE\_SET\_3BYTE 和 IVE\_DMA\_MODE\_SET\_8BYTE 模式。

参数名称	支持类型	地址对齐	分辨率
pstSrc	IVE_DATA_S	1 byte	32x1~1920x1080
pstDst	IVE_DST_DATA_S	1 byte	直接拷贝时同 pstSrc； 间隔拷贝时比 pstSrc 小。

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

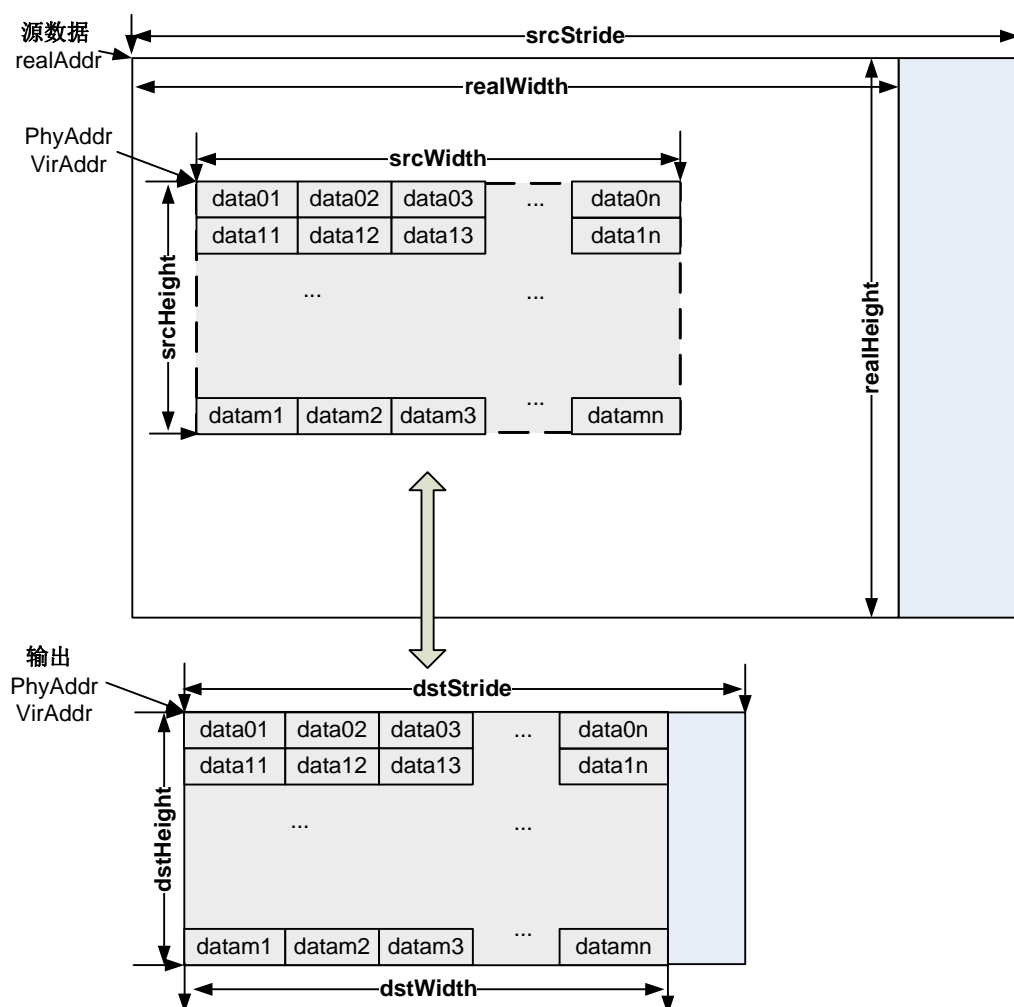
### 【注意】

- IVE\_DMA\_MODE\_DIRECT\_COPY：快速拷贝模式  
可实现从大块内存中扣取小块内存，如图 2-1 所示，计算公式如下：

$$I_{out}(x, y) = I(x, y) \quad (0 \leq x \leq width, 0 \leq y \leq height)$$

其中  $I(x, y)$  对应 pstSrc， $I_{out}(x, y)$  对应 pstDst。

图2-1 快速拷贝示意图



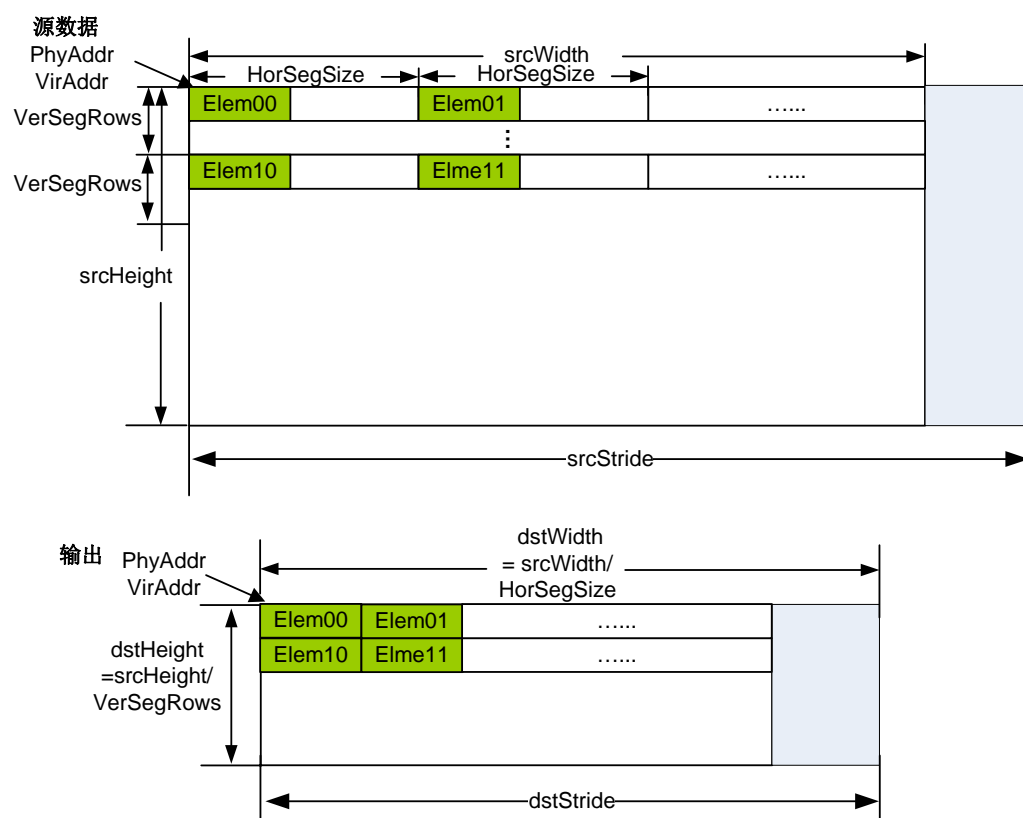
- IVE\_DMA\_MODE\_INTERVAL\_COPY：间隔拷贝模式
  - 要求源数据宽度为 u8HorSegSize 的倍数；
  - 间隔拷贝的方式：将每 u8VerSegRows 行中第一行数据分割为 u8HorSegSize 大小的段，拷贝每段中的前 u8ElemSize 大小的字节。如图 2-2 所示。
- IVE\_DMA\_MODE\_SET\_3BYTE：3 字节填充模式

仅使用 `pstSrc`，用 `u64Val` 的低 3 字节对源数据进行填充操作；当一行末尾不够 3 字节时，用 `u64Val` 的低字节填充。

- IVE\_DMA\_MODE\_SET\_8BYTE: 8 字节填充模式

仅使用 `pstSrc`，用 `u64Val` 对源数据进行填充操作；当一行的末尾不足 8 字节时，用 `u64Val` 的低字节填充。

图2-2 间隔拷贝示意图



【举例】

无。

【相关主题】

无。

## HI\_MPI\_IVE\_Filter

【描述】

创建 5x5 模板滤波任务，通过配置不同的模板系数，可以实现不同的滤波。

【语法】

```
HI_S32 HI_MPI_IVE_Filter(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,
IVE_DST_IMAGE_S *pstDst, IVE_FILTER_CTRL_S *pstFltCtrl, HI_BOOL bInstant);
```



### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstFltCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1、YUV420SP、YUV422SP	16 byte	64x64~1920x1024
pstDst	同 pstSrc	16 byte	同 pstSrc

注：U8C1\YUV420SP\YUV422SP 均为 [IVE\\_IMAGE\\_TYPE\\_E](#) 成员的简写，后续其他的成员在表述中也用相同的规则简写。

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】


























- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

### 【注意】


























- 当源数据为 YUV420SP、YUV422SP 类型时，要求输出数据跨度一致。
- Filter 计算公式示意如[图 2-3](#)所示。



图2-3 Filter 计算公式示意图

$I(x-2,y-2)$	$I(x-1,y-2)$	$I(x,y-2)$	$I(x+1,y-2)$	$I(x+2,y-2)$
				
$I(x-2,y-1)$	$I(x-1,y-1)$	$I(x,y-1)$	$I(x+1,y-1)$	$I(x+2,y-1)$
				
$I(x-2,y)$	$I(x-1,y)$	$I(x,y)$	$I(x+1,y)$	$I(x+2,y)$
				
$I(x-2,y+1)$	$I(x-1,y+1)$	$I(x,y+1)$	$I(x+1,y+1)$	$I(x+2,y+1)$
				
$I(x-2,y+2)$	$I(x-1,y+2)$	$I(x,y+2)$	$I(x+1,y+2)$	$I(x+2,y+2)$
				

$coef(-2,-2)$	$coef(-1,-2)$	$coef(0,-2)$	$coef(1,-2)$	$coef(2,-2)$
mask[0]	mask[1]	mask[2]	mask[3]	mask[4]
				
$coef(-2,-1)$	$coef(-1,-1)$	$coef(0,-1)$	$coef(1,-1)$	$coef(2,-1)$
mask[5]	mask[6]	mask[7]	mask[8]	mask[9]
				
$coef(-2,0)$	$coef(-1,0)$	$coef(0,0)$	$coef(1,0)$	$coef(2,0)$
mask[10]	mask[11]	mask[12]	mask[13]	mask[14]
				
$coef(-2,1)$	$coef(-1,1)$	$coef(0,1)$	$coef(1,1)$	$coef(2,1)$
mask[15]	mask[16]	mask[17]	mask[18]	mask[19]
				
$coef(-2,2)$	$coef(-1,2)$	$coef(0,2)$	$coef(1,2)$	$coef(2,2)$
mask[20]	mask[21]	mask[22]	mask[23]	mask[24]
				

$$I_{out}(x,y) = \left\{ \sum_{-2 \leq j \leq 2} \sum_{-2 \leq i \leq 2} I(x+i,y+j) \bullet coef(i,j) \right\} >> norm$$

其中， $I(x,y)$  对应 pstSrc， $I_{out}(x,y)$  对应 pstDst， $coef(mask)$  对应 pstFltCtrl 中的 as8Mask[25]，norm 对应 pstFltCtrl 中的 u8Norm。

- 经典高斯模板如下。

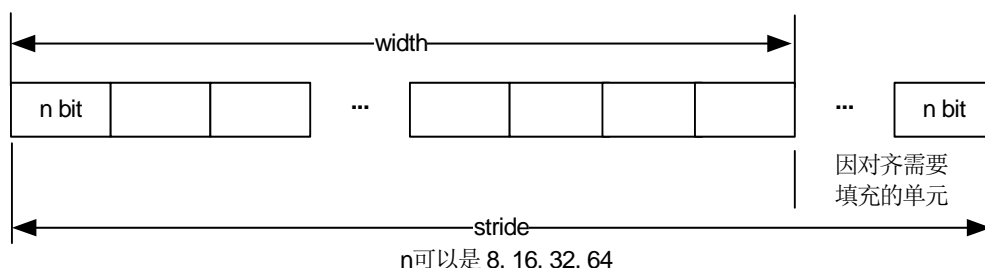
$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 5 & 6 & 5 & 2 \\ 3 & 6 & 8 & 6 & 3 \\ 2 & 5 & 6 & 5 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix} * 3$	$\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$
$u8Norm = 4$	$u8Norm = 8$	$u8Norm = 8$



- IVE\_DATA\_S 二维数据跨度，表示二维数据一行的字节数，即为图 1-1 中  $n=8$  的情况。

可以将 IVE\_DATA\_S 看成一个“像素”用 8bit 表示的图像，那么跨度即统一表述为图像或二维数据的一行以“像素”计算的单元个数。

图1-1 跨度 (stride) 示意图



- 对齐

硬件为了快速访问内存首地址或者跨行访问数据，要求内存地址或内存跨度必须为对齐系数的倍数。

- 数据内存首地址对齐

当前 IVE 算子对其输入输出有要求 1byte 对齐、2byte 对齐以及 16byte 对齐的，具体见各算子 API 参考中的参数要求。

- 跨度对齐

对于二维广义图像、二维单分量数据以及一维数组数据的跨度均必须满足 16 “像素”对齐。

- 输入、输出数据类型（具体结构定义请参见“3 数据类型”）

- 二维广义图像数据

IVE\_IMAGE\_S、IVE\_SRC\_IMAGE\_S、IVE\_DST\_IMAGE\_S，图像的类型参考 IVE\_IMAGE\_TYPE\_E，具体的内存分配如图 1-2~图 1-10 所示。

**注意：**当前所有算子输入输出的二维广义图像数据的高宽均需为偶数。

- 二维单分量数据

IVE\_DATA\_S，以 byte 为单位的二维数据，主要用于 DMA 等，其内存如图 1-11 所示；根据类型 IVE\_IMAGE\_S 可以转化为单个或多个 IVE\_DATA\_S。

- 一维数据

IVE\_MEM\_INFO\_S、IVE\_SRC\_MEM\_INFO\_S、IVE\_DST\_MEM\_INFO\_S，表示一维数据，如 Hist 的统计数据、GMM 的模型数据、LKOpticalFlow 的角点输入等；其内存如图 1-12 所示。

- 二维广义图像类型

类型	图像描述	内存地址	跨度
IVE_IMAGE_TYPE_U8C1	8bit 无符号单通道图像，如图 1-2 所示	仅用到 IVE_IMAGE_S 中的 u32PhyAddr[0]、pu8VirAddr[0]	仅用到 u16Stride[0]



【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_FilterAndCSC](#)
- [HI\\_MPI\\_IVE\\_OrdStatFilter](#)

## HI\_MPI\_IVE\_CSC

【描述】

创建色彩空间转换任务，可实现 YUV2RGB\YUV2HSV\YUV2LAB\RGB2YUV 的色彩空间转换。

【语法】

```
HI_S32 HI_MPI_IVE_CSC(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_DST_IMAGE_S *pstDst, IVE_CSC_CTRL_S *pstCscCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstCscCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	YUV420SP、YUV422SP、 U8C3_PLANAR、 U8C3_PACKAGE	16 byte	64x64~1920x1080
pstDst	U8C3_PLANAR、 U8C3_PACKAGE、 YUV420SP、YUV422SP	16 byte	同 pstSrc



#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 当输出数据为 U8C3\_PLANAR、YUV420SP、YUV422SP 类型时，要求输出数据跨度一致。
- 支持 12 种工作模式，不同的模式其输出的取值范围不一样，具体请参见[IVE\\_CSC\\_MODE\\_E](#)。
- YUV2HSV、YUV2LAB 参考 OpenCV 中的实现方法。



说明

本文档中所提到的 OpenCV，均指 OpenCV 2.4.8 版本。

#### 【举例】

无。

#### 【相关主题】

[HI\\_MPI\\_IVE\\_FilterAndCSC](#)

## HI\_MPI\_IVE\_FilterAndCSC

#### 【描述】

创建 5x5 模板滤波和 YUV2RGB 色彩空间转换复合任务，通过一次创建完成两种功能。

#### 【语法】

```
HI_S32 HI_MPI_IVE_FilterAndCSC(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstDst, IVE_FILTER_AND_CSC_CTRL_S
*pstFltCscCtrl, HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出



参数名称	描述	输入/输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstFltCscCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	YUV420SP、YUV422SP	16 byte	64x64~1920x1024
pstDst	U8C3_PLANAR、U8C3_PACKAGE	16 byte	同 pstSrc

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 当输出数据为 U8C3\_PLANAR 类型时，要求输出数据跨度一致。
- 仅支持 YUV2RGB 的 4 种工作模式，具体参见 [IVE\\_CSC\\_MODE\\_E](#)。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_Filter](#)



## HI\_MPI\_IVE\_Sobel

## 【描述】

创建 5x5 模板 Sobel-like 梯度计算任务。

## 【语法】

```
HI_S32 HI_MPI_IVE_Sobel(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,
IVE_DST_IMAGE_S *pstDstH, IVE_DST_IMAGE_S *pstDstV, IVE_SOBEL_CTRL_S
*pstSobelCtrl, HI_BOOL bInstant);
```

## 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDstH	由模板直接滤波得到的梯度分量图像 H 指针。 根据 pstSobelCtrl→enOutCtrl，若需要输出则不能为空。 高、宽同 pstSrc。	输出
pstDstV	由转置后的模板滤波得到的梯度分量图像 V 指针。根据 pstSobelCtrl→enOutCtrl，若需要输出则不能为空。 高、宽同 pstSrc。	输出
pstSobelCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1024
pstDstH	S16C1	16 byte	同 pstSrc
pstDstV	S16C1	16 byte	同 pstSrc

## 【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，参见 <a href="#">错误码</a> 。

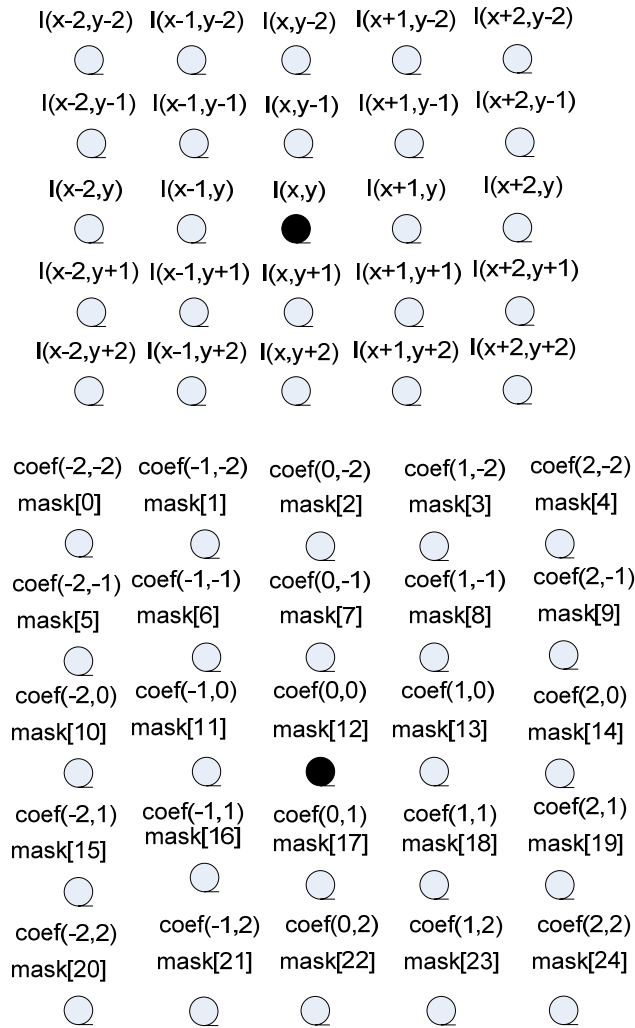
**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

- 可配置 3 种输出模式，参考 [IVE\\_SOBEL\\_OUT\\_CTRL\\_E](#)。
- 当输出模式为 IVE\_SOBEL\_OUT\_CTRL\_BOTH 时，要求 pstDstH 和 pstDstV 跨度一致。
- Sobel 计算公式示意如[图 2-4](#)所示。

图2-4 Sobel 计算公式示意图



$$Hout(x, y) = \sum_{-2 < j < 2} \sum_{-2 < i < 2} I(x+i, y+j) \bullet coef(i, j)$$

$$Vout(x, y) = \sum_{-2 < j < 2} \sum_{-2 < i < 2} I(x+i, y+j) \bullet coef(j, i)$$

其中， $I(x, y)$  对应 `pstSrc`， $Hout(x, y)$  对应 `pstDstH`， $Vout(x, y)$  对应 `pstDstV`， $coef$  (`mask`) 为 `pstSobelCtrl` 中的 `as8Mask[25]`

- Sobel 模板





$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -2 & 0 & 2 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

- Scharr 模板

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 3 & 0 \\ 0 & -10 & 0 & 10 & 0 \\ 0 & -3 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & -10 & -3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 10 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- 拉普拉斯模板

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -8 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & -1 & 8 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_MagAndAng](#)
- [HI\\_MPI\\_IVE\\_NormGrad](#)

## HI\_MPI\_IVE\_MagAndAng

【描述】



创建 5x5 模板梯度幅值与幅角计算任务。

【语法】

```
HI_S32 HI_MPI_IVE_MagAndAng( IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstDstMag, IVE_DST_IMAGE_S *pstDstAng,
IVE_MAG_AND_ANG_CTRL_S *pstMagAndAngCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDstMag	输出幅值图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstDstAng	输出幅角图像指针。 根据 pstMagAndAngCtrl→enOutCtrl，需要输出 则不能为空。 高、宽同 pstSrc。	输出
pstMagAndAngCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1024
pstDstMag	U16C1	16 byte	同 pstSrc
pstDstAng	U8C1	16 byte	同 pstSrc

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。



【需求】

- 头文件: hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件: libive.a (PC 上模拟用 ive\_clib2.x.lib)

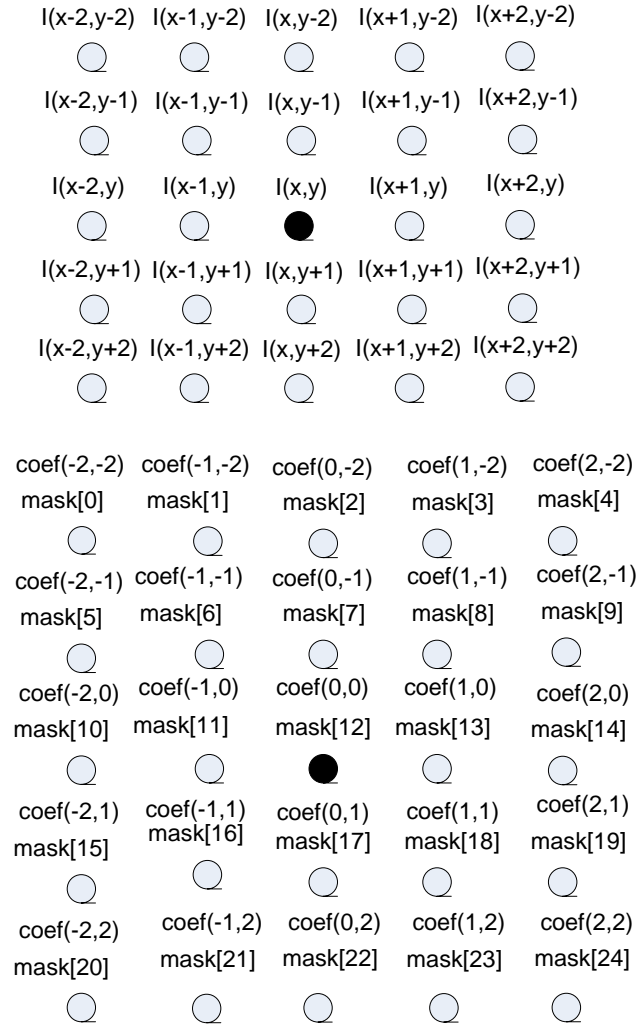
【注意】

- 可配置 2 种输出模式, 具体参见 [IVE\\_MAG\\_AND\\_ANG\\_OUT\\_CTRL\\_E](#)。
- 当输出模式为 IVE\_MAG\_AND\_ANG\_OUT\_CTRL\_MAG\_AND\_ANG 时, 要求 pstDstMag 和 pstDstAng 跨度一致。
- 用户可以通过 pstMagAndAngCtrl→u16Thr 对幅值图进行 thresh 操作(可以用来实现 EOH), 计算公式如下:

$$Mag(x, y) = \begin{cases} 0 & (Mag(x, y) < u16Thr) \\ Mag(x, y) & (Mag(x, y) \geq u16Thr) \end{cases}$$

其中,  $Mag(x, y)$  对应 pstDstMag。

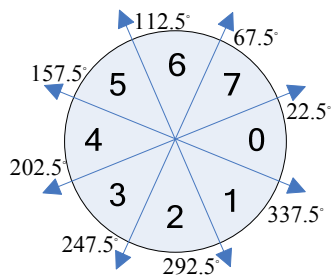
图2-5 MagAndAng 计算示意图



$$H_{out}(x, y) = \sum_{-2 < j < 2} \sum_{-2 < i < 2} I(x+i, y+j) \bullet coef(i, j)$$

$$V_{out}(x, y) = \sum_{-2 < j < 2} \sum_{-2 < i < 2} I(x+i, y+j) \bullet coef(j, i)$$

$$Mag(x, y) = abs(H_{out}(x, y)) + abs(V_{out}(x, y))$$





$\theta(x, y)$  根据  $H_{out}(x, y)$ 、 $V_{out}(x, y)$  以及  $\arctan(\frac{V_{out}}{H_{out}})$  取对应上图中 0~7 的方向值。

其中,  $I(x, y)$  对应 pstSrc,  $Mag(x, y)$  对应 pstDstMag,  $\theta(x, y)$  对应 pstDstAng,  $coef(mask)$  为 pstMagAndAngCtrl 中的 as8Mask[25]。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_CannyHysEdge](#)
- [HI\\_MPI\\_IVE\\_CannyEdge](#)
- [HI\\_MPI\\_IVE\\_Sobel](#)

## HI\_MPI\_IVE\_Dilate

#### 【描述】

创建二值图像 5x5 模板膨胀任务。

#### 【语法】

```
HI_S32 HI_MPI_IVE_Dilate(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,
IVE_DST_IMAGE_S *pstDst, IVE_DILATE_CTRL_S *pstDilateCtrl, HI_BOOL
bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstDilateCtrl	控制信息指针。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1 的二值图	16 byte	64x64~1920x1024
pstDst	U8C1 的二值图	16 byte	同 pstSrc



【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

【需求】

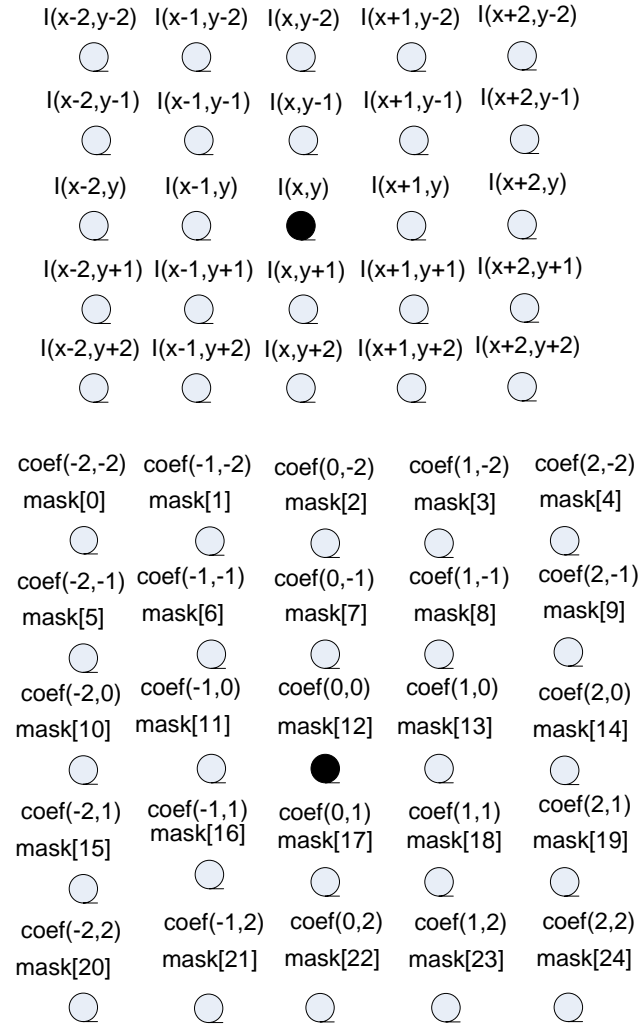
- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

【注意】

- 模板系数只能为 0 或 255。
- 模板样例

$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 255 & & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 255 & & 0 \\ 0 & 0 & 255 & 0 & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 255 & 0 & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & 255 & 255 & 255 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 0 \end{bmatrix}$	$\begin{bmatrix} 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \end{bmatrix}$

图2-6 Dilate 计算公式示意图



$$I_{out}(x,y) = f(I(x+(k \& 5)-1, y+(k \% 5)-1) \& coef((k \& 5)-1, (k \% 5)-1), |, 0, 24)$$

$$f(A_k, \Theta, c_{min}, c_{max}) = A_{c_{min}} \Theta A_{c_{min+1}} \cdots \Theta A_{c_{max}}$$

其中，公式中 $|$ 为位或运算， $\&$ 为位与运算， $\%$ 为取余运算。 $I(x,y)$ 对应 `pstSrc`， $I_{out}(x,y)$ 对应 `pstDst`， $coef(mask)$ 对应 `pstDilateCtrl` 中的 `au8Mask[25]`。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_Erode](#)
- [HI\\_MPI\\_IVE\\_OrdStatFilter](#)



## HI\_MPI\_IVE\_Erode

### 【描述】

创建二值图像 5x5 模板腐蚀任务。

### 【语法】

```
HI_S32 HI_MPI_IVE_Erode(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_DST_IMAGE_S *pstDst, IVE_ERODE_CTRL_S *pstErodeCtrl, HI_BOOL bInstant);
```

### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstErodeCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1 的二值图	16 byte	64x64~1920x1024
pstDst	U8C1 的二值图	16 byte	同 pstSrc

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）



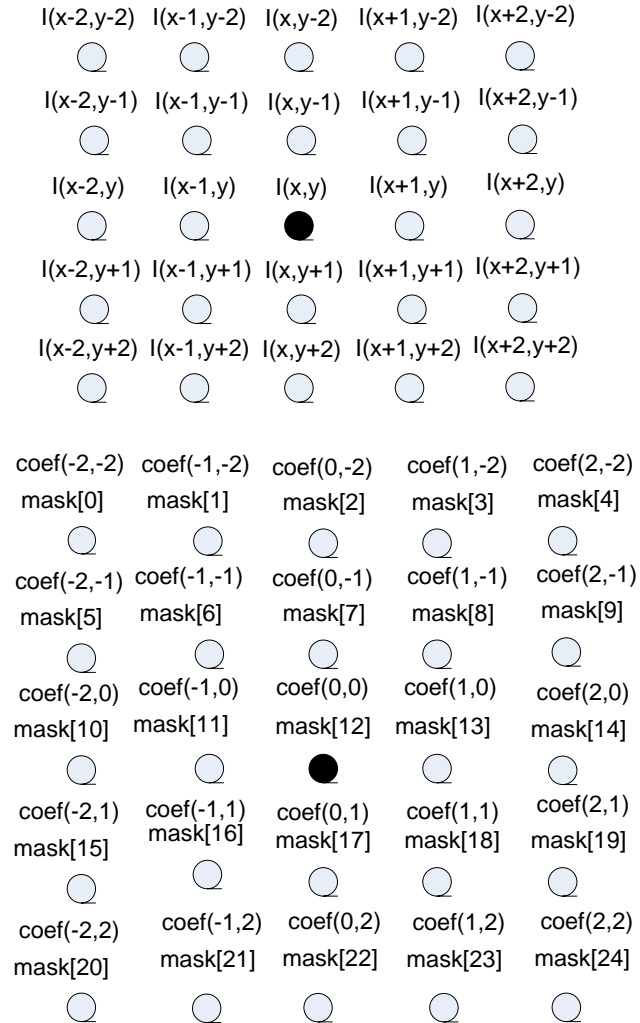


【注意】

- 模板系数只能为 0 或 255。
- 模板样例

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 255 & & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 255 & & 0 \\ 0 & 0 & 255 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 255 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 255 & 255 & 255 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 0 \end{bmatrix} \quad \begin{bmatrix} 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 \end{bmatrix}$$

图2-7 Erode 计算公式示意图



$$I_{out}(x, y) = f(I(x + (k \& 5) - 1, y + (k \% 5) - 1) | \text{coef}((k \& 5) - 1, (k \% 5) - 1), \& 0, 24)$$

$$f(A_k, \Theta, c_{\min}, c_{\max}) = A_{c_{\min}} \Theta A_{c_{\min+1}} \cdots \Theta A_{c_{\max}}$$

其中，公式中 $|$ 为位或运算， $\&$ 为位与运算， $\%$ 为取余运算。 $I(x, y)$ 对应  $\text{pstSrc}$ ， $I_{out}(x, y)$ 对应  $\text{pstDst}$ ， $\text{coef}(\text{mask})$ 对应  $\text{pstErodeCtrl}$  中的  $\text{au8Mask}[25]$ 。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_Dilate](#)
- [HI\\_MPI\\_IVE\\_OrdStatFilter](#)



## HI\_MPI\_IVE\_Thresh

### 【描述】

创建灰度图像阈值化任务。

### 【语法】

```
HI_S32 HI_MPI_IVE_Thresh(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_DST_IMAGE_S *pstDst, IVE_THRESH_CTRL_S *pstThrCtrl, HI_BOOL bInstant);
```

### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstThrCtrl	控制信息指针。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	1 byte	64x64~1920x1080
pstDst	U8C1	1 byte	同 pstSrc

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

- 可以配置 8 种运算模式，具体参见 [IVE\\_THRESH\\_MODE\\_E](#)。
- 计算公式

- IVE\_THRESH\_MODE\_BINARY:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ maxVal & (I(x, y) > lowThr) \end{cases}$$

*midVal*、*highThr* 无需赋值。

- IVE\_THRESH\_MODE\_TRUNC:

$$I_{out}(x, y) = \begin{cases} I(x, y) & (I(x, y) \leq lowThr) \\ maxVal & (I(x, y) > lowThr) \end{cases}$$

*minVal*、*midVal*、*highThr* 无需赋值。

- IVE\_THRESH\_MODE\_TO\_MINVAL:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ I(x, y) & (I(x, y) > lowThr) \end{cases}$$

*midVal*、*maxVal*、*highThr* 无需赋值。

- IVE\_THRESH\_MODE\_MIN\_MID\_MAX:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ midVal & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

- IVE\_THRESH\_MODE\_ORI\_MID\_MAX:

$$I_{out}(x, y) = \begin{cases} I(x, y) & (I(x, y) \leq lowThr) \\ midVal & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

*minVal* 无需赋值。

- IVE\_THRESH\_MODE\_MIN\_MID\_ORI:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ midVal & (lowThr < I(x, y) \leq highThr) \\ I(x, y) & (I(x, y) > highThr) \end{cases}$$

*maxVal* 无需赋值。

- IVE\_THRESH\_MODE\_MIN\_ORI\_MAX:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ I(x, y) & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

*midVal* 无需赋值。

- IVE\_THRESH\_MODE\_ORI\_MID\_ORI:

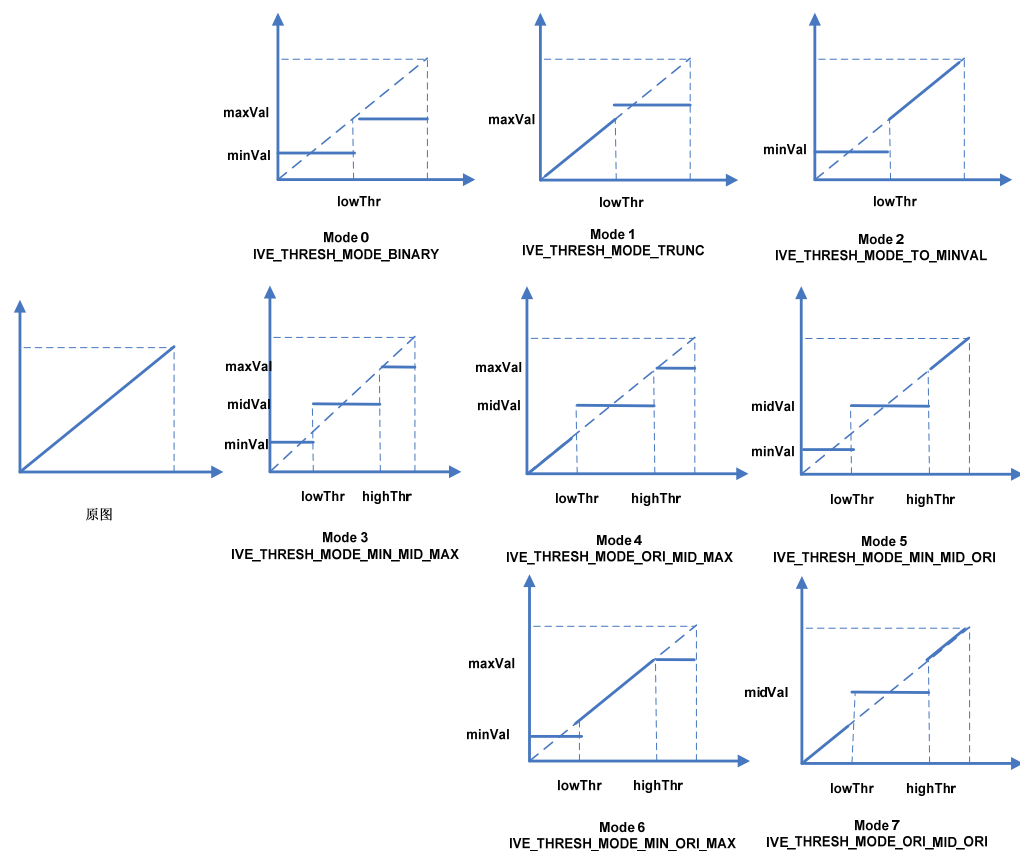
$$I_{out}(x, y) = \begin{cases} I(x, y) & (I(x, y) \leq lowThr) \\ midVal & (lowThr < I(x, y) \leq highThr) \\ I(x, y) & (I(x, y) > highThr) \end{cases}$$

$minVal$ 、 $maxVal$  无需赋值。

其中， $I(x, y)$  对应  $pstSrc$ ， $I_{out}(x, y)$  对应  $pstDst$ ， $mode$ 、 $lowThr$ 、 $highThr$ 、 $minVal$ 、 $midVal$  和  $maxVal$  分别对应  $pstThrCtrl$  的  $enMode$ 、 $u8LowThr$ 、 $u8HighThr$ 、 $u8MinVal$ 、 $u8MidVal$  和  $u8MaxVal$ 。具体示意图如图 2-8 所示。

- $pstThrCtrl$  中的  $u8MinVal$ 、 $u8MidVal$  和  $u8MaxVal$  并不需要满足变量命名含义中的大小关系。

图2-8 Thresh 8 种阈值化模式示意图



#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_Thresh\\_S16](#)
- [HI\\_MPI\\_IVE\\_Thresh\\_U16](#)



## HI\_MPI\_IVE\_And

## 【描述】

创建两二值图像相与任务。

## 【语法】

```
HI_S32 HI_MPI_IVE_And(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc1,
IVE_SRC_IMAGE_S *pstSrc2, IVE_DST_IMAGE_S *pstDst, HI_BOOL bInstant);
```

## 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc1	源图像 1 指针。 不能为空。	输入
pstSrc2	源图像 2 指针。 不能为空。 高、宽同 pstSrc1。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc1。	输出
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc1	U8C1 的二值图	1 byte	64x64~1920x1080
pstSrc2	U8C1 的二值图	1 byte	同 pstSrc1
pstDst	U8C1 的二值图	1 byte	同 pstSrc1

## 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

## 【需求】



- 头文件: hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件: libive.a (PC 上模拟用 ive\_clib2.x.lib)

【注意】

计算公式如下:

$$I_{out}(x, y) = I_{src1}(x, y) \& I_{src2}(x, y)$$

其中,  $I_{src1}(x, y)$  对应 pstSrc1,  $I_{src2}(x, y)$  对应 pstSrc2,  $I_{out}(x, y)$  对应 pstDst

【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_Or](#)
- [HI\\_MPI\\_IVE\\_Xor](#)

## HI\_MPI\_IVE\_Sub

【描述】

创建两灰度图像相减任务。

【语法】

```
HI_S32 HI_MPI_IVE_Sub(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc1,  
IVE_SRC_IMAGE_S *pstSrc2, IVE_DST_IMAGE_S *pstDst, IVE_SUB_CTRL_S  
*pstSubCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。	输出
pstSrc1	源图像 1 指针。 不能为空。	输入
pstSrc2	源图像 2 指针。 不能为空。 高、宽同 pstSrc1。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc1。	输出
pstSubCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入



参数名称	支持图像类型	地址对齐	分辨率
pstSrc1	U8C1	1 byte	64x64~1920x1080
pstSrc2	U8C1	1 byte	同 pstSrc1
pstDst	U8C1、S8C1	1 byte	同 pstSrc1

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 可以配置 2 种输出格式，具体参见 [IVE\\_SUB\\_MODE\\_E](#)。
- IVE\_SUB\_MODE\_ABS
  - 计算公式： $I_{out}(x, y) = abs(I_{src1}(x, y) - I_{src2}(x, y))$
  - 输出格式：U8C1
- IVE\_SUB\_MODE\_SHIFT
  - 计算公式： $I_{out}(x, y) = (I_{src1}(x, y) - I_{src2}(x, y)) >> 1$
  - 输出格式：S8C1

其中， $I_{src1}(x, y)$  对应 pstSrc1， $I_{src2}(x, y)$  对应 pstSrc2， $I_{out}(x, y)$  对应 pstDst。

#### 【举例】

无。

#### 【相关主题】

[HI\\_MPI\\_IVE\\_Add](#)

## HI\_MPI\_IVE\_Or

#### 【描述】

创建两二值图像相或任务。

#### 【语法】





```
HI_S32 HI_MPI_IVE_Or(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc1,
IVE_SRC_IMAGE_S *pstSrc2, IVE_DST_IMAGE_S *pstDst, HI_BOOL bInstant);
```

**【参数】**

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc1	源图像 1 指针。 不能为空。	输入
pstSrc2	源图像 2 指针。 不能为空。 高、宽同 pstSrc1。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc1。	输出
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc1	U8C1	1 byte	64x64~1920x1080
pstSrc2	U8C1	1 byte	同 pstSrc1
pstDst	U8C1	1 byte	同 pstSrc1

**【返回值】**

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

计算公式如下：



$$I_{out}(x, y) = I_{src1}(x, y) | I_{src2}(x, y)$$

其中,  $I_{src1}(x, y)$  对应 pstSrc1,  $I_{src2}(x, y)$  对应 pstSrc2,  $I_{out}(x, y)$  对应 pstDst。

【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_And](#)
- [HI\\_MPI\\_IVE\\_Xor](#)

## HI\_MPI\_IVE\_Integ

【描述】

创建灰度图像的积分图计算任务。

【语法】

```
HI_S32 HI_MPI_IVE_Integ(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_DST_IMAGE_S *pstDst, IVE_INTEG_CTRL_S *pstIntegCtrl, HI_BOOL  
bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstIntegCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	32x16~1920x1080
pstDst	U32C1、U64C1	16 byte	同 pstSrc



## 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

## 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

## 【注意】

- IVE\_INTEG\_OUT\_CTRL\_COMBINE，组合输出模式，输出图像类型必须为 IVE\_IMAGE\_TYPE\_U64C1，参见[图 1-13](#)，计算公式如下：

$$I_{sum}(x, y) = \sum_{i \geq 0} \sum_{j \geq 0}^{i \leq x, j \leq y} I(i, j)$$

$$I_{sq}(x, y) = \sum_{i \geq 0} \sum_{j \geq 0}^{i \leq x, j \leq y} (I(i, j) \cdot I(i, j))$$

$$I_{out}(x, y) = (I_{sq}(x, y) \ll 28) | (I_{sum}(x, y) \& 0xFFFFFFFF)$$

- IVE\_INTEG\_OUT\_CTRL\_SUM，仅和积分图输出模式，输出图像类型必须为 IVE\_IMAGE\_TYPE\_U32C1，计算公式如下：

$$I_{sum}(x, y) = \sum_{i \geq 0} \sum_{j \geq 0}^{i \leq x, j \leq y} I(i, j)$$

$$I_{out}(x, y) = I_{sum}(x, y)$$

- IVE\_INTEG\_OUT\_CTRL\_SQSUM，仅平方和积分图输出，输出图像类型必须为 IVE\_IMAGE\_TYPE\_U64C1，计算公式如下：

$$I_{sq}(x, y) = \sum_{i \geq 0} \sum_{j \geq 0}^{i \leq x, j \leq y} (I(i, j) \cdot I(i, j))$$

$$I_{out}(x, y) = I_{sq}(x, y)$$

其中， $I(x, y)$  对应 pstSrc， $I_{out}(x, y)$  对应 pstDst。

## 【举例】

无。

## 【相关主题】

无。

## HI\_MPI\_IVE\_Hist

## 【描述】



创建灰度图像的直方图统计任务。

### 【语法】

```
HI_S32 HI_MPI_IVE_Hist(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,
IVE_DST_MEM_INFO_S *pstDst, HI_BOOL bInstant);
```

### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出数据指针。 不能为空。 内存至少配置 1024 字节，如 <a href="#">图 1-14</a> ；	输出
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1080
pstDst	-	16 byte	-

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

### 【注意】

计算公式如下：

$$I_{out}(x) = \sum_i \sum_j ((I(i, j) == x) ? 1 : 0) \quad x = 0 \dots 255$$



其中,  $I(i, j)$  对应 pstSrc,  $I_{out}(x)$  对应 pstDst。

**【举例】**

无。

**【相关主题】**

无。

## HI\_MPI\_IVE\_Thresh\_S16

**【描述】**

创建 S16 数据到 8bit 数据的阈值化任务。

**【语法】**

```
HI_S32 HI_MPI_IVE_Thresh_S16(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstDst, IVE_THRESH_S16_CTRL_S *pstThrS16Ctrl,
HI_BOOL bInstant);
```

**【参数】**

参数名称	描述	输入/输出
pIveHandle	handle 指针。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstThrS16Ctrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	S16C1	2 byte	64x64~1920x1080
pstDst	U8C1、S8C1	1 byte	同 pstSrc

**【返回值】**



返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

- 可配置 4 种运算模式，参考 [IVE\\_THRESH\\_S16\\_MODE\\_S16\\_TO\\_S8\\_MIN\\_MID\\_MAX](#)。
- 计算公式

- IVE\_THRESH\_S16\_MODE\_S16\_TO\_S8\_MIN\_MID\_MAX:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ midVal & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

要求：-32768 ≤ lowThr ≤ highThr ≤ 32767;

-128 ≤ minVal、midVal、maxVal ≤ 127。

- IVE\_THRESH\_S16\_MODE\_S16\_TO\_S8\_MIN\_ORI\_MAX:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ I(x, y) & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

要求：-129 ≤ lowThr ≤ highThr ≤ 127;

-128 ≤ minVal、maxVal ≤ 127;

- IVE\_THRESH\_S16\_MODE\_S16\_TO\_U8\_MIN\_MID\_MAX:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ midval & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

要求：-32768 ≤ lowThr ≤ highThr ≤ 32767;

0 ≤ minVal、midVal、maxVal ≤ 255。

- IVE\_THRESH\_S16\_MODE\_S16\_TO\_U8\_MIN\_ORI\_MAX:

$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ I(x, y) & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

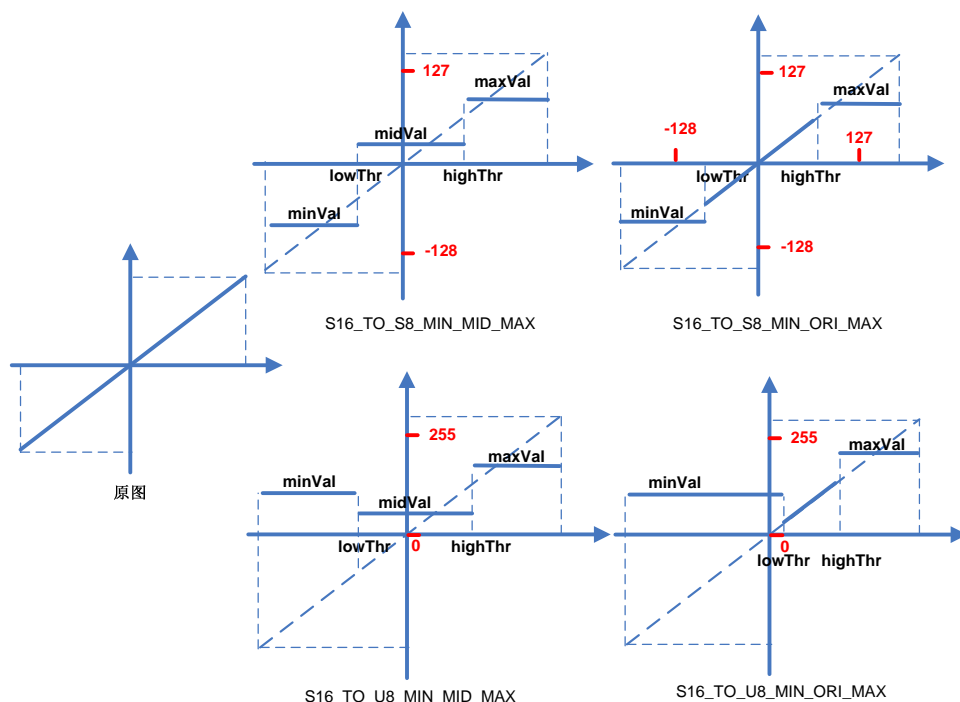
要求：-1 ≤ lowThr ≤ highThr ≤ 255;

$$0 \leq \minVal, \maxVal \leq 255.$$

其中,  $I(x, y)$  对应  $pstSrc$ ,  $I_{out}(x, y)$  对应  $pstDst$ ,  $mode$ 、 $lowThr$ 、 $highThr$ 、 $\minVal$ 、 $\midVal$  和  $\maxVal$  分别对应  $pstThrS16Ctrl$  的  $enMode$ 、 $s16LowThr$ 、 $s16HighThr$ 、 $un8MinVal$ 、 $un8MidVal$  和  $un8MaxVal$ 。具体示意图如图 2-9 所示。

- $pstThrS16Ctrl$  中的  $un8MinVal$ 、 $un8MidVal$  和  $un8MaxVal$  并不需要满足变量命名含义中的大小关系。

图2-9 Thresh\_S16 4 种阈值化模式示意图



#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_Thresh\\_U16](#)
- [HI\\_MPI\\_IVE\\_16BitTo8Bit](#)

## HI\_MPI\_IVE\_Thresh\_U16

#### 【描述】

创建 U16 数据到 U8 数据的阈值化任务。

#### 【语法】

```
HI_S32 HI_MPI_IVE_Thresh_U16(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
```



```
*pstSrc, IVE_DST_IMAGE_S *pstDst, IVE_THRESH_U16_CTRL_S *pstThrU16Ctrl,  
HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstThrU16Ctrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U16C1	2 byte	64x64~1920x1080
pstDst	U8C1	1 byte	同 pstSrc

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 可配置 2 种运算模式，参考 [IVE\\_THRESH\\_U16\\_MODE\\_E](#)。
- 计算公式
  - IVE\_THRESH\_U16\_MODE\_U16\_TO\_U8\_MIN\_MID\_MAX:



$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ midVal & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

要求:  $0 \leq lowThr \leq highThr \leq 65535$ ;

- IVE\_THRESH\_U16\_MODE\_U16\_TO\_U8\_MIN\_ORI\_MAX:

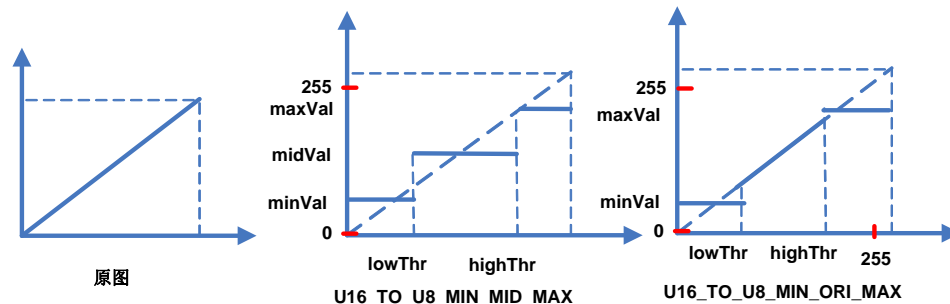
$$I_{out}(x, y) = \begin{cases} minVal & (I(x, y) \leq lowThr) \\ I(x, y) & (lowThr < I(x, y) \leq highThr) \\ maxVal & (I(x, y) > highThr) \end{cases}$$

要求:  $0 \leq lowThr \leq highThr \leq 255$ ;

其中,  $I(x, y)$  对应 pstSrc,  $I_{out}(x, y)$  对应 pstDst, mode、lowThr、highThr、minVal、midVal 和 maxVal 分别对应 pstThrU16Ctrl 的 enMode、u16LowThr、u16HighThr、u8MinVal、u8MidVal 和 u8MaxVal。具体示意图如图 2-10 所示。

- pstThrU16Ctrl 中的 u8MinVal、u8MidVal 和 u8MaxVal 并不需要满足变量命名含义中的大小关系。

图2-10 Thresh\_U16 2 种阈值化模式示意图



#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_Thresh\\_S16](#)
- [HI\\_MPI\\_IVE\\_16BitTo8Bit](#)

## HI\_MPI\_IVE\_16BitTo8Bit

#### 【描述】

创建 16bit 图像数据到 8bit 图像数据的线性转化任务。

#### 【语法】



```
HI_S32 HI_MPI_IVE_16BitTo8Bit(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstDst, IVE_16BIT_TO_8BIT_CTRL_S
*pst16BitTo8BitCtrl, HI_BOOL bInstant);
```

**【参数】**

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pst16BitTo8BitCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U16C1、S16C1	2 byte	64x64~1920x1080
pstDst	U8C1、S8C1	1 byte	同 pstSrc

**【返回值】**

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

- 可配置 4 种模式，具体参考 [IVE\\_16BIT\\_TO\\_8BIT\\_MODE\\_E](#)。
- 计算公式



- IVE\_16BIT\_TO\_8BIT\_MODE\_S16\_TO\_S8:

$$I_{out}(x, y) = \begin{cases} -128 & (\frac{a}{b}I(x, y) < -128) \\ \frac{a}{b}I(x, y) & (-128 \leq \frac{a}{b}I(x, y) \leq 127) \\ 127 & (\frac{a}{b}I(x, y) > 127) \end{cases}$$

- IVE\_16BIT\_TO\_8BIT\_MODE\_S16\_TO\_U8\_ABS:

$$I_{out}(x, y) = \begin{cases} \left| \frac{a}{b}I(x, y) \right| & (\left| \frac{a}{b}I(x, y) \right| \leq 255) \\ 255 & (\left| \frac{a}{b}I(x, y) \right| > 255) \end{cases}$$

- IVE\_16BIT\_TO\_8BIT\_MODE\_S16\_TO\_U8\_BIAS:

$$I_{out}(x, y) = \begin{cases} 0 & (\frac{a}{b}I(x, y) + bias < 0) \\ \frac{a}{b}I(x, y) + bias & (0 \leq \frac{a}{b}I(x, y) + bias \leq 255) \\ 255 & (\frac{a}{b}I(x, y) + bias > 255) \end{cases}$$

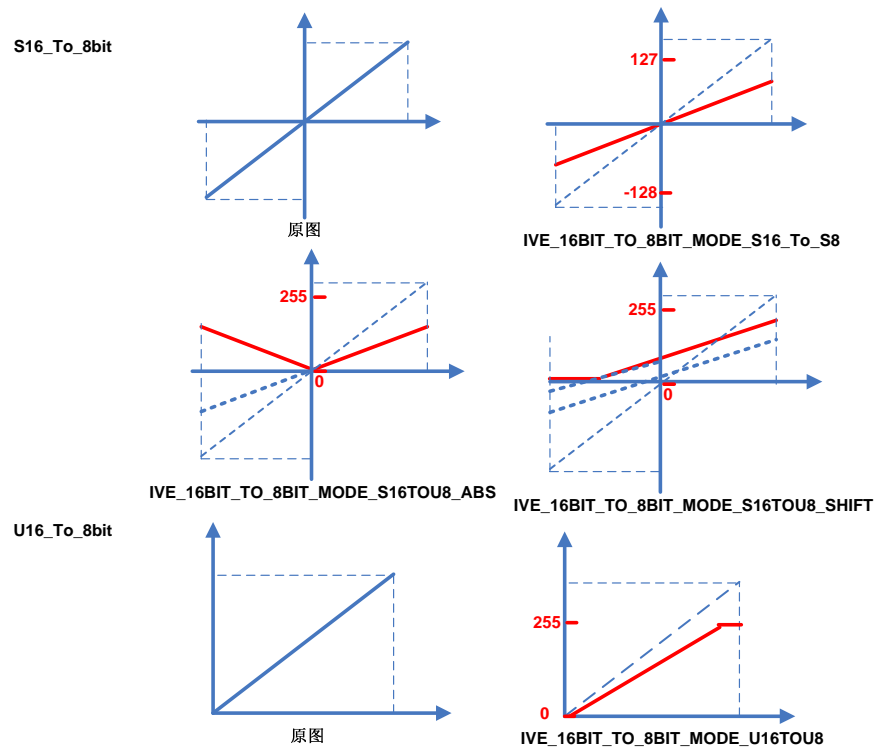
- IVE\_16BIT\_TO\_8BIT\_MODE\_U16\_TO\_U8:

$$I_{out}(x, y) = \begin{cases} 0 & (\frac{a}{b}I(x, y) < 0) \\ \frac{a}{b}I(x, y) & (0 \leq \frac{a}{b}I(x, y) \leq 255) \\ 255 & (\frac{a}{b}I(x, y) > 255) \end{cases}$$

其中,  $I(x, y)$  对应 `pstSrc`,  $I_{out}(x, y)$  对应 `pstDst`, `mode`、 $a$ 、 $b$  和  $bias$  分别对应 `pst16BitTo8BitCtrl` 的 `enMode`、`u8Numerator`、`u16Denominator`、`s8Bias`。具体示意图如图 2-11 所示。

要求: `u8Numerator`  $\leq$  `u16Denominator`, 且 `u16Denominator`  $\neq 0$ 。

图2-11 16BitTo8Bit 4 种转换模式示意图



【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_Thresh\\_S16](#)
- [HI\\_MPI\\_IVE\\_Thresh\\_U16](#)

## HI\_MPI\_IVE\_OrdStatFilter

【描述】

创建 3x3 模板顺序统计量滤波任务，可进行 Median、Max、Min 滤波。

【语法】

```
HI_S32 HI_MPI_IVE_OrdStatFilter(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstDst, IVE_ORD_STAT_FILTER_CTRL_S
*pstOrdStatFltCtrl, HI_BOOL bInstant);
```

【参数】



参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstOrdStatFltCtrl	控制参数指针 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1024
pstDst	U8C1	16 byte	同 pstSrc

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 可配置 3 种滤波模式，参考 [IVE\\_ORD\\_STAT\\_FILTER\\_MODE\\_E](#)。

- 计算公式

– IVE\_ORD\_STAT\_FILTER\_MODE\_MEDIAN:

$$I_{out}(x, y) = \underset{\substack{-1 \leq i \leq 1 \\ -1 \leq j \leq 1}}{\text{median}} \{I(x+i, y+j)\}$$

– IVE\_ORD\_STAT\_FILTER\_MODE\_MAX:



$$I_{out}(x, y) = \max_{\substack{-1 \leq i \leq 1 \\ -1 \leq j \leq 1}} \{I(x+i, y+j)\}$$

- IVE\_ORD\_STAT\_FILTER\_MODE\_MIN:

$$I_{out}(x, y) = \min_{\substack{-1 \leq i \leq 1 \\ -1 \leq j \leq 1}} \{I(x+i, y+j)\}$$

其中,  $I(x, y)$  对应 pstSrc,  $I_{out}(x, y)$  对应 pstDst。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_Filter](#)
- [HI\\_MPI\\_IVE\\_Dilate](#)
- [HI\\_MPI\\_IVE\\_Erode](#)

## HI\_MPI\_IVE\_Map

#### 【描述】

创建 Map（映射赋值）任务，对源图像中的每个像素，查找 Map 查找表中的值，赋予目标图像相应像素查找表中的值，支持 U8C1→U8C1、U8C1→U16C1、U8C1→S16C1 3 种模式的映射。

#### 【语法】

```
HI_S32 HI_MPI_IVE_Map(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_SRC_MEM_INFO_S *pstMap, IVE_DST_IMAGE_S *pstDst, IVE_MAP_CTRL_S  
*pstMapCtrl, HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstMap	映射表信息指针。 不能为空。 内存至少配置：sizeof(IVE_MAP_LUT_MEM_S)。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
bInstant	及时返回结果标志。	输入



参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	1 byte	64x64~1920x1080
pstMap	-	16 byte	-
pstDst	U8C1	1 byte	同 pstSrc

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持
Hi3536V100	支持
Hi3521AV100	支持
Hi3518EV200	支持
Hi3531AV100	支持
Hi3519V100	不支持

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.0.lib）

#### 【注意】

计算公式如下：

$$I_{out}(x, y) = map[I(x, y)]$$

其中， $I(x, y)$  对应 pstSrc， $I_{out}(x, y)$  对应 pstDst， $map$  对应 pstMap。

#### 【举例】

无。



【相关主题】

无。

## HI\_MPI\_IVE\_Map

【描述】

创建 Map（映射赋值）任务，对源图像中的每个像素，查找 Map 查找表中的值，赋予目标图像相应像素查找表中的值，支持 U8C1→U8C1、U8C1→U16C1、U8C1→S16C1 3 种模式的映射。

【语法】

```
HI_S32 HI_MPI_IVE_Map(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_SRC_MEM_INFO_S *pstMap, IVE_DST_IMAGE_S *pstDst, IVE_MAP_CTRL_S  
*pstMapCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstMap	映射表信息指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstMapCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16byte	64x64~1920x1080
pstMap	-	16 byte	-
pstDst	U8C1、U16C1、S16C1	16 byte	同 pstSrc

【返回值】





返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	不支持
Hi3536V100	不支持
Hi3521AV100	不支持
Hi3518EV200	不支持
Hi3531AV100	不支持
Hi3519V100	支持

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

#### 【注意】

- 计算公式如下：

$$I_{out}(x, y) = map[I(x, y)]$$

其中， $I(x, y)$  对应 pstSrc， $I_{out}(x, y)$  对应 pstDst， $map$  对应 pstMap。

- pstMap 的内存配置根据 pstMapCtrl→enMode 配置不同：
  - IVE\_MAP\_MODE\_U8，配置 sizeof(IVE\_MAP\_U8BIT\_LUT\_MEM\_S);
  - IVE\_MAP\_MODE\_U16，配置 sizeof(IVE\_MAP\_U16BIT\_LUT\_MEM\_S);
  - IVE\_MAP\_MODE\_S16，配置 sizeof(IVE\_MAP\_S16BIT\_LUT\_MEM\_S).

#### 【举例】

无。

#### 【相关主题】

无。

## HI\_MPI\_IVE\_EqualizeHist

#### 【描述】



创建灰度图像的直方图均衡化计算任务。

#### 【语法】

```
HI_S32 HI_MPI_IVE_EqualizeHist(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S  
*pstSrc, IVE_DST_IMAGE_S *pstDst, IVE_EQUALIZE_HIST_CTRL_S  
*pstEqualizeHistCtrl, HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstEqualizeHistCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1080
pstDst	U8C1	16 byte	同 pstSrc
pstEqualizeHistCtrl→ stMem	-	16 byte	-

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h



- 库文件: libive.a (PC 上模拟用 ive\_clib2.x.lib)

【注意】

- pstEqualizeHistCtrl 中的 stMem, 至少需开辟 sizeof(IVE\_EQUALIZE\_HIST\_CTRL\_MEM\_S)字节大小。
- 与 OpenCV 中直方图均衡化计算过程一致。

【举例】

无。

【相关主题】

无。

## HI\_MPI\_IVE\_Add

【描述】

创建两灰度图像的加权加计算任务。

【语法】

```
HI_S32 HI_MPI_IVE_Add(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S  
*pstSrc1, IVE_SRC_IMAGE_S *pstSrc2, IVE_DST_IMAGE_S *pstDst,  
IVE_ADD_CTRL_S *pstAddCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc1	源图像 1 指针。 不能为空。	输入
pstSrc2	源图像 2 指针。 不能为空。 高、宽同 pstSrc1。	输入
pstDst	输出图像指针。 高、宽同 pstSrc1; 不能为空。	输出
pstAddCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入



参数名称	支持图像类型	地址对齐	分辨率
pstSrc1	U8C1	1 byte	64x64~1920x1080
pstSrc2	U8C1	1 byte	同 pstSrc
pstDst	U8C1	1 byte	同 pstSrc

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

计算公式如下：

$I_{out}(i, j) = x * I_1(i, j) + y * I_2(i, j)$  其中， $I_1(i, j)$  对应 pstSrc1， $I_2(i, j)$  对应 pstSrc2， $I_{out}(i, j)$  对应 pstDst； $x$ ， $y$  为 pstAddCtrl 中的 u0q16X，u0q16Y；要求定点化前的  $0 < x < 1$ ， $0 < y < 1$ ，且  $x + y = 1$ 。

#### 【举例】

无。

#### 【相关主题】

[HI\\_MPI\\_IVE\\_Sub](#)

## HI\_MPI\_IVE\_Xor

#### 【描述】

创建两二值图的异或计算任务。

#### 【语法】

```
HI_S32 HI_MPI_IVE_Xor(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc1,
IVE_SRC_IMAGE_S *pstSrc2, IVE_DST_IMAGE_S *pstDst, HI_BOOL bInstant);
```

#### 【参数】



参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc1	源图像 1 指针。 不能为空。	输入
pstSrc2	源图像 1 指针。 不能为空。 高、宽同 pstSrc1。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc1。	输出
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc1	U8C1	1 byte	64x64~1920x1080
pstSrc2	U8C1	1 byte	同 pstSrc
pstDst	U8C1	1 byte	同 pstSrc

**【返回值】**

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

计算公式如下：

$$I_{dst}(x, y) = I_{src1}(x, y) \wedge I_{src2}(x, y)$$

其中， $I_{src1}(x, y)$  对应 pstSrc1， $I_{src2}(x, y)$  对应 pstSrc2， $I_{dst}(x, y)$  对应 pstDst。

**【举例】**

无。

**【相关主题】**

- [HI\\_MPI\\_IVE\\_And](#)
- [HI\\_MPI\\_IVE\\_Or](#)

## HI\_MPI\_IVE\_NCC

**【描述】**

创建两相同分辨率灰度图像的归一化互相关系数计算任务。

**【语法】**

```
HI_S32 HI_MPI_IVE_NCC(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc1,
IVE_SRC_IMAGE_S *pstSrc2, IVE_DST_MEM_INFO_S *pstDst, HI_BOOL bInstant);
```

**【参数】**

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc1	源 1 图像指针。 不能为空。	输入
pstSrc2	源 2 图像指针。 不能为空。 高、宽同 pstSrc1。	输入
pstDst	输出数据指针。 不能为空。 内存至少需配置：sizeof(IVE_NCC_DST_MEM_S)。	输出
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc1	U8C1	1 byte	32x32~1920x1080
pstSrc2	U8C1	1 byte	同 pstSrc
pstDst	-	16 byte	-

**【返回值】**



返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

- 计算公式如下

$$NCC(I_{src1}, I_{src2}) = \frac{\sum_{i=1}^w \sum_{j=1}^h (I_{src1}(i, j) * I_{src2}(i, j))}{\sqrt{\sum_{i=1}^w \sum_{j=1}^h (I_{src1}^2(i, j))} \sqrt{\sum_{i=1}^w \sum_{j=1}^h (I_{src2}^2(i, j))}}$$

- 仅输出上面公式的分子、开方之前的两个分母项，即 pstDst→u64Numerator、pstDst→u64QuadSum1、pstDst→u64QuadSum2 分别对应上面公式的

$$\sum_{i=1}^w \sum_{j=1}^h (I_{src1}(i, j) * I_{src2}(i, j))、\sum_{i=1}^w \sum_{j=1}^h (I_{src1}^2(i, j))、\sum_{i=1}^w \sum_{j=1}^h (I_{src2}^2(i, j))。$$

**【举例】**

无。

**【相关主题】**

无。

## HI\_MPI\_IVE\_CCL

**【描述】**

创建二值图像的连通区域标记任务。

**【语法】**

```
HI_S32 HI_MPI_IVE_CCL(IVE_HANDLE *pIveHandle, IVE_IMAGE_S *pstSrcDst,
IVE_DST_MEM_INFO_S *pstBlob, IVE_CCL_CTRL_S *pstCclCtrl, HI_BOOL
bInstant);
```

**【参数】**

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出



参数名称	描述	输入/输出
pstSrcDst	源图像指针，连通区域标记在源图像上进行，即源图像同时也是标记图像输出。 不能为空。	输入、输出
pstBlob	连通区域信息指针。 不能为空。 内存至少需配置为 sizeof(IVE_CCBLOB_S)大小，最多输出 254 个有效的连通区域。	输出
pstCclCtrl	控制参数指针 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrcDst	U8C1	16 byte	见【芯片差异】
pstBlob	-	16 byte	-

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	pstSrcDst 分辨率 64x64~720x640 仅支持 8 连通
Hi3536V100	pstSrcDst 分辨率 64x64~720x640 仅支持 8 连通
Hi3521AV100	pstSrcDst 分辨率 64x64~720x640 仅支持 8 连通
Hi3518EV200	pstSrcDst 分辨率 64x64~720x640 仅支持 8 连通





芯片类型	差异
Hi3531AV100	pstSrcDst 分辨率 64x64~720x640 仅支持 8 连通
Hi3519V100	pstSrcDst 分辨率 64x64~1280x720 支持 4 连通和 8 连通

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 连通区域的信息保存在 pstBlob→astRegion 中。
- pstBlob→u8RegionNum 表示有效的连通区域数目，最多 254 个有效的连通区域；有效的连通区域的面积大于 pstBlob→u16CurAreaThr，标记号为其所在 pstBlob→astRegion 数组元素的下标+1。有效的连通区域并不一定连续地存储在数组中，而很可能是间断的分布在数组中。
- 若 pstBlob→s8LabelStatus 为 0，则标记成功（一个区域一个标记）；若为-1，则标记失败（一个区域多个标记或者多个区域共用一个标记）。对于后者，若用户需要正确的标记号，还需要再次根据 pstBlob 中的外接矩形信息重新标记。不管标记是否成功，连通区域的外接矩形信息一定是正确可用的。
- 输出的连通区域会用 pstCclCtrl→u16InitAreaThr 进行筛选，面积小于等于 pstCclCtrl→u16InitAreaThr 均会被置为 0。
- 当连通区域数目大于 254，会用 pstCclCtrl→u16InitAreaThr 删除面积小的连通区域；若 pstCclCtrl→u16InitAreaThr 不满足删除条件，会以 pstCclCtrl→u16Step 为步长，增大删除连通区域的面积阈值。
- 最终的面积阈值存储在 pstBlob→u16CurAreaThr 中。

#### 【举例】

无。

#### 【相关主题】

无。

## HI\_MPI\_IVE\_GMM

#### 【描述】

创建 GMM 背景建模任务，支持灰度图、RGB\_PACKAGE 图像的 GMM 背景建模，高斯模型个数为 3 或者 5。

#### 【语法】

```
HI_S32 HI_MPI_IVE_GMM(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_DST_IMAGE_S *pstFg, IVE_DST_IMAGE_S *pstBg, IVE_MEM_INFO_S *pstModel,
```



```
IVE_GMM_CTRL_S *pstGmmCtrl, HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstFg	前景图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstBg	背景图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstModel	GMM 模型参数指针。 不能为空。	输入、输出
pstGmmCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1、U8C3_PACKAGE	16 byte	见【芯片差异】
pstFg	U8C1 的二值图	16 byte	同 pstSrc
pstBg	同 pstSrc	16 byte	同 pstSrc
pstModel	-	16 byte	-

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】



芯片类型	差异
Hi3516AV100	支持, pstSrc 分辨率 720x576
Hi3536V100	支持, pstSrc 分辨率 720x576
Hi3521AV100	支持, pstSrc 分辨率 720x576
Hi3518EV200	不支持
Hi3531AV100	支持, pstSrc 分辨率 720x576
Hi3519V100	支持, pstSrc 分辨率 1280x720

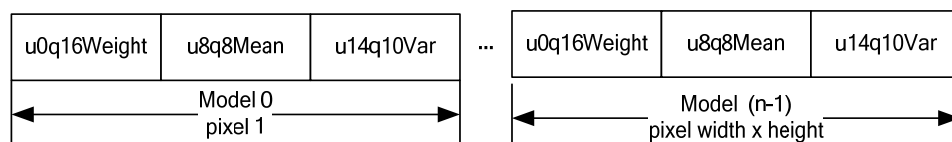
#### 【需求】

- 头文件: hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件: libive.a (PC 上模拟用 ive\_clib2.x.lib)

#### 【注意】

- GMM 的实现方式参考了 OpenCV 中的 MOG 和 MOG2。
- 源图像类型只能为 U8C1 或 U8C3\_PACKAGE, 分别用于灰度图和 RGB 图的 GMM 背景建模。
- 前景图像是二值图, 类型只能为 U8C1; 背景图像与源图像类型一致。
- 灰度图像 GMM 采用 n 个(n=3 或 5)高斯模型, pstModel 的内存排列方式如图 2-12 所示。

图2-12 灰度图像 GMM 模型的内存配置示意图

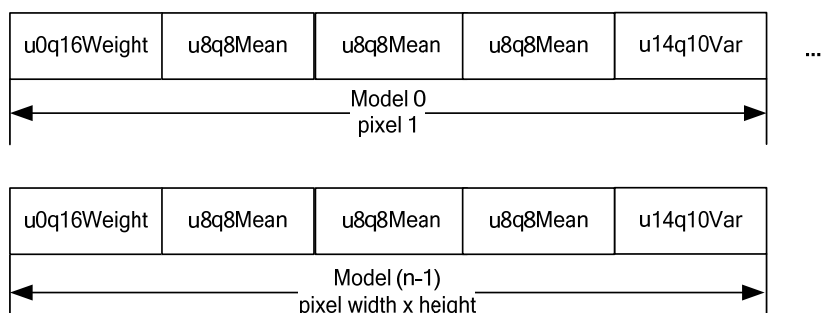


一个像素的单个高斯模型参数 weight 用 2 字节、mean 用 2 字节、var 用 3 字节; 因此 pstModel 需要分配的内存大小:

```
pstModel→u32Size = 7 * pstSrc→u16Width * pstSrc→u16Height *  
pstGmmCtrl→u8ModeNum
```

- RGB 图像 GMM 采用 n 个(n=3 或 5)高斯模型, pstModel 的内存排列方式如图 2-13 所示。

图2-13 RGB 图像 GMM 模型的内存配置示意图



一个像素的单个高斯模型参数 `weight` 用 2 字节、`mean[3]` 用 2\*3 字节、`var` 用 3 字节；因此 `pstModel` 需要分配的内存大小：

```
pstModel→u32Size = 11 * pstSrc→u16Width * pstSrc→u16Height *
pstGmmCtrl→u8ModeNum
```

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_MatchBgModel](#)
- [HI\\_MPI\\_IVE\\_UpdateBgModel](#)
- [HI\\_MPI\\_IVE\\_GMM2](#)

## HI\_MPI\_IVE\_GMM2

#### 【描述】

创建 GMM 背景建模任务，支持 1-5 个高斯模型，支持灰度图和 RGB\_PACKAGE 图输入，支持全局及像素级别的灵敏度系数以及前景模型时长更新系数。

#### 【语法】

```
HI_S32 HI_MPI_IVE_GMM2(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,
IVE_SRC_IMAGE_S *pstFactor, IVE_DST_IMAGE_S *pstFg, IVE_DST_IMAGE_S
*pstBg, IVE_DST_IMAGE_S *pstMatchModelInfo, IVE_MEM_INFO_S *pstModel,
IVE_GMM2_CTRL_S *pstGmm2Ctrl, HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
<code>pIveHandle</code>	handle 指针。 不能为空。	输出



参数名称	描述	输入/输出
pstSrc	源图像指针。 不能为空。	输入
pstFactor	模型更新参数指针。 当且仅当 pstGmm2Ctrl→enSnsFactorMode pstGmm2Ctrl→enLifeUpdateFactorMode 均使用全局模式时可以为空。	输入
pstFg	前景图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstBg	背景图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstMatchModelInfo	模型匹配系数指针。 不能为空。	输出
pstModel	GMM 模型参数指针。 不能为空。	输入、输出
pstGmm2Ctrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1、 U8C3_PACKAGE	16 byte	64x64~1280x720
pstFactor	U16C1	16 byte	同 pstSrc
pstFg	U8C1 的二值图	16 byte	同 pstSrc
pstBg	同 pstSrc	16 byte	同 pstSrc
pstMatchModelInfo	U8C1	16 byte	同 pstSrc
pstModel	-	16 byte	-

#### 【返回值】



返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	不支持
Hi3536V100	不支持
Hi3521AV100	不支持
Hi3518EV200	不支持
Hi3531AV100	不支持
Hi3519V100	支持

#### 【需求】

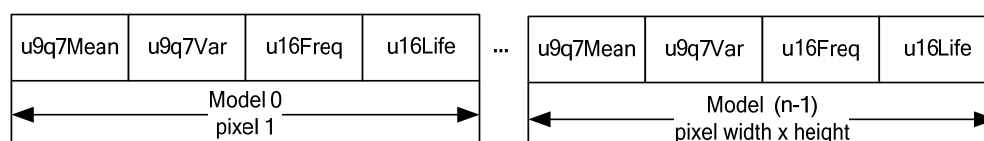
- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- GMM2 在参考了 OPENCV 的 MOG 和 MOG2 的基础上，增加了像素级别的参数控制。
- 源图像 pstSrc 类型只能为 U8C1 或 U8C3\_PACKAGE，分别用于灰度图和 RGB 图的 GMM 背景建模。
- 模型更新参数 pstFactor 为 U16C1 图像：每个元素用 16 bit 表示，低 8 bit 为灵敏度系数，用于控制模型匹配时方差倍数；高 8 bit 为前景模型时长更新参数，用于控制背景模型形成时间。
- 模型匹配系数指针 pstMatchModelInfo 为 U8C1 图像：每个元素用 8bit 表示，低 1 bit 为高斯模型匹配标志，0 表示匹配失败，1 表示匹配成功；高 7 bit 为频率最大模型序号。
- GMM2 的频率参数（pstGmm2Ctrl 中的 u16FreqInitVal、u16FreqReduFactor、u16FreqAddFactor、u16FreqThr）用于控制模型排序和模型有效时间。
  - u16FreqInitVal 越大，模型有效时间越大；
  - u16FreqReduFactor 越大，模型有效时间越长，模型频率通过乘以频率衰减系数 u16FreqReduFactor/65536，达到频率衰减的目的；
  - u16FreqAddFactor 越大，模型有效时间越长；
  - u16FreqThr 越大，模型有效时间越短。

- GMM2 的模型时长参数(pstGmm2Ctrl 中的 u16LifeThr)用于控制前景模型成为背景的时间。
  - u16LifeThr 越大，前景持续时间越长；
  - 单高斯模型下，模型时长参数不生效。
- 灰度图像 GMM2 采用  $n$  个 ( $1 \leq n \leq 5$ ) 高斯模型，pstModel 的内存排列方式如图 2-14 所示。

图2-14 灰度图像 GMM2 模型的内存配置示意图

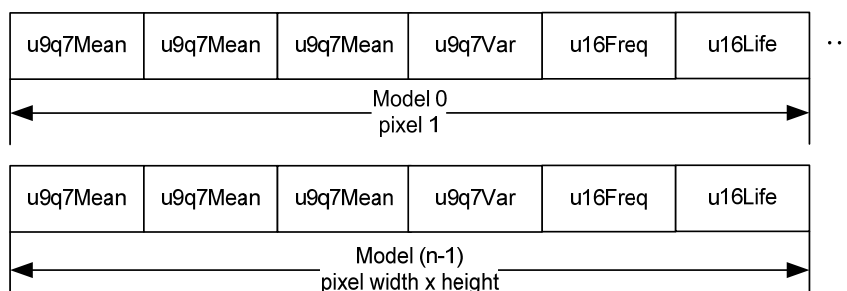


一个像素的单个高斯模型参数 mean 用 2 字节，var 用 2 字节，freq 用 2 字节，life 用 2 字节；因此 pstModel 需要分配的内存大小：

```
pstModel->u32Size = 8 * pstSrc->u16Width * pstSrc->u16Height * pstGmm2Ctrl->u8ModelNum
```

- RGB 图像 GMM2 采用  $n$  个 ( $1 \leq n \leq 5$ ) 高斯模型，pstModel 的内存排列方式如图 2-15 所示。

图2-15 RGB 图像 GMM2 模型的内存配置示意图



一个像素的单个高斯模型参数 mean[3]用 6 字节，var 用 2 字节，freq 用 2 字节，life 用 2 字节；因此 pstModel 需要分配的内存大小：

```
pstModel->u32Size = 12 * pstSrc->u16Width * pstSrc->u16Height * pstGmm2Ctrl->u8ModelNum
```

#### 【举例】

无。

#### 【相关主题】



- [HI\\_MPI\\_IVE\\_MatchBgModel](#)
- [HI\\_MPI\\_IVE\\_UpdateBgModel](#)
- [HI\\_MPI\\_IVE\\_GMM](#)

## HI\_MPI\_IVE\_CannyHysEdge

### 【描述】

灰度图的 Canny 边缘提取的前半部：求梯度、计算梯度幅值幅角、磁滞阈值化及非极大抑制。

### 【语法】

```
HI_S32 HI_MPI_IVE_CannyHysEdge(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstEdge, IVE_DST_MEM_INFO_S *pstStack,
IVE_CANNY_HYS_EDGE_CTRL_S *pstCannyHysEdgeCtrl, HI_BOOL bInstant);
```

### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstEdge	强弱边缘标志图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstStack	强边缘点坐标栈。 不能为空。 内存至少配置： pstSrc→u16Width * pstSrc→u16Height * (sizeof(IVE_POINT_U16_S)) + sizeof(IVE_CANNY_STACK_SIZE_S)	输出
pstCannyHysEdgeCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1024
pstEdge	U8C1	16 byte	同 pstSrc





参数名称	支持图像类型	地址对齐	分辨率
pstStack	-	16 byte	-
pstCannyHysEdgeCtrl →stMem	-	16 byte	-

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- pstEdge 仅有 0、1、2 三个取值：
  - 0 表示弱边缘点
  - 1 表示非边缘点
  - 2 表示强边缘点
- pstStack 中存储强边缘点的坐标信息。
- pstCannyHysEdgeCtrl→stMem 至少需要分配的内存大小  
pstCannyHysEdgeCtrl→stMem.u32Size  
= IveGetStride(pstSrc→u16Width, IVE\_STRIDE\_ALIGN)\* 3 \* pstSrc→u16Height。
- 该任务完成后，必须要使用 [HI\\_MPI\\_IVE\\_CannyEdge](#) 函数才能输出 Canny 边缘图像。

#### 【举例】

无。

#### 【相关主题】

[HI\\_MPI\\_IVE\\_CannyEdge](#)

## HI\_MPI\_IVE\_CannyEdge

#### 【描述】

灰度图的 Canny 边缘提取的后半部：连接边缘点，形成 Canny 边缘图。

#### 【语法】



```
HI_S32 HI_MPI_IVE_CannyEdge(IVE_IMAGE_S *pstEdge, IVE_MEM_INFO_S  
*pstStack);
```

#### 【参数】

参数名称	描述	输入/输出
pstEdge	作为输入是强弱边缘标志图像指针；作为输出是边缘二值图像指针。 不能为空。	输入、输出
pstStack	强边缘点坐标栈。 不能为空。	输入、输出

参数名称	支持图像类型	地址对齐	分辨率
pstEdge	U8C1	16 byte	64x64~1920x1024
pstStack	-	16 byte	-

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 iva\_clib2.x.lib）

#### 【注意】

使用该接口前必须调用 [HI\\_MPI\\_IVE\\_CannyHysEdge](#)，在保证 [HI\\_MPI\\_IVE\\_CannyHysEdge](#) 任务完成的情况下，使用 [HI\\_MPI\\_IVE\\_CannyHysEdge](#) 的输出 pstEdge、pstStack 作为该接口的参数输入。

#### 【举例】

无。

#### 【相关主题】

[HI\\_MPI\\_IVE\\_CannyHysEdge](#)



## HI\_MPI\_IVE\_LBP

### 【描述】

创建 LBP 计算任务。

### 【语法】

```
HI_S32 HI_MPI_IVE_LBP(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc,  
IVE_DST_IMAGE_S *pstDst, IVE_LBP_CTRL_S *pstLbpCtrl, HI_BOOL bInstant);
```

### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDst	输出图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstLbpCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1024
pstDst	U8C1	16 byte	64x64~1920x1024

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【芯片差异】



芯片类型	差异
Hi3516AV100	支持
Hi3536V100	支持
Hi3521AV100	支持
Hi3518EV200	不支持
Hi3531AV100	支持
Hi3519V100	支持

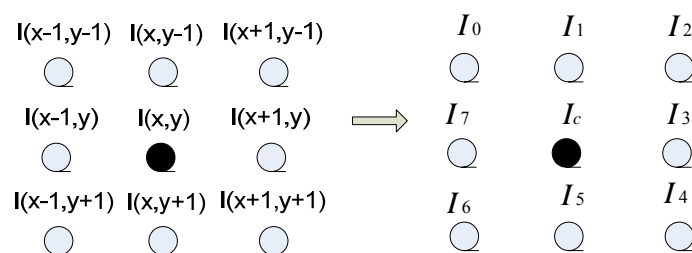
#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

LBP 计算公式如图 2-16 所示。

图2-16 LBP 计算公式示意图



- IVE\_LBP\_CMP\_NORMAL

$$lbp(x, y) = \sum_{i=0}^7 ((I_i - I_c) \geq thr) \ll (7 - i), \text{其中 } thr \in [-128, 127];$$

- IVE\_LBP\_CMP\_ABS

$$lbp(x, y) = \sum_{i=0}^7 (abs(I_i - I_c) \geq thr) \ll (7 - i), \text{其中 } thr \in [0, 255];$$

- 其中， $I(x, y)$  对应 pstSrc， $lbp(x, y)$  对应 pstDst， $thr$  对应 pstLbpCtrl→un8BitThr。

#### 【举例】

无。

#### 【相关主题】



无。

## HI\_MPI\_IVE\_NormGrad

### 【描述】

创建归一化梯度计算任务，梯度分量均归一化到 S8。

### 【语法】

```
HI_S32 HI_MPI_IVE_NormGrad(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstDstH, IVE_DST_IMAGE_S *pstDstV,
IVE_DST_IMAGE_S *pstDstHV, IVE_NORM_GRAD_CTRL_S *pstNormGradCtrl, HI_BOOL
bInstant);
```

### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstDstH	由模板直接滤波并归一到 S8 后得到的梯度分量图像(H)指针。 根据 pstNormGradCtrl→enOutCtrl，若需要输出则不能为空。	输出
pstDstV	由转置后的模板滤波并归一到 S8 后得到的梯度分量图像(V)指针。 根据 pstNormGradCtrl→enOutCtrl，若需要输出则不能为空。	输出
pstDstHV	由模板和转置后的模板直接滤波，并且均归一到 S8 后，采用 package 格式存储（如图 1-7）的图像指针。 根据 pstNormGradCtrl→enOutCtrl，若需要输出则不能为空。	输出
pstNormGradCtrl	控制信息指针。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	64x64~1920x1024
pstDstH	S8C1	16 byte	同 pstSrc



参数名称	支持图像类型	地址对齐	分辨率
pstDstV	S8C1	16 byte	同 pstSrc
pstDstHV	S8C2_PACKAGE	16 byte	同 pstSrc

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

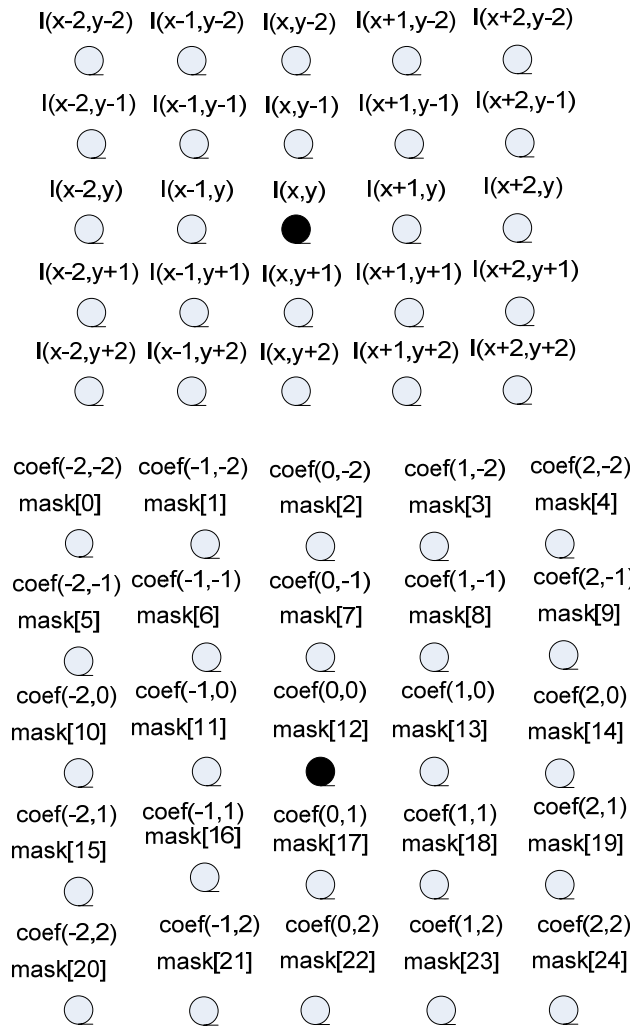
#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 控制参数中输出模式如下：
  - IVE\_NORM\_GRAD\_OUT\_CTRL\_HOR\_AND\_VER 时，pstDstH 和 pstDstV 指针不能为空，且要求跨度一致；
  - IVE\_NORM\_GRAD\_OUT\_CTRL\_HOR 时，pstDstH 不能为空；
  - IVE\_NORM\_GRAD\_OUT\_CTRL\_VER 时，pstDstV 不能为空；
  - IVE\_NORM\_GRAD\_OUT\_CTRL\_COMBINE 时，pstDstHV 不能为空。
- NormGrad 计算公式如[图 2-17](#)所示。

图2-17 NormGrad 计算公式示意图



$$I_{out}(x,y) = \left\{ \sum_{-2 < j < 2} \sum_{-2 < i < 2} I(x+i, y+j) \bullet \text{coef}(i,j) \right\} \gg \text{norm}$$

#### 【举例】

无。

#### 【相关主题】

[HI\\_MPI\\_IVE\\_Sobel](#)

## HI\_MPI\_IVE\_LKOpticalFlow

#### 【描述】

创建单层 LK 光流计算任务。

#### 【语法】



```
HI_S32 HI_MPI_IVE_LKOpticalFlow(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrcPre, IVE_SRC_IMAGE_S *pstSrcCur, IVE_SRC_MEM_INFO_S *pstPoint,
IVE_MEM_INFO_S *pstMv, IVE_LK_OPTICAL_FLOW_CTRL_S *pstLkOptiFlowCtrl,
HI_BOOL bInstant);
```

## 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrcPre	前一帧图像指针。 不能为空。	输入
pstSrcCur	当前图像指针。 不能为空。 高、宽同 pstSrcPre。	输入
pstPoint	当前金字塔层的初始特征点坐标。 不能为空。 坐标只能为 IVE_POINT_S25Q7_S 类型；内存至少需分配：pstLkOptiFlowCtrl→u16CornerNum * sizeof(IVE_POINT_S25Q7_S)。	输入
pstMv	对应于 pstPoint 的特征点运动位移矢量。 不能为空。 首次计算需初始化为 0 输入；后续层计算需输入上一层计算得到的运动位移矢量；位移只能为 IVE_MV_S9Q7_S 类型；内存至少需分配：pstLkOptiFlowCtrl→u16CornerNum * sizeof(IVE_MV_S9Q7_S)	输入、输出
pstLkOptiFlowCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrcPre	U8C1	16 byte	64x64~720x576
pstSrcCur	U8C1	16 byte	同 pstSrcPre

## 【返回值】





返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持
Hi3536V100	支持
Hi3521AV100	支持
Hi3518EV200	不支持
Hi3531AV100	支持
Hi3519V100	不支持

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.0.lib）

#### 【注意】

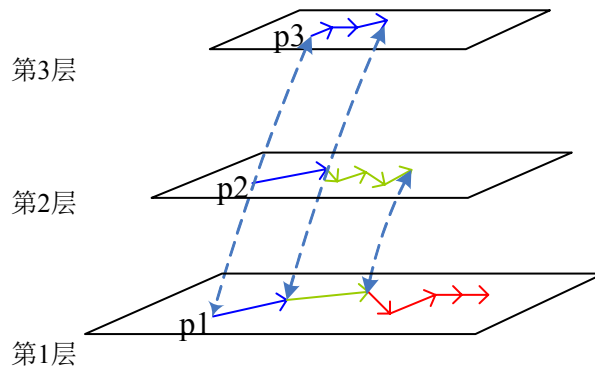
- 求解下面的光流方程中，仅用到特征点周围 7X7 像素的来计算对应的  $I_x$ 、 $I_y$ 、 $I_t$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

其中， $I_x$ 、 $I_y$ 、 $I_t$  分别表示当前图像在  $x$ 、 $y$  方向的偏导，当前图像与前一帧图像的差分。

- 以 3 层金字塔 LK 光流计算为例，要求每层图像的高、宽是上一层图像高、宽的一半，其计算示意图如[图 2-18](#) 所示。

图2-18 3 层金字塔 LK 光流计算示意图



- 根据输入的特征点坐标，计算出 3 层金字塔特征点对应的坐标：p0，p1，p2；
- 以 p2 和初始为 0 的 mv2 作为输入调用 LK 算子求出在第 2 层上的位移 mv2；
- 以 p1 和 mv2 作为输入调用 LK 算子求出第 1 层上的位移 mv1；
- 以 p0 和 mv1 作为输入调用 LK 算子求出第 0 层上的位移 mv0；
- 若第 0 层不是原始图像，根据第 0 层与原始图像的的比例关系可以得到 LK 光流的真正位移 mv。

请注意设计和使用限制：每个特征点仅以该特征点为中心固定大小窗口的数据进行计算，若迭代计算过程中，该特征点位移目标点超出该固定大小窗口会导致计算光流失败。

#### 【举例】

无。

#### 【相关主题】

无。

## HI\_MPI\_IVE\_LKOpticalFlowPyr

#### 【描述】

创建多层金字塔 LK 光流计算任务。

#### 【语法】

```
HI_S32 HI_MPI_IVE_LKOpticalFlowPyr(IVE_HANDLE *pIveHandle,
IVE_SRC_IMAGE_S astSrcPrevPyr[], IVE_SRC_IMAGE_S astSrcNextPyr[],
IVE_SRC_MEM_INFO_S *pstPrevPts, IVE_MEM_INFO_S *pstNextPts,
IVE_DST_MEM_INFO_S *pstStatus, IVE_DST_MEM_INFO_S *pstErr,
IVE_LK_OPTICAL_FLOW_PYR_CTRL_S *pstLkOptiFlowPyrCtrl, HI_BOOL bInstant);
```

#### 【参数】



参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
astSrcPrevPyr[]	前一帧图像的金字塔图像数组，金字塔层数由 pstLkOptiFlowPyrCtrl→u8MaxLevel 控制。 不能为空。	输入
astSrcNextPyr[]	下一帧图像的金字塔图像数组，与 astSrcPrevPyr[]有相同的层数，每层图像大小类型均相同。 不能为空。	输入
pstPrevPts	前一帧图像金字塔第 0 层（astSrcPrevPyr[0]） 的初始特征点数组。 不能为空。 坐标只能为 IVE_POINT_S25Q7_S 类型；内存 至少需分配： pstLkOptiFlowPyrCtrl→u16PtsNum * sizeof(IVE_POINT_S25Q7_S)。	输入
pstNextPts	特征点 pstPrevPts 经过金字塔计算 LK 光流得到 对应于下一帧图像金字塔第 0 层 （astSrcNextPyr[]）的坐标。当 pstLkOptiFlowPyrCtrl→bUseInitFlow 为 true 时，需要初始化该特征点数组。 不能为空。 坐标只能为 IVE_POINT_S25Q7_S 类型；内存 至少需分配： pstLkOptiFlowPyrCtrl→u16PtsNum * sizeof(IVE_POINT_S25Q7_S)。	输入、输出
pstStatus	pstNextPts 中每个特征点对应一个 HI_U8 的跟 踪状态信息，1 表示成功，0 表示失败。	输出
pstErr	对 pstNextPts 中每个跟踪成功的特征点，对比 pstPrevPts 中对应特征点周边进行的相似度误差 估计（HI_U9Q7 类型），跟踪失败的特征点不 做估计。	输出
pstLkOptiFlowPyrCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入



参数名称	支持图像类型	地址对齐	分辨率
astSrcPrevPyr[0]	U8C1	16 byte	64x64~1280x720
astSrcNextPyr[0]	U8C1	16 byte	astSrcPrevPyr[0]
astSrcPrevPyr[n] (金字塔第 n 层, $0 \leq n \leq 3$ )	U8C1	16 byte	astSrcPrevPyr[0]对应高、宽右移 n
astSrcNextPyr[n] (金字塔第 n 层)	U8C1	16 byte	astSrcPrevPyr[0]对应高、宽右移 n
pstPrevPts	-	16byte	-
pstNextPts	-	16byte	-
pstStatus	-	16byte	-
pstErr	-	16byte	-

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	不支持
Hi3536V100	不支持
Hi3521AV100	不支持
Hi3518EV200	不支持
Hi3531AV100	不支持
Hi3519V100	支持

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

#### 【注意】

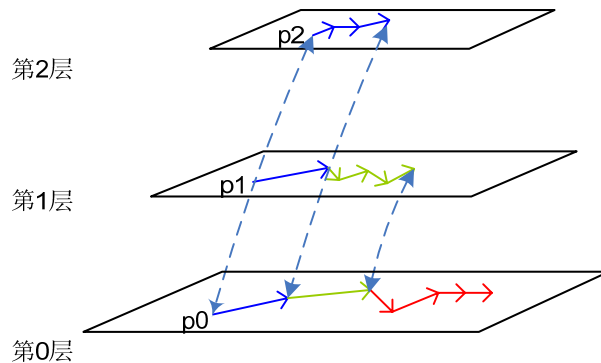
- `pstLkOptiFlowPyrCtrl` → `u8MaxLevel` 取值范围[0, 3]，对应金字塔层数为[1, 4]。
- 求解下面的光流方程中，仅用到特征点周围 7X7 像素的来计算对应的  $I_x$ 、 $I_y$ 、 $I_t$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

其中， $I_x$ 、 $I_y$ 、 $I_t$  分别表示当前图像在  $x$ 、 $y$  方向的偏导，当前图像与前一帧图像的差分。

- 以 3 层金字塔 LK 光流计算为例，要求每层图像的高、宽是上一层图像高、宽的一半，其计算示意图如图 2-19 所示。

图2-19 3 层金字塔 LK 光流计算示意图



- 根据输入的特征点坐标，计算出 3 层金字塔特征点对应的坐标：p0，p1，p2（若需要初始光流，计算出 m0，m1，m2）；
- 以 p2（若需要初始光流，则需要 m2）作为输入求出在第 2 层上的光流终点 n2；
- 计算出 n2 在第 1 层的对应坐标 n1，以 n1 作为输入求出第 1 层上的光流终点 q1；
- 计算出 q1 在第一层的对应坐标 q0，以 q0 作为输入求出第 0 层上的光流终点 q；
- 若第 0 层不是原始图像，根据第 0 层与原始图像的的比例关系可以得到 LK 光流的最终点 p。

**请注意设计和使用限制：**每个特征点仅以该特征点为中心固定大小窗口的数据进行计算，若迭代计算过程中，该特征点位移目标点超出该固定大小窗口会导致计算光流失败。

#### 【举例】

无。

#### 【相关主题】

无。



## HI\_MPI\_IVE\_STCandiCorner

## 【描述】

灰度图像 Shi-Tomasi-like 角点计算的前半部：计算候选角点。

## 【语法】

```
HI_S32 HI_MPI_IVE_STCandiCorner(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstSrc, IVE_DST_IMAGE_S *pstCandiCorner, IVE_ST_CANDI_CORNER_CTRL_S
*pstStCandiCornerCtrl, HI_BOOL bInstant);
```

## 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	源图像指针。 不能为空。	输入
pstCandiCorner	候选角点响应值图像指针。 不能为空。 高、宽同 pstSrc。	输出
pstStCandiCornerCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc	U8C1	16 byte	见【芯片差异】
pstCandiCorner	U8C1	16 byte	同 pstSrc
pstStCandiCornerCtrl →stMem	-	16 byte	-

## 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。



### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持, pstSrc 分辨率 64x64~720x576
Hi3536V100	支持, pstSrc 分辨率 64x64~720x576
Hi3521AV100	支持, pstSrc 分辨率 64x64~720x576
Hi3518EV200	不支持
Hi3531AV100	支持, pstSrc 分辨率 64x64~720x576
Hi3519V100	支持, pstSrc 分辨率 64x64~1280x720

### 【需求】

- 头文件: hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件: libive.a (PC 上模拟用 ive\_clib2.x.lib)

### 【注意】

- 与 OpenCV 中 ShiTomas 角点计算原理类似。
- pstStCandiCornerCtrl→stMem 至少需开辟的内存大小:  
pstStCandiCornerCtrl→stMem.u32Size  
= 4 \* IveGetStride(pstSrc→u16Width, IVE\_STRIDE\_ALIGN) \* pstSrc→u16Height  
+ sizeof(IVE\_ST\_MAX\_EIG\_S)。
- 该任务完成后, 必须要使用 HI\_MPI\_IVE\_STCorner 函数才能得到真正的角点。

### 【举例】

无。

### 【相关主题】

[HI\\_MPI\\_IVE\\_STCorner](#)

## HI\_MPI\_IVE\_STCorner

### 【描述】

灰度图像 Shi-Tomasi-like 角点计算的后半部: 按规则挑选角点。

### 【语法】

```
HI_S32 HI_MPI_IVE_STCorner(IVE_SRC_IMAGE_S * pstCandiCorner,  
IVE_DST_MEM_INFO_S *pstCorner, IVE_ST_CORNER_CTRL_S *pstStCornerCtrl);
```

### 【参数】



参数名称	描述	输入/输出
pstCandiCorner	候选角点响应值图像指针。 不能为空。	输入
pstCorner	角点坐标信息指针。 不能为空。 内存至少需配置： <code>sizeof(IVE_ST_CORNER_INFO_S)</code>	输出
pstStCornerCtrl	控制参数指针。 不能为空。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstCandiCorner	U8C1	16 byte	见【芯片差异】
pstCorner	-	16 byte	-

## 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

## 【芯片差异】

芯片类型	差异
Hi3516AV100	pstSrc 分辨率 64x64~720x576，最多输出 200 个角点
Hi3536V100	pstSrc 分辨率 64x64~720x576，最多输出 200 个角点
Hi3521AV100	pstSrc 分辨率 64x64~720x576，最多输出 200 个角点
Hi3518EV200	不支持
Hi3531AV100	pstSrc 分辨率 64x64~720x576，最多输出 200 个角点
Hi3519V100	pstSrc 分辨率 64x64~1280x720，最多输出 500 个角点

## 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h





- 库文件: libive.a (PC 上模拟用 ive\_clib2.x.lib)

【注意】

- 与 OpenCV 中 ShiTomas 角点计算原理类似。
- pstCorner→u16CornerNum 表示最终得到的角点数目。
- 使用该接口前必须调用 [HI\\_MPI\\_IVE\\_STCandiCorner](#)，在保证 [HI\\_MPI\\_IVE\\_STCandiCorner](#) 任务完成的情况下，使用 [HI\\_MPI\\_IVE\\_STCandiCorner](#) 的输出 pstCandiCorner 作为该接口的参数输入。

【举例】

无。

【相关主题】

[HI\\_MPI\\_IVE\\_STCandiCorner](#)

## HI\_MPI\_IVE\_SAD

【描述】

计算两幅图像按 4x4\8x8\16x16 分块的 16 bit\8 bit SAD 图像，以及对 SAD 进行阈值化输出。

【语法】

```
HI_S32 HI_MPI_IVE_SAD(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S *pstSrc1,  
IVE_SRC_IMAGE_S *pstSrc2, IVE_DST_IMAGE_S *pstSad, IVE_DST_IMAGE_S  
*pstThr, IVE_SAD_CTRL_S *pstSadCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc1	源图像 1 指针。 不能为空。	输入
pstSrc2	源图像 2 指针。 不能为空。 高、宽同 pstSrc1。	输入
pstSad	输出 SAD 图像指针。 根据 pstSadCtrl→enOutCtrl，若需要输出则不能为空。 根据 pstSadCtrl→enMode，对应 4x4、8x8、16x16 分块模式，高、宽分别为 pstSrc1 的 1/4、1/8、1/16。	输出



参数名称	描述	输入/输出
pstThr	输出 SAD 阈值化图像指针。 根据 pstSadCtrl→enOutCtrl，若需要输出则不能为空。 根据 pstSadCtrl→enMode，对应 4x4、8x8、16x16 分块模式，高、宽分别为 pstSrc1 的 1/4、1/8、1/16。	输出
pstSadCtrl	控制信息指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstSrc1	U8C1	1 byte	64x64~1920x1080
pstSrc2	U8C1	1 byte	同 pstSrc1
pstSad	U8C1 、U16C1	16 byte	根据 pstSadCtrl→enMode，对应 4x4、8x8、16x16 分块模式，高、宽分别为 pstSrc1 的 1/4、1/8、1/16。
pstThr	U8C1	16 byte	根据 pstSadCtrl→enMode，对应 4x4、8x8、16x16 分块模式，高、宽分别为 pstSrc1 的 1/4、1/8、1/16。

**【返回值】**

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

**【芯片差异】**

芯片类型	差异
Hi3516AV100	不支持
Hi3536 V100	不支持
Hi3521AV100	支持
Hi3518EV200	支持



芯片类型	差异
Hi3531AV100	支持
Hi3519V100	支持

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

**【注意】**

计算公式如下：

$$SAD_{out}(x, y) = \sum_{\substack{n*x \leq i < n*(x+1) \\ n*y \leq j < n*(y+1)}} |I_1(i, j) - I_2(i, j)| ;$$

$$THR_{out}(x, y) = \begin{cases} minVal & (SAD_{out}(x, y) \leq Thr) \\ maxVal & (SAD_{out}(x, y) > Thr) \end{cases}$$

其中， $I_1(i, j)$  对应 pstSrc1， $I_2(i, j)$  对应 pstSrc2， $SAD_{out}(x, y)$  对应 pstSad， $n$  与 pstSadCtrl→enMode 相关，对应 IVE\_SAD\_MODE\_MB\_4X4、IVE\_SAD\_MODE\_MB\_8X8、IVE\_SAD\_MODE\_MB\_16X16 时分别取 4、8、16；

$THR_{out}(x, y)$  对应 pstThr， $Thr$ 、 $minVal$  和  $maxVal$  分别对应 pstSadCtrl→u16Thr、pstSadCtrl→u8MinVal 和 pstSadCtrl→u8MaxVal。

**【举例】**

无。

**【相关主题】**

无。

## HI\_MPI\_IVE\_Resize

**【描述】**

创建图像缩放任务，支持 bilinear、area 插值缩放，支持多张 U8C1\U8C3\_PLANAR 图像同时输入做一种类型的缩放。

**【语法】**



```
HI_S32 HI_MPI_IVE_Resize(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S astSrc[],  
IVE_DST_IMAGE_S astDst[], IVE_RESIZE_CTRL_S *pstResizeCtrl, HI_BOOL  
bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
astSrc[]	源图像数组。 不能为空。	输入
astDst[]	输出图像数组。 不能为空。 每张图像高、宽、类型同 pstSrc。	输出
pstResizeCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
astSrc[]	U8C1、U8C3_PLANAR	16byte	32x12~1920x1080
astDst[]	U8C1、U8C3_PLANAR	16 byte	32x12~1920x1080

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	不支持
Hi3536V100	不支持
Hi3521AV100	不支持



Hi3518EV200	不支持
Hi3531AV100	不支持
Hi3519V100	支持

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

#### 【注意】

- 基于 OpenCV 中 resize 实现，IVE\_RESIZE\_MODE\_LINEAR、IVE\_RESIZE\_MODE\_AREA 分别对应 OpenCV resize 的 INTER\_LINEAR、INTER\_AREA。
- 支持 U8C1、U8C3\_PLANAR 混合图像数组输入，但所有图像的缩放模式相同。
- 最大支持 16 倍缩放。
- pstResizeCtrl→stMem 内存至少需要  $25 * U8C1\_NUM + 49 * (pstResizeCtrl \rightarrow u16Num - U8C1\_NUM)$  字节，其中 U8C1\_NUM 为混合图像数组中 U8C1 图像的数目。

#### 【举例】

无。

#### 【相关主题】

无。

## HI\_MPI\_IVE\_GradFg

#### 【描述】

根据背景图像和当前帧图像的梯度信息计算梯度前景图像。

#### 【语法】

```
HI_S32 HI_MPI_IVE_GradFg(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstBgDiffFg, IVE_SRC_IMAGE_S *pstCurGrad, IVE_SRC_IMAGE_S *pstBgGrad,
IVE_DST_IMAGE_S *pstGradFg, IVE_GRAD_FG_CTRL_S *pstGradFgCtrl, HI_BOOL
bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstBgDiffFg	背景差分前景图像指针。 不能为空。	输入



参数名称	描述	输入/输出
pstCurGrad	当前帧梯度图像指针。 不能为空。 高、宽同 pstCurGrad。	输入
pstBgGrad	背景梯度图像指针。 不能为空。 高、宽同 pstCurGrad。	输入
pstGradFg	梯度前景图像指针。 不能为空。 高、宽同 pstCurGrad。	输出
pstGradFgCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstDiffFg	S8C1	16 byte	见【芯片差异】
pstCurGrad	S8C2_PACKAGE	16 byte	同 pstDiffFg
pstBgGrad	S8C2_PACKAGE	16 byte	同 pstDiffFg
pstGradFg	U8C1	16 byte	同 pstDiffFg

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持，pstDiffFg 分辨率 720x576
Hi3536V100	支持，pstDiffFg 分辨率 720x576
Hi3521AV100	支持，pstDiffFg 分辨率 720x576



Hi3518EV200	不支持
Hi3531AV100	支持，pstDiffFg 分辨率 720x576
Hi3519V100	支持，pstDiffFg 分辨率 1280x720

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

背景梯度图像和当前梯度图像的类型为 S8C2\_PACKAGE，水平和垂直方向梯度按照 [xyxyxy...] 格式存储。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_MatchBgModel](#)
- [HI\\_MPI\\_IVE\\_UpdateBgModel](#)
- [HI\\_MPI\\_IVE\\_GMM](#)

## HI\_MPI\_IVE\_MatchBgModel

#### 【描述】

基于 Codebook 演进的背景模型匹配。

#### 【语法】

```
HI_S32 HI_MPI_IVE_MatchBgModel(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S
*pstCurImg, IVE_DATA_S *pstBgModel, IVE_IMAGE_S *pstFgFlag,
IVE_DST_IMAGE_S *pstBgDiffFg, IVE_DST_IMAGE_S *pstFrmDiffFg,
IVE_DST_MEM_INFO_S *pstStatData, IVE_MATCH_BG_MODEL_CTRL_S
*pstMatchBgModelCtrl, HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstCurImg	当前帧灰度图像指针。 不能为空。	输入



参数名称	描述	输入/输出
pstBgModel	背景模型数据指针。 不能为空。 高同 pstCurImg，宽 = pstCurImg→ u16Width * sizeof(IVE_BG_MODEL_PIX_S)。	输入、输出
pstFgFlag	前景状态图像指针。 不能为空。 高、宽同 pstCurImg。	输入、输出
pstBgDiffFg	背景差分前景图像指针。 不能为空。 高、宽同 pstCurImg。	输出
pstFrmDiffFg	帧间差分前景图像指针； 不能为空。 高、宽同 pstCurImg。	输出
pstStatData	前景状态数据指针。 不能为空。 内存至少需配置 sizeof(IVE_FG_STAT_DATA_S)。	输出
pstMatchBgModelCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstCurImg	U8C1	16 byte	见【芯片差异】
pstBgModel	-	16 byte	-
pstFgFlag	U8C1	16 byte	同 pstCurImg
pstBgDiffFg	S8C1	16 byte	同 pstCurImg
pstFrmDiffFg	S8C1	16 byte	同 pstCurImg

#### 【返回值】

返回值	描述
0	成功。





返回值	描述
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持，pstCurImg 分辨率 720x576
Hi3536V100	支持，pstCurImg 分辨率 720x576
Hi3521AV100	支持，pstCurImg 分辨率 720x576
Hi3518EV200	不支持
Hi3531AV100	支持，pstCurImg 分辨率 720x576
Hi3519V100	支持，pstCurImg 分辨率 1280x720

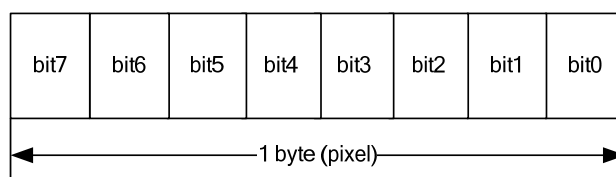
#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 要求 pstFgFlag、pstBgDiffFg、pstFrmDiffFg 跨度一致。
- 背景模型数据 pstBgModel 中每个像素以 [IVE\\_BG\\_MODEL\\_PIX\\_S](#)(24 字节)表示，即  
 $\text{pstModel} \rightarrow \text{u16Width} = \text{sizeof}(\text{IVE\_BG\_MODEL\_PIX\_S}) * \text{pstSrc} \rightarrow \text{u16Width},$   
 $\text{pstModel} \rightarrow \text{u16Height} = \text{pstSrc} \rightarrow \text{u16Height},$  至少需要分配内存大小为  
 $\text{IveGetStride}(\text{sizeof}(\text{IVE\_BG\_MODEL\_PIX\_S}) * \text{pstSrc} \rightarrow \text{u16Width},$   
 $\text{IVE\_STRIDE\_ALIGN}) * \text{pstModel} \rightarrow \text{u16Height}.$
- 前景状态图像 pstFgFlag 为 U8C1 类型，其各比特位表示不同的状态信息，单个像素各比特位示意图如[图 2-20](#)，按从右到左由低位到高位顺序排布：

图2-20 前景状态标志图形单个像素各比特位示意图





其各个比特位表示的含义如下：

- 比特位只用到 bit0、bit1、bit2、bit5、bit6；其中 bit0、bit1、bit2 是由本算子计算作为输出，bit5、bit6 是由外部函数计算作为输入。
- bit1 为 1 时表示像素为前景；
- bit1 为 1 且 bit0 为 1 时表示像素为运动前景；
- bit1 为 1 且 bit0 为 0 时表示像素为变化前景；
- bit2 为 1 时表示像素的背景模型处于工作状态；
- bit5 和 bit6 表示外部函数对前景状态的反馈，bit5 为 1 时表示前景像素需要短时间保持，bit6 为 1 时表示前景像素需要长时间保持。

**【举例】**

无。

**【相关主题】**

- [HI\\_MPI\\_IVE\\_UpdateBgModel](#)
- [HI\\_MPI\\_IVE\\_GradFg](#)
- [HI\\_MPI\\_IVE\\_GMM](#)

## HI\_MPI\_IVE\_UpdateBgModel

**【描述】**

基于 Codebook 演进的背景模型更新，对背景模型的内部状态进行更新。

**【语法】**

```
HI_S32 HI_MPI_IVE_UpdateBgModel(IVE_HANDLE *pIveHandle, IVE_DATA_S
*pstBgModel, IVE_IMAGE_S *pstFgFlag, IVE_DST_IMAGE_S *pstBgImg,
IVE_DST_IMAGE_S *pstChgStaImg, IVE_DST_IMAGE_S *pstChgStaFg,
IVE_DST_IMAGE_S *pstChgStaLife, IVE_DST_MEM_INFO_S *pstStatData,
IVE_UPDATE_BG_MODEL_CTRL_S *pstUpdateBgModelCtrl, HI_BOOL bInstant);
```

**【参数】**

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstBgModel	背景模型数据指针。 不能为空。	输入/输出
pstFgFlag	前景状态图像指针。 不能为空。	输入/输出



参数名称	描述	输入/输出
pstBgImg	背景灰度图像指针。 不能为空。 高、宽同 pstFgFlag。	输出
pstChgStaImg	变化状态灰度图像指针。 当 pstUpdateBgModelCtrl→u8DetChgRegion 为 0 时，可以为空。 高、宽同 pstFgFlag。	输出
pstChgStaFg	变化状态前景图像指针。 当 pstUpdateBgModelCtrl→u8DetChgRegion 为 0 时，可以为空。 高、宽同 pstFgFlag。	输出
pstChgStaLife	变化状态像素的生命时间图像指针。 当 pstUpdateBgModelCtrl→u8DetChgRegion 为 0 时，可以为空。 高、宽同 pstFgFlag。	输出
pstStatData	背景状态数据指针。 不能为空。 内存至少需配置 <code>sizeof(IVE_BG_STAT_DATA_S)</code> 。	输出
pstUpdateBgModelCtrl	控制参数指针。 不能为空。	输入
bInstant	及时返回结果标志。	输入

参数名称	支持图像类型	地址对齐	分辨率
pstBgModel	-	16 byte	-
pstFgFlag	U8C1	16 byte	见【芯片差异】
pstBgImg	U8C1	16 byte	同 pstFgFlag
pstChgStaImg	U8C1	16 byte	同 pstFgFlag
pstChgStaFg	S8C1	16 byte	同 pstFgFlag
pstChgStaLife	U16C1	16 byte	同 pstFgFlag

#### 【返回值】



返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持，pstFgFlag 分辨率 720x576
Hi3536V100	支持，pstFgFlag 分辨率 720x576
Hi3521AV100	支持，pstFgFlag 分辨率 720x576
Hi3518EV200	不支持
Hi3531AV100	支持，pstFgFlag 分辨率 720x576
Hi3519V100	支持，pstFgFlag 分辨率 1280x720

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

#### 【注意】

- 要求 pstFgFlag、pstBgImg、pstChgStaImg（不为空时）、pstChgStaFg（不为空时）跨度一致。
- 背景模型数据 pstModel 参考 [HI\\_MPI\\_IVE\\_MatchBgModel](#) 中的说明。
- pstChgStaFg 表示变化状态前景图像，其中像素非 0 表示前景，否则表示背景。
- pstChgStaLife 表示变化状态前景像素的生命时间图像，其像素值表示变化前景的持续时间。
- 变化状态指像素值发生变化而成为前景，并且变化后的像素值较长时间都保持稳定的状态，这一般是由静止遗留物或者静止移走物在图像中产生。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_MatchBgModel](#)
- [HI\\_MPI\\_IVE\\_GradFg](#)
- [HI\\_MPI\\_IVE\\_GMM](#)



## HI\_MPI\_IVE\_ANN\_MLP\_LoadModel

### 【描述】

读取 ANN\_MLP 模型文件，初始化模型数据。

### 【语法】

```
HI_S32 HI_MPI_IVE_ANN_MLP_LoadModel(const HI_CHAR *pchFileName,  
IVE_ANN_MLP_MODEL_S *pstAnnMlpModel)
```

### 【参数】

参数名称	描述	输入/输出
pchFileName	模型文件路径及文件名。 不能为空。	输入
pstAnnMlpModel	模型数据结构体指针。 不能为空。	输出

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

### 【注意】

- 文件名必须以.bin 为后缀；.bin 文件必须用海思提供的工具 ive\_tool\_xml2bin.exe 生成。
- 该接口必须和 [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_UnloadModel](#) 配套使用。

### 【举例】

无。

### 【相关主题】

- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_UnloadModel](#)
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_Predict](#)



## HI\_MPI\_IVE\_ANN\_MLP\_UnloadModel

### 【描述】

去初始化 ANN 模型数据。

### 【语法】

```
HI_VOID HI_MPI_IVE_ANN_MLP_UnloadModel( IVE_ANN_MLP_MODEL_S  
*pstAnnMlpModel)
```

### 【参数】

参数名称	描述	输入/输出
pstAnnMlpModel	模型数据结构体指针。 不能为空。	输入

### 【返回值】

返回值	描述
无	无

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

### 【注意】

该接口必须和 [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_LoadModel](#) 配套使用。

### 【举例】

无。

### 【相关主题】

- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_Predict](#)

## HI\_MPI\_IVE\_ANN\_MLP\_Predict

### 【描述】

创建单个样本 ANN\_MLP 预测任务。

### 【语法】

```
HI_S32 HI_MPI_IVE_ANN_MLP_Predict(IVE_HANDLE *pIveHandle,  
IVE_SRC_MEM_INFO_S *pstSrc, IVE_LOOK_UP_TABLE_S *pstActivFuncTab,
```



```
IVE_ANN_MLP_MODEL_S *pstAnnMlpModel, IVE_DST_MEM_INFO_S *pstDst, HI_BOOL  
bInstant);
```

## 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	输入样本向量（特征向量）指针。 不能为空。	输入
pstActivFuncTab	用于激活函数计算的查找表信息指针。 不能为空。	输入
pstAnnMlpModel	模型数据结构体指针。 不能为空。	输入
pstDst	预测结果向量指针。 不能为空。	输出
bInstant	及时返回结果标志。	输入

参数名称	支持类型	地址对齐	向量维数
pstSrc	一维 SQ16.16 向量，实际截断到 SQ8.16 计算	16 byte	取值范围：1~256； 实际值：pstAnnMlpModel→au16LayerCount[0] 注意内存至少需要 sizeof(SQ16.16) * (pstAnnMlpModel→au16LayerCount[0] + 1)
pstDst	一维 SQ16.16 向量	16 byte	取值范围：1~256； 实际值：pstAnnMlpModel→au16LayerCount[pstAnnMlpModel→u8LayerNum - 1]

注：SQ16.16、SQ8.16 等定点表示说明参看“定点数据类型”。

## 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。



### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持
Hi3536V100	支持
Hi3521AV100	不支持
Hi3518EV200	不支持
Hi3531AV100	支持
Hi3519V100	不支持

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.0.lib）

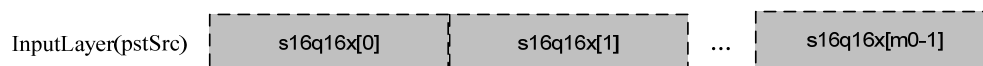
### 【注意】

- 原理与 OpenCV 中 ANN\_MLP 类似。
- 激活函数公式  
Identity 激活函数： $f(u) = u$ 。  
Sigmoid 对称激活函数： $f(u) = \beta(\frac{2}{1 + e^{-\alpha u}} - 1)$ 。  
Gaussian 激活函数： $f(u) = \beta e^{-\alpha u * u}$ 。
- 输入样本向量（输入层）维数、输出预测结果向量（输出层）维数及各隐藏层神经元个数最大为 256。
- pstSrc 至少要分配比输入层维数多 1 维的空间。
- pstActivFuncTab 是用于激活函数  $f(u)$  计算的查找表，其数据均为 S1Q15 类型数据，最多 4096 个；鉴于当前 ANN 中支持的 Identify、Sigmoid、Gaussian 激活函数为奇或偶函数，查找表仅对输入  $u \in [0, \text{pstActivFuncTab} \rightarrow \text{s32TabInUpper}]$  建表及查表；对 ANN 激活函数建立查找表时用于归一化的 u8TabOutNorm 是表示移位的数目。
- 以 u8LayerNum = 4，u8LayerCount[8] = {m0, m1, m2, m3, 0, 0, 0, 0} 为例：
  - 输入样本向量（输入层）为 SQ16.16 的 m0 维向量，实际上仅支持 SQ8.16，超出部分会截断：



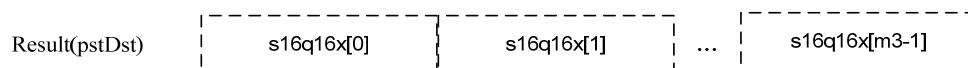


图2-21 ANN\_MLP 输入样本向量示意图



- 输出预测结果向量为 SQ16.16 的 m3 维向量:

图2-22 ANN\_MLP 输出预测结果示意图



【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_UnloadModel](#)

## HI\_MPI\_IVE\_ANN\_MLP\_Predict

【描述】

创建同一模型多个样本 ANN\_MLP 预测任务。

【语法】

```
HI_S32 HI_MPI_IVE_ANN_MLP_Predict(IVE_HANDLE *pIveHandle, IVE_SRC_DATA_S
*pstSrc, IVE_LOOK_UP_TABLE_S *pstActivFuncTab, IVE_ANN_MLP_MODEL_S
*pstAnnMlpModel, IVE_DST_DATA_S *pstDst, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	输入样本向量（特征向量）数组指针。 不能为空。 宽为样本向量维数 * sizeof(HI_S32)。 高为向量个数。 不能为空。	输入
pstActivFuncTab	用于激活函数计算的查找表信息指针。 不能为空。	输入



参数名称	描述	输入/输出
pstAnnMlpModel	模型数据结构体指针。 不能为空。	输入
pstDst	预测结果向量数组指针。 宽为类别数 * sizeof(HI_S32)。 高为向量个数。 不能为空。	输出
bInstant	及时返回结果标志。	输入

参数名称	支持类型	地址对齐	向量维数
pstSrc	一维 SQ16.16 或者 SQ18.14 向量数组，每个元素在计算时实际截断到 SQ8.16 或者 SQ10.14 计算	16 byte	取值范围：1~1024； 实际值：pstAnnMlpModel→au16LayerCount[0]
pstDst	一维 SQ16.16 或者 SQ18.14 向量数组	16 byte	取值范围：1~256； 实际值：pstAnnMlpModel→au16LayerCount[pstAnnMlpModel→u8LayerNum - 1]

注：SQ16.16、SQ8.16 等定点表示说明参看“定点数据类型”。

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	不支持
Hi3536V100	不支持
Hi3521AV100	不支持
Hi3518EV200	不支持
Hi3531AV100	不支持



Hi3519V100	支持
------------	----

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

**【注意】**

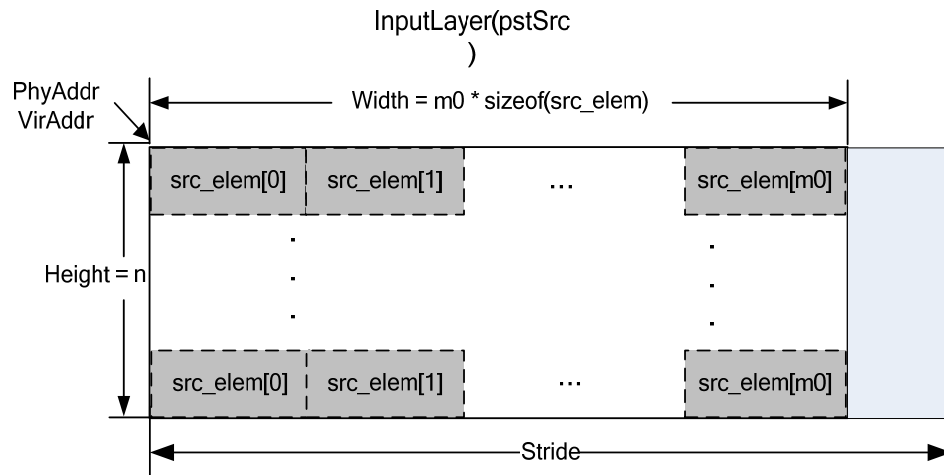
- 原理与 OpenCV 中 ANN\_MLP 类似。
- 激活函数公式

Identity 激活函数： $f(u) = u$ 。

Sigmoid 对称激活函数： $f(u) = \beta\left(\frac{2}{1 + e^{-\alpha u}} - 1\right)$ 。

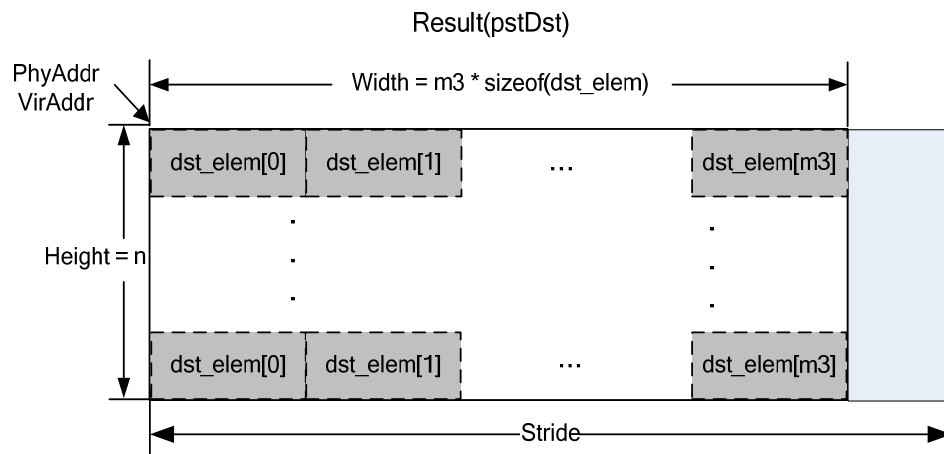
- Gaussian 激活函数： $f(u) = \beta e^{-\alpha u^2}$ 。
- 输入样本向量（输入层）维数最大 1024 维，输出预测结果向量（输出层）维数及各隐藏层神经元个数最大为 256。
- 支持 2 中数据精度类型，参考 IVE\_ANN\_MLP\_ACCURATE\_E。
- pstActivFuncTab 是用于激活函数  $f(u)$  计算的查找表，其数据均为 S1Q15 类型数据，最多 4096 个；鉴于当前 ANN 中支持的 Identify、Sigmoid、Gaussian 激活函数为奇或偶函数，查找表仅对输入  $u \in [0, \text{pstActivFuncTab} \rightarrow \text{s32TabInUpper}]$  建表及查表；对 ANN 激活函数建立查找表时用于归一化的 u8TabOutNorm 是表示移位的数目。
- 以 u8LayerNum = 4，u8LayerCount[8] = {m0, m1, m2, m3, 0, 0, 0, 0}，样本数量 = n，为例：
  - 输入 n 个样本向量（输入层），每个向量均是包含 m0 个 src\_elem 类型为 SQ16.16 或者 SQ18.14 的向量，实际计算时每个 elem 截断到 SQ8.16 或者 SQ10.14：

图2-23 ANN\_MLP 输入样本向量数组示意图



- 输出  $n$  个预测结果向量，每个向量均是包含  $m3$  个 `dst_elem` 类型为 SQ16.16 或者 SQ18.14 的向量：

图2-24 ANN\_MLP 输出预测结果示意图



【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_UnloadModel](#)

## HI\_MPI\_IVE\_SVM\_LoadModel

【描述】



读取 SVM 模型文件，初始化模型数据。

【语法】

```
HI_S32 HI_MPI_IVE_SVM_LoadModel(const HI_CHAR *pchFileName,  
IVE_SVM_MODEL_S *pstSvmModel);
```

【参数】

参数名称	描述	输入/输出
pchFileName	模型文件路径及文件名。 不能为空。	输入
pstSvmModel	模型数据结构体指针。 不能为空。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

【注意】

- 文件名必须以.bin 为后缀；.bin 文件必须用海思提供的工具 ive\_tool\_xml2bin.exe 生成。
- 该接口必须和 [HI\\_MPI\\_IVE\\_SVM\\_UnloadModel](#) 配套使用。

【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_SVM\\_UnloadModel](#)
- [HI\\_MPI\\_IVE\\_SVM\\_Predict](#)

## HI\_MPI\_IVE\_SVM\_UnloadModel

【描述】

去初始化 SVM 模型数据。



#### 【语法】

```
HI_VOID HI_MPI_IVE_SVM_UnloadModel(IVE_SVM_MODEL_S *pstSvmModel);
```

#### 【参数】

参数名称	描述	输入/输出
pstSvmModel	模型数据结构体指针。 不能为空。	输入

#### 【返回值】

返回值	描述
无	无

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.0.lib）

#### 【注意】

该接口必须和 [HI\\_MPI\\_IVE\\_SVM\\_LoadModel](#) 配套使用。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_SVM\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_SVM\\_Predict](#)

## HI\_MPI\_IVE\_SVM\_Predict

#### 【描述】

创建单个样本 SVM 预测任务。

#### 【语法】

```
HI_S32 HI_MPI_IVE_SVM_Predict(IVE_HANDLE *pIveHandle, IVE_SRC_MEM_INFO_S  
*pstSrc, IVE_LOOK_UP_TABLE_S *pstKernelTab, IVE_SVM_MODEL_S *pstSvmModel,  
IVE_DST_MEM_INFO_S *pstDstVote, HI_BOOL bInstant);
```

#### 【参数】



参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	输入样本向量（特征向量）指针。 不能为空。	输入
pstKernelTab	用于核函数计算的查找表信息指针。 不能为空。	输入
pstSvmModel	模型数据结构体指针。 不能为空。	输入
pstDstVote	“1-VS-1 SVM”各个类别的投票数向量指针。 不能为空。	输出
bInstant	及时返回结果标志。	输入

参数名称	支持类型	地址对齐	向量维数
pstSrc	一维 SQ16.16 向量，实际截断到 SQ8.16 计算	16 byte	取值范围：1~256； 实际值：pstSvmModel→u16FeatureDim
pstDstVote	一维 HI_U16 向量	16 byte	取值范围：1~80； 实际值：pstSvmModel→u8ClassCount

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	支持
Hi3536V100	支持
Hi3521AV100	不支持



Hi3518EV200	不支持
Hi3531AV100	支持
Hi3519V100	不支持

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.0.lib）

#### 【注意】

- 原理与 OpenCV 中 SVM\_Predict 类似。
- 核函数计算公式

线性核函数：  $K(x_i, x_j) = x_i^T x_j$

多项式核函数：  $K(x_i, x_j) = (\mathcal{X}_i^T x_j + coef0)^{deg_{ree}}, \gamma > 0$

径向基核函数：  $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \gamma > 0$

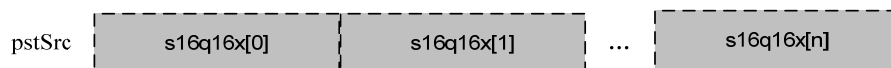
Sigmoid 核函数：  $K(x_i, x_j) = \tanh(\mathcal{X}_i^T x_j + coef0)$

- 判决函数计算公式

$$\text{sgn}(\omega^T \phi(x) + b) = \text{sgn}\left(\sum_{i=1}^l (y_i \alpha_i K(x_i, x)) + b\right)$$

- pstKernelTab 是用于核函数  $K(x_i, x_j)$  计算的查找表，其数据均为 S1Q15 类型数据，最多 2048 个；对 SVM 核函数建立查找表时，查找表的输入是  $x_i^T x_j$  或者  $\|x_i - x_j\|^2$ ，8TabOutNorm 可以表示除法的除数（不能为 0，SvmDivisor = u8TabOutNorm）或者移位的数目（可以为 0，此时 SvmDivisor = 1 << u8TabOutNorm），同样使用工具 ive\_tool\_xml2bin.exe 时需要将 SvmDivisor 作为参数传入，SvmDivisor 见 ive\_tool\_xml2bin.exe 的使用说明。
- 以 u16FeatureDim = n，u8ClassCount = N 为例：
  - 输入样本向量是 SQ16.16 的 n 维向量（最大 256 维），实际上仅支持 SQ8.16，超出部分会截断：

图2-25 SVM 输入样本向量示意图

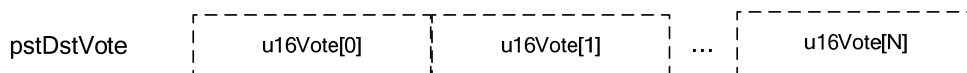


- 输出预测结果向量为 HI\_U16 类型的 N 维向量：





图2-26 SVM 预测结果示意图



【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_SVM\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_SVM\\_UnloadModel](#)

## HI\_MPI\_IVE\_SVM\_Predict

【描述】

创建同一模型的多个样本 SVM 预测任务。

【语法】

```
HI_S32 HI_MPI_IVE_SVM_Predict(IVE_HANDLE *pIveHandle, IVE_SRC_DATA_S
*pstSrc, IVE_LOOK_UP_TABLE_S *pstKernelTab, IVE_SVM_MODEL_S *pstSvmModel,
IVE_DST_DATA_S *pstDstVote, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
pstSrc	输入样本向量（特征向量）数组指针。 宽为样本向量维数 * sizeof(HI_S16Q16)。 高为向量个数。 不能为空。	输入
pstKernelTab	用于核函数计算的查找表信息指针。 不能为空。	输入
pstSvmModel	模型数据结构体指针。 不能为空。	输入
pstDstVote	“1-VS-1 SVM”各个类别的投票数向量数组指针。 宽为投票类别数 * sizeof(HI_U16)。 高为向量个数。 不能为空。	输出
bInstant	及时返回结果标志。	输入



参数名称	支持类型	地址对齐	向量维数
pstSrc	一维 SQ16.16 向量数组，每个元素用于计算时截断到 SQ8.16 计算	16 byte	取值范围：1~1024； 实际值：pstSvmModel→u16FeatureDim
pstDstVote	一维 HI_U16 向量数组	16 byte	取值范围：1~80； 实际值：pstSvmModel→u8ClassCount

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	不支持
Hi3536V100	不支持
Hi3521AV100	不支持
Hi3518EV200	不支持
Hi3531AV100	不支持
Hi3519V100	支持

#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

#### 【注意】

- 原理与 OpenCV 中 SVM\_Predict 类似。
- 核函数计算公式

线性核函数： $K(x_i, x_j) = x_i^T x_j$

多项式核函数： $K(x_i, x_j) = (\mathcal{K}_i^T x_j + coef0)^{degree}, \gamma > 0$



径向基核函数:  $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \gamma > 0$

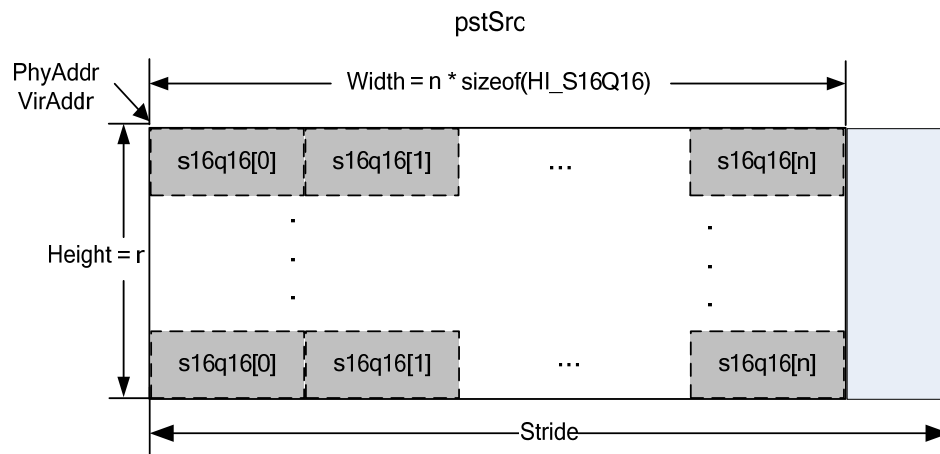
Sigmoid 核函数:  $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + coef0)$

- 判决函数计算公式

$$\text{sgn}(\omega^T \phi(x) + b) = \text{sgn}(\sum_{i=1}^l (y_i \alpha_i K(x_i, x)) + b)$$

- pstKernelTab 是用于核函数  $K(x_i, x_j)$  计算的查找表，其数据均为 S1Q15 类型数据，最多 2048 个；对 SVM 核函数建立查找表时，查找表的输入是  $x_i^T x_j$  或者  $\|x_i - x_j\|^2$ ，8TabOutNorm 可以表示除法的除数（不能为 0，SvmDivisor = u8TabOutNorm）或者移位的数目（可以为 0，此时 SvmDivisor = 1 << u8TabOutNorm），同样使用工具 ive\_tool\_xml2bin.exe 时需要将 SvmDivisor 作为参数传入，SvmDivisor 见 ive\_tool\_xml2bin.exe 的使用说明。
- 以 u16FeatureDim = n，u8ClassCount = N，样本数量 = r，为例：
  - r 个输入样本向量，每个都是 SQ16.16 类型的 n 维向量（最大 1024 维），实际上仅支持 SQ8.16，超出部分会截断：

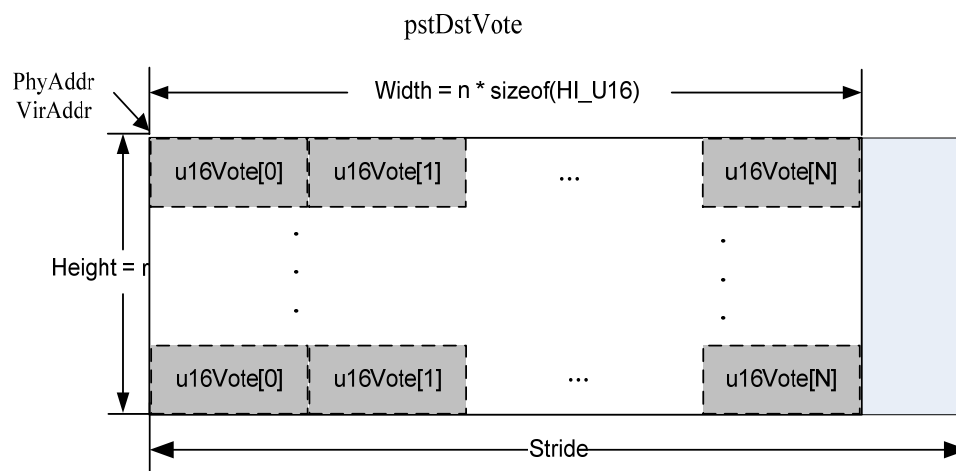
图2-27 SVM 输入样本向量数组示意图



- 输出 r 个预测结果向量，每个均为 HI\_U16 类型的 N 维向量：



图2-28 SVM 预测结果示意图



【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_SVM\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_SVM\\_UnloadModel](#)

## HI\_MPI\_IVE\_CNN\_LoadModel

【描述】

读取 CNN 模型文件，初始化 CNN 模型数据。

【语法】

```
HI_S32 HI_MPI_IVE_CNN_LoadModel(const HI_CHAR *pchFileName,  
IVE_CNN_MODEL_S *pstCnnModel);
```

【参数】

参数名称	描述	输入/输出
<code>pchFileName</code>	模型文件路径及文件名。 不能为空。	输入
<code>pstCnnModel</code>	CNN 网络模型结构体指针 不能为空。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

**【注意】**

- 文件名必须以.bin 为后缀；.bin 文件必须用海思提供的工具 ive\_tool\_cnn.exe 生成。
- 该接口必须和 [HI\\_MPI\\_IVE\\_CNN\\_UnloadModel](#) 配套使用。

**【举例】**

无。

**【相关主题】**

- [HI\\_MPI\\_IVE\\_CNN\\_UnloadModel](#)
- [HI\\_MPI\\_IVE\\_CNN\\_Predict](#)
- [HI\\_MPI\\_IVE\\_CNN\\_GetResult](#)

## HI\_MPI\_IVE\_CNN\_UnloadModel

**【描述】**

去初始化 CNN 模型数据。

**【语法】**

```
HI_VOID HI_MPI_IVE_CNN_UnloadModel(IVE\_CNN\_MODEL\_S *pstCnnModel);
```

**【参数】**

参数名称	描述	输入/输出
pstCnnModel	CNN 网络模型结构体指针。 不能为空。	输入

**【返回值】**

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。



#### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

#### 【注意】

该接口必须和 [HI\\_MPI\\_IVE\\_CNN\\_LoadModel](#) 配套使用。

#### 【举例】

无。

#### 【相关主题】

- [HI\\_MPI\\_IVE\\_CNN\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_CNN\\_Predict](#)
- [HI\\_MPI\\_IVE\\_CNN\\_GetResult](#)

## HI\_MPI\_IVE\_CNN\_Predict

#### 【描述】

创建一个 CNN 模型的单个或多个样本预测任务，并输出特征向量。

#### 【语法】

```
HI_S32 HI_MPI_IVE_CNN_Predict(IVE_HANDLE *pIveHandle, IVE_SRC_IMAGE_S  
astSrc[], IVE_CNN_MODEL_S *pstCnnModel, IVE_DST_DATA_S *pstDst,  
IVE_CNN_CTRL_S *pstCnnCtrl, HI_BOOL bInstant);
```

#### 【参数】

参数名称	描述	输入/输出
pIveHandle	handle 指针。 不能为空。	输出
astSrc[]	输入样本图像数组。最多 64 张样本图像。 不能为空。	输入
pstCnnModel	CNN 模型结构体指针。 不能为空。	输入
pstDst	特征向量数组指针，存放 CNN 全连接最后一层结果。 不能为空。	输出
pstCnnCtrl	控制参数指针。 pstCnnCtrl->stMem 内存分配见【注意】。 不能为空。	输入



参数名称	描述	输入/输出
bInstant	及时返回结果标志	输入

参数名称	支持类型	地址对齐	分辨率
astSrc[]	U8C1、 U8C3_PLANAR	16 byte	宽 w: 16~80; 高 h: 16~1280/w

参数名称	向量个数	地址对齐	向量描述
pstDst	取值范围: 1~64 实际值: pstDst->u16Height	16 byte	维数取值范围: 1~256; 维数实际值: pstCnnModel->stFullConnect.au16LayerCnt [pstCnnModel->stFullConnect.u8LayerNum-1] 元素类型: SQ18.14

注: SQ18.14 定点表示说明参看“定点数据类型”。

#### 【返回值】

返回值	描述
0	成功。
非 0	失败, 参见 <a href="#">错误码</a> 。

#### 【芯片差异】

芯片类型	差异
Hi3516AV100	不支持
Hi3536V100	不支持
Hi3521AV100	不支持
Hi3518EV200	不支持
Hi3531AV100	不支持
Hi3519V100	支持

#### 【需求】

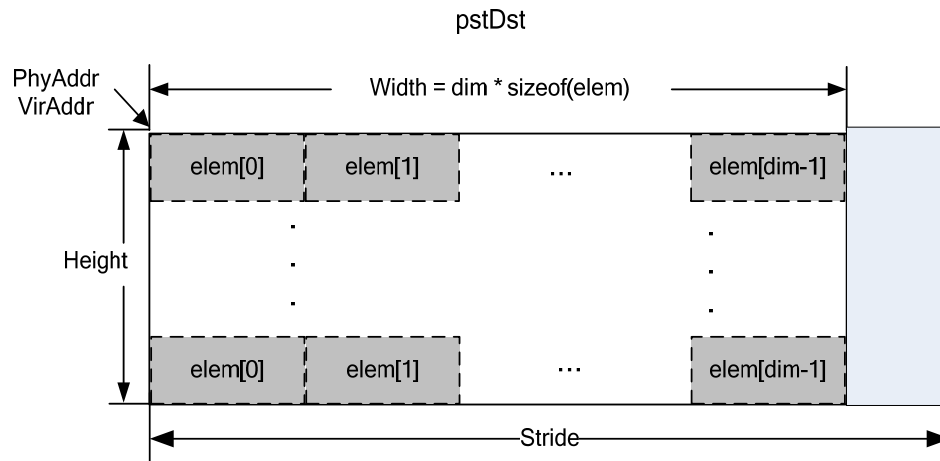
- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

#### 【注意】

- pstCnnCtrl->stMem 内存分配至少需分配：  

$$\text{IveAlign}(m * \text{pstCnnCtrl} \rightarrow \text{u32Num} * \text{sizeof}(\text{HI\_U32}), 16) + \text{IveAlign}(\text{pstCnnModel} \rightarrow \text{stFullConnect.au16LayerCnt}[0] * \text{sizeof}(\text{HI\_U32}), 16) * \text{pstCnnCtrl} \rightarrow \text{u32Num},$$
 其中，pstCnnModel->enType 为 U8C1 时，m=1;为 U8C3\_PLANAR 时，m=3。
- 样本数组 astSrc[] 的图像类型、宽、高必须与 CNN 网络模型中 pstCnnModel 中的 enType、u16Width、u16Height 一致，数组元素个数为 pstCnnCtrl->u32Num。
- 特征向量数组 pstDst 的向量数目 pstDst->u16Height 与图像数目 pstCnnCtrl->u32Num 相等。
  - 输出的特征向量内存如图 2-29 所示，每个向量维数  $\text{dim} = \text{pstCnnModel} \rightarrow \text{stFullConnect.au16LayerCnt}[\text{pstCnnModel} \rightarrow \text{stFullConnect.u8LayerNum}-1]$ ，向量的元素 elem 类型为 SQ18.14，向量个数  $\text{Height} = \text{pstCnnCtrl} \rightarrow \text{u32Num}$ 。

图2-29 CNN 输出特征向量数组示意图



- 该接口和 [HI\\_MPI\\_IVE\\_CNN\\_GetResult](#) 配套使用，特征向量数组 pstDst 是 [HI\\_MPI\\_IVE\\_CNN\\_GetResult](#) 的输入。
- CNN 网络模型支持最多 8 层 Conv-ReLU-Pooling 和 8 层全连接层；Conv-ReLU-Pooling 层的卷积核仅支持 3x3，ReLU(Rectified Linear Units)和 Pooling 可配（见 [IVE\\_CNN\\_ACTIV\\_FUNC\\_E](#) 和 [IVE\\_CNN\\_POOLING\\_E](#)），每层 Conv-ReLU-Pooling 最多输出 50 张 feature map；全连接层仅支持 ReLU 激活函数，层数范围[3, 8]：全连接输入层（即 Conv-ReLU-Pooling 的最终输出）维数[1, 1024]，中间隐藏层神经元数目[2,256]，输出层维数[1, 256]。具体参数配置参见下列表格：





表2-1 CNN 模型中单层 Conv-ReLU-Pooling 运算包参数配置表

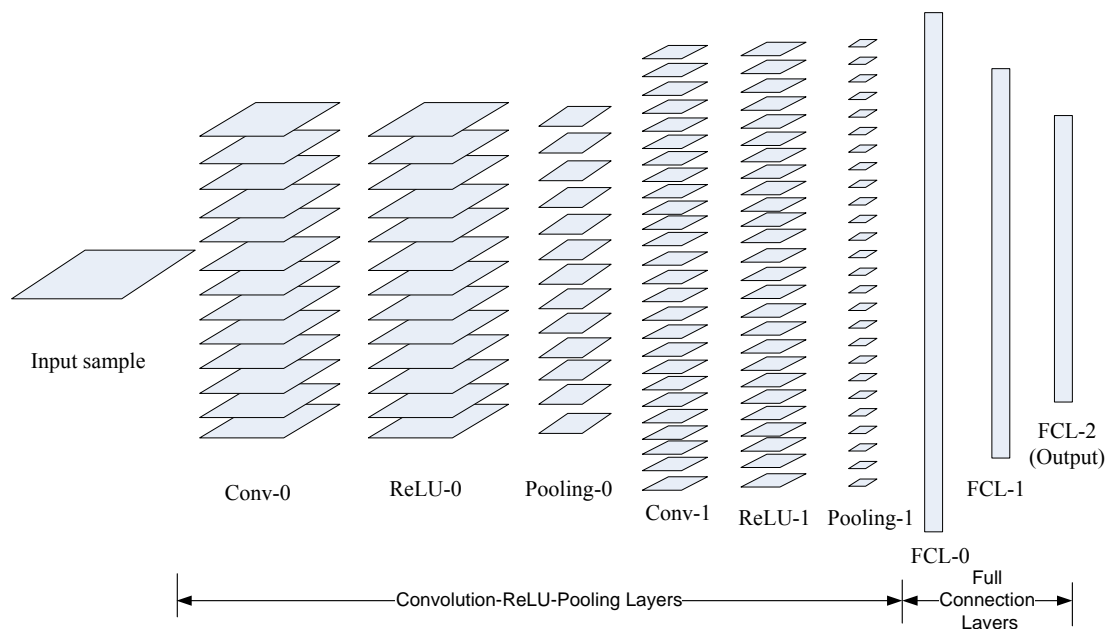
Conv-ReLU-Pooling 运算包	模式	数量	大小	步长	边界填充
Convolution	-	1-50	3x3	1	-
Activation	None\ReLU	-	-	-	-
Pooling	None\Max\Average	-	2x2	2	向上取偶数对齐，复制边界。

表2-2 CNN 模型中全连接运算包参数配置表

全连接层运算包	层数 (含输入层)	输入层 维数	中间隐藏层节点数	输出层 维数	隐藏层激活 函数
	3-8	1-1024	2-256	1-256	ReLU

- 以单个输入样本， $n+1$  ( $1 \leq n+1 \leq 8$ ) 层 Conv-ReLU-Pooling 和  $m+1$  ( $3 \leq m+1 \leq 8$ ) 层全连接层为例，CNN 网络模型如图 2-30 所示。注意图示中，FCL-0 是 Pooling-n 各图像数据拉成的列向量，实际计算过程中 Pooling-n 的结果会直接以 FCL-0 的形式输出。

图2-30 CNN 网络模型示意图





### 【举例】

无。

### 【相关主题】

- [HI\\_MPI\\_IVE\\_CNN\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_CNN\\_UnloadModel](#)
- [HI\\_MPI\\_IVE\\_CNN\\_GetResult](#)

## HI\_MPI\_IVE\_CNN\_GetResult

### 【描述】

接收 CNN\_Predict 结果，执行 Softmax 运算来预测每个样本图像类别，并输出置信度。

### 【语法】

```
HI_S32 HI_MPI_IVE_CNN_GetResult(IVE_SRC_DATA_S *pstSrc,  
IVE_DST_MEM_INFO_S *pstDst, IVE_CNN_MODEL_S *pstCnnModel, IVE_CNN_CTRL_S  
*pstCnnCtrl);
```

### 【参数】

参数名称	描述	输入/输出
pstSrc	源数据指针。源数据为 <a href="#">HI_MPI_IVE_CNN_Predict</a> 的输出。不能为空。	输入
pstDst	预测结果结构体指针，指向 IVE_CNN_RESULT_S 的数组，表示各个样本的类别和置信度。 不能为空。	输出
pstCnnModel	CNN 模型结构体指针。 不能为空。	输入
pstCnnCtrl	控制参数指针。不能为空。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h

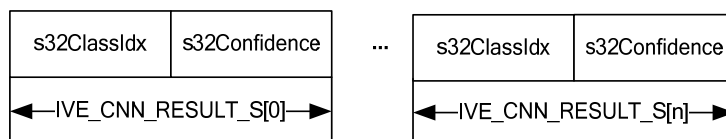


- 库文件：libive.a（PC 上模拟用 ive\_clib2.1.lib）

【注意】

- 源数据 pstSrc 必须为 [HI\\_MPI\\_IVE\\_CNN\\_Predict](#) 的输出，pstCnnModel 和 pstCnnCtrl 必须与调用 [HI\\_MPI\\_IVE\\_CNN\\_Predict](#) 时的参数一致。
- 预测结果 pstDst 指向 [IVE\\_CNN\\_RESULT\\_S](#) 的数组，数组元素数目  $n = \text{pstCnnCtrl} \rightarrow \text{u32Num}$ ，其内存排布如[图 2-31](#)所示。

图2-31 CNN 各样本预测结果示意图



【举例】

无。

【相关主题】

- [HI\\_MPI\\_IVE\\_CNN\\_LoadModel](#)
- [HI\\_MPI\\_IVE\\_CNN\\_UnloadModel](#)
- [HI\\_MPI\\_IVE\\_CNN\\_Predict](#)

## HI\_MPI\_IVE\_Query

【描述】

查询已创建任务完成情况。

【语法】

```
HI_S32 HI_MPI_IVE_Query(IVE_HANDLE IveHandle, HI_BOOL *pbFinish, HI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
IveHandle	任务的 handle。	输入
pbFinish	任务完成状态指针。 不能为空。	输出
bBlock	是否阻塞查询标志。	输入



### 【返回值】

返回值	描述
0	成功。
非 0	失败，参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_ive.h、hi\_ive.h、mpi\_ive.h
- 库文件：libive.a（PC 上模拟用 ive\_clib2.x.lib）

### 【注意】

- 在用户使用 IVE 任务结果前，为确保 IVE 任务已完成，用户可以使用阻塞方式调用此接口查询。
- IVE 内部是按任务创建顺序依次执行任务的，所以用户不必每次都使用查询接口，如用户依次创建了 A，B 两个任务，那么如果 B 任务完成了，这个时候 A 任务肯定也完成了，此时使用 A 任务的结果时不必再次调用查询接口。
- 返回值为 HI\_ERR\_IVE\_QUERY\_TIMEOUT（查询超时）时，可以继续查询。
- 返回值为 HI\_ERR\_IVE\_SYS\_TIMEOUT（系统超时）时，用户的 IVE 任务必须全部重新提交。

### 【举例】

```
HI_S32 s32Ret = HI_SUCCESS;
IVE_HANDLE IveHandle;
IVE_SRC_DATA_S stSrc;
IVE_DST_DATA_S stDst;
IVE_DMA_CTRL_S stDmaCtrl = { IVE_DMA_MODE_DIRECT_COPY, 0};
HI_BOOL bInstant;
HI_BOOL bFinish, bBlock;

stSrc.u32PhyAddr      = 0;
stSrc.pu8VirAddr      = HI_NULL;
stSrc.ul6Stride       = 352;
stSrc.ul6Height       = 288;
stSrc.ul6Width        = 352;

stDst.u32PhyAddr      = 0;
stDst.pu8VirAddr      = HI_NULL;
stDst.ul6Stride       = 352;
stDst.ul6Height       = 288;
stDst.ul6Width        = 352;

bInstant              = HI_TRUE;
```



```
s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&stSrc.u32PhyAddr,
&stSrc.pu8VirAddr, "User", HI_NULL, stSrc.ul6Height*stSrc.ul6Stride);
if(HI_SUCCESS!=s32Ret)
{
return s32Ret;
}
memset(stSrc.pu8VirAddr, 1, stSrc.ul6Height * stSrc.ul6Stride);

s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&stDst.u32PhyAddr, &stDst.pu8VirAddr,
"User", HI_NULL, stDst.ul6Height*stDst.ul6Stride);
if(HI_SUCCESS!=s32Ret)
{
HI_MPI_SYS_MmzFree(stSrc.u32PhyAddr, stSrc.pu8VirAddr);
return s32Ret;
}
memset(stDst.pu8VirAddr,0, stDst.ul6Height * stDst.ul6Stride);
s32Ret = HI_MPI_SYS_MmzFlushCache(0, NULL, 0);
if(HI_SUCCESS!=s32Ret)
{
HI_MPI_SYS_MmzFree(stSrc.u32PhyAddr, stSrc.pu8VirAddr);
HI_MPI_SYS_MmzFree(stDst.u32PhyAddr, stDst.pu8VirAddr);
return s32Ret;
}
s32Ret = HI_MPI_IVE_DMA(&IveHandle, &stSrc, &stDst, &stDmaCtrl,
bInstant);
if(HI_SUCCESS!=s32Ret)
{
HI_MPI_SYS_MmzFree(stSrc.u32PhyAddr, stSrc.pu8VirAddr);
HI_MPI_SYS_MmzFree(stDst.u32PhyAddr, stDst.pu8VirAddr);
return s32Ret;
}
bBlock = HI_FALSE;
s32Ret = HI_MPI_IVE_Query(IveHandle, &bFinish, bBlock);
if (SUCCESS == s32Ret)
{
printf("bFinish=%d\n", bFinish);
}
HI_MPI_SYS_MmzFree(stSrc.u32PhyAddr, stSrc.pu8VirAddr);
HI_MPI_SYS_MmzFree(stDst.u32PhyAddr, stDst.pu8VirAddr);
return s32Ret;
```

#### 【相关主题】

无。



# 3 数据类型和数据结构

IVE 相关数据类型、数据结构定义如下：

- [IVE\\_HIST\\_NUM](#)：定义直方图统计 bin 数目。
- [IVE\\_MAP\\_NUM](#)：定义映射查找表项数目。
- [IVE\\_MAX\\_REGION\\_NUM](#)：定义最大连通区域数目。
- [IVE\\_ST\\_MAX\\_CORNER\\_NUM](#)：定义 Shi-Tomasi-like 角点最大数目。
- [IVE\\_IMAGE\\_TYPE\\_E](#)：定义二维广义图像支持的图像类型。
- [IVE\\_IMAGE\\_S](#)：定义二维广义图像信息。
- [IVE\\_SRC\\_IMAGE\\_S](#)：定义源图像。
- [IVE\\_DST\\_IMAGE\\_S](#)：定义输出图像。
- [IVE\\_DATA\\_S](#)：定义以 byte 为单位的二维图像信息。
- [IVE\\_SRC\\_DATA\\_S](#)：定义以 byte 为单位的二维源数据信息。
- [IVE\\_DST\\_DATA\\_S](#)：定义 byte 为单位的二维输出数据信息。
- [IVE\\_MEM\\_INFO\\_S](#)：定义一维数据内存信息。
- [IVE\\_SRC\\_MEM\\_INFO\\_S](#)：定义一维源数据。
- [IVE\\_DST\\_MEM\\_INFO\\_S](#)：定义一维输出数据。
- [IVE\\_8BIT\\_U](#)：定义 8bit 数据共用体。
- [IVE\\_POINT\\_U16\\_S](#)：定义 U16 表示的点信息结构体。
- [IVE\\_POINT\\_S25Q7\\_S](#)：定义 S25Q7 定点表示的点信息结构体。
- [IVE\\_RECT\\_S](#)：定义 U16 表示的矩形信息结构体
- [IVE\\_DMA\\_MODE\\_E](#)：定义 DMA 运算模式。
- [IVE\\_DMA\\_CTRL\\_S](#)：定义 DMA 控制信息。
- [IVE\\_FILTER\\_CTRL\\_S](#)：定义模板滤波控制信息
- [IVE\\_CSC\\_MODE\\_E](#)：定义色彩空间转换模式。
- [IVE\\_CSC\\_CTRL\\_S](#)：定义色彩空间转换控制信息。
- [IVE\\_FILTER\\_AND\\_CSC\\_CTRL\\_S](#)：定义模板滤波加色彩空间转换复合功能控制信息。
- [IVE\\_SOBEL\\_OUT\\_CTRL\\_E](#)：定义 sobel 输出控制信息。



- [IVE\\_SOBEL\\_CTRL\\_S](#): 定义 sobel 边缘提取控制信息。
- [IVE\\_MAG\\_AND\\_ANG\\_OUT\\_CTRL\\_E](#): 定义 canny 边缘幅值与角度计算的输出格式。
- [IVE\\_MAG\\_AND\\_ANG\\_CTRL\\_S](#): 定义 canny 边缘幅值和幅角计算的控制信息。
- [IVE\\_DILATE\\_CTRL\\_S](#): 定义膨胀控制信息。
- [IVE\\_ERODE\\_CTRL\\_S](#): 定义腐蚀控制信息。
- [IVE\\_THRESH\\_MODE\\_E](#): 定义图像二值化输出格式。
- [IVE\\_THRESH\\_CTRL\\_S](#): 定义图像二值化控制信息。
- [IVE\\_SUB\\_MODE\\_E](#): 定义两图像相减输出格式。
- [IVE\\_SUB\\_CTRL\\_S](#): 定义两图像相减控制参数。
- [IVE\\_INTEG\\_OUT\\_CTRL\\_E](#): 定义积分图输出控制参数。
- [IVE\\_INTEG\\_CTRL\\_S](#): 定义积分图计算控制参数。
- [IVE\\_THRESH\\_S16\\_MODE\\_E](#): 定义 16bit 有符号图像的阈值化模式。
- [IVE\\_THRESH\\_S16\\_CTRL\\_S](#): 定义 16bit 有符号图像的阈值化控制参数。
- [IVE\\_THRESH\\_U16\\_MODE\\_E](#): 定义 16bit 无符号图像的阈值化模式。
- [IVE\\_THRESH\\_U16\\_CTRL\\_S](#): 定义 16bit 无符号图像的阈值化控制参数。
- [IVE\\_16BIT\\_TO\\_8BIT\\_MODE\\_E](#): 定义 16bit 图像到 8bit 图像的转化模式。
- [IVE\\_16BIT\\_TO\\_8BIT\\_CTRL\\_S](#): 定义 16bit 图像到 8bit 图像的转化控制参数。
- [IVE\\_ORD\\_STAT\\_FILTER\\_MODE\\_E](#): 定义顺序统计量滤波模式。
- [IVE\\_ORD\\_STAT\\_FILTER\\_CTRL\\_S](#): 定义顺序统计量滤波控制参数。
- [IVE\\_MAP\\_LUT\\_MEM\\_S](#): 定义 Map 算子的查找表内存信息。
- [IVE\\_MAP\\_U8BIT\\_LUT\\_MEM\\_S](#): 定义 Map U8C1→U8C1 的查找表内存。
- [IVE\\_MAP\\_U16BIT\\_LUT\\_MEM\\_S](#): 定义 Map U8C1→U16C1 的查找表内存。
- [IVE\\_MAP\\_S16BIT\\_LUT\\_MEM\\_S](#): 定义 Map U8C1→S16C1 的查找表内存。
- [IVE\\_MAP\\_MODE\\_E](#): 定义 Map 的模式。
- [IVE\\_MAP\\_CTRL\\_S](#): 定义 Map 控制参数。
- [IVE\\_EQUALIZE\\_HIST\\_CTRL\\_MEM\\_S](#): 定义直方图均衡化辅助内存。
- [IVE\\_EQUALIZE\\_HIST\\_CTRL\\_S](#): 定义直方图均衡化控制参数。
- [IVE\\_ADD\\_CTRL\\_S](#): 定义两图像的加权加控制参数。
- [IVE\\_NCC\\_DST\\_MEM\\_S](#): 定义 NCC 的输出内存信息。
- [IVE\\_REGION\\_S](#): 定义连通区域信息。
- [IVE\\_CCBLOB\\_S](#): 定义连通区域标记的输出信息。
- [IVE\\_CCL\\_MODE\\_E](#): 定义连通区域模式。
- [IVE\\_CCL\\_CTRL\\_S](#): 定义连通区域标记控制参数。
- [IVE\\_GMM\\_CTRL\\_S](#): 定义 GMM 背景建模的控制参数。
- [IVE\\_GMM2\\_SNS\\_FACTOR\\_MODE\\_E](#): 定义灵敏度系数模式。
- [IVE\\_GMM2\\_LIFE\\_UPDATE\\_FACTOR\\_MODE\\_E](#): 定义模型时长参数更新模式。
- [IVE\\_GMM2\\_CTRL\\_S](#): 定义 GMM2 背景建模的控制参数。



- [IVE\\_CANNY\\_STACK\\_SIZE\\_S](#): 定义 Canny 边缘前半部分计算时强边缘点栈大小结构体。
- [IVE\\_CANNY\\_HYS\\_EDGE\\_CTRL\\_S](#): 定义 Canny 边缘前半部分计算任务的控制参数。
- [IVE\\_LBP\\_CMP\\_MODE\\_E](#): 定义 LBP 纹理计算控制参数。
- [IVE\\_LBP\\_CTRL\\_S](#): 定义 LBP 纹理计算控制参数。
- [IVE\\_NORM\\_GRAD\\_OUT\\_CTRL\\_E](#): 定义归一化梯度信息计算任务输出控制枚举类型。
- [IVE\\_NORM\\_GRAD\\_CTRL\\_S](#): 定义归一化梯度信息计算控制参数。
- [IVE\\_MV\\_S9Q7\\_S](#): 定义位移结构体。
- [IVE\\_LK\\_OPTICAL\\_FLOW\\_CTRL\\_S](#): 定义 LK 光流计算控制参数。
- [IVE\\_LK\\_OPTICAL\\_FLOW\\_PYR\\_OUT\\_MODE\\_E](#): 定义金字塔 LK 光流计算输出模式。
- [IVE\\_LK\\_OPTICAL\\_FLOW\\_PYR\\_CTRL\\_S](#): 定义金字塔 LK 光流计算控制参数。
- [IVE\\_ST\\_MAX\\_EIG\\_S](#): 定义 Shi-Tomas-like 角点计算时最大角点响应值结构体。
- [IVE\\_ST\\_CANDI\\_CORNER\\_CTRL\\_S](#): 定义 Shi-Tomas-like 候选角点计算控制参数。
- [IVE\\_ST\\_CORNER\\_INFO\\_S](#): 定义 Shi-Tomas-like 角点计算输出的角点信息结构体。
- [IVE\\_ST\\_CORNER\\_CTRL\\_S](#): 定义 Shi-Tomas-like 角点筛选控制参数。
- [IVE\\_SAD\\_MODE\\_E](#): 定义 SAD 计算模式。
- [IVE\\_SAD\\_OUT\\_CTRL\\_E](#): 定义 SAD 输出控制模式。
- [IVE\\_SAD\\_CTRL\\_S](#): 定义 SAD 控制参数。
- [IVE\\_RESIZE\\_MODE\\_E](#): 定义 Resize 的模式。
- [IVE\\_RESIZE\\_CTRL\\_S](#): 定义 Resize 控制参数。
- [IVE\\_GRAD\\_FG\\_MODE\\_E](#): 定义梯度前景计算模式。
- [IVE\\_GRAD\\_FG\\_CTRL\\_S](#): 定义计算梯度前景控制参数。
- [IVE\\_CANDI\\_BG\\_PIX\\_S](#): 定义候选背景模型数据。
- [IVE\\_WORK\\_BG\\_PIX\\_S](#): 定义工作背景模型数据。
- [IVE\\_BG\\_LIFE\\_S](#): 定义背景生命力数据。
- [IVE\\_BG\\_MODEL\\_PIX\\_S](#): 定义背景模型数据。
- [IVE\\_FG\\_STAT\\_DATA\\_S](#): 定义前景状态数据。
- [IVE\\_BG\\_STAT\\_DATA\\_S](#): 定义背景状态数据。
- [IVE\\_MATCH\\_BG\\_MODEL\\_CTRL\\_S](#): 定义背景匹配控制参数。
- [IVE\\_UPDATE\\_BG\\_MODEL\\_CTRL\\_S](#): 定义背景更新控制参数。
- [IVE\\_LOOK\\_UP\\_TABLE\\_S](#): 定义查找表结构体。
- [IVE\\_ANN\\_MLP\\_ACCURATE\\_E](#): 定义 ANN\_MLP 输入特征向量类型。
- [IVE\\_ANN\\_MLP\\_ACTIV\\_FUNC\\_E](#): 定义 ANN\_MLP 激活函数枚举类型。
- [IVE\\_ANN\\_MLP\\_MODEL\\_S](#): 定义 ANN\_MLP 模型数据结构体。





- [IVE\\_SVM\\_TYPE\\_E](#): 定义 SVM 类型。
- [IVE\\_SVM\\_KERNEL\\_TYPE\\_E](#): 定义 SVM 核函数类型。
- [IVE\\_SVM\\_MODEL\\_S](#): 定义 SVM 模型数据结构体。
- [IVE\\_CNN\\_ACTIV\\_FUNC\\_E](#): 定义 CNN 激活函数枚举类型。
- [IVE\\_CNN\\_POOLING\\_E](#): 定义 CNN 汇聚操作枚举类型。
- [IVE\\_CNN\\_CONV\\_POOLING\\_S](#): 定义 CNN 单层 Conv-ReLU-Pooling 运算包参数结构体。
- [IVE\\_CNN\\_FULL\\_CONNECT\\_S](#): 定义 CNN 全链接网络参数结构体。
- [IVE\\_CNN\\_MODEL\\_S](#): 定义 CNN 模型参数结构体。
- [IVE\\_CNN\\_CTRL\\_S](#): 定义 CNN 预测任务的控制参数。
- [IVE\\_CNN\\_RESULT\\_S](#): 定义 CNN 单个样本预测结果结构体。

## 定点数据类型

### 【说明】

定义定点化的数据类型。

### 【定义】

```
typedef unsigned char      HI_U0Q8;
typedef unsigned char      HI_U1Q7;
typedef unsigned char      HI_U5Q3;
typedef unsigned short     HI_U0Q16;
typedef unsigned short     HI_U4Q12;
typedef unsigned short     HI_U6Q10;
typedef unsigned short     HI_U8Q8;
typedef unsigned short     HI_U14Q2;
typedef unsigned short     HI_U12Q4;
typedef short              HI_S14Q2;
typedef short              HI_S9Q7;
typedef unsigned int       HI_U22Q10;
typedef unsigned int       HI_U25Q7;
typedef int                HI_S25Q7;
typedef unsigned short     HI_U8Q4F4; /*8bits unsigned integer, 4bits
decimal fraction, 4bits flag bits*/
```

### 【成员】

成员名称	描述
HI_U0Q8	用 0bit 表示整数部分，8bit 表示小数部分。文档中用 UQ0.8 来表示。
HI_U1Q7	用高 1bit 无符号数据表示整数部分，低 7bit 表示小数部分。文档中用 UQ1.7 来表示。



成员名称	描述
HI_U5Q3	用高 5bit 无符号数据表示整数部分，低 3bit 表示小数部分。文档中用 UQ5.3 来表示。
HI_U0Q16	用 0bit 表示整数部分，16bit 表示小数部分。文档中用 UQ0.16 来表示。
HI_U4Q12	用高 4bit 无符号数据表示整数部分，低 12bit 表示小数部分。文档中用 UQ4.12 来表示。
HI_U6Q10	用高 6bit 无符号数据表示整数部分，低 10bit 表示小数部分。文档中用 UQ6.10 来表示。
HI_U8Q8	用高 8bit 无符号数据表示整数部分，低 8bit 表示小数部分。文档中用 UQ8.8 来表示。
HI_U14Q2	用高 14bit 无符号数据表示整数部分，低 2bit 表示小数部分。文档中用 UQ14.2 来表示。
HI_U12Q4	用高 12bit 无符号数据表示整数部分，低 4bit 表示小数部分。文档中用 UQ12.4 来表示。
HI_S14Q2	用高 14bit 有符号数据表示整数部分，低 2bit 表示小数部分。文档中用 SQ14.2 来表示。
HI_S9Q7	用高 9bit 有符号数据表示整数部分，低 7bit 表示小数部分。文档中用 SQ9.7 来表示。
HI_U22Q10	用高 22bit 无符号数据表示整数部分，低 10bit 表示小数部分。文档中用 UQ22.10 来表示。
HI_U25Q7	用高 25bit 无符号数据表示整数部分，低 7bit 表示小数部分。文档中用 UQ25.7 来表示。
HI_S25Q7	用高 25bit 有符号数据表示整数部分，低 7bit 表示小数部分。文档中用 SQ25.7 来表示。
HI_U8Q4F4	用高 8bit 无符号数据表示整数部分，中间 4bit 表示小数部分，低 4bit 表示标志位。文档中用 UQF8.4.4 来表示。

#### 【注意事项】

HI\_UxQyFz\HI\_SxQy:

- U 后面的数字 x 表示是用 x bit 无符号数据表示整数部分；
- S 后面的数字 x 表示用 x bit 有符号数据表示整数部分；
- Q 后面的数字 y 表示用 y bit 数据表示小数部分；
- F 后面的数字 z 表示用 z bit 来表示标志位；
- 从左到右依次表示高 bit 位到低 bit 位。

#### 【相关数据类型及接口】



无。

## IVE\_HIST\_NUM

### 【说明】

定义直方图统计 bin 数目。

### 【定义】

```
#define IVE_HIST_NUM    256
```

### 【成员】

无。

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## IVE\_MAP\_NUM

### 【说明】

定义映射查找表项数目。

### 【定义】

```
#define IVE_MAP_NUM    256
```

### 【成员】

无。

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## IVE\_MAX\_REGION\_NUM

### 【说明】

定义最大连通区域数目。

### 【定义】

```
#define IVE_MAX_REGION_NUM    254
```

### 【成员】



无。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_ST\_MAX\_CORNER\_NUM

**【说明】**

定义 Shi-Tomasi-like 角点最大数目。

**【定义】**

```
#define IVE_ST_MAX_CORNER_NUM    200
```

**【成员】**

无。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_IMAGE\_TYPE\_E

**【说明】**

定义二维广义图像支持的图像类型。

**【定义】**

```
typedef enum hiIVE_IMAGE_TYPE_E
{
    IVE_IMAGE_TYPE_U8C1          = 0x0,
    IVE_IMAGE_TYPE_S8C1          = 0x1,

    IVE_IMAGE_TYPE_YUV420SP      = 0x2,          /*YUV420 SemiPlanar*/
    IVE_IMAGE_TYPE_YUV422SP      = 0x3,          /*YUV422 SemiPlanar*/
    IVE_IMAGE_TYPE_YUV420P       = 0x4,          /*YUV420 Planar */
    IVE_IMAGE_TYPE_YUV422P       = 0x5,          /*YUV422 planar */

    IVE_IMAGE_TYPE_S8C2_PACKAGE  = 0x6,
    IVE_IMAGE_TYPE_S8C2_PLANAR   = 0x7,

    IVE_IMAGE_TYPE_S16C1         = 0x8,
```



```

IVE_IMAGE_TYPE_U16C1      = 0x9,

IVE_IMAGE_TYPE_U8C3_PACKAGE = 0xa,
IVE_IMAGE_TYPE_U8C3_PLANAR  = 0xb,

IVE_IMAGE_TYPE_S32C1      = 0xc,
IVE_IMAGE_TYPE_U32C1      = 0xd,

IVE_IMAGE_TYPE_S64C1      = 0xe,
IVE_IMAGE_TYPE_U64C1      = 0xf,

IVE_IMAGE_TYPE_BUTT

} IVE_IMAGE_TYPE_E;

```

#### 【成员】

成员名称	描述
IVE_IMAGE_TYPE_U8C1	每个像素用 1 个 8bit 无符号数据表示的单通道图像。请参见图 1-2。
IVE_IMAGE_TYPE_S8C1	每个像素用 1 个 8bit 有符号数据表示的单通道图像。请参见图 1-2。
IVE_IMAGE_TYPE_YUV420SP	YUV420 Semiplanar 格式的图像。 请参见图 1-3。
IVE_IMAGE_TYPE_YUV422SP	YUV422 Semiplanar 格式的图像。 请参见图 1-4。
IVE_IMAGE_TYPE_YUV420P	YUV420 Planar 格式的图像。请参见图 1-5。
IVE_IMAGE_TYPE_YUV422P	YUV422 Planar 格式的图像。请参见图 1-6。
IVE_IMAGE_TYPE_S8C2_PACKAGE	每个像素用 2 个 8bit 有符号数据表示，且以 package 格式存储 2 通道图像。 请参见图 1-7。
IVE_IMAGE_TYPE_S8C2_PLANAR	每个像素用 2 个 8bit 有符号数据表示，且以 planar 格式存储 2 通道图像。 请参见图 1-8。
IVE_IMAGE_TYPE_S16C1	每个像素用 1 个 16bit 有符号数据表示单通道图像。请参见图 1-2。
IVE_IMAGE_TYPE_U16C1	每个像素用 1 个 16bit 无符号数据表示单通道图像。请参见图 1-2。



成员名称	描述
IVE_IMAGE_TYPE_U8C3_PACK AGE	每个像素用 3 个 8bit 无符号数据表示且以 planar 格式存储 3 通道图像。 请参见图 1-9。
IVE_IMAGE_TYPE_U8C3_PLAN AR	每个像素用 3 个 8bit 无符号数据表示 1 个像素 的 3 通道图像，且以 planar 格式存储。 请参见图 1-10。
IVE_IMAGE_TYPE_S32C1	每个像素用 1 个 32bit 有符号数据表示单通道 图像。请参见图 1-2。
IVE_IMAGE_TYPE_U32C1	每个像素用 1 个 32bit 无符号数据表示单通道 图像。请参见图 1-2。
IVE_IMAGE_TYPE_S64C1	每个像素用 1 个 64bit 有符号数据表示单通道 图像。请参见图 1-2。
IVE_IMAGE_TYPE_U64C1	每个像素用 1 个 64bit 无符号数据表示单通道 图像。请参见图 1-2。

【注意事项】

无。

【相关数据类型及接口】

- IVE\_IMAGE\_S
- IVE\_SRC\_IMAGE\_S
- IVE\_DST\_IMAGE\_S

## IVE\_IMAGE\_S

【说明】

定义二维广义图像信息。

【定义】

```
typedef struct hiIVE_IMAGE_S
{
    IVE_IMAGE_TYPE_E  enType;
    HI_U32  u32PhyAddr[3];
    HI_U8  *pu8VirAddr[3];
    HI_U16 u16Stride[3];
    HI_U16 u16Width;
    HI_U16 u16Height;
    HI_U16 u16Reserved;    /*Can be used such as elemSize*/
}IVE_IMAGE_S;
```



【成员】

成员名称	描述
enType	广义图像的图像类型。
u32PhyAddr[3]	广义图像的物理地址数组。
pu8VirAddr[3]	广义图像的虚拟地址数组。
u16Stride[3]	广义图像的跨度。
u16Width	广义图像的宽度。
u16Height	广义图像的高度。
u16Reserved	保留位。

【注意事项】

- 不同的算子对图像图像的输入输出地址是否对齐有不同的要求。
- u16Width、u16Height 和 u16Stride 均是以像素为度量单位的。
- 每种 type 下的图像示意图请参见图 1-2～图 1-10。

【相关数据类型及接口】

- [IVE\\_IMAGE\\_TYPE\\_E](#)
- [IVE\\_SRC\\_IMAGE\\_S](#)
- [IVE\\_DST\\_IMAGE\\_S](#)

## IVE\_SRC\_IMAGE\_S

【说明】

定义源图像。

【定义】

```
typedef IVE_IMAGE_S IVE_SRC_IMAGE_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

- [IVE\\_IMAGE\\_S](#)
- [IVE\\_DST\\_IMAGE\\_S](#)



## IVE\_DST\_IMAGE\_S

### 【说明】

定义输出图像。

### 【定义】

```
typedef IVE_IMAGE_S IVE_DST_IMAGE_S;
```

### 【成员】

无。

### 【注意事项】

无。

### 【相关数据类型及接口】

- [IVE\\_IMAGE\\_S](#)
- [IVE\\_SRC\\_IMAGE\\_S](#)

## IVE\_DATA\_S

### 【说明】

定义以 byte 为单位的二维数据信息。

### 【定义】

```
typedef struct hiIVE_DATA_S
{
    HI_U32  u32PhyAddr; /*Physical address of the data*/
    HI_U8   *pu8VirAddr;
    HI_U16  u16Stride; /*Data stride by byte*/
    HI_U16  u16Height; /*Data height by byte*/
    HI_U16  u16Width; /*Data width by byte*/
    HI_U16  u16Reserved;
} IVE_DATA_S;
```

### 【成员】

成员名称	描述
u32PhyAddr	图像物理地址。
pu8VirAddr	图像虚拟地址。
u16Stride	图像跨度。
u16Height	图像宽度。
u16Width	图像高度。





成员名称	描述
u16Reserved	保留位。

**【注意事项】**

表示以 byte 为单位的二维数据；可以与 [IVE\\_IMAGE\\_S](#) 图像进行转换。

**【相关数据类型及接口】**

无。

## IVE\_SRC\_DATA\_S

**【说明】**

定义以 byte 为单位的二维源数据信息。

**【定义】**

```
typedef IVE_DATA_S IVE_SRC_DATA_S;
```

**【成员】**

无。

**【注意事项】**

无。

**【相关数据类型及接口】**

- [IVE\\_IMAGE\\_S](#)
- [IVE\\_DST\\_DATA\\_S](#)

## IVE\_DST\_DATA\_S

**【说明】**

定义 byte 为单位的二维输出数据信息。

**【定义】**

```
typedef IVE_DATA_S IVE_DST_DATA_S;
```

**【成员】**

无。

**【注意事项】**

无。

**【相关数据类型及接口】**

- [IVE\\_IMAGE\\_S](#)



- [IVE\\_SRC\\_IMAGE\\_S](#)

## IVE\_MEM\_INFO\_S

### 【说明】

定义一维数据内存信息。

### 【定义】

```
typedef struct hiIVE_MEM_INFO_S
{
    HI_U32  u32PhyAddr;
    HI_U8   *pu8VirAddr;
    HI_U32  u32Size;
} IVE_MEM_INFO_S;
```

### 【成员】

成员名称	描述
u32PhyAddr	一维数据物理地址。
pu8VirAddr	一维数据虚拟地址。
u32Size	一维数据 byte 数目。

### 【注意事项】

无。

### 【相关数据类型及接口】

- [IVE\\_SRC\\_MEM\\_INFO\\_S](#)
- [IVE\\_DST\\_MEM\\_INFO\\_S](#)

## IVE\_SRC\_MEM\_INFO\_S

### 【说明】

定义一维源数据。

### 【定义】

```
typedef IVE_MEM_INFO_S IVE_SRC_MEM_INFO_S;
```

### 【成员】

无。

### 【注意事项】

无。



【相关数据类型及接口】

- [IVE\\_MEM\\_INFO\\_S](#)
- [IVE\\_DST\\_MEM\\_INFO\\_S](#)

## IVE\_DST\_MEM\_INFO\_S

【说明】

定义一维输出数据。

【定义】

```
typedef IVE_MEM_INFO_S IVE_DST_MEM_INFO_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

- [IVE\\_MEM\\_INFO\\_S](#)
- [IVE\\_SRC\\_MEM\\_INFO\\_S](#)

## IVE\_8BIT\_U

【说明】

定义 8bit 数据联合体。

【定义】

```
typedef union hiIVE_8BIT_U
{
    HI_S8 s8Val;
    HI_U8 u8Val;
}IVE_8BIT_U;
```

【成员】

成员名称	描述
s8Val	有符号 8bit 值。
u8Val	无符号 8bit 值。

【注意事项】

无



【相关数据类型及接口】

无。

## IVE\_POINT\_U16\_S

【说明】

定义 U16 表示的点信息结构体。

【定义】

```
typedef struct hiIVE_POINT_U16_S
{
    HI_U16 u16X;
    HI_U16 u16Y;
} IVE_POINT_U16_S;
```

【成员】

成员名称	描述
u16X	点的 x 坐标。
u16Y	点的 y 坐标。

【注意事项】

无。

【相关数据类型及接口】

无

## IVE\_POINT\_S25Q7\_S

【说明】

定义 S25Q7 定点表示的点信息结构体。

【定义】

```
typedef struct hiIVE_POINT_S25Q7_S
{
    HI_S25Q7    s25q7X;           /*X coordinate*/
    HI_S25Q7    s25q7Y;           /*Y coordinate*/
} IVE_POINT_S25Q7_S;
```

【成员】



成员名称	描述
s25q7X	点的 x 坐标，以 SQ25.7 表示。
s25q7Y	点的 y 坐标，以 SQ25.7 表示。

【注意事项】

无。

【相关数据类型及接口】

无

## IVE\_RECT\_S

【说明】

定义 U16 表示的矩形信息结构体。

【定义】

```
typedef struct hiIVE_RECT_S
{
    HI_U16 u16X;
    HI_U16 u16Y;
    HI_U16 u16Width;
    HI_U16 u16Height;
} IVE_RECT_S;
```

【成员】

成员名称	描述
u16X	矩形相对于坐标原点最近点的 x 坐标。
u16Y	矩形相对于坐标原点最近点的 y 坐标。
u16Width	矩形的宽。
u16Height	矩形的高。

【注意事项】

无。

【相关数据类型及接口】

无。



## IVE\_DMA\_MODE\_E

### 【说明】

定义 DMA 操作模式。

### 【定义】

```
typedef enum hiIVE_DMA_MODE_E
{
    IVE_DMA_MODE_DIRECT_COPY    = 0x0,
    IVE_DMA_MODE_INTERVAL_COPY  = 0x1,
    IVE_DMA_MODE_SET_3BYTE      = 0x2,
    IVE_DMA_MODE_SET_8BYTE      = 0x3,
    IVE_DMA_MODE_BUTT
} IVE_DMA_MODE_E;
```

### 【成员】

成员名称	描述
IVE_DMA_MODE_DIRECT_COPY	直接快速拷贝模式。
IVE_DMA_MODE_INTERVAL_COPY	间隔拷贝模式，请参见 <a href="#">HI_MPI_IVE_DMA</a> 【注意】说明。
IVE_DMA_MODE_SET_3BYTE	3byte 赋值模式，请参见 <a href="#">HI_MPI_IVE_DMA</a> 【注意】说明。
IVE_DMA_MODE_SET_8BYTE	8byte 赋值模式，请参见 <a href="#">HI_MPI_IVE_DMA</a> 【注意】说明。

### 【注意事项】

无。

### 【相关数据类型及接口】

[IVE\\_DMA\\_CTRL\\_S](#)

## IVE\_DMA\_CTRL\_S

### 【说明】

定义 DMA 控制信息。

### 【定义】

```
typedef struct hiIVE_DMA_CTRL_S
{
    IVE\_DMA\_MODE\_E enMode;
    HI_U64 u64Val;
```



```
HI_U8 u8HorSegSize;  
HI_U8 u8ElemSize;  
HI_U8 u8VerSegRows;  
}IVE_DMA_CTRL_S;
```

#### 【成员】

成员名称	描述
enMode	DMA 操作模式。
u64Val	仅赋值模式使用，用于对内存赋值，3byte 赋值模式用低 3byte 保存。
u8HorSegSize	仅间隔拷贝模式使用，水平方向将源图像一行分割的段大小。 取值范围：{2, 3, 4, 8, 16}。
u8ElemSize	仅间隔拷贝模式使用，分割的每一段中前 u8ElemSizebyte 为有效的拷贝字段。 取值范围：[1, u8HorSegSize-1]。
u8VerSegRows	仅间隔拷贝模式使用，将每 u8VerSegRows 行中第一行数据分割为 u8HorSegSize 大小的段，拷贝每段中的前 u8ElemSize 大小的字节 取值范围：[1, min{65535/srcStride, srcHeight}]。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_DMA\\_MODE\\_E](#)

## IVE\_FILTER\_CTRL\_S

#### 【说明】

定义模板滤波控制信息。

#### 【定义】

```
typedef struct hiIVE_FILTER_CTRL_S  
{  
    HI_S8 as8Mask[25];        /*Template parameter filter coefficient*/  
    HI_U8 u8Norm;              /*Normalization parameter, by right shift*/  
}IVE_FILTER_CTRL_S;
```

#### 【成员】



成员名称	描述
as8Mask[5]	5x5 模板系数，外围系数设为 0 可实现 3x3 模板滤波。
u8Norm	归一化参数。 取值范围：[0, 13]。

**【注意事项】**

通过配置不同的模板系数可以达到不同的滤波效果。

**【相关数据类型及接口】**

无。

## IVE\_CSC\_MODE\_E

**【说明】**

定义色彩空间转换模式。

**【定义】**

```
typedef enum hiIVE_CSC_MODE_E
{
    /*CSC: YUV2RGB, video transfer mode, RGB value range [16, 235]*/
    IVE_CSC_MODE_VIDEO_BT601_YUV2RGB = 0x0,
    /*CSC: YUV2RGB, video transfer mode, RGB value range [16, 235]*/
    IVE_CSC_MODE_VIDEO_BT709_YUV2RGB = 0x1,
    /*CSC: YUV2RGB, picture transfer mode, RGB value range [0, 255]*/
    IVE_CSC_MODE_PIC_BT601_YUV2RGB = 0x2,
    /*CSC: YUV2RGB, picture transfer mode, RGB value range [0, 255]*/
    IVE_CSC_MODE_PIC_BT709_YUV2RGB = 0x3,
    /*CSC: YUV2HSV, picture transfer mode, HSV value range [0, 255]*/
    IVE_CSC_MODE_PIC_BT601_YUV2HSV = 0x4,
    /*CSC: YUV2HSV, picture transfer mode, HSV value range [0, 255]*/
    IVE_CSC_MODE_PIC_BT709_YUV2HSV = 0x5,

    /*CSC: YUV2LAB, picture transfer mode, Lab value range [0, 255]*/
    IVE_CSC_MODE_PIC_BT601_YUV2LAB = 0x6,
    /*CSC: YUV2LAB, picture transfer mode, Lab value range [0, 255]*/
    IVE_CSC_MODE_PIC_BT709_YUV2LAB = 0x7,

    /*CSC: RGB2YUV, video transfer mode, YUV value range [0, 255]*/
    IVE_CSC_MODE_VIDEO_BT601_RGB2YUV = 0x8,
    /*CSC: RGB2YUV, video transfer mode, YUV value range [0, 255]*/
    IVE_CSC_MODE_VIDEO_BT709_RGB2YUV = 0x9,
```





```

/*CSC: RGB2YUV, picture transfer mode, Y:[16, 235],U\V:[16, 240]*/
IVE_CSC_MODE_PIC_BT601_RGB2YUV = 0xa,
/*CSC: RGB2YUV, picture transfer mode, Y:[16, 235],U\V:[16, 240]*/
IVE_CSC_MODE_PIC_BT709_RGB2YUV = 0xb,

IVE_CSC_MODE_BUTT
}IVE_CSC_MODE_E;

```

**【成员】**

成员名称	描述
IVE_CSC_MODE_VIDEO_BT601_YUV2RGB	BT601 的 YUV2RGB 视频变换。
IVE_CSC_MODE_VIDEO_BT709_YUV2RGB	BT709 的 YUV2RGB 视频变换。
IVE_CSC_MODE_PIC_BT601_YUV2RGB	BT601 的 YUV2RGB 图像变换。
IVE_CSC_MODE_PIC_BT709_YUV2RGB	BT709 的 YUV2RGB 图像变换。
IVE_CSC_MODE_PIC_BT601_YUV2HSV	BT601 的 YUV2HSV 图像变换。
IVE_CSC_MODE_PIC_BT709_YUV2HSV	BT709 的 YUV2HSV 图像变换。
IVE_CSC_MODE_PIC_BT601_YUV2LAB	BT601 的 YUV2LAB 图像变换。
IVE_CSC_MODE_PIC_BT709_YUV2LAB	BT709 的 YUV2LAB 图像变换。
IVE_CSC_MODE_VIDEO_BT601_RGB2YUV	BT601 的 RGB2YUV 视频变换。
IVE_CSC_MODE_VIDEO_BT709_RGB2YUV	BT709 的 RGB2YUV 视频变换。
IVE_CSC_MODE_PIC_BT601_RGB2YUV	BT601 的 RGB2YUV 图像变换。
IVE_CSC_MODE_PIC_BT709_RGB2YUV	BT709 的 RGB2YUV 图像变换。

**【注意事项】**

- IVE\_CSC\_MODE\_VIDEO\_BT601\_YUV2RGB 和 IVE\_CSC\_MODE\_VIDEO\_BT709\_YUV2RGB 模式，输出满足  $16 \leq R, G, B \leq 235$ 。
- IVE\_CSC\_MODE\_PIC\_BT601\_YUV2RGB 和 IVE\_CSC\_MODE\_PIC\_BT709\_YUV2RGB 模式，输出满足  $0 \leq R, G, B \leq 255$ 。
- IVE\_CSC\_MODE\_PIC\_BT601\_YUV2HSV 和 IVE\_CSC\_MODE\_PIC\_BT709\_YUV2HSV 模式，输出满足  $0 \leq H, S, V \leq 255$ 。
- IVE\_CSC\_MODE\_PIC\_BT601\_YUV2LAB 和 IVE\_CSC\_MODE\_PIC\_BT709\_YUV2LAB 模式，输出满足  $0 \leq L, A, B \leq 255$ 。
- IVE\_CSC\_MODE\_VIDEO\_BT601\_RGB2YUV 和 IVE\_CSC\_MODE\_VIDEO\_BT709\_RGB2YUV 模式，输出满足  $0 \leq Y, U, V \leq 255$ 。



- IVE\_CSC\_MODE\_PIC\_BT601\_RGB2YUV 和 IVE\_CSC\_MODE\_PIC\_BT709\_RGB2YUV 模式，输出满足  $0 \leq Y \leq 235$ ， $0 \leq U, V \leq 240$ 。

【相关数据类型及接口】

- [IVE\\_CSC\\_CTRL\\_S](#)
- [IVE\\_FILTER\\_AND\\_CSC\\_CTRL\\_S](#)

## IVE\_CSC\_CTRL\_S

【说明】

定义色彩空间转换控制信息。

【定义】

```
typedef struct hiIVE_CSC_CTRL_S
{
    IVE_CSC_MODE_E    enMode; /*Working mode*/
} IVE_CSC_CTRL_S;
```

【成员】

成员名称	描述
enMode	工作模式。

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_CSC\\_MODE\\_E](#)

## IVE\_FILTER\_AND\_CSC\_CTRL\_S

【说明】

定义模板滤波加色彩空间转换复合功能控制信息。

【定义】

```
typedef struct hiIVE_FILTER_AND_CSC_CTRL_S
{
    IVE_CSC_MODE_E    enMode; /*CSC working mode*/
    HI_S8    as8Mask[25];    /*Template parameter filter coefficient*/
    HI_U8    u8Norm;          /*Normalization parameter, by right shift*/
} IVE_FILTER_AND_CSC_CTRL_S;
```

【成员】



成员名称	描述
enMode	工作模式。
as8Mask[25]	5x5 模板系数。
u8Norm	归一化参数。 取值范围：[0, 13]。

【注意事项】

仅支持 YUV2RGB 的 4 种模式。

【相关数据类型及接口】

[IVE\\_CSC\\_MODE\\_E](#)

## IVE\_SOBEL\_OUT\_CTRL\_E

【说明】

定义 Sobel 输出控制信息。

【定义】

```
typedef enum hiIVE_SOBEL_OUT_CTRL_E
{
    IVE_SOBEL_OUT_CTRL_BOTH = 0x0, /*Output horizontal and vertical*/
    IVE_SOBEL_OUT_CTRL_HOR   = 0x1, /*Output horizontal*/
    IVE_SOBEL_OUT_CTRL_VER   = 0x2, /*Output vertical*/
    IVE_SOBEL_OUT_CTRL_BUTT
} IVE_SOBEL_OUT_CTRL_E;
```

【成员】

成员名称	描述
IVE_SOBEL_OUT_CTRL_BOTH	同时输出用模板和转置模板滤波的结果。
IVE_SOBEL_OUT_CTRL_HOR	仅输出用模板直接滤波的结果。
IVE_SOBEL_OUT_CTRL_VER	仅输出用转置模板滤波的结果。

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_SOBEL\\_CTRL\\_S](#)



## IVE\_SOBEL\_CTRL\_S

### 【说明】

定义 Sobel-like 梯度计算控制信息。

### 【定义】

```
typedef struct hiIVE_SOBEL_CTRL_S
{
    IVE_SOBEL_OUT_CTRL_E enOutCtrl; /*Output format*/
    HI_S8 as8Mask[25];               /*Template parameter*/
} IVE_SOBEL_CTRL_S;
```

### 【成员】

成员名称	描述
enOutCtrl	输出控制枚举参数。
as8Mask[25]	5x5 模板系数。

### 【注意事项】

无。

### 【相关数据类型及接口】

[IVE\\_SOBEL\\_OUT\\_CTRL\\_E](#)

## IVE\_MAG\_AND\_ANG\_OUT\_CTRL\_E

### 【说明】

定义梯度幅值与角度计算的输出格式。

### 【定义】

```
typedef enum hiIVE_MAG_AND_ANG_OUT_CTRL_E
{
    IVE_MAG_AND_ANG_OUT_CTRL_MAG          = 0x0,
    IVE_MAG_AND_ANG_OUT_CTRL_MAG_AND_ANG = 0x1,
    IVE_MAG_AND_ANG_OUT_CTRL_BUTT
} IVE_MAG_AND_ANG_OUT_CTRL_E;
```

### 【成员】

成员名称	描述
IVE_MAG_AND_ANG_OUT_CTRL_MAG	仅输出幅值。
IVE_MAG_AND_ANG_OUT_CTRL_MAG_AND_ANG	同时输出幅值和角度值。



【注意事项】

无。

【相关数据类型及接口】

[IVE\\_MAG\\_AND\\_ANG\\_CTRL\\_S](#)

## IVE\_MAG\_AND\_ANG\_CTRL\_S

【说明】

定义梯度幅值和幅角计算的控制信息。

【定义】

```
typedef struct hiIVE_MAG_AND_ANG_CTRL_S
{
    IVE_MAG_AND_ANG_OUT_CTRL_E enOutCtrl;
    HI_U16 u16Thr;
    HI_S8 as8Mask[25];          /*Template parameter.*/
} IVE_MAG_AND_ANG_CTRL_S;
```

【成员】

成员名称	描述
enOutCtrl	输出格式。
u16Thr	用于对幅值进行阈值化的阈值。
as8Mask[25]	5x5 模板系数。

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_MAG\\_AND\\_ANG\\_OUT\\_CTRL\\_E](#)

## IVE\_DILATE\_CTRL\_S

【说明】

定义膨胀控制信息。

【定义】

```
typedef struct hiIVE_DILATE_CTRL_S
{
    HI_U8 au8Mask[25]; /*The template parameter value must be 0 or 255.*/
}
```



```
} IVE_DILATE_CTRL_S;
```

#### 【成员】

成员名称	描述
au8Mask[25]	5x5 模板系数。 取值范围：0 或 255。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

无。

### IVE\_ERODE\_CTRL\_S

#### 【说明】

定义腐蚀控制信息。

#### 【定义】

```
typedef struct hiIVE_ERODE_CTRL_S
{
    HI_U8 au8Mask[25]; /*The template parameter value must be 0 or 255.*/
} IVE_ERODE_CTRL_S;
```

#### 【成员】

成员名称	描述
au8Mask[25]	5x5 模板系数。 取值范围：0 或 255。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

无。

### IVE\_THRESH\_MODE\_E

#### 【说明】

定义图像二值化输出格式。



### 【定义】

```
typedef enum hiIVE_THRESH_MODE_E
{
    IVE_THRESH_MODE_BINARY      = 0x0, /*srcVal <= lowThr, dstVal =
minVal; srcVal > lowThr, dstVal = maxVal.*/
    IVE_THRESH_MODE_TRUNC       = 0x1, /*srcVal <= lowThr, dstVal =
srcVal; srcVal > lowThr, dstVal = maxVal.*/
    IVE_THRESH_MODE_TO_MINVAL   = 0x2, /*srcVal <= lowThr, dstVal =
minVal; srcVal > lowThr, dstVal = srcVal.*/
    IVE_THRESH_MODE_MIN_MID_MAX = 0x3, /*srcVal <= lowThr, dstVal =
minVal; lowThr < srcVal <= highThr, dstVal = midVal; srcVal > highThr,
dstVal = maxVal.*/
    IVE_THRESH_MODE_ORI_MID_MAX = 0x4, /*srcVal <= lowThr, dstVal =
srcVal; lowThr < srcVal <= highThr, dstVal = midVal; srcVal > highThr,
dstVal = maxVal.*/
    IVE_THRESH_MODE_MIN_MID_ORI = 0x5, /*srcVal <= lowThr, dstVal =
minVal; lowThr < srcVal <= highThr, dstVal = midVal; srcVal > highThr,
dstVal = srcVal.*/
    IVE_THRESH_MODE_MIN_ORI_MAX = 0x6, /*srcVal <= lowThr, dstVal =
minVal; lowThr < srcVal <= highThr, dstVal = srcVal; srcVal > highThr,
dstVal = maxVal.*/
    IVE_THRESH_MODE_ORI_MID_ORI = 0x7, /*srcVal <= lowThr, dstVal =
srcVal; lowThr < srcVal <= highThr, dstVal = midVal; srcVal > highThr,
dstVal = srcVal.*/

    IVE_THRESH_MODE_BUTT
}IVE_THRESH_MODE_E;
```

### 【成员】

成员名称	描述
IVE_THRESH_MODE_BINARY	srcVal $\leq$ lowThr, dstVal = minVal; srcVal > lowThr, dstVal = maxVal。
IVE_THRESH_MODE_TRUNC	srcVal $\leq$ lowThr, dstVal = srcVal; srcVal > lowThr, dstVal = maxVal。
IVE_THRESH_MODE_TO_MINVAL	srcVal $\leq$ lowThr, dstVal = minVal; srcVal > lowThr, dstVal = srcVal。
IVE_THRESH_MODE_MIN_MID_MAX	srcVal $\leq$ lowThr, dstVal = minVal; lowThr < srcVal $\leq$ highThr, dstVal = midVal; srcVal > highThr, dstVal = maxVal。



成员名称	描述
IVE_THRESH_MODE_ORI_MID_MAX	$\text{srcVal} \leq \text{lowThr}$ , $\text{dstVal} = \text{srcVal}$ ; $\text{lowThr} < \text{srcVal} \leq \text{highThr}$ , $\text{dstVal} = \text{midVal}$ ; $\text{srcVal} > \text{highThr}$ , $\text{dstVal} = \text{maxVal}$ 。
IVE_THRESH_MODE_MIN_MID_ORI	$\text{srcVal} \leq \text{lowThr}$ , $\text{dstVal} = \text{minVal}$ ; $\text{lowThr} < \text{srcVal} \leq \text{highThr}$ , $\text{dstVal} = \text{midVal}$ ; $\text{srcVal} > \text{highThr}$ , $\text{dstVal} = \text{srcVal}$ 。
IVE_THRESH_MODE_MIN_ORI_MAX	$\text{srcVal} \leq \text{lowThr}$ , $\text{dstVal} = \text{minVal}$ ; $\text{lowThr} < \text{srcVal} \leq \text{highThr}$ , $\text{dstVal} = \text{srcVal}$ ; $\text{srcVal} > \text{highThr}$ , $\text{dstVal} = \text{maxVal}$ 。
IVE_THRESH_MODE_ORI_MID_ORI	$\text{srcVal} \leq \text{lowThr}$ , $\text{dstVal} = \text{srcVal}$ ; $\text{lowThr} < \text{srcVal} \leq \text{highThr}$ , $\text{dstVal} = \text{midVal}$ ; $\text{srcVal} > \text{highThr}$ , $\text{dstVal} = \text{srcVal}$ 。

#### 【注意事项】

计算公式请参见 [HI\\_MPI\\_IVE\\_Thresh](#) 中的【注意】，示意图请参见图 2-8。

#### 【相关数据类型及接口】

[IVE\\_THRESH\\_CTRL\\_S](#)

## IVE\_THRESH\_CTRL\_S

#### 【说明】

定义图像二值化控制信息。

#### 【定义】

```
typedef struct hiIVE_THRESH_CTRL_S
{
    IVE_THRESH_MODE_E enMode;
    HI_U8 u8LowThr;          /*user-defined threshold, 0<=u8LowThr<=255 */
    HI_U8 u8HighThr;         /*user-defined threshold, if
enMode<IVE_THRESH_MODE_MIN_MID_MAX, u8HighThr is not used, else
0<=u8LowThr<=u8HighThr<=255; */
    HI_U8 u8MinVal;          /*Minimum value when tri-level thresholding*/
    HI_U8 u8MidVal;          /*Middle value when tri-level thresholding, if
enMode<2, u32MidVal is not used; */
```





```
        HI_U8 u8MaxVal;           /*Maxmum value when tri-level thresholding*/  
    } IVE_THRESH_CTRL_S;
```

#### 【成员】

成员名称	描述
enMode	阈值化运算模式。
u8LowThresh	低阈值。 取值范围：[0,255]。
u8HighThresh	高阈值。 $0 \leq u8LowThresh \leq u8HighThresh \leq 255$ 。
u8MinVal	最小值。 取值范围：[0,255]。
u8MidVal	中间值。 取值范围：[0,255]。
u8MaxVal	最大值。 取值范围：[0,255]。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_THRESH\\_MODE\\_E](#)

### IVE\_SUB\_MODE\_E

#### 【说明】

定义两图像相减输出格式。

#### 【定义】

```
typedef enum hiIVE_SUB_MODE_E  
{  
    IVE_SUB_MODE_ABS      = 0x0,    /*Absolute value of the difference*/  
    IVE_SUB_MODE_SHIFT    = 0x1,    /*The output result is obtained by  
shifting the result one digit right to reserve the signed bit.*/  
    IVE_SUB_MODE_BUTT  
} IVE_SUB_MODE_E;
```

#### 【成员】



成员名称	描述
IVE_SUB_MODE_ABS	取差的绝对值。
IVE_SUB_MODE_SHIFT	将结果右移一位输出，保留符号位。

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_SUB\\_CTRL\\_S](#)

## IVE\_SUB\_CTRL\_S

【说明】

定义两图像相减控制参数。

【定义】

```
typedef struct hiIVE_SUB_CTRL_S
{
    IVE\_SUB\_MODE\_E enMode;
} IVE_SUB_CTRL_S;
```

【成员】

成员名称	描述
enMode	两图像相减模式

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_SUB\\_MODE\\_E](#)

## IVE\_INTEG\_OUT\_CTRL\_E

【说明】

定义积分图输出控制参数。

【定义】

```
typedef enum hiIVE_INTEG_OUT_CTRL_E
{
    IVE_INTEG_OUT_CTRL_COMBINE = 0x0,
```



```
IVE_INTEG_OUT_CTRL_SUM      = 0x1,
IVE_INTEG_OUT_CTRL_SQSUM    = 0x2,
IVE_INTEG_OUT_CTRL_BUTT
} IVE_INTEG_OUT_CTRL_E;
```

#### 【成员】

成员名称	描述
IVE_INTEG_OUT_CTRL_COMBINE	和、平方和积分图组合输出，如图 1-13。
IVE_INTEG_OUT_CTRL_SUM	仅和积分图输出。
IVE_INTEG_OUT_CTRL_SQSUM	仅平方和积分图输出。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_INTEG\\_CTRL\\_S](#)

### IVE\_INTEG\_CTRL\_S

#### 【说明】

定义积分图计算控制参数。

#### 【定义】

```
typedef struct hiIVE_INTEG_CTRL_S
{
    IVE_INTEG_OUT_CTRL_E enOutCtrl;
} IVE_INTEG_CTRL_S;
```

#### 【成员】

成员名称	描述
enOutCtrl	积分图输出控制参数

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_INTEG\\_OUT\\_CTRL\\_E](#)



## IVE\_THRESH\_S16\_MODE\_E

### 【说明】

定义 16bit 有符号图像的阈值化模式。

### 【定义】

```
typedef enum hiIVE_THRESH_S16_MODE_E
{
    IVE_THRESH_S16_MODE_S16_TO_S8_MIN_MID_MAX = 0x0,
    IVE_THRESH_S16_MODE_S16_TO_S8_MIN_ORI_MAX = 0x1,
    IVE_THRESH_S16_MODE_S16_TO_U8_MIN_MID_MAX = 0x2,
    IVE_THRESH_S16_MODE_S16_TO_U8_MIN_ORI_MAX = 0x3,
    IVE_THRESH_S16_MODE_BUTT
} IVE_THRESH_S16_MODE_E;
```

### 【成员】

成员名称	描述
IVE_THRESH_S16_MODE_S16_TO_S8_MIN_MID_MAX	srcVal $\leq$ lowThr, dstVal = minVal; lowThr < srcVal $\leq$ highThr, dstVal = midVal; srcVal > highThr, dstVal = maxVal;
IVE_THRESH_S16_MODE_S16_TO_S8_MIN_ORI_MAX	srcVal $\leq$ lowThr, dstVal = minVal; lowThr < srcVal $\leq$ highThr, dstVal = srcVal; srcVal > highThr, dstVal = maxVal;
IVE_THRESH_S16_MODE_S16_TO_U8_MIN_MID_MAX	srcVal $\leq$ lowThr, dstVal = minVal; lowThr < srcVal $\leq$ highThr, dstVal = midVal; srcVal > highThr, dstVal = maxVal;



成员名称	描述
IVE_THRESH_S16_MODE_S16_TO_U8_MIN_ORI_MAX	$\text{srcVal} \leq \text{lowThr}$ , $\text{dstVal} = \text{minVal}$ ; $\text{lowThr} < \text{srcVal} \leq \text{highThr}$ , $\text{dstVal} = \text{srcVal}$ ; $\text{srcVal} > \text{highThr}$ , $\text{dstVal} = \text{maxVal}$ ;

【注意事项】

计算公式请参见 [HI\\_MPI\\_IVE\\_Thresh\\_S16](#) 中的【注意】，示意图请参见图 2-9。

【相关数据类型及接口】

[IVE\\_THRESH\\_S16\\_CTRL\\_S](#)

## IVE\_THRESH\_S16\_CTRL\_S

【说明】

定义 16bit 有符号图像的阈值化控制参数。

【定义】

```
typedef struct hiIVE_THRESH_S16_CTRL_S
{
    IVE_THRESH_S16_MODE_E enMode;
    HI_S16 s16LowThr;        /*user-defined threshold*/
    HI_S16 s16HighThr;       /*user-defined threshold*/
    IVE_8BIT_U un8MinVal;    /*Minimum value when tri-level thresholding*/
    IVE_8BIT_U un8MidVal;    /*Middle value when tri-level thresholding*/
    IVE_8BIT_U un8MaxVal;    /*Maximum value when tri-level thresholding*/
} IVE_THRESH_S16_CTRL_S;
```

【成员】

成员名称	描述
enMode	阈值化运算模式。
s16LowThr	低阈值。
s16HighThr	高阈值。
un8MinVal	最小值。
un8MidVal	中间值。
un8MaxVal	最大值。



【注意事项】

计算公式请参见 [HI\\_MPI\\_IVE\\_Thresh\\_S16](#) 中的【注意】，示意图请参见图 2-9。

【相关数据类型及接口】

[IVE\\_THRESH\\_S16\\_MODE\\_E](#)

## IVE\_THRESH\_U16\_MODE\_E

【说明】

定义 16bti 无符号图像的阈值化模式。

【定义】

```
typedef enum hiIVE_THRESH_U16_MODE_E
{
    IVE_THRESH_U16_MODE_U16_TO_U8_MIN_MID_MAX = 0x0,
    IVE_THRESH_U16_MODE_U16_TO_U8_MIN_ORI_MAX = 0x1,
    IVE_THRESH_U16_MODE_BUTT
} IVE_THRESH_U16_MODE_E;
```

【成员】

成员名称	描述
IVE_THRESH_U16_MODE_U16_TO_U8_MIN_MID_MAX	srcVal $\leq$ lowThr, dstVal = minVal; lowThr < srcVal $\leq$ highThr, dstVal = midVal; srcVal > highThr, dstVal = maxVal;
IVE_THRESH_U16_MODE_U16_TO_U8_MIN_ORI_MAX	srcVal $\leq$ lowThr, dstVal = minVal; lowThr < srcVal $\leq$ highThr, dstVal = srcVal; srcVal > highThr, dstVal = maxVal;

【注意事项】

计算公式请参见 [HI\\_MPI\\_IVE\\_Thresh\\_U16](#) 中的【注意】，示意图请参见图 2-10。

【相关数据类型及接口】

[IVE\\_THRESH\\_U16\\_CTRL\\_S](#)



## IVE\_THRESH\_U16\_CTRL\_S

### 【说明】

定义 16bit 无符号图像的阈值化控制参数。

### 【定义】

```
typedef struct hiIVE_THRESH_U16_CTRL_S
{
    IVE_THRESH_U16_MODE_E enMode;
    HI_U16 u16LowThr;
    HI_U16 u16HighThr;
    HI_U8 u8MinVal;
    HI_U8 u8MidVal;
    HI_U8 u8MaxVal;
} IVE_THRESH_U16_CTRL_S;
```

### 【成员】

成员名称	描述
enMode	阈值化运算模式。
u16LowThr	低阈值。
u16HighThr	高阈值。
u8MinVal	最小值。 取值范围：[0,255]。
u8MidVal	中间值。 取值范围：[0,255]。
u8MaxVal	最大值。 取值范围：[0,255]。

### 【注意事项】

计算公式请参见 [HI\\_MPI\\_IVE\\_Thresh\\_U16](#) 中的【注意】，示意图请参见图 2-10。

### 【相关数据类型及接口】

[IVE\\_THRESH\\_U16\\_MODE\\_E](#)

## IVE\_16BIT\_TO\_8BIT\_MODE\_E

### 【说明】

定义 16bit 图像数据到 8bit 图像数据的转化模式。

### 【定义】



```
typedef enum hiIVE_16BIT_TO_8BIT_MODE_E
{
    IVE_16BIT_TO_8BIT_MODE_S16_TO_S8      = 0x0,
    IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_ABS  = 0x1,
    IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_BIAS = 0x2,
    IVE_16BIT_TO_8BIT_MODE_U16_TO_U8      = 0x3,
    IVE_16BIT_TO_8BIT_MODE_BUTT
}IVE_16BIT_TO_8BIT_MODE_E;
```

#### 【成员】

成员名称	描述
IVE_16BIT_TO_8BIT_MODE_S16_TO_S8	S16 数据到 S8 数据的线性变换。
IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_ABS	S16 数据线性变换到 S8 数据后取绝对值得到 S8 数据。
IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_BIAS	S16 数据线性变换到 S8 数据且平移后截断到 U8 数据。
IVE_16BIT_TO_8BIT_MODE_U16_TO_U8	S16 数据线性变换到 U8 数据。

#### 【注意事项】

计算公式请参见 [HI\\_MPI\\_IVE\\_16BitTo8Bit](#) 中的【注意】，示意图请参见图 2-11。

#### 【相关数据类型及接口】

[IVE\\_16BIT\\_TO\\_8BIT\\_CTRL\\_S](#)

### IVE\_16BIT\_TO\_8BIT\_CTRL\_S

#### 【说明】

定义 16bit 图像数据到 8bit 图像数据的转化控制参数。

#### 【定义】

```
typedef struct hiIVE_16BIT_TO_8BIT_CTRL_S
{
    IVE_16BIT_TO_8BIT_MODE_E enMode;
    HI_U16 u16Denominator;
    HI_U8  u8Numerator;
    HI_S8  s8Bias;
}IVE_16BIT_TO_8BIT_CTRL_S;
```

#### 【成员】





成员名称	描述
enMode	16bit 数据到 8bit 数据的转换模式。
u16Denominator	线性变换中的分母。 取值范围: $[\max\{1, u8Numerator\}, 65535]$
u8Numerator	线性变换中的分子。 取值范围: $[0, 255]$ 。
s8Bias	线性变换中的平移项。 取值范围: $[-128, 127]$ 。

【注意事项】

- 计算公式请参见 [HI\\_MPI\\_IVE\\_Thresh\\_U16](#) 中的【注意】，示意图请参见图 2-10。
- $u8Numerator \leq u16Denominator$ ，且  $u16Denominator \neq 0$ ；

【相关数据类型及接口】

[IVE\\_16BIT\\_TO\\_8BIT\\_MODE\\_E](#)

## IVE\_ORD\_STAT\_FILTER\_MODE\_E

【说明】

定义顺序统计量滤波模式。

【定义】

```
typedef enum hiIVE_ORD_STAT_FILTER_MODE_E
{
    IVE_ORD_STAT_FILTER_MODE_MEDIAN = 0x0,
    IVE_ORD_STAT_FILTER_MODE_MAX    = 0x1,
    IVE_ORD_STAT_FILTER_MODE_MIN    = 0x2,
    IVE_ORD_STAT_FILTER_MODE_BUTT
} IVE_ORD_STAT_FILTER_MODE_E;
```

【成员】

成员名称	描述
IVE_ORD_STAT_FILTER_MODE_MEDIAN	中值滤波。
IVE_ORD_STAT_FILTER_MODE_MAX	最大值滤波，等价于灰度图的膨胀。
IVE_ORD_STAT_FILTER_MODE_MIN	最小值滤波，等价于灰度图的腐蚀。

【注意事项】



无。

【相关数据类型及接口】

[IVE\\_ORD\\_STAT\\_FILTER\\_CTRL\\_S](#)

## IVE\_ORD\_STAT\_FILTER\_CTRL\_S

【说明】

定义顺序统计量滤波控制参数。

【定义】

```
typedef struct hiIVE_ORD_STAT_FILTER_CTRL_S
{
    IVE_ORD_STAT_FILTER_MODE_E enMode;
} IVE_ORD_STAT_FILTER_CTRL_S;
```

【成员】

成员名称	描述
enMode	顺序统计量滤波模式

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_ORD\\_STAT\\_FILTER\\_MODE\\_E](#)

## IVE\_MAP\_LUT\_MEM\_S

【说明】

定义 Map 算子的查找表内存信息。

【定义】

```
typedef struct hiIVE_MAP_LUT_MEM_S
{
    HI_U8 au8Map[IVE_MAP_NUM];
} IVE_MAP_LUT_MEM_S;
```

【成员】

成员名称	描述
au8Map[IVE_MAP_NUM]	Map 查找表数组。



【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_MAP\_U8BIT\_LUT\_MEM\_S

【说明】

定义 Map U8C1→U8C1 的查找表内存。

【定义】

```
typedef struct hiIVE_MAP_U8BIT_LUT_MEM_S
{
    HI_U8  au8Map[IVE_MAP_NUM];
}IVE_MAP_U8BIT_LUT_MEM_S;
```

【成员】

成员名称	描述
au8Map[IVE_MAP_NUM]	Map 查找表数组。

【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_MAP\_U16BIT\_LUT\_MEM\_S

【说明】

定义 Map U8C1→U16C1 的查找表内存。

【定义】

```
typedef struct hiIVE_MAP_U16BIT_LUT_MEM_S
{
    HI_U16  au16Map[IVE_MAP_NUM];
}IVE_MAP_U16BIT_LUT_MEM_S;
```

【成员】

成员名称	描述
au16Map[IVE_MAP_NUM]	Map 查找表数组。



【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_MAP\_S16BIT\_LUT\_MEM\_S

【说明】

定义 Map U8C1→S16C1 的查找表内存。

【定义】

```
typedef struct hiIVE_MAP_S16BIT_LUT_MEM_S
{
    HI_S16  as16Map[IVE_MAP_NUM];
} IVE_MAP_S16BIT_LUT_MEM_S;
```

【成员】

成员名称	描述
as16Map[IVE_MAP_NUM]	Map 查找表数组。

【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_MAP\_MODE\_E

【说明】

定义 Map 的模式。

【定义】

```
typedef enum hiIVE_MAP_MODE_E
{
    IVE_MAP_MODE_U8   = 0x0,
    IVE_MAP_MODE_S16  = 0x1,
    IVE_MAP_MODE_U16  = 0x2,
    IVE_MAP_MODE_BUTT
} IVE_MAP_MODE_E;
```



【成员】

成员名称	描述
IVE_MAP_MODE_U8	U8C1→U8C1 Map 模式。
IVE_MAP_MODE_S16	U8C1→U16C1 Map 模式。
IVE_MAP_MODE_U16	U8C1→S16C1 Map 模式

【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_MAP\_CTRL\_S

【说明】

定义 Map 控制参数。

【定义】

```
typedef struct hiIVE_MAP_CTRL_S
{
    IVE_MAP_MODE_E enMode;
} IVE_MAP_CTRL_S;
```

【成员】

成员名称	描述
enMode	Map 模式。

【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_EQUALIZE\_HIST\_CTRL\_MEM\_S

【说明】

定义直方图均衡化辅助内存。

【定义】



```
typedef struct hiIVE_EQUALIZE_HIST_CTRL_MEM_S
{
    HI_U32  au32Hist[IVE_HIST_NUM];
    HI_U8   au8Map[IVE_MAP_NUM];
}IVE_EQUALIZE_HIST_CTRL_MEM_S;
```

#### 【成员】

成员名称	描述
au32Hist[IVE_HIST_NUM]	直方图统计的输出。
au8Map[IVE_MAP_NUM]	根据统计直方图计算得到的 map 查找表。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_EQUALIZE\\_HIST\\_CTRL\\_S](#)

## IVE\_EQUALIZE\_HIST\_CTRL\_S

#### 【说明】

定义直方图均衡化控制参数。

#### 【定义】

```
typedef struct hiIVE_EQUALIZE_HIST_CTRL_S
{
    IVE_MEM_INFO_S stMem;
}IVE_EQUALIZE_HIST_CTRL_S;
```

#### 【成员】

成员名称	描述
stMem	需开辟 sizeof( <a href="#">IVE_EQUALIZE_HIST_CTRL_MEM_S</a> )字节大小的内存。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_EQUALIZE\\_HIST\\_CTRL\\_MEM\\_S](#)



## IVE\_ADD\_CTRL\_S

### 【说明】

定义两图像的加权加控制参数。

### 【定义】

```
typedef struct hiIVE_ADD_CTRL_S
{
    HI_U0Q16 u0q16X;          /*x of "xA+yB"*/
    HI_U0Q16 u0q16Y;          /*y of "xA+yB"*/
} IVE_ADD_CTRL_S;
```

### 【成员】

成员名称	描述
u0q16X	加权加“xA+yB”中的权重“x”。 取值范围：[1, 65535]。
u0q16Y	加权加“xA+yB”中的权重“y”。 取值范围：{65536 - u0q16X}。

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## IVE\_NCC\_DST\_MEM\_S

### 【说明】

定义 NCC 的输出内存信息。

### 【定义】

```
typedef struct hiIVE_NCC_DST_MEM_S
{
    HI_U64 u64Numerator;
    HI_U64 u64QuadSum1;
    HI_U64 u64QuadSum2;
} IVE_NCC_DST_MEM_S;
```

### 【成员】



成员名称	描述
u64Numerator	NCC 计算公式的分子-- $\sum_{i=1}^w \sum_{j=1}^h (I_{src1}(i, j) * I_{src2}(i, j))$
u64QuadSum1	NCC 计算公式的分母--根号内部分: $\sum_{i=1}^w \sum_{j=1}^h (I_{src1}^2(i, j))。$
u64QuadSum2	NCC 计算公式的分母--根号内部分: $\sum_{i=1}^w \sum_{j=1}^h (I_{src2}^2(i, j))$

**【注意事项】**

计算公式请参见 [HI\\_MPI\\_IVE\\_NCC](#) 中的**【注意】**。

**【相关数据类型及接口】**

无。

## IVE\_REGION\_S

**【说明】**

定义连通区域信息。

**【定义】**

```
typedef struct hiIVE_REGION_S
{
    HI_U32 u32Area;           /*Represented by the pixel number*/
    HI_U16 u16Left;           /*Circumscribed rectangle left border*/
    HI_U16 u16Right;          /*Circumscribed rectangle right border*/
    HI_U16 u16Top;            /*Circumscribed rectangle top border*/
    HI_U16 u16Bottom;         /*Circumscribed rectangle bottom border*/
} IVE_REGION_S;
```

**【成员】**

成员名称	描述
u32Area	连通区域面积，以连通区域像素数目表示。
u16Left	连通区域外接矩形的最左边坐标。
u16Right	连通区域外接矩形的最右边坐标。
u16Top	连通区域外接矩形的最上面坐标。





成员名称	描述
u16Bottom	连通区域外接矩形的最下面坐标。

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_CCBLOB\\_S](#)

## IVE\_CCBLOB\_S

【说明】

定义连通区域标记的输出信息。

【定义】

```
typedef struct hiIVE_CCBLOB_S
{
    HI_U16 u16CurAreaThr; /*Threshold of the result regions' area*/
    HI_S8 s8LabelStatus; /*-1: Labeled failed ; 0: Labeled successfully*/
    HI_U8 u8RegionNum; /*Number of valid region, non-continuous stored*/
    IVE_REGION_S astRegion[IVE_MAX_REGION_NUM]; /*Valid regions with
'u32Area>0' and 'label = ArrayIndex+1'*/
}IVE_CCBLOB_S;
```

【成员】

成员名称	描述
u16CurAreaThr	有效连通区域的面积阈值，astRegion 中面积小于这个阈值的都被置为 0。
s8LabelStatus	连通区域标记是否成功。 -1：标记失败； 0：标记成功。
u8RegionNum	有效连通区域个数。
astRegion[IVE_MAX_REGION_NUM]	连通区域信息：有效的连通区域其面积大于 0，对应标记为数组下标加 1。

【注意事项】

无。

【相关数据类型及接口】



## IVE\_REGION\_S

### IVE\_CCL\_MODE\_E

#### 【说明】

定义连通区域模式。

#### 【定义】

```
typedef enum hiIVE_CCL_MODE_E
{
    IVE_CCL_MODE_4C = 0x0, /*4-connectivity*/
    IVE_CCL_MODE_8C = 0x1, /*8-connectivity*/
    IVE_CCL_MODE_BUTT
} IVE_CCL_MODE_E;
```

#### 【成员】

成员名称	描述
IVE_CCL_MODE_4C	4-连通。
IVE_CCL_MODE_8C	8-连通。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

无。

### IVE\_CCL\_CTRL\_S

#### 【说明】

定义连通区域标记控制参数。

#### 【定义】

#### 【Hi3519 定义】

```
typedef struct hiIVE_CCL_CTRL_S
{
    IVE_CCL_MODE_E    enMode;    /*Mode*/
    HI_U16             ul6InitAreaThr; /*Init threshold of region area*/
    HI_U16             ul6Step;    /*Increase area step for once*/
} IVE_CCL_CTRL_S;
```

#### 【其他芯片定义】

```
typedef struct hiIVE_CCL_CTRL_S
```



```
{  
    HI_U16 u16InitAreaThr;    /*Init threshold of region area*/  
    HI_U16 u16Step;           /*Increase area step for once*/  
}IVE_CCL_CTRL_S;
```

#### 【成员】

成员名称	描述
enMode	连通区域模式。
u16InitAreaThr	初始面积阈值。 取值范围：[0, 65535]。 参考取值：4。
u16Step	面积阈值增长步长。 取值范围：[1, 65535]。 参考取值：2。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_CCBLOB\\_S](#)

## IVE\_GMM\_CTRL\_S

#### 【说明】

定义 GMM 背景建模的控制参数。

#### 【定义】

```
typedef struct hiIVE_GMM_CTRL_S  
{  
    HI_U22Q10    u22q10NoiseVar;    /*Initial noise Variance*/  
    HI_U22Q10    u22q10MaxVar;      /*Max Variance*/  
    HI_U22Q10    u22q10MinVar;      /*Min Variance*/  
    HI_U0Q16     u0q16LearnRate;     /*Learning rate*/  
    HI_U0Q16     u0q16BgRatio;       /*Background ratio*/  
    HI_U8Q8      u8q8VarThr;         /*Variance Threshold*/  
    HI_U0Q16     u0q16InitWeight;    /*Initial Weight*/  
    HI_U8         u8ModelNum;        /*Model number: 3 or 5*/  
}IVE_GMM_CTRL_S;
```

#### 【成员】



成员名称	描述
u22q10NoiseVar	初始噪声方差。 取值范围: [0x1, 0xFFFFFFFF]。 对灰度的 GMM, 对应 OpenCV MOG 中灰度模型中的 noiseSigma * noiseSigma。 参考取值: 15*15*(1<<10)。 对 RGB 的 GMM, 对应 OpenCV MOG 中 RGB 模型中的 3 * noiseSigma * noiseSigma。 参考取值: 3*15*15*(1<<10)。
u22q10MaxVar	模型方差的最大值。 取值范围: [0x1, 0xFFFFFFFF]。 对应 OpenCV MOG2 中 fVarMax。 参考取值: 3*4000<<10 (RGB), 2000<<10 (灰度)。
u22q10MinVar	模型方差的最小值。 取值范围: [0x1, u22q10MaxVar]。 对应 OpenCV MOG2 中 fVarMin。 参考取值: 600<<10 (RGB), 200<<10 (灰度)。
u0q16LearnRate	学习速率。 取值范围: [1, 65535]。 对应 OpenCV MOG2 中 learningRate。 参考取值: if (frameNum<500) (1/frameNum)*((1<<16)-1); else ((1/500)*((1<<16)-1))。
u0q16BgRatio	背景比例阈值。 取值范围: [1, 65535]。 对应 OpenCV MOG 中 backgroundRatio。 参考取值: 0.8*((1<<16)-1)。
u8q8VarThr	方差阈值。 取值范围: [1, 65535]。 对应 OpenCV MOG 中 varThreshold, 用于决定一个像素是否命中当前模型。 参考取值: 6.25*(1<<8)。
u0q16InitWeight	初始权重。 取值范围: [1, 65535]。 对应 OpenCV MOG 中的 defaultInitialWeight。 参考取值: 0.05*((1<<16)-1)。



成员名称	描述
u8ModelNum	模型个数。 取值范围：{3,5}。 对应 OpenCV MOG 中 nmixtures。

【注意事项】

无。

【相关数据类型及接口】

无

## IVE\_GMM2\_SNS\_FACTOR\_MODE\_E

【说明】

定义灵敏度系数模式。

【定义】

```
typedef enum hiIVE_GMM2_SNS_FACTOR_MODE_E
{
    IVE_GMM2_SNS_FACTOR_MODE_GLB = 0x0,    /*Global sensitivity factor
mode*/
    IVE_GMM2_SNS_FACTOR_MODE_PIX = 0x1,    /*Pixel sensitivity factor
mode*/
    IVE_GMM2_SNS_FACTOR_MODE_BUTT
} IVE_GMM2_SNS_FACTOR_MODE_E;
```

【成员】

成员名称	描述
IVE_GMM2_SNS_FACTOR_MODE_GLB	全局灵敏度系数模式，每个像素在模型匹配过程中，方差灵敏度使用 IVE_GMM2_CTRL_S 的 u8GlbSnsFactor。
IVE_GMM2_SNS_FACTOR_MODE_PIX	像素级灵敏度系数模式，每个像素在模型匹配过程中，方差灵敏度使用 pstFactor 的灵敏度系数。

【注意事项】

无。

【相关数据类型及接口】



无。

## IVE\_GMM2\_LIFE\_UPDATE\_FACTOR\_MODE\_E

### 【说明】

定义模型时长参数更新模式。

### 【定义】

```
typedef enum hiIVE_GMM2_LIFE_UPDATE_FACTOR_MODE_E
{
    IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_GLB = 0x0, /*Global life update
factor mode*/
    IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_PIX = 0x1, /*Pixel life update
factor mode*/
    IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_BUTT
} IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_E ;
```

### 【成员】

成员名称	描述
IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_GLB	模型时长参数全局更新模式，每个像素模型时长参数在更新时使用 IVE_GMM2_CTRL_S 的 u16GlbLifeUpdateFactor。
IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_PIX	模型时长参数像素级更新模式，每个像素模型时长在更新时使用 pstFactor 的模型更新参数。

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## IVE\_GMM2\_CTRL\_S

### 【说明】

定义 GMM2 背景建模的控制参数。

### 【定义】

```
typedef struct hiIVE_GMM2_CTRL_S
{
    IVE_GMM2_SNS_FACTOR_MODE_E enSnsFactorMode; /*Sensitivity
```



```
factor mode*/
    IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_E enLifeUpdateFactorMode; /*Life
update factor mode*/
    HI_U16      u16GlbLifeUpdateFactor; /*Global life update factor
(default: 4)*/
    HI_U16      u16LifeThr; /*Life threshold (default: 5000)*/
    HI_U16      u16FreqInitVal; /*Initial frequency (default: 20000)*/
    HI_U16      u16FreqReduFactor; /*Frequency reduction factor (default:
0xFF00)*/
    HI_U16      u16FreqAddFactor; /*Frequency adding factor (default:
0xEF)*/
    HI_U16      u16FreqThr; /*Frequency threshold (default: 12000)*/
    HI_U16      u16VarRate; /*Variation update rate (default: 1)*/
    HI_U9Q7     u9q7MaxVar; /*Max variation (default: (16 * 16)<<7)*/
    HI_U9Q7     u9q7MinVar; /*Min variation (default: (8 * 8)<<7)*/
    HI_U8      u8GlbSnsFactor; /*Global sensitivity factor (default: 8)*/
    HI_U8      u8ModelNum; /*Model number (range: 1~5, default: 3)*/
} IVE_GMM2_CTRL_S;
```

#### 【成员】

成员名称	描述
enSnsFactorMode	灵敏度模式，默认全局模式。全局模式使用 u8GlbSnsFactor 作为灵敏度系数；像素模式使用 HI_MPI_IVE_GMM2 中 pstFactor 的低 8bit 值作为灵敏度系数。
enLifeUpdateFactorMode	模型时长更新模式，默认全局模式。全局模式使用 u16GlbLifeUpdateFactor 作为前进模型更新参数；像素模式使用 HI_MPI_IVE_GMM2 中 pstFactor 的高 8bit 值作为前进模型时长更新参数。
u16GlbLifeUpdateFactor	全局模型更新参数，范围：0-65535，默认：4。
u16LifeThr	背景模型生成时间，表示一个模型从前景模型转成背景模型需要的时间，范围：0-65535，默认：5000。
u16FreqInitVal	初始频率，范围：0-65535，默认：20000。
u16FreqReduFactor	频率衰减系数，范围：0-65535，默认：0xFF00。
u16FreqAddFactor	模型匹配频率增加系数，范围：0-65535，默认：0xEF。
u16FreqThr	模型失效频率阈值，范围：0-65535，默认：12000。
u16VarRate	方差更新率，范围：0-65535，默认：1。
u9q7MaxVar	方差最大值，范围：0-65535，默认：(16x16)<<7。
u9q7MinVar	方差最小值，范围：0-方差最大值，默认：(8x8)<<7。



成员名称	描述
u8GlbSnsFactor	全局灵敏度参数，范围：0-255，默认：8。
u8ModelNum	模型数量，范围 1-5，默认：3。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_CANNY\_STACK\_SIZE\_S

**【说明】**

定义 Canny 边缘前半部分计算时强边缘点栈大小结构体。

**【定义】**

```
typedef struct hiIVE_CANNY_STACK_SIZE_S
{
    HI_U32 u32StackSize; /*Stack size for output*/
    HI_U8 u8Reserved[12]; /*For 16 byte align*/
}IVE_CANNY_STACK_SIZE_S;
```

**【成员】**

成员名称	描述
u32StackSize	栈大小(强边缘点的个数)。
u8Reserved[12]	保留位。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_CANNY\_HYS\_EDGE\_CTRL\_S

**【说明】**

定义 Canny 边缘前半部分计算任务的控制参数。

**【定义】**





```
typedef struct hiIVE_CANNY_HYS_EDGE_CTRL_S
{
    IVE_MEM_INFO_S stMem;
    HI_U16 u16LowThr;
    HI_U16 u16HighThr;
    HI_S8 as8Mask[25];
} IVE_CANNY_HYS_EDGE_CTRL_S;
```

#### 【成员】

成员名称	描述
stMem	辅助内存。内存配置大小说明见 <a href="#">HI_MPI_IVE_CannyHysEdge</a> 的【注意】。
u16LowThr	低阈值。 取值范围：[0,255]。
u16HighThr	高阈值。 取值范围：[u16LowThr,255]。
as8Mask[25]	用于计算梯度的参数模板。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

无。

## IVE\_LBP\_CMP\_MODE\_E

#### 【说明】

定义 LBP 计算的比较模式。

#### 【定义】

```
typedef enum hiIVE_LBP_CMP_MODE_E
{
    IVE_LBP_CMP_MODE_NORMAL = 0x0, /* P(x)-P(center)>= un8BitThr.s8Val,
s(x)=1; else s(x)=0; */
    IVE_LBP_CMP_MODE_ABS = 0x1, /* abs(P(x)-
P(center))>=un8BitThr.u8Val, s(x)=1; else s(x)=0; */
    IVE_LBP_CMP_MODE_BUTT
} IVE_LBP_CMP_MODE_E;
```

#### 【成员】



成员名称	描述
IVE_LBP_CMP_MODE_NORMAL	LBP 简单比较模式。
IVE_LBP_CMP_MODE_ABS	LBP 绝对值比较模式。

【注意事项】

计算公式参考 [HI\\_MPI\\_IVE\\_LBP](#) 中的【注意】，示意图请参考图 2-16。

【相关数据类型及接口】

[IVE\\_LBP\\_CTRL\\_S](#)

## IVE\_LBP\_CTRL\_S

【说明】

定义 LBP 纹理计算控制参数。

【定义】

```
typedef struct hiIVE_LBP_CTRL_S
{
    IVE_LBP_CMP_MODE_E enMode;
    IVE_8BIT_U un8BitThr;
} IVE_LBP_CTRL_S;
```

【成员】

成员名称	描述
enMode	LBP 比较模式。
un8BitThr	LBP 比较阈值。 IVE_LBP_CMP_MODE_NORMAL 下的取值范围：[-128,127]。 IVE_LBP_CMP_MODE_ABS 下的取值范围：[0,255]。

【注意事项】

计算公式参考 [HI\\_MPI\\_IVE\\_LBP](#) 中的【注意】，示意图请参考图 2-16。

【相关数据类型及接口】

- [IVE\\_LBP\\_CMP\\_MODE\\_E](#)
- [IVE\\_8BIT\\_U](#)

## IVE\_NORM\_GRAD\_OUT\_CTRL\_E

【说明】



定义归一化梯度信息计算任务输出控制枚举类型。

【定义】

```
typedef enum hiIVE_NORM_GRAD_OUT_CTRL_E
{
    IVE_NORM_GRAD_OUT_CTRL_HOR_AND_VER = 0x0,
    IVE_NORM_GRAD_OUT_CTRL_HOR         = 0x1,
    IVE_NORM_GRAD_OUT_CTRL_VER         = 0x2,
    IVE_NORM_GRAD_OUT_CTRL_COMBINE     = 0x3,
    IVE_NORM_GRAD_OUT_CTRL_BUTT
} IVE_NORM_GRAD_OUT_CTRL_E;
```

【成员】

成员名称	描述
IVE_NORM_GRAD_OUT_CTRL_HOR_AND_VER	同时输出梯度信息的 H、V 分量图(H、V 定义见 HI_MPI_IVE_NormGrad 的【参数】)。
IVE_NORM_GRAD_OUT_CTRL_HOR	仅输出梯度信息的 H 分量图。
IVE_NORM_GRAD_OUT_CTRL_VER	仅输出梯度信息的 V 分量图。
IVE_NORM_GRAD_OUT_CTRL_COMBINE	输出梯度信息以 package 存储（如图 1-7）的 HV 图。

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_NORM\\_GRAD\\_CTRL\\_S](#)

## IVE\_NORM\_GRAD\_CTRL\_S

【说明】

定义归一化梯度信息计算控制参数。

【定义】

```
typedef struct hiIVE_NORM_GRAD_CTRL_S
{
    IVE_NORM_GRAD_OUT_CTRL_E enOutCtrl;
    HI_S8 as8Mask[25];
    HI_U8 u8Norm;
} IVE_NORM_GRAD_CTRL_S;
```



【成员】

成员名称	描述
enOutCtrl	梯度信息输出控制模式。
as8Mask[25]	计算梯度需要的模板。
u8Norm	归一化参数。 取值范围：[1,13]。

【注意事项】

无。

【相关数据类型及接口】

[IVE\\_NORM\\_GRAD\\_OUT\\_CTRL\\_E](#)

## IVE\_MV\_S9Q7\_S

【说明】

定义 LK 光流位移结构体。

【定义】

```
typedef struct hiIVE_MV_S9Q7_S
{
    HI_S32      s32Status;    /*Result of tracking: 0-success; -1-failure*/
    HI_S9Q7     s9q7Dx;      /*X-direction component of the movement*/
    HI_S9Q7     s9q7Dy;      /*Y-direction component of the movement*/
} IVE_MV_S9Q7_S;
```

【成员】

成员名称	描述
s32Status	特征点跟踪的状态。 0：成功； 1：失败。
s9q7Dx	特征点位移的 x 分量。
s9q7Dy	特征点位移的 y 分量。

【注意事项】

无。

【相关数据类型及接口】



无

## IVE\_LK\_OPTICAL\_FLOW\_CTRL\_S

### 【说明】

定义 LK 光流计算控制参数。

### 【定义】

```
typedef struct hiIVE_LK_OPTICAL_FLOW_CTRL_S
{
    HI_U16  u16CornerNum;    /*Number of the feature points,<200*/
    HI_U0Q8  u0q8MinEigThr; /*Minimum eigenvalue threshold*/
    HI_U8    u8IterCount;    /*Maximum iteration times*/
    HI_U0Q8  u0q8Epsilon;    /*Threshold of iteration for  $dx^2 + dy^2 < u0q8Epsilon$  */
} IVE_LK_OPTICAL_FLOW_CTRL_S;
```

### 【成员】

成员名称	描述
u16CornerNum	输入的角点\特征点数目。 取值范围：[1,200]。
u0q8MinEigThr	最小特征值阈值。 取值范围：[1,255]。
u8IterCount	最大迭代次数。 取值范围：[1,20]。
u0q8Epsilon	迭代收敛条件： $dx^2 + dy^2 < u0q8Epsilon$ 。 取值范围：[1,255]。 参考取值：2。

### 【注意事项】

无

### 【相关数据类型及接口】

无

## IVE\_LK\_OPTICAL\_FLOW\_PYR\_OUT\_MODE\_E

### 【说明】

定义金字塔 LK 光流计算输出模式。



### 【定义】

```
typedef enum hiIVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_E
{
    IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_NONE    = 0,    /*Output none*/
    IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_STATUS = 1,    /*Output status*/
    IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_BOTH    = 2,    /*Output status and
err*/
    IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_BUTT
}IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_E;
```

### 【成员】

成员名称	描述
IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_NONE	pstStatus 以及 pstErr 均不输出。
IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_STATUS	仅输出 pstStatus。
IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_BOTH	同时输出 pstStatus 和 pstErr。

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## IVE\_LK\_OPTICAL\_FLOW\_PYR\_CTRL\_S

### 【说明】

定义金字塔 LK 光流计算控制参数。

### 【定义】

```
typedef struct hiIVE_LK_OPTICAL_FLOW_PYR_CTRL_S
{
    IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_E enOutMode;
    HI_BOOL    bUseInitFlow;    /*where to use initial flow*/
    HI_U16    u16PtsNum;    /*Number of the feature points,<=500*/
    HI_U8    u8MaxLevel;    /*0<=u8MaxLevel<=3*/
    HI_U0Q8    u0q8MinEigThr;    /*Minimum eigenvalue threshold*/
    HI_U8    u8IterCnt;    /*Maximum iteration times, <=20*/
    HI_U0Q8    u0q8Eps;    /*Used for exit criteria: dx^2 + dy^2 <
u0q8Eps */
}IVE_LK_OPTICAL_FLOW_PYR_CTRL_S;
```



【成员】

成员名称	描述
enOutMode	pstStatus 以及 pstErr 的输出模式控制。
bUseInitFlow	是否使用初始光流计算（pstNextPts 是否需要初始化）： HI_TRUE 表示使用初始光流，HI_FALSE 表示不适用初始光流。
u16PtsNum	pstPrevPts、pstNextPts 中特征点个数，同时也是 pstStatus 和 pstErr 的数组大小。 取值范围：[1, 500]。
u8MaxLevel	u8MaxLevel+1 为金字塔层数相关。 取值范围：[0, 3]，对应金字塔层数[1, 4]。 参考取值：2。
u0q8MinEigThr	最小特征值阈值。 取值范围：[1,255]。
u8IterCnt	最大迭代次数。 取值范围：[1,20]。
u0q8Eps	迭代收敛条件： $dx^2 + dy^2 < u0q8Epsilon$ 。 取值范围：[1, 255]。 参考取值：2。

【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_ST\_MAX\_EIG\_S

【说明】

定义 Shi-Tomas-like 角点计算时最大角点响应值结构体。

【定义】

```
typedef struct hiIVE_ST_MAX_EIG_S
{
    HI_U16 u16MaxEig;           /*Shi-Tomasi second step output MaxEig*/
    HI_U8  u8Reserved[14];     /*For 16 byte align*/
}IVE_ST_MAX_EIG_S;
```



【成员】

成员名称	描述
u16MaxEig	最大角点响应值。
u8Reserved[14]	保留位。

【注意事项】

无

【相关数据类型及接口】

无

## IVE\_ST\_CANDI\_CORNER\_CTRL\_S

【说明】

定义 Shi-Tomas-like 候选角点计算控制参数。

【定义】

```
typedef struct hiIVE_ST_CANDI_CORNER_CTRL_S
{
    IVE_MEM_INFO_S stMem;
    HI_U0Q8 u0q8QualityLevel;
} IVE_ST_CANDI_CORNER_CTRL_S;
```

【成员】

成员名称	描述
stMem	辅助内存。内存配置大小见 <a href="#">HI_MPI_IVE_STCandiCorner</a> 的【注意】。
u0q8QualityLevel	ShiTomas 角点质量控制参数，角点响应值小于“u0q8QualityLevel * 最大角点响应值”的点将直接被确认为非角点。 取值范围：[1,255]。 参考取值：25。

【注意事项】

无。

【相关数据类型及接口】

无





## IVE\_ST\_CORNER\_INFO\_S

### 【说明】

定义 Shi-Tomas-like 角点计算输出的角点信息结构体。

### 【定义】

```
typedef struct hiIVE_ST_CORNER_INFO_S
{
    HI_U16  ul6CornerNum;
    IVE_POINT_U16_S  astCorner[IVE_ST_MAX_CORNER_NUM];
} IVE_ST_CORNER_INFO_S;
```

### 【成员】

成员名称	描述
ul6CornerNum	有效角点数目。
astCorner[IVE_ST_MAX_CORNER_NUM]	角点坐标数组。

### 【注意事项】

无

### 【相关数据类型及接口】

无

## IVE\_ST\_CORNER\_CTRL\_S

### 【说明】

定义 Shi-Tomas-like 角点筛选控制参数。

### 【定义】

```
typedef struct hiIVE_ST_CORNER_CTRL_S
{
    HI_U16  ul6MaxCornerNum;
    HI_U16  ul6MinDist;
} IVE_ST_CORNER_CTRL_S;
```

### 【成员】

成员名称	描述
ul6MaxCornerNum	最大角点数目。 取值范围：[1,200]。



成员名称	描述
u16MinDist	相邻角点最小距离。 取值范围：[1,255]。 参考取值：10。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_SAD\_MODE\_E

**【说明】**

定义 SAD 计算模式。

**【定义】**

```
typedef enum hiIVE_SAD_MODE_E
{
    IVE_SAD_MODE_MB_4X4      = 0x0, /*4x4*/
    IVE_SAD_MODE_MB_8X8      = 0x1, /*8x8*/
    IVE_SAD_MODE_MB_16X16    = 0x2, /*16x16*/
    IVE_SAD_MODE_BUTT
} IVE_SAD_MODE_E;
```

**【成员】**

成员名称	描述
IVE_SAD_MODE_MB_4X4	按 4x4 像素块计算 SAD。
IVE_SAD_MODE_MB_8X8	按 8x8 像素块计算 SAD。
IVE_SAD_MODE_MB_16X16	按 16x16 像素块计算 SAD。

**【注意事项】**

无。

**【相关数据类型及接口】**

[IVE\\_SAD\\_CTRL\\_S](#)



## IVE\_SAD\_OUT\_CTRL\_E

### 【说明】

定义 SAD 输出控制模式。

### 【定义】

```
typedef enum hiIVE_SAD_OUT_CTRL_E
{
    IVE_SAD_OUT_CTRL_16BIT_BOTH= 0x0, /*Output 16 bit sad and thresh*/
    IVE_SAD_OUT_CTRL_8BIT_BOTH = 0x1, /*Output 8 bit sad and thresh*/
    IVE_SAD_OUT_CTRL_16BIT_SAD = 0x2, /*Output 16 bit sad*/
    IVE_SAD_OUT_CTRL_8BIT_SAD = 0x3, /*Output 8 bit sad*/
    IVE_SAD_OUT_CTRL_THRESH = 0x4, /*Output thresh,16 bits sad */
    IVE_SAD_OUT_CTRL_BUTT
}IVE_SAD_OUT_CTRL_E;
```

### 【成员】

成员名称	描述
IVE_SAD_OUT_CTRL_16BIT_BOTH	16 bit SAD 图和阈值化图输出模式。
IVE_SAD_OUT_CTRL_8BIT_BOTH	8 bit SAD 图和阈值化图输出模式。
IVE_SAD_OUT_CTRL_16BIT_SAD	16 bit SAD 图输出模式。
IVE_SAD_OUT_CTRL_8BIT_SAD	8 bit SAD 图输出模式。
IVE_SAD_OUT_CTRL_THRESH	阈值化图输出模式。

### 【注意事项】

无。

### 【相关数据类型及接口】

[IVE\\_SAD\\_CTRL\\_S](#)

## IVE\_SAD\_CTRL\_S

### 【说明】

定义 SAD 控制参数。

### 【定义】

```
typedef struct hiIVE_SAD_CTRL_S
{
    IVE_SAD_MODE_E enMode;
    IVE_SAD_OUT_CTRL_E enOutCtrl;
    HI_U16 u16Thr; /*srcVal <= u16Thr, dstVal = minVal;
```



```
srcVal > u16Thr, dstVal = maxVal.*/  
    HI_U8 u8MinVal;          /*Min value*/  
    HI_U8 u8MaxVal;          /*Max value*/  
}IVE_SAD_CTRL_S;
```

#### 【成员】

成员名称	描述
enMode	SAD 计算模式。
enOutCtrl	SAD 输出控制模式。
u16Thr	对计算的 SAD 图进行阈值化的阈值。
u8MinVal	阈值化不超过 u16Thr 时的取值。
u8MaxVal	阈值化超过 u16Thr 时的取值。

#### 【注意事项】

无

#### 【相关数据类型及接口】

- [IVE\\_SAD\\_MODE\\_E](#)
- [IVE\\_SAD\\_OUT\\_CTRL\\_E](#)

## IVE\_RESIZE\_MODE\_E

#### 【说明】

定义 Resize 的模式。

#### 【定义】

```
typedef enum hiIVE_RESIZE_MODE_E  
{  
    IVE_RESIZE_MODE_LINEAR    = 0x0, /*Bilinear interpolation*/  
    IVE_RESIZE_MODE_AREA      = 0x1, /*Area-based (or super)  
interpolation*/  
    IVE_RESIZE_MODE_BUTT  
}IVE_RESIZE_MODE_E;
```

#### 【成员】

成员名称	描述
IVE_RESIZE_MODE_LINEAR	双向性插值缩放模式。
IVE_RESIZE_MODE_AREA	区域插值缩放模式。



【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_RESIZE\_CTRL\_S

【说明】

定义 Resize 控制参数。

【定义】

```
typedef struct hiIVE_RESIZE_CTRL_S
{
    IVE_RESIZE_MODE_E enMode;
    IVE_MEM_INFO_S stMem;
    HI_U16 u16Num;
} IVE_RESIZE_CTRL_S;
```

【成员】

成员名称	描述
enMode	缩放模式。
stMem	辅助内存。开辟参照 <a href="#">HI_MPI_IVE_Resize</a> 中【注意】。
u16Num	图像数目。

【注意事项】

无。

【相关数据类型及接口】

无。

## IVE\_GRAD\_FG\_MODE\_E

【说明】

定义梯度前景计算模式。

【定义】

```
typedef enum hiIVE_GRAD_FG_MODE_E
{
    IVE_GRAD_FG_MODE_USE_CUR_GRAD = 0x0,
```



```
IVE_GRAD_FG_MODE_FIND_MIN_GRAD = 0x1,  
IVE_GRAD_FG_MODE_BUTT  
}IVE_GRAD_FG_MODE_E;
```

#### 【成员】

成员名称	描述
IVE_GRAD_FG_MODE_USE_CUR_GRAD	当前位置梯度计算模式。
IVE_GRAD_FG_MODE_FIND_MIN_GRAD	周边最小梯度计算模式。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

[IVE\\_GRAD\\_FG\\_CTRL\\_S](#)

## IVE\_GRAD\_FG\_CTRL\_S

#### 【说明】

定义计算梯度前景控制参数。

#### 【定义】

```
typedef struct hiIVE_GRAD_FG_CTRL_S  
{  
    IVE_GRAD_FG_MODE_E enMode;        /*Calculation mode*/  
    HI_U16 u16EdwFactor;                /*Edge width adjustment factor (range: 500  
to 2000; default: 1000)*/  
    HI_U8 u8CrlCoefThr;                 /*Gradient vector correlation  
coefficient threshold (ranges: 50 to 100; default: 80)*/  
    HI_U8 u8MagCrlThr;                 /*Gradient amplitude threshold (range:  
0 to 20; default: 4)*/  
    HI_U8 u8MinMagDiff;                 /*Gradient magnitude difference  
threshold (range: 2 to 8; default: 2)*/  
    HI_U8 u8NoiseVal;                  /*Gradient amplitude noise threshold  
(range: 1 to 8; default: 1)*/  
    HI_U8 u8EdwDark;                   /*Black pixels enable flag (range: 0 (no),  
1 (yes); default: 1)*/  
}IVE_GRAD_FG_CTRL_S;
```

#### 【成员】



成员名称	描述
enMode	梯度前景计算模式。参考 <a href="#">IVE_GRAD_FG_MODE_E</a> 。
u16EdwFactor	边缘宽度调节因子。 取值范围：[500, 2000]。 参考取值：1000。
u8CrlCoefThr	梯度向量相关系数阈值。 取值范围[50, 100]。 参考取值：80。
u8MagCrlThr	梯度幅值阈值。 取值范围[0, 20]。 参考取值：4。
u8MinMagDiff	梯度幅值差值阈值。 取值范围[2, 8]。 参考取值：2。
u8NoiseVal	梯度幅值噪声阈值。 取值范围[1, 8]。 参考取值：1。
u8EdwDark	黑像素使能标志，0 表示不开启，1 表示开启。 参考取值：1。

**【注意事项】**

无。

**【相关数据类型及接口】**

[IVE\\_GRAD\\_FG\\_MODE\\_E](#)

## IVE\_CANDI\_BG\_PIX\_S

**【说明】**

定义候选背景模型数据。

**【定义】**

```
typedef struct hiIVE_CANDI_BG_PIX_S
{
    HI_U8Q4F4 u8q4f4Mean;    /*Candidate background grays value */
    HI_U16 u16StartTime;     /*Candidate Background start time */
    HI_U16 u16SumAccessTime; /*Candidate Background cumulative access time
*/
```



```
HI_U16 u16ShortKeepTime; /*Candidate background short hold time*/
HI_U8 u8ChgCond;          /*Time condition for candidate background
into the changing state*/
HI_U8 u8PotenBgLife; /*Potential background cumulative access time */
}IVE_CANDI_BG_PIX_S;
```

#### 【成员】

成员名称	描述
u8q4f4Mean	候选背景灰度均值，高 12 位为候选背景像素值，低 4 位为状态标识。
u16StartTime	候选背景起始时间。
u16SumAccessTime	候选背景累计访问时间。
u16ShortKeepTime	候选背景短时保持时间。
u8ChgCond	候选背景进入变换状态时间条件。
u8PotenBgLife	潜在背景生命值。

#### 【注意事项】

无

#### 【相关数据类型及接口】

- [IVE\\_WORK\\_BG\\_PIX\\_S](#)
- [IVE\\_BG\\_MODEL\\_PIX\\_S](#)

## IVE\_WORK\_BG\_PIX\_S

#### 【说明】

定义工作背景模型数据。

#### 【定义】

```
typedef struct hiIVE_WORK_BG_PIX_S
{
    HI_U8Q4F4 u8q4f4Mean;          /*0# background grays value */
    HI_U16 u16AccTime;              /*Background cumulative access time */
    HI_U8 u8PreGray;                /*Gray value of last pixel */
    HI_U5Q3 u5q3DiffThr;           /*Differential threshold */
    HI_U8 u8AccFlag;                /*Background access flag */
    HI_U8 u8BgGray[3];             /*1# ~ 3# background grays value */
}IVE_WORK_BG_PIX_S;
```

#### 【成员】





成员名称	描述
u8q4f4Mean	0 号工作背景灰度值，高 12 位为 0 号工作背景像素值，低 4 位为工作背景状态标识。
u16AccTime	用于工作背景有效性检查的背景累计访问时间。
u8PreGray	前一帧像素灰度值。
u5q3DiffThr	差分阈值。
u8AccFlag	工作背景访问标识。
u8BgGray[3]	1~3 号背景灰度值。

【注意事项】

无。

【相关数据类型及接口】

- [IVE\\_CANDI\\_BG\\_PIX\\_S](#)
- [IVE\\_BG\\_MODEL\\_PIX\\_S](#)

## IVE\_BG\_LIFE\_S

【说明】

定义背景生命力数据。

【定义】

```
typedef struct hiIVE_BG_LIFE_S
{
    HI_U8 u8WorkBgLife[3];           /*1# ~ 3# background vitality */
    HI_U8 u8CandiBgLife;             /*Candidate background vitality */
} IVE_BG_LIFE_S;
```

【成员】

成员名称	描述
u8WorkBgLife[3]	1~3 号工作背景生命力。
u8CandiBgLife	候选背景生命力。

【注意事项】

无。

【相关数据类型及接口】



## IVE\_BG\_MODEL\_PIX\_S

### IVE\_BG\_MODEL\_PIX\_S

#### 【说明】

定义背景模型数据。

#### 【定义】

```
typedef struct hiIVE_BG_MODEL_PIX_S
{
    IVE_WORK_BG_PIX_S    stWorkBgPixel;    /*Working background */
    IVE_CANDI_BG_PIX_S    stCandiPixel;    /*Candidate background */
    IVE_BG_LIFE_S         stBgLife;        /*Background vitality */
} IVE_BG_MODEL_PIX_S;
```

#### 【成员】

成员名称	描述
stWorkBgPixel	工作背景数据。
stCandiPixel	候选背景数据。
stBgLife	背景生命力。

#### 【注意事项】

无

#### 【相关数据类型及接口】

- [IVE\\_CANDI\\_BG\\_PIX\\_S](#)
- [IVE\\_WORK\\_BG\\_PIX\\_S](#)
- [IVE\\_BG\\_LIFE\\_S](#)

### IVE\_FG\_STAT\_DATA\_S

#### 【说明】

定义前景状态数据。

#### 【定义】

```
typedef struct hiIVE_FG_STAT_DATA_S
{
    HI_U32 u32PixNum;
    HI_U32 u32SumLum;
} IVE_FG_STAT_DATA_S;
```

#### 【成员】



成员名称	描述
u32PixNum	前景像素数目。
u32SumLum	输入图像的所有像素亮度累加和。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_BG\_STAT\_DATA\_S

**【说明】**

定义背景状态数据。

**【定义】**

```
typedef struct hiIVE_BG_STAT_DATA_S
{
    HI_U32 u32PixNum;
    HI_U32 u32SumLum;
} IVE_BG_STAT_DATA_S;
```

**【成员】**

成员名称	描述
u32PixNum	背景像素数目。
u32SumLum	背景图像的所有像素亮度累加和。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_MATCH\_BG\_MODEL\_CTRL\_S

**【说明】**

定义背景匹配控制参数。

**【定义】**



```
typedef struct hiIVE_MATCH_BG_MODEL_CTRL_S
{
    HI_U32 u32CurFrmNum;    /*Current frame timestamp, in frame units */
    HI_U32 u32PreFrmNum;    /*Previous frame timestamp, in frame units */
    HI_U16 u16TimeThr;      /*Potential background replacement time
threshold (range: 2 to 100 frames; default: 20) */

    HI_U8 u8DiffThrCrlCoef; /*Correlation coefficients between
differential threshold and gray value (range: 0 to 5; default: 0) */
    HI_U8 u8DiffMaxThr;     /*Maximum of background differential threshold
(range: 3 to 15; default: 6) */
    HI_U8 u8DiffMinThr;     /*Minimum of background differential threshold
(range: 3 to 15; default: 4) */
    HI_U8 u8DiffThrInc;     /*Dynamic Background differential threshold
increment (range: 0 to 6; default: 0) */
    HI_U8 u8FastLearnRate;  /*Quick background learning rate (range: 0 to
4; default: 2) */
    HI_U8 u8DetChgRegion;   /*Whether to detect change region (range: 0
(no), 1 (yes); default: 0) */
} IVE_MATCH_BG_MODEL_CTRL_S;
```

#### 【成员】

成员名称	描述
u32CurFrmNum	当前帧时间。
u32PreFrmNum	前一帧时间。 要求 u32PreFrmNum< u32CurFrmNum。
u16TimeThr	潜在背景替换时间阈值。 取值范围：[2,100]。 参考取值：20。
u8DiffThrCrlCoef	差分阈值与灰度值相关系数。 取值范围：[0,5]。 参考取值：0。
u8DiffMaxThr	背景差分阈值调整上限。 取值范围：[3,15]。 参考取值：6。
u8DiffMinThr	背景差分阈值调整下限。 取值范围：[3, u8DiffMaxThr]。 参考取值：4。



成员名称	描述
u8DiffThrInc	动态背景下差分阈值增量。 取值范围：[0,6]。 参考取值：0。
u8FastLearnRate	快速背景学习速率。 取值范围：[0,4]。 参考取值：2。
u8DetChgRegion	是否检测变化区域。 取值范围：{0,1}，0 表示不检测，1 表示检测。 参考取值：0。

【注意事项】

无

【相关数据类型及接口】

无

## IVE\_UPDATE\_BG\_MODEL\_CTRL\_S

【说明】

定义背景更新控制参数。

【定义】

```
typedef struct hiIVE_UPDATE_BG_MODEL_CTRL_S
{
    HI_U32 u32CurFrmNum;    /*Current frame timestamp, in frame units */
    HI_U32 u32PreChkTime;   /*The last time when background status is
checked */
    HI_U32 u32FrmChkPeriod; /*Background status checking period (range: 0
to 2000 frames; default: 50) */
    HI_U32 u32InitMinTime;  /*Background initialization shortest time
(range: 20 to 6000 frames; default: 100)*/
    HI_U32 u32StyBgMinBlendTime; /*Steady background integration shortest
time (range: 20 to 6000 frames; default: 200)*/
    HI_U32 u32StyBgMaxBlendTime; /*Steady background integration longest
time (range: 20 to 40000 frames; default: 1500)*/
    HI_U32 u32DynBgMinBlendTime; /*Dynamic background integration shortest
time (range: 0 to 6000 frames; default: 0)*/
    HI_U32 u32StaticDetMinTime; /*Still detection shortest time (range: 20
to 6000 frames; default: 80)*/
}
```



```

    HI_U16 u16FgMaxFadeTime;    /*Foreground disappearing longest time
(range: 1 to 255 seconds; default: 15)*/
    HI_U16 u16BgMaxFadeTime; /*Background disappearing longest time
(range: 1 to 255 seconds ; default: 60)*/
    HI_U8 u8StyBgAccTimeRateThr; /*Steady background access time ratio
threshold (range: 10 to 100; default: 80)*/
    HI_U8 u8ChgBgAccTimeRateThr; /*Change background access time ratio
threshold (range: 10 to 100; default: 60)*/
    HI_U8 u8DynBgAccTimeThr; /*Dynamic background access time ratio
threshold (range: 0 to 50; default: 0)*/
    HI_U8 u8DynBgDepth; /*Dynamic background depth (range: 0
to 3; default: 3)*/
    HI_U8 u8BgEffStaRateThr; /*Background state time ratio
threshold when initializing (range: 90 to 100; default: 90)*/
    HI_U8 u8AcceBgLearn; /*Whether to accelerate background
learning (range: 0 (no), 1 (yes); default: 0)*/
    HI_U8 u8DetChgRegion; /*Whether to detect change region
(range: 0 (no), 1 (yes); default: 0)*/
} IVE_UPDATE_BG_MODEL_CTRL_S;

```

**【成员】**

成员名称	描述
u32CurFrmNum	当前帧时间。
u32PreChkTime	上次进行背景状态检查的时间点。
u32FrmChkPeriod	背景状态检查时间周期。 取值范围：[0, 2000]。 参考取值：50。
u32InitMinTime	背景初始化最短时间。 取值范围：[20, 6000]。 参考取值：100。
u32StyBgMinBlendTime	稳态背景融入最短时间。 取值范围：[u32InitMinTime, 6000]。 参考取值：200。
u32StyBgMaxBlendTime	稳态背景融入最长时间。 取值范围：[u32StyBgMinBlendTime, 40000]。 参考取值：1500。



成员名称	描述
u32DynBgMinBlendTime	动态背景融入最短时间。 取值范围：[0, 6000]。 参考取值：0。
u32StaticDetMinTime	静物检测最短时间。 取值范围：[20, 6000]。 参考取值：80。
u16FgMaxFadeTime	前景持续消失最长时间。 取值范围：[1, 255]。 参考取值：15。
u16BgMaxFadeTime	背景持续消失最长时间。 取值范围：[1, 255]。 参考取值：60。
u8StyBgAccTimeRateThr	稳态背景访问时间比率阈值。 取值范围：[10, 100]。 参考取值：80。
u8ChgBgAccTimeRateThr	变化背景访问时间比率阈值。 取值范围：[10, 100]。 参考取值：60。
u8DynBgAccTimeThr	动态背景访问时间比率阈值。 取值范围：[0, 50]。 参考取值：0。
u8DynBgDepth	动态背景深度。 取值范围：[0, 3]。 参考取值：3。
u8BgEffStaRateThr	景初始化阶段背景状态时间比率阈值。 取值范围：[90, 100]。 参考取值：90。
u8AcceBgLearn	是否加速背景学习。 取值范围：{0, 1}，0 表示不加速，1 表示加速。 参考取值：0。
u8DetChgRegion	是否检测变化区域。 取值范围：{0, 1}，0 表示不检测，1 表示检测。 参考取值：0。



【注意事项】

要求  $u32InitMinTime \leq u32StyBgMinBlendTime \leq u32StyBgMaxBlendTime$ 。

【相关数据类型及接口】

无。

## IVE\_LOOK\_UP\_TABLE\_S

【说明】

定义查找表结构体。

【定义】

```
typedef struct hiIVE_LOOK_UP_TABLE_S
{
    IVE_MEM_INFO_S stTable;
    HI_U16 u16ElemNum;          /*LUT's elements number*/
    HI_U8 u8TabInPreci;
    HI_U8 u8TabOutNorm;
    HI_S32 s32TabInLower;       /*LUT's original input lower limit*/
    HI_S32 s32TabInUpper;       /*LUT's original input upper limit*/
} IVE_LOOK_UP_TABLE_S;
```

【成员】

成员名称	描述
stTable	查找表建立后的数据内存块信息。
u32ElemNum	查找表元素个数。
s32TabInLower	建立查找表的数值范围的下限。
s32TabInUpper	建立查找表的数值范围的上限。
u8TabInPreci	建立查找表的精度， $(s32TabInUpper - s32TabInLower) / (1 << u8TabInPreci)$ 表示建立查找表的间隔。
u8TabOutNorm	表示建立查找表时为将原始数据归一化进行移位的位数或者进行除法的除数。

【注意事项】

无。

【相关数据类型及接口】

无。





## IVE\_ANN\_MLP\_ACCURATE\_E

### 【说明】

定义 ANN\_MLP 输入特征向量类型。

### 【定义】

```
typedef enum hiIVE_ANN_MLP_ACCURATE_E
{
    IVE_ANN_MLP_ACCURATE_SRC16_WGT16    = 0x0, /*input decimals' accurate
16 bit, weight 16bit*/
    IVE_ANN_MLP_ACCURATE_SRC14_WGT20    = 0x1, /*input decimals' accurate
14 bit, weight 20bit*/
    IVE_ANN_MLP_ACCURATE_BUTT
}IVE_ANN_MLP_ACCURATE_E;
```

### 【成员】

成员名称	描述
IVE_ANN_MLP_ACCURATE_SRC16_WGT16	输入特征向量类型为 SQ16.16。
IVE_ANN_MLP_ACCURATE_SRC14_WGT20	输入特征向量类型为 SQ18.14。

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## IVE\_ANN\_MLP\_ACTIV\_FUNC\_E

### 【说明】

定义 ANN\_MLP 激活函数枚举类型。

### 【定义】

```
typedef enum hiIVE_ANN_MLP_ACTIV_FUNC_E
{
    IVE_ANN_MLP_ACTIV_FUNC_IDENTITY      = 0x0,
    IVE_ANN_MLP_ACTIV_FUNC_SIGMOID_SYM   = 0x1,
    IVE_ANN_MLP_ACTIV_FUNC_GAUSSIAN      = 0x2,
    IVE_ANN_MLP_ACTIV_FUNC_BUTT
}IVE_ANN_MLP_ACTIV_FUNC_E;
```

### 【成员】



成员名称	描述
IVE_ANN_MLP_ACTIV_FUNC_IDENTITY	Identity 激活函数。
IVE_ANN_MLP_ACTIV_FUNC_SIGMOID_SYM	Sigmoid 对称激活函数。
IVE_ANN_MLP_ACTIV_FUNC_GAUSSIAN	Gaussian 激活函数。

【注意事项】

对应函数定义参见 [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_Predict](#) 中【注意】。

【相关数据类型及接口】

无。

## IVE\_ANN\_MLP\_MODEL\_S

【说明】

定义 ANN\_MLP 模型数据结构体。

【定义】

【Hi3519 定义】

```
typedef struct hiIVE_ANN_MLP_MODEL_S
{
    IVE_ANN_MLP_ACTIV_FUNC_E enActivFunc;
    IVE_ANN_MLP_ACCURATE_E enAccurate;
    IVE_MEM_INFO_S stWeight;
    HI_U32 u32TotalWeightSize;
    HI_U16 au16LayerCount[8]; /*8 layers, including input and output
layer, every layerCount<=256*/
    HI_U16 u16MaxCount; /*MaxCount<=256*/
    HI_U8 u8LayerNum; /*2<layerNum<=8*/
    HI_U8 u8Reserve;
}IVE_ANN_MLP_MODEL_S;
```

【其他芯片定义】

```
typedef struct hiIVE_ANN_MLP_MODEL_S
{
    IVE_ANN_MLP_ACTIV_FUNC_E enActivFunc;
    IVE_MEM_INFO_S stWeight;
    HI_U32 u32TotalWeightSize;
    HI_U16 au16LayerCount[8]; /*8 layers, including input and output
layer, every layerCount<=256*/
    HI_U16 u16MaxCount; /*MaxCount<=256*/
    HI_U8 u8LayerNum; /*2<layerNum<=8*/
```



```
} IVE_ANN_MLP_MODEL_S;
```

#### 【成员】

成员名称	描述
enActivFunc	激活函数类型。
enAccurate	输入特征向量类型。
stWeight	模型数据权重。
au16LayerCount[8]	输入层 1 层→隐含层若干层（至少 1 层，最多 6 层）→输出层 1 层，分别存储输入、输出层的特征维度（取值范围：[1, 256]），各隐藏层的神经元数目（取值范围：[2, 256]）。
u16MaxCount	所有层中神经元数目或者特征维度的最大值： $\max_{0 \leq i < u8LayerNum} \{au16LayerCount[i]\}。$
u8LayerNum	ANN_MLP 系统中的层数。 取值范围：[3, 8]。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

- [IVE\\_ANN\\_MLP\\_ACTIV\\_FUNC\\_E](#)
- [IVE\\_ANN\\_MLP\\_ACCURATE\\_E](#)

## IVE\_SVM\_TYPE\_E

#### 【说明】

定义 SVM 类型。

#### 【定义】

```
typedef enum hiIVE_SVM_TYPE_E
{
    IVE_SVM_TYPE_C_SVC    = 0x0,
    IVE_SVM_TYPE_NU_SVC   = 0x1,
    IVE_SVM_TYPE_BUTT
} IVE_SVM_TYPE_E;
```

#### 【成员】



成员名称	描述
IVE_SVM_TYPE_C_SVC	分类模式。
IVE_SVM_TYPE_NU_SVC	回归模式。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_SVM\_KERNEL\_TYPE\_E

**【说明】**

定义 SVM 核函数类型。

**【定义】**

```
typedef enum hiIVE_SVM_KERNEL_TYPE_E
{
    IVE_SVM_KERNEL_TYPE_LINEAR    = 0x0,
    IVE_SVM_KERNEL_TYPE_POLY      = 0x1,
    IVE_SVM_KERNEL_TYPE_RBF       = 0x2,
    IVE_SVM_KERNEL_TYPE_SIGMOID   = 0x3,
    IVE_SVM_KERNEL_TYPE_BUTT
} IVE_SVM_KERNEL_TYPE_E;
```

**【成员】**

成员名称	描述
IVE_SVM_KERNEL_TYPE_LINEAR	线性核函数。
IVE_SVM_KERNEL_TYPE_POLY	多项式核函数。
IVE_SVM_KERNEL_TYPE_RBF	径向基核函数。
IVE_SVM_KERNEL_TYPE_SIGMOID	Sigmoid 核函数。

**【注意事项】**

对应核函数定义参见 [HI\\_MPI\\_IVE\\_SVM\\_Predict](#) 中**【注意】**。

**【相关数据类型及接口】**

无。



## IVE\_SVM\_MODEL\_S

### 【说明】

定义 SVM 模型数据结构体。

### 【定义】

```
typedef struct hiIVE_SVM_MODEL_S
{
    IVE_SVM_TYPE_E enType;
    IVE_SVM_KERNEL_TYPE_E enKernelType;
    IVE_MEM_INFO_S stSv;    /*SV memory*/
    IVE_MEM_INFO_S stDf;    /*Decision functions memory*/
    HI_U32 u32TotalDfSize; /*All decision functions coef size in byte*/
    HI_U16 u16FeatureDim;
    HI_U16 u16SvTotal;
    HI_U8 u8ClassCount;
} IVE_SVM_MODEL_S;
```

### 【成员】

成员名称	描述
enType	SVM 类型，当前仅支持分类 IVE_SVM_TYPE_C_SVC 模式。
enKernelType	SVM 核函数类型，参考 <a href="#">IVE_SVM_KERNEL_TYPE_E</a> 。
stSv	模型数据中的支持向量，其内存排布参看 <a href="#">HI_MPI_IVE_SVM_Predict</a> 的【注意】。
stDf	模型数据中的判决函数参数部分。
u32TotalDfSize	所有判决函数参数总字节数。
u16FeatureDim	输入特征维数。 取值范围：[1, 256]。
u16SvTotal	总支持向量个数。 取值范围：[1, 3000]。
u8ClassCount	类别数目。 取值范围：[2, 80]。

### 【注意事项】

无。

### 【相关数据类型及接口】



- [IVE\\_SVM\\_TYPE\\_E](#)
- [IVE\\_SVM\\_KERNEL\\_TYPE\\_E](#)

## IVE\_CNN\_ACTIV\_FUNC\_E

### 【说明】

定义 CNN 激活函数枚举类型。

### 【定义】

```
typedef enum hiIVE_CNN_ACTIV_FUNC_E
{
    IVE_CNN_ACTIV_FUNC_NONE      = 0x0,   /*f(x)=x*/
    IVE_CNN_ACTIV_FUNC_RELU      = 0x1,   /*f(x)=max(0, x)*/
    IVE_CNN_ACTIV_FUNC_SIGMOID   = 0x2,   /*f(x)=1/(1+exp(-x)), not support*/
    IVE_CNN_ACTIV_FUNC_BUTT
} IVE_CNN_ACTIV_FUNC_E;
```

### 【成员】

成员名称	描述
IVE_CNN_ACTIV_FUNC_NONE	不使用激活函数。
IVE_CNN_ACTIV_FUNC_RELU	使用 ReLU 激活函数。
IVE_CNN_ACTIV_FUNC_SIGMOID	使用 Sigmoid 激活函数，暂不支持。

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## IVE\_CNN\_POOLING\_E

### 【说明】

定义 CNN 汇聚操作枚举类型。

### 【定义】

```
typedef enum hiIVE_CNN_POOLING_E
{
    IVE_CNN_POOLING_NONE = 0x0, /*Do not taking a pooling action*/
    IVE_CNN_POOLING_MAX  = 0x1, /*Using max value of every pooling area*/
    IVE_CNN_POOLING_AVG  = 0x2, /*Using average value of every pooling
area*/
    IVE_CNN_POOLING_BUTT
}
```



```
} IVE_CNN_POOLING_E;
```

#### 【成员】

成员名称	描述
IVE_CNN_POOLING_NONE	不执行图像汇聚。
IVE_CNN_POOLING_MAX	对汇聚范围的特征点求最大值。
IVE_CNN_POOLING_AVG	对汇聚范围的特征点求平均值。

#### 【注意事项】

无。

#### 【相关数据类型及接口】

无。

## IVE\_CNN\_CONV\_POOLING\_S

#### 【说明】

定义 CNN 单层 Conv-ReLU-Pooling 运算包参数结构体。

#### 【定义】

```
typedef struct hiIVE_CNN_CONV_POOLING_S
{
    IVE_CNN_ACTIV_FUNC_E enActivFunc; /*Type of activation function*/
    IVE_CNN_POOLING_E    enPooling;   /*Mode of pooling method*/
    HI_U8  u8FeatureMapNum; /*Number of feature maps*/
    HI_U8  u8KernelSize;   /*Kernel size, only support 3 currently*/
    HI_U8  u8ConvStep;     /*Convolution step, only support 1
currently*/
    HI_U8  u8PoolSize;     /*Pooling size, only support 2 currently*/
    HI_U8  u8PoolStep;     /*Pooling step, only support 2 currently*/
    HI_U8  u8Reserved[3];
} IVE_CNN_CONV_POOLING_S;
```

#### 【成员】

成员名称	描述
enActivFunc	激活函数类型。
enPooling	汇聚操作类型。
u8FeatureMapNum	卷积运算包中输出特征图的数目。支持范围：1~50。
u8KernelSize	卷积运算包中卷积核的尺寸。仅支持 3x3。



成员名称	描述
u8ConvStep	卷积运算包中卷积核移动的步长。仅支持 1。
u8PoolSize	汇聚操作的窗口大小。仅支持 2x2。
u8PoolStep	汇聚操作窗口移动的步长。仅支持 2。

【注意事项】

无。

【相关数据类型及接口】

- [IVE\\_CNN\\_ACTIV\\_FUNC\\_E](#)
- [IVE\\_CNN\\_POOLING\\_E](#)

## IVE\_CNN\_FULL\_CONNECT\_S

【说明】

定义 CNN 全链接网络参数结构体。

【定义】

```
typedef struct hiIVE_CNN_FULL_CONNECT_S
{
    HI_U16 au16LayerCnt[8];    /*Neuron number of every fully connected
layers*/
    HI_U16 u16MaxCnt;          /*Max neuron number in all fully connected
layers*/
    HI_U8 u8LayerNum;          /*Number of fully connected layer*/
    HI_U8 u8Reserved;
}IVE_CNN_FULL_CONNECT_S;
```

【成员】

成员名称	描述
au16LayerCnt[8]	全连接各层神经元节点数目。输入层（即 Conv-ReLU-Pooling 的输出）支持范围 1~1024；中间隐藏层支持范围：2~256；输出层支持范围：1~256。
u16MaxCnt	全连接各层节点数目的最大值。
u8LayerNum	全连接层数。支持范围：3~8。

【注意事项】

无。





### 【相关数据类型及接口】

无。

## IVE\_CNN\_MODEL\_S

### 【说明】

定义 CNN 模型参数结构体。

### 【定义】

```
typedef struct hiIVE_CNN_MODEL_S
{
    IVE_CNN_CONV_POOLING_S astConvPool[8]; /*Conv-ReLU-Pooling layers
info*/
    IVE_CNN_FULL_CONNECT_S stFullConnect; /*Fully connected layers
info*/
    IVE_MEM_INFO_S stConvKernelBias; /*Conv-ReLU-Pooling layers'
kernels and bias*/
    HI_U32 u32ConvKernelBiasSize; /*Size of Conv-ReLU-Pooling
layer' kernels and bias*/

    IVE_MEM_INFO_S stFCLWgtBias; /*Fully Connection Layers'
weights and bias*/
    HI_U32 u32FCLWgtBiasSize; /*Size of fully connection
layers weights and bias*/
    HI_U32 u32TotalMemSize; /*Total memory size of all
kernels, weights, bias*/
    IVE_IMAGE_TYPE_E enType; /*Image type used for the CNN model*/
    HI_U16 u16Width; /*Image width used for the model*/
    HI_U16 u16Height; /*Image height used for the model*/
    HI_U16 u16ClassCount; /*Number of classes*/
    HI_U8 u8ConvPoolLayerNum; /*Number of Conv-ReLU-Pooling layers*/
    HI_U8 u8Reserved;
}IVE_CNN_MODEL_S;
```

### 【成员】

成员名称	描述
astConvPool[8]	各层卷积运算包参数配置。
stFullConnect	全连接运算包参数配置。
stConvKernelBias	所有卷积运算包中卷积核系数和偏置系数。
u32ConvKernelBiasSize	所有卷积运算包中卷积核系数和偏置系数的字节数。
stFCLWgtBias	全连接运算包中权重系数和偏置系数。



成员名称	描述
u32FCLWgtBiasSize	全连接运算包中权重系数和偏置系数的字节数。
u32TotalMemSize	CNN 网络计算中所需要的包括卷积核、权重、偏置系数等的内存字节数。
enType	输入 CNN 模型的图像类型。仅支持 U8C1 灰度图或者 RGB_PLANAR 彩色图。
u16Width	输入 CNN 模型的图像宽度。w 支持范围：16~80。
u16Height	输入 CNN 模型的图像高度。h 支持范围：16~(1280/w)。
u16ClassCount	CNN 模型分类任务的类别数目（与全连接最后一层输出一致）。支持范围：1~256。
u8ConvPoolLayerNum	CNN 模型卷积运算包的数目。支持范围：1~8。

**【注意事项】**

CNN 网络模型结构参考 [HI\\_MPI\\_IVE\\_CNN\\_Predict](#) 中的**【注意】**。

**【相关数据类型及接口】**

- [IVE\\_CNN\\_ACTIV\\_FUNC\\_E](#)
- [IVE\\_CNN\\_POOLING\\_E](#)
- [IVE\\_CNN\\_CONV\\_POOLING\\_S](#)
- [IVE\\_CNN\\_FULL\\_CONNECT\\_S](#)

## IVE\_CNN\_CTRL\_S

**【说明】**

定义 CNN 预测任务的控制参数。

**【定义】**

```
typedef struct hiIVE_CNN_CTRL_S
{
    IVE_MEM_INFO_S stMem;    /*Assist memory*/
    HI_U32 u32Num;           /*Input image number*/
} IVE_CNN_CTRL_S;
```

**【成员】**

成员名称	描述
stMem	CNN 预测计算过程的辅助内存，所需内存大小参考 <a href="#">HI_MPI_IVE_CNN_Predict</a> 中的 <b>【注意】</b> 。
u32Num	输入 CNN 模型的图像数目。



**【注意事项】**

无。

**【相关数据类型及接口】**

无。

## IVE\_CNN\_RESULT\_S

**【说明】**

定义 CNN 单个样本预测结果结构体。

**【定义】**

```
typedef struct hiIVE_CNN_RESULT_S
{
    HI_S32 s32ClassIdx;    /*The most possible index of the
classification*/
    HI_S32 s32Confidence; /*The confidence of the classification*/
} IVE_CNN_RESULT_S;
```

**【成员】**

成员名称	描述
s32ClassIdx	CNN 模型的预测类别索引。
s32Confidence	CNN 模型所预测类别的置信度。

**【注意事项】**

无。

**【相关数据类型及接口】**

无。



# 4 错误码

## 4.1 IVE 错误码

智能加速引擎 API 错误码如表 4-1 所示。

表4-1 智能加速引擎 API 错误码

错误代码	宏定义	描述
0xA01D8001	HI_ERR_IVE_INVALID_DEVID	设备 ID 超出合法范围
0xA01D8002	HI_ERR_IVE_INVALID_CHNID	通道组号错误或无效区域句柄
0xA01D8003	HI_ERR_IVE_ILLEGAL_PARAM	参数超出合法范围
0xA01D8004	HI_ERR_IVE_EXIST	重复创建已存在的设备、通道或资源
0xA01D8005	HI_ERR_IVE_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
0xA01D8006	HI_ERR_IVE_NULL_PTR	函数参数中有空指针
0xA01D8007	HI_ERR_IVE_NOT_CONFIG	模块没有配置
0xA01D8008	HI_ERR_IVE_NOT_SUPPORT	不支持的参数或者功能
0xA01D8009	HI_ERR_IVE_NOT_PERM	该操作不允许，如试图修改静态配置参数
0xA01D800C	HI_ERR_IVE_NOMEM	分配内存失败，如系统内存不足
0xA01D800D	HI_ERR_IVE_NOBUF	分配缓存失败，如申请的图像缓冲区太大
0xA01D800E	HI_ERR_IVE_BUF_EMPTY	缓冲区中无图像
0xA01D800F	HI_ERR_IVE_BUF_FULL	缓冲区中图像满
0xA01D8010	HI_ERR_IVE_NOTREADY	系统没有初始化或没有加载相应模块



错误代码	宏定义	描述
0xA01D8011	HI_ERR_IVE_BADADDR	地址非法
0xA01D8012	HI_ERR_IVE_BUSY	系统忙
0xA01D8040	HI_ERR_IVE_SYS_TIMEOUT	系统超时
0xA01D8041	HI_ERR_IVE_QUERY_TIMEOUT	Query 查询超时
0xA01D8042	HI_ERR_IVE_OPEN_FILE	打开文件失败
0xA01D8043	HI_ERR_IVE_READ_FILE	读文件失败
0xA01D8044	HI_ERR_IVE_WRITE_FILE	写文件失败



# 5 Proc 调试信息

## 5.1 概述

调试信息采用了 Linux 下的 proc 文件系统，可实时反映当前系统的运行状态，所记录的信息可供问题定位及分析时使用。

### 【文件目录】

/proc/umap

### 【信息查看方法】

- 在控制台上可以使用 cat 命令查看信息，cat /proc/umap/ive; 也可以使用其他常用的文件操作命令，例如 cp /proc/umap/ive ./，将文件拷贝到当前目录。
- 在应用程序中可以将上述文件当作普通只读文件进行读操作，例如 fopen、fread 等。

### 说明

参数在描述时有以下 2 种情况需要注意：

- 取值为 {0, 1} 的参数，如未列出具体取值和含义的对应关系，则参数为 1 时表示肯定，为 0 时表示否定。
- 取值为 {aaa, bbb, ccc} 的参数，未列出具体取值和含义的对应关系，但可直接根据取值 aaa、bbb 或 ccc 判断参数含义

## 5.2 Proc 信息说明

### 【调试信息】

```
~ # cat /proc/umap/ive
```

```
[IVE] Version: [Hi3516A_MPP_V1.0.0.0 B00 Debug], Build Time[Aug 22 2014, 15:00:18]
```

```
-----MODULE PARAM-----
```

```
save_power
```

```
0
```



```

-----IVE QUEUE INFO-----
    Wait      Busy   WaitCurId WaitEndId BusyCurId BusyEndId
      1        -1       0         0         0         0

-----IVE TASK INFO-----
    Hnd    TaskFsh   LastId    TaskId    HndWrap   FshWrap
      5         5         0         0         0         0

-----IVE RUN-TIME INFO-----
LastInst  CntPerSec  MaxCntPerSec  TotalIntCntLastSec  TotalIntCnt  QTCnt
STCnt
      1         1         1             4             5         0
0
CostTm   MCostTm   CostTmPerSec  MCostTmPerSec  IntTotalCostTm  RunTm
      42       641         48         641             833         42

-----IVE INVOKE INFO-----
    DMA      Filter    CSC     FltCsc     Sobel     MagAng    Dilate     Erode
      5         0         0         0         0         0         0         0

    Thresh    And      Sub      Or      Integ     Hist     ThreshS16  ThreshU16
      0         0         0         0         0         0         0         0

    16to8  OrdStatFlt  BernSen     Map     EqualH     Add     Xor     NCC
      0         0         0         0         0         0         0         0

    CCL      GMM      Canny     LBP     NormGrad     LK     ShiTomasi  GradFg
      0         0         0         0         0         0         0         0

    MatchMod UpdateMod   Radon     ANN     SVM     AdpThr   LineFltH
    NoiseRmH
      0         0         0         0         0         0         0         0

    PlateChar   SAD
      0         0

-----IVE UTILI INFO-----
Utili
0

```

### 【调试信息分析】

记录当前 IVE 工作状态资源信息和算子调用信息。

### 【参数说明】



参数		描述
MODULE PARAM 模块 参数	save_power	低功耗标志。0：关闭低功耗；1：打开低功耗。
IVE QUEUE INFO IVE 队列信息	Wait	等待队列编号（0 或 1）。
	Busy	正在调度队列编号（0, 1 或-1），-1 表示 IVE 硬件空闲。
	WaitCurId	等待队列的首个有效任务 ID。
	WaitEndId	等待队列的末尾有效任务 ID + 1。
	BusyCurId	调度队列的首个有效任务 ID。
	BusyEndId	调度队列的末尾有效任务 ID + 1。
IVE TASK INFO IVE TASK 相 关信息	Hnd	当前可分配的任务 handle 号。
	TaskFsh	当前已完成任务的个数。
	LastId	上一次中断完成的任务 ID。
	TaskId	本次中断完成的任务 ID。
	HndWrap	用户 handle 号分配发生回写的次数。
	FshWrap	完成任务数发生回写的次数。
IVE RUN- TIME INFO IVE 运行时相 关信息	LastInst	用户最后一次提交任务时传入的 bInstant 值。
	CntPerSec	最近一次的 1 秒内中断执行次数。
	MaxCntPerSec	历史上的 1 秒内最大的中断执行次数。
	TotalIntCntLastSec	上一秒上报中断总次数。
	TotalIntCnt	IVE 产生中断的总次数。
	QTCnt	查询 IVE 链表超时次数。
	STCnt	IVE 系统超时次数。
	CostTm	最近一次执行中断的执行耗时。 单位：us
	MCostTm	执行一次中断的最大耗时。 单位：us
	CostTmPerSec	最近一秒执行中断的执行耗时。 单位：us
	MCostTmPerSec	历史上一秒执行中断的最大执行耗时。 单位：us





参数		描述
	ntTotalCostTm	中断处理总时间。 单位：us
	RunTm	IVE 运行总时间。 单位：s
IVE INVOKE INFO <b>IVE</b> 调用信息	DMA	DMA 的调用次数。其它不再一一列出。
IVE UTILI INFO <b>IVE</b> 利用率	Utili	IVE 利用率。

【注意】

- 建议代码调试阶段关闭低功耗，调试完成再打开低功耗。
- 关闭低功耗情况下 IVE 占用率统计 Utili 不能使用。



# 6 FAQ

## 6.1 使用 PC 端 IVE Clib 与板端 IVE SDK 开发算法的差异

序号	关键词	PC 端 IVE Clib	板端 SDK
1	handle	无效	当有需要时与 <a href="#">HI_MPI_IVE_Query</a> 配合查询算子是否完成。详细参考 <a href="#">1.2.1 重要概念的“句柄(handle)”</a>
2	bInstant	无效	根据算法设置可减少中断，提升性能。详细参考 <a href="#">1.2.1 重要概念的“及时返回结果标志 bInstant”</a>
3	Query	不需要查询，查询永远返回成功	当用户需要使用 IVE 硬算子的结果时，必须查询任务是否完成。详细参考 <a href="#">1.2.1 重要概念的“查询(query)”</a> 以及 <a href="#">HI_MPI_IVE_Query</a> 接口说明。
4	内存开辟、物理地址、虚拟地址	内存使用 malloc 开辟。由于 malloc 出来均为虚拟地址，所以赋值给虚拟地址。为模拟 IVE 硬件使用物理地址的特性，Clib 也使用物理地址，所以物理地址必须赋值，且必须赋值为虚拟地址的 HI_U32 强制类转化。	IVE 硬件使用物理地址。内存使用 <a href="#">HI_MPI_SYS_MmzMalloc/HI_MPI_SYS_MmzAlloc_Cached</a> 接口详细信息请参见《HiMPP Vx.y 媒体处理软件开发参考》）开辟，物理地址和虚拟地址由此生成；或者使用其他模块的 VB 内存。
5	地址对齐	Clib 不要求地址对齐	硬件地址按要求对齐。
6	芯片差异	Clib 是功能全集，接口会更新到最新版。	芯片根据需求，支持的功能是 Clib 中的子集。某些接口由于升级可能与最新版 Clib 不一致。
7	异步、同步、并行、串行	Clib 的执行与算法软件均在 CPU 中串行执行，不存在异步问题。	IVE 硬件与 CPU 异步执行，可以此特性让 IVE 与 CPU 并行工作，提高性能。但是在 CPU 需要使用 IVE 的结果数据时，需要同步。



## 6.2 使用 IVE 与 OpenCV 开发算法的区别

1. IVE 与 CPU 异步，CPU 必须查询 IVE 任务是否完成；用 OpenCV 开发算法不需要；
2. IVE 的参数一般是定点化的，内部计算也是定点的；OpenCV 一般是浮点的参数和计算；故同样的功能，IVE 相比 OpenCV 有范围和精度上的限制。
3. IVE 使用物理地址，且对起始地址以及跨度有对齐要求；OpenCV 不需要物理地址，也无对齐要求；
4. IVE 有软硬件分工，有些算子硬件实现一部分，软件实现一部分，也就是会有硬件+软件多个接口实现一个 OpenCV 算子的情况。

## 6.3 ANN/SVM 查找表的建立

1. 以函数  $f(u)$  建立查找表为例，建立查找表的步骤如下(下面提到的 s32TabInLower, s32TabInUpper, u8TabInPreci, u16ElemNum, u8TabOutNorm, stTable 请参考 [IVE\\_LOOK\\_UP\\_TABLE\\_S](#) 结构体说明)；
2. 明确自变量  $u$  的范围：若  $u$  在  $[a, b]$  之间，假设  $r=b-a$ ； $a$ 、 $b$  分别对应 s32TabInLower, s32TabInUpper；（ANN 建表时  $u$  即为自变量  $u$ ；SVM 建表时  $u$  对应  $x_i^T x_j$  或者  $\|x_i - x_j\|^2$ ，具体公式可参考 [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_Predict](#) 和 [HI\\_MPI\\_IVE\\_SVM\\_Predict](#) 的【注意】）
3. 明确自变量 1 个单位的采样数  $g$ ，则  $g=1 \ll u8TabInPreci$ ；整个查找表的元素个数  $u16ElemNum=r*g \ll u8TabInPreci$ ；(ANN 和 SVM 均对查找表有最大数目限制要求，具体参考 [HI\\_MPI\\_IVE\\_ANN\\_MLP\\_Predict](#) 和 [HI\\_MPI\\_IVE\\_SVM\\_Predict](#) 的【注意】)
4. 明确值域  $f(u)$  的范围，由于一般需要将值域约束到  $[-1, 1]$ ，这时候，可以使用 u8TabOutNorm 或者  $1 \ll u8TabOutNorm$  做除数来对  $f(u)$  做归一化；(ANN 仅支持  $1 \ll u8TabOutNorm$  做除数；SVM 支持 2 种，所以 ive\_xml2bin\_ui.exe 中对 SVM 的模型转换时要求输入 divisor，且 divisor 必须与建立查找表的归一化除数一致)
5. 根据  $f(u)$  的具体公式以及  $u$  的采样值，生成对应的  $f(u)$  查找表，保存在 stTable 中。

## 6.4 Cache 内存的使用

内存开辟是否带 cache，与算法软件对这片内存的使用主体相关。由于 IVE 是直接读取 DDR 内存数据，若此时使用的内存带有 cache，必须刷 cache 来保证数据的一致性。所以若使用主体为 IVE，CPU 不使用或者仅使用一次，那么建议这片内存不带 cache；若 CPU 是使用主体，建议这片内存带 cache。