HiIVE

API Reference

Issue	**09**
Date	**2016-11-25**

Trademarks and Permissions

, HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

About This Document

Purpose

This document provides reference information including the media processing platform (MPP) programming interfaces (MPIs), header files, error codes, and proc information for the programmers that develop products or solutions using the intelligent video engine (IVE) of HiSilicon media processors.

📖 **NOTE**

- Unless otherwise specified, the contents of the Hi3516A also apply to the Hi3516D.
- Unless otherwise specified, the contents of the Hi3521A also apply to Hi3520D V300.
- Unless otherwise specified, the contents of Hi3518E V200 also apply to Hi3518E V201 and Hi3516C V200.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3516A	V100
Hi3516D	V100
Hi3536	V100
Hi3521A	V100
Hi3531A	V100
Hi3520D	V300
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3519	V100
Hi3519	V101
Hi3516C	V300
Hi3559	V100

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
⚠ DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
⚠ WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
⚠ CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
TIP	Provides a tip that may help you solve a problem or save time.
NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 09 (2016-11-25)

This issue is the ninth official release, which incorporates the following changes:

Chapter 2 API Reference

The description in the **Note** field of HI_MPI_IVE_CNN_Predict is modified.

Issue 08 (2016-08-30)

This issue is the eighth official release, which incorporates the following changes:

The contents related to Hi3559 V100 are added.

Issue 07 (2016-07-10)

This issue is the seventh official release, which incorporates the following changes:

The contents related to Hi3516C V300 are added.

Chapter 2 API Reference

The description in the **Note** field of HI_MPI_IVE_Add is modified.

Chapter 3 Data Structures

HI_MPI_IVE_Resize2 and IVE_RESIZE2_CTRL_S are added.

Issue 06 (2016-05-10)

This issue is the sixth official release, which incorporates the following changes:

The contents related to Hi3519 V101 are added.

Chapter 1 Introduction to the IVE

Section 1.2.1 is modified.

Chapter 2 API Reference

HI_MPI_IVE_CNN_GetResult is modified.

Chapter 3 Data Structures

IVE_MODULE_PARAMS_S is added.

Issue 05 (2015-12-15)

This issue is the fifth official release, which incorporates the following changes:

Chapter 3 Data Structures

The description in the **Member** field of IVE_CCBLOB_S is modified.

Issue 04 (2015-09-20)

This issue is the fourth official release, which incorporates the following changes:

Chapter 6 FAQ

This chapter is added.

The contents related to the Hi3519 V100 is added.

Issue 03 (2015-07-29)

This issue is the third official release, which incorporates the following changes:

The contents related to the Hi3521A, Hi3520D V300, Hi3531A, Hi3518E V200, Hi3518E V201, and Hi3516C V200 are added.

Issue 02 (2015-01-19)

This issue is the second official release, which incorporates the following changes:

Chapter 2 API Reference

The description in the **Example** field of HI_MPI_IVE_Query is modified.

Issue 01 (2014-12-20)

This issue is the first official release, which incorporates the following changes:

The contents related to the Hi3516D is added.

Chapter 3 Data Structures

IVE_DST_DATA_S and IVE_MEM_INFO_S are added.

Issue 00B03 (2014-09-25)

This issue is the third draft release, which incorporates the following changes:

Chapter 1 Introduction to the IVE

In section 1.2.1, the description of **bInstant** is modified.

Issue 00B02 (2014-09-14)

This issue is the second draft release, which incorporates the following changes:

Chapter 5 Proc Debugging Information

In section 5.2, debugging information and IVE parameters are modified.

Issue 00B01 (2014-08-12)

This issue is the first draft release.

Contents

1 Introduction to the IVE 1

1.1 Overview 1

1.2 Functions 1

1.2.1 Important Concepts 1

1.2.2 Instruction 11

2 API Reference 12

3 Data Structures 138

4 Error Codes 224

5 Proc Debugging Information 226

5.1 Overview 226

5.2 Proc Information 226

6 FAQs 230

6.1 Differences Between the IVE Clib on the PC and IVE SDK on the Board for Algorithm Development 230

6.2 Differences Between the IVE and OpenCV for Algorithm Development 231

6.3 Establishment of the ANN/SVM Lookup Table 231

6.4 Cache Memory 232

Figures

Figure 1-1 Stride schematic diagram 2

Figure 1-2 IVE_IMAGE_S image of the 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, or IVE_IMAGE_TYPE_U64C1 type 5

Figure 1-3 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV420SP type 5

Figure 1-4 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV422SP type 6

Figure 1-5 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV420P type 6

Figure 1-6 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV422P type 7

Figure 1-7 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_S8C2_PACKAGE type 7

Figure 1-8 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_S8C2_PLANAR type 8

Figure 1-9 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_U8C3_PACKAGE type 8

Figure 1-10 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_U8C3_PLANAR type 9

Figure 1-11 Memory for storing IVE_DATA_S data 10

Figure 1-12 Memory for storing IVE_MEM_INFO_S data 10

Figure 1-13 Integrogram combined output (IVE_IMAGE_TYPE_U64C1) 10

Figure 1-14 Histogram output format 10

Figure 2-1 Fast copy mode 16

Figure 2-2 Indirect copy mode 17

Figure 2-3 Filter formula 19

Figure 2-4 Sobel formula 25

Figure 2-5 MagAndAng formula 29

Figure 2-6 Dilate formula 32

Figure 2-7 Erode formula 35

Figure 2-8 Eight Thresh threshold modes 38

Figure 2-9 Four Thresh_S16 threshold modes 49

Figure 2-10 Two Thresh_U16 threshold modes 51

Figure 2-11 Four conversion modes from 16-bit data to 8-bit data 54

Figure 2-12 Memory allocation of GMMs for gray-scale images 71

Figure 2-13 Memory allocation of GMMs for RGB images 71

Figure 2-14 Memory allocation of GMM2s for gray-scale images 74

Figure 2-15 Memory allocation of GMM2s for RGB images 75

Figure 2-16 LBP formula 80

Figure 2-17 NormGrad formula 82

Figure 2-18 LK optical flow calculation of a 3-layer pyramid 85

Figure 2-19 LK optical flow calculation of a 3-layer pyramid 89

Figure 2-20 Bits of a pixel of the foreground status flag image 105

Figure 2-21 Input sample vector of ANN_MLP 113

Figure 2-22 Output prediction result of ANN_MLP 113

Figure 2-23 Input sample vector array of ANN_MLP 116

Figure 2-24 Output prediction result of ANN_MLP 117

Figure 2-25 Input sample vector of SVM 122

Figure 2-26 Prediction result of SVM 122

Figure 2-27 Input sample vector array of SVM 125

Figure 2-28 Prediction result of SVM 125

Figure 2-29 Output eigenvector array of the CNN 131

Figure 2-30 CNN model 132

Figure 2-31 Prediction results of CNN samples 134

1 Introduction to the IVE

1.1 Overview

The IVE is a hardware acceleration module in the intelligent analysis system of the HiSilicon media processor. Intelligent analysis solutions can be developed based on the IVE to accelerate intelligent analysis and reduce the CPU usage. Current IVE operators allow you to develop intelligent analysis solutions for video diagnosis and boundary security.

1.2 Functions

1.2.1 Important Concepts

- Handle

  When you call operators to create tasks, the system allocates a handle to each task to identify tasks.

- **bInstant** (instant returned result flag)

  bInstant is a flag indicating whether to return results in time. If you want to obtain completion information about a task, you must set **bInstant** to **HI_TRUE** when creating this task. If you do not concern about completion information, you are advised to set **bInstant** to **HI_FALSE**. This task can form a task link with subsequent tasks and these tasks can be executed together. In this way, the number of interrupts is reduced and the performance is improved.

- Query

  Based on the handle returned by the system, you can query whether the task corresponding to an operator is complete by calling HI_MPI_IVE_Query.

- Flushing cache

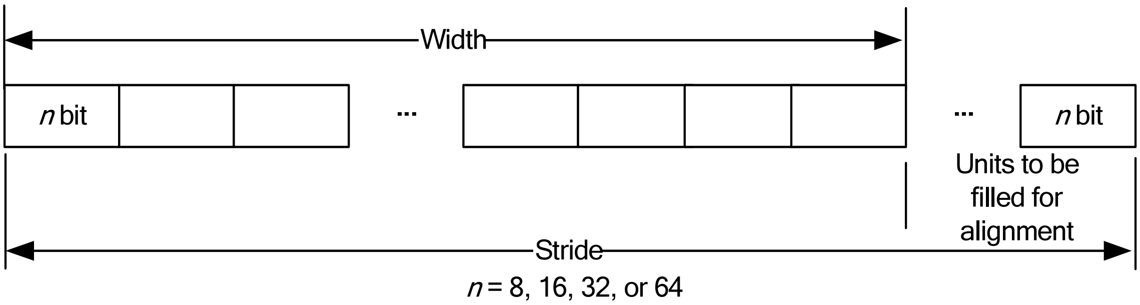
  The IVE hardware can obtain data only for the DDR. If the CPU has accessed a cacheable space when you call an IVE task, you need to flush data from the cache to the DDR by calling HI_MPI_SYS_MmzFlushCache. This ensures that the input data and output data of the IVE are not interfered by the CPU cache. For details about HI_MPI_SYS_MmzFlushCache, see the *HiMPP Vx.y Media Processing Software Development Guide*.

- Stride: a metric consistent with the width of an image or two-dimensional (2D) data, as shown in Figure 1-1.

- The stride of the IVE_IMAGE_S image data indicates the number of units for measuring an image row by pixel. The pixel bit width can be 8 bits or 16 bits.
- Stride of the IVE_DATA_S 2D data indicates the number of bytes of a two-dimensional data row, that is, the value when *n* is **8** in Figure 1-1.

IVE_DATA_S can be considered as an image of which a pixel is expressed by 8 bits. The stride can be expressed as the number of units for measuring an image or 2D data row by pixel.

Figure 1-1 Stride schematic diagram



- Alignment

  The memory address or memory stride must be a multiple of the alignment coefficient to ensure that hardware can rapidly access the memory start address or access data across rows.

  - Alignment of the start address for the data memory

    The inputs and outputs of current IVE operators must be 1-byte-aligned, 2-byte-aligned, or 16-byte-aligned. For details, see the API of each operator.

  - Stride alignment

    The stride of the generalized 2D image, 2D single-component data, or 1D array data must be 16-pixel-aligned.

CAUTION

It is recommended that the start address be 256-byte-aligned and the stride be aligned based on the odd multiple of 256 pixels to improve the memory access rate when Hi3519 V101/Hi3559 V100 uses the DDR4.

- Types of input and output data (for details, see chapter 3 "Data Structures")
  - Generalized 2D image data

    For details about the image types of IVE_IMAGE_S, IVE_SRC_IMAGE_S, and IVE_DST_IMAGE_S, see the data structure IVE_IMAGE_TYPE_E. For details about the memory allocation for IVE_IMAGE_S, IVE_SRC_IMAGE_S, and IVE_DST_IMAGE_S, see Figure 1-2 to Figure 1-10.

NOTE

The width and height of the input and output generalized 2D images for all operators must be even numbers.

- 2D single-component data

  IVE_DATA_S indicates 2D data in byte and is mainly used by direct memory access (DMA). Its memory allocation is shown in Figure 1-11. IVE_IMAGE_S can be converted into one or more IVE_DATA_S data segments.

- 1D data

  IVE_MEM_INFO_S, IVE_SRC_MEM_INFO_S, and IVE_DST_MEM_INFO_S indicate 1D data such as Hist statistics, Gaussian mixture model (GMM) data, and LKOpticalFlo corner point input. The memory allocation is shown in Figure 1-2.

- Types of generalized 2D images

Type	Description	Memory Address	Stride
IVE_IMAGE_TYPE_U8C1	8-bit unsigned single-channel image, see Figure 1-2.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_S8C1	8-bit signed single-channel image, see Figure 1-2.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_YUV420SP	YCbCr420 semi-planar image, see Figure 1-3.	Only **u32PhyAddr[0]/pu8VirAddr[0]** (luminance Y) and **u32PhyAddr[1]/pu8VirAddr[1]** (chrominance and UV are alternatively arranged) of IVE_IMAGE_S are used. Inconsecutive luminance and chrominance memories are supported but not recommended.	Only **u16Stride[0]** (luminance stride) and **u16Stride[1]** (chrominance stride) are used.
IVE_IMAGE_TYPE_YUV422SP	YCbCr422 semi-planar image, see Figure 1-4.	Only **u32PhyAddr[0]/pu8VirAddr[0]** (luminance Y) and **u32PhyAddr[1]/pu8VirAddr[1]** (chrominance and UV are alternatively stored) of IVE_IMAGE_S are used. Inconsecutive luminance and chrominance memories are supported but not recommended.	Only **u16Stride[0]** (luminance stride) and **u16Stride[1]** (chrominance stride) are used.
IVE_IMAGE_TYPE_YUV420P	YCbCr420 planar image, see Figure 1-5.	Only **u32PhyAddr[0]/pu8VirAddr[0]** (luminance Y), **u32PhyAddr[1]/pu8VirAddr[1]** (chrominance U), and **u32PhyAddr[2]/pu8VirAddr[2]** (chrominance V) of IVE_IMAGE_S are used. Inconsecutive Y, U, and V memories are supported but not recommended.	Only **u16Stride[0]** (luminance Y stride), **u16Stride[1]** (chrominance U stride), and **u16Stride[2]** (chrominance V stride) are used.

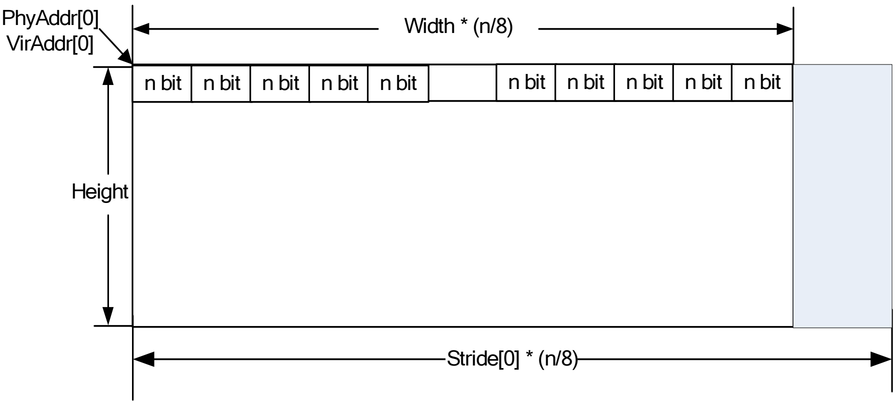
Type	Description	Memory Address	Stride
IVE_IMAGE_TYPE_YUV422P	YCbCr422 planar image, see Figure 1-6.	Only **u32PhyAddr[0]/pu8VirAddr[0]** (luminance Y), **u32PhyAddr[1]/pu8VirAddr[1]** (chrominance U), and **u32PhyAddr[2]/pu8VirAddr[2]** (chrominance V) of IVE_IMAGE_S are used.	Only **u16Stride[0]** (luminance stride), **u16Stride[1]** (chrominance U stride), and **u16Stride[2]** (chrominance V stride) are used.
IVE_IMAGE_TYPE_S8C2_PACKAGE	8-bit signed dual-channel image that is stored in package format, see Figure 1-7.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_S8C2_PLANAR	8-bit signed dual-channel image that is stored in planar format, see Figure 1-8.	Only **u32PhyAddr[0]**, **pu8VirAddr[0]**, **u32PhyAddr[1]**, and **pu8VirAddr[1]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** and **u16Stride[1]** are used.
IVE_IMAGE_TYPE_S16C1	16-bit signed single-channel image, see Figure 1-2.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_U16C1	16-bit unsigned single-channel image, see Figure 1-2.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_U8C3_PACKAGE	8-bit unsigned three-channel image that is stored in package format, see Figure 1-9.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_U8C3_PLANAR	8-bit unsigned three-channel image that is stored in planar format, see Figure 1-10.	Only **u32PhyAddr[0]**, **pu8VirAddr[0]**, **u32PhyAddr[1]**, **pu8VirAddr[1]**, **u32PhyAddr[2]**, and **pu8VirAddr[2]** of IVE_IMAGE_S are used.	Only **u16Stride[0]**, **u16Stride[1]**, and **u16Stride[2]** are used.
IVE_IMAGE_TYPE_S32C1	32-bit signed single-channel image, see Figure 1-2.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_U32C1	32-bit unsigned single-channel image, see Figure 1-2.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_S64C1	64-bit signed single-channel image, see Figure 1-2.	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of IVE_IMAGE_S are used.	Only **u16Stride[0]** is used.
IVE_IMAGE_TYPE_U64C1	64-bit unsigned single-channel image, see Figure	Only **u32PhyAddr[0]** and **pu8VirAddr[0]** of	Only **u16Stride[0]** is used.

Type	Description	Memory Address	Stride
	1-2.	IVE_IMAGE_S are used.	

- Special output data formats
  - Integrogram combined output (IVE_INTEG_OUT_CTRL_COMBINE)

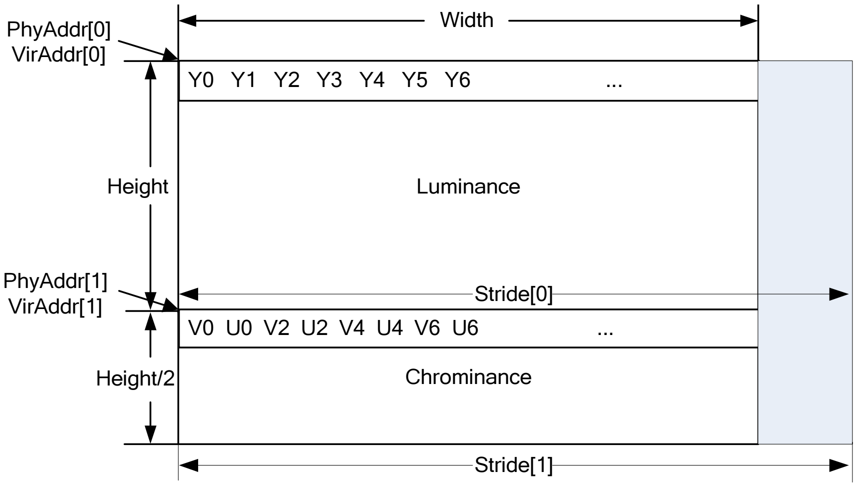
    For the IVE_IMAGE_S image of the IVE_IMAGE_TYPE_U64C1 type, S (image sum) occupies lower 28 bits, and SQ (sum of squares) occupies upper 36 bits. See Figure 1-13.
  - Histogram output format. See Figure 1-14.

Figure 1-2 IVE_IMAGE_S image of the 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, or IVE_IMAGE_TYPE_U64C1 type



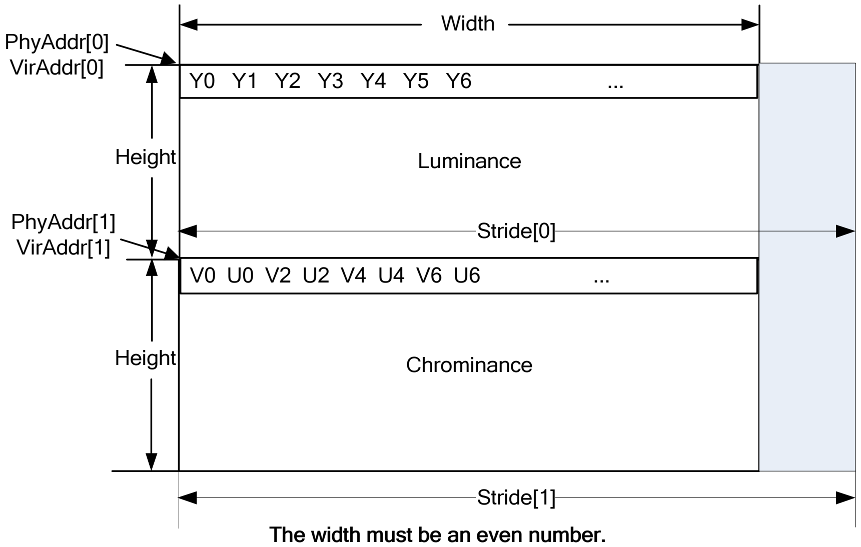
n indicates the number of bits, and the width and stride are in the unit of *n* bits.

Figure 1-3 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV420SP type



The width and height must be even numbers.

Figure 1-4 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV422SP type

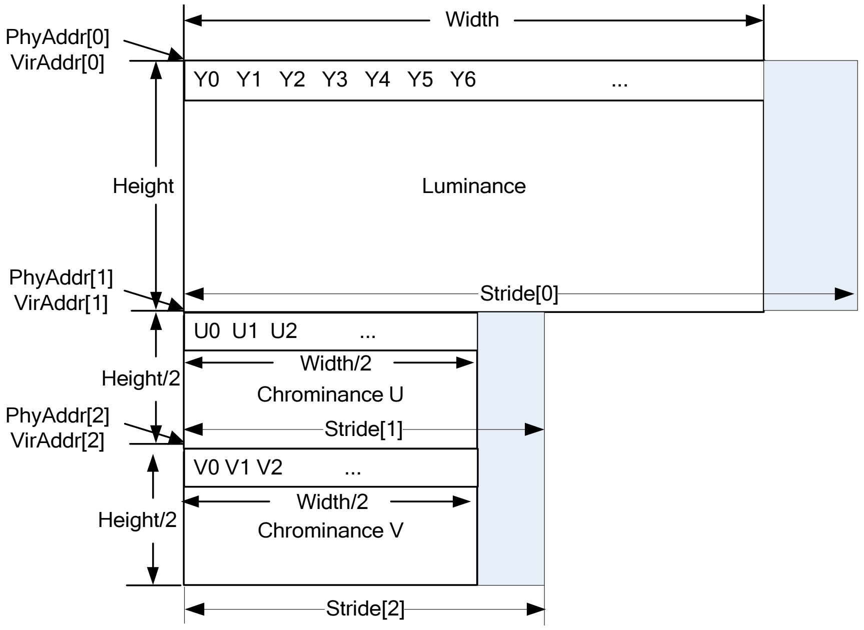


The width must be an even number.

NOTE

The chrominance V is before the chrominance U. **u32PhyAddr[2]** and **pu8VirAddr[2]** can set to the start addresses for the chrominance V, that is, **u32PhyAddr[1] + 1** and **pu8VirAddr[1] + 1**.

Figure 1-5 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV420P type



The width and height must be even numbers.

Figure 1-6 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_YUV422P type

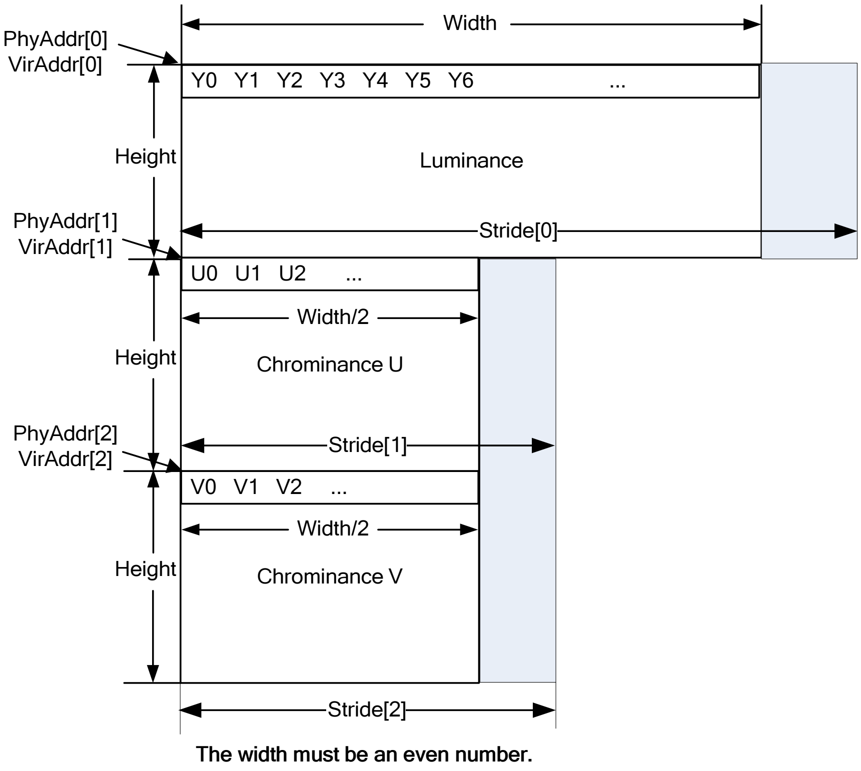


Figure 1-7 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_S8C2_PACKAGE type

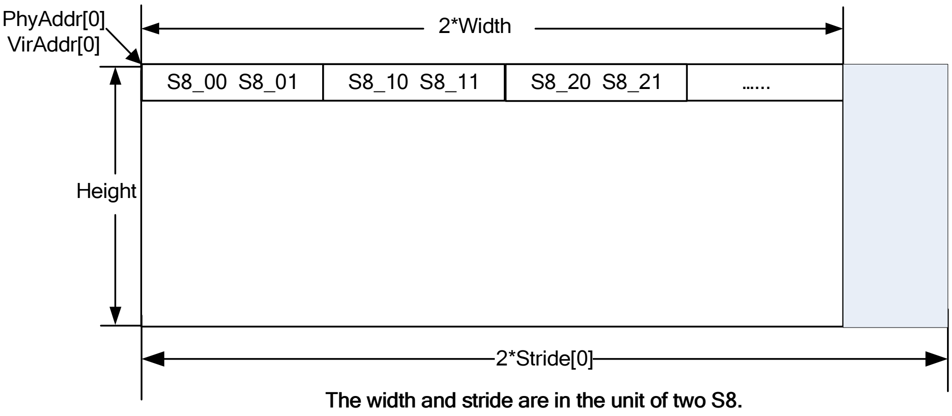


Figure 1-8 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_S8C2_PLANAR type

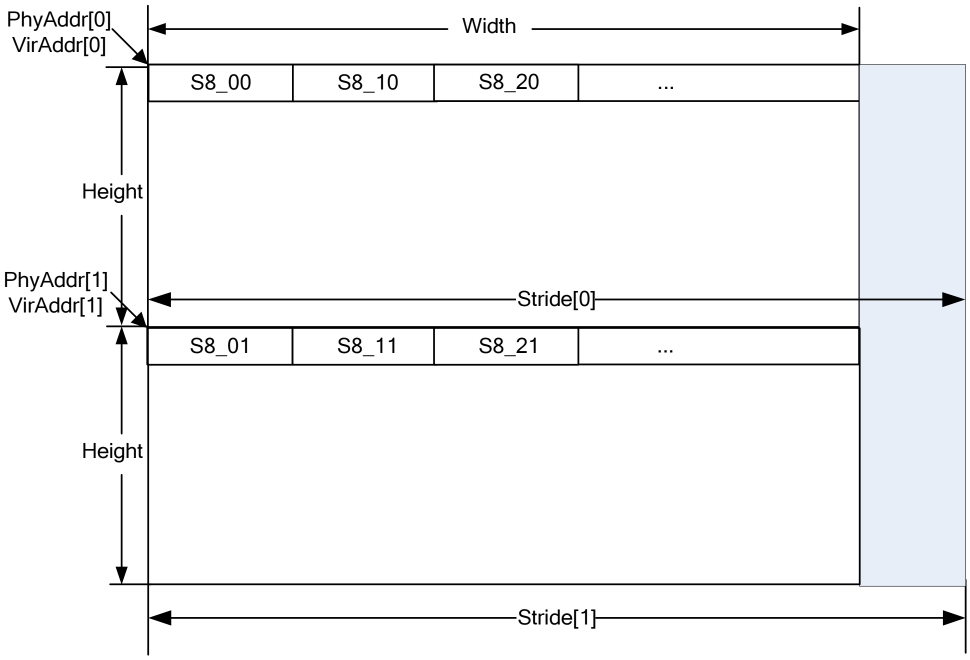
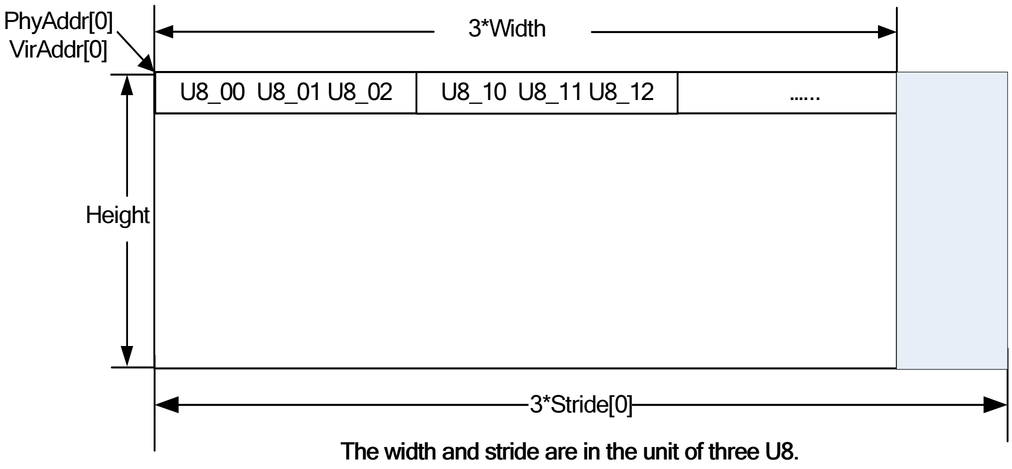


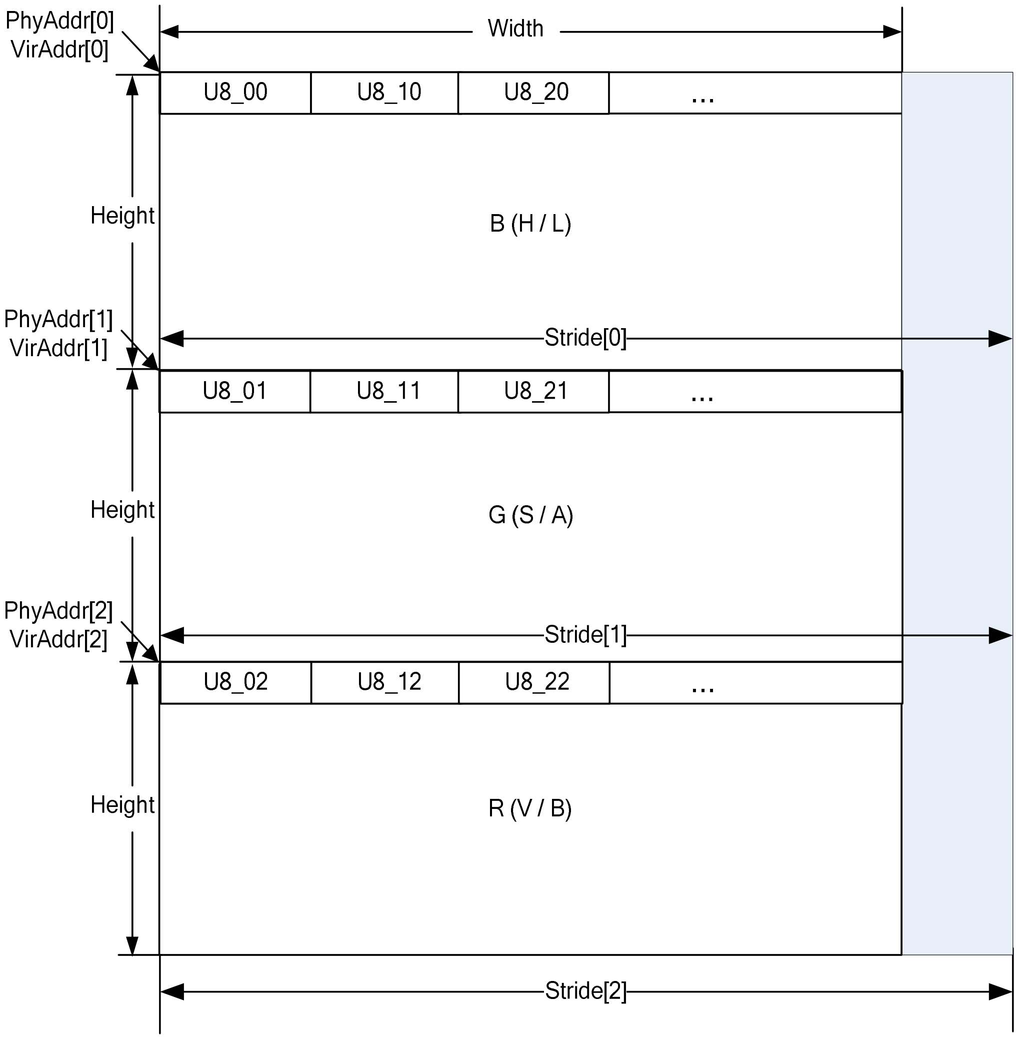
Figure 1-9 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_U8C3_PACKAGE type



NOTE

- The RGB_PACKAGE image is stored in B0G0R0B1G1R1… format, and B is in the front.
- The HSV_PACKAGE image is stored in H0S0V0H1S1V1… format, and H is in the front.
- The LAB_PACKAGE image is stored in L0A0B0L1A1B1… format, and L is in the front.

Figure 1-10 IVE_IMAGE_S image of the IVE_IMAGE_TYPE_U8C3_PLANAR type



NOTE

- For the RGB_PLANAR image, the pointer array **VirAddr[3]** stores the B, G, and R pointers in sequence, and the array **Stride[3]** indicates the strides of B, G, and R.
- For the HSV_PLANAR image, the pointer array **VirAddr[3]** stores the H, S, and V pointers in sequence, and the array **Stride[3]** indicates the strides of H, S, and V.
- For the LAB_PLANAR image, the pointer array **VirAddr[3]** stores the L, A, and B pointers in sequence, and the array **Stride[3]** indicates the strides of L, A, and B.

Figure 1-11 Memory for storing IVE_DATA_S data

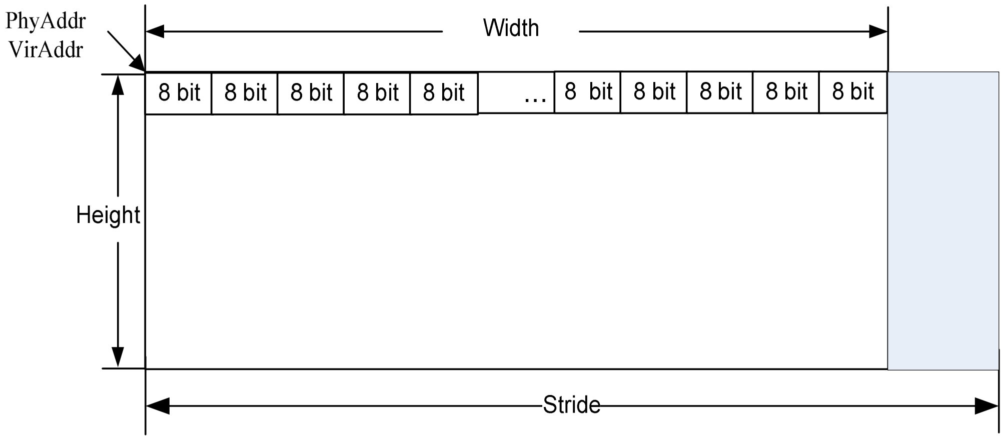


Figure 1-12 Memory for storing IVE_MEM_INFO_S data

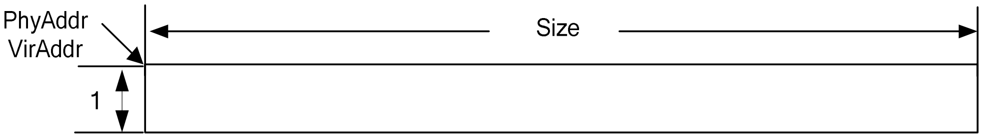


Figure 1-13 Integrogram combined output (IVE_IMAGE_TYPE_U64C1)

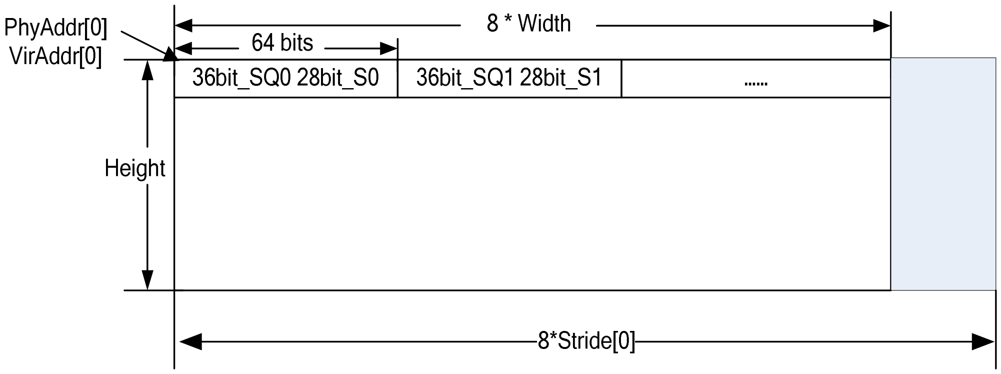
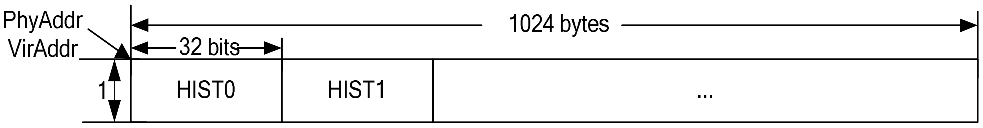


Figure 1-14 Histogram output format



1.2.2 Instruction

- Call a corresponding operator MPI to create a task, specify the value of **bInstant**, and record the returned handle ID of the task.
- Specify the block mode based on the returned handle ID to query the completion status of a task.

  For details, see the **Example** field of HI_MPI_IVE_Query.

2 API Reference

The IVE provides the following MPIs for creating and querying tasks:

- HI_MPI_IVE_DMA: Creates a DMA task.
- HI_MPI_IVE_Filter: Creates a 5x5 template filter task.
- HI_MPI_IVE_CSC: Creates a color space conversion (CSC) task.
- HI_MPI_IVE_FilterAndCSC: Creates a task combined with template filter and CSC.
- HI_MPI_IVE_Sobel: Create a 5x5 template task for calculating the Sobel-like gradient.
- HI_MPI_IVE_MagAndAng: Create a 5x5 template task for calculating the gradient magnitude and argument.
- HI_MPI_IVE_Dilate: Creates a dilate task.
- HI_MPI_IVE_Erode: Creates an erode task.
- HI_MPI_IVE_Thresh: Creates a thresh task.
- HI_MPI_IVE_And: Creates an AND task for two images.
- HI_MPI_IVE_Sub: Creates a subtraction task for two images.
- HI_MPI_IVE_Or: Creates an OR task for two images.
- HI_MPI_IVE_Integ: Creates an integrogram statistics task.
- HI_MPI_IVE_Hist: Creates a histogram statistics task.
- HI_MPI_IVE_Thresh_S16: Creates a threshold task from S16 data to 8-bit data.
- HI_MPI_IVE_Thresh_U16: Creates a threshold task from U16 data to U8 data.
- HI_MPI_IVE_16BitTo8Bit: Creates a linear conversion task from 16-bit data to 8-bit data.
- HI_MPI_IVE_OrdStatFilter: Creates a 3x3 template order statistics filter task.
- HI_MPI_IVE_Map: Creates a map (assignment for U8→U8 mapping) task.
- HI_MPI_IVE_Map: Creates a map (assignment for U8→U8/U8→U16/U8→S16 mapping) task.
- HI_MPI_IVE_EqualizeHist: Creates a histogram equalization task for gray-scale images.
- HI_MPI_IVE_Add: Creates a weighted addition task for two gray-scale images.
- HI_MPI_IVE_Xor: Creates an XOR task for two binary images.
- HI_MPI_IVE_NCC: Creates a normalized cross-correlation coefficient (NCC) coefficient calculation task for two images with the same resolution.
- HI_MPI_IVE_CCL: Creates a connected component labeling (CCL) task for binary images.

- HI_MPI_IVE_GMM: Creates a GMM background modeling task.
- HI_MPI_IVE_GMM2: Creates a GMM background modeling task for gray-scale input images and RGB_PACKAGE input images.
- HI_MPI_IVE_CannyHysEdge: Creates a Canny edge extraction task for gray-scale images (first phase of Canny edge extraction for gray-scale images).
- HI_MPI_IVE_CannyEdge: Connects edge points to form a Canny image (latter phase of Canny edge extraction for gray-scale images).
- HI_MPI_IVE_LBP: Creates a LBP calculation task.
- HI_MPI_IVE_NormGrad: Creates a normalized gradient calculation task. All gradient components are normalized to S8.
- HI_MPI_IVE_LKOpticalFlow: Creates a single-layer LK optical flow calculation task.
- HI_MPI_IVE_LKOpticalFlowPyr: Creates a multi-layer pyramid LK optical flow calculation task.
- HI_MPI_IVE_STCandiCorner: Calculates candidate corner points (first phase of Shi-Tomasi-like corner point calculation for gray-scale images).
- HI_MPI_IVE_STCorner: Selects corner points based on rules (latter phase of Shi-Tomasi-like corner point calculation for gray-scale images).
- HI_MPI_IVE_SAD: Calculates the sum of absolute difference (SAD) of two source images in 4x4, 8x8, or 16x16 blocking mode, and outputs the 16-bit/8-bit SAD images as well as SAD thresh image.
- HI_MPI_IVE_Resize: Creates a picture scaling task.
- HI_MPI_IVE_Resize2: Creates a picture scaling task. The bilinear interpolation scaling is supported. Multiple U8C1 pictures can be scaled at the same time.
- HI_MPI_IVE_GradFg: Calculates the gradient foreground image based on the gradient of the background image and current frame.
- HI_MPI_IVE_MatchBgModel: Matches the background model based on the codebook evolution.
- HI_MPI_IVE_UpdateBgModel: Updates the background model based on the codebook evolution.
- HI_MPI_IVE_ANN_MLP_LoadModel: Reads the artificial neural network (ANN) multi-layer perceptron (MLP) model file and initializes model data.
- HI_MPI_IVE_ANN_MLP_UnloadModel: Deinitializes ANN model data.
- HI_MPI_IVE_ANN_MLP_Predict: Creates an ANN_MLP prediction calculation task for a single sample.
- HI_MPI_IVE_ANN_MLP_Predict: Creates ANN_MLP prediction tasks for multiple samples of the same model.
- HI_MPI_IVE_ANN_MLP_LoadModel: Reads the artificial neural network (ANN) multi-layer perceptron (MLP) model file and initializes model data.
- HI_MPI_IVE_ANN_MLP_UnloadModel: Deinitializes ANN model data.
- HI_MPI_IVE_SVM_LoadModel: Reads the SVM model file and initializes model data.
- HI_MPI_IVE_SVM_UnloadModel: Deinitializes SVM model data.
- HI_MPI_IVE_SVM_Predict: Creates an SVM prediction task for a single sample.
- HI_MPI_IVE_CNN_LoadModel: Reads the convolutional neural network (CNN) model file and initializes the CNN model data.
- HI_MPI_IVE_CNN_UnloadModel: Deinitializes the CNN model data.

- HI_MPI_IVE_CNN_Predict: Creates one or multiple sample prediction tasks of a CNN model and outputs the eigenvector.
- HI_MPI_IVE_CNN_GetResult: Receives the CNN prediction result, executes the Softmax operation to predict the type of each sample picture, and output the classification (Rank-1) with the highest confidence as well as the corresponding confidence.
- HI_MPI_IVE_Query: Queries the completion status of an existing task.

HI_MPI_IVE_DMA

[Description]

Creates a DMA task. Fast copying, indirect copying, and memory filling are supported. Data can be rapidly copied from a memory to another one or data can be regularly copied from a memory to another one. In addition, a memory can be filled.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to source data It cannot be null.	Input (output in set mode)
pstDst	Pointer to output data It cannot be empty in copy mode.	Output
pstDmaCtrl	Pointer to DMA control parameters It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

NOTE

- The copy mode indicates IVE_DMA_MODE_DIRECT_COPY or IVE_DMA_MODE_INTERVAL_COPY.
- The set mode indicates IVE_DMA_MODE_SET_3BYTE or IVE_DMA_MODE_SET_8BYTE.

Parameter	Supported Type	Address Alignment Mode	Resolution
pstSrc	IVE_DATA_S	1 byte	32 x 1 to 1920 x 1080

Parameter	Supported Type	Address Alignment Mode	Resolution
pstDst	IVE_DST_DATA_S	1 byte	• 32 x 1 to 1920 x 1080 in direct copy mode • Less than 32 x 1 to 1920 x 1080 in indirect copy mode

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

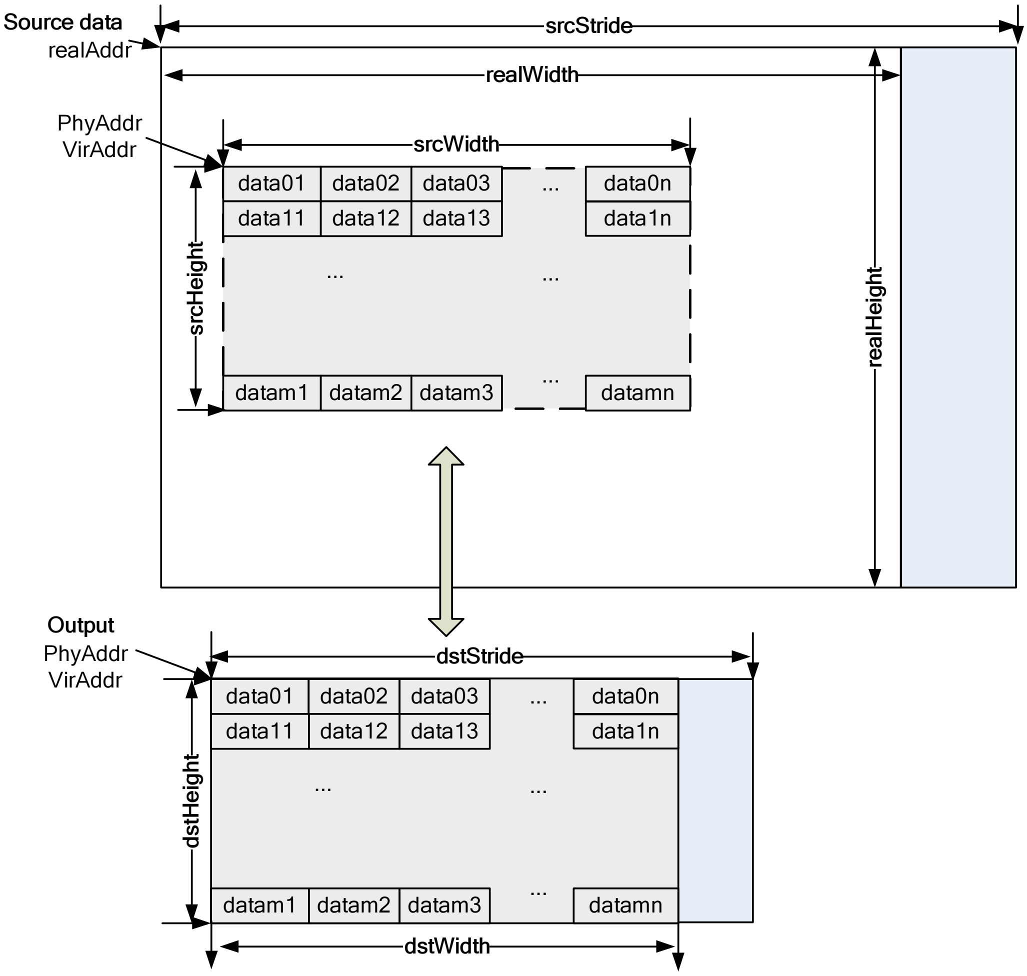
- IVE_DMA_MODE_DIRECT_COPY: fast copy mode

  In this mode, a small memory can be obtained from a large memory, as shown in Figure 2-1. The following is the calculation formula:

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

  $I(x,y)$ corresponds to **pstSrc**, and $I_{out}(x,y)$ corresponds to **pstDst**.

Figure 2-1 Fast copy mode

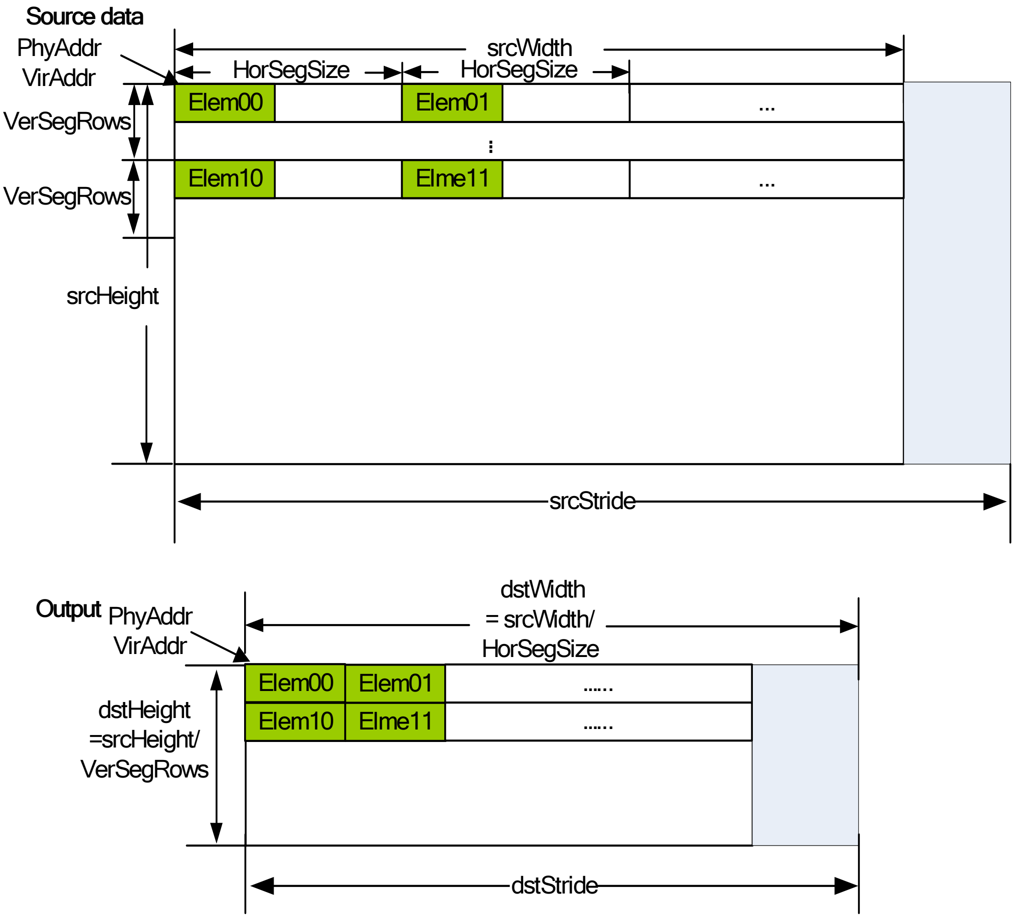


- IVE_DMA_MODE_INTERVAL_COPY: indirect copy mode
  - The source data width must be a multiple of **u8HorSegSize**.
  - In indirect copy mode, data in the first row of every **u8VerSegRows** rows is split into data segments with the size of **u8HorSegSize** each, and the first bytes with the size of **u8ElemSize** in each segment are copied. See Figure 2-2.
- IVE_DMA_MODE_SET_3BYTE: 3-byte filling mode

  Only **pstSrc** is used. Source data is filled with the lower three bytes of **u64Val**. If the number of last bytes in a row is less than 3 bytes, data is filled with the lower bytes of **u64Val**.
- IVE_DMA_MODE_SET_8BYTE: 8-byte filling mode

  Only **pstSrc** is used. Source data is filled with the bytes of **u64Val**. If the number of last bytes in a row is less than 8 bytes, data is filled with the lower bytes of **u64Val**.

Figure 2-2 Indirect copy mode



[Example]

None

[See Also]

None

HI_MPI_IVE_Filter

[Description]

Creates a 5x5 template filter task. You can set template coefficients to implement various filter effects.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstFltCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1, YUV420SP, and YUV422SP	16 bytes	64 x 64 to 1920 x 1024
pstDst	U8C1, YUV420SP, and YUV422SP	16 bytes	64 x 64 to 1920 x 1024

NOTE

U8C1, YUV420SP, and YUV422SP are the members **IVE_IMAGE_TYPE_U8C1**, **IVE_IMAGE_TYPE_YUV420SP**, and **IVE_IMAGE_TYPE_YUV422SP** of IVE_IMAGE_TYPE_E for short respectively. This rule applies to the other members.

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

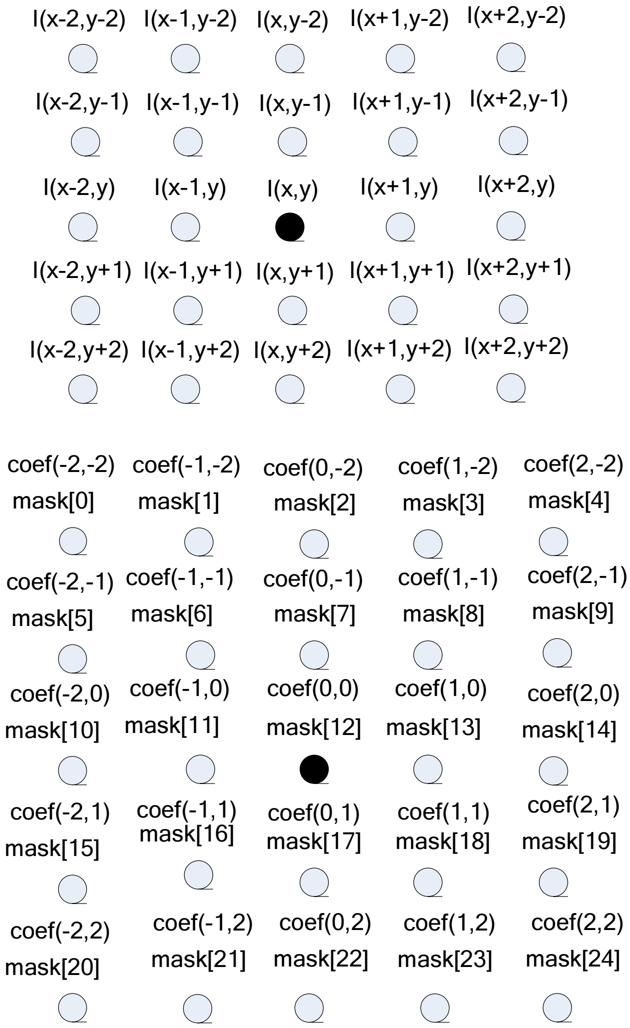
- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- If the source data format is YUV420SP or YUV422SP, output data strides must be the same.

- Figure 2-3 shows the filter formula.

Figure 2-3 Filter formula


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

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

$I(x,y)$ corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, $coef$ (mask) corresponds to **as8Mask[25]** in **pstFltCtrl**, and **norm** corresponds to **u8Norm** in **pstFltCtrl**.

- Classic Gaussian template

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

$$u8Norm = 4 \qquad u8Norm = 8 \qquad u8Norm = 8$$

[Example]

None

[See Also]

- HI_MPI_IVE_FilterAndCSC
- HI_MPI_IVE_OrdStatFilter

HI_MPI_IVE_CSC

[Description]

Creates a CSC task for implementing YUV2RGB/YUV2HSV/YUV2LAB/RGB2YUV conversion.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstCscCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	YUV420SP, YUV422SP, U8C3_PLANAR, and U8C3_PACKAGE	16 bytes	64 x 64 to 1920 x 1080
pstDst	U8C3_PLANAR, U8C3_PACKAGE, YUV420SP, and YUV422SP	16 bytes	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- If the output data format is U8C3_PLANAR, YUV420SP, or YUV422SP, output data strides must be the same.
- A total of 12 working modes are supported. The value range varies according to the working mode. For details, see IVE_CSC_MODE_E.
- For details about YUV2HSV and YUV2LAB, see the implementation in OpenCV.

NOTE

OpenCV in this document indicates OpenCV 2.4.8.

[Example]

None

[See Also]

HI_MPI_IVE_FilterAndCSC

HI_MPI_IVE_FilterAndCSC

[Description]

Creates a task combined with 5x5 template filter and YUV2RGB CSC. This MPI enables two functions to be implemented at a time.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstFltCscCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	YUV420SP and YUV422SP	16 bytes	64 x 64 to 1920 x 1024
pstDst	U8C3_PLANAR and U8C3_PACKAGE	16 bytes	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- If the output data format is U8C3_PLANAR, output data strides must be the same.

- Only the four YUV2RGB working modes are supported. For details, see IVE_CSC_MODE_E.

[Example]

None

[See Also]

- HI_MPI_IVE_Filter
- HI_MPI_IVE_FilterAndCSC

HI_MPI_IVE_Sobel

[Description]

Create a 5x5 template task for calculating the Sobel-like gradient.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDstH	Pointer to the gradient component image H that is obtained after filtering based on the template. It cannot be null if the output is required based on **pstSobelCtrl→enOutCtrl**. The height and width are the same as those of **pstSrc**.	Output
pstDstV	Pointer to the gradient component image V that is obtained after filtering based on the transposed template. It cannot be null if the output is required based on **pstSobelCtrl→enOutCtrl**. The height and width are the same as those of **pstSrc**.	Output
pstSobelCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstDstH	S16C1	16 bytes	64 x 64 to 1920 x 1024
pstDstV	S16C1	16 bytes	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

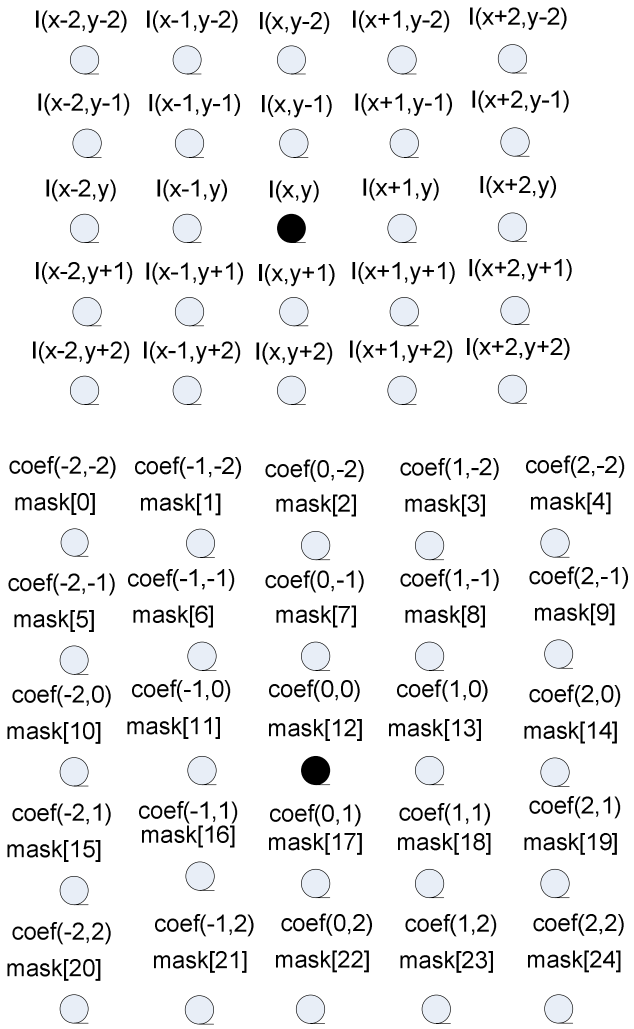
[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Three output modes are supported. For details, see IVE_SOBEL_OUT_CTRL_E.
- If the output mode is **IVE_SOBEL_OUT_CTRL_BOTH**, the strides of **pstDstH** and **pstDstV** must be the same.
- Figure 2-4 shows the Sobel formula.

Figure 2-4 Sobel formula



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

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

$I(x,y)$ corresponds to **pstSrc**, $Hout(x,y)$ corresponds to **pstDstH**, $Vout(x,y)$ corresponds to **pstDstV**, and $coef$ (mask) corresponds to **as8Mask[25]** in **pstSobelCtrl**.

- The following shows the Sobel template:

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

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

- The following shows the Scharr template:

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

- The following shows the Laplace template:

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

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

[Example]

None

[See Also]

- HI_MPI_IVE_MagAndAng

- HI_MPI_IVE_NormGrad

HI_MPI_IVE_MagAndAng

[Description]

Create a 5x5 template task for calculating the gradient magnitude and argument.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDstMag	Pointer to the output magnitude image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstDstAng	Pointer to the output argument image It cannot be null if the output is required based on **pstMagAndAngCtrl→enOutCtrl**. The height and width are the same as those of **pstSrc**.	Output
pstMagAndAngCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstDstMag	U16C1	16 bytes	64 x 64 to 1920 x 1024
pstDstAng	U8C1	16 bytes	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

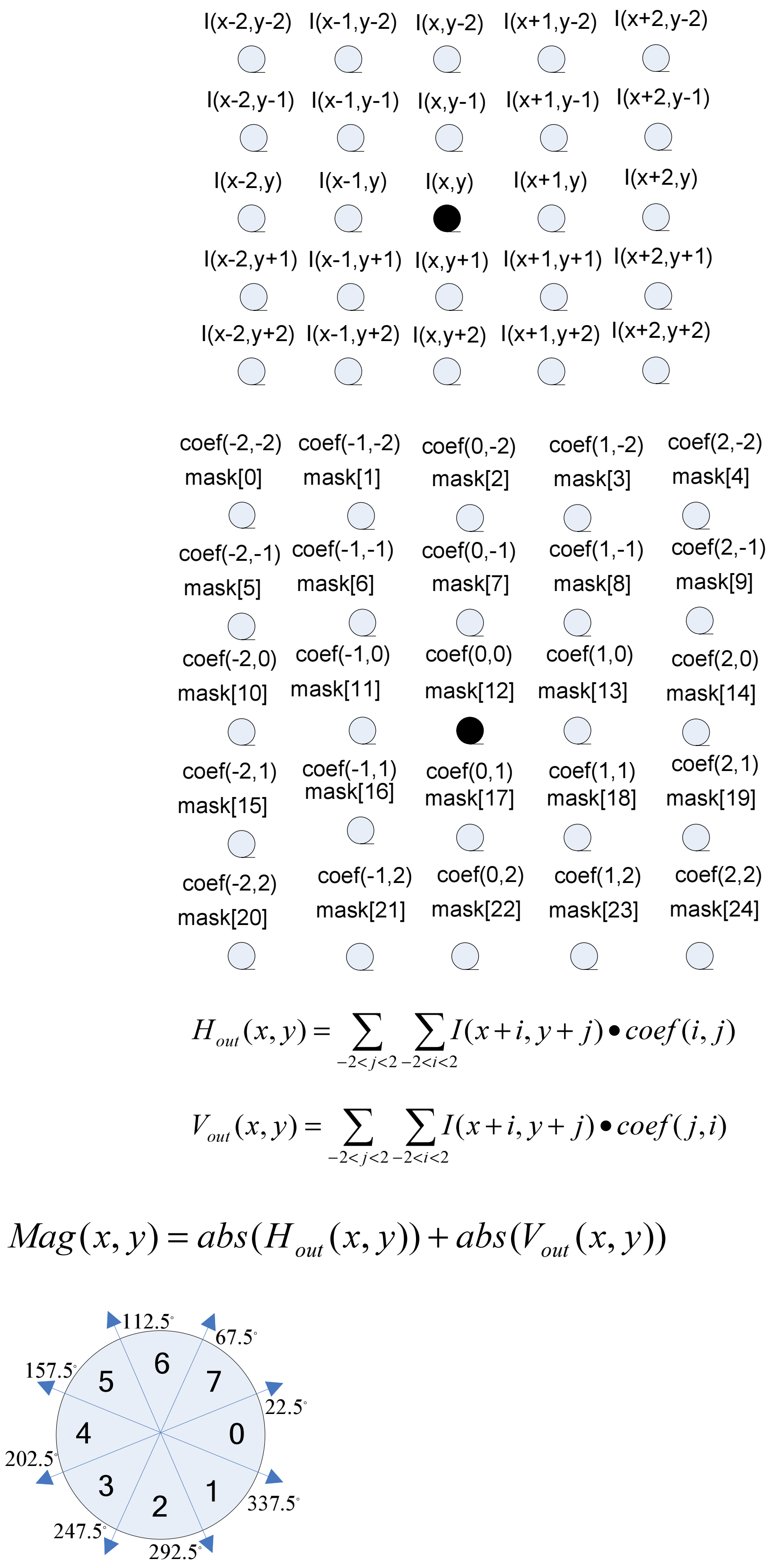
[Note]

- Two output formats are supported. For details, see IVE_MAG_AND_ANG_OUT_CTRL_E.
- If the output mode is **IVE_MAG_AND_ANG_OUT_CTRL_MAG_AND_ANG**, the strides of **pstDstMag** and **pstDstAng** must be the same.
- You can perform a thresh operation on a magnitude image by using **pstMagAndAngCtrl→u16Thr** to implement edge orientation histogram (EOH). The formula is as follows:

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

$Mag(x,y)$ corresponds to **pstDstMag**.

Figure 2-5 MagAndAng formula



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

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

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

The value of $\theta(x, y)$ is selected from the preceding directions 0–7 based on $H_{out}(x, y)$, $V_{out}(x, y)$, and $\arctan(\frac{V_{out}}{H_{out}})$. $I(x, y)$ corresponds to **pstSrc**, $Mag(x, y)$ corresponds to **pstDstMag**, $\theta(x, y)$ corresponds to **pstDstAng**, and $coef$ (mask) corresponds to **as8Mask[25]** in **pstMagAndAngCtrl**.

[Example]

None

[See Also]

- HI_MPI_IVE_CannyHysEdge
- HI_MPI_IVE_CannyEdge
- HI_MPI_IVE_Sobel

HI_MPI_IVE_Dilate

[Description]

Creates a 5x5 template dilate task for binary images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstDilateCtrl	Pointer to control information	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1 binary image	16 bytes	64 x 64 to 1920 x 1024

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstDst	U8C1 binary image	16 bytes	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

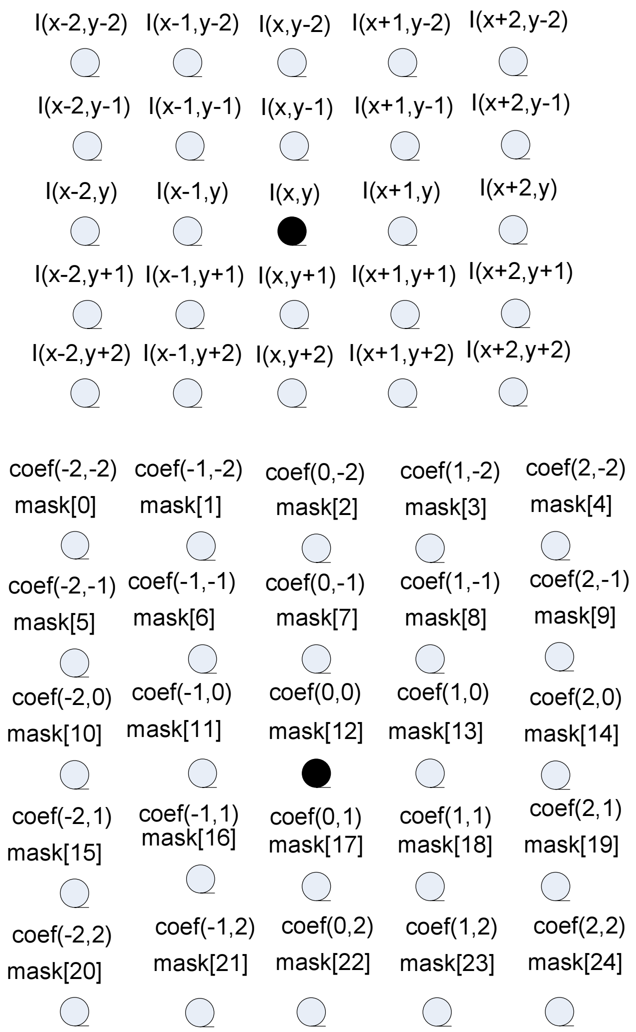
- The template coefficient must be 0 or 255.
- The following are template samples:

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

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

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

Figure 2-6 Dilate formula



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

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

| indicates bitwise OR operation, & indicates bitwise AND operation, and % indicates REM operation. $I(x,y)$ corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, and $coef$ (mask) corresponds to **au8Mask[25]** in **pstDilateCtrl**.

[Example]

None

[See Also]

- HI_MPI_IVE_Erode
- HI_MPI_IVE_OrdStatFilter

HI_MPI_IVE_Erode

[Description]

Creates a 5x5 template erode task for binary images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstErodeCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1 binary image	16 byte	64 x 64 to 1920 x 1024
pstDst	U8C1 binary image	16 byte	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

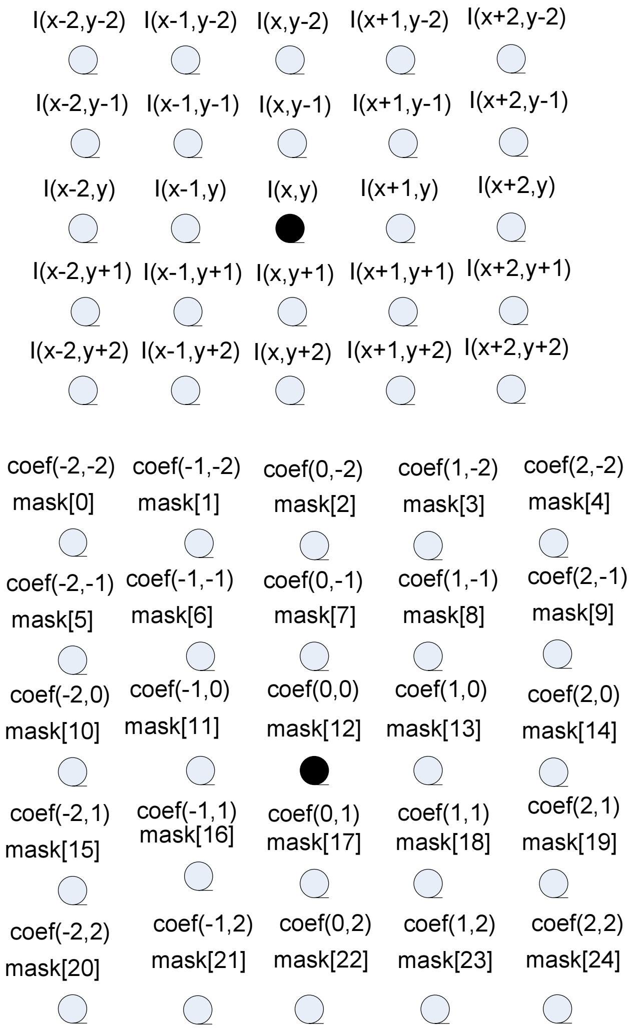
- The template coefficient must be 0 or 255.
- The following are template samples:

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

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

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

Figure 2-7 Erode formula



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

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

| indicates bitwise OR operation, & indicates bitwise AND operation, and % indicates REM operation. $I(x,y)$ corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, and $coef$ (mask) corresponds to **au8Mask[25]** in **pstErodeCtrl**.

[Example]

None

[See Also]

- HI_MPI_IVE_Dilate
- HI_MPI_IVE_OrdStatFilter

HI_MPI_IVE_Thresh

[Description]

Creates a thresh task for gray-scale images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstThrCtrl	Pointer to control information	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	1 byte	64 x 64 to 1920 x 1080
pstDst	U8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Eight operation modes are supported. For details, see IVE_THRESH_MODE_E.
- The following are related formulas:
  - IVE_THRESH_MODE_BINARY

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

    No value needs to be assigned to $midVal$ and $highThr$.
  - IVE_THRESH_MODE_TRUNC

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

    No value needs to be assigned to $minVal$, $midVal$, and $highThr$.
  - IVE_THRESH_MODE_TO_MINVAL

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

    No value needs to be assigned to $midVal$, $maxVal$, and $highThr$.
  - IVE_THRESH_MODE_MIN_MID_MAX

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

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

    No value needs to be assigned to $minVal$.
  - IVE_THRESH_MODE_MIN_MID_ORI

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

    No value needs to be assigned to $maxVal$.
  - IVE_THRESH_MODE_MIN_ORI_MAX

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

    No value needs to be assigned to $midVal$.
  - IVE_THRESH_MODE_ORI_MID_ORI

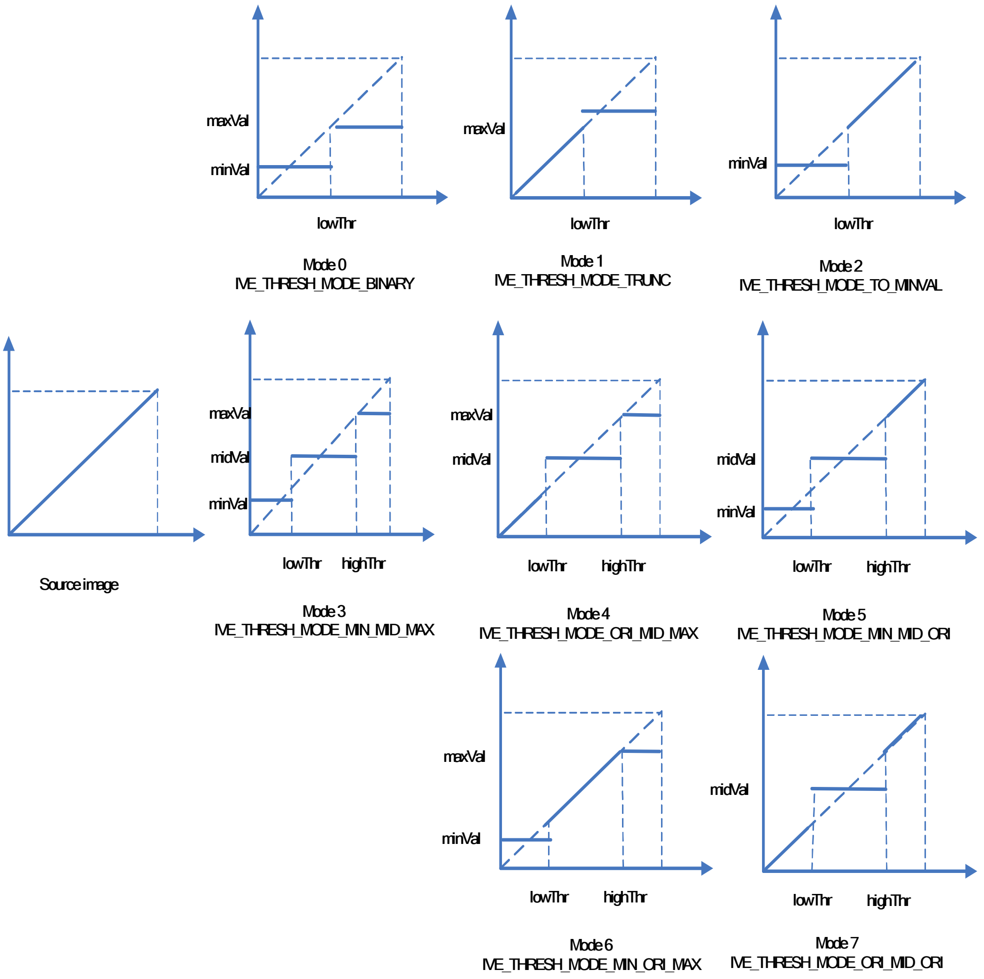
$$I_{out}(x,y)=\begin{cases} I(x,y) & (I(x,y)\le lowThr) \\ midVal & (lowThr < I(x,y)\le highThr) \\ I(x,y) & (I(x,y) > highThr) \end{cases}$$

No value needs to be assigned to $minVal$ and $maxVal$.

$I(x,y)$ corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, and **mode**, **lowThr**, **highThr**, **minVal**, **midVal**, and **maxVal** correspond to **enMode**, **u8LowThr**, **u8HighThr**, **u8MinVal**, **u8MidVal**, and **u8MaxVal** in **pstThrCtrl** respectively. For details, see Figure 2-8.

- It is not required that values of **u8MinVal**, **u8MidVal**, and **u8MaxVal** in **pstThrCtrl** meet the requirements in the variable name conventions.

Figure 2-8 Eight Thresh threshold modes



Mode 0
IVE_THRESH_MODE_BINARY

Mode 1
IVE_THRESH_MODE_TRUNC

Mode 2
IVE_THRESH_MODE_TO_MINVAL

Source image

Mode 3
IVE_THRESH_MODE_MIN_MID_MAX

Mode 4
IVE_THRESH_MODE_ORI_MID_MAX

Mode 5
IVE_THRESH_MODE_MIN_MID_ORI

Mode 6
IVE_THRESH_MODE_MIN_ORI_MAX

Mode 7
IVE_THRESH_MODE_ORI_MID_ORI

[Example]

None

[See Also]

- HI_MPI_IVE_Thresh_S16
- HI_MPI_IVE_Thresh_U16

HI_MPI_IVE_And

[Description]

Creates an AND task for two images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc1	Pointer to source image 1 It cannot be null.	Input
pstSrc2	Pointer to source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc1**.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc1	U8C1 binary image	1 byte	64 x 64 to 1920 x 1080
pstSrc2	U8C1 binary image	1 byte	64 x 64 to 1920 x 1080
pstDst	U8C1 binary image	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The following is the formula:

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

$I_{src1}(x,y)$ corresponds to **pstSrc1**, $I_{src2}(x,y)$ corresponds to **pstSrc2**, and $I_{out}(x,y)$ corresponds to **pstDst**.

[Example]

None

[See Also]

- HI_MPI_IVE_Or
- HI_MPI_IVE_Xor

HI_MPI_IVE_Sub

[Description]

Creates a subtraction task for two gray-scale images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle	Output
pstSrc1	Pointer to source image 1 It cannot be null.	Input

Parameter	Description	Input/Output
pstSrc2	Pointer to source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc1**.	Output
pstSubCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc1	U8C1	1 byte	64 x 64 to 1920 x 1080
pstSrc2	U8C1	1 byte	64 x 64 to 1920 x 1080
pstDst	U8C1 and S8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Two output formats are supported. For details, see IVE_SUB_MODE_E.
- IVE_SUB_MODE_ABS
  - Formula: $I_{out}(x,y) = abs(\mathrm{I}_{src1}(x,y) - I_{src2}(x,y))$
  - Output format: U8C1
- IVE_SUB_MODE_SHIFT

- Formula: $I_{out}(x,y) = (\mathrm{I}_{src1}(x,y) - I_{src2}(x,y)) >> 1$
- Output format: S8C1

$I_{src1}(x,y)$ corresponds to **pstSrc1**, $I_{src2}(x,y)$ corresponds to **pstSrc2**, and $I_{out}(x,y)$ corresponds to **pstDst**.

[Example]

None

[See Also]

HI_MPI_IVE_Add

HI_MPI_IVE_Or

[Description]

Creates an OR task for two binary images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc1	Pointer to source image 1 It cannot be null.	Input
pstSrc2	Pointer to source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc1**.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc1	U8C1	1 byte	64 x 64 to 1920 x 1080

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc2	U8C1	1 byte	64 x 64 to 1920 x 1080
pstDst	U8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The following is the formula:

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

$I_{src1}(x,y)$ corresponds to **pstSrc1**, $I_{src2}(x,y)$ corresponds to **pstSrc2**, and $I_{out}(x,y)$ corresponds to **pstDst**.

[Example]

None

[See Also]

- HI_MPI_IVE_And
- HI_MPI_IVE_Xor

HI_MPI_IVE_Integ

[Description]

Creates an integrogram statistics task for gray-scale images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstIntegCtrl	Pointer to control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	32 x 16 to 1920 x 1080
pstDst	U32C1 and U64C1	16 bytes	32 x 16 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- **IVE_INTEG_OUT_CTRL_COMBINE** indicates the combined output mode. In this mode, the output image type must be **IVE_IMAGE_TYPE_U64C1**, as shown in Figure 1-13. The following is the formula:

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

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

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

- **IVE_INTEG_OUT_CTRL_SUM** indicates the sum integrogram output mode. In this mode, the output image type must be **IVE_IMAGE_TYPE_U32C1**. The following is the formula:

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

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

- **IVE_INTEG_OUT_CTRL_SQSUM** indicates the square sum integrogram output mode. In this mode, the output image type must be **IVE_IMAGE_TYPE_U64C1**. The following is the formula:

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

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

  $I(x,y)$ corresponds to **pstSrc**, and $I_{out}(x,y)$ corresponds to **pstDst**.

[Example]

None

[See Also]

None

HI_MPI_IVE_Hist

[Description]

Creates a histogram statistics task for gray-scale images.

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input

Parameter	Description	Input/Output
pstDst	Pointer to output data It cannot be null. The minimum memory size is 1024 bytes. See Figure 1-14.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1080
pstDst	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The following is the formula:

$$I_{out}(x) = \sum_{i}\sum_{j}((I(i,j) == x) ?1:0) \qquad x = 0\ldots255$$

$I(i,j)$ corresponds to **pstSrc**, and $I_{out}(x)$ corresponds to **pstDst**.

[Example]

None

[See Also]

None

HI_MPI_IVE_Thresh_S16

[Description]

Creates a threshold task from S16 data to 8-bit data.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstThrS16Ctrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	S16C1	2 bytes	64 x 64 to 1920 x 1080
pstDst	U8C1 and S8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Four operation modes are supported. For details, see IVE_THRESH_S16_MODE_E.
- The following are related formulas:
  - IVE_THRESH_S16_MODE_S16_TO_S8_MIN_MID_MAX

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

    Requirements:

    $-32768 \le lowThr \le highThr \le 32767$

    $-128 \le minVal, midVal, maxVal \le 127$
  - IVE_THRESH_S16_MODE_S16_TO_S8_MIN_ORI_MAX

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

    Requirements:

    $-129 \le lowThr \le highThr \le 127$

    $-128 \le minVal, maxVal \le 127$
  - IVE_THRESH_S16_MODE_S16_TO_U8_MIN_MID_MAX

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

    Requirements:

    $-32768 \le lowThr \le highThr \le 32767$

    $0 \le minVal, midVal, maxVal \le 255$
  - IVE_THRESH_S16_MODE_S16_TO_U8_MIN_ORI_MAX

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

    Requirements:

    $-1 \le lowThr \le highThr \le 255$

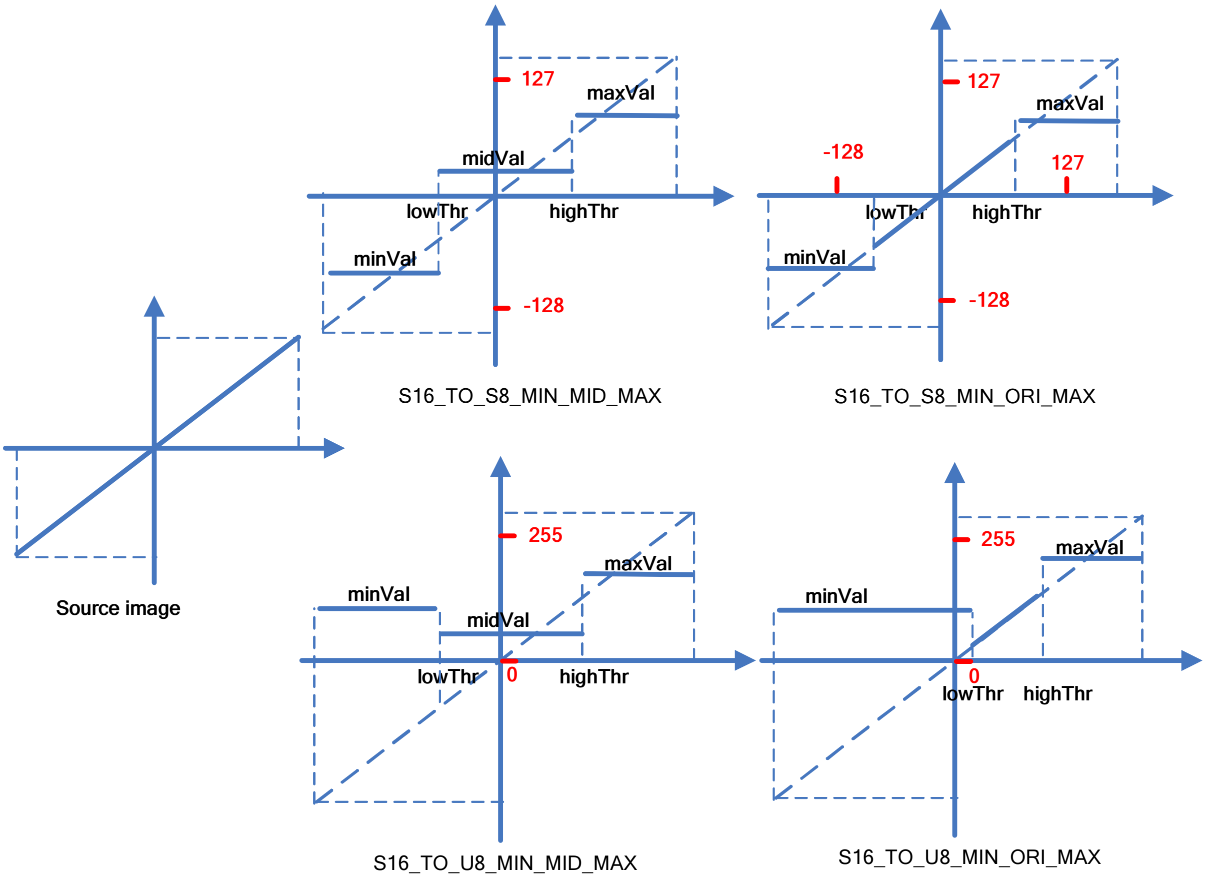
    $0 \le minVal, maxVal \le 255$

$I(x,y)$ corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, and **mode**, **lowThr**, **highThr**, **minVal**, **midVal**, and **maxVal** correspond to **enMode**, **s16LowThr**,

s16HighThr, **un8MinVal**, **un8MidVal**, and **un8MaxVal** in **pstThrS16Ctrl** respectively. For details, see Figure 2-9.

- It is not required that values of **un8MinVal**, **un8MidVal**, and **un8MaxVal** in **pstThrS16Ctrl** meet the requirements in the variable name conventions.

Figure 2-9 Four Thresh_S16 threshold modes



[Example]

None

[See Also]

- HI_MPI_IVE_Thresh_U16
- HI_MPI_IVE_16BitTo8Bit

HI_MPI_IVE_Thresh_U16

[Description]

Creates a threshold task from U16 data to U8 data.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstThrU16Ctrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U16C1	2 byte	64 x 64 to 1920 x 1080
pstDst	U8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Two operation modes are supported. For details, see IVE_THRESH_U16_MODE_E.
- The following are related formulas:

- IVE_THRESH_U16_MODE_U16_TO_U8_MIN_MID_MAX

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

Requirement: $0 \le$ $lowThr$ $\le$ $highThr$ ≤ 65535

- IVE_THRESH_U16_MODE_U16_TO_U8_MIN_ORI_MAX

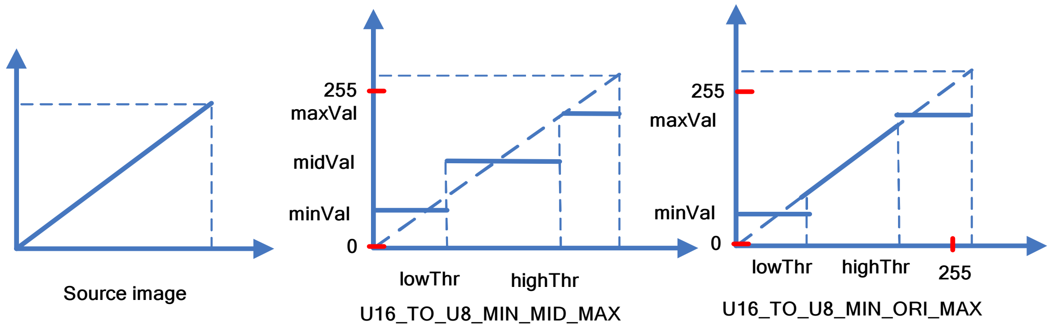
$$I_{out}(x,y)=\begin{cases} minVal & (I(x,y)\le lowThr) \\ I(x,y) & (lowThr < I(x,y) \le highThr) \\ maxVal & (I(x,y) > highThr) \end{cases}$$

Requirement: $0 \le$ $lowThr$ $\le$ $highThr$ ≤ 255

$I(x,y)$ corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, and **mode**, **lowThr**, **highThr**, **minVal**, **midVal**, and **maxVal** correspond to **enMode**, **u16LowThr**, **u16HighThr**, **u8MinVal**, **u8MidVal**, and **u8MaxVal** in **pstThrU16Ctrl** respectively. For details, see Figure 2-10.

- It is not required that values of **u8MinVal**, **u8MidVal**, and **u8MaxVal** in **pstThrU16Ctrl** meet the requirements in the variable name conventions.

Figure 2-10 Two Thresh_U16 threshold modes



[Example]

None

[See Also]

- HI_MPI_IVE_Thresh_S16
- HI_MPI_IVE_16BitTo8Bit

HI_MPI_IVE_16BitTo8Bit

[Description]

Creates a linear conversion task from 16-bit data to 8-bit data.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pst16BitTo8BitCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U16C1 and S16C1	2 bytes	64 x 64 to 1920 x 1080
pstDst	U8C1 and S8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Four modes are supported. For details, see IVE_16BIT_TO_8BIT_MODE_E.
- The following are related formulas:
  - IVE_16BIT_TO_8BIT_MODE_S16_TO_S8

$$I_{out}(x,y)=\begin{cases}-128 & (\frac{a}{b}I(x,y)<-128)\\ \frac{a}{b}I(x,y) & (-128\le\frac{a}{b}I(x,y)\le127)\\ 127 & (\frac{a}{b}I(x,y)>127)\end{cases}$$

  - IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_ABS

$$I_{out}(x,y)=\begin{cases}\left|\frac{a}{b}I(x,y)\right| & (\left|\frac{a}{b}I(x,y)\right|\le255)\\ 255 & (\left|\frac{a}{b}I(x,y)\right|>255)\end{cases}$$

  - IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_BIAS

$$I_{out}(x,y)=\begin{cases}0 & (\frac{a}{b}I(x,y)+bais<0)\\ \frac{a}{b}I(x,y)+bias & (0\le\frac{a}{b}I(x,y)+bias\le255)\\ 255 & (\frac{a}{b}I(x,y)+bais>255)\end{cases}$$

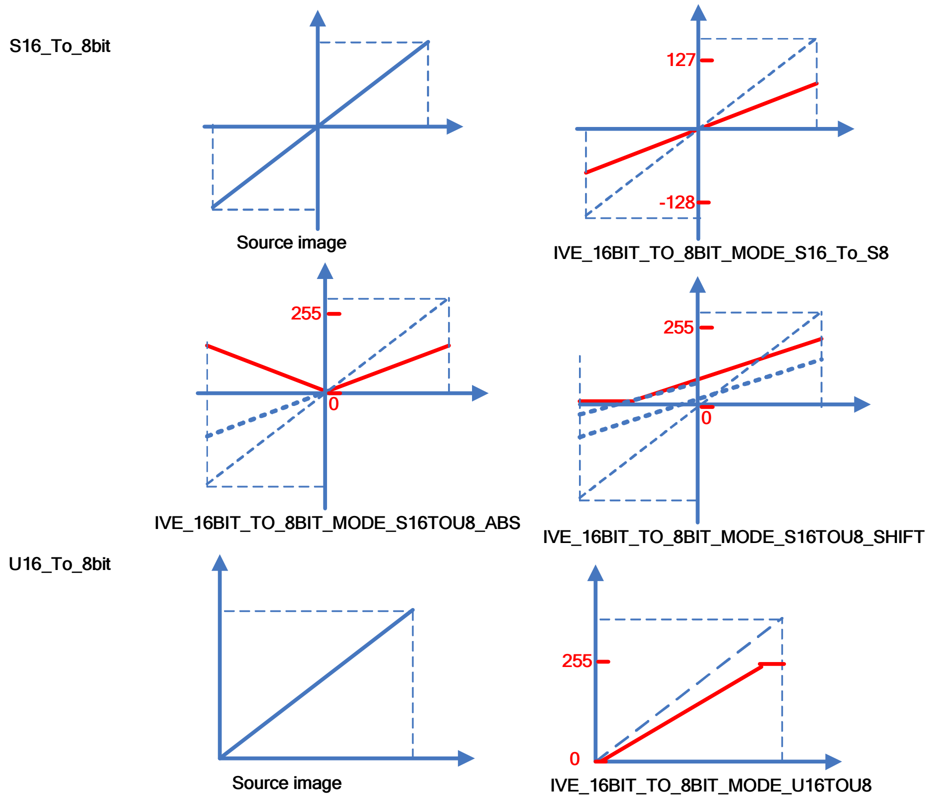
  - IVE_16BIT_TO_8BIT_MODE_U16_TO_U8

$$I_{out}(x,y)=\begin{cases}0 & (\frac{a}{b}I(x,y)<0)\\ \frac{a}{b}I(x,y) & (0\le\frac{a}{b}I(x,y)\le255)\\ 255 & (\frac{a}{b}I(x,y)>255)\end{cases}$$

$I(x,y)$ corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, and **mode**, **a**, **b**, and **bias** correspond to **enMode**, **u8Numerator**, **u16Denominator**, and **s8Bias** in **pst16BitTo8BitCtrl** respectively. For details, see Figure 2-11.

The following requirement must be met: **u8Numerator** ≤ **u16Denominator** and **u16Denominator** ≠ **0**.

Figure 2-11 Four conversion modes from 16-bit data to 8-bit data



[Example]

None

[See Also]

- HI_MPI_IVE_Thresh_S16
- HI_MPI_IVE_Thresh_U16

HI_MPI_IVE_OrdStatFilter

[Description]

Creates a 3x3 template order statistics filter task for median, maximum, or minimum filtering.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstOrdStatFltCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstDst	U8C1	16 bytes	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Three filtering modes are supported. For details, see IVE_ORD_STAT_FILTER_MODE_E.
- The following are related formulas:
  - IVE_ORD_STAT_FILTER_MODE_MEDIAN

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

- IVE_ORD_STAT_FILTER_MODE_MAX

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

- IVE_ORD_STAT_FILTER_MODE_MIN

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

$I(x,y)$ corresponds to **pstSrc**, and $I_{out}(x,y)$ corresponds to **pstDst**.

[Example]

None

[See Also]

- HI_MPI_IVE_Filter
- HI_MPI_IVE_Dilate
- HI_MPI_IVE_Erode

HI_MPI_IVE_Map

[Description]

Creates a map task for searching for the values in the lookup table corresponding to each pixel in the source image and assigning the values to the pixels.

[Syntax]

```
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,HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstMap	Pointer to mapping table information It cannot be null. The minimum memory size is **sizeof(**IVE_MAP_LUT_MEM_S**)**.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output

Parameter	Description	Input/Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	1 byte	64 x 64 to 1920 x 1080
pstMap	-	16 bytes	-
pstDst	U8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Supported
Hi3518E V200	Supported
Hi3531A V100	Supported
Hi3519 V100	Not supported
Hi3519 V101	Not supported
Hi3516C V300	Not supported
Hi3559 V100	Not supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The following is the formula:

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

$I(x, y)$ corresponds to **pstSrc**, $I_{out}(x, y)$ corresponds to **pstDst**, and map corresponds to **pstMap**.

[Example]

None

[See Also]

None

HI_MPI_IVE_Map

[Description]

Creates a map task for searching for the values in the lookup table corresponding to each pixel in the source image and assigning the values to the pixels. Three mapping modes are supported, including U8C1 to U8C1, U8C1 to U16C1, and U8C1 to S16C1.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstMap	Pointer to mapping table information It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstMapCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1080
pstMap	-	16 bytes	-
pstDst	U8C1, U16C1, and S16C1	16 bytes	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

- The formula is as follows:

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

  I(x,y) corresponds to **pstSrc**, **I_{out}(x,y)** corresponds to **pstDst**, and **map** corresponds to **pstMap**.
- The memory configuration of **pstMap** varies according to the configuration of **pstMapCtrl→enMode**:

- If **pstMapCtrl→enMode** is IVE_MAP_MODE_U8, set **pstMap** to **sizeof(IVE_MAP_U8BIT_LUT_MEM_S)**.
- If **pstMapCtrl→enMode** is IVE_MAP_MODE_U16, set **pstMap** to **sizeof(IVE_MAP_U16BIT_LUT_MEM_S)**.
- If **pstMapCtrl→enMode** is IVE_MAP_MODE_S16, set **pstMap** to **sizeof(IVE_MAP_S16BIT_LUT_MEM_S)**.

[Example]

None

[See Also]

None

HI_MPI_IVE_EqualizeHist

[Description]

Creates a histogram equalization task for gray-scale images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstEqualizeHistCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1080

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstDst	U8C1	16 bytes	64 x 64 to 1920 x 1080
pstEqualizeHistCtrl→stMem	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The minimum value of **stMem** in **pstEqualizeHistCtrl** is **sizeof(**IVE_EQUALIZE_HIST_CTRL_MEM_S**)** bytes.
- The histogram equalization calculation process is the same as that of OpenCV.

[Example]

None

[See Also]

None

HI_MPI_IVE_Add

[Description]

Creates a weighted addition task for two gray-scale images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc1	Pointer to source image 1 It cannot be null.	Input
pstSrc2	Pointer to source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc1**.	Output
pstAddCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc1	U8C1	1 byte	64 x 64 to 1920 x 1080
pstSrc2	U8C1	1 byte	64 x 64 to 1920 x 1080
pstDst	U8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The following is the formula:

$$I_{out}(i,j) = x * I_1(i,j) + y * I_2(i,j)$$

$I_1(i,j)$ corresponds to **pstSrc1**, $I_2(i,j)$ corresponds to **pstSrc2**, $I_{out}(i,j)$ corresponds to **pstDst**, and **x** and **Y** correspond to **u0q16X** and **u0q16Y** in **pstAddCtrl**. If the sum of x and y are greater than 1 before fix-point processing, the lower 8 bits are used as the final result when the calculation result exceeds 8 bits.

[Example]

None

[See Also]

HI_MPI_IVE_Sub

HI_MPI_IVE_Xor

[Description]

Creates an XOR task for two binary images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc1	Pointer to source image 1 It cannot be null.	Input
pstSrc2	Pointer to source image 1 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc1**.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc1	U8C1	1 byte	64 x 64 to 1920 x 1080
pstSrc2	U8C1	1 byte	64 x 64 to 1920 x 1080

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstDst	U8C1	1 byte	64 x 64 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The following is the formula:

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

$I_{src1}(x,y)$ corresponds to **pstSrc1**, $I_{src2}(x,y)$ corresponds to **pstSrc2**, and $I_{dst}(x,y)$ corresponds to **pstDst**.

[Example]

None

[See Also]

- HI_MPI_IVE_And
- HI_MPI_IVE_Or

HI_MPI_IVE_NCC

[Description]

Creates an NCC coefficient calculation task for two gray-scale images with the same resolution.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc1	Pointer to source image 1 It cannot be null.	Input
pstSrc2	Pointer to source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstDst	Pointer to output data It cannot be null. The minimum memory size is **sizeof(**IVE_NCC_DST_MEM_S**)**.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc1	U8C1	1 byte	32 x 32 to 1920 x 1080
pstSrc2	U8C1	1 byte	32 x 32 to 1920 x 1080
pstDst	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The following is the formula:

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

- Only the the numerator and two denominators before root extraction are output. That is, **pstDst→u64Numerator**, **pstDst→u64QuadSum1**, and **pstDst→u64QuadSum2** correspond to $\sum_{i=1}^{w}\sum_{j=1}^{h}(I_{src1}(i,j) * I_{src2}(i,j))$, $\sum_{i=1}^{w}\sum_{j=1}^{h}(I_{src1}^{2}(i,j))$, and $\sum_{i=1}^{w}\sum_{j=1}^{h}(I_{src2}^{2}(i,j))$ respectively.

[Example]

None

[See Also]

None

HI_MPI_IVE_CCL

[Description]

Creates a CCL task for binary images.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrcDst	Pointer to the source image. The source image is modified during CCL, that is, the source image is also the labeling image output. The pointer cannot be null.	Input, output
pstBlob	Pointer to connected component information It cannot be null. The minimum memory size is **sizeof(**IVE_CCBLOB_S**)**, and a maximum of 254 valid connected components are output.	Output
pstCclCtrl	Pointer to the control parameter It cannot be null.	Input

Parameter	Description	Input/Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrcDst	U8C1	16 bytes	See the **Chip Difference** field.
pstBlob	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	**pstSrcDst** resolution: 64 x 64 to 720 x 640 Only the 8-connected-component mode is supported.
Hi3536 V100	**pstSrcDst** resolution: 64 x 64 to 720 x 640 Only eight connected components are supported.
Hi3521A V100	**pstSrcDst** resolution: 64 x 64 to 720 x 640 Only eight connected components are supported.
Hi3518E V200	**pstSrcDst** resolution: 64 x 64 to 720 x 640 Only eight connected components are supported.
Hi3531A V100	**pstSrcDst** resolution: 64 x 64 to 720 x 640 Only eight connected components are supported.
Hi3519 V100	**pstSrcDst** resolution: 64 x 64 to 1280 x 720 Both the 4-connected-component mode and 8-connected-component mode are supported.
Hi3519 V101	**pstSrcDst** resolution: 64 x 64 to 1280 x 720 Both the 4-connected-component mode and 8-connected-component mode are supported.

Chip	Difference
Hi3516C V300	**pstSrcDst** resolution: 64 x 64 to 1280 x 720 Both the 4-connected-component mode and 8-connected-component mode are supported.
Hi3559 V100	**pstSrcDst** resolution: 64 x 64 to 1280 x 720 Both the 4-connected-component mode and 8-connected-component mode are supported.

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Connected component information is stored in **pstBlob→astRegion**.
- **pstBlob→u8RegionNum** indicates the number of valid connected components, and its maximum value is 254. The area of valid connected components is greater than **pstBlob→u16CurAreaThr,** and the label ID is the subscript of the **pstBlob→astRegion** array element plus 1. Valid connected components may be stored in the array inconsecutively.
- If **pstBlob→s8LabelStatus** is **0**, components are labeled successfully (a label for a component). If **pstBlob→s8LabelStatus** is **−1**, components fail to be labeled (a label for a component or a label shared by multiple components). In this case, you can label components again based on external rectangle information in **pstBlob**. No matter whether a connected component is labeled successfully, its external rectangle information is correct and available.
- The output connected components are filtered based on **pstCclCtrl→u16InitAreaThr**. The components whose area is less than or equal to **pstCclCtrl→u16InitAreaThr** are set to **0**.
- If there are more than 254 connected components, the components with a smaller area size are deleted based on **pstCclCtrl→u16InitAreaThr**. If **pstCclCtrl→u16InitAreaThr** is too small to delete components, the area threshold for connected components is increased by step of **pstCclCtrl→u16Step**.
- The final area threshold is stored in **pstBlob→u16CurAreaThr**.

[Example]

None

[See Also]

None

HI_MPI_IVE_GMM

[Description]

Creates a GMM background modeling task for gray-scale images or RGB_PACKAGE images. Three or five GMMs are supported.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstFg	Pointer to the foreground image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstBg	Pointer to the background image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstModel	Pointer to the GMM parameter It cannot be null.	Input, output
pstGmmCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1 and U8C3_PACKAGE	16 bytes	See the **Chip Difference** field.
pstFg	U8C1 binary image	16 bytes	See the **Chip Difference** field.
pstBg	U8C1 and U8C3_PACKAGE	16 bytes	See the **Chip Difference** field.
pstModel	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported **pstSrc** resolution: 720 x 576
Hi3536 V100	Supported **pstSrc** resolution: 720 x 576
Hi3521A V100	Supported **pstSrc** resolution: 720 x 576
Hi3518E V200	Not supported
Hi3531A V100	Supported **pstSrc** resolution: 720 x 576
Hi3519 V100	Supported **pstSrc** resolution: 1280 x 720
Hi3519 V101	Supported **pstSrc** resolution: 1280 x 720
Hi3516C V300	Supported **pstSrc** resolution: 1280 x 720
Hi3559 V100	Supported **pstSrc** resolution: 1280 x 720

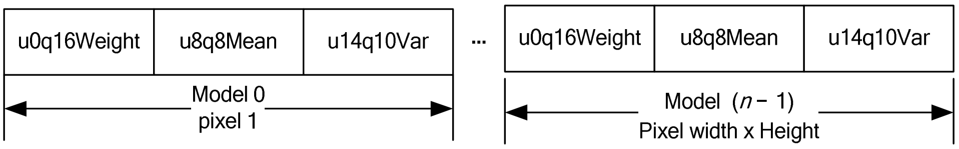
[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- GMMs are implemented by referring to MOG and MOG2 in the OpenCV.
- The source image type must be U8C1 or U8C3_PACKAGE during GMM background modeling for gray-scale images or RGB images respectively.
- The foreground image must be a binary image and its type must be U8C1. The background image type must be the same as the source image type.
- Multiple GMMs are used for gray-scale images. Figure 2-12 shows the memory allocation mode of **pstModel**.

Figure 2-12 Memory allocation of GMMs for gray-scale images

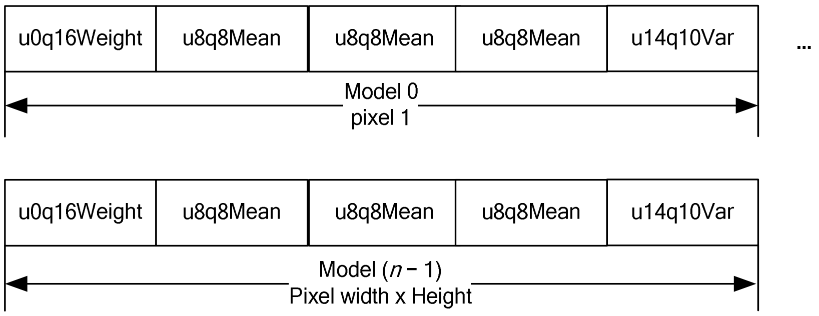


For a GMM of a pixel, its **weight** parameter occupies two bytes, its **mean** parameter occupied 2 bytes, and its **var** parameter occupies three bytes. The following is the formula for calculating the memory size required by **pstModel**:

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

- Multiple GMMs are used for RGB images. Figure 2-13 shows the memory allocation mode of **pstModel**.

Figure 2-13 Memory allocation of GMMs for RGB images



For a GMM of a pixel, its **weight** parameter occupies two bytes, its **mean[3]** parameter occupied 2x3 bytes, and its **var** parameter occupies three bytes. The following is the formula for calculating the memory size required by **pstModel**:

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

[Example]

None

[See Also]

- HI_MPI_IVE_MatchBgModel
- HI_MPI_IVE_UpdateBgModel
- HI_MPI_IVE_GMM2

HI_MPI_IVE_GMM2

[Description]

Creates a GMM background modeling task for gray-scale input images and RGB_PACKAGE input images. One to five GMMs are supported. The global or pixel-level sensitivity coefficient and foreground model duration update coefficient are supported.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstFactor	Pointer to the model update parameter pstFactor can be null only when both **pstGmm2Ctrl→enSnsFactorMode** and **pstGmm2Ctrl→enLifeUpdateFactorMode** are set to the global mode.	Input
pstFg	Pointer to the foreground image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstBg	Pointer to the background image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstMatchModelInfo	Pointer to the model matching coefficient It cannot be null.	Output
pstModel	Pointer to the GMM parameter It cannot be null.	Input/Output
pstGmm2Ctrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1 and U8C3_PACKAGE	16 bytes	64 x 64 to 1280 x 720
pstFactor	U16C1	16 bytes	64 x 64 to 1280 x 720
pstFg	U8C1 binary image	16 bytes	64 x 64 to 1280 x 720
pstBg	U8C1 and U8C3_PACKAGE	16 bytes	64 x 64 to 1280 x 720
pstMatchModelInfo	U8C1	16 bytes	64 x 64 to 1280 x 720
pstModel	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Supported
Hi3559 V100	Supported

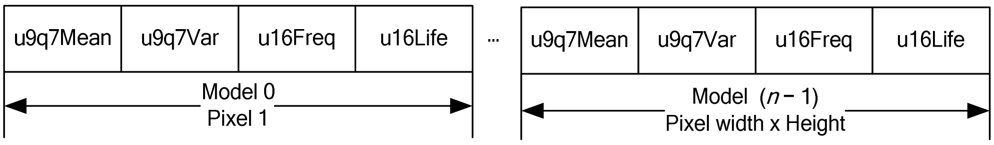
[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The pixel-level parameter control is added to GMM2 by referring to MOG and MOG2 of the OpenCV.
- The source image type of **pstSrc** must be U8C1 or U8C3_PACKAGE during GMM background modeling for gray-scale images or RGB images respectively.
- The model update parameter **pstFactor** indicates the U16C1 image type. Each element is represented by 16 bits. The lower eight bits are the sensitivity coefficient for controlling the variance multiple during model matching, and the upper eight bits are the foreground model duration update coefficient.
- The pointer to the model matching coefficient (**pstMatchModelInfo**) indicates the U8C1 image. Each element is represented by eight bits. The lowermost one bit is the Gaussian model matching flag. If this bit is 0, the matching fails; if this bit is 1, the matching is successful. The upper seven bits indicate the number of the model with the highest frequency.
- The GMM2 frequency parameters (**u16FreqInitVal**, **u16FreqReduFactor**, **u16FreqAddFactor**, and **u16FreqThr** in **pstGmm2Ctrl**) are used to control the model sorting and model validity time.
  - A larger **u16FreqInitVal** indicates longer model validity time.
  - A larger **u16FreqReduFactor** indicates longer model validity time. The model frequency is attenuated after it is multiplied by the frequency attenuation coefficient (**u16FreqReduFactor**/65536).
  - A larger **u16FreqAddFactor** indicates longer model validity time.
  - A larger **u16FreqThr** indicates shorter model validity time.
- The GMM2 model duration parameter (**u16LifeThr** in **pstGmm2Ctrl**) is used to control the time for converting a foreground model into a background model.
  - A larger **u16LifeThr** indicates longer foreground duration.
  - The model duration parameter does not take effect when a single Gaussian model is used.
- *n* (1 ≤ *n* ≤ 5) GMM2s are used for the gray-scale image. Figure 2-14 shows the memory allocation mode of **pstModel**.

Figure 2-14 Memory allocation of GMM2s for gray-scale images

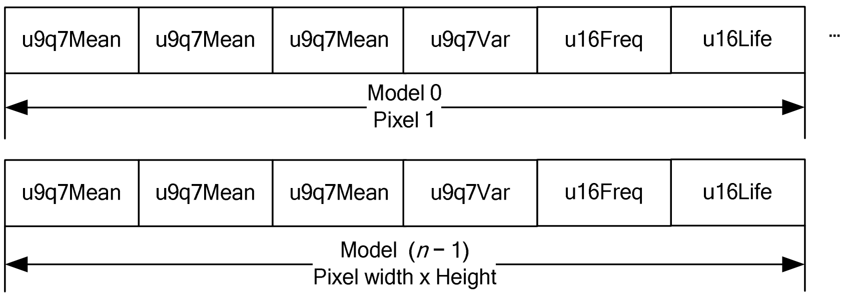


For a GMM of a pixel, its **mean**, **var**, **freq**, and **life** parameters each occupies two bytes. The following is the formula for calculating the memory size required by **pstModel**:

pstModel→u32Size = 8 x pstSrc→u16Width x pstSrc→u16Height x pstGmm2Ctrl→u8ModelNum

- *n* (1 ≤ *n* ≤ 5) GMM2s are used for the RGB image. Figure 2-15 shows the memory allocation mode of **pstModel**.

Figure 2-15 Memory allocation of GMM2s for RGB images



For a GMM of a pixel, its **mean[3]** parameter occupies six bytes, and its **var**, **freq**, and **life** parameters each occupies two bytes. The following is the formula for calculating the memory size required by **pstModel**:

pstModel→u32Size = 12 x pstSrc→u16Width x pstSrc→u16Height x pstGmm2Ctrl→u8ModelNum

[Example]

None

[See Also]

- HI_MPI_IVE_MatchBgModel
- HI_MPI_IVE_UpdateBgModel
- HI_MPI_IVE_GMM

HI_MPI_IVE_CannyHysEdge

[Description]

Creates a Canny edge extraction task for gray-scale images (latter phase of Canny edge extraction for gray-scale images) for calculating the gradient, gradient magnitude and argument, hysteresis threshold, and non-maximum suppression.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input

Parameter	Description	Input/Output
pstEdge	Pointer to the strong/weak edge flag image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstStack	Coordinate stack of the strong edge point It cannot be null. The minimum memory size must be **pstSrc→u16Width** x **pstSrc→u16Height** x **(sizeof(**IVE_POINT_U16_S**))** + **sizeof(**IVE_CANNY_STACK_SIZE_S**)**.	Output
pstCannyHysEdgeCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstEdge	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstStack	-	16 bytes	-
pstCannyHysEdgeCtrl→stMem	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- **pstEdge** can be only **0**, **1**, or **2**.
  - 0: weak edge point

  - 1: non-edge point
  - 2: strong edge point
- Coordinate information about the strong edge point is stored in **pstStack**.
- The following is the formula for calculating the minimum memory size of **pstCannyHysEdgeCtrl→stMem**:

  pstCannyHysEdgeCtrl→stMem.u32Size = IveGetStride(pstSrc→u16Width, IVE_STRIDE_ALIGN) x 3 x pstSrc→u16Height
- After this task is complete, a Canny edge image is output only when HI_MPI_IVE_CannyEdge is called.

[Example]

None

[See Also]

HI_MPI_IVE_CannyEdge

HI_MPI_IVE_CannyEdge

[Description]

Connects edge points to form a Canny image (latter phase of Canny edge extraction for gray-scale images).

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
pstEdge	Pointer to the strong/weak edge flag image as the input or pointer to the edge binary image as the output It cannot be null.	Input, output
pstStack	Coordinate stack of the strong edge point It cannot be null.	Input, output

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstEdge	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstStack	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

Call HI_MPI_IVE_CannyHysEdge before calling HI_MPI_IVE_CannyEdge. That is, ensure that the HI_MPI_IVE_CannyHysEdge task is complete and use the outputs **pstEdge** and **pstStack** of HI_MPI_IVE_CannyHysEdge as the parameter inputs of HI_MPI_IVE_CannyEdge.

[Example]

None

[See Also]

HI_MPI_IVE_CannyHysEdge

HI_MPI_IVE_LBP

[Description]

Creates an LBP calculation task.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to the output image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstLbpCtrl	Pointer to control information It cannot be null.	Input

Parameter	Description	Input/Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstDst	U8C1	16 bytes	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

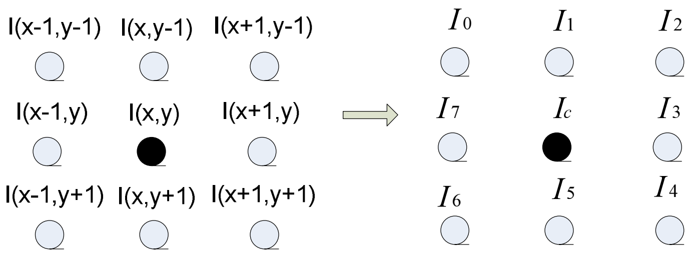
[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

Figure 2-16 shows the LBP formula.

Figure 2-16 LBP formula



- IVE_LBP_CMP_NORMAL

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

- IVE_LBP_CMP_ABS

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

$I(x,y)$ corresponds to **pstSrc**, $lpb(x,y)$ corresponds to **pstDst**, and thr corresponds to **pstLbpCtrl→un8BitThr**.

[Example]

None

[See Also]

None

HI_MPI_IVE_NormGrad

[Description]

Creates a normalized gradient calculation task. All gradient components are normalized to S8.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input

Parameter	Description	Input/Output
pstDstH	Pointer to the gradient component image H that is obtained after filtering based on the template and normalization to S8 It cannot be null if the output is required based on **pstNormGradCtrl→enOutCtrl**.	Output
pstDstV	Pointer to the gradient component image V that is obtained after filtering based on the transposed template and normalization to S8 It cannot be null if the output is required based on **pstNormGradCtrl→enOutCtrl**.	Output
pstDstHV	Pointer to the image that is stored in package format (see Figure 1-7) and obtained after filtering based on the command template and transposed template and normalization to S8 It cannot be null if the output is required based on **pstNormGradCtrl→enOutCtrl**.	Output
pstNormGradCtrl	Pointer to control information	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1024
pstDstH	S8C1	16 bytes	64 x 64 to 1920 x 1024
pstDstV	S8C1	16 bytes	64 x 64 to 1920 x 1024
pstDstHV	S8C2_PACKAGE	16 bytes	64 x 64 to 1920 x 1024

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

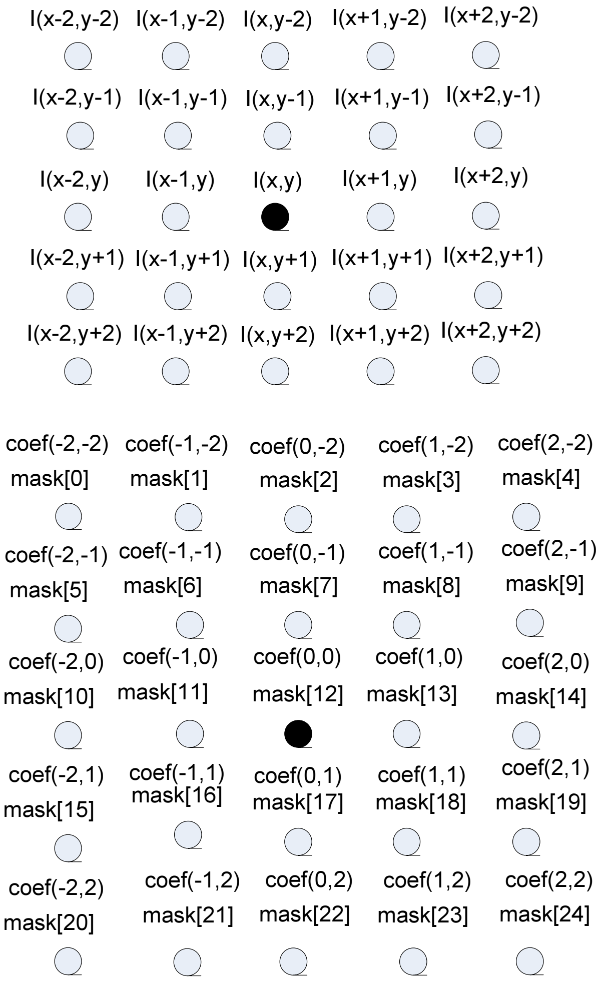
[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The following output modes are supported:
  - In **IVE_NORM_GRAD_OUT_CTRL_HOR_AND_VER** mode, the **pstDstH** and **pstDstV** pointers cannot be null, and their strides must be the same.
  - In **IVE_NORM_GRAD_OUT_CTRL_HOR** mode, the **pstDstH** pointer cannot be null.
  - In **IVE_NORM_GRAD_OUT_CTRL_VER** mode, the **pstDstV** pointer cannot be null.
  - In **IVE_NORM_GRAD_OUT_CTRL_COMBINE** mode, the **pstDstHV** pointer cannot be null.
- Figure 2-17 shows the NormGrad formula.

Figure 2-17 NormGrad formula


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

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

[Example]

None

[See Also]

HI_MPI_IVE_Sobel

HI_MPI_IVE_LKOpticalFlow

[Description]

Creates a single-layer LK optical flow calculation task.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrcPre	Pointer to the previous frame It cannot be null.	Input
pstSrcCur	Pointer to the current image It cannot be null. The height and width are the same as those of **pstSrcPre**.	Input
pstPoint	Pointer to the coordinate of the initial feature point at the current layer of the pyramid It cannot be null. The coordinate type must be IVE_POINT_S25Q7_S, and the minimum memory size is **pstLkOptiFlowCtrl→u16CornerNum** x **sizeof(**IVE_POINT_S25Q7_S**)**.	Input
pstMv	Pointer to the displacement vector of the feature point corresponding to **pstPoint** It cannot be null. The input needs to be initialized as 0 for the initial calculation. The displacement vector obtained in the calculation of the previous layer needs to be entered for the calculation of subsequent layers. The displacement vector type must be IVE_MV_S9Q7_S, and the minimum memory size is **pstLkOptiFlowCtrl→u16CornerNum** x	Input, output

Parameter	Description	Input/Output
	sizeof(IVE_MV_S9Q7_S**).**	
pstLkOptiFlowCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrcPre	U8C1	16 bytes	64 x 64 to 720 x 576
pstSrcCur	U8C1	16 bytes	64 x 64 to 720 x 576

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Not supported
Hi3519 V101	Not supported
Hi3516C V300	Not supported
Hi3559 V100	Not supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.0.lib used on the PC for simulation)

[Note]

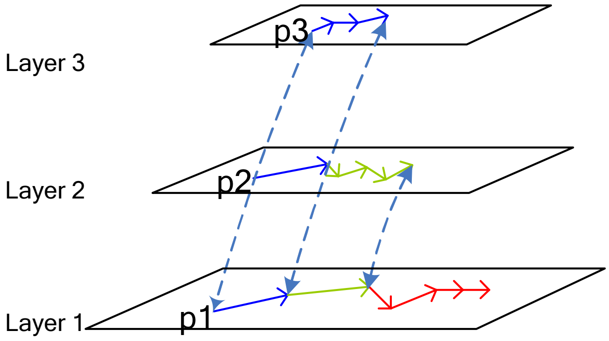
- Only the 7 x 7 pixels around the feature point are used to calculate I_x, I_y, and I_t in the following optical flow equation:

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

  I_x indicates the horizontal offset of the current image, I_y indicates the vertical offset of the current image, and I_t indicates the difference between the current image and the previous frame.

- Figure 2-18 shows the LK optical flow calculation by using a 3-layer pyramid as an example. It is required that the height and width of the image at a layer be half of those of the image at the upper-level layer.

Figure 2-18 LK optical flow calculation of a 3-layer pyramid



- Calculate the coordinates (p0, p1, and p2) corresponding to the feature points of the 3-layer pyramid based on the input feature point coordinates.
- Invoke an LK operator by using p2 and mv2 (initialized as 0) as inputs to calculate the displacement mv2 at layer 2.
- Invoke an LK operator by using p1 and mv2 as inputs to calculate the displacement mv1 at layer 1.
- Invoke an LK operator by using p0 and mv1 as inputs to calculate the displacement mv0 at layer 0.
- If the image at layer 0 is not the source image, obtain the actual displacement mv of the LK optical flow based on the ratio of the image at layer 0 and the source image.

Note that each feature point is calculated based on the data of a fixed widow that is centered on the feature point. During iterative calculation, if the target displacement point of the feature point does not fall within the fixed window, optical flow calculation fails.

[Example]

None

[See Also]

None

HI_MPI_IVE_LKOpticalFlowPyr

[Description]

Creates a multi-layer pyramid LK optical flow calculation task.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
astSrcPrevPyr[]	Pyramid image array of the previous frame of image. The number of pyramid layers is controlled by **pstLkOptiFlowPyrCtrl→u8MaxLevel**. It cannot be null.	Input
astSrcNextPyr[]	Pyramid image array of the next frame of image. **astSrcNextPyr[]** and **astSrcPrevPyr[]** have the same number of pyramid layers. The sizes and types of images at each layer are the same. It cannot be null.	Input
pstPrevPts	Initial feature point array for pyramid layer 0 of the previous frame of image It cannot be null. At most 500 feature points are supported, and the coordinate type must be IVE_POINT_S25Q7_S. The minimum size of the required memory is calculated as follows: Minimum memory size = pstLkOptiFlowPyrCtrl→u16PtsNum x sizeof(IVE_POINT_S25Q7_S).	Input
pstNextPts	Coordinates of the pyramid layer 0 (astSrcNextPyr**[]**) of the next frame of image, obtained after the pyramid LK optical flow calculation of the feature point array **pstPrevPts**. This feature point array (**pstNextPts**) needs to be initialized when **pstLkOptiFlowPyrCtrl→bUseInitFlow** is **true**. It cannot be null.	Input/Output

Parameter	Description	Input/Output
	The number of supported feature points for **pstNextPts** is the same as that of **pstPrevPts**, and the coordinate type must be IVE_POINT_S25Q7_S. The minimum size of the required memory is calculated as follows: Minimum memory size = pstLkOptiFlowPyrCtrl→u16PtsNum x sizeof(IVE_POINT_S25Q7_S).	
pstStatus	HI_U8 tracing status information about each feature point in **pstNextPts**. The value 1 indicates that the tracking is successful, and the value 0 indicates that the tracking fails.	Output
pstErr	Estimated similarity error (HI_U9Q7 type) obtained by comparing the feature points in **pstNextPts** that are successfully traced with the surrounding points of the corresponding feature points in **pstPrevPts**. The feature points that fail to be traced are not estimated.	Output
pstLkOptiFlowPyrCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
astSrcPrevPyr[0]	U8C1	16 bytes	64 x 64 to 1280 x 720
astSrcNextPyr[0]	U8C1	16 bytes	astSrcPrevPyr[0]
astSrcPrevPyr[n] *n* layer of the pyramid (0 ≤ *n* ≤ 3)	U8C1	16 bytes	Rightward shift *n* of the height and width corresponding to **astSrcPrevPyr[0]**
astSrcNextPyr[n] *n* layer of the pyramid	U8C1	16 bytes	Rightward shift *n* of the height and width corresponding to **astSrcPrevPyr[0]**
pstPrevPts	-	16 bytes	-
pstNextPts	-	16 bytes	-
pstStatus	-	16 bytes	-
pstErr	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

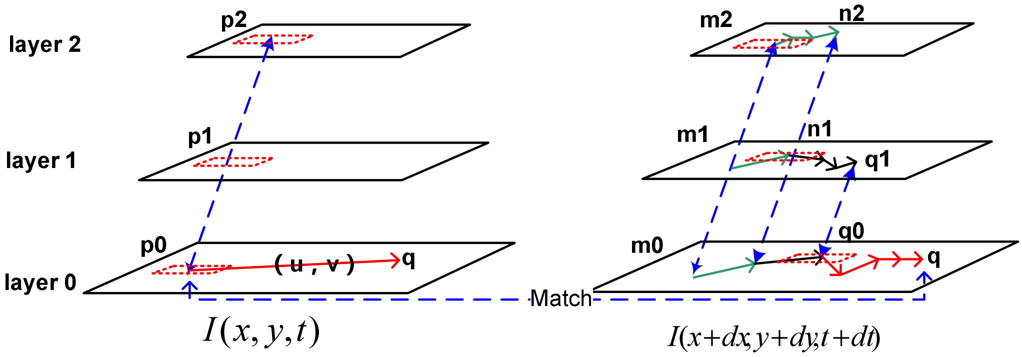
- The value range of **pstLkOptiFlowPyrCtrl→u8MaxLevel** is [0, 3], and the number of corresponding pyramid layers ranges from 1 to 4.
- Only the 7 x 7 pixels around the feature point are used to calculate I_x, I_y, and I_t in the following optical flow equation:

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

  where

  I_x indicates the horizontal offset of the current image, I_y indicates the vertical offset of the current image, and I_t indicates the difference between the current image and the previous frame of image.
- Figure 2-19 shows the LK optical flow calculation by using a 3-layer pyramid as an example. It is required that the height and width of the image at a layer be half of those of the image at the upper-level layer.

Figure 2-19 LK optical flow calculation of a 3-layer pyramid



- Calculate the corresponding coordinates (p0, p1, and p2) of the 3-layer pyramid feature points based on the input feature point coordinates. Calculate m0, m1, and m2 if the initial optical flow is required. Otherwise, the relationship is as follows: m0 = p0, m1 = p1, m2 = p2.
- Calculate the optical flow end point n2 at layer 2 by using m2 as the input.
- Calculate corresponding coordinate (n1) of n2 at layer 1, and calculate the optical flow end point q1 at layer 1 by using n1 as the input.
- Calculate corresponding coordinate (q0) of q1 at layer 1, and calculate the optical flow end point q at layer 0 by using q0 as the input.
- If the image at layer 0 is not the source image, obtain the final end point p of the LK optical flow based on the ratio of the image at layer 0 to the source image.

Note that each feature point is calculated based on the data of a fixed widow that is centered on the feature point. During iterative calculation, if the target displacement point of the feature point does not fall within the fixed window, optical flow calculation fails.

[Example]

None

[See Also]

None

HI_MPI_IVE_STCandiCorner

[Description]

Calculates candidate corner points (first phase of Shi-Tomasi-like corner point calculation for gray-scale images).

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the source image It cannot be null.	Input
pstCandiCorner	Pointer to the candidate corner point response image It cannot be null. The height and width are the same as those of **pstSrc**.	Output
pstStCandiCornerCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1	16 bytes	See the **Chip Difference** field.
pstCandiCorner	U8C1	16 bytes	See the **Chip Difference** field.
pstStCandiCornerCtrl→stMem	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported **pstSrc** resolution: 64 x 64 to 720 x 576
Hi3536 V100	Supported **pstSrc** resolution: 64 x 64 to 720 x 576
Hi3521A V100	Supported **pstSrc** resolution: 64 x 64 to 720 x 576
Hi3518E V200	Not supported
Hi3531A V100	Supported **pstSrc** resolution: 64 x 64 to 720 x 576
Hi3519 V100	Supported **pstSrc** resolution: 64 x 64 to 1280 x 720
Hi3519 V101	Supported **pstSrc** resolution: 64 x 64 to 1280 x 720
Hi3516C V300	Not supported
Hi3559 V100	Supported **pstSrc** resolution: 64 x 64 to 1280 x 720

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The calculation is similar to the ShiTomas corner point calculation in OpenCV.
- The following is the formula for calculating the minimum memory size of **pstStCandiCornerCtrl→stMem**:

  pstStCandiCornerCtrl→stMem.u32Size = 4 x IveGetStride(pstSrc→u16Width, IVE_STRIDE_ALIGN) x pstSrc→u16Height + sizeof(IVE_ST_MAX_EIG_S)
- After this task is complete, a corner point is obtained only when HI_MPI_IVE_STCorner is called.

[Example]

None

[See Also]

HI_MPI_IVE_STCorner

HI_MPI_IVE_STCorner

[Description]

Selects corner points based on rules (latter phase of Shi-Tomasi-like corner point calculation for gray-scale images).

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
pstCandiCorner	Pointer to the candidate corner point response image It cannot be null.	Input
pstCorner	Pointer to corner point coordinate information It cannot be null. The minimum memory size is **sizeof(**IVE_ST_CORNER_INFO_S**)**.	Output
pstStCornerCtrl	Pointer to the control parameter It cannot be null.	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstCandiCorner	U8C1	16 bytes	See the **Chip Difference** field.
pstCorner	-	16 bytes	-

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	**pstSrc** resolution: 64 x 64 to 720 x 576 At most 200 corner points can be output.
Hi3536 V100	**pstSrc** resolution: 64 x 64 to 720 x 576 At most 200 corner points can be output.

Chip	Difference
Hi3521A V100	**pstSrc** resolution: 64 x 64 to 720 x 576 At most 200 corner points can be output.
Hi3518E V200	Not supported
Hi3531A V100	**pstSrc** resolution: 64 x 64 to 720 x 576 At most 200 corner points can be output.
Hi3519 V100	**pstSrc** resolution: 64 x 64 to 1280 x 720 At most 500 corner points can be output.
Hi3519 V101	**pstSrc** resolution: 64 x 64 to 1280 x 720 At most 500 corner points can be output.
Hi3516C V300	Not supported
Hi3559 V100	**pstSrc** resolution: 64 x 64 to 1280 x 720 At most 500 corner points can be output.

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The calculation is similar to the ShiTomas corner point calculation in OpenCV.
- **pstCorner→u16CornerNum** indicates the number of obtained corner numbers.
- Call HI_MPI_IVE_STCandiCorner before calling HI_MPI_IVE_STCorner. That is, ensure that the HI_MPI_IVE_STCandiCorner task is complete and use the output **pstCandiCorner** of HI_MPI_IVE_STCandiCorner as the parameter input of HI_MPI_IVE_STCorner.

[Example]

None

[See Also]

HI_MPI_IVE_STCandiCorner

HI_MPI_IVE_SAD

[Description]

Calculates the SAD of two source images in 4x4, 8x8, or 16x16 blocking mode, and outputs the 16-bit/8-bit SAD images as well as SAD thresholding image.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc1	Pointer to source image 1 It cannot be null.	Input
pstSrc2	Pointer to source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstSad	Pointer to the output SAD image It cannot be null if the output is required based on **pstSadCtrl→enOutCtrl**. According to **pstSadCtrl→enMode**, the height and width are 1/4, 1/8, or 1/16 those of **pstSrc1** in 4x4, 8x8, or 16x16 blocking mode respectively.	Output
pstThr	Pointer to the output SAD thresh image It cannot be null if the output is required based on **pstSadCtrl→enOutCtrl**. According to **pstSadCtrl→enMode**, the height and width are 1/4, 1/8, or 1/16 those of **pstSrc1** in 4x4, 8x8, or 16x16 blocking mode respectively.	Output
pstSadCtrl	Pointer to the control information It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstSrc1	U8C1	1 byte	64 x 64 to 1920 x 1080
pstSrc2	U8C	1 byte	64 x 64 to 1920 x 1080
pstSad	U8C1, U16C1	16 bytes	According to **pstSadCtrl→enMode**, the height and width are 1/4, 1/8, or 1/16 those of **pstSrc1** in 4x4, 8x8, or 16x16 blocking mode respectively.

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstThr	U8C1	16 bytes	According to **pstSadCtrl→enMode**, the height and width are 1/4, 1/8, or 1/16 those of **pstSrc1** in 4x4, 8x8, or 16x16 blocking mode respectively.

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Supported
Hi3518E V200	Supported
Hi3531A V100	Supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The following is the calculation formula:

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

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

where $I_1(i,j)$ corresponds to **pstSrc1**, $I_2(i,j)$ corresponds to **pstSrc2**, and $SAD_{out}(x,y)$ corresponds to **pstSad**. *n* is related to **pstSadCtrl→enMode**, and its value is 4, 8, and 16 when it corresponds to IVE_SAD_MODE_MB_4X4, IVE_SAD_MODE_MB_8X8, and IVE_SAD_MODE_MB_16X16 respectively.

$THR_{out}(x,y)$ corresponds to **pstThr**, Thr corresponds to **pstSadCtrl→u16Thr**, $minVal$ corresponds to **pstSadCtrl→u8MinVal**, and $maxVal$ corresponds to **pstSadCtrl→u8MaxVal**.

[Example]

None

[See Also]

None

HI_MPI_IVE_Resize

[Description]

Creates a picture scaling task. The bilinear interpolation scaling and area interpolation scaling are supported. The same type of scaling can be implemented for multiple U8C1_PLANAR or U8C3_PLANAR picture inputs.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
astSrc[]	Source image array It cannot be null.	Input
astDst[]	Output image array It cannot be null.	Output

Parameter	Description	Input/Output
	The type of each picture is the same as that of **astSrc**.	
pstResizeCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
astSrc[]	U8C1 and U8C3_PLANAR	16 bytes	32 x 12 to 1920 x 1080
astDst[]	U8C1 and U8C3_PLANAR	16 bytes	32 x 12 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	For Hi3519 V100, the scaling multiple cannot be 1.
Hi3519 V101	For Hi3519 V101, the scaling multiple can only be 1.
Hi3516C V300	Not supported
Hi3559 V100	For Hi3559 V100, the scaling multiple can only be 1.

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h

- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

- The function of creating a picture scaling task is implemented based on the resize function in the OpenCV. IVE_RESIZE_MODE_LINEAR and IVE_RESIZE_MODE_AREA correspond to INTER_LINEAR and INTER_AREA of the OpenCV resize function respectively.
- The combined inputs of U8C1 and U8C3_PLANAR image arrays are supported. Note that the scaling modes of all the images must be the same.
- At most 16x scaling is supported.
- The minimum size (in byte) of the **pstResizeCtrl→stMem** memory is calculated as follows: Minimum size = 25 x U8C1_NUM + 49 x (pstResizeCtrl→u16Num − U8C1_NUM). **U8C1_NUM** indicates the number of U8C1 images in the combined image arrays.

[Example]

None

[See Also]

HI_MPI_IVE_Resize2

HI_MPI_IVE_Resize2

[Description]

Creates a picture scaling task. The bilinear interpolation scaling is supported. Multiple U8C1 pictures can be scaled at the same time.

[Syntax]

```
HI_S32 HI_MPI_IVE_Resize2(IVE_SRC_IMAGE_S astSrc[],IVE_DST_IMAGE_S
astDst[], IVE_RESIZE2_CTRL_S *pstResize2Ctrl);
```

[Parameter]

Parameter	Description	Input/Output
astSrc[]	Source image array It cannot be null.	Input
astDst[]	Output image array It cannot be null. The type of each picture is the same as that of **astSrc**	Output
pstResize2Ctrl	Pointer to the control parameter It cannot be null.	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
astSrc[]	U8C1	16 byte	32 x 12 to 1920 x 1080

Parameter	Supported Image Type	Address Alignment Mode	Resolution
astDst[]	U8C1	16 byte	32 x 12 to 1920 x 1080

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Not supported
Hi3519 V101	Not supported
Hi3516C V300	Supported
Hi3559 V100	Not supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

At most 15x scaling is supported.

[Example]

None

[See Also]

HI_MPI_IVE_Resize

HI_MPI_IVE_GradFg

[Description]

Calculates the gradient foreground image based on the gradient of the background image and current frame.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstBgDiffFg	Pointer to the foreground image obtained based on the current image and background difference It cannot be null.	Input
pstCurGrad	Pointer to the gradient image of the current frame It cannot be null. The height and width are the same as those of **pstCurGrad**.	Input
pstBgGrad	Pointer to the background gradient image It cannot be null. The height and width are the same as those of **pstCurGrad**.	Input
pstGradFg	Pointer to the gradient foreground image It cannot be null. The height and width are the same as those of **pstCurGrad**.	Output
pstGradFgCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstDiffFg	S8C1	16 bytes	See the **Chip Difference** field.

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstCurGrad	S8C2_PACKAGE	16 bytes	See the **Chip Difference** field.
pstBgGrad	S8C2_PACKAGE	16 bytes	See the **Chip Difference** field.
pstGradFg	U8C1	16 bytes	See the **Chip Difference** field.

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported **pstDiffFg** resolution: 720 x 576
Hi3536 V100	Supported **pstDiffFg** resolution: 720 x 576
Hi3521A V100	Supported **pstDiffFg** resolution: 720 x 576
Hi3518E V200	Not supported
Hi3531A V100	Supported **pstDiffFg** resolution: 720 x 576
Hi3519 V100	Supported **pstDiffFg** resolution: 1280 x 720
Hi3519 V101	Supported **pstDiffFg** resolution: 1280 x 720
Hi3516C V300	Supported **pstDiffFg** resolution: 1280 x 720
Hi3559 V100	Supported **pstDiffFg** resolution: 1280 x 720

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

The type of the background gradient image and current gradient image is S8C2_PACKAGE. The horizontal and vertical gradients are stored in the format of [xyxyxy…].

[Example]

None

[See Also]

- HI_MPI_IVE_MatchBgModel
- HI_MPI_IVE_UpdateBgModel
- HI_MPI_IVE_GMM

HI_MPI_IVE_MatchBgModel

[Description]

Matches the background model based on the codebook evolution.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstCurImg	Pointer to the gray-scale image of the current frame It cannot be null.	Input
pstBgModel	Pointer to background model data It cannot be null. The height is the same as that of **pstCurImg**, and the width is **pstCurImg→u16Width** x **sizeof(**IVE_BG_MODEL_PIX_S**)**.	Input, output
pstFgFlag	Pointer to the foreground status image It cannot be null. The height and width are the same as those of **pstCurImg**.	Input, output

Parameter	Description	Input/Output
pstBgDiffFg	Pointer to the foreground image obtained based on the current image and background difference It cannot be null. The height and width are the same as those of **pstCurImg**.	Output
pstFrmDiffFg	Pointer to the inter-frame differential foreground image It cannot be null. The height and width are the same as those of **pstCurImg**.	Output
pstStatData	Pointer to foreground status data It cannot be null. The minimum buffer size is **sizeof(**IVE_FG_STAT_DATA_S**)**.	Output
pstMatchBgModelCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstCurImg	U8C1	16 bytes	See the **Chip Difference** field.
pstBgModel	-	16 bytes	-
pstFgFlag	U8C1	16 bytes	See the **Chip Difference** field.
pstBgDiffFg	S8C1	16 bytes	See the **Chip Difference** field.
pstFrmDiffFg	S8C1	16 bytes	See the **Chip Difference** field.

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported **pstCurImg** resolution: 720 x 576
Hi3536 V100	Supported **pstCurImg** resolution: 720 x 576
Hi3521A V100	Supported **pstCurImg** resolution: 720 x 576
Hi3518E V200	Not supported
Hi3531A V100	Supported **pstCurImg** resolution: 720 x 576
Hi3519 V100	Supported **pstCurImg** resolution: 1280 x 720
Hi3519 V101	Supported **pstCurImg** resolution: 1280 x 720
Hi3516C V300	Supported **pstCurImg** resolution: 1280 x 720
Hi3559 V100	Supported **pstCurImg** resolution: 1280 x 720

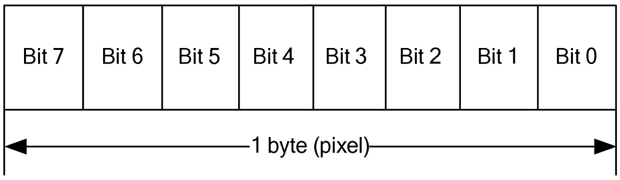
[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The strides of **pstFgFlag**, **pstBgDiffFg**, and **pstFrmDiffFg** are the same.
- Each pixel in **pstBgModel** is expressed by IVE_BG_MODEL_PIX_S(24 bytes). That is, the following conditions are met: **pstModel→u16Width** = **sizeof(**IVE_BG_MODEL_PIX_S**)** x **pstSrc→u16Width**, **pstModel→u16Height** = **pstSrc→u16Heigh**. The minimum memory size is **IveGetStride (sizeof(**IVE_BG_MODEL_PIX_S**)** x **pstSrc→u16Width**, **IVE_STRIDE_ALIGN**) x **pstModel→u16Height**.
- The type of **pstFgFlag** is U8C1. Each bit indicates different status information. Figure 2-20 shows the bits of a pixel. The bits are arranged from lower bits to upper bits.

Figure 2-20 Bits of a pixel of the foreground status flag image



The following describes bit definitions:

- Only bits 0–2, bit 5, and bit 6 are used. Bits 0–2 act as outputs of the current operator calculation, and bit 5 and bit 6 act as inputs of the external function calculation.
- When bit 1 is **1**, the pixel is the foreground pixel.
- When bit 0 and bit 1 is **1**, the pixel is the moving foreground pixel.
- When bit 1 is **1** and bit 0 is **0**, the pixel is the variable foreground pixel.
- When bit 2 is **1**, the pixel background model is working.
- Bit 5 and bit 6 indicate the foreground status feedback from external functions. When bit 5 is **1**, the foreground pixel needs to be retained for a short period; when bit 6 is **1**, the foreground pixel needs to be retained for a long period.

[Example]

None

[See Also]

- HI_MPI_IVE_UpdateBgModel
- HI_MPI_IVE_GradFg
- HI_MPI_IVE_GMM

HI_MPI_IVE_UpdateBgModel

[Description]

Updates the internal status of the background model based on the codebook evolution.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output

Parameter	Description	Input/Output
pstBgModel	Pointer to background model data It cannot be null.	Input, output
pstFgFlag	Pointer to the foreground status image It cannot be null.	Input, output
pstBgImg	Pointer to the background gray-scale image It cannot be null. The height and width are the same as those of **pstFgFlag**.	Output
pstChgStaImg	Pointer to the gray-scale image with changed status It can be null when **pstUpdateBgModelCtrl→u8DetChgRegion** is **0**. The height and width are the same as those of **pstFgFlag**.	Output
pstChgStaFg	Pointer to the foreground image with changed status It can be null when **pstUpdateBgModelCtrl→u8DetChgRegion** is **0**. The height and width are the same as those of **pstFgFlag**.	Output
pstChgStaLife	Pointer to the life image of the pixels with changed status It can be null when **pstUpdateBgModelCtrl→u8DetChgRegion** is **0**. The height and width are the same as those of **pstFgFlag**.	Output
pstStatData	Pointer to background status data It cannot be null. The minimum memory size is **sizeof(**IVE_BG_STAT_DATA_S**)**.	Output
pstUpdateBgModelCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
pstBgModel	-	16 bytes	-
pstFgFlag	U8C1	16 bytes	See the **Chip Difference** field.
pstBgImg	U8C1	16 bytes	See the **Chip Difference** field.
pstChgStaImg	U8C1	16 bytes	See the **Chip Difference** field.
pstChgStaFg	S8C1	16 bytes	See the **Chip Difference** field.
pstChgStaLife	U16C1	16 bytes	See the **Chip Difference** field.

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported **pstFgFlag** resolution: 720 x 576
Hi3536 V100	Supported **pstFgFlag** resolution: 720 x 576
Hi3521A V100	Supported **pstFgFlag** resolution: 720 x 576
Hi3518E V200	Not supported
Hi3531A V100	Supported **pstFgFlag** resolution: 720 x 576
Hi3519 V100	Supported **pstFgFlag** resolution: 1280 x 720
Hi3519 V101	Supported **pstFgFlag** resolution: 1280 x 720

Chip	Difference
Hi3516C V300	Supported **pstFgFlag** resolution: 1280 x 720
Hi3559 V100	Supported **pstFgFlag** resolution: 1280 x 720

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The strides of **pstFgFlag**, **pstBgImg**, **pstChgStaImg** (if any), and **pstChgStaFg** (if any) must be the same.
- For details about background model data **pstModel**, see HI_MPI_IVE_MatchBgModel.
- **pstChgStaFg** indicates the foreground image with changed status. If the pixel is non-zero, the image is the foreground; if the pixel is 0, the image is the background.
- **pstChgStaLife** indicates life image of the foreground pixels with changed status. The pixel value indicates the duration of the foreground change.
- The status change indicates that the pixel becomes the foreground after the pixel value changes and the changed pixel value is retained for a long period of time. This is caused by a static residue or moving object.

[Example]

None

[See Also]

- HI_MPI_IVE_MatchBgModel
- HI_MPI_IVE_GradFg
- HI_MPI_IVE_GMM

HI_MPI_IVE_ANN_MLP_LoadModel

[Description]

Reads the ANN_MLP model file and initializes model data.

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
pchFileName	Model file path and file name It cannot be null.	Input

Parameter	Description	Input/Output
pstAnnMlpModel	Pointer to the model data structure It cannot be null.	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The file name extension must be .bin. The .bin file must be generated by using **ive_tool_xml2bin.exe** developed by HiSilicon.
- This MPI must work with HI_MPI_IVE_ANN_MLP_UnloadModel.

[Example]

None

[See Also]

- HI_MPI_IVE_ANN_MLP_UnloadModel

- HI_MPI_IVE_ANN_MLP_Predict

HI_MPI_IVE_ANN_MLP_UnloadModel

[Description]

Deinitializes ANN model data.

[Syntax]

```
HI_VOID HI_MPI_IVE_ANN_MLP_UnLoadModel(IVE_ANN_MLP_MODEL_S
*pstAnnMlpModel)
```

[Parameter]

Parameter	Description	Input/Output
pstAnnMlpModel	Pointer to the model data structure It cannot be null.	Input

[Return Value]

None

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

This MPI must work with HI_MPI_IVE_ANN_MLP_LoadModel.

[Example]

None

[See Also]

- HI_MPI_IVE_ANN_MLP_LoadModel
- HI_MPI_IVE_ANN_MLP_Predict

HI_MPI_IVE_ANN_MLP_Predict

[Description]

Creates an ANN_MLP prediction task for a single sample.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the input sample vector (character vector) It cannot be null.	Input
pstActivFuncTab	Pointer to lookup table information for activating function calculation It cannot be null.	Input
pstAnnMlpModel	Pointer to the model data structure It cannot be null.	Input
pstDst	Pointer to the prediction result vector It cannot be null.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Type	Address Alignment Mode	Number of Vector Dimensions
pstSrc	1D SQ16.16 vector that is truncated to SQ8.16	16 bytes	Value range: 1–256 Actual value: pstAnnMlpModel→au16LayerCount[0] Note that the minimum memory size is sizeof(SQ16.16) * (pstAnnMlpModel→au16LayerCount[0] + 1)
pstDst	1D SQ16.16 vector	16 bytes	Value range: 1–256 Actual value: pstAnnMlpModel→au16LayerCount[pstAnnMlpModel→u8LayerNum − 1]

NOTE

For details about fixed points such as SQ16.16 and SQ8.16, see "Fixed-Point Data Structure."

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Not supported
Hi3519 V101	Not supported
Hi3516C V300	Not supported
Hi3559 V100	Not supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.0.lib used on the PC for simulation)

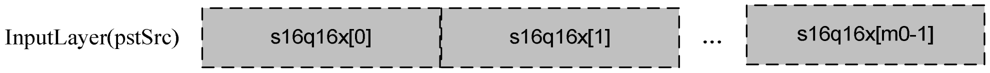
[Note]

- The principle is similar to that of ANN_MLP in OpenCV.
- The following are activation functions:

  Identity activation function: $f(u) = u$

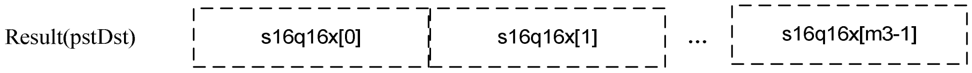
  Sigmoid symmetric activation function: $f(u) = \beta(\frac{2}{1+e^{-\alpha u}} - 1)$
- Gaussian activation function: $f(u) = \beta e^{-\alpha u * u}$
- The maximum number of dimensions for the input sample vector (input layer), the maximum number of dimensions for the output prediction result vector (output layer), or the maximum number of neurons at hidden layers is 256.
- At least one more dimension is allocated to **pstSrc** than to the input layer.
- **pstActivFuncTab** is the lookup table for the activation function calculation, the type of the lookup data is S1Q15, and a maximum of 4096 data segments are supported. Because the Identity, Sigmoid, and Gaussian functions supported by the ANN are odd or even functions, tables are created and searched only in [0, pstActivFuncTab→s32TabInUpper]. **u8TabOutNorm** indicates the number of shifts and is used for normalization during lookup table creation.
- The following example assumes that **u8LayerNum** is **4**, and **u8LayerCount[8]** is **{m0, m1, m2, m3, 0, 0, 0, 0}**:
  - The input sample vector (input layer) is the m0-dimensional vector of SQ16.16. Only SQ8.16 is supported, and the exceeded part is truncated.

Figure 2-21 Input sample vector of ANN_MLP



  - The output predication result vector is the m3-dimensional vector of SQ16.16.

Figure 2-22 Output prediction result of ANN_MLP



[Example]

None

[See Also]

- HI_MPI_IVE_ANN_MLP_LoadModel

- HI_MPI_IVE_ANN_MLP_UnloadModel

HI_MPI_IVE_ANN_MLP_Predict

[Description]

Creates ANN_MLP prediction tasks for multiple samples of the same model.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the input sample vector (eigenvector) array It cannot be null. The width is the number of sample vector dimensions multiplied by **sizeof(HI_S32)**. The height is the number of vectors. It cannot be null.	Input
pstActivFuncTab	Pointer to lookup table information for activating function calculation It cannot be null.	Input
pstAnnMlpModel	Pointer to the model data structure It cannot be null.	Input
pstDst	Pointer to the prediction result vector array The width is the number of types multiplied by **sizeof(HI_S32)**. The height is the number of vectors. It cannot be null.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Type	Address Alignment Mode	Number of Vector Dimensions
pstSrc	One-dimensional SQ16.16 or SQ18.14 vector array. Each element is truncated to SQ8.16 or SQ10.14 during calculation.	16 bytes	Value range: 1−1024 Actual value: **pstAnnMlpModel→au16LayerCount[0]**
pstDst	One-dimensional SQ16.16 or SQ18.14 vector array	16 bytes	Value range: 1–256 Actual value: **pstAnnMlpModel→au16LayerCount[pstAnnMlpModel→u8LayerNum − 1]**

For details about fixed points such as SQ16.16 and SQ8.16, see "Fixed-Point Data Structure."

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

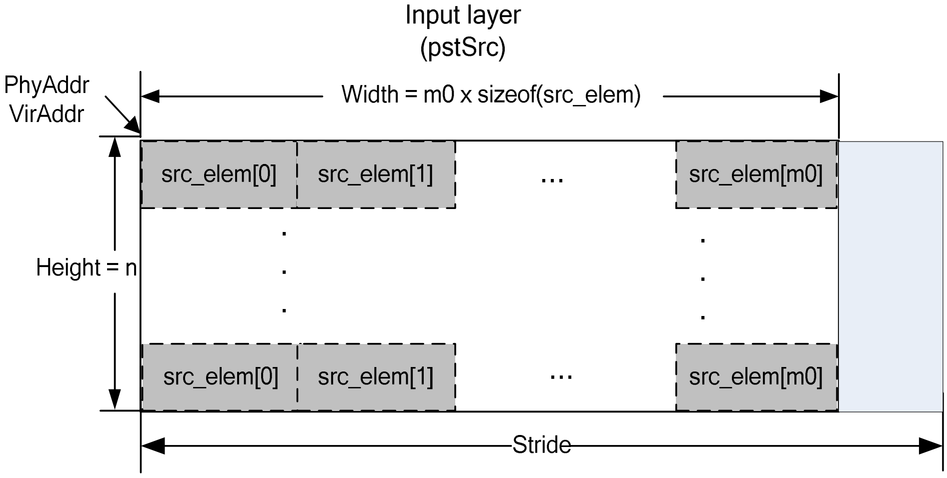
[Note]

- The principle is similar to that of ANN_MLP in OpenCV.
- The following are activation functions:

  Identity activation function: $f(u) = u$

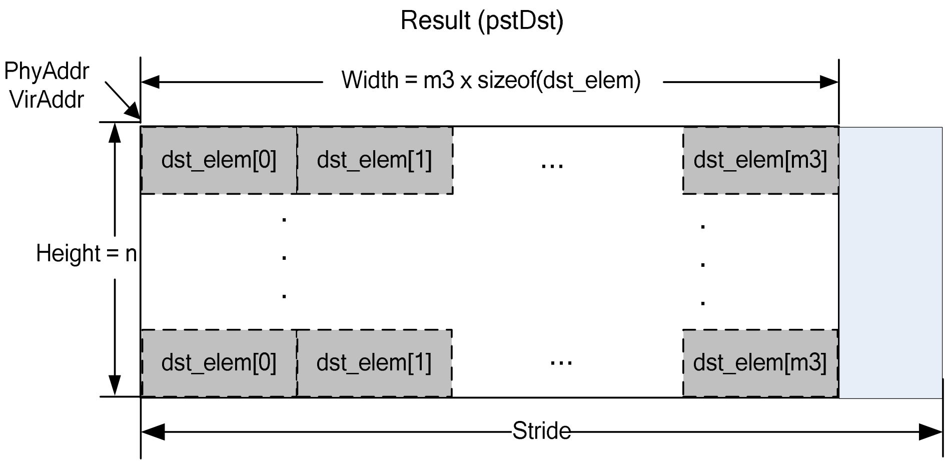
  Sigmoid symmetric activation function: $f(u) = \beta(\frac{2}{1+e^{-\alpha u}} - 1)$
- Gaussian activation function: $f(u) = \beta e^{-\alpha u * u}$
- The maximum number of dimensions for the input sample vector (at the input layer) is 1024, and the maximum number of dimensions for the output prediction result vector (at the output layer) and the maximum number of neurons at hidden layers are 256.
- Two data precision types are supported. For details, see IVE_ANN_MLP_ACCURATE_E.
- **pstActivFuncTab** is the lookup table for the calculation of the activation function $f(u)$, the type of the data is S1Q15, and a maximum of 4096 data segments are supported. Because the Identity, Sigmoid, and Gaussian activation functions supported by the ANN are odd or even functions, lookup tables are created and searched only when the input **u** falls within [0, pstActivFuncTab→s32TabInUpper]. **u8TabOutNorm** indicates the number of shifts and is used for normalization during lookup table creation.
- The following example assumes that **u8LayerNum** is **4**, **u8LayerCount[8]** is **{m0, m1, m2, m3, 0, 0, 0, 0}**, and the number of samples is *n*:
  - *n* sample vectors are input (at the input layer). Each vector contains **m0** elements with the **src_elem** type of SQ16.16 or SQ18.14. During actual calculation, each element is truncated to SQ8.16 or SQ10.14.

Figure 2-23 Input sample vector array of ANN_MLP



  - *n* prediction result vectors are output. Each vector contains **m3** elements with the **dst_elem** type of SQ16.16 or SQ18.14.

Figure 2-24 Output prediction result of ANN_MLP



[Example]

None

[See Also]

- HI_MPI_IVE_ANN_MLP_LoadModel
- HI_MPI_IVE_ANN_MLP_UnloadModel

HI_MPI_IVE_SVM_LoadModel

[Description]

Reads the SVM model file and initializes model data.

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
pchFileName	Model file path and file name It cannot be null.	Input
pstSvmModel	Pointer to the model data structure It cannot be null.	Output

[Return Value]

Return Value	Description
0	Success

Return Value	Description
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- The file name extension must be .bin. The .bin file must be generated by using **ive_tool_xml2bin.exe** developed by HiSilicon.
- This MPI must work with HI_MPI_IVE_SVM_UnloadModel.

[Example]

None

[See Also]

- HI_MPI_IVE_SVM_UnloadModel
- HI_MPI_IVE_SVM_Predict

HI_MPI_IVE_SVM_UnloadModel

[Description]

Deinitializes SVM model data.

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
pstSvmModel	Pointer to the model data structure It cannot be null.	Input

[Return Value]

None

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.0.lib used on the PC for simulation)

[Note]

This MPI must work with HI_MPI_IVE_SVM_LoadModel.

[Example]

None

[See Also]

- HI_MPI_IVE_SVM_LoadModel
- HI_MPI_IVE_SVM_Predict

HI_MPI_IVE_SVM_Predict

[Description]

Creates an SVM prediction task for a single sample.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the input sample vector (character vector) It cannot be null.	Input
pstKernelTab	Pointer to lookup table information for the kernel function calculation It cannot be null.	Input
pstSvmModel	Pointer to the model data structure It cannot be null.	Input
pstDstVote	Pointer to the vote vector of the 1-VS-1 SVM type It cannot be null.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Type	Address Alignment Mode	Number of Vector Dimensions
pstSrc	1D SQ16.16 vector that is truncated to SQ8.16	16 bytes	Value range: 1–256 Actual value: pstSvmModel→u16FeatureDim
pstDstVote	1D HI_U16 vector	16 bytes	Value range: 1−80 Actual value: pstSvmModel→u8ClassCount

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Supported
Hi3536 V100	Supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Supported
Hi3519 V100	Not supported
Hi3519 V101	Not supported
Hi3516C V300	Not supported
Hi3559 V100	Not supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.0.lib used on the PC for simulation)

[Note]

- The principle is similar to that of SVM_Predict in OpenCV.
- The following are kernel functions:

  Linear kernel function: $K(x_i,x_j)=x_i^T x_j$

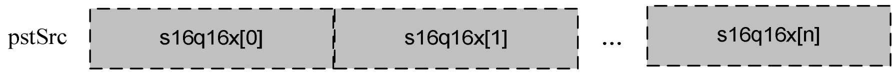
  Polynomial kernel function: $K(x_i,x_j)=(\gamma x_i^T x_j+coef0)^{\deg ree},\gamma>0$

  Radial basis kernel function: $K(x_i,x_j)=e^{-\gamma\|x_i-x_j\|^2},\gamma>0$
- Sigmoid kernel function: $K(x_i,x_j)=\tanh(\gamma x_i^T x_j+coef0)$
- The following is the decision function:

  $$\mathrm{sgn}(\omega^T\phi(x)+b)=\mathrm{sgn}(\sum_{i=1}^{l}(y_i\alpha_i K(x_i,x))+b)$$
- **pstKernelTab** is the lookup table for calculating the $K(x_i,x_j)$ kernel function, the type of the lookup data is S1Q15, and a maximum of 2048 elements are supported. When the SVM kernel function creates the lookup table, the input of the lookup table is $x_i^T x_j$ or $\|x_i-x_j\|^2$, and **u8TabOutNorm** can be the divisor (cannot be **0** when **SvmDivisor** = **u8TabOutNorm**) or the number of shifts (can be **0** when **SvmDivisor** = 1 << **u8TabOutNorm**). If **ive_tool_xml2bin.exe** is used, use **SvmDivisor** as the input parameter. For details about **SvmDivisor**, see the description of **ive_tool_xml2bin.exe**.
- The following example assumes that **u16FeatureDim** is *n*, and **u8ClassCount** is *N*:

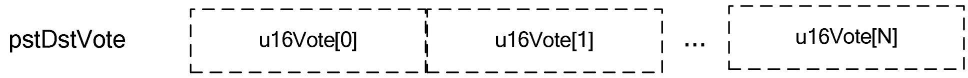
- The input sample vector is the *n*-dimensional vector (*n* is 256 at most) of SQ16.16. Only SQ8.16 is supported, and the exceeded part is truncated.

Figure 2-25 Input sample vector of SVM



- The output predication result vector is the *N*-dimensional vector of the HI_U16 type.

Figure 2-26 Prediction result of SVM



[Example]

None

[See Also]

- HI_MPI_IVE_SVM_LoadModel
- HI_MPI_IVE_SVM_UnloadModel

HI_MPI_IVE_SVM_Predict

[Description]

Creates SVM prediction tasks for multiple samples of the same model.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
pstSrc	Pointer to the input sample vector (eigenvector) array The width is the number of sample vector dimensions multiplied by **sizeof(HI_S16Q16)**. The height is the number of vectors. It cannot be null.	Input

Parameter	Description	Input/Output
pstKernelTab	Pointer to lookup table information for the kernel function calculation It cannot be null.	Input
pstSvmModel	Pointer to the model data structure It cannot be null.	Input
pstDstVote	Pointer to the vote number vector array of types obtained after the classification by the 1-VS-1 SVM The width is the number of vote types multiplied by **sizeof(HI_U16)**. The height is the number of vectors. It cannot be null.	Output
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Type	Address Alignment Mode	Number of Vector Dimensions
pstSrc	One-dimensional SQ16.16 vector array. Each element is truncated to SQ8.16 for calculation.	16 bytes	Value range: 1–1024 Actual value: **pstSvmModel→u16FeatureDim**
pstDstVote	One-dimensional HI_U16 vector array	16 bytes	Value range: 1–80 Actual value: **pstSvmModel→u8ClassCount**

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported

Chip	Difference
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Supported. Hi3519 V100 supports at most 2048 elements in the lookup table.
Hi3519 V101	Supported. Hi3519 V101 supports at most 4096 elements in the lookup table.
Hi3516C V300	Not supported
Hi3559 V100	Supported. Hi3559 V100 supports at most 4096 elements in the lookup table.

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

- The principle is similar to that of SVM_Predict in OpenCV.
- The following are kernel functions:

  Linear kernel function: $K(x_i,x_j)=x_i^T x_j$

  Polynomial kernel function: $K(x_i,x_j)=(\gamma x_i^T x_j+coef0)^{degree},\gamma>0$

  Radial basis kernel function: $K(x_i,x_j)=e^{-\gamma\|x_i-x_j\|^2},\gamma>0$

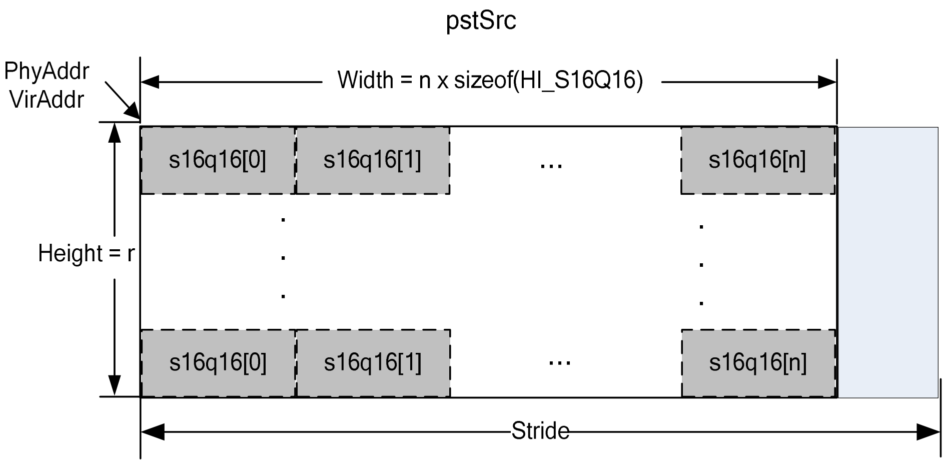
  Sigmoid kernel function: $K(x_i,x_j)=\tanh(\gamma x_i^T x_j+coef0)$

- The following is the decision function:

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

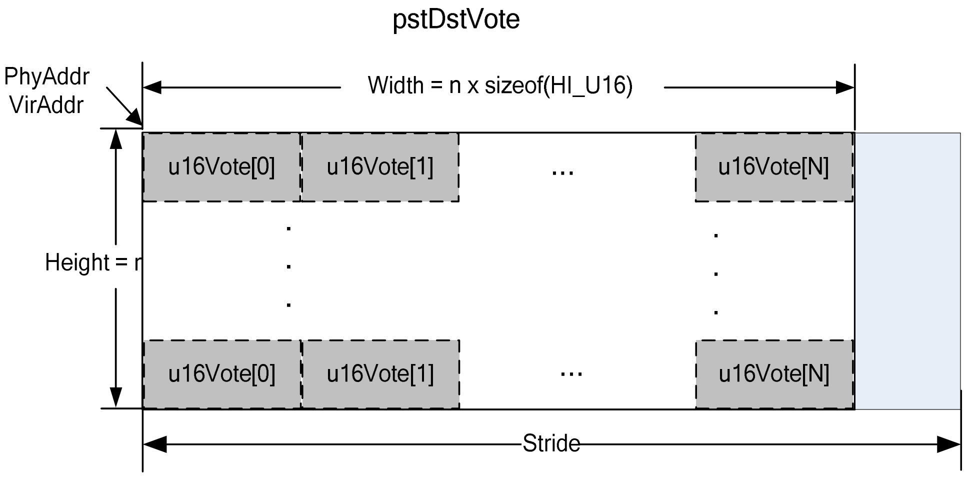
- **pstKernelTab** is the lookup table for calculating the *K(x_i,x_j)* kernel function, and the type of the data is S1Q15. For details about the number of elements in the lookup table, see the **Chip Difference** field. When the SVM kernel function creates the lookup table, the input of the lookup table is $x_i^T x_j$ or $\|x_i-x_j\|^2$, and **u8TabOutNorm** can be the divisor (cannot be 0 when **SvmDivisor** = **u8TabOutNorm**) or the number of shifts (can be 0 when **SvmDivisor** = 1 << **u8TabOutNorm**). If **ive_tool_xml2bin.exe** is used, **SvmDivisor** needs to serve as the input parameter. For details about **SvmDivisor**, see the description of **ive_tool_xml2bin.exe**.
- The following example assumes that **u16FeatureDim** is n, **u8ClassCount** is N, and the number of samples is r:
  - Each of the *r* input sample vectors is an *n*-dimensional vector (*n* is 1024 at most) of the SQ16.16 type. SQ8.16 is supported actually, and the exceeded part is truncated.

Figure 2-27 Input sample vector array of SVM



- Each of the *r* output predication result vectors is an *N*-dimensional vector of the HI_U16 type.

Figure 2-28 Prediction result of SVM



[Example]

None

[See Also]

- HI_MPI_IVE_SVM_LoadModel
- HI_MPI_IVE_SVM_UnloadModel

HI_MPI_IVE_CNN_LoadModel

[Description]

Reads the CNN model file and initializes the CNN model data.

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
pchFileName	Model file path and file name It cannot be null.	Input
pstCnnModel	Pointer to the CNN model structure It cannot be null.	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516AV100	Not supported
Hi3536V100	Not supported
Hi3521AV100	Not supported
Hi3518EV200	Not supported
Hi3531AV100	Not supported
Hi3519V100	Supported
Hi3519V101	Supported
Hi3516CV300	Not supported
Hi3559V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

- The file name extension must be .bin. The .bin file must be generated by using **ive_tool_caffe** (for details, see the *HiIVE Tool User Guide*) developed by HiSilicon.
- This MPI must work with HI_MPI_IVE_CNN_UnloadModel.

[Example]

None

[See Also]

- HI_MPI_IVE_CNN_UnloadModel
- HI_MPI_IVE_CNN_Predict
- HI_MPI_IVE_CNN_GetResult

HI_MPI_IVE_CNN_UnloadModel

[Description]

Deinitializes the CNN model data.

[Syntax]

```
HI_VOID HI_MPI_IVE_CNN_UnloadModel(IVE_CNN_MODEL_S *pstCnnModel);
```

[Parameter]

Parameter	Description	Input/Output
pstCnnModel	Pointer to the CNN model structure It cannot be null	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Supported

Chip	Difference
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

This MPI must work with HI_MPI_IVE_CNN_LoadModel.

[Example]

None

[See Also]

- HI_MPI_IVE_CNN_LoadModel
- HI_MPI_IVE_CNN_Predict
- HI_MPI_IVE_CNN_GetResult

HI_MPI_IVE_CNN_Predict

[Description]

Creates one or multiple sample prediction tasks of a CNN model and outputs the eigenvector.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle It cannot be null.	Output
astSrc[]	Input sample picture array. At most 64 sample pictures are supported. It cannot be null.	Input
pstCnnModel	Pointer to the CNN model structure It cannot be null.	Input

Parameter	Description	Input/Output
pstDst	Pointer to the eigenvector array that stores the result of the last CNN full connection layer It cannot be null.	Output
pstCnnCtrl	Pointer to the control parameter For details about the memory allocation of **pstCnnCtrl→stMem**, see the **Note** field. It cannot be null.	Input
bInstant	Flag indicating whether results need to be returned instantly	Input

Parameter	Supported Image Type	Address Alignment Mode	Resolution
astSrc[]	U8C1 and U8C3_PLANAR	16 bytes	Width (w): 16–80 Height (h): 16–(1280/w)

Parameter	Number of Vectors	Address Alignment Mode	Vector Description
pstDst	Value range: 1–64 Actual value: **pstDst→u16Height**	16 bytes	Value range for the number of dimensions: 1–256 Actual value of the number of dimensions: **pstCnnModel→stFullConnect.au16LayerCnt[pstCnnModel→stFullConnect.u8LayerNum − 1]** Element type: SQ18.14

For details about fixed points such as SQ18.14, see Fixed-Point Data Structure.

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported
Hi3531A V100	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

- If data is pre-processed during the training (for example, the average value is subtracted), the same pre-processing is required for the input data before this MPI is called (the average value needs to be subtracted before this MPI is called).
- The training requires that the normalization from [0, 255] to [0, 1] be implemented on the data. That is, there must be transform_param{scale: 0.00390625} in the training .prototxt for the raw data. Normalization is not required when this interface is called to perform prediction because of automatic processing in the hardware.
- The minimum size of the memory allocated to **pstCnnCtrl→stMem** is calculated as follows:

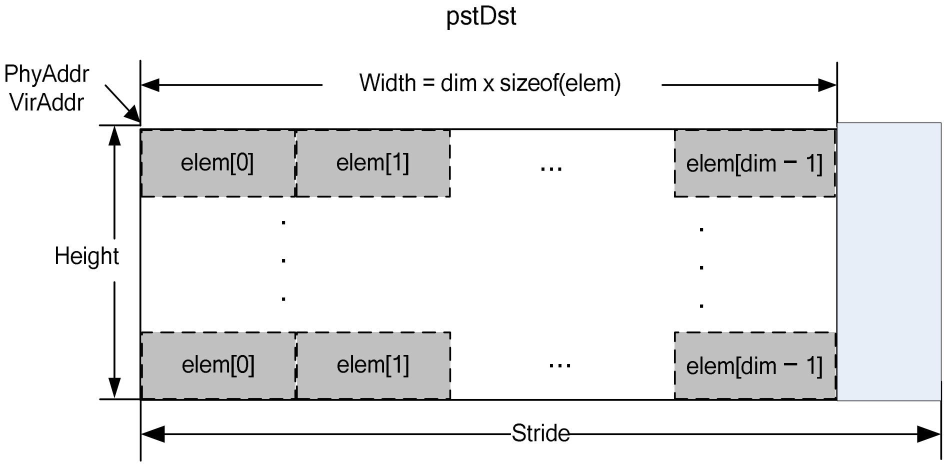
  IveAlign(m x pstCnnCtrl→u32Num x sizeof(HI_U32), 16) + IveAlign(pstCnnModel→stFullConnect.au16LayerCnt[0] x sizeof(HI_U32), 16) x pstCnnCtrl→u32Num

  where

  m is 1 when **pstCnnModel→enType** is **U8C1**, and *m* is 3 when **pstCnnModel→enType** is **U8C3_PLANAR**.
- The image type, width, and height of the sample array astSrc[] must be the same as **enType**, **u16Width**, and **u16Height** of **pstCnnModel** in the CNN model respectively. The number of array elements is **pstCnnCtrl→u32Num**.
- The number of vectors (**pstDst→u16Height**) in the eigenvector array (**pstDst**) is the same as the number of images (**pstCnnCtrl→u32Num**).

  Figure 2-29 shows the memory allocation for the output eigenvectors. The number of dimensions (**dim**) for each vector is calculated as follows: dim = pstCnnModel→stFullConnect.au16LayerCnt[pstCnnModel→stFullConnect.u8LayerNum − 1]. The type of the vector element is SQ18.14, and the number of vectors (**Height**) is **pstCnnCtrl→u32Num**.

Figure 2-29 Output eigenvector array of the CNN



- This MPI works with HI_MPI_IVE_CNN_GetResult. The eigenvector array **pstDst** is the input of HI_MPI_IVE_CNN_GetResult.
- The CNN model supports at most eight Conv-ReLU-Pooling layers and eight full-connection layers. The convolution kernel of the Conv-ReLU-Pooling layer supports only the 3 x 3 size. The rectified linear unit (ReLU) and pooling are configurable (see IVE_CNN_ACTIV_FUNC_E and IVE_CNN_POOLING_E). Each Conv-ReLU-Pooling outputs at most 50 feature maps. The full-connection layers 3–8 support only the ReLU activation function. The number of dimensions of the full-connection input layer (final output of Conv-ReLU-Pooling) ranges from 1 to 1024, the number of neurons at the middle hidden layer ranges from 2 to 256, and the number of dimensions of the output layer ranges from 1 to 256. For details about the parameter configuration, see Table 2-1 and Table 2-2.

Table 2-1 Parameter configuration for the single-layer Conv-ReLU-Pooling operation package in the CNN model

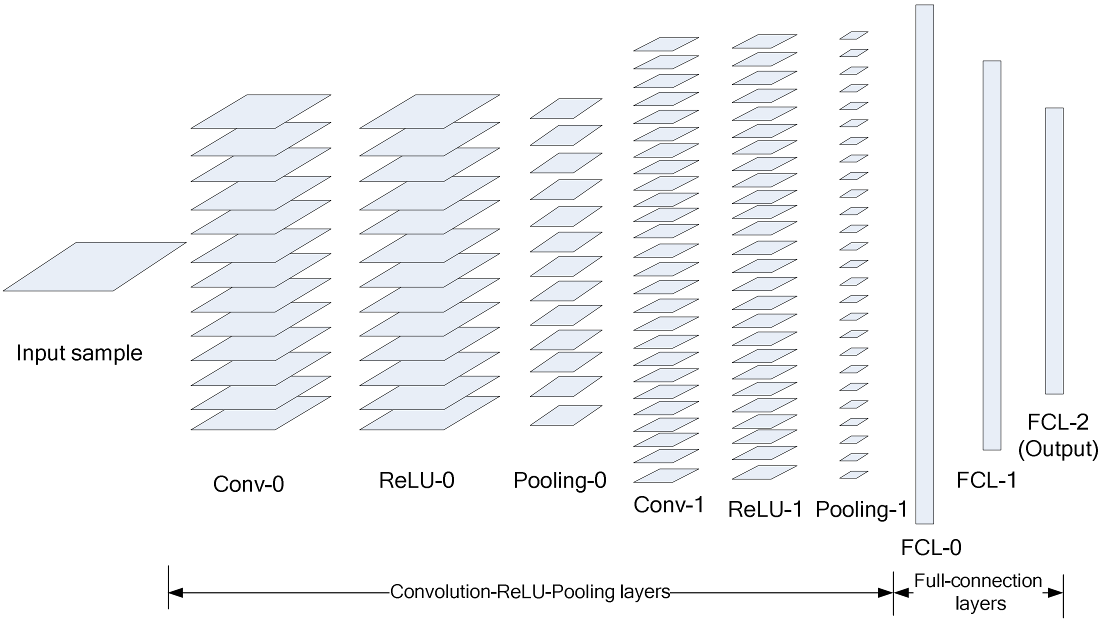
Conv-ReLU-Pooling Operation Package	Mode	Number	Size	Stride	Boundary Padding
Convolution	-	1–50	3 x 3	1	-
Activation	None\ReLU	-	-	-	-
Pooling	None\Max\Average	-	2 x 2	2	Round the value up to an even number and align it. Duplicate the boundary.

Table 2-2 Parameter configuration for the full-connection layer operation package in the CNN model

Number of Layers (Including the Input Layer)	Number of Input Layer Dimensions	Number of Nodes at the Middle Hidden Layer	Number of Output Layer Dimensions	Activation Function at the Hidden Layer
3–8	1–1024	2–256	1–256	ReLU

- Figure 2-30 shows the CNN model by taking a single input sample, $(n + 1)$ $(1 \leq n + 1 \leq 8)$ Conv-ReLU-Pooling layers, and $(m + 1)$ $(3 \leq n + 1 \leq 8)$ full-connection layers as an example. Note that FCL-0 indicates the column vector corresponding to the image data of Pooling-*n*. During actual calculation, the result of Pooling-*n* is directly output in the format of FCL-0.

Figure 2-30 CNN model



[Example]

None

[See Also]

- HI_MPI_IVE_CNN_LoadModel
- HI_MPI_IVE_CNN_UnloadModel
- HI_MPI_IVE_CNN_GetResult

HI_MPI_IVE_CNN_GetResult

[Description]

Receives the CNN prediction result, executes the Softmax operation to predict the type of each sample picture, and output the classification (Rank-1) with the highest confidence as well as the corresponding confidence.

[Syntax]

```
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);
```

[Parameter]

Parameter	Description	Input/Output
pstSrc	Pointer to the source data. The source data is the output of HI_MPI_IVE_CNN_Predict. It cannot be null.	Input
pstDst	Pointer to the prediction result structure, pointing to the array of IVE_CNN_RESULT_S and indicating the type and confidence of each sample It cannot be null.	Output
pstCnnModel	Pointer to the CNN model structure It cannot be null.	Input
pstCnnCtrl	Pointer to the control parameter It cannot be null.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see chapter 4 "Error Codes."

[Chip Difference]

Chip	Difference
Hi3516A V100	Not supported
Hi3536 V100	Not supported
Hi3521A V100	Not supported
Hi3518E V200	Not supported

Chip	Difference
Hi3531A V100	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported
Hi3559 V100	Supported

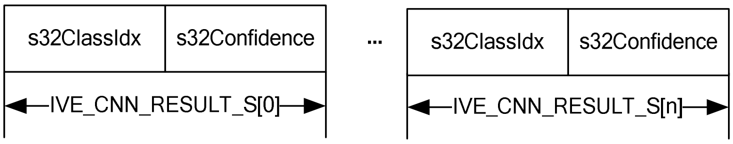
[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.1.lib used on the PC for simulation)

[Note]

- The source data **pstSrc** must be the output of HI_MPI_IVE_CNN_Predict. **pstCnnModel** and **pstCnnCtrl** must be the same as the parameters when HI_MPI_IVE_CNN_Predict is called.
- The prediction result **pstDst** points to the array of IVE_CNN_RESULT_S, and the number of array elements (**n**) is **pstCnnCtrl→u32Num**. Figure 2-31 shows the memory configuration.

Figure 2-31 Prediction results of CNN samples



- This MPI implements the Rank-1 version. The user can implement the Rank-n version (the most possible n classifications) as required by performing the following steps:
  - Calculate the confidence of each classification by using the Softmax operation.
  - Sort the confidence values.
  - Output the result of Rank-n.

[Example]

None

[See Also]

- HI_MPI_IVE_CNN_LoadModel
- HI_MPI_IVE_CNN_UnloadModel
- HI_MPI_IVE_CNN_Predict

HI_MPI_IVE_Query

[Description]

Queries the completion status of an existing task.

[Syntax]

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

[Parameter]

Parameter	Description	Input/Output
IveHandle	Task handle	Input
pbFinish	Pointer to the task completion status It cannot be null.	Output
bBlock	Flag indicating whether a task is blocked	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see chapter 4 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, hi_ive.h, mpi_ive.h
- Library file: libive.a (ive_clib2.x.lib used on the PC for simulation)

[Note]

- Before using the result of an IVE task, call this API in block mode to check whether the IVE task is complete.
- Because the IVE performs tasks based on the task creation sequence, you do not need to call this API to query the completion status of each task. For example, if tasks A and B are created in sequence and task B is complete, task A must be also complete. In this case, you can use the result of task A without querying the completion status of task A.
- If the return value is **HI_ERR_IVE_QUERY_TIMEOUT** (indicating that querying times out), you can continue to query the status.
- If the return value is **HI_ERR_IVE_SYS_TIMEOUT** (indicating that the system times out), all IVE tasks must be submitted again.

[Example]

```
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.u16Stride        = 352;
stSrc.u16Height        = 288;
stSrc.u16Width         = 352;

stDst.u32PhyAddr       = 0;
stDst.pu8VirAddr       = HI_NULL;
stDst.u16Stride        = 352;
stDst.u16Height        = 288;
stDst.u16Width         = 352;

bInstant               = HI_TRUE;

s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&stSrc.u32PhyAddr,
&stSrc.pu8VirAddr, "User", HI_NULL, stSrc.u16Height*stSrc.u16Stride);
if(HI_SUCCESS!=s32Ret)
{
   return s32Ret;
}
memset(stSrc.pu8VirAddr, 1, stSrc.u16Height * stSrc.u16Stride);

s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&stDst.u32PhyAddr, &stDst.pu8VirAddr,
"User", HI_NULL, stDst.u16Height*stDst.u16Stride);
if(HI_SUCCESS!=s32Ret)
{
   HI_MPI_SYS_MmzFree(stSrc.u32PhyAddr, stSrc.pu8VirAddr);
   return s32Ret;
}
memset(stDst.pu8VirAddr,0, stDst.u16Height * stDst.u16Stride);
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;
```

[See Also]

None

3 Data Structures

The IVE provides the following data structures:

- IVE_HIST_NUM: Defines the number of bins in a histogram.
- IVE_MAP_NUM: Defines the number of map lookup entries.
- IVE_MAX_REGION_NUM: Defines the maximum number of connected components.
- IVE_ST_MAX_CORNER_NUM: Defines the maximum number of Shi-Tomas-like corner points.
- IVE_IMAGE_TYPE_E: Defines the type of supported generalized 2D images.
- IVE_IMAGE_S: Defines the information about generalized 2D images.
- IVE_SRC_IMAGE_S: Defines the source image.
- IVE_DST_IMAGE_S: Defines the output image.
- IVE_DATA_S: Defines the information about 2D images in the unit of byte.
- IVE_SRC_DATA_S: Defines the information about 2D source data in the unit of byte.
- IVE_DST_DATA_S: Defines the information about 2D output data in the unit of byte.
- IVE_MEM_INFO_S: Defines the information about the memory for storing 1D data.
- IVE_SRC_MEM_INFO_S: Defines 1D source data.
- IVE_DST_MEM_INFO_S: Defines 1D output data.
- IVE_8BIT_U: Defines an 8-bit data union.
- IVE_POINT_U16_S: Defines U16 point information.
- IVE_POINT_S25Q7_S: Defines the points expressed by S25Q7 fixed points.
- IVE_RECT_S: Defines U16 rectangle information.
- IVE_DMA_MODE_E: Defines the DMA operation mode.
- IVE_DMA_CTRL_S: Defines DMA control information.
- IVE_FILTER_CTRL_S: Defines the control information about template filter.
- IVE_CSC_MODE_E: Defines the CSC mode.
- IVE_CSC_CTRL_S: Defines CSC control information.
- IVE_FILTER_AND_CSC_CTRL_S: Defines the control information about template filter and CSC.
- IVE_SOBEL_OUT_CTRL_E: Define Sobel output control information.
- IVE_SOBEL_CTRL_S: Defines the control information about Sobel edge extraction.

- IVE_MAG_AND_ANG_OUT_CTRL_E: Defines the output format of the calculated Canny edge magnitude and angle.
- IVE_MAG_AND_ANG_CTRL_S: Defines the control information about the Canny edge magnitude and argument calculation.
- IVE_DILATE_CTRL_S: Defines dilate control information.
- IVE_ERODE_CTRL_S: Defines erode control information.
- IVE_THRESH_MODE_E: Defines the thresh output format.
- IVE_THRESH_CTRL_S: Defines thresh control information.
- IVE_SUB_MODE_E: Defines the output format after the subtraction operation between two images.
- IVE_SUB_CTRL_S: Defines the control information about the subtraction operation between two images.
- IVE_INTEG_OUT_CTRL_E: Defines integrogram output control parameters.
- IVE_INTEG_CTRL_S: Defines the integrogram calculation control parameter.
- IVE_THRESH_S16_MODE_E: Defines the threshold mode of 16-bit signed images.
- IVE_THRESH_S16_CTRL_S: Defines the threshold control parameters of 16-bit signed images.
- IVE_THRESH_U16_MODE_E: Defines the threshold mode of 16-bit unsigned images.
- IVE_THRESH_U16_CTRL_S: Defines the threshold control parameters of 16-bit unsigned images.
- IVE_16BIT_TO_8BIT_MODE_E: Defines the conversion mode from a 16-bit image to an 8-bit image.
- IVE_16BIT_TO_8BIT_CTRL_S: Defines the control parameters for converting a 16-bit image to an 8-bit image.
- IVE_ORD_STAT_FILTER_MODE_E: Defines the order statistics filter mode.
- IVE_ORD_STAT_FILTER_CTRL_S: Defines the control parameter for order statistics filter.
- IVE_MAP_LUT_MEM_S: Defines the information about the lookup table memory of the map operator.
- IVE_MAP_U8BIT_LUT_MEM_S: Defines the lookup table memory for U8C1→U8C1 mapping.
- IVE_MAP_U16BIT_LUT_MEM_S: Defines the lookup table memory for U8C1→U16C1 mapping.
- IVE_MAP_S16BIT_LUT_MEM_S: Defines the lookup table memory for U8C1→S16C1 mapping.
- IVE_MAP_MODE_E: Defines the mapping mode.
- IVE_MAP_CTRL_S: Defines the mapping control parameters.
- IVE_EQUALIZE_HIST_CTRL_MEM_S: Defines the histogram equalization auxiliary memory.
- IVE_EQUALIZE_HIST_CTRL_S: Defines the histogram equalization control parameter.
- IVE_ADD_CTRL_S: Defines weighted addition control parameters for two images.
- IVE_NCC_DST_MEM_S: Defines the information about the NCC output memory.
- IVE_REGION_S: Defines connected component information.
- IVE_CCBLOB_S: Defines CCL output information.
- IVE_CCL_MODE_E: Defines the connected component mode.

- IVE_CCL_CTRL_S: Defines CCL control parameters.
- IVE_GMM_CTRL_S: Defines the control parameters for GMM background modeling.
- IVE_GMM2_SNS_FACTOR_MODE_E: Defines the sensitivity coefficient mode.
- IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_E: Defines the update mode of the model duration parameter.
- IVE_GMM2_CTRL_S: Defines the control parameters for GMM2 background modeling.
- IVE_CANNY_STACK_SIZE_S: Defines the stack size of strong edge points in the first phase of Canny edge extraction.
- IVE_CANNY_HYS_EDGE_CTRL_S: Defines calculation task control parameters in the first phase of Canny edge extraction.
- IVE_LBP_CMP_MODE_E: Defines the comparison mode during LBP calculation.
- IVE_LBP_CTRL_S: Defines LBP texture calculation control parameters.
- IVE_NORM_GRAD_OUT_CTRL_E: Defines the output control enumeration type for the normalized gradient calculation.
- IVE_NORM_GRAD_CTRL_S: Defines control parameters for the normalized gradient calculation.
- IVE_MV_S9Q7_S: Defines the LK optical flow displacement.
- IVE_LK_OPTICAL_FLOW_CTRL_S: Defines control parameters for the LK optical flow calculation.
- IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_E: Defines the output mode for the pyramid LK optical flow calculation.
- IVE_LK_OPTICAL_FLOW_PYR_CTRL_S: Defines control parameters for the pyramid LK optical flow calculation.
- IVE_ST_MAX_EIG_S: Defines the maximum response value of the corner point during the Shi-Tomas-like corner point calculation.
- IVE_ST_CANDI_CORNER_CTRL_S: Defines the control parameters for calculating Shi-Tomas-like candidate corner points.
- IVE_ST_CORNER_INFO_S: Defines the output corner point information after the Shi-Tomas-like corner point calculation.
- IVE_ST_CORNER_CTRL_S: Defines the control parameters for filtering Shi-Tomas-like corner points.
- IVE_SAD_MODE_E: Defines the SAD calculation mode.
- IVE_SAD_OUT_CTRL_E: Defines SAD output control mode.
- IVE_SAD_CTRL_S: Defines SAD control parameters.
- IVE_RESIZE_MODE_E: Defines the resize mode.
- IVE_RESIZE2_CTRL_S: Defines resize2 control parameters.
- IVE_RESIZE_CTRL_S: Defines resize control parameters.
- IVE_GRAD_FG_MODE_E: Defines the gradient foreground calculation mode.
- IVE_GRAD_FG_CTRL_S: Defines gradient foreground calculation control parameters.
- IVE_CANDI_BG_PIX_S: Defines candidate background model data.
- IVE_WORK_BG_PIX_S: Defines working background model data.
- IVE_BG_LIFE_S: Defines background life data.
- IVE_BG_MODEL_PIX_S: Defines background model data.
- IVE_FG_STAT_DATA_S: Defines foreground status data.
- IVE_BG_STAT_DATA_S: Defines background status data.

- IVE_MATCH_BG_MODEL_CTRL_S: Defines background match control parameters.
- IVE_UPDATE_BG_MODEL_CTRL_S: Defines background update control parameters.
- IVE_LOOK_UP_TABLE_S: Defines the lookup table.
- IVE_ANN_MLP_ACCURATE_E: Defines the type of the ANN_MLP input eigenvector.
- IVE_ANN_MLP_ACTIV_FUNC_E: Defines the enumeration type of ANN_MLP activation functions.
- IVE_ANN_MLP_MODEL_S: Defines ANN_MLP model data.
- IVE_SVM_TYPE_E: Defines the SVM type.
- IVE_SVM_KERNEL_TYPE_E: Defines the SVM kernel function type.
- IVE_SVM_MODEL_S: Defines SVM model data.
- IVE_CNN_ACTIV_FUNC_E: Defines the enumeration types of CNN activation functions.
- IVE_CNN_POOLING_E: Defines the enumeration types of the CNN pooling operation.
- IVE_CNN_CONV_POOLING_S: Defines the parameters of the CNN single-layer Conv-ReLU-Pooling convolution operation package.
- IVE_CNN_FULL_CONNECT_S: Defines the CNN full-connection network parameters.
- IVE_CNN_MODEL_S: Defines the parameters of the CNN model.
- IVE_CNN_CTRL_S: Defines the control parameters for a CNN prediction task.
- IVE_CNN_RESULT_S: Defines the prediction result of a single CNN sample.
- IVE_MODULE_PARAMS_S: Defines IVE module parameters.

Fixed-Point Data Structure

[Description]

Defines the type of fixed points.

[Syntax]

```
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;   /*8-bit unsigned integer,
4bits decimal fraction, 4-bit flag bits*/
```

[Member]

Member	Description
HI_U0Q8	No bit expresses the integral part, and 8 bits express the decimal part. This type is called UQ0.8 in this document.
HI_U1Q7	The upper one bit expresses the integral part, and lower 7 bits express the decimal part. This type is called UQ1.7 in this document.
HI_U5Q3	The upper five bits express the integral part, and lower 3 bits express the decimal part. This type is called UQ5.3 in this document.
HI_U0Q16	No bit expresses the integral part, and 16 bits express the decimal part. This type is called UQ0.16 in this document.
HI_U4Q12	The upper-4-bit unsigned data expresses the integral part, and lower 12 bits express the decimal part. This type is called UQ4.12 in this document.
HI_U6Q10	The upper-6-bit unsigned data expresses the integral part, and lower 10 bits express the decimal part. This type is called UQ6.10 in this document.
HI_U8Q8	The upper-8-bit unsigned data expresses the integral part, and lower 8 bits express the decimal part. This type is called UQ8.8 in this document.
HI_U14Q2	The upper-14-bit unsigned data expresses the integral part, and lower 2 bits express the decimal part. This type is called UQ14.2 in this document.
HI_U12Q4	The upper-12-bit unsigned data expresses the integral part, and lower 4 bits express the decimal part. This type is called UQ12.4 in this document.
HI_S14Q2	The upper-14-bit signed data expresses the integral part, and lower 2 bits express the decimal part. This type is called SQ14.2 in this document.
HI_S9Q7	The upper-9-bit signed data expresses the integral part, and lower 7 bits express the decimal part. This type is called SQ9.7 in this document.
HI_U22Q10	The upper-22-bit unsigned data expresses the integral part, and lower 10 bits express the decimal part. This type is called UQ22.10 in this document.
HI_U25Q7	The upper-25-bit unsigned data expresses the integral part, and lower 7 bits express the decimal part. This type is called UQ25.7 in this document.
HI_S25Q7	The upper-25-bit signed data expresses the integral part, and lower 7 bits express the decimal part. This type is called SQ25.7 in this document.
HI_U8Q4F4	The upper-8-bit unsigned data expresses the integral part, and middle 4 bits express the decimal part, and the lower 4 bits express the flag. This type is called UQF8.4.4 in this document.

[Note]

For HI_U*x*Q*y*F*z*\HI_S*x*Q*y*:

- The variable *x* after U indicates that *x*-bit unsigned data expresses the integral part.
- The variable *x* after S indicates that *x*-bit signed data expresses the integral part.
- The variable *y* after Q indicates that *y*-bit data expresses the decimal part.
- The variable *z* after F indicates that *z*-bit data expresses the flag.

- The bits are upper bits to lower bits from left to right.

[See Also]

None

IVE_HIST_NUM

[Description]

Defines the number of bins in a histogram.

[Syntax]

```
#define IVE_HIST_NUM   256
```

[Member]

None

[Note]

None

[See Also]

None

IVE_MAP_NUM

[Description]

Defines the number of map lookup entries.

[Syntax]

```
#define IVE_MAP_NUM    256
```

[Member]

None

[Note]

None

[See Also]

None

IVE_MAX_REGION_NUM

[Description]

Defines the maximum number of connected components.

[Syntax]

```
#define IVE_MAX_REGION_NUM   254
```

[Member]

None

[Note]

None

[See Also]

None

IVE_ST_MAX_CORNER_NUM

[Description]

Defines the maximum number of Shi-Tomas-like corner points.

[Syntax]

```
#define IVE_ST_MAX_CORNER_NUM   200
```

[Member]

None

[Note]

None

[See Also]

None

IVE_IMAGE_TYPE_E

[Description]

Defines the type of supported generalized 2D images.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_IMAGE_TYPE_U8C1	Single-channel image of which each pixel is expressed by an 8-bit unsigned data segment. For details, see Figure 1-2.
IVE_IMAGE_TYPE_S8C1	Single-channel image of which each pixel is expressed by an 8-bit signed data segment. For details, see Figure 1-2.
IVE_IMAGE_TYPE_YUV420SP	YUV420 semi-planar image. For details, see Figure 1-3.
IVE_IMAGE_TYPE_YUV422SP	YUV422 semi-planar image. For details, see Figure 1-4.
IVE_IMAGE_TYPE_YUV420P	YUV420 planar image. For details, see Figure 1-5.
IVE_IMAGE_TYPE_YUV422P	YUV422 planar image. For details, see Figure 1-6.
IVE_IMAGE_TYPE_S8C2_PACKAGE	Dual-channel image (stored in package format) of which each pixel is expressed by two 8-bit signed data segments. For details, see Figure 1-7.
IVE_IMAGE_TYPE_S8C2_PLANAR	Dual-channel image (stored in planar format) of which each pixel is expressed by two 8-bit signed data segments. For details, see Figure 1-8.
IVE_IMAGE_TYPE_S16C1	Single-channel image of which each pixel is expressed by a 16-bit signed data segment. For details, see Figure 1-2.
IVE_IMAGE_TYPE_U16C1	Single-channel image of which each pixel is expressed by a 16-bit unsigned data segment. For details, see Figure 1-2.
IVE_IMAGE_TYPE_U8C3_PACKAGE	Three-channel image (stored in package format) of which each pixel is expressed by three 8-bit unsigned data segments. For details, see Figure 1-9.
IVE_IMAGE_TYPE_U8C3_PLANAR	Three-channel image (stored in planar format) of which each pixel is expressed by three 8-bit unsigned data segments. For details, see Figure 1-10.

Member	Description
IVE_IMAGE_TYPE_S32C1	Single-channel image of which each pixel is expressed by a 32-bit signed data segment. For details, see Figure 1-2.
IVE_IMAGE_TYPE_U32C1	Single-channel image of which each pixel is expressed by a 32-bit unsigned data segment. For details, see Figure 1-2.
IVE_IMAGE_TYPE_S64C1	Single-channel image of which each pixel is expressed by a 64-bit signed data segment. For details, see Figure 1-2.
IVE_IMAGE_TYPE_U64C1	Single-channel image of which each pixel is expressed by a 64-bit unsigned data segment. For details, see Figure 1-2.

[Note]

None

[See Also]

- IVE_IMAGE_S
- IVE_SRC_IMAGE_S
- IVE_DST_IMAGE_S

IVE_IMAGE_S

[Description]

Defines the information about generalized 2D images.

[Syntax]

```
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;
```

[Member]

Member	Description
enType	Generalized image type
u32PhyAddr[3]	Physical address array of a generalized image
pu8VirAddr[3]	Virtual address array of a generalized image
u16Stride[3]	Generalized image stride

Member	Description
u16Width	Generalized image width
u16Height	Generalized image height
u16Reserved	Reserved

[Note]

- The alignment requirements on the input and output image addresses vary according to operators.
- The unit of **u16Width**, **u16Height**, and **u16Stride** is pixel.
- For details about each type of image, see Figure 1-2 to Figure 1-10.

[See Also]

- IVE_IMAGE_TYPE_E
- IVE_SRC_IMAGE_S
- IVE_DST_IMAGE_S

IVE_SRC_IMAGE_S

[Description]

Defines the source image.

[Syntax]

```
typedef IVE_IMAGE_S IVE_SRC_IMAGE_S;
```

[Member]

None

[Note]

None

[See Also]

- IVE_IMAGE_S
- IVE_DST_IMAGE_S

IVE_DST_IMAGE_S

[Description]

Defines the output image.

[Syntax]

```
typedef IVE_IMAGE_S IVE_DST_IMAGE_S;
```

[Member]

None

[Note]

None

[See Also]

- IVE_IMAGE_S
- IVE_SRC_IMAGE_S

IVE_DATA_S

[Description]

Defines the information about 2D images in the unit of byte.

[Syntax]

```
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;
```

[Member]

Member	Description
u32PhyAddr	Image physical address
pu8VirAddr	Image virtual address
u16Stride	Image stride
u16Height	Image height
u16Width	Image width
u16Reserved	Reserved

[Note]

The image is 2D data in the unit of byte. The image and IVE_IMAGE_S image can be converted into each other.

[See Also]

None

IVE_SRC_DATA_S

[Description]

Defines the information about 2D source data in the unit of byte.

[Syntax]

```
typedef IVE_DATA_S IVE_SRC_DATA_S;
```

[Member]

None

[Note]

None

[See Also]

- IVE_IMAGE_S
- IVE_DST_DATA_S

IVE_DST_DATA_S

[Description]

Defines the information about 2D output data in the unit of byte.

[Syntax]

```
typedef IVE_DATA_S IVE_DST_DATA_S;
```

[Member]

None

[Note]

None

[See Also]

- IVE_IMAGE_S
- IVE_SRC_IMAGE_S

IVE_MEM_INFO_S

[Description]

Defines the information about the memory for storing 1D data.

[Syntax]

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

[Member]

Member	Description
u32PhyAddr	Physical address for 1D data
pu8VirAddr	Virtual address for 1D data
u32Size	Number of 1D data bytes

[Note]

None

[See Also]

- IVE_SRC_MEM_INFO_S
- IVE_DST_MEM_INFO_S

IVE_SRC_MEM_INFO_S

[Description]

Defines 1D source data.

[Syntax]

```
typedef IVE_MEM_INFO_S IVE_SRC_MEM_INFO_S;
```

[Member]

None

[Note]

None

[See Also]

- IVE_MEM_INFO_S
- IVE_DST_MEM_INFO_S

IVE_DST_MEM_INFO_S

[Description]

Defines 1D output data.

[Syntax]

```
typedef IVE_MEM_INFO_S IVE_DST_MEM_INFO_S;
```

[Member]

None

[Note]

None

[See Also]

- IVE_MEM_INFO_S
- IVE_SRC_MEM_INFO_S

IVE_8BIT_U

[Description]

Defines an 8-bit data union.

[Syntax]

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

[Member]

Member	Description
s8Val	Signed 8-bit value
u8Val	Unsigned 8-bit value

[Note]

None

[See Also]

None

IVE_POINT_U16_S

[Description]

Defines U16 point information.

[Syntax]

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

[Member]

Member	Description
u16X	Horizontal coordinate of a point
u16Y	Vertical coordinate of a point

[Note]

None

[See Also]

None

IVE_POINT_S25Q7_S

[Description]

Defines the points expressed by S25Q7 fixed points.

[Syntax]

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

[Member]

Member	Description
s25q7X	Horizontal coordinate of a point, expressed by SQ25.7
s25q7Y	Vertical coordinate of a point, expressed by SQ25.7

[Note]

None

[See Also]

None

IVE_RECT_S

[Description]

Defines U16 rectangle information.

[Syntax]

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

[Member]

Member	Description
u16X	Horizontal coordinate of the rectangle point nearest to the origin
u16Y	Vertical coordinate of the rectangle point nearest to the origin
u16Width	Rectangle width
u16Height	Rectangle height

[Note]

None

[See Also]

None

IVE_DMA_MODE_E

[Description]

Defines the DMA operation mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_DMA_MODE_DIRECT_COPY	Direct fast copy mode
IVE_DMA_MODE_INTERVAL_COPY	Indirect copy mode. For details, see the **Note** field ofHI_MPI_IVE_DMA.
IVE_DMA_MODE_SET_3BYTE	3-byte set mode. For details, see the **Note** field of HI_MPI_IVE_DMA.
IVE_DMA_MODE_SET_8BYTE	8-byte set mode. For details, see the **Note** field of HI_MPI_IVE_DMA.

[Note]

None

[See Also]

IVE_DMA_CTRL_S

IVE_DMA_CTRL_S

[Description]

Defines DMA control information.

[Syntax]

```
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;
```

[Member]

Member	Description
enMode	DMA mode
u64Val	Used for assigning a value to the memory, valid only in set mode. The lower three bytes are used to store data in 3-byte mode.
u8HorSegSize	Size of a segment of a horizontal row in the source image, valid only in indirect copy mode Value range: {2, 3, 4, 8, 16}
u8ElemSize	Valid only in indirect copy mode. The first **u8ElemSizebyte** bytes in each segment can be copied. Value range: [1, u8HorSegSize − 1]
u8VerSegRows	Valid only in indirect copy mode. Data in the first row of every **u8VerSegRows** rows is split into data segments with the size of **u8HorSegSize** each, and the first bytes with the size of **u8ElemSize** in each segment are copied. Value range: [1, min{65535/srcStride, srcHeight}]

[Note]

None

[See Also]

IVE_DMA_MODE_E

IVE_FILTER_CTRL_S

[Description]

Defines the control information about template filter.

[Syntax]

```
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;
```

[Member]

Member	Description
as8Mask[5]	5x5 template coefficient. When the peripheral coefficient is set to **0**, the member can be used for 3x3 template filter.
u8Norm	Normalization parameter Value range: [0, 13]

[Note]

You can configure different template coefficients to implement various filtering effects.

[See Also]

None

IVE_CSC_MODE_E

[Description]

Defines the CSC mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_CSC_MODE_VIDEO_BT601_YUV2RGB	BT601 YUV2RGB video conversion
IVE_CSC_MODE_VIDEO_BT709_YUV2RGB	BT709 YUV2RGB video conversion
IVE_CSC_MODE_PIC_BT601_YUV2RGB	BT601 YUV2RGB image conversion
IVE_CSC_MODE_PIC_BT709_YUV2RGB	BT709 YUV2RGB image conversion
IVE_CSC_MODE_PIC_BT601_YUV2HSV	BT601 YUV2HSV image conversion
IVE_CSC_MODE_PIC_BT709_YUV2HSV	BT709 YUV2HSV image conversion
IVE_CSC_MODE_PIC_BT601_YUV2LAB	BT601 YUV2LAB image conversion
IVE_CSC_MODE_PIC_BT709_YUV2LAB	BT709 YUV2LAB image conversion
IVE_CSC_MODE_VIDEO_BT601_RGB2YUV	BT601 RGB2YUV video conversion
IVE_CSC_MODE_VIDEO_BT709_RGB2YUV	BT709 RGB2YUV video conversion
IVE_CSC_MODE_PIC_BT601_RGB2YUV	BT601 RGB2YUV image conversion
IVE_CSC_MODE_PIC_BT709_RGB2YUV	BT709 RGB2YUV image conversion

[Note]

- In IVE_CSC_MODE_VIDEO_BT601_YUV2RGB and IVE_CSC_MODE_VIDEO_BT709_YUV2RGB modes, the output format must meet the following condition: 16 pixels ≤ R, G, B ≤ 235 pixels.

- In IVE_CSC_MODE_PIC_BT601_YUV2RGB and IVE_CSC_MODE_PIC_BT709_YUV2RGB modes, the output format must meet the following condition: 0 ≤ R, G, B ≤ 255 pixels.
- In IVE_CSC_MODE_PIC_BT601_YUV2HSV and IVE_CSC_MODE_PIC_BT709_YUV2HSV modes, the output format must meet the following condition: 0 ≤ H, S, V ≤ 255 pixels.
- In IVE_CSC_MODE_PIC_BT601_YUV2LAB and IVE_CSC_MODE_PIC_BT709_YUV2LAB modes, the output format must meet the following condition: 0 ≤ L, A, B ≤ 255 pixels.
- In IVE_CSC_MODE_VIDEO_BT601_RGB2YUV and IVE_CSC_MODE_VIDEO_BT709_RGB2YUV modes, the output format must meet the following condition: 0 ≤ Y, U, V ≤ 255 pixels.
- In IVE_CSC_MODE_PIC_BT601_RGB2YUV and IVE_CSC_MODE_PIC_BT709_RGB2YUV modes, the output format must meet the following condition: 0 ≤ Y ≤ 235 pixels, 0 ≤ U, V ≤ 240 pixels.

[See Also]

- IVE_CSC_CTRL_S
- IVE_FILTER_AND_CSC_CTRL_S

IVE_CSC_CTRL_S

[Description]

Defines CSC control information.

[Syntax]

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

[Member]

Member	Description
enMode	Working mode

[Note]

None

[See Also]

IVE_CSC_MODE_E

IVE_FILTER_AND_CSC_CTRL_S

[Description]

Defines the control information about template filter and CSC.

[Syntax]

```
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;
```

[Member]

Member	Description
enMode	Working mode
as8Mask[25]	5x5 template coefficient
u8Norm	Normalization parameter Value range: [0, 13]

[Note]

Only the four YUV2RGB modes are supported.

[See Also]

IVE_CSC_MODE_E

IVE_SOBEL_OUT_CTRL_E

[Description]

Define Sobel output control information.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_SOBEL_OUT_CTRL_BOTH	Output results after filtering by using the common template and transposed template
IVE_SOBEL_OUT_CTRL_HOR	Output results after filtering by using only the common template

Member	Description
IVE_SOBEL_OUT_CTRL_VER	Output results after filtering by using only the transposed template

[Note]

None

[See Also]

IVE_SOBEL_CTRL_S

IVE_SOBEL_CTRL_S

[Description]

Defines the control information about Sobel edge extraction.

[Syntax]

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

[Member]

Member	Description
enOutCtrl	Output control enumeration parameter
as8Mask[25]	5x5 template coefficient

[Note]

None

[See Also]

IVE_SOBEL_OUT_CTRL_E

IVE_MAG_AND_ANG_OUT_CTRL_E

[Description]

Defines the output format of the calculated Canny edge magnitude and angle.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_MAG_AND_ANG_OUT_CTRL_MAG	Output only magnitude
IVE_MAG_AND_ANG_OUT_CTRL_MAG_AND_ANG	Output both magnitude and angle

[Note]

None

[See Also]

IVE_MAG_AND_ANG_CTRL_S

IVE_MAG_AND_ANG_CTRL_S

[Description]

Defines the control information about the Canny edge magnitude and argument calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
enOutCtrl	Output format
u16Thr	Threshold for implementing thresh of the magnitude
as8Mask[25]	5x5 template coefficient

[Note]

None

[See Also]

IVE_MAG_AND_ANG_OUT_CTRL_E

IVE_DILATE_CTRL_S

[Description]

Defines dilate control information.

[Syntax]

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

[Member]

Member	Description
au8Mask[25]	5x5 template coefficient Value: 0 or 255

[Note]

None

[See Also]

None

IVE_ERODE_CTRL_S

[Description]

Defines erode control information.

[Syntax]

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

[Member]

Member	Description
au8Mask[25]	5x5 template coefficient Value: 0 or 255

[Note]

None

[See Also]

None

IVE_THRESH_MODE_E

[Description]

Defines the thresh output format.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_THRESH_MODE_BINARY	srcVal ≤ lowThr, dstVal = minVal srcVal > lowThr, dstVal = maxVal
IVE_THRESH_MODE_TRUNC	srcVal ≤ lowThr, dstVal = srcVal srcVal > lowThr, dstVal = maxVal

Member	Description
IVE_THRESH_MODE_TO_MINVAL	srcVal ≤ lowThr, dstVal = minVal srcVal > lowThr, dstVal = srcVal
IVE_THRESH_MODE_MIN_MID_MAX	srcVal ≤ lowThr, dstVal = minVal lowThr < srcVal ≤ highThr, dstVal = midVal srcVal > highThr, dstVal = maxVal
IVE_THRESH_MODE_ORI_MID_MAX	srcVal ≤ lowThr, dstVal = srcVal lowThr < srcVal ≤ highThr dstVal = midVal srcVal > highThr, dstVal = maxVal
IVE_THRESH_MODE_MIN_MID_ORI	srcVal ≤ lowThr, dstVal = minVal lowThr < srcVal ≤ highThr dstVal = midVal srcVal > highThr, dstVal = srcVal
IVE_THRESH_MODE_MIN_ORI_MAX	srcVal ≤ lowThr, dstVal = minVal lowThr < srcVal ≤ highThr dstVal = srcVal srcVal > highThr, dstVal = maxVal
IVE_THRESH_MODE_ORI_MID_ORI	srcVal ≤ lowThr, dstVal = srcVal lowThr < srcVal ≤ highThr dstVal = midVal srcVal > highThr, dstVal = srcVal

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_Thresh and Figure 2-8.

[See Also]

IVE_THRESH_CTRL_S

IVE_THRESH_CTRL_S

[Description]

Defines thresh control information.

[Syntax]

```
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;     /*Maximum value when tri-level thresholding*/
}IVE_THRESH_CTRL_S;
```

[Member]

Member	Description
enMode	DMA operation mode
u8LowThresh	Low threshold Value range: [0, 255]
u8HighThresh	High threshold 0 ≤ u8LowThresh ≤ u8HighThresh ≤ 255
u8MinVal	Minimum value Value range: [0, 255]
u8MidVal	Median value Value range: [0, 255]
u8MaxVal	Maximum value Value range: [0, 255]

[Note]

None[See Also]

IVE_THRESH_MODE_E

IVE_SUB_MODE_E

[Description]

Defines the output format after the subtraction operation between two images.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_SUB_MODE_ABS	Absolute value of the difference
IVE_SUB_MODE_SHIFT	The output result is obtained by shifting the result one digit right to reserve the signed bit.

[Note]

None

[See Also]

IVE_SUB_CTRL_S

IVE_SUB_CTRL_S

[Description]

Defines the control information about the subtraction operation between two images.

[Syntax]

```
typedef struct hiIVE_SUB_CTRL_S
{
    IVE_SUB_MODE_E enMode;
}IVE_SUB_CTRL_S;
```

[Member]

Member	Description
enMode	Subtraction mode of two images

[Note]

None

[See Also]

IVE_SUB_MODE_E

IVE_INTEG_OUT_CTRL_E

[Description]

Defines integrogram output control parameters.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_INTEG_OUT_CTRL_COMBINE	Combined output of the sum integrogram and square sum integrogram (see Figure 1-13)
IVE_INTEG_OUT_CTRL_SUM	Output of only the sum integrogram
IVE_INTEG_OUT_CTRL_SQSUM	Output of only the square sum integrogram

[Note]

None

[See Also]

IVE_INTEG_CTRL_S

IVE_INTEG_CTRL_S

[Description]

Defines the integrogram calculation control parameter.

[Syntax]

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

[Member]

Member	Description
enOutCtrl	Integrogram output control parameter

[Note]

None

[See Also]

IVE_INTEG_OUT_CTRL_E

IVE_THRESH_S16_MODE_E

[Description]

Defines the threshold mode of 16-bit signed images.

[Syntax]

```
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;
```

[Member]

Member	Description

IVE_THRESH_S16_MODE_S16_TO_S8_MIN_MID_MAX	srcVal ≤ lowThr
	dstVal = minVal
	lowThr < srcVal ≤ highThr,
	dstVal = midVal
	srcVal > highThr
	dstVal = maxVal

IVE_THRESH_S16_MODE_S16_TO_S8_MIN_ORI_MAX	srcVal ≤ lowThr
	dstVal = minVal
	lowThr < srcVal ≤ highThr
	dstVal = srcVal
	srcVal > highThr
	dstVal = maxVal

IVE_THRESH_S16_MODE_S16_TO_U8_MIN_MID_MAX	srcVal ≤ lowThr
	dstVal = minVal
	lowThr < srcVal ≤ highThr
	dstVal = midVal
	srcVal > highThr
	dstVal = maxVal

IVE_THRESH_S16_MODE_S16_TO_U8_MIN_ORI_MAX	srcVal ≤ lowThr
	dstVal = minVal
	lowThr < srcVal ≤ highThr
	dstVal = srcVal
	srcVal > highThr
	dstVal = maxVal

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_Thresh_S16 and Figure 2-9.

[See Also]

IVE_THRESH_S16_CTRL_S

IVE_THRESH_S16_CTRL_S

[Description]

Defines the threshold control parameters of 16-bit signed images.

[Syntax]

```
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;
```

[Member]

Member	Description
enMode	Thresh operation mode
s16LowThr	Low threshold
s16HighThr	High threshold
un8MinVal	Minimum value
un8MidVal	Median value
un8MaxVal	Maximum value

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_Thresh_S16 and Figure 2-9.

[See Also]

IVE_THRESH_S16_MODE_E

IVE_THRESH_U16_MODE_E

[Description]

Defines the threshold mode of 16-bit unsigned images.

[Syntax]

```
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;
```

[Member]

Member	Description
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	$srcVal \leq lowThr$ $dstVal = minVal$ $lowThr < srcVal \leq highThr$ $dstVal = srcVal$ $srcVal > highThr$ $dstVal = maxVal$

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_Thresh_U16 and Figure 2-10.

[See Also]

IVE_THRESH_U16_CTRL_S

IVE_THRESH_U16_CTRL_S

[Description]

Defines the threshold control parameters of 16-bit unsigned images.

[Syntax]

```
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;
```

[Member]

Member	Description
enMode	Thresh operation mode
u16LowThr	Low threshold
u16HighThr	High threshold
u8MinVal	Minimum value Value range: [0, 255]
u8MidVal	Median value Value range: [0, 255]
u8MaxVal	Maximum value Value range: [0, 255]

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_Thresh_U16 and Figure 2-10.

[See Also]

IVE_THRESH_U16_MODE_E

IVE_16BIT_TO_8BIT_MODE_E

[Description]

Defines the conversion mode from a 16-bit image to an 8-bit image.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_16BIT_TO_8BIT_MODE_S16_TO_S8	Linear transformation from S16 data to S8 data
IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_ABS	S8 data obtained after linear transformation from S16 data to S8 data and then absolute value operation
IVE_16BIT_TO_8BIT_MODE_S16_TO_U8_BIAS	U8 data obtained after linear transformation from S16 data to S8 data, translation, and then truncation
IVE_16BIT_TO_8BIT_MODE_U16_TO_U8	Linear transformation from S16 data to U8 data

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_16BitTo8Bit and Figure 2-11.

[See Also]

IVE_16BIT_TO_8BIT_CTRL_S

IVE_16BIT_TO_8BIT_CTRL_S

[Description]

Defines the control parameters for converting a 16-bit image to an 8-bit image.

[Syntax]

```
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;
```

[Member]

Member	Description
enMode	Conversion from 16-bit data to 8-bit data
u16Denominator	Denominator during linear transformation Value range: [max{1, u8Numerator}, 65535]
u8Numerator	Numerator during linear transformation Value range: [0, 255]

Member	Description
s8Bias	Translation item during linear transformation Value range: [−128, +127]

[Note]

- For details about the related formula, see the **Note** field of HI_MPI_IVE_Thresh_U16 and Figure 2-10.
- The following condition must be met:
- u8Numerator ≤ u16Denominator and u16Denominator ≠ 0

[See Also]

IVE_16BIT_TO_8BIT_MODE_E

IVE_ORD_STAT_FILTER_MODE_E

[Description]

Defines the order statistics filter mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_ORD_STAT_FILTER_MODE_MEDIAN	Median filtering
IVE_ORD_STAT_FILTER_MODE_MAX	Maximum filtering, equivalent to the dilate task for the gray-scale images
IVE_ORD_STAT_FILTER_MODE_MIN	Minimum filtering, equivalent to the erode task for the gray-scale images

[Note]

None

[See Also]

IVE_ORD_STAT_FILTER_CTRL_S

IVE_ORD_STAT_FILTER_CTRL_S

[Description]

Defines the the control parameter for order statistics filter.

[Syntax]

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

[Member]

Member	Description
enMode	Order statistics filter mode

[Note]

None

[See Also]

IVE_ORD_STAT_FILTER_MODE_E

IVE_MAP_LUT_MEM_S

[Description]

Defines the information about the lookup table memory of the map operator.

[Syntax]

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

[Member]

Member	Description
au8Map[IVE_MAP_NUM]	Map lookup table array

[Note]

None

[See Also]

None

IVE_MAP_U8BIT_LUT_MEM_S

[Description]

Defines the lookup table memory for U8C1→U8C1 mapping.

[Syntax]

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

[Member]

Member	Description
au8Map[IVE_MAP_NUM]	Mapping lookup table array

[Note]

None

[See Also]

None

IVE_MAP_U16BIT_LUT_MEM_S

[Description]

Defines the lookup table memory for U8C1→U16C1 mapping.

[Syntax]

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

[Member]

Member	Description
au16Map[IVE_MAP_NUM]	Mapping lookup table array

[Note]

None

[See Also]

None

IVE_MAP_S16BIT_LUT_MEM_S

[Description]

Defines the lookup table memory for U8C1→S16C1 mapping.

[Syntax]

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

[Member]

Member	Description
as16Map[IVE_MAP_NUM]	Mapping lookup table array

[Note]

None

[See Also]

None

IVE_MAP_MODE_E

[Description]

Defines the mapping mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_MAP_MODE_U8	U8C1→U8C1 mapping mode
IVE_MAP_MODE_S16	U8C1→U16C1 mapping mode
IVE_MAP_MODE_U16	U8C1→S16C1 mapping mode

[Note]

None

[See Also]

None

IVE_MAP_CTRL_S

[Description]

Defines the mapping control parameters.

[Syntax]

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

[Member]

Member	Description
enMode	Mapping mode

[Note]

None

[See Also]

None

IVE_EQUALIZE_HIST_CTRL_MEM_S

[Description]

Defines the histogram equalization auxiliary memory.

[Syntax]

```
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;
```

[Member]

Member	Description
au32Hist[IVE_HIST_NUM]	Histogram statistics output

Member	Description
au8Map[IVE_MAP_NUM]	Map lookup table calculated based on the histogram statistics

[Note]

None

[See Also]

IVE_EQUALIZE_HIST_CTRL_S

IVE_EQUALIZE_HIST_CTRL_S

[Description]

Defines the histogram equalization control parameter.

[Syntax]

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

[Member]

Member	Description
stMem	The memory with the size of **sizeof(**IVE_EQUALIZE_HIST_CTRL_MEM_S**)** bytes is required.

[Note]

None

[See Also]

IVE_EQUALIZE_HIST_CTRL_MEM_S

IVE_ADD_CTRL_S

[Description]

Defines weighted addition control parameters for two images.

[Syntax]

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

[Member]

Member	Description
u0q16X	Weight x in the weighted addition "xA + yB" Value range: [1, 65535]
u0q16Y	Weight y in the weighted addition "xA + yB" Value range: {65536 − u0q16X}

[Note]

None

[See Also]

None

IVE_NCC_DST_MEM_S

[Description]

Defines the information about the NCC output memory.

[Syntax]

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

[Member]

Member	Description
u64Numerator	Numerator in the NCC formula: $\sum_{i=1}^{w}\sum_{j=1}^{h}(I_{src1}(i,j)*I_{src2}(i,j))$
u64QuadSum1	Denominator in the NCC formula (part in the radical sign): $\sum_{i=1}^{w}\sum_{j=1}^{h}(I_{src1}^{2}(i,j))$
u64QuadSum2	Denominator in the NCC formula (part in the radical sign): $\sum_{i=1}^{w}\sum_{j=1}^{h}(I_{src2}^{2}(i,j))$

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_NCC.

[See Also]

None

IVE_REGION_S

[Description]

Defines connected component information.

[Syntax]

```
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;
```

[Member]

Member	Description
u32Area	Connected component area, in pixel
u16Left	Coordinate of the left border of the circumscribed rectangle of a connected component
u16Right	Coordinate of the right border of the circumscribed rectangle of a connected component
u16Top	Coordinate of the top border of the circumscribed rectangle of a connected component
u16Bottom	Coordinate of the bottom border of the circumscribed rectangle of a connected component

[Note]

None

[See Also]

IVE_CCBLOB_S

IVE_CCBLOB_S

[Description]

Defines CCL output information.

[Syntax]

```
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;
```

[Member]

Member	Description
u16CurAreaThr	Area threshold for valid connected components. The components in **astRegion** whose area is below the threshold are set to **0**.
s8LabelStatus	Connected component labeling status –1: failure 0: success
u8RegionNum	Number of valid connected components
astRegion[IVE_MAX_REGION_NUM]	Connected component information. The area of a valid connected component is greater than 0, and its ID is the array subscript plus 1.

[Note]

None

[See Also]

IVE_REGION_S

IVE_CCL_MODE_E

[Description]

Defines the connected component mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_CCL_MODE_4C	Four connected components
IVE_CCL_MODE_8C	Eight connected components

[Note]

None

[See Also]

None

IVE_CCL_CTRL_S

[Description]

Defines CCL control parameters.

[Syntax]

For the Hi3519:

```
typedef struct hiIVE_CCL_CTRL_S
{
    IVE_CCL_MODE_E enMode;    /*Mode*/
   HI_U16 u16InitAreaThr;    /*Init threshold of region area*/
   HI_U16 u16Step;           /*Increase area step for once*/
}IVE_CCL_CTRL_S;
```

For other chips:

```
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;
```

[Member]

Member	Description
enMode	Connected component mode
u16InitAreaThr	Initial area threshold Value range: [0, 65535] Reference value: 4
u16Step	Area threshold increase step Value range: [1, 65535] Reference value: 2

[Note]

None

[See Also]

IVE_CCBLOB_S

IVE_GMM_CTRL_S

[Description]

Defines the control parameters for GMM background modeling.

[Syntax]

```
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;
```

[Member]

Member	Description
u22q10NoiseVar	Initial noise variance Value range: [0x1, 0xFFFFFF] For the gray-scale GMM, the member corresponds to (**noiseSigma** x **noiseSigma**) in the gray-scale model in OpenCV MOG. Reference value: 15 x 15 x (1<<10) For the RGB GMM, the member corresponds to (3 x **noiseSigma** x **noiseSigma**) in the RGB model in OpenCV MOG. Reference value: 3 x 15 x 15 x (1<<10)
u22q10MaxVar	Maximum value of the model variance Value range: [0x1, 0xFFFFFF] The member corresponds to **fVarMax** in OpenCV MOG2. Reference value: 3 x 4000 << 10 (RGB), 2000 << 10 (gray scale)

Member	Description
u22q10MinVar	Minimum value of the model variance Value range: [0x1, u22q10MaxVar] The member corresponds to **fVarMin** in OpenCV MOG2. Reference value: 600 << 10 (RGB), 200 << 10 (gray scale)
u0q16LearnRate	Learning rate Value range: [1,65535] The member corresponds to **learningRate** in OpenCV MOG2. Reference value: if (frameNum < 500) (1/frameNum) x ((1 << 16) − 1); else ((1/500) x ((1<<16) − 1)
u0q16BgRatio	Background ratio threshold Value range: [1, 65535] The member corresponds to **backgroundRatio** in OpenCV MOG. Reference value: 0.8 x ((1 << 16)− 1)
u8q8VarThr	Variance threshold Value range: [1, 65535] The member corresponds to **varThreshold** in OpenCV MOG and is used to determine whether a pixel hits the current model. Reference value: 6.25 x (1 << 8)
u0q16InitWeight	Initial weight Value range: [1, 65535] The member corresponds to **defaultInitialWeight** in OpenCV MOG. Reference value: 0.05 x ((1 << 16) − 1)
u8ModelNum	Number of models Value range: {3, 5} The member corresponds to **nmixtures** in OpenCV MOG.

[Note]

None

[See Also]

None

IVE_GMM2_SNS_FACTOR_MODE_E

[Description]

Defines the sensitivity coefficient mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_GMM2_SNS_FACTOR_MODE_GLB	Global sensitivity coefficient mode. During model matching of each pixel, **u8GlbSnsFactor** of IVE_GMM2_CTRL_S is used as the variance sensitivity.
IVE_GMM2_SNS_FACTOR_MODE_PIX	Pixel-level sensitivity coefficient mode. During model matching of each pixel, the sensitivity coefficient of **pstFactor** is used as the variance sensitivity.

[Note]

None

[See Also]

None

IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_E

[Description]

Defines the update mode of the model duration parameter.

[Syntax]

```
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 ;
```

[Member]

Member	Description
IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_GLB	Global update mode of the model duration parameter. **u16GlbLifeUpdateFactor** of IVE_GMM2_CTRL_S is used when the model duration parameter of each pixel is updated.
IVE_GMM2_LIFE_UPDATE_FACTOR_MODE_PIX	Pixel-level update mode of the model duration parameter. The model update parameter of **pstFactor** is used when the model duration parameter of each pixel is updated.

[Note]

None

[See Also]

None

IVE_GMM2_CTRL_S

[Description]

Defines the control parameters for GMM2 background modeling.

[Syntax]

```
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;
```

[Member]

Member	Description
enSnsFactorMode	Sensitivity mode, global mode by default In global mode, **u8GlbSnsFactor** is used as the sensitivity coefficient. In pixel mode, the lower eight bits of **pstFactor** in HI_MPI_IVE_GMM2 are used as the sensitivity coefficient.
enLifeUpdateFactorMode	Model duration update mode, global mode by default In global mode, **u16GlbLifeUpdateFactor** is used as the foreground model duration update parameter. In pixel mode, the upper eight bits of **pstFactor** in HI_MPI_IVE_GMM2 are used as the foreground model duration update parameter.
u16GlbLifeUpdateFactor	Global model update parameter Value range: 0–65535 Default value: 4
u16LifeThr	Time for converting a foreground model into a background model Value range: 0–65535 Default value: 5000
u16FreqInitVal	Initial frequency Value range: 0–65535 Default value: 20000
u16FreqReduFactor	Frequency attenuation coefficient Value range: 0–65535 Default value: 0xFF00
u16FreqAddFactor	Model matching frequency increase coefficient Value range: 0–65535 Default value: 0xEF
u16FreqThr	Model failure frequency threshold Value range: 0–65535 Default value: 12000
u16VarRate	Variance update rate Value range: 0–65535 Default value: 1
u9q7MaxVar	Maximum variance Value range: 0–65535 Default value: (16 x 16)<<7
u9q7MinVar	Minimum variance Value range: 0–maximum variance Default value: (8 x 8)<<7

Member	Description
u8GlbSnsFactor	Global sensitivity parameter Value range: 0–255 Default value: 8
u8ModelNum	Number of models Value range: 1–5 Default value: 3

[Note]

None

[See Also]

None

IVE_CANNY_STACK_SIZE_S

[Description]

Defines the stack size of strong edge points in the first phase of Canny edge extraction.

[Syntax]

```
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;
```

[Member]

Member	Description
u32StackSize	Stack size (number of strong edge points)
u8Reserved[12]	Reserved

[Note]

None

[See Also]

None

IVE_CANNY_HYS_EDGE_CTRL_S

[Description]

Defines calculation task control parameters in the first phase of Canny edge extraction.

[Syntax]

```
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;
```

[Member]

Member	Description
stMem	Auxiliary memory For details about the memory size, see the **Note** field of HI_MPI_IVE_CannyHysEdge.
u16LowThr	Low threshold Value range: [0, 255]
u16HighThr	High threshold Value range: [u16LowThr, 255]
as8Mask[25]	Parameter template for calculating the gradient

[Note]

None

[See Also]

None

IVE_LBP_CMP_MODE_E

[Description]

Defines the comparison mode during LBP calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_LBP_CMP_MODE_NORMAL	Simple comparison mode
IVE_LBP_CMP_MODE_ABS	Absolute value comparison mode

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_LBP and Figure 2-16.

[See Also]

IVE_LBP_CTRL_S

IVE_LBP_CTRL_S

[Description]

Defines LBP texture calculation control parameters.

[Syntax]

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

[Member]

Member	Description
enMode	LBP comparison mode
un8BitThr	LBP comparison threshold Value range in IVE_LBP_CMP_MODE_NORMAL mode: [−128, +127] Value range in IVE_LBP_CMP_MODE_ABS mode: [0, 255]

[Note]

For details about the related formula, see the **Note** field of HI_MPI_IVE_LBP and Figure 2-16.

[See Also]

- IVE_LBP_CMP_MODE_E
- IVE_8BIT_U

IVE_NORM_GRAD_OUT_CTRL_E

[Description]

Defines the output control enumeration type for the normalized gradient calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_NORM_GRAD_OUT_CTRL_HOR_AND_VER	Output of both the H, V component diagrams in the gradient information. For details about H and V, see the **Parameter** field of HI_MPI_IVE_NormGrad.
IVE_NORM_GRAD_OUT_CTRL_HOR	Output of only the H component diagram in the gradient information
IVE_NORM_GRAD_OUT_CTRL_VER	Output of only the V component diagram in the gradient information
IVE_NORM_GRAD_OUT_CTRL_COMBINE	Output of the HV diagram in the gradient information that is stored in package format (see Figure 1-7)

[Note]

None

[See Also]

IVE_NORM_GRAD_CTRL_S

IVE_NORM_GRAD_CTRL_S

[Description]

Defines control parameters for the normalized gradient calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
enOutCtrl	Output control mode of gradient information
as8Mask[25]	Template required for the gradient calculation
u8Norm	Normalization parameter Value range: [1, 13]

[Note]

None

[See Also]

IVE_NORM_GRAD_OUT_CTRL_E

IVE_MV_S9Q7_S

[Description]

Defines the LK optical flow displacement.

[Syntax]

```
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;
```

[Member]

Member	Description
s32Status	Feature point tracking status 0: success 1: failure
s9q7Dx	x component of the feature point displacement
s9q7Dy	y component of the feature point displacement

[Note]

None

[See Also]

None

IVE_LK_OPTICAL_FLOW_CTRL_S

[Description]

Defines control parameters for the LK optical flow calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
u16CornerNum	Number of input corner points (also called feature points) Value range: [1, 200]
u0q8MinEigThr	Minimum feature point threshold Value range: [1, 255]
u8IterCount	Maximum iteration times Value range: [1, 20]
u0q8Epsilon	The following is the iteration convergence condition: dx^2 + dy^2 < u0q8Epsilon Value range: [1, 255] Reference value: 2

[Note]

None

[See Also]

None

IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_E

[Description]

Defines the output mode for the pyramid LK optical flow calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_NONE	Neither **pstStatus** nor **pstErr** is output.
IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_STATUS	Only **pstStatus** is output.
IVE_LK_OPTICAL_FLOW_PYR_OUT_MODE_BOTH	Both **pstStatus** and **pstErr** are output.

[Note]

None

[See Also]

None

IVE_LK_OPTICAL_FLOW_PYR_CTRL_S

[Description]

Defines control parameters for the pyramid LK optical flow calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
enOutMode	Output mode for **pstStatus** and **pstErr**
bUseInitFlow	Whether to use the initial optical flow for calculation (whether **pstNextPts** needs to be initialized) HI_TRUE: use HI_FALSE: not use
u16PtsNum	Number of feature points in **pstPrevPts/pstNextPts**, or the size of the array of **pstStatus/pstErr** Value range: [1, 500]
u8MaxLevel	Number of pyramid layers minus 1 Value range: [0, 3], corresponding to pyramid layers 1–4 Reference value: 2
u0q8MinEigThr	Minimum feature point threshold Value range: [1, 255]
u8IterCnt	Maximum iteration times Value range: [1, 20]
u0q8Eps	Iteration convergence condition: dx^2 + dy^2 < **u0q8Epsilon** Value range: [1, 255] Reference value: 2

[Note]

None

[See Also]

None

IVE_ST_MAX_EIG_S

[Description]

Defines the maximum response value of the corner point during the Shi-Tomas-like corner point calculation.

[Syntax]

```
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;
```

[Member]

Member	Description
u16MaxEig	Maximum response value of the corner point
u8Reserved[14]	Reserved

[Note]

None

[See Also]

None

IVE_ST_CANDI_CORNER_CTRL_S

[Description]

Defines the control parameters for calculating Shi-Tomas-like candidate corner points.

[Syntax]

```
typedef struct hiIVE_ST_CANDI_CORNER_CTRL_S
{
    IVE_MEM_INFO_S stMem;
    HI_U0Q8 u0q8QualityLevel;
}IVE_ST_CANDI_CORNER_CTRL_S;
```

[Member]

Member	Description
stMem	Auxiliary memory For details about the memory size, see the **Note** field of HI_MPI_IVE_STCandiCorner.
u0q8QualityLevel	ShiTomasi corner point quality control. The points whose response value is less than (u0q8QualityLevel x Maximum response value of the corner point) are not considered as corner points. Value range: [1, 255] Reference value: 25

[Note]

None

[See Also]

None

IVE_ST_CORNER_INFO_S

[Description]

Defines the output corner point information after the Shi-Tomas-like corner point calculation.

[Syntax]

```
typedef struct hiIVE_ST_CORNER_INFO_S
{
    HI_U16 u16CornerNum;
    IVE_POINT_U16_S astCorner[IVE_ST_MAX_CORNER_NUM];
}IVE_ST_CORNER_INFO_S;
```

[Member]

Member	Description
u16CornerNum	Number of valid corner points
astCorner[IVE_ST_MAX_CORNER_NUM]	Corner point coordinate array

[Note]

None

[See Also]

None

IVE_ST_CORNER_CTRL_S

[Description]

Defines the control parameters for filtering Shi-Tomas-like corner points.

[Syntax]

```
typedef struct hiIVE_ST_CORNER_CTRL_S
{
    HI_U16 u16MaxCornerNum;
    HI_U16 u16MinDist;
}IVE_ST_CORNER_CTRL_S;
```

[Member]

Member	Description
u16MaxCornerNum	Maximum number of corner points Value range: [1, 200]
u16MinDist	Minimum distance between two adjacent corner points Value range: [1, 255] Reference value: 10

[Note]

None

[See Also]

None

IVE_SAD_MODE_E

[Description]

Defines the SAD calculation mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_SAD_MODE_MB_4X4	SAD calculation based on the 4x4 pixel block
IVE_SAD_MODE_MB_8X8	SAD calculation based on the 8x8 pixel block
IVE_SAD_MODE_MB_16X16	SAD calculation based on the 16x16 pixel block

[Note]

None

[See Also]

IVE_SAD_CTRL_S

IVE_SAD_OUT_CTRL_E

[Description]

Defines the SAD output control mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_SAD_OUT_CTRL_16BIT_BOTH	Output mode of the 16-bit SAD image and thresh image
IVE_SAD_OUT_CTRL_8BIT_BOTH	Output mode of the 8-bit SAD image and thresh image
IVE_SAD_OUT_CTRL_16BIT_SAD	Output mode of the 16-bit SAD image
IVE_SAD_OUT_CTRL_8BIT_SAD	Output mode of the 8-bit SAD image
IVE_SAD_OUT_CTRL_THRESH	Output mode of the thresh image

[Note]

None

[See Also]

IVE_SAD_CTRL_S

IVE_SAD_CTRL_S

[Description]

Defines SAD control parameters.

[Syntax]

```
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;
```

[Member]

Member	Description
enMode	SAD calculation mode

Member	Description
enOutCtrl	SAD output control mode
u16Thr	Threshold for determining whether to perform thresholding on the calculated SAD image
u8MinVal	Output value when the input is less than or equal to **u16Thr**
u8MaxVal	Output value when the input is greater than **u16Thr**

[Note]

None

[See Also]

- IVE_SAD_MODE_E
- IVE_SAD_OUT_CTRL_E

IVE_RESIZE_MODE_E

[Description]

Defines the resize mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_RESIZE_MODE_LINEAR	Bidirectional linear interpolation scaling mode
IVE_RESIZE_MODE_AREA	Area interpolation scaling mode

[Note]

None

[See Also]

None

IVE_RESIZE_CTRL_S

[Description]

Defines resize control parameters.

[Syntax]

```
typedef struct hiIVE_RESIZE_CTRL_S
{
    IVE_RESIZE_MODE_E enMode;
     IVE_MEM_INFO_S stMem;
    HI_U16  u16Num;
}IVE_RESIZE_CTRL_S;
```

[Member]

Member	Description
enMode	Scaling mode
stMem	Auxiliary memory. For details, see the **Note** field of HI_MPI_IVE_Resize.
u16Num	Number of images

[Note]

None

[See Also]

None

IVE_RESIZE2_CTRL_S

[Description]

Defines resize2 control parameters.

[Syntax]

```
typedef struct hiIVE_RESIZE2_CTRL_S
{
    HI_U16  u16Num;
}IVE_RESIZE2_CTRL_S;
```

[Member]

Member	Description
u16Num	Number of images

[Note]

None

[See Also]

None

IVE_GRAD_FG_MODE_E

[Description]

Defines the gradient foreground calculation mode.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_GRAD_FG_MODE_USE_CUR_GRAD	Gradient calculation mode of the current position
IVE_GRAD_FG_MODE_FIND_MIN_GRAD	Minimum gradient calculation mode

[Note]

None

[See Also]

IVE_GRAD_FG_CTRL_S

IVE_GRAD_FG_CTRL_S

[Description]

Defines gradient foreground calculation control parameters.

[Syntax]

```
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;
```

[Member]

Member	Description
enMode	Gradient foreground calculation mode. See IVE_GRAD_FG_MODE_E.
u16EdwFactor	Edge width adjustment factor Value range: [500, 2000] Reference value: 1000
u8CrlCoefThr	Gradient vector coefficient threshold Value range: [50, 100] Reference value: 80
u8MagCrlThr	Gradient magnitude threshold Value range: [0, 20] Reference value: 4
u8MinMagDiff	Gradient magnitude difference threshold Value range: [2, 8] Reference value: 2
u8NoiseVal	Gradient magnitude noise threshold Value range: [1, 8] Reference value: 1
u8EdwDark	Black pixel enable (0: disabled; 1: enabled) Reference value: 1

[Note]

None

[See Also]

IVE_GRAD_FG_MODE_E

IVE_CANDI_BG_PIX_S

[Description]

Defines candidate background model data.

[Syntax]

```
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;
```

[Member]

Member	Description
u8q4f4Mean	Mean value of the candidate background gray scale. The upper 12 bits indicate the candidate background pixel value, and the lower four bits indicate the status flag.
u16StartTime	Candidate background start time
u16SumAccessTime	Accumulated access time of the candidate background
u16ShortKeepTime	Short hold time of the candidate background
u8ChgCond	Time condition for triggering the candidate background to change
u8PotenBgLife	Potential background life

[Note]

None

[See Also]

- IVE_WORK_BG_PIX_S
- IVE_BG_MODEL_PIX_S

IVE_WORK_BG_PIX_S

[Description]

Defines working background model data.

[Syntax]

```
typedef struct hiIVE_WORK_BG_PIX_S
{
    HI_U8Q4F4 u8q4f4Mean;          /*0# background gray 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;
```

[Member]

Member	Description
u8q4f4Mean	Gray scale of working background 0. The upper 12 bits indicate the pixel value of working background 0, and the lower four bits indicate the status flag of the working background.
u16AccTime	Accumulated background access time for checking the validity of the working background
u8PreGray	Gray scale of the pixels in the previous frame
u5q3DiffThr	Difference threshold
u8AccFlag	Working background access flag
u8BgGray[3]	Gray scale of backgrounds 1–3

[Note]

None

[See Also]

- IVE_CANDI_BG_PIX_S
- IVE_BG_MODEL_PIX_S

IVE_BG_LIFE_S

[Description]

Defines background life data.

[Syntax]

```
typedef struct hiIVE_BG_LIFE_S
{
    HI_U8 u8WorkBgLife[3];         /*1# ~ 3# background vitality */
    HI_U8 u8CandiBgLife;           /*Candidate background vitality */
}IVE_BG_LIFE_S;
```

[Member]

Member	Description
u8WorkBgLife[3]	Life of working backgrounds 1–3
u8CandiBgLife	Candidate background life

[Note]

None

[See Also]

IVE_BG_MODEL_PIX_S

IVE_BG_MODEL_PIX_S

[Description]

Defines background model data.

[Syntax]

```
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;
```

[Member]

Member	Description
stWorkBgPixel	Working background data
stCandiPixel	Candidate background data
stBgLife	Background life

[Note]

None

[See Also]

- IVE_CANDI_BG_PIX_S
- IVE_WORK_BG_PIX_S
- IVE_BG_LIFE_S

IVE_FG_STAT_DATA_S

[Description]

Defines foreground status data.

[Syntax]

```
typedef struct hiIVE_FG_STAT_DATA_S
{
   HI_U32 u32PixNum;
   HI_U32 u32SumLum;
}IVE_FG_STAT_DATA_S;
```

[Member]

Member	Description
u32PixNum	Number of foreground pixels
u32SumLum	Accumulated sum of the luminance of all pixels of the input image

[Note]

None

[See Also]

None

IVE_BG_STAT_DATA_S

[Description]

Defines background status data.

[Syntax]

```
typedef struct hiIVE_BG_STAT_DATA_S
{
   HI_U32 u32PixNum;
   HI_U32 u32SumLum;
}IVE_BG_STAT_DATA_S;
```

[Member]

Member	Description
u32PixNum	Number of background pixels
u32SumLum	Accumulated sum of the luminance of all pixels of the background image

[Note]

None

[See Also]

None

IVE_MATCH_BG_MODEL_CTRL_S

[Description]

Defines background match control parameters.

[Syntax]

```
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;
```

[Member]

Member	Description
u32CurFrmNum	Current frame time
u32PreFrmNum	Previous frame time Requirement: u32PreFrmNum < u32CurFrmNum
u16TimeThr	Potential background replace time threshold Value range: [2, 100] Reference value: 20
u8DiffThrCrlCoef	Difference threshold and gray scale coefficient Value range: [0, 5] Reference value: 0

Member	Description
u8DiffMaxThr	Upper limit for adjusting the background difference threshold Value range: [3, 15] Reference value: 6
u8DiffMinThr	Lower limit for adjusting the background difference threshold Value range: [3, u8DiffMaxThr] Reference value: 4
u8DiffThrInc	Difference threshold increment under the dynamic background Value range: [0, 6] Reference value: 0
u8FastLearnRate	Fast background learning rate Value range: [0, 4] Reference value: 2
u8DetChgRegion	Changed region detection flag Value range: {0, 1} 0: not detected; 1: detected Reference value: 0

[Note]

None

[See Also]

None

IVE_UPDATE_BG_MODEL_CTRL_S

[Description]

Defines background update control parameters.

[Syntax]

```
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;
```

[Member]

Member	Description
u32CurFrmNum	Current frame time
u32PreChkTime	Last time of checking the background status
u32FrmChkPeriod	Period of checking the background status Value range: [0, 2000] Reference value: 50
u32InitMinTime	Minimum background initialization time Value range: [20, 6000] Reference value: 100
u32StyBgMinBlendTime	Minimum time of blending the steady background Value range: [u32InitMinTime, 6000] Reference value: 200
u32StyBgMaxBlendTime	Maximum time of blending the steady background

Member	Description
	Value range: [u32StyBgMinBlendTime, 40000] Reference value: 1500
u32DynBgMinBlendTime	Minimum time of blending the dynamic background Value range: [0, 6000] Reference value: 0
u32StaticDetMinTime	Minimum time of detecting static objects Value range: [20, 6000] Reference value: 80
u16FgMaxFadeTime	Maximum foreground fade-out time Value range: [1, 255] Reference value: 15
u16BgMaxFadeTime	Maximum background fade-out time Value range: [1, 255] Reference value: 60
u8StyBgAccTimeRateThr	Threshold for the steady background access time rate Value range: [10, 100] Reference value: 80
u8ChgBgAccTimeRateThr	Threshold for the variable background access time rate Value range: [10, 100] Reference value: 60
u8DynBgAccTimeThr	Threshold for the dynamic background access time rate Value range: [0, 50] Reference value: 0
u8DynBgDepth	Dynamic background depth Value range: [0, 3] Reference value: 3
u8BgEffStaRateThr	Threshold for the background status time rate in the background initialization phase Value range: [90, 100] Reference value: 90
u8AcceBgLearn	Background learning acceleration flag Value range: {0, 1} 0: not accelerated; 1: accelerated Reference value: 0
u8DetChgRegion	Changed region detection flag Value range: {0, 1} 0: not detected; 1: detected

Member	Description
	Reference value: 0

[Note]

The following condition must be met:

u32InitMinTime ≤ u32StyBgMinBlendTime ≤ u32StyBgMaxBlendTime

[See Also]

None

IVE_LOOK_UP_TABLE_S

[Description]

Defines the lookup table.

[Syntax]

```
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;
```

[Member]

Member	Description
stTable	Information about the data memory after a lookup table is created
u32ElemNum	Number of lookup table elements
s32TabInLower	Lower limit of the value range of a created lookup table
s32TabInUpper	Upper limit of the value range of a created lookup table
u8TabInPreci	Lookup table creation precision. The interval of creating lookup tables is (**s32TabInUpper** – **s32TabInLower**)/(1 << **u8TabInPreci**).
u8TabOutNorm	Number of bits to be shifted or divisor for raw data normalization when lookup tables are being created

[Note]

None

[See Also]

None

IVE_ANN_MLP_ACCURATE_E

[Description]

Defines the type of the ANN_MLP input eigenvector.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_ANN_MLP_ACCURATE_SRC16_WGT16	SQ16.16 input eigenvector type
IVE_ANN_MLP_ACCURATE_SRC14_WGT20	SQ18.14 input eigenvector type

[Note]

None

[See Also]

None

IVE_ANN_MLP_ACTIV_FUNC_E

[Description]

Defines the enumeration type of ANN_MLP activation functions.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_ANN_MLP_ACTIV_FUNC_IDENTITY	Identity activation function
IVE_ANN_MLP_ACTIV_FUNC_SIGMOID_SYM	Sigmoid symmetric activation function
IVE_ANN_MLP_ACTIV_FUNC_GAUSSIAN	Gaussian activation function

[Note]

For details about the function definition, see the **Note** field of HI_MPI_IVE_ANN_MLP_Predict.

[See Also]

None

IVE_ANN_MLP_MODEL_S

[Description]

Defines ANN_MLP model data.

[Syntax]

For the Hi3519:

```
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;
```

For other chips:

```
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
   layers, every layerCount ≤ 256*/
   HI_U16 u16MaxCount;         /*MaxCount ≤ 256*/
   HI_U8 u8LayerNum;            /*2 < layerNum ≤ 8*/
}IVE_ANN_MLP_MODEL_S;
```

[Member]

Member	Description
enActivFunc	Activation function type
enAccurate	Input eigenvector type
stWeight	Model data weight
au16LayerCount[8]	One input layer, several hidden layers (1–6), and one output layer that store the number of characteristics dimensions at the input layer ([1, 256]), number of neurons at each hidden layer ([2, 256]), and number of characteristics dimensions at the output layer ([1, 256]) respectively
u16MaxCount	Number of neurons at all layers or the maximum characteristics dimension. The formula is as follows: $\max\limits_{0\le i<\mathrm{u8LayerNum}}\{au16LayerCount[i]\}$
u8LayerNum	Number of layers in the ANN_MLP system Value range: [3, 8]

[Note]

None

[See Also]

- IVE_ANN_MLP_ACTIV_FUNC_E
- IVE_ANN_MLP_ACCURATE_E

IVE_SVM_TYPE_E

[Description]

Defines the SVM type.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_SVM_TYPE_C_SVC	Classification mode
IVE_SVM_TYPE_NU_SVC	Regression mode

[Note]

None

[See Also]

None

IVE_SVM_KERNEL_TYPE_E

[Description]

Defines the SVM kernel function type.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_SVM_KERNEL_TYPE_LINEAR	Linear kernel function
IVE_SVM_KERNEL_TYPE_POLY	Polynomial kernel function
IVE_SVM_KERNEL_TYPE_RBF	Radial basis kernel function
IVE_SVM_KERNEL_TYPE_SIGMOID	Sigmoid kernel function

[Note]

For details about the core function definition, see the **Note** field of HI_MPI_IVE_SVM_Predict.

[See Also]

None

IVE_SVM_MODEL_S

[Description]

Defines SVM model data.

[Syntax]

```
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;
```

[Member]

Member	Description
enType	SVM type. Only the IVE_SVM_TYPE_C_SVC mode is supported currently.
enKernelType	SVM kernel function type. For details, see IVE_SVM_KERNEL_TYPE_E.
stSv	Supported vector in model data. For details about memory allocation, see the **Note** field of HI_MPI_IVE_SVM_Predict.
stDf	Parameter of the decision function in model data
u32TotalDfSize	Total number of bytes of all decision function parameters
u16FeatureDim	Number of input characteristics dimensions Value range: [1, 256]
u16SvTotal	Number of supported vectors Value range: [1, 3000]
u8ClassCount	Number of classifications Value range: [2, 80]

[Note]

None

[See Also]

- IVE_SVM_TYPE_E

- IVE_SVM_KERNEL_TYPE_E

IVE_CNN_ACTIV_FUNC_E

[Description]

Defines the enumeration types of CNN activation functions.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_CNN_ACTIV_FUNC_NONE	No activation function
IVE_CNN_ACTIV_FUNC_RELU	ReLU activation function
IVE_CNN_ACTIV_FUNC_SIGMOID	Sigmoid activation function, not supported currently

[Note]

None

[See Also]

None

IVE_CNN_POOLING_E

[Description]

Defines the enumeration types of the CNN pooling operation.

[Syntax]

```
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;
```

[Member]

Member	Description
IVE_CNN_POOLING_NONE	No image pooling
IVE_CNN_POOLING_MAX	Maximum value of the feature points in the pooling range
IVE_CNN_POOLING_AVG	Average value of the feature points in the pooling range

[Note]

None

[See Also]

None

IVE_CNN_CONV_POOLING_S

[Description]

Defines the parameters of the CNN single-layer Conv-ReLU-Pooling convolution operation package.

[Syntax]

```
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;
```

[Member]

Member	Description
enActivFunc	Activation function type
enPooling	Pooling type
u8FeatureMapNum	Number of output feature images in the convolution operation package Value range: 1−50

Member	Description
u8KernelSize	Size of the convolution kernel in the convolution operation package, which can only be 3 x 3
u8ConvStep	Shift stride of the convolution kernel in the convolution operation package, which can only be 1
u8PoolSize	Size of the pooling window, which can only be 2 x 2
u8PoolStep	Shift stride for the pooling window, which can only be 2

[Note]

None

[See Also]

- IVE_CNN_ACTIV_FUNC_E
- IVE_CNN_POOLING_E

IVE_CNN_FULL_CONNECT_S

[Description]

Defines the CNN full-connection network parameters.

[Syntax]

```
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;
```

[Member]

Member	Description
au16LayerCnt[8]	Number of neuron nodes at each full-connection layer. The value range is 1–1024 for the input layer (output of Conv-ReLU-Pooling), 2–256 for the middle hidden layer, and 1–256 for the output layer.
u16MaxCnt	Maximum number of nodes at each full-connection layer
u8LayerNum	Number of full-connection layers, ranging from 3 to 8

[Note]

None

[See Also]

None

IVE_CNN_MODEL_S

[Description]

Defines the parameters of the CNN model.

[Syntax]

```
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;
```

[Member]

Member	Description
astConvPool[8]	Parameter configuration for the convolution operation package at each layer
stFullConnect	Parameter configuration for the full-connection operation package
stConvKernelBias	Convolution kernel and bias coefficients in all the convolution operation packages

Member	Description
u32ConvKernelBiasSize	Number of bytes for the convolution kernel and bias coefficients in all the convolution operation packages
stFCLWgtBias	Weight and bias coefficients in the full-connection operation package
u32FCLWgtBiasSize	Number of bytes for the weight and bias coefficients in the full-connection operation package
u32TotalMemSize	Number of memory bytes for the convolution kernel, weight, and bias coefficients required for CNN calculation
enType	Image type of the input CNN model. Only the U8C1 gray-scale image or RGB_PLANAR color image are supported.
u16Width	Image width of the input CNN model Value range: 16–80
u16Height	Image height of the input CNN model Value range: 16–(1280/width)
u16ClassCount	Number of types of the CNN model classification tasks (consistent with the output of the last full-connection layer) Value range: 1–256
u8ConvPoolLayerNum	Number of CNN model convolution operation packages Value range: 1–8

[Note]

For details about the structure of the CNN model, see the **Note** field of HI_MPI_IVE_CNN_Predict.

[See Also]

- IVE_CNN_ACTIV_FUNC_E
- IVE_CNN_POOLING_E
- IVE_CNN_CONV_POOLING_S
- IVE_CNN_FULL_CONNECT_S

IVE_CNN_CTRL_S

[Description]

Defines the control parameters for a CNN prediction task.

[Syntax]

```
typedef struct hiIVE_CNN_CTRL_S
{
    IVE_MEM_INFO_S stMem;   /*Assist memory*/
   HI_U32 u32Num;          /*Input image number*/
}IVE_CNN_CTRL_S;
```

[Member]

Member	Description
stMem	Auxiliary memory for CNN prediction calculation. For details about the required memory size, see the **Note** field of HI_MPI_IVE_CNN_Predict.
u32Num	Number of image for the input CNN model

[Note]

None

[See Also]

None

IVE_CNN_RESULT_S

[Description]

Defines the prediction result of a single CNN sample.

[Syntax]

```
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;
```

[Member]

Member	Description
s32ClassIdx	Prediction type index for the CNN model
s32Confidence	Confidence of the type predicted by the CNN model

[Note]

None

[See Also]

None

IVE_MODULE_PARAMS_S

[Description]

Defines IVE module parameters.

[Syntax]

```
typedef struct hiIVE_MODULE_PARAMS_S
{
    HI_BOOL bSavePowerEn;
}IVE_MODULE_PARAMS_S;
```

[Member]

Member	Description
bSavePowerEn	Whether to enable the power consumption reduction function

[Note]

This data structure is described in **hi_module_param.h**. When HuaweiLite OS is used, this data structure can be used to set IVE module parameters in the initialization function. This data structure applies only to the HuaweiLite OS version, and is not contained in the Linux version.

[See Also]

ive_mod_init

4 Error Codes

Table 4-1 describes IVE MPI error codes.

Table 4-1 IVE MPI error codes

Error Code	Macro Definition	Description
0xA01D8001	HI_ERR_IVE_INVALID_DEVID	The device ID is invalid.
0xA01D8002	HI_ERR_IVE_INVALID_CHNID	The channel ID or the region handle is invalid.
0xA01D8003	HI_ERR_IVE_ILLEGAL_PARAM	The parameter is invalid.
0xA01D8004	HI_ERR_IVE_EXIST	The device, channel, or resource to be created exists.
0xA01D8005	HI_ERR_IVE_UNEXIST	The device, channel, or resource to be used or destroyed does not exist.
0xA01D8006	HI_ERR_IVE_NULL_PTR	The pointer is null.
0xA01D8007	HI_ERR_IVE_NOT_CONFIG	The module is not configured.
0xA01D8008	HI_ERR_IVE_NOT_SUPPORT	The parameter or function is not supported.
0xA01D8009	HI_ERR_IVE_NOT_PERM	The operation, for example, attempting to modify the value of a static parameter, is forbidden.
0xA01D800C	HI_ERR_IVE_NOMEM	The memory fails to be allocated for the reasons such as insufficient system memory.
0xA01D800D	HI_ERR_IVE_NOBUF	The buffer fails to be allocated. For example, the requested data buffer is too large.
0xA01D800E	HI_ERR_IVE_BUF_EMPTY	There is no image in the buffer.
0xA01D800F	HI_ERR_IVE_BUF_FULL	The buffer is full of images.

Error Code	Macro Definition	Description
0xA01D8010	HI_ERR_IVE_NOTREADY	The system is not initialized or the corresponding driver is not loaded.
0xA01D8011	HI_ERR_IVE_BADADDR	The address is invalid.
0xA01D8012	HI_ERR_IVE_BUSY	The system is busy.
0xA01D8040	HI_ERR_IVE_SYS_TIMEOUT	The system times out.
0xA01D8041	HI_ERR_IVE_QUERY_TIMEOUT	Querying times out.
0xA01D8042	HI_ERR_IVE_OPEN_FILE	Opening a file fails.
0xA01D8043	HI_ERR_IVE_READ_FILE	Reading a file fails.
0xA01D8044	HI_ERR_IVE_WRITE_FILE	Writing to a file fails.

5 Proc Debugging Information

5.1 Overview

The debugging information is obtained from the proc file system on Linux. The information reflects the current system status and can be used to locate and analyze problems.

[File Directory]

/proc/umap

View the proc information in either of the following ways:

- Run a **cat** command such as **cat /proc/umap/ive** on the console or run file operation commands such as **cp /proc/umap/ive ./** to copy all the proc files to the current directory.
- Read the preceding files as common read-only files through applications such as fopen and fread.

NOTE

Note the following when reading parameter descriptions:

- For the parameter whose value is **0** or **1**, if mapping between the values and definitions is not specified, the value **1** indicates affirmative and the value **0** indicates negative.
- For the parameter whose value is **aaa**, **bbb**, or **ccc**, if the mapping between the values and the definitions is not specified, identify the parameter definitions based on **aaa**, **bbb**, or **ccc**.

5.2 Proc Information

[Debugging Information]

```
~ # 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
```

[Analysis]

This section records the current status, resources, and operators of the IVE.

[Parameter Description]

Parameter		Description
MODULE PARAM	save_power	Low-power mode enable 0: disabled 1: enabled
IVE QUEUE INFO	Wait	Waiting queue ID (0 or 1)
	Busy	ID of the queue that is being scheduled (0, 1, or −1) The value **−1** indicates that IVE hardware is idle.
	WaitCurId	ID of the first valid task in the waiting queue
	WaitEndId	ID of the last valid task in the waiting queue + 1
	BusyCurId	ID of the first valid task in the queue being scheduled

Parameter		Description
	BusyEndId	ID of the last valid task in the queue being scheduled + 1
IVE TASK INFO	Hnd	ID of the handle of the current task that can be allocated
	TaskFsh	Number of completed tasks
	LastId	ID of the previously completed task
	TaskId	ID of the currently completed task
	HndWrap	Number of times that the user handle ID is wrapped
	FshWrap	Number of times that completed tasks are wrapped
IVE RUN-TIME INFO	LastInst	**bInstant** value transferred when users submit tasks last time
	CntPerSec	Number of times that interrupt functions are called in the last second
	MaxCntPerSec	Historical maximum number of times that interrupt functions are called in one second
	TotalIntCntLastSec	Number of times that interrupts are reported in the last second
	TotalIntCnt	Number of times that the IVE generates interrupts
	QTCnt	Number of times that querying IVE linked lists times out
	STCnt	Number of times that the IVE system times out
	CostTm	Duration (in µs) in which interrupt functions are called last time
	MCostTm	Maximum duration (in µs) in which interrupt functions are called
	CostTmPerSec	Number of times that interrupt functions are called in the last second
	MCostTmPerSec	Historical maximum duration (in µs) in which interrupt functions are called in one second
	IntTotalCostTm	Total time (in µs) of handling interrupts
	RunTm	Total running time of the IVE (in s)
IVE INVOKE INFO	DMA	DMA calling times
IVE UTILI INFO	Utili	IVE utilization

[Note]

- It is recommended that the low-power mode be disabled during code debugging and be enabled after debugging is complete.
- When the low-power mode is disabled, the IVE usage statistical function (**Utili**) is unavailable.

6 FAQs

6.1 Differences Between the IVE Clib on the PC and IVE SDK on the Board for Algorithm Development

No.	Keyword	IVE Clib on the PC	IVE SDK on the Board
1	Handle	Invalid	The handle can work with HI_MPI_IVE_Query to query whether an operator is complete when necessary. For details, see the description of "Handle" in section 1.2.1 "Important Concepts."
2	bInstant	Invalid	**bInstant** can be configured based on the algorithm to reduce interrupts and improve the performance. For details, see the description of "**bInstant** (instant returned result flag)" in section 1.2.1 "Important Concepts."
3	Query	The query operation is unnecessary. Success is returned for each query operation.	To use the result of the IVE hardware operator, the user must query whether a task is complete. For details, see the description of "Query" in section 1.2.1 "Important Concepts" and HI_MPI_IVE_Query.
4	Memory allocation, physical address, and virtual address	The memory is allocated by calling the malloc function. Values need to be assigned to the virtual addresses obtained after memory allocation. To simulate the feature that the IVE hardware uses the physical address, the Clib also uses the physical address. The virtual addresses that are forcibly converted into the HI_U32 type must be assigned to the physical addresses.	The IVE hardware uses the physical address. The memory is allocated by calling HI_MPI_SYS_MmzMalloc or HI_MPI_SYS_MmzAlloc_Cached (for details, see the *HiMPP Vx.y Media Processing Software Development Reference*). The physical address and virtual address are generated after memory allocation. In addition to memory allocation, the VB memory of other modules can also be used.
5	Address alignment	The Clib does not require address alignment.	The hardware addresses need to be aligned as required.

No.	Keyword	IVE Clib on the PC	IVE SDK on the Board
6	Chip difference	The Clib is a function universal set. The interfaces are updated to the latest version.	The functions supported by the chip are a subset of the Clib functions based on the actual requirements. Some upgraded interfaces may be different from the corresponding interfaces in the latest Clib version.
7	Asynchronous, synchronous, parallel, or serial execution	The Clib and algorithms are executed serially in the CPU. Therefore, there is no asynchronization issue.	The IVE hardware and the CPU work in asynchronous mode. Based on this feature, the IVE and the CPU can work in parallel to improve the performance. However, the IVE and the CPU need to be synchronous when the CPU needs to use the IVE result.

6.2 Differences Between the IVE and OpenCV for Algorithm Development

- The IVE and the CPU are asynchronous. Therefore, the CPU must query whether an IVE task is complete. However, the query operation is not required when the OpenCV is used for developing algorithms.
- Generally the IVE parameters are fixed-point parameters, and fixed-point numbers are used for internal calculation. The OpenCV parameters are floating ones, and floating numbers are used for internal calculation. Therefore, for the same function, the IVE has limits on the value range and precision compared with the OpenCV.
- The IVE uses the physical address, and the start address and stride need to be aligned. The OpenCV does not use the physical address and has no alignment requirement.
- An operator is implemented by both hardware and software. Therefore, multiple IVE hardware interfaces and software interfaces may be required to implement one OpenCV operator.

6.3 Establishment of the ANN/SVM Lookup Table

To establish a lookup table by using the f(u) function, perform the following steps:

Step 1 Specify the value range of the independent variable **u**. If the value range of **u** is [a, b], assume that r = b − a (a and b correspond to **s32TabInLower** and **s32TabInUpper** respectively). When the ANN lookup table is established, **u** is the independent variable. When the SVM lookup table is established, **u** corresponds to $x_i^T x_j$ or $\left\|x_i - x_j\right\|^2$. For details about the formulas, see the **Note** fields in HI_MPI_IVE_ANN_MLP_Predict and HI_MPI_IVE_SVM_Predict.

Step 2 Specify the sampling number **g** in one unit of the independent variable. **g** is calculated as follows: g = 1<<u8TabInPreci. The number of elements in the lookup table is calculated as follows: u16ElemNum = r x g=r<<u8TabInPreci. There are limits on the maximum number of elements in the ANN/SVM lookup table. For details, see the **Note** fields in HI_MPI_IVE_ANN_MLP_Predict and HI_MPI_IVE_SVM_Predict.

Step 3 Specify the value range of f(u). Typically the value range needs to be restricted to [−1, +1]. Therefore, **u8TabOutNorm** or **1<<u8TabOutNorm** can serve as the divisor and be used to normalize f(u). For the ANN, only **1<<u8TabOutNorm** can be used as the divisor. For the SVM, both **u8TabOutNorm** and **1<<u8TabOutNorm** can be used as the divisor. Therefore, the divisor needs to be input for the SVM model conversion in **ive_xml2bin_ui.exe**, and the input divisor must be the same as the normalization divisor for establishing the lookup table.

Step 4 Generate the f(u) lookup table based on the corresponding f(u) formulas and the sampling value of **u**, and save the lookup table in **stTable**.

NOTE

For details about the parameters **s32TabInLower**, **s32TabInUpper**, **u8TabInPreci**, **u16ElemNum**, **u8TabOutNorm**, and **stTable** in the preceding steps, see the description of IVE_LOOK_UP_TABLE_S.

----End

6.4 Cache Memory

Whether the allocated memory has the cache depends on the memory user. The IVE directly reads the DDR memory data. Therefore, if the memory used by the IVE has the cache, the cache must be flushed to ensure data consistency. If the IVE uses the memory and the CPU does not use the memory or uses the memory only once, it is recommended that the memory without cache be allocated. If the CPU uses the memory, it is recommended that the memory with cache be allocated.