MVE User Manual

V1.2

REVISION HISTORY

Revision No.	Description	Date
1.0	• Created	11/30/2015
1.1	• Add support of HW accelerator	1/9/2017
1.2	• Change typedef of variable • Rename return eumn	3/28/2017
1.3	• Add new function description	4/10/2017

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

1.1. Overview

The MStar video engine (MVE) is a software controllable module in the intelligent analysis system of MStar. Current MVE allows you to develop intelligent analysis solutions for video diagnosis and boundary security.

2.API REFERENCE

2.1. MVE function list

MI_MVE_Init: Performs MVE initialization task.
MI_MVE_Uninit: Performs MVE uninitialization task.
MI_MVE_AllocateImage: Allocate an Image
MI_MVE_FreeImage: Free an Image
MI_MVE_AllocateBuffer: Allocate a buffer
MI_MVE_FreeImage: Free a buffer
MI_MVE_Filter: Performs a 5x5 template filtering task.
MI_MVE_CSC: Performs a color space conversion (CSC) task.
MI_MVE_FilterAndCSC: Performs a task combined with template filter and CSC.
MI_MVE_Sobel: Performs a 5x5 template task for calculating the Sobel-like gradient.
MI_MVE_MagAndArg: Performs a 5x5 template task for calculating the gradient magnitude and argument.
MI_MVE_Dilate: Performs a dilate task.
MI_MVE_Erode: Performs an erode task.
MI_MVE_Thresh: Performs a thresh task.
MI_MVE_And: Performs an AND task for two images.
MI_MVE_Sub: Performs a subtraction task for two images.
MI_MVE_Or: Performs an OR task for two images.
MI_MVE_Xor: Performs an XOR task for two images.
MI_MVE_Add: Performs a weighted addition task for two gray-scale images.
MI_MVE_Thresh_S16: Performs a threshold task from S16 data to 8-bit data.
MI_MVE_Thresh_U16: Performs a threshold task from S16 data to U8 data.
MI_MVE_16BitTo8Bit: Performs a linear conversion task from 16-bit data to 8-bit data.
MI_MVE_OrdStatFilter: Performs a 3x3 template order statistics filter task.
MI_MVE_Bernsen: Performs a Bernsen thresh task for the 3x3 and 5x5 windows.
MI_MVE_Map: Performs a map task.
MI_MVE_NCC: Performs a normalized cross-correlation coefficient (NCC) coefficient calculation task for two images with the same resolution.
MI_MVE_Integ: Creates an integrogram statistics task.
MI_MVE_CCL: Creates a connected component labeling (CCL) task for binary images.
MI_MVE_Hist: Performs a histogram statistics task for gray-scale images.
MI_MVE_EqualizeHist: Performs a histogram equalization task for gray-scale images.
MI_MVE_SAD: Performs a sum of the absolute pixel differences task for two gray-scale images.

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

MI_MVE_LBP: Creates a local binary pattern (LBP) calculation task.

MI_MVE_GMM: Creates a GMM background modeling task.

MI_MVE_CannyHysEdge: Creates a Canny edge extraction task for gray-scale images (first phase of Canny edge extraction for gray-scale images).

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

MI_MVE_LineFilterHor: Creates a horizontal density filter task for binary images.

MI_MVE_LineFilterVer : Creates a vertical density filter task for binary images.

MI_MVE_NoiseRemoveHor: Creates a horizontal noise removal task for binary images.

MI_MVE_NoiseRemoveVer: Creates a vertical noise removal task for binary images.

MI_MVE_AdpThresh: Creates an adaptive thresh task.

MI_MVE_LKOpticalFlow: Create an optical flow tracking task.

2.2. MI_MVE_Init

2.2.1 [Description]

Performs MVE initialization task.

2.2.2 [Syntax]

MVE_HANDLE MI_MVE_Init(MVE_IMG_INFO * pstImgInfo);

2.2.3 [Parameter]

Parameter	Description	Input/Output
pstImgInfo	Pointer to the source image information. It cannot be null.	Input

2.2.4 [Return Value]

Return Value	Description
0	Initialization failure.
Other value	MVE_HANDLE pointer address.

2.2.5 [Requirement]

Header files: **mi_mve.h**

2.2.6 [Note]

- None

2.2.7 [See Also]

- MVE_IMG_INFO

2.3. MI_MVE_Uninit

2.3.1 [Description]

Performs MVE uninitialization task.

2.3.2 [Syntax]

MI_RET MI_MVE_Uninit(MVE_HANDLE pMVEHandle);

2.3.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the MVE_HANDLE created by MI_MVE_Init. It cannot be null.	Input

2.3.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.3.5 [Requirement]

Header files: **mi_mve.h**

2.3.6 [Note]

- None

2.3.7 [See Also]

- MVE_HANDLE

2.4. MI_MVE_AllocateImage

2.4.1 [Description]

Allocate an Image for MVE tasks

2.4.2 [Syntax]

MI_RET MI_MVE_AllocateImage(MVE_HANDLE pMVEHandle, MVE_IMAGE_S *pstImage, MVE_IMAGE_TYPE_E enImageType, U16 u16Stride, U16 u16Width, U16 u16Height, const char *pszName, MVE_MEM_ALLOC_TYPE_E enAllocType);

2.4.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstImage	Pointer to the image which needs memory allocation.	Output
enImageType	Image type	Input
u16Stride	Image stride	Input
u16Width	Image width	Input
u16Height	Image height	Input
pszName	A unique name for the image, the length of name is 14 byte.	Input
enAllocType	Memory type of image	Input

2.4.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.4.5 [Requirement]

Header files: **mi_mve.h**

2.4.6 [Note]

MI_MVE_AllocateImage() is a wrapper for structure MVE_IMAGE_S, include source and destination images, and also an interface to allocate a physically-continues memory for HW accelerator.

2.4.7 [See Also]

2.5. MI_MVE_FreeImage

2.5.1 [Description]

Free an Image for MVE tasks

2.5.2 [Syntax]

MI_RET MI_MVE_FreeImage(MVE_HANDLE pMVEHandle, MVE_IMAGE_S *pstImage);

2.5.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle	Input

	It cannot be null.	
pstImage	Pointer to the image which needs free memory.	Input, Output

2.5.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.5.5 [Requirement]

Header files: **mi_mve.h**

2.5.6 [Note]

MI_MVE_FreeImage() and MI_MVE_AllocateImage() pair of API for image allocation and free.

2.5.7 [See Also]

2.6. MI_MVE_AllocateBuffer

2.6.1 [Description]

Allocate a buffer for MVE tasks

2.6.2 [Syntax]

MI_RET MI_MVE_AllocateBuffer(MVE_HANDLE pMVEHandle, MVE_MEM_INFO_S *pstBuffer, S32 u32Size, const char *pszName, MVE_MEM_ALLOC_TYPE_E enAllocType);

2.6.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstBuffer	Pointer to the buffer which needs memory allocation.	Output
u32Size	Buffer size	Input
pszName	A unique name for the buffer, the length of name is 14 byte.	Input
enAllocType	Memory type of buffer	Input

2.6.4 [Return Value]

Return Value	Description

MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.6.5 [Requirement]

Header files: **mi_mve.h**

2.6.6 [Note]

MI_MVE_AllocateBuffer() is a wrapper for structure MVE_MEM_INFO_S, include source and destination images, and also an interface to allocate a physically-continues memory for HW accelerator.

2.6.7 [See Also]

2.7. MI_MVE_FreeBuffer

2.7.1 [Description]

Free a buffer for MVE tasks

2.7.2 [Syntax]

MI_RET MI_MVE_FreeBuffer(MVE_HANDLE pMVEHandle, MVE_MEM_INFO_S *pstBuffer);

2.7.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstBuffer	Pointer to the buffer which needs free memory.	Input, Output

2.7.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.7.5 [Requirement]

Header files: **mi_mve.h**

2.7.6 [Note]

MI_MVE_FreeBuffer() and MI_MVE_AllocateBuffer() pair of API for buffer allocation and free.

2.7.7 [See Also]

2.8. MI_MVE_Filter

2.8.1 [Description]

Performs a 5x5 template filtering task. User can set template coefficients to implement various filter effects.

2.8.2 [Syntax]

MI_RET MI_MVE_Filter(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_FILTER_CTRL_S *pstFltCtrl, bool bInstant);

2.8.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	U8C1, YUV420SP, and YUV422SP
pstDst	U8C1, YUV420SP, and YUV422SP

2.8.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.8.5 [Requirement]

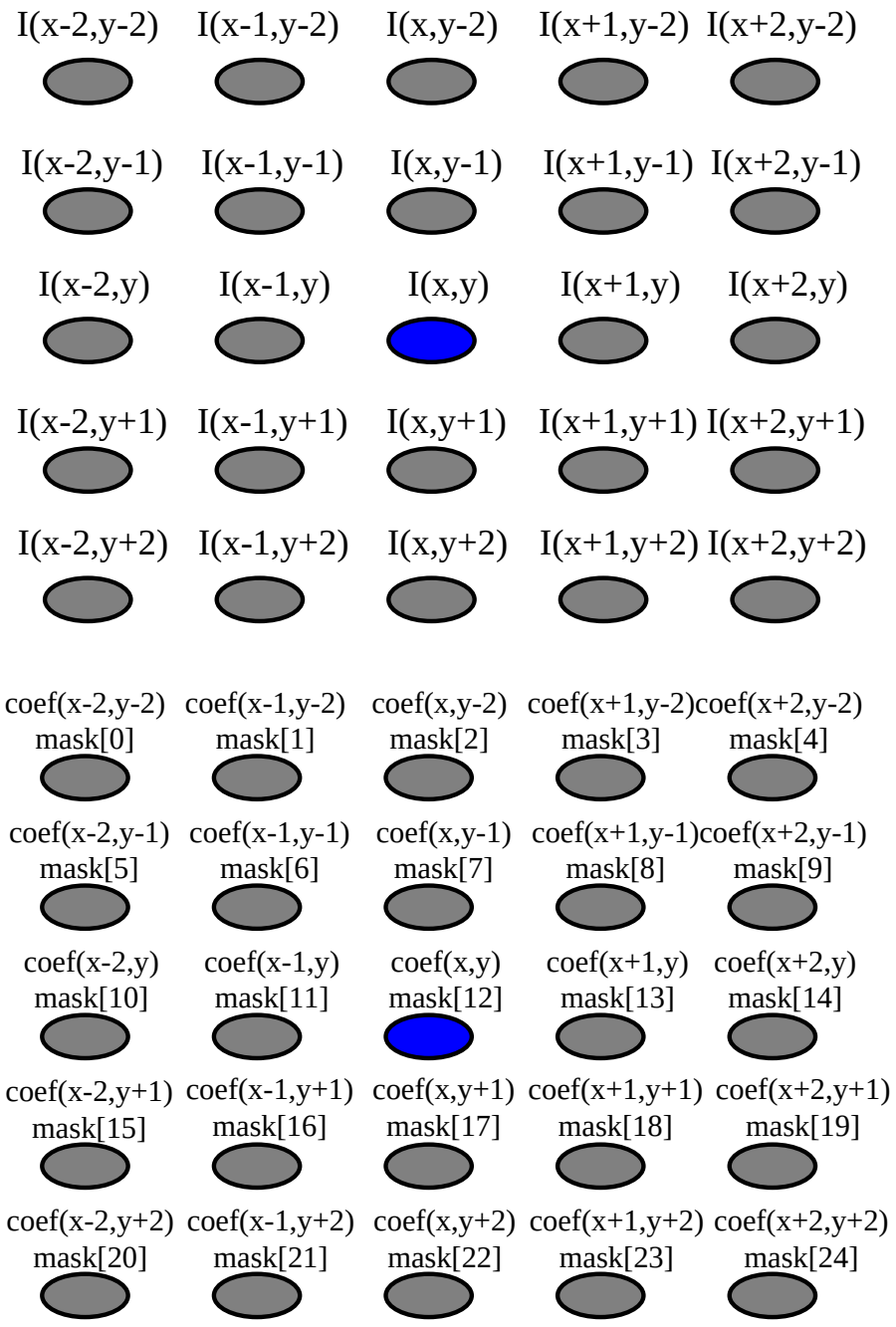
Header files: **mi_mve.h**

2.8.6 [Note]

For every pixel in support format and filter coefficient defined by user (See

MVE_FILTER_CTRL_S), we perform the following equation:

Figure 2-1 Filter formula



$$I_{out}(x,y) = \{ \sum_{-2 \le i \le 2} \sum_{-2 \le j \le 2} I_{in}(x+i, y+j) * coef\,(x+i, y+j) \} >> norm$$

Classic Gaussian template

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad u8Norm = 4$$

$$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 5 & 6 & 5 & 2 \\ 3 & 6 & 8 & 6 & 3 \\ 2 & 5 & 6 & 5 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix} \quad u8Norm = 8$$

$$*3$$

$$\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \quad u8Norm = 8$$

2.8.7 [See Also]

2.9. MI_MVE_CSC

2.9.1 [Description]

Performs a CSC task for implementing YUV2RGB/ RGB2YUV conversion.

2.9.2 [Syntax]

MI_RET MI_MVE_CSC(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_CSC_CTRL_S *pstCscCtrl, bool bInstant)

2.9.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	YUV420SP, YUV422SP U8C3_PLANAR, and U8C3_PACKAGE
pstDst	YUV420SP, YUV422SP U8C3_PLANAR, and U8C3_PACKAGE

2.9.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success

Other values	Failure. For details, see Error Code

2.9.5 [Requirement]

Header files: **mi_mve.h**

2.9.6 [Note]

A total of 2 working mode are supported. The value range varies according to the working mode. For details, see MVE_CSC_MODE_E.

2.9.7 [See Also]

2.10. MI_MVE_FilterAndCSC

2.10.1 [Description]

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

2.10.2 [Syntax]

MI_RET MI_MVE_FilterAndCSC(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_FILTER_AND_CSC_CTRL_S *pstFltCscCtrl, bool bInstant);

2.10.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	YUV420SP and YUV422SP
pstDst	U8C3_PLANAR and U8C3_PACKAGE

2.10.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.10.5 [Requirement]

Header files: **mi_mve.h**

2.10.6 [Note]

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

2.10.7 [See Also]

2.11. MI_MVE_Sobel

2.11.1 [Description]

Performs a 5x5 template task fr calculatinog the Sobel-like gradient.

2.11.2 [Syntax]

MI_RET MI_MVE_Sobel(MVE_HANDLE pMVEHandle,
MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDstH,
MVE_DST_IMAGE_S *pstDstV, MVE_SOBEL_CTRL_S *pstSobelCtrl,
bool bInstant);

2.11.3 [Parameter]

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

	component image V 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**.	
pstSobelCtrl	Pointer to control information It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	U8C1
pstDstH	S16C1
pstDstV	S16C1

2.11.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see **Error Code**

2.11.5 [Requirement]

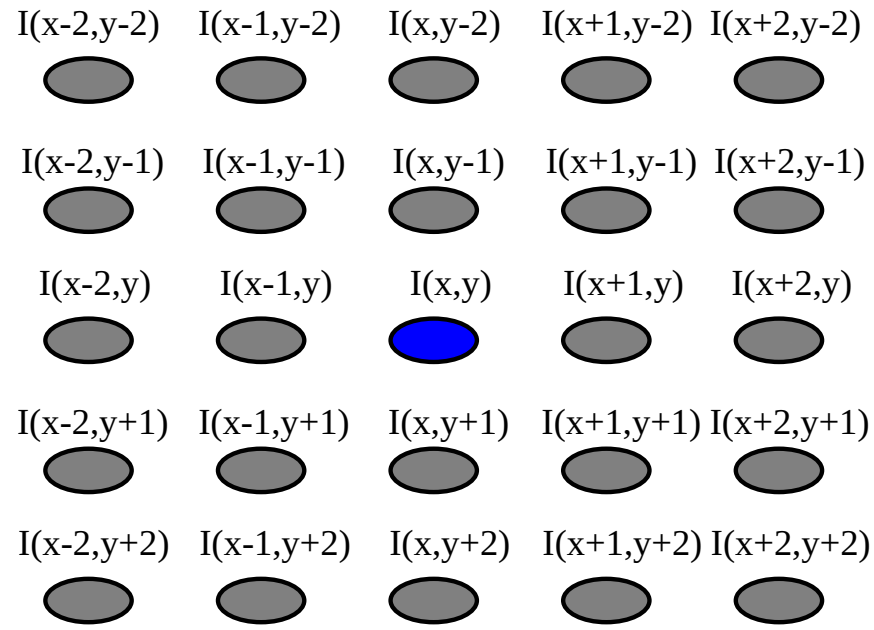
Header files: **mi_mve.h**

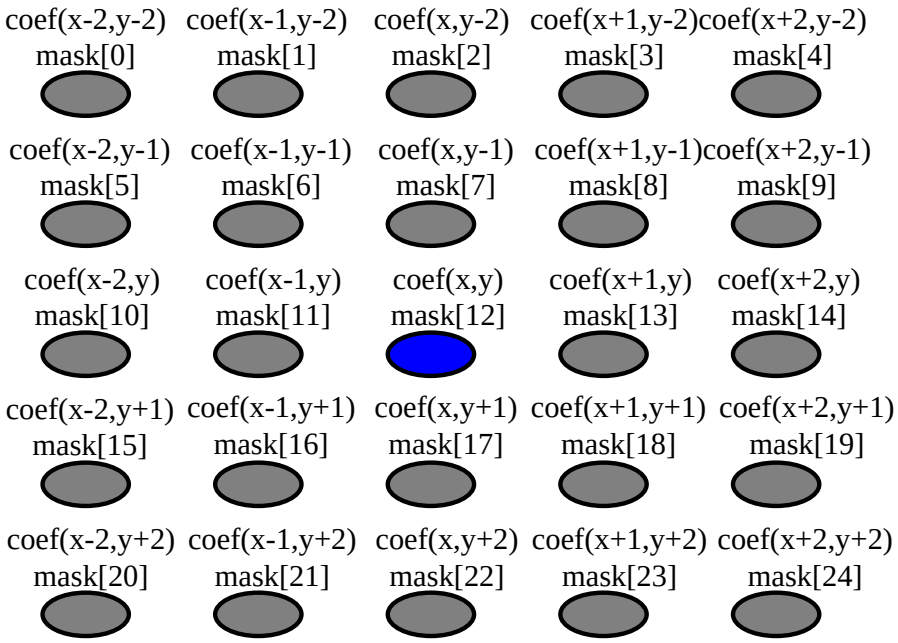
2.11.6 [Note]

Three output modes are supported. For details, see MVE_SOBEL_OUT_CTRL_E.

If the output mode is **MVE_SOBEL_OUT_CTRL_BOTH**, the strides of pstDstH and pstDstV must be the same.

Figure 2-2 Sobel formula





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

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

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:

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

The following shows the Scharr template:

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

The following shows the Laplace template:

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

2.11.7 [See Also]

2.12. MI_MVE_MagAndAng

2.12.1 [Description]

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

2.12.2 [Syntax]

MI_RET MI_MVE_MagAndAng(MVE_HANDLE pMVEHandle,
MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDstMag,
MVE_DST_IMAGE_S *pstDstAng, MVE_MAG_AND_ANG_CTRL_S *pstMagAndAngCtrl,
bool bInstant);

2.12.3 [Parameter]

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

	It cannot be null.	
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	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	U8C1
pstDstMag	U16C1
pstDstAng	U8C1

2.12.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see **Error Code**

2.12.5 [Requirement]

Header files: **mi_mve.h**

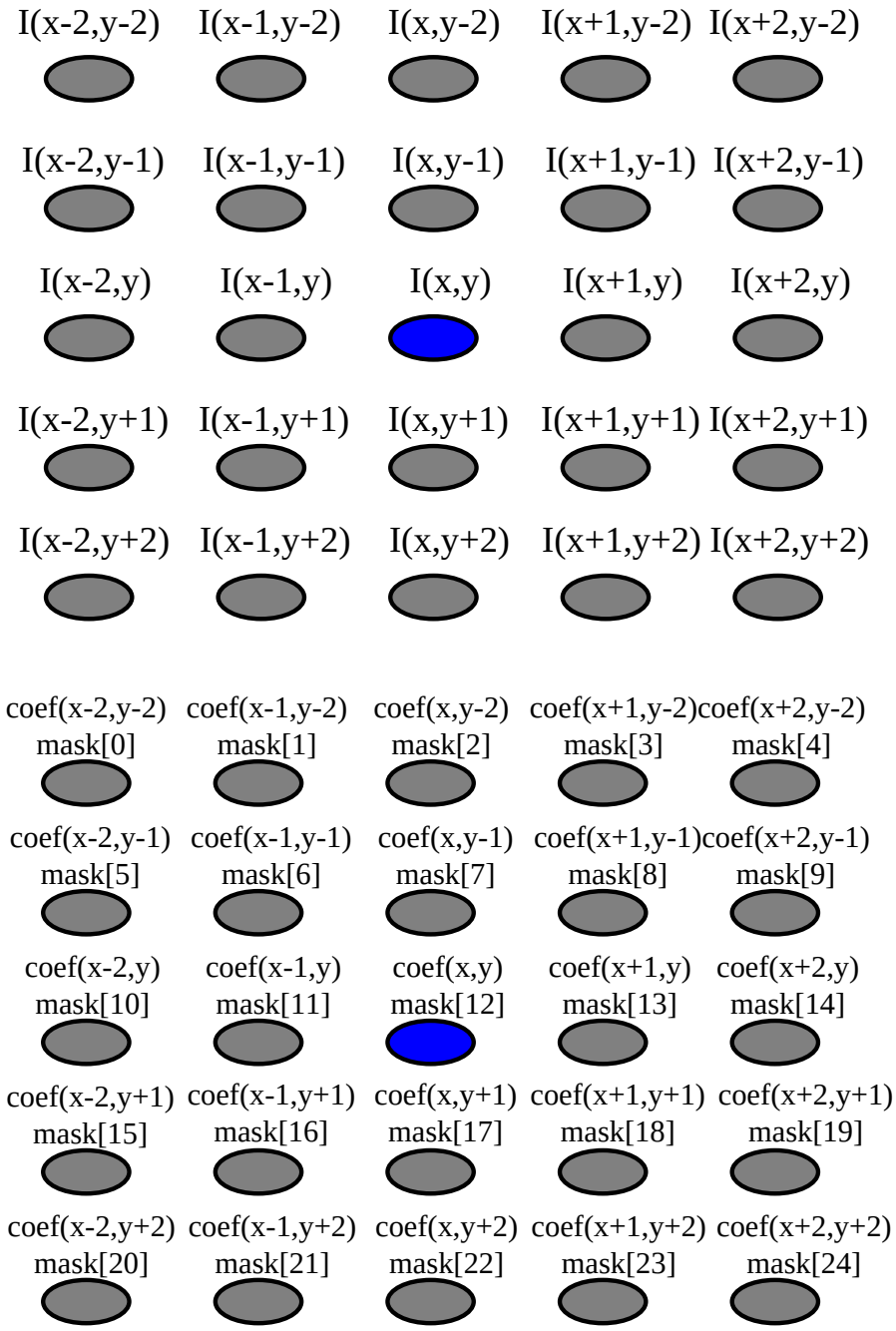
2.12.6 [Note]

Two output formats are supported. For details, see MVE_MAG_AND_ANG_OUT_CTRL_E.
If the output mode is MVE_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:

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

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

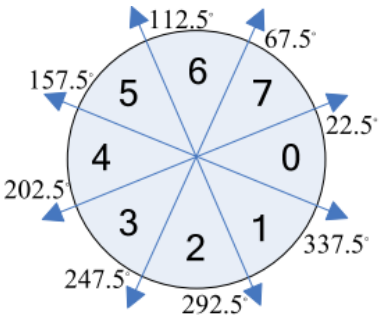
Figure 2-3 MagAndAng formula



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

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

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



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

2.12.7 [See Also]

2.13. MI_MVE_Dilate

2.13.1 [Description]

Performs a 5x5 template dilate task for binary images.

2.13.2 [Syntax]

MI_RET MI_MVE_Dilate(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_DILATE_CTRL_S *pstDilateCtrl, bool bInstant);

2.13.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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 It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc	U8C1 binary image

pstDst	U8C1 binary image

2.13.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see **Error Code**

2.13.5 [Requirement]

Header files: **mi_mve.h**

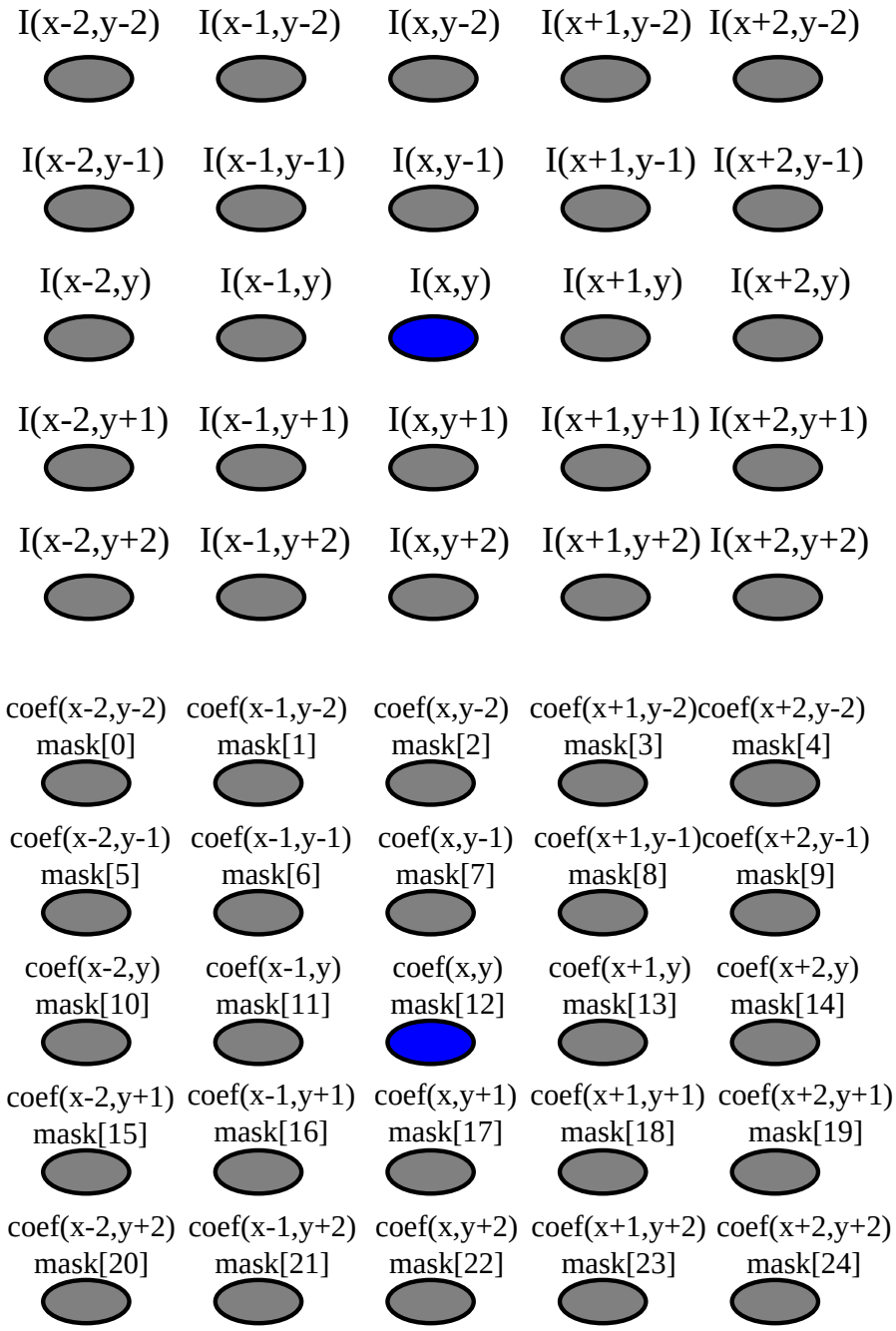
2.13.6 [Note]

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

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

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

Figure 2-4 Dilate formula



$$I_{out}(x,y) = \underset{-2\le i\le 2}{\mathrm{O}}\left(\underset{-2\le j\le 2}{\mathrm{O}}\left(f(i,j)\right)\right)$$

where

$$f(i,j) = I(x-i,y-j)\,\&\,coef(x-i,y-j)$$

and

$$\underset{-2\le k\le 2}{\mathrm{O}}\left(g(k)\right) = g(-2)\,|\,g(-1)\,|\,g(0)\,|\,g(1)\,|\,g(2),$$

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

2.13.7 [See Also]

2.14. MI_MVE_Erode

2.14.1 [Description]

Performs a 5x5 template erode task for binary images.

2.14.2 [Syntax]

MI_RET MI_MVE_Erode(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_ERODE_CTRL_S *pstErodeCtrl, bool bInstant);

2.14.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc	U8C1 binary image
pstDst	U8C1 binary image

2.14.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.14.5 [Requirement]

Header files: **mi_mve.h**

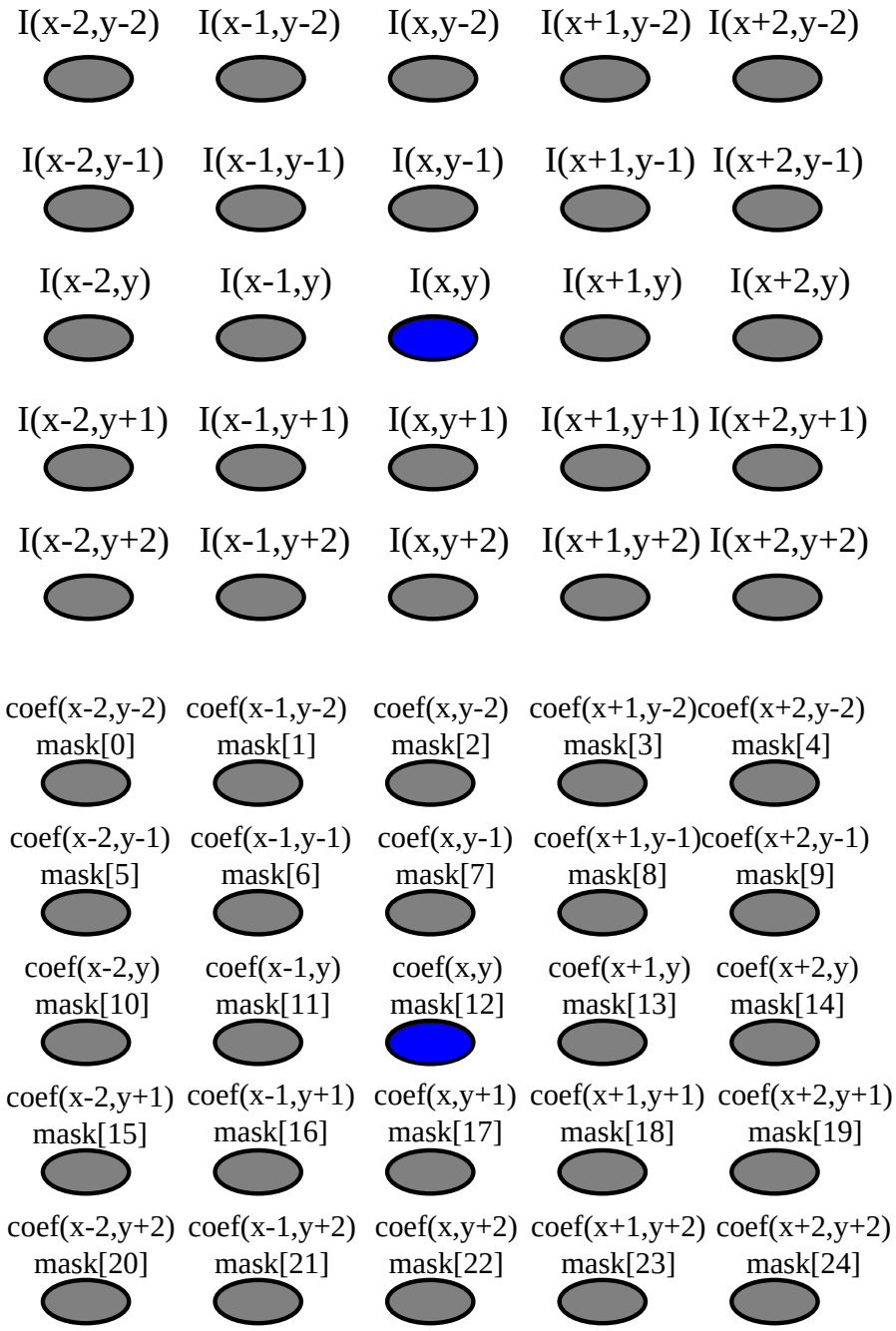
2.14.6 [Note]

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

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

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

Figure 2-5 Erode formula



$$I_{out}(x,y) = \underset{-2\le i\le 2}{\mathrm{O}}\left(\underset{-2\le j\le 2}{\mathrm{O}}\left(f(i,j)\right)\right)$$

where

$$f(i,j) = I(x-i,y-j)\,|\,\left(255 - coef\,(x-i,y-j)\right)$$

and

$$\underset{-2\le k\le 2}{\mathrm{O}}\left(g(k)\right) = g(-2)\,\&\,g(-1)\,\&\,g(0)\,\&\,g(1)\,\&\,g(2),$$

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

2.14.7 [See Also]

2.15. MI_MVE_Thresh

2.15.1 [Description]

Performs a thresh task for gray-scale images.

2.15.2 [Syntax]

MI_RET MI_MVE_Thresh(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_THRESH_CTRL_S *pstThrCtrl, bool bInstant);

2.15.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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 It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc	U8C1
pstDst	U8C1

2.15.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see **Error Code**

2.15.5 [Requirement]

Header files: **mi_mve.h**

2.15.6 [Note]

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

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

  - MVE_THRESH_MODE_TRUNC

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

  - MVE_THRESH_MODE_TO_MINVAL

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

  - MVE_THRESH_MODE_MIN_MID_MAX

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

  - MVE_THRESH_MODE_ORI_MID_MAX

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

  - MVE_THRESH_MODE_MIN_MID_ORI

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

  - MVE_THRESH_MODE_MIN_ORI_MAX

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

- MVE_THRESH_MODE_ORI_MID_ORI

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

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.

2.15.7 [See Also]

2.16. MI_MVE_And

2.16.1 [Description]

Performs an AND task for two images.

2.16.2 [Syntax]

MI_RET MI_MVE_And(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S*pstSrc1,MVE_SRC_IMAGE_S *pstSrc2, MVE_DST_IMAGE_S *pstDst, bool bInstant)

2.16.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrc1	Pointer to the source image 1 It cannot be null.	Input
pstSrc2	Pointer to the 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	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc1	U8C1 binary image
pstSrc2	U8C1 binary image
pstDst	U8C1 binary image

2.16.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.16.5 [Requirement]

Header files: **mi_mve.h**

2.16.6 [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**.

2.16.7 [See Also]

2.17. MI_MVE_Sub

2.17.1 [Description]

Performs a subtraction task for two gray-scale images.

2.17.2 [Syntax]

MI_RET MI_MVE_Sub(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc1, MVE_SRC_IMAGE_S *pstSrc2, MVE_DST_IMAGE_S *pstDst, MVE_SUB_CTRL_S *pstSubCtrl, bool bInstant)

2.17.3 [Parameter]

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

pstSrc1	Pointer to the source image 1 It cannot be null.	Input
pstSrc2	Pointer to the 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	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc1	U8C1
pstSrc2	U8C1
pstDst	U8C1 and S8C1

2.17.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.17.5 [Requirement]

Header files: **mi_mve.h**

2.17.6 [Note]

Two output formats are supported. For details, see MVE_SUB_MODE_E.

MVE_SUB_MODE_ABS

– Formula:

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

– Output format: U8C1

MVE_SUB_MODE_SHIFT

Formula:

$$I_{out}(x, y) = (I_{src1}(x, y) - I_{src2}(x, y)) >> 1$$

– 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**.

2.17.7 [See Also]

2.18. MI_MVE_Or

2.18.1 [Description]

Performs an OR task for two binary images.

2.18.2 [Syntax]

MI_RET MI_MVE_Or(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc1, MVE_SRC_IMAGE_S *pstSrc2, MVE_DST_IMAGE_S *pstDst, bool bInstant);

2.18.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrc1	Pointer to the source image 1 It cannot be null.	Input
pstSrc2	Pointer to the 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	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc1	U8C1
pstSrc2	U8C1
pstDst	U8C1

2.18.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure.

	For details, see Error Code

2.18.5 [Requirement]

Header files: **mi_mve.h**

2.18.6 [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**.

2.18.7 [See Also]

None

2.19. MI_MVE_Xor

2.19.1 [Description]

Performs an XOR task for two binary images.

2.19.2 [Syntax]

MI_RET MI_MVE_Xor(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc1, MVE_SRC_IMAGE_S *pstSrc2, MVE_DST_IMAGE_S *pstDst, bool bInstant);

2.19.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrc1	Pointer to the source image 1 It cannot be null.	Input
pstSrc2	Pointer to the 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	Reserved Flag	Input

Parameter	Supported Image Type

pstSrc1	U8C1
pstSrc2	U8C1
pstDst	U8C1

2.19.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.19.5 [Requirement]

Header files: **mi_mve.h**

2.19.6 [Note]

The following is the formula :

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

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

2.19.7 [See Also]

None

2.20. MI_MVE_Add

2.20.1 [Description]

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

2.20.2 [Syntax]

MI_RET MI_MVE_Add(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc1, MVE_SRC_IMAGE_S *pstSrc2, MVE_DST_IMAGE_S *pstDst, MVE_ADD_CTRL_S *pstAddCtrl, bool bInstant);

2.20.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrc1	Pointer to the source image 1 It cannot be null.	Input
pstSrc2	Pointer to the source image 2 It cannot be null.	Input

	The height and width are the same as those of **pstSrc1**	
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	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc1	U8C1
pstSrc2	U8C1
pstDst	U8C1

2.20.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.20.5 [Requirement]

Header files: **mi_mve.h**

2.20.6 [Note]

The following is the formula :

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

$I_{src1}(x,y)$ corresponds to **pstSrc1**, $I_{src2}(x,y)$ corresponds to **pstSrc2**, and $I_{out}(x,y)$ corresponds to **pstDst**. x and y correspond to u0q16X and u0q16Y in pstAddCtrl. The following conditions must be met before the fixed points:

$$\begin{cases} 0 \le x \le 1 \\ 0 \le y \le 1 \\ x + y = 1 \end{cases}$$

2.20.7 [See Also]

None

2.21. MI_MVE_Thresh_S16

2.21.1 [Description]

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

2.21.2 [Syntax]

MI_RET MI_MVE_Thresh_S16(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_THRESH_S16_CTRL_S *pstThrS16Ctrl, bool bInstant);

2.21.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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 **pstSrc1**	Output
pstThrS16Ctrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc	S16C1
pstDst	U8C1 and S8C1

2.21.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.21.5 [Requirement]

Header files: **mi_mve.h**

2.21.6 [Note]

- Four operation modes are supported. For details, see MVE_THRESH_S16_MODE_E
- The following are related formulas:

  - MVE_THRESH_S16_MODE_S16_TO_S8_MIN_MID_MAX

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

Requirements:

$$-32768\le \text{lowThr}\le \text{highThr}\le 32767$$

$$-128\le \text{minVal},\ \text{midVal},\ \text{maxVal}\le 127$$

  - MVE_THRESH_S16_MODE_S16_TO_S8_MIN_ORI_MAX

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

Requirements:

$$-129\le \text{lowThr}\le \text{highThr}\le 127$$

$$-128\le \text{minVal},\ \text{maxVal}\le 127$$

  - MVE_THRESH_S16_MODE_S16_TO_U8_MIN_MID_MAX

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

Requirements:

$$-32768\le \text{lowThr}\le \text{highThr}\le 32767$$

$$0\le \text{minVal},\ \text{midVal},\ \text{maxVal}\le 255$$

  - MVE_THRESH_S16_MODE_S16_TO_U8_MIN_ORI_MAX

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

Requirements:

$$-1\le\text{lowThr}\le\text{highThr}\le 255$$

$$0\le\text{minVal},\ \text{midVal},\ \text{maxVal}\ \le 255$$

I (x, y) corresponds to pstSrc, I (x, y) out corresponds to pstDst, and mode, lowThr, highThr, minVal, midVal, and maxVal correspond to enMode, s16LowThr, s16HighThr, un8MinVal, un8MidVal, and un8MaxVal in pstThrS16Ctrl respectively.

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

2.21.7 [See Also]

None

2.22. MI_MVE_Thresh_U16

2.22.1 [Description]

Performs a threshold task from U16 data to U8 data.

2.22.2 [Syntax]

MI_RET MI_MVE_Thresh_U16(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_THRESH_U16_CTRL_S *pstThrU16Ctrl, bool bInstant);

2.22.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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 **pstSrc1**	Output
pstThrU16Ctrl	Pointer to the control parameter It cannot be null.	Input

bInstant	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc	U16C1
pstDst	U8C1

2.22.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.22.5 [Requirement]

Header files: **mi_mve.h**

2.22.6 [Note]

- Four operation modes are supported. For details, see MVE_THRESH_U16_MODE_E
- The following are related formulas:
  - MVE_THRESH_U16_MODE_U16_TO_U8_MIN_MID_MAX

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

Requirements:

$$0\le \text{lowThr}\le \text{highThr}\le 65535$$

$$0\le \text{minVal},\ \text{midVal},\ \text{maxVal}\ \le 255$$

  - MVE_THRESH_U16_MODE_U16_TO_U8_MIN_ORI_MAX

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

Requirements:

$$0 \le lowThr \le highThr \le 255$$

$$0 \le minVal,\ maxVal \le 255$$

I (x, y) corresponds to pstSrc, I (x, y) out corresponds to pstDst, and mode, lowThr, highThr, minVal, midVal, and maxVal correspond to enMode, u16LowThr, u16HighThr, un8MinVal, un8MidVal, and un8MaxVal in pstThrU16Ctrl respectively.

- It is not required that values of un8MinVal, un8MidVal, and un8MaxVal in pstThrU16Ctrl meet the requirements in the variable name conventions

2.22.7 [See Also]

None

2.23. MI_MVE_16BitTo8Bit

2.23.1 [Description]

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

2.23.2 [Syntax]

MI_RET MI_MVE_16BitTo8Bit(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_16BIT_TO_8BIT_CTRL_S *pst16BitTo8BitCtrl, bool bInstant);

2.23.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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 **pstSrc1**	Output
pst16BitTo8BitCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc	U16C1 and S16C1

pstDst	U8C1 and S8C1

2.23.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.23.5 [Requirement]

Header files: **mi_mve.h**

2.23.6 [Note]

- Four operation modes are supported. For details, see MVE_16BIT_TO_8BIT_MODE_E
- The following are related formulas:
  - MVE_16BIT_TO_8BIT_MODE_S16_TO_S8

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

  - MVE_16BIT_TO_8BIT_MODE_S16_TO_S8

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

  - MVE_16BIT_TO_8BIT_MODE_S16_TO_U8_BIAS

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

– MVE_16BIT_TO_8BIT_MODE_U16_TO_U8

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

I (x, y) corresponds to pstSrc, I (x, y) out corresponds to pstDst, and mode, a, b, and bias correspond to enMode, u8Numerator, u16Denominator, and s8Bias in pst16BitTo8BitCtrl respectively.
The following requirement must be met: u8Numerator ≤ u16Denominator and u16Denominator ≠ 0.

2.23.7 [See Also]

None

2.24. MI_MVE_OrdStatFilter

2.24.1 [Description]

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

2.24.2 [Syntax]

MI_RET MI_MVE_OrdStatFilter(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc,MVE_DST_IMAGE_S *pstDst, MVE_ORD_STAT_FILTER_CTRL_S *pstOrdStatFltCtrl, bool bInstant);

2.24.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle	Input

	It cannot be null.	
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	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	U8C1
pstDst	U8C1

2.24.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.24.5 [Requirement]

Header files: **mi_mve.h**

2.24.6 [Note]

- Three filtering modes are supported. For details, see **MVE_ORD_STAT_FILTER_MODE_E**.
- The following are related formulas:
  - MVE_ORD_STAT_FILTER_MODE_MEDIAN

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

  - MVE_ORD_STAT_FILTER_MODE_MAX

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

  - MVE_ORD_STAT_FILTER_MODE_MIN

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

2.24.7 [See Also]

2.25. MI_MVE_Bernsen

2.25.1 [Description]

Performs a Bernsen thresh task for the 3x3 and 5x5 windows.

2.25.2 [Syntax]

MI_RET MI_MVE_Bernsen(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc,MVE_DST_IMAGE_S *pstDst, MVE_BERNSEN_CTRL_S *pstBernsenCtrl, bool bInstant);

2.25.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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
pstBernsenCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	U8C1
pstDst	U8C1

2.25.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.25.5 [Requirement]

Header files: **mi_mve.h**

2.25.6 [Note]

- Two modes are supported. For details, see **MVE_BERSEN_MODE_E**.

- The following are related formulas:
  - MVE_BERNSEN_MODE_NORMAL

$$T(x,y)=0.5\times\left(\max_{-\omega\le i\le\omega,-\omega\le j\le\omega} I(x+i,y+j)+\min_{-\omega\le i\le\omega,-\omega\le j\le\omega} I(x+i,y+j)\right)$$

$$I_{out}(x,y)=\begin{cases}0 & I(x,y)<T(x,y)\\ 1 & I(x,y)\ge T(x,y)\end{cases}$$

    No value needs to be assigned to **pstBernsenCtrl->u8Thr**.
  - MVE_BERNSEN_MODE_THRESH

$$T(x,y)=0.5\times\left(\max_{\substack{-\omega\le i\le\omega\\-\omega\le j\le\omega}} I(x+i,y+j)+\min_{\substack{-\omega\le i\le\omega\\-\omega\le j\le\omega}} I(x+i,y+j)\right)$$

$$I_{out}(x,y)=\begin{cases}0 & I(x,y)<0.5\times(T(x,y)+Thr)\\ 1 & I(x,y)\ge 0.5\times(T(x,y)+Thr)\end{cases}$$

    I(x,y) corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, and **Thr** corresponds to **u8thr** in **pstBernsenCtrl.**

2.25.7 [See Also]

2.26. MI_MVE_Map

2.26.1 [Description]

Performs 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.

2.26.2 [Syntax]

MI_RET MI_MVE_Map(MVE_HANDLE pMVEHandle,MVE_SRC_IMAGE_S *pstSrc, MVE_MEM_INFO_S *pstMap, MVE_DST_IMAGE_S *pstDst, bool bInstant);

2.26.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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(MVE_MAP_LUT_MEM_S)	Input
pstDst	Pointer to the output image	Output

	It cannot be null. The height and width are the same as those of **pstSrc**.	
bInstant	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	U8C1
pstDst	U8C1

2.26.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.26.5 [Requirement]

Header files: **mi_mve.h**

2.26.6 [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.**

2.26.7 [See Also]

2.27. MI_MVE_NCC

2.27.1 [Description]

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

2.27.2 [Syntax]

MI_RET MI_MVE_NCC(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc1, MVE_SRC_IMAGE_S *pstSrc2, MVE_NCC_DST_MEM_S *pstDst, bool bInstant);

2.27.3 [Parameter]

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

	It cannot be null.	
pstSrc2	Pointer to the source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**.	Input
pstDst	Pointer to the output data It cannot be null.	Output
bInstant	Reserved Flag	Input

Parameter	Support Image Type
pstSrc1	U8C1
pstSrc2	U8C1

2.27.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.27.5 [Requirement]

Header files: **mi_mve.h**

2.27.6 [Note]

- The following is the formula:

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

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

2.27.7 [See Also]

2.28. MI_MVE_Integ

2.28.1 [Description]

Creates an integrogram statistics task for gray-scale images.

2.28.2 [Syntax]

MI_RET MI_MVE_Integ(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_INTEG_CTRL_S *pstIntegCtrl, bool bInstant);

2.28.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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	Reserved Flag	Input

Parameter	Support Image Type
pstSrc	U8C1
pstDst	U32C1 and U64C1

2.28.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.28.5 [Requirement]

Header files: **mi_mve.h**

2.28.6 [Note]

- **MVE_INTEG_OUT_CTRL_COMBINE** indicates the combined output mode. In this mode, the output image type must be **MVE_IMAGE_TYPE_U64C1**. The following is the formula:

$$I_{sum}(x, y)=\sum_{i\le x}\sum_{j\le y} I(i, j)$$

$$I_{sq}(x, y)=\sum_{i\le x}\sum_{j\le y}(I(i, j)\bullet I(i, j))$$

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

- **MVE_INTEG_OUT_CTRL_SUM** indicates the sum integrogram output mode. In this mode, the output image type must be **MVE_IMAGE_TYPE_U32C1**. The following is the formula: $I_{sum}(x, y)=\sum_{i\le x}\sum_{j\le y} I(i, j)$

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

- **MVE_INTEG_OUT_CTRL_SQSUM** indicates the square sum integrogram output mode. In this mode, the output image type must be **MVE_IMAGE_TYPE_U64C1**. The following is the formula:

  $$I_{sq}(x, y)=\sum_{i\le x}\sum_{j\le y}(I(i, j)\bullet I(i, j))$$

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

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

2.28.7 [See Also]

2.29. MI_MVE_CCL

2.29.1 [Description]

Creates a CCL task for binary images.

2.29.2 [Syntax]

MI_RET MI_MVE_CCL(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrcDst, **MVE_CCBLOB_S** *pstBlob, **MVE_CCL_CTRL_S** *pstCclCtrl, bool bInstant);

2.29.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrcDst	Pointer to the source image. The	Input, output

	source image is modified during CCL, that is, the source image is also the labeling image output. The pointer cannot be null.	
pstBlob	Pointer to connected component information It cannot be null. The minimum memory size is **sizeof(MVE_CCBLOB_S)**, and a maximum of 255 valid connected components are output.	Output
pstCclCtrl	Pointer to control parameter It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Support Image Type
pstSrcDst	U8C1
pstBlob	

2.29.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.29.5 [Requirement]

Header files: **mi_mve.h**

2.29.6 [Note]

- The labeling of eight connected components is used.
- Connected component information is stored in **pstBlob→astRegion**.
- **pstBlob→u8RegionNum** indicates the number of valid connected components, and its maximum value is 255. 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 255 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**.

2.29.7 [See Also]

2.30. MI_MVE_Hist

2.30.1 [Description]

Creates a histogram statistics task for gray-scale images.

2.30.2 [Syntax]

MI_RET MI_MVE_Hist(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_MEM_INFO_S *pstDst, bool bInstant);

2.30.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrc	Pointer to the source image It cannot be null.	Input
pstDst	Pointer to mapping table information It cannot be null. The minimum memory size is sizeof(MVE_HIST_TABLE_MEM_S).	Output
bInstant	Reserved Flag	Input

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

2.30.4 [Return Value]

Return Value	Description
0	Success

Other values	Failure. For details, see Error Code

2.30.5 [Requirement]

Header files: **mi_mve.h**

2.30.6 [Note]

- The following is the formula:

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

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

2.30.7 [Example]

None

2.30.8 [See Also]

None

2.31. MI_MVE_EqualizeHist

2.31.1 [Description]

Creates a histogram equalization task for gray-scale images.

2.31.2 [Syntax]

MI_RET MI_MVE_EqualizeHist(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_EQUALIZE_HIST_CTRL_S *pstEqualizeHistCtrl, bool bInstant);

2.31.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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	Reserved Flag	Input

Parameter	Support Image Type	Address Alignment Mode	Resolution

pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1080
pstDst	U8C1	16 bytes	64 x 64 to 1920 x 1080
pstEqualizeHistCtrl→stMem	-	16 bytes	-

2.31.4 [Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see Error Code

2.31.5 [Requirement]

Header files: **mi_mve.h**

2.31.6 [Note]

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

2.31.7 [Example]

None

2.31.8 [See Also]

None

2.32. MI_MVE_SAD

2.32.1 [Description]

Performs a sum of the absolute pixel differences task for two gray-scale images. It calculates the accumulated sum of the absolute differences corresponding to two images by block.

2.32.2 [Syntax]

MI_RET MI_MVE_SAD(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc1, MVE_SRC_IMAGE_S *pstSrc2, MVE_DST_IMAGE_S *pstDst1, MVE_DST_IMAGE_S * pstDst2, **MVE_SAD_CTRL_S** *pstSADCtrl, bool bInstant)

2.32.3 [Parameter]

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

	It cannot be null.	
pstSrc2	Pointer to the source image 2 It cannot be null. The height and width are the same as those of **pstSrc1**	Input
pstDst1	Pointer to SAD output image. It cannot be null if the output is required based on **pstSADCtrl→enOutCtrl**. The height and width depends on **pstSADCtrl→enMode**.	Output
pstDst2	Pointer to SAD output image. It cannot be null if the output is required based on **pstSADCtrl→enOutCtrl**. The height and width depends on **pstSADCtrl→enMode**.	Output
pstSADCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag	Input

Parameter	Supported Image Type
pstSrc1	U8C1
pstSrc2	U8C1
pstSad	U8C1 and U16C1
pstThr	U8C1

2.32.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.32.5 [Requirement]

Header files: **mi_mve.h**

2.32.6 [Note]

– Five output formats are supported. For details, see **MVE_SAD_OUT_CTRL_E**.
– For two W x H input images, output size is (W/M) x (H/M), M could be 4, 8 or 16, corresponds to **enMode** in **pstSADCtrl.**
– SAD formula:

$$O(x,y)=\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}abs(I_{src1}(M*x+i,M*y+j)-I_{src2}(M*x+i,M*y+j))$$

$$Thr(x,y)=\begin{cases}minVal & O(x,y)\le Thresh\\ maxVal & O(x,y)>Thresh\end{cases}$$

For 16-bit SAD output, SAD(x,y) = O(x,y).
For 8-bit SAD output, SAD(x,y) = O(x,y)/M^2.
$I_{src1}(x,y)$ corresponds to **pstSrc1**, $I_{src2}(x,y)$ corresponds to **pstSrc2**, *SAD(x,y)* corresponds to **pstSad** and *Thr(x,y)* corresponds to **pstThr**.

2.32.7 [See Also]

- **MVE_SAD_CTRL_S**

2.33. MI_MVE_NormGrad

2.33.1 [Description]

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

2.33.2 [Syntax]

MI_RET MI_MVE_NormGard(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDstH, MVE_DST_IMAGE_S *pstDstV, MVE_DST_IMAGE_S *pstDstHV, MVE_NORM_GRAD_CTRL_S *pstNormGradCtrl, bool bInstant);

2.33.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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 and normalization to S8	Output

	It cannot be null if the output is required based on **pstNormGradCtrl→enOutCtrl**.	
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 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	Reserved Flag.	Input

Parameter	Support 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

2.33.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

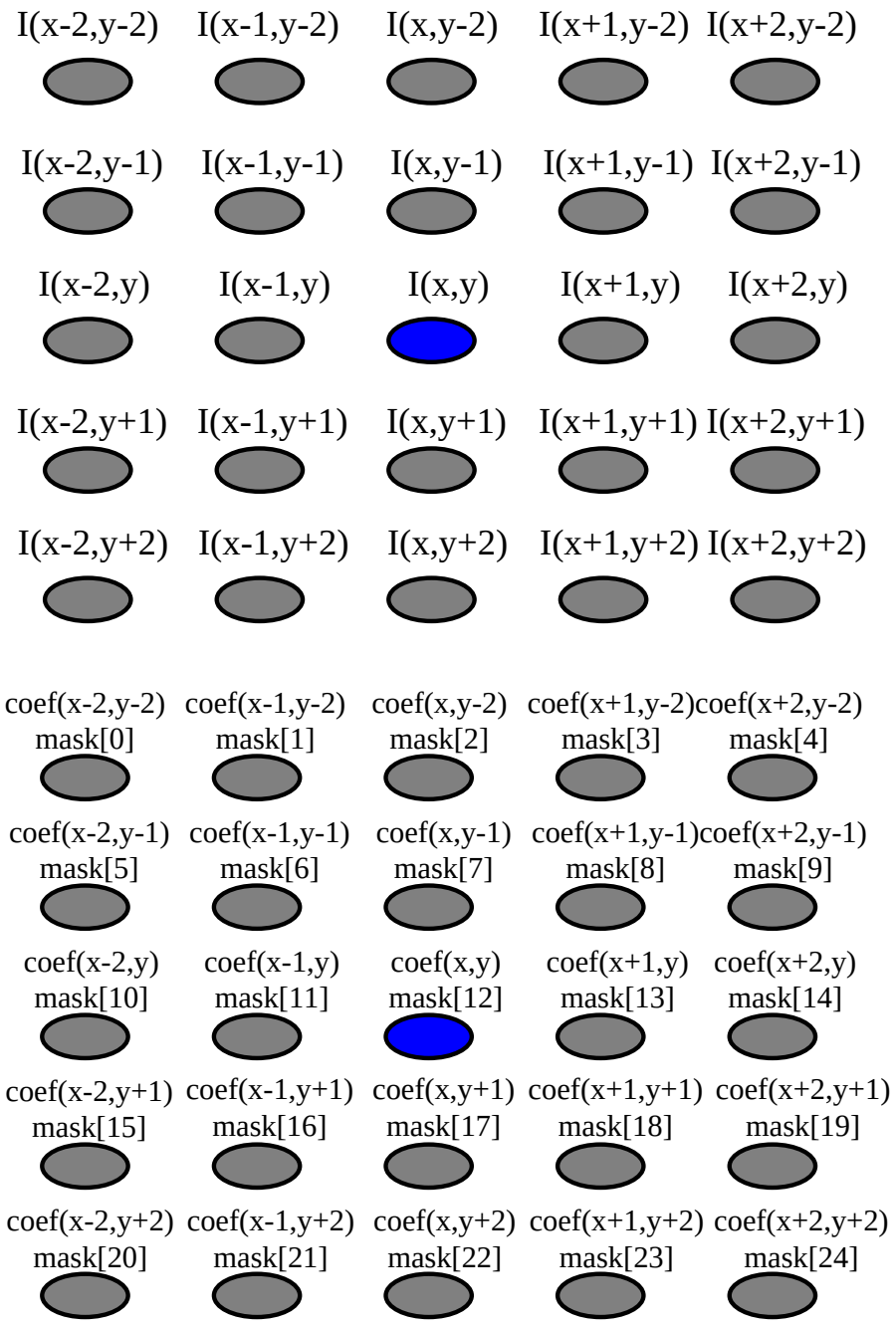
2.33.5 [Requirement]

Header files: **mi_mve.h**

2.33.6 [Note]

- The following output modes are supported:
  – In **MVE_NORM_GRAD_OUT_CTRL_HOR_AND_VER** mode, the **pstDstH** and **pstDstV** pointers cannot be null, and their strides must be the same.
  – In **MVE_NORM_GRAD_OUT_CTRL_HOR** mode, the pstDstH pointer cannot be null.
  – In **MVE_NORM_GRAD_OUT_CTRL_VER** mode, the pstDstV pointer cannot be null.
  – In **MVE_NORM_GRAD_OUT_CTRL_COMBINE** mode, the pstDstHV pointer cannot be null.
- Figure 2-6 shows the NormGard formula.

Figure 2-6 NormGrad formula



$$I_{out}(x,y) = \{ \sum_{-2 \le i \le 2} \sum_{-2 \le j \le 2} I(x+i, y+j) * coef\,(x+i, y+j) \} >> norm$$

2.33.7 [See Also]

MI_MVE_Sobel

2.34. MI_MVE_LBP

2.34.1 [Description]

Creates an LBP calculation task.

2.34.2 [Syntax]

MI_RET MI_MVE_LBP(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstDst, MVE_LBP_CTRL_S *pstLbpCtrl, bool bInstant);

2.34.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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
bInstant	Reserved Flag.	Input

Parameter	Support 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

2.34.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

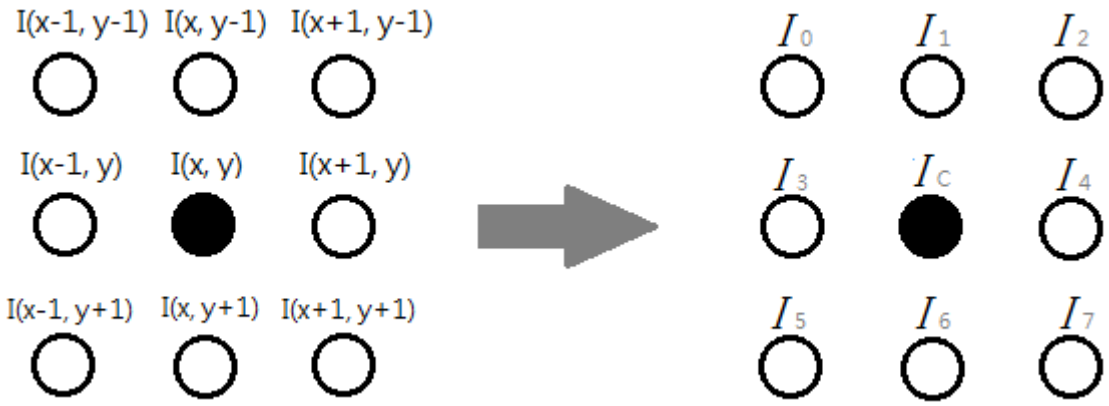
2.34.5 [Requirement]

Header files: **mi_mve.h**

2.34.6 [Note]

Figure 2-7 shows the LBP formula.

Figure 2-7 LBP formula



- MVE_LBP_CMP_NORMAL

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

- MVE_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**, lbp(x, y) corresponds to **pstDst**, and thr corresponds to **pstLbpCtrl→un8BitThr**.

2.34.7 [See Also]

None.

2.35. MI_MVE_GMM

2.35.1 [Description]

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

2.35.2 [Syntax]

MI_RET MI_MVE_GMM(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstFg, MVE_DST_IMAGE_S *pstBg, MVE_MEM_INFO_S *pstModel, MVE_GMM_CTRL_S *pstGmmCtrl, bool bInstant);

2.35.3 [Parameter]

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

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	Reserved Flag .	Input

Parameter	Support Image Type	Address Alignment Mode	Resolution
pstSrc	U8C1 and U8C3_PACKAGE	16 bytes	64 x 64 to 720 x 576
pstFg	U8C1 binary image	16 bytes	64 x 64 to 720 x 576
pstBg	U8C1 and U8C3_PACKAGE	16 bytes	64 x 64 to 720 x 576
pstModel		16 bytes	

2.35.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.35.5 [Requirement]

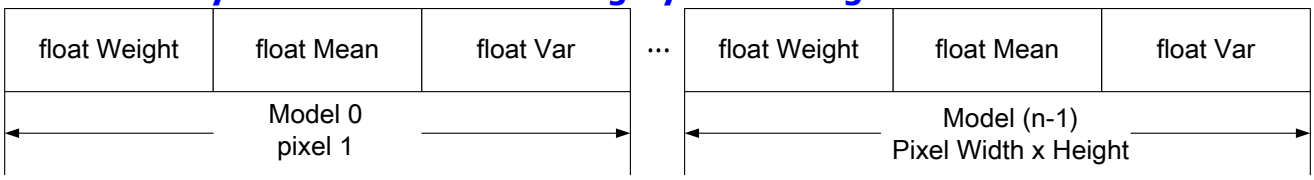
Header files: **mi_mve.h**

2.35.6 [Note]

- GMMs are implemented by referring to MOG and MOG2 in the OpenCV.
- Set u22q10MaxVar and u22q10MinVar of structure MVE_GMM_CTRL_S to nonzero value will run MOG2, otherwise, will run MOG.

- 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 2-8 shows the memory allocation mode of **pstModel**.

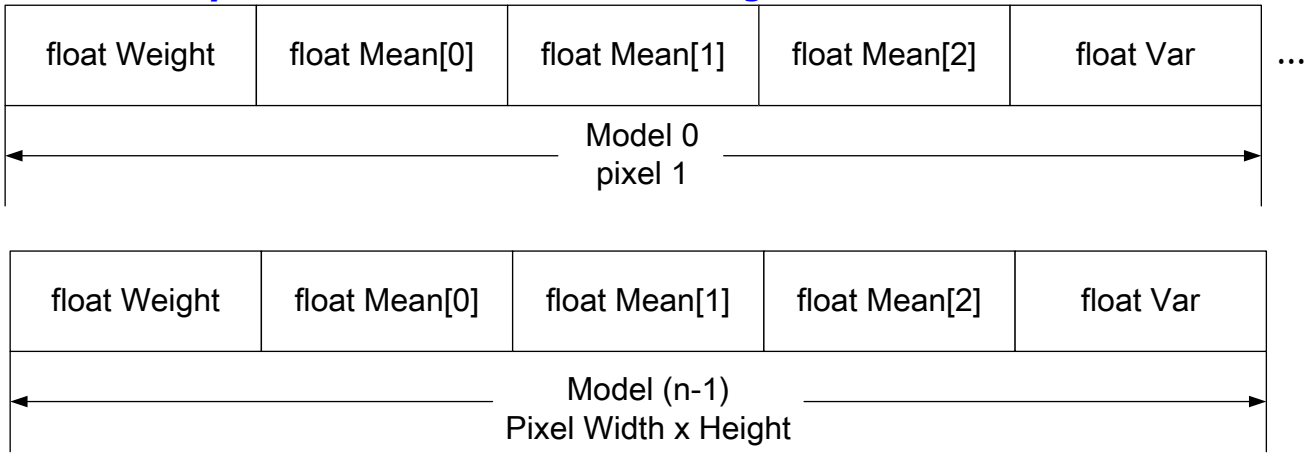
Figure 2-8 Memory allocation of GMMs for gray-scale images



For a GMM of a pixel, each parameter occupies sizeof(float) bytes. The following is the formula of calculating the memory size required by **pstModel**:
pstModel→u32Size = sizeof(float) x 3 x **pstSrc→u16Width** x **pstSrc→u16Height** x **pstGmmCtrl→u8ModeNum**

- Multiple GMMs are used for RGB images. Figure 2-9 shows the memory allocation mode of **pstModel**.

Figure 2-9 Memory allocation of GMMs for RGB images



For a GMM of a pixel, each parameter occupies sizeof(float) bytes. The following is the formula of calculating the memory size required by **pstModel**:
pstModel→u32Size = sizeof(float) x 5 x **pstSrc→u16Width** x **pstSrc→u16Height** x **pstGmmCtrl→u8ModeNum**

2.35.7 [See Also]

2.36. MI_MVE_CannyHysEdge

2.36.1 [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-maxima suppression.

2.36.2 [Syntax]

MI_RET MI_MVE_CannyHysEdge(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_DST_IMAGE_S *pstEdge, MVE_DST_MEM_INFO_S *pstStack, MVE_CANNY_HYS_EDGE_CTRL_S *pstCannyHysEdgeCtrl, bool bInstant);

2.36.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrc	Pointer to the source image It cannot be null.	Input
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. **pstSrc→u16Width** x **pstSrc→ u16Height** x (**sizeof**(**MVE_POINT_U16_S**)) + **sizeof**(**MVE_CANNY_STACK_SIZE_S**)	Output
pstCannyHysEdgeCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag .	Input

Parameter	Support 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	

2.36.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.36.5 [Requirement]

Header files: **mi_mve.h**

2.36.6 [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 of calculating the minimum memory size of **pstCannyHysEdgeCtrl→stMem**:
  pstCannyHysEdgeCtrl→stMem.u32Size = pstSrc→u16Width x **(pstSrc→u16Height +3)** x **4**
- After this task is complete, a Canny edge image is output only when MI_MVE_CannyEdge is called.

2.36.7 [See Also]

- MI_MVE_CannyEdge

2.37. MI_MVE_CannyEdge

2.37.1 [Description]

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

2.37.2 [Syntax]

MI_RET MI_MVE_CannyEdge(MVE_HANDLE pMVEHandle, MVE_IMAGE_S *pstEdge, MVE_MEM_INFO_S *pstStack);

2.37.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstEdge	Pointer to the strong/weak edge flag image as the input or pointer to the edge binary image as the output	Input, output

	It cannot be null.	
pstStack	Coordinate stack of the strong edge point It cannot be null.	Input, output

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

2.37.4 [Return Value]

Return Value	Description
MI_RET_SUCCESS	Success
Other values	Failure. For details, see Error Code

2.37.5 [Requirement]

Header files: **mi_mve.h**

2.37.6 [Note]

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

2.37.7 [See Also]

- MI_MVE_CannyHysEdge

2.38. MI_MVE_LineFilterHor

2.38.1 [Description]

Creates a horizontal density filter task for binary images.

2.38.2 [Syntax]

MI_RET MI_MVE_LineFilterHor(MVE_HANDLE pMVEHandle, MVE_IMAGE_S*pstSrcDst, **MVE_LINE_FILTER_HOR_CTRL_S** *pstLineFilterHorCtrl, bool bInstant);

2.38.3 [Parameter]

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

pstSrcDst	Pointer to the source image or the output after processing It cannot be null.	Input, output
pstLineFilterHorCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag .	Input

Parameter	Supported Image Type	**Address Alignment Mode**	**Resolution**
pstSrcDst	U8C1 binary image	16 bytes	64 x 64 to 1920 x 1080

2.38.4 [Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see Error Code

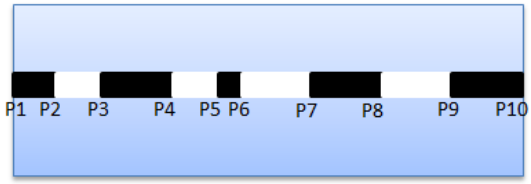
2.38.5 [Requirement]

Header files: **mi_mve.h**

2.38.6 [Note]

The images are counted in the horizontal direction. Each line consists of line segments that appear alternatively (white line segments) and gaps between the line segments (black line segments). The length of a black line segment is satisfied the following condition, the block line segment is set to a white line segment.
The following is the calculation principle:

Step 1 : Horizontal scan the binary image, and record the result, as shown in the figure below.



Step 2: Assume the current gap is line_black56, if conform to the condition as below, the black line segment will be set to a white line segment.

Condition 1 : line_black34 > thr1

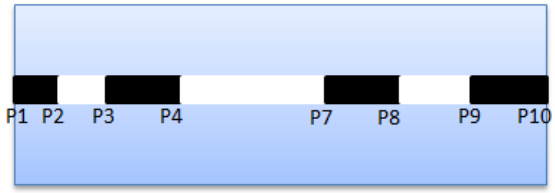
Condition 2 : line_black78 > thr1

Condition 3 :

(line_white45 + line_black56 + line_white67) <= (line_white45 + line_white67) × thr2

Condition 4 : (line_white45 + line_white67) > 1

Step 3 : Following these step (step 1 and step 2), until handle the whole binary image.
The result is shown in figure below.



Step 4 : Horizontal scan the binary image again. According to the relationship between the white line segment and black line segment, transform the black line segment into the white line segment.

Step 5 : Assume the current gap is line_black34, if conform to the condition as below, the black line segment will be set to a white line segment.

Condition 1 : line_block34 < thr3

Condition 2 : line_white47 > 2×thr3

Condition 3 : line_white23 > 9

Condition 4 : line_white23 < 3×thr3

Step 6 : Following these step (step 4 and step 5), until handle the whole binary image.

2.38.7 [Example]

None

2.38.8 [See Also]

MI_MVE_LineFilterVer

2.39. MI_MVE_LineFilterVer

2.39.1 [Description]

Creates a vertical density filter task for binary images.

2.39.2 [Syntax]

MI_RET MI_MVE_LineFilterVer(MVE_HANDLE pMVEHandle, MVE_IMAGE_S*pstSrcDst, **MVE_LINE_FILTER_VER_CTRL** *pstLineFilterVerrCtrl, bool bInstant);

2.39.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrcDst	Pointer to the source image or the output after processing It cannot be null.	Input, output

pstLineFilterVerCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag .	Input

Parameter	Supported Image Type	**Address Alignment Mode**	**Resolution**
pstSrcDst	U8C1 binary image	16 bytes	64 x 64 to 1920 x 1080

2.39.4 [Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see Error Code

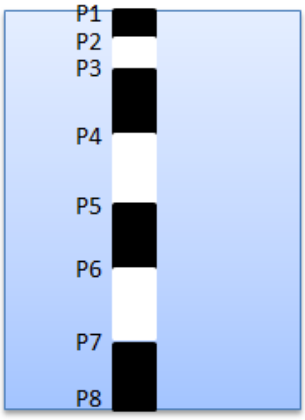
2.39.5 [Requirement]

Header files: **mi_mve.h**

2.39.6 [Note]

The images are counted in the vertical direction. Each line consists of line segments that appear alternatively white line segments and gaps between the line segments (black line segments). The length of a black line segment is satisfied the following condition, the block line segment is set to a white line segment.
The following is the calculation principle:

Step 1 : Vertical scan the binary image, and record the result, as shown in the figure below.



Step 2: Assume the current gap is line_black56, if conform to the condition as below, the black line segment will be set to a white line segment.

Condition 1 : line_black56 < thr

Condition 2 : line_black67 > 6 & line_whitr67 < 25

Condition 3 : line_white45 < 12

Step 3 : Following these step (step 1 and step 2), until handle the whole binary image.
The result is shown in figure below.

2.39.7 [Example]

None

2.39.8 [See Also]

MI_MVE_LineFilterHor

2.40. MI_MVE_NoiseRemoveHor

2.40.1 [Description]

Creates a horizontal noise removal task for binary images.

2.40.2 [Syntax]

MI_RET MI_MVE_NoiseRemoveHor(MVE_HANDLE pMVEHandle, MVE_IMAGE_S *pstSrcDst, MVE_NOISE_REMOVE_HOR_CTRL_S *pstNoiseRemoveHorCtrl, bool bInstant);

2.40.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrcDst	Pointer to the source image or the stored result after processing It cannot be null.	Input, output
pstNoiseRemoveHorCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag .	Input

Parameter	Supported Image Type	**Address Alignment Mode**	**Resolution**
pstSrcDst	U8C1 binary image	16 bytes	64 x 64 to 1920 x 1080

2.40.4 [Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see Error Code

2.40.5 [Requirement]

Header files: **mi_mve.h**

2.40.6 [Note]

The images are counted in the horizontal direction. Each line consists of line segments that appear alternatively (white line segments) and gaps between the line segments (black line segments). The length of a line segment is smaller than the thr1 or bigger than the thr2, the line segment is set to a gap.

2.40.7 [Example]

None

2.40.8 [See Also]

MI_MVE_NoiseRemoveVer

2.41. MI_MVE_NoiseRemoveVer

2.41.1 [Description]

Creates a vertical noise removal task for binary images.

2.41.2 [Syntax]

MI_RET MI_MVE_NoiseRemoveVer(MVE_HANDLE pMVEHandle, MVE_IMAGE_S *pstSrcDst, MVE_NOISE_REMOVE_VER_CTRL_S *pstNoiseRemoveVerCtrl, bool bInstant);

2.41.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrcDst	Pointer to the source image or the stored result after processing It cannot be null.	Input, output
pstNoiseRemoveVerCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag .	Input

Parameter	Supported Image Type	**Address Alignment Mode**	**Resolution**
pstSrcDst	U8C1 binary image	16 bytes	64 x 64 to 1920 x 1080

2.41.4 [Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see Error Code

2.41.5 [Requirement]

Header files: **mi_mve.h**

2.41.6 [Note]

The images are counted in the vertical direction. Each line consists of line segments that appear alternatively (white line segments) and gaps between the line segments (black line segments). The length of a line segment is smaller than the thr1 or bigger than the thr2, the line segment is set to a gap.

2.41.7 [Example]

None

2.41.8 [See Also]

MI_MVE_NoiseRemoveHor

2.42. MI_MVE_AdpThresh

2.42.1 [Description]

Creates an adaptive thresh task.

2.42.2 [Syntax]

MI_RET MI_MVE_AdpThresh(MVE_HANDLE pMVEHandle, MVE_SRC_IMAGE_S *pstSrc, MVE_SRC_IMAGE_S *pstInteg, MVE_DST_IMAGE_S *pstDst, MVE_ADP_THRESH_CTRL_S *pstAdpThrCtrl, bool bInstant);

2.42.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
pstSrc	Pointer to the source image It cannot be null.	Input
pstInteg	Pointer to the integrogram of the source image It cannot be null.	Input
pstDst	Pointer to the target binary image It cannot be null.	Output
pstAdpThrCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag .	Input

Parameter	Supported Image Type	**Address Alignment**	**Resolution**

		Mode	
pstSrc	U8C1	16 bytes	64 x 64 to 1920 x 1080
pstInteg	U32C1	16 bytes	64 x 64 to 1920 x 1080
pstDst	U8C1	16 bytes	64 x 64 to 1920 x 1080

2.42.4 [Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see Error Code

2.42.5 [Requirement]

Header files: **mi_mve.h**

2.42.6 [Note]

The following is the formula:

$$T(x,y)=\frac{1}{w\times h}\times\sum_{i=-h/2}^{h/2}\sum_{j=-w/2}^{w/2}I(x+i,y+j)$$

$$w = u8HalfMaskx$$

$$h = u8HalfMasky$$

$$RateThr = u8RateThr/10$$

$$Offset = s16Offset$$

$$ValueThr = u8ValueThr$$

$$I_{out}(x,y)=\begin{cases}0 & I(x,y)\le(T(x,y)\times RateThr)-Offset \quad \&\ I(x,y)<ValueThr\\ 1 & I(x,y)>(T(x,y)\times RateThr)-Offset \quad \|\ I(x,y)\ge ValueThr\end{cases}$$

I(x, y) corresponds to **pstSrc**, $I_{out}(x,y)$ corresponds to **pstDst**, **W** , **h** , **RateThr** , **0ffset** and **ValueThr** correspond to **u8HalfMaskx** , **u8HalfMasky** , **u8RateThr** , **s16Offset** and **u8ValueThr** in **pstAdpThrCtrl**. The integrogram **pstInteg** is adopted to rapidly calculate the area by using.

$$\sum_{i=-h/2}^{h/2}\sum_{j=-w/2}^{w/2}I(x+i,y+j)$$

2.42.7 [Example]

None

2.42.8 [See Also]

None

2.43. MI_MVE_LKOpticalFlow

2.43.1 [Description]

Create an optical tracking task.

2.43.2 [Syntax]

MI_RET MI_MVE_LKOpticalFlow(MVE_HANDLE pMveHandle, MVE_SRC_IMAGE_S* pstSrcPre, MVE_SRC_IMAGE_S* pstSrcCur, MVE_SRC_MEM_INFO_S *pstPoint, MVE_MEM_INFO_S *pstMv, MVE_LK_OPTICAL_FLOW_CTRL_S *pstLkOptiFlowCtrl, bool bInstant);

2.43.3 [Parameter]

Parameter	Description	Input/Output
pMVEHandle	Pointer to the handle It cannot be null.	Input
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 pstSrc1.	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 MVE_POINT_S25Q7_S, and the minimum memory size is pstLkOptiFlowCtrl→ u16CornerNum x sizeof(MVE_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	Input, Output

	displacement vector type must be MVE_MV_S9Q7_S, and the minimum memory size is pstLkOptiFlowCtrl→ u16CornerNum x sizeof(MVE_MV_S9Q7_S).	
pstLkOptiFlowCtrl	Pointer to the control parameter It cannot be null.	Input
bInstant	Reserved Flag .	Input

Parameter	Supported Image Type	**Address Alignment Mode**	**Resolution**
pstSrc1	U8C1	16 bytes	64 x 64 to 1920 x 1080
pstSrc2	U8C1	16 bytes	64 x 64 to 1920 x 1080

2.43.4 [Return Value]

Return Value	Description
0	Success
Other values	Failure. For details, see Error Code

2.43.5 [Requirement]

Header files: **mi_mve.h**

2.43.6 [Note]

1. Prepare initial corner points
2. input image must be sequential.
   Eg: MI_MVE_LKOpticalFlow(img1, img2)
   MI_MVE_LKOpticalFlow(img2, img3)
   MI_MVE_LKOpticalFlow(img3, img4)
   MI_MVE_LKOpticalFlow(img4, img5)

2.43.7 [Example]

```
MI_RET ret;
MVE_HANDLE handle;
MVE_SRC_IMAGE_S src1, src2;
MVE_IMG_INFO info = {(S16)InputWidth, (S16)InputHeight};
MVE_SRC_MEM_INFO_S Point;
MVE_MEM_INFO_S Mv;
MVE_LK_OPTICAL_FLOW_CTRL_S ctrl;

// Init MVE
MI_MVE_Init(&info);
```

```
// control setting
ctrl.u16CornerNum = 49;
ctrl.u0q8MinEigThr = 255;
ctrl.u8IterCount = 10;
ctrl.u0q8Epsilon = 26;
ctrl.u16PyrLevel = 15;

// prepare point & mv buffers
Point.u32Size        = sizeof(MVE_POINT_S25Q7_S) * ctrl.u16CornerNum;
Point.pu8VirAddr     = (U8*)malloc( Point.u32Size );
Point.pu32PhyAddr    = Point.pu8VirAddr;
Mv.u32Size           = sizeof(MVE_MV_S9Q7_S) * ctrl.u16CornerNum;
Mv.pu8VirAddr        = (U8*)malloc( Mv.u32Size );
Mv.pu32PhyAddr       = Mv.pu8VirAddr;

// init point
MVE_POINT_S25Q7_S* pP = (MVE_POINT_S25Q7_S*)(Point.pu8VirAddr);
pP = (MVE_POINT_S25Q7_S*)(Point.pu8VirAddr);
for (i=0; i<(S32)(Point.u32Size); i++)
    Point.pu8VirAddr[i] = 0;   // init
data_count = 0;
for (i=0; i<7; i++)   // X
    for (j=0; j<7; j++) {   // Y
        pP[data_count].s25q7X =((src1.u16Width / 13)*(i+3)) << 7;
        pP[data_count].s25q7Y = ((src1.u16Height / 13)*(j+3)) << 7;
        data_count++;
    }

// init MV=0
MVE_MV_S9Q7_S* pMv = (MVE_MV_S9Q7_S*)(Mv.pu8VirAddr);
pMv = (MVE_MV_S9Q7_S*)(Mv.pu8VirAddr);
for (i=0; i<ctrl.u16CornerNum; i++) {
    pMv[i].s32Status = 0;
    pMv[i].s9q7Dx = 0;
    pMv[i].s9q7Dy = 0;
}

// Process loop
for (tt=0; tt<frame_count; tt++) {
    // prepare input buffer content (src1, src2)

    // update points
    pP = (MVE_POINT_S25Q7_S*)(Point.pu8VirAddr);
```

```
        pMv = (MVE_MV_S9Q7_S*)(Mv.pu8VirAddr);
        for (i=0; i<ctrl.u16CornerNum; i++) {
            if ( pMv[i].s32Status == 0 ) {
                pP[i].s25q7X += pMv[i].s9q7Dx;   // set
                pP[i].s25q7Y += pMv[i].s9q7Dy;   // set
            } else {
                pP[i].s25q7X = 0;   // set
                pP[i].s25q7Y = 0;   // set
            }
        }

        ret = MI_MVE_LKOpticalFlow( handle,
                                    &src1,
                                    &src2,
                                    &Point,
                                    &Mv,
                                    &ctrl,
                                    (bool)0 );
    }
```

2.43.8 [See Also]

None

3.DATA STRUCTURES

3.1. MVE data structures list

MVE function provides the following data structures:

MVE_MAP_NUM: Defines the number of map lookup entries.

MVE_IMAGE_TYPE_E: Defines the type of supported generalized 2D images.

MVE_MEM_ALLOC_TYPE_E: Defines the type of memory allocation.

MVE_CSC_MODE_E: Defines the CSC mode.

MVE_SOBEL_OUT_CTRL_E: Define Sobel output control information.

MVE_THRESH_MODE_E : Defines the thresh output format.

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

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

MVE_ORD_STAT_FILTER_MODE_E: Defines the order statistics filter mode.

MVE_BERNSEN_MODE_E: Defines the Bernsen thresh mode.

MVE_THRESH_S16_MODE_E: Defines the threshold mode of 16-bit signed images.

MVE_THRESH_U16_MODE_E: Defines the threshold mode of 16-bit signed images.

MVE_16BIT_TO_8BIT_MODE_E: Defines the threshold mode of 16-bit signed images.

MVE_SAD_OUT_CTRL_E: Defines the SAD output control information.

MVE_SAD_MODE_E: Defines SAD block size.

MVE_IMAGE_S: Defines the information about generalized 2D images.
MVE_SRC_IMAGE_S: Defines the source images.
MVE_DST_IMAGE_S: Defines the output images.
MVE_FILTER_CTRL_S: Defines the control information about template filter.
MVE_CSC_CTRL_S: Defines CSC control information.
MVE_FILTER_AND_CSC_CTRL_S: Defines the control information about template filter and CSC.
MVE_SOBEL_CTRL_S : Defines the control information about Sobel edge extraction.
MVE_DILATE_CTRL_S : Defines dilate control information.
MVE_ERODE_CTRL_S : Defines erode control information.
MVE_THRESH_CTRL_S: Defines thresh control information.
MVE_THRESH_S16_CTRL_S: Defines the threshold control parameters of 16-bit signed images.
MVE_THRESH_U16_CTRL_S: Defines the threshold control parameters of 16-bit unsigned images.
MVE_16BIT_TO_8BIT_CTRL_S: Defines the control parameters for converting a 16-bit image to an 8-bit image.

MVE_MAG_AND_ANG_CTRL_S: Defines the control information about the Canny edge magnitude and argument calculation.
MVE_SUB_CTRL_S: Defines the control information about the subtraction operation between two images.
MVE_ADD_CTRL_S: Defines weighted addition control parameters for two images.
MVE_ORD_STAT_FILTER_CTRL_S: Defines the control parameter for order statistics filter.
MVE_BERNSEN_CTRL_S: Defines Bernsen thresh control parameters.
MVE_MEM_INFO_S: Defines the information about the memory for storing 1D data.
MVE_SRC_MEM_INFO_S: Defines 1D source data.
MVE_DST_MEM_INFO_S: Defines 1D destination data.
MVE_MAP_LUT_MEM_S: Defines the information about the lookup table memory of the map operator.
MVE_NCC_DST_MEM_S: Defines the information about the NCC output memory.
MVE_HANDLE: Defines MVE handle.
MVE_INTEG_CTRL_S: Defines the integrogram calculation control parameter.
MVE_INTEG_OUT_CTRL_E: Defines integrogram output control parameters.
MVE_CCBLOB_S: Defines CCL output information.
MVE_CCL_CTRL_S: Defines CCL control parameters.
MVE_REGION_S: Defines connected component information.
MVE_HIST_NUM: Defines the number of bins in a histogram.
MVE_HIST_TABLE_MEM_S: Defines the information about the lookup table memory of the

hist operator.

MVE_EQUALIZE_HIST_CTRL_S: Defines the histogram equalization control parameter.

MVE_EQUALIZE_HIST_CTRL_MEM_S: Defines the histogram equalization auxiliary memory.

MVE_SAD_CTRL_S: Defines SAD control parameters.

MVE_NORM_GRAD_CTRL_S: Defines control parameters for the normalized gradient calculation.

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

MVE_LBP_CTRL_S: Defines LBP texture calculation control parameters.

MVE_LBP_CMP_MODE_E: Defines the comparison mode during LBP calculation.

MVE_LOOK_UP_TABLE_S: Defines the lookup table.

MVE_GMM_CTRL_S: Defines the control parameters for GMM background modeling.

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

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

MVE_LINE_FILTER_HOR_CTRL_S: Defines control parameters for filtering the horizontal density of binary images.

MVE_LINE_FILTER_VER_CTRL_S: Defines control parameters for filtering the vertical density of binary images.

MVE_NOISE_REMOVE_HOR_CTRL_S: Defines the horizontal noise removal control parameter for the binary image.

MVE_NOISE_REMOVE_VER_CTRL_S: Defines the vertical noise removal control parameter for the binary image.

MVE_ADP_THRESH_CTRL_S: Defines adaptive thresh control parameters.

MVE_LK_OPTICAL_FLOW_CTRL_S: Define optical flow tracking control parameters.

MVE_POINT_S25Q7_S: Define point position.

MVE_MV_S9Q7_S: Define moving distance.

3.2. MVE_MAP_NUM

3.2.1 [Description]

Defines the number of map lookup entries.

3.2.2 [Syntax]

```
#define MVE_MAP_NUM        256
```

3.2.3 [Member]

- None

3.2.4 [Note]

- None

3.2.5 [See Also]

- None

3.3. MVE_IMAGE_TYPE_E

3.3.1 [Description]

Defines the type of supported generalized 2D images.

3.3.2 [Syntax]

```
typedef enum _MVE_IMAGE_TYPE_E
{
    MVE_IMAGE_TYPE_U8C1             = 0x0,
    MVE_IMAGE_TYPE_S8C1             = 0x1,
    MVE_IMAGE_TYPE_YUV420SP         = 0x2,
    MVE_IMAGE_TYPE_YUV422SP         = 0x3,
    MVE_IMAGE_TYPE_YUV420P          = 0x4,
    MVE_IMAGE_TYPE_YUV422P          = 0x5,

    MVE_IMAGE_TYPE_S8C2_PACKAGE     = 0x6,
    MVE_IMAGE_TYPE_S8C2_PLANAR      = 0x7,

    MVE_IMAGE_TYPE_S16C1            = 0x8,
    MVE_IMAGE_TYPE_U16C1            = 0x9,

    MVE_IMAGE_TYPE_U8C3_PACKAGE     = 0xa,
    MVE_IMAGE_TYPE_U8C3_PLANAR      = 0xb,
```

```
    MVE_IMAGE_TYPE_S32C1        = 0xc,
    MVE_IMAGE_TYPE_U32C1        = 0xd,

    MVE_IMAGE_TYPE_S64C1        = 0xe,
    MVE_IMAGE_TYPE_U64C1        = 0xf,

    MVE_IMAGE_TYPE_BUTT

} MVE_IMAGE_TYPE_E;
```

3.3.3 [Member]

Member	Description
MVE_IMAGE_TYPE_U8C1	Single-channel image of which each pixel is expressed by an 8-bit unsigned data segment.
MVE_IMAGE_TYPE_S8C1	Single-channel image of which each pixel is expressed by an 8-bit signed data segment.
MVE_IMAGE_TYPE_YUV420SP	YUV420 semi-planar image.
MVE_IMAGE_TYPE_YUV422SP	YUV422 semi-planar image.
MVE_IMAGE_TYPE_Y UV420P	YUV420 planar image.
MVE_IMAGE_TYPE_YUV422P	YUV422 planar image.
MVE_IMAGE_TYPE_S8C2_PACKAGE	Dual-channel image (stored in package format) of which each pixel is expressed by two 8-bit signed data segments.
MVE_IMAGE_TYPE_S8C2_PLANAR	Dual-channel image (stored in planar format) of which each pixel is expressed by two 8-bit signed data segments.
MVE_IMAGE_TYPE_S16C1	Single-channel image of which each pixel is expressed by a 16-bit signed data segment.
MVE_IMAGE_TYPE_U16C1	Single-channel image of which each pixel is expressed by a 16-bit unsigned data segment.
MVE_IMAGE_TYPE_U8C3_PACKAGE	Three-channel image (stored in package format) of which each pixel is expressed by three 8-bit unsigned data segments.
MVE_IMAGE_TYPE_U8C3_PLANAR	Three-channel image (stored in planar format) of which each pixel is expressed by three 8-bit unsigned data segments.
MVE_IMAGE_TYPE_S32C1	Single-channel image of which each pixel is expressed by a 32-bit signed data segment.
MVE_IMAGE_TYPE_U32C1	Single-channel image of which each pixel is expressed by a 32-bit unsigned data segment.

MVE_IMAGE_TYPE_S64C1	Single-channel image of which each pixel is expressed by a 64-bit signed data segment.
MVE_IMAGE_TYPE_U64C1	Single-channel image of which each pixel is expressed by a 64-bit unsigned data segment.

3.3.4 [Note]

- None

3.3.5 [See Also]

- **MVE_IMAGE_S**
- MVE_SRC_IMAGE_S
- MVE_DST_IMAGE_S

3.4. MVE_MEM_ALLOC_TYPE_E

3.4.1 [Description]

Defines the memory allocation type.

3.4.2 [Syntax]

```
typedef enum _MVE_MEM_ALLOC_TYPE_E
{
    MVE_MEM_ALLOC_TYPE_VIRTUAL          = 0x0,
    MVE_MEM_ALLOC_TYPE_PHYSICAL         = 0x1
} MVE_MEM_ALLOC_TYPE_E;
```

3.4.3 [Member]

Member	Description
MVE_MEM_ALLOC_TYPE_VIRTUAL	Normal virtual memory
MVE_MEM_ALLOC_TYPE_PHYSICAL	Physically continuous memory

3.4.4 [Note]

Allocation via MVE_MEM_ALLOC_TYPE_PHYSICAL is used for HW accelerator.

3.4.5 [See Also]

3.5. MVE_CSC_MODE_E

3.5.1 [Description]

Defines the CSC mode.

3.5.2 [Syntax]

```
typedef enum _MVE_CSC_MODE_E
{
  MVE_CSC_MODE_PIC_BT601_YUV2RGB    = 0x0,
  MVE_CSC_MODE_PIC_BT601_RGB2YUV    = 0x1,

  MVE_CSC_MODE_BUTT
} MVE_CSC_MODE_E;
```

3.5.3 [Member]

Member	Description
MVE_CSC_MODE_PIC_BT601_YUV2RGB	BT601 YUV2RGB image conversion
MVE_CSC_MODE_PIC_BT601_RGB2YUV	BT601 RGB2YUV image conversion

3.5.4 [Note]

- In MVE_CSC_MODE_PIC_BT601_YUV2RGB and mode, the output format must meet the following condition: 0 ≤ R, G, B ≤ 255 pixels.
- In MVE_CSC_MODE_PIC_BT601_RGB2YUV mode, the output format must meet the following condition: 0 ≤ Y ≤ 235 pixels, 0 ≤ U, V ≤ 240 pixels.

3.5.5 [See Also]

- **MVE_CSC_CTRL_S**
- **MVE_FILTER_AND_CSC_CTRL_S**

3.6. MVE_SOBEL_OUT_CTRL_E

3.6.1 [Description]

Defines Sobel output control information.

3.6.2 [Syntax]

```
typedef enum _MVE_SOBEL_OUT_CTRL_E
{
  MVE_SOBEL_OUT_CTRL_BOTH = 0x0,
  MVE_SOBEL_OUT_CTRL_HOR  = 0x1,
  MVE_SOBEL_OUT_CTRL_VER  = 0x2,

  MVE_SOBEL_OUT_CTRL_BUTT
} MVE_SOBEL_OUT_CTRL_E;
```

3.6.3 [Member]

Member	Description
MVE_SOBEL_OUT_CTRL_BOTH	Output results after filtering by using the

	common template and transposed template
MVE_SOBEL_OUT_CTRL_HOR	Output results after filtering by using only the common template
MVE_SOBEL_OUT_CTRL_VER	Output results after filtering by using only the transposed template

3.6.4 [Note]

- None

3.6.5 [See Also]

- **MVE_SOBEL_CTRL_S**

3.7. MVE_THRESH_MODE_E

3.7.1 [Description]

Defines the thresh output format.

3.7.2 [Syntax]

```
typedef enum _MVE_THRESH_MODE_E
{
    MVE_THRESH_MODE_BINARY = 0x0,
    MVE_THRESH_MODE_TRUNC = 0x1,
    MVE_THRESH_MODE_TO_MINVAL = 0x2,
    MVE_THRESH_MODE_MIN_MID_MAX = 0x3,
    MVE_THRESH_MODE_ORI_MID_MAX = 0x4,
    MVE_THRESH_MODE_MIN_MID_ORI = 0x5,
    MVE_THRESH_MODE_MIN_ORI_MAX = 0x6,
    MVE_THRESH_MODE_ORI_MID_ORI = 0x7,

    MVE_THRESH_MODE_BUTT
} MVE_THRESH_MODE_E;
```

3.7.3 [Member]

Member	Description
MVE_THRESH_MODE_BINARY	srcVal ≤ lowThr, dstVal = minVal srcVal > lowThr, dstVal = maxVal
MVE_THRESH_MODE_TRUNC	srcVal ≤ lowThr, dstVal = srcVal srcVal > lowThr, dstVal = maxVal
MVE_THRESH_MODE_TO_MINVAL	srcVal ≤ lowThr, dstVal = minVal srcVal > lowThr, dstVal = srcVal
MVE_THRESH_MODE_MIN_MID_MAX	srcVal ≤ lowThr, dstVal = minVal

	lowThr < srcVal ≤ highThr, dstVal = midVal srcVal > highThr, dstVal = maxVal
MVE_THRESH_MODE_ORI_MID_MAX	srcVal ≤ lowThr, dstVal = srcVal lowThr < srcVal ≤ highThr, dstVal = midVal srcVal > highThr, dstVal = maxVal
MVE_THRESH_MODE_MIN_MID_ORI	srcVal ≤ lowThr, dstVal = minVal lowThr < srcVal ≤ highThr, dstVal = midVal srcVal > highThr, dstVal = srcVal
MVE_THRESH_MODE_MIN_ORI_MAX	srcVal ≤ lowThr, dstVal = minVal lowThr < srcVal ≤ highThr, dstVal = srcVal srcVal > highThr, dstVal = maxVal
MVE_THRESH_MODE_ORI_MID_ORI	srcVal ≤ lowThr, dstVal = srcVal srcVal > highThr, dstVal = srcVal lowThr < srcVal ≤ highThr , dstVal = midVal

3.7.4 [Note]

- None

3.7.5 [See Also]

- **MVE_THRESH_CTRL_S**

3.8. MVE_MAG_AND_ANG_OUT_CTRL_E

3.8.1 [Description]

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

3.8.2 [Syntax]

```
typedef enum _MVE_MAG_AND_ANG_OUT_CTRL_E
{
    MVE_MAG_AND_ANG_OUT_CTRL_MAG              = 0x0,
    MVE_MAG_AND_ANG_OUT_CTRL_MAG_AND_ANG      = 0x1,

    MVE_MAG_AND_ANG_OUT_CTRL_BUTT
} MVE_MAG_AND_ANG_OUT_CTRL_E;
```

3.8.3 [Member]

Member	Description
MVE_MAG_AND_ANG_OUT_CTRL_MAG	Output only magnitude
MVE_MAG_AND_ANG_OUT_CTRL_MAG_AND_ANG	Output both magnitude and angle

3.8.4 [Note]

- None

3.8.5 [See Also]

- **MVE_MAG_AND_ANG_CTRL_S**

3.9. MVE_SUB_MODE_E

3.9.1 [Description]

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

3.9.2 [Syntax]

```
typedef enum _MVE_SUB_MODE_E
{
    MVE_SUB_MODE_ABS = 0x0,
    MVE_SUB_MODE_SHIFT = 0x1,

    MVE_SUB_MODE_BUTT
} MVE_SUB_MODE_E;
```

3.9.3 [Member]

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

3.9.4 [Note]

- None

3.9.5 [See Also]

- **MVE_SUB_CTRL_S**

3.10. MVE_ORD_STAT_FILTER_MODE_E

3.10.1 [Description]

Defines the order statistics filter mode.

3.10.2 [Syntax]

```
typedef enum _MVE_ORD_STAT_FILTER_MODE_E
{
```

```
    MVE_ORD_STAT_FILTER_MODE_MEDIAN   = 0x0,
    MVE_ORD_STAT_FILTER_MODE_MAX       = 0x1,
    MVE_ORD_STAT_FILTER_MODE_MIN       = 0x2,

    MVE_ORD_STAT_FILTER_MODE_BUTT
} MVE_ORD_STAT_FILTER_MODE_E;
```

3.10.3 [Member]

Member	Description
MVE_ORD_STAT_FILTER_MODE_MEDIAN	Median filtering
MVE_ORD_STAT_FILTER_MODE_MAX	Maximum filtering, equivalent to the dilate task for the gray-scale images.
MVE_ORD_STAT_FILTER_MODE_MIN	Minimum filtering, equivalent to the erode task for the gray-scale images

3.10.4 [Note]

- None

3.10.5 [See Also]

- **MVE_ORD_STAT_FILTER_CTRL_S**

3.11. MVE_BERNSEN_MODE_E

3.11.1 [Description]

Defines the Bernsen thresh mode.

3.11.2 [Syntax]

```
typedef enum _MVE_BERNSEN_MODE_E
{
    MVE_BERNSEN_MODE_NORMAL = 0x0,
    MVE_BERNSEN_MODE_THRESH = 0x1,

    MVE_BERNSEN_MODE_BUTT
} MVE_BERNSEN_MODE_E;
```

3.11.3 [Member]

Member	Description
MVE_BERNSEN_MODE_NORMAL	Simple Bernsen thresh
MVE_BERNSEN_MODE_THRESH	Thresh based on the global threshold and local Bernsen threshold

3.11.4 [Note]

- None

3.11.5 [See Also]

- **MVE_BERNSEN_CTRL_S**

3.12. MVE_THRESH_S16_MODE_E

3.12.1 [Description]

Defines the threshold mode of 16-bit signed images.

3.12.2 [Syntax]

```
typedef enum _MVE_THRESH_S16_MODE_E
{
    MVE_THRESH_S16_MODE_S16_TO_S8_MIN_MID_MAX = 0x0,
    MVE_THRESH_S16_MODE_S16_TO_S8_MIN_ORI_MAX = 0x1,
    MVE_THRESH_S16_MODE_S16_TO_U8_MIN_MID_MAX = 0x2,
    MVE_THRESH_S16_MODE_S16_TO_U8_MIN_ORI_MAX = 0x3,

    MVE_THRESH_S16_MODE_BUTT
} MVE_THRESH_S16_MODE_E;
```

3.12.3 [Member]

Member	Description
MVE_THRESH_S16_MODE_S16_TO_S8_MIN_MID_MAX	srcVal ≤ lowThr dstVal = minVal lowThr < srcVal ≤ highThr, dstVal = midVal srcVal > highThr dstVal = maxVal
MVE_THRESH_S16_MODE_S16_TO_S8_MIN_ORI_MAX	srcVal ≤ lowThr dstVal = minVal lowThr < srcVal ≤ highThr dstVal = srcVal srcVal > highThr dstVal = maxVal
MVE_THRESH_S16_MODE_S16_TO_U8_MIN_MID_MAX	srcVal ≤ lowThr dstVal = minVal lowThr < srcVal ≤ highThr dstVal = midVal srcVal > highThr

	dstVal = maxVal
MVE_THRESH_S16_MODE_S16_TO_U8_MIN_ORI_MAX	srcVal ≤ lowThr dstVal = minVal lowThr < srcVal ≤ highThr dstVal = srcVal srcVal > highThr dstVal = maxVal

3.12.4 [Note]

- For details about the related formula, see the Note field of MI_MVE_Thresh_S16

3.12.5 [See Also]

- **MVE_THRESH_S16_CTRL_S**

3.13. MVE_THRESH_U16_MODE_E

3.13.1 [Description]

Defines the threshold mode of 16-bit unsigned images.

3.13.2 [Syntax]

```
typedef enum _MVE_THRESH_U16_MODE_E
{
    MVE_THRESH_U16_MODE_U16_TO_U8_MIN_MID_MAX = 0x0,
    MVE_THRESH_U16_MODE_U16_TO_U8_MIN_ORI_MAX = 0x1,

    MVE_THRESH_U16_MODE_BUTT
} MVE_THRESH_U16_MODE_E;
```

3.13.3 [Member]

Member	Description
MVE_THRESH_U16_MODE_U16_TO_U8_MIN_MID_MAX	srcVal ≤ lowThr dstVal = minVal lowThr < srcVal ≤ highThr dstVal = midVal srcVal > highThr dstVal = maxVal
MVE_THRESH_U16_MODE_U16_TO_U8_MIN_ORI_MAX	srcVal ≤ lowThr dstVal = minVal lowThr < srcVal ≤ highThr dstVal = srcVal srcVal > highThr

	dstVal = maxVal

3.13.4 [Note]

- For details about the related formula, see the Note field of MI_MVE_Thresh_U16

3.13.5 [See Also]

- **MVE_THRESH_U16_CTRL_S**

3.14. MVE_16BIT_TO_8BIT_MODE_E

3.14.1 [Description]

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

3.14.2 [Syntax]

```
typedef enum _MVE_16BIT_TO_8BIT_MODE_E
{
    MVE_16BIT_TO_8BIT_MODE_S16_TO_S8 = 0x0,
    MVE_16BIT_TO_8BIT_MODE_S16_TO_U8_ABS = 0x1,
    MVE_16BIT_TO_8BIT_MODE_S16_TO_U8_BIAS = 0x2,
    MVE_16BIT_TO_8BIT_MODE_U16_TO_U8 = 0x3,

    MVE_16BIT_TO_8BIT_MODE_BUTT
} MVE_16BIT_TO_8BIT_MODE_E;
```

3.14.3 [Member]

Member	Description
MVE_16BIT_TO_8BIT_MODE_S16_TO_S8	Linear transformation from S16 data to S8 data
MVE_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
MVE_16BIT_TO_8BIT_MODE_S16_TO_U8_BIAS	U8 data obtained after linear transformation from S16 data to S8 data, translation, and then truncation
MVE_16BIT_TO_8BIT_MODE_U16_TO_U8	Linear transformation from S16 data to U8 data

3.14.4 [Note]

- For details about the related formula, see the Note field of MI_MVE_16BitTo8Bit

3.14.5 [See Also]

- MVE_16BIT_TO_8BIT_CTRL_S

3.15. MVE_IMAGE_S

3.15.1 [Description]

Defines the information about generalized 2D images.

3.15.2 [Syntax]

```
typedef struct _MVE_IMAGE_S
{
    MVE_MAP_NUM
```

[Description]

Defines the number of map lookup entries.

3.15.3 [Syntax]

```
#define MVE_MAP_NUM        256
```

3.15.4 [Member]

- None

3.15.5 [Note]

- None

3.15.6 [See Also]

- None

```
    MVE_IMAGE_TYPE_E   enType;
    void *u32PhyAddr[3];
    U8 *pu8Addr[3];
    U16 u16Stride[3];
    U16 u16Width;
    U16 u16Height;
    U16 u16Reserved;
} MVE_IMAGE_S;
```

3.15.7 [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
u16Width	Generalized image width
u16Height	Generalized image height
u16Reserved	Reserved

3.15.8 [Note]

- The unit of **u16Width**, **u16Height**, and **u16Stride** is pixel.
- For details about each type of image, see Figure 1-2 to Figure 1-10.

3.15.9 [See Also]

- MVE_IMAGE_TYPE_E
- MVE_SRC_IMAGE_S
- MVE_DST_IMAGE_S

3.16. MVE_SRC_IMAGE_S

3.16.1 [Description]

Defines the source image.

3.16.2 [Syntax]

```
typedef MVE_IMAGE_S MVE_SRC_IMAGE_S;
```

3.16.3 [Member]

- None

3.16.4 [Note]

- None

3.16.5 [See Also]

- **MVE_IMAGE_S**
- MVE_DST_IMAGE_S

3.17. MVE_DST_IMAGE_S

3.17.1 [Description]

Defines the output image.

3.17.2 [Syntax]

```
typedef MVE_IMAGE_S MVE_DST_IMAGE_S;
```

3.17.3 [Member]

- None

3.17.4 [Note]

- None

3.17.5 [See Also]

- **MVE_IMAGE_S**
- MVE_SRC_IMAGE_S

3.18. MVE_FILTER_CTRL_S

3.18.1 [Description]

Defines the control information about template filter.

3.18.2 [Syntax]

```
typedef struct _MVE_FILTER_CTRL_S
{
    S8 as8Mask[25];
    U8 u8Norm;
} MVE_FILTER_CTRL_S;
```

3.18.3 [Member]

Member	Description
as8Mask[25]	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]

3.18.4 [Note]

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

3.18.5 [See Also]

- None

3.19. MVE_CSC_CTRL_S

3.19.1 [Description]

Defines CSC control information.

3.19.2 [Syntax]

```
typedef struct _MVE_CSC_CTRL_S
{
    MVE_CSC_MODE_E      enMode;
} MVE_CSC_CTRL_S;
```

3.19.3 [Member]

Member	Description
enMode	Working mode

3.19.4 [Note]

- None

3.19.5 [See Also]

- MVE_CSC_MODE_E

3.20. MVE_FILTER_AND_CSC_CTRL_S

3.20.1 [Description]

Defines the control information about template filter and CSC.

3.20.2 [Syntax]

```
typedef struct _MVE_FILTER_AND_CSC_CTRL_S
{
    MVE_CSC_MODE_E   enMode;
    S8   as8Mask[25];
    U8   u8Norm;
} MVE_FILTER_AND_CSC_CTRL_S;
```

3.20.3 [Member]

Member	Description
enMode	Working mode
as8Mask[25]	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]

3.20.4 [Note]

- Only the YUV2RGB modes are supported.

3.20.5 [See Also]

- MVE_CSC_MODE_E

3.21. MVE_SOBEL_CTRL_S

3.21.1 [Description]

Defines the control information about Sobel edge extraction.

3.21.2 [Syntax]

```
typedef struct _MVE_SOBEL_CTRL_S
{
    MVE_SOBEL_OUT_CTRL_E enOutCtrl;
    S8 as8Mask[25];
} MVE_SOBEL_CTRL_S;
```

3.21.3 [Member]

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

3.21.4 [Note]

- None

3.21.5 [See Also]

- MVE_SOBEL_OUT_CTRL_E

3.22. MVE_DILATE_CTRL_S

3.22.1 [Description]

Defines dilate control information.

3.22.2 [Syntax]

```
typedef struct _MVE_DILATE_CTRL_S
{
    U8 au8Mask[25];
} MVE_DILATE_CTRL_S;
```

3.22.3 [Member]

Member	Description

as8Mask[25]	5x5 template coefficient Value: 0 or 255

3.22.4 [Note]

- None

3.22.5 [See Also]

- None

3.23. MVE_ERODE_CTRL_S

3.23.1 [Description]

Defines erode control information.

3.23.2 [Syntax]

```
typedef struct _MVE_ERODE_CTRL_S
{
    U8 au8Mask[25];
} MVE_ERODE_CTRL_S;
```

3.23.3 [Member]

Member	Description
as8Mask[25]	5x5 template coefficient Value: 0 or 255

3.23.4 [Note]

- None

3.23.5 [See Also]

- None

3.24. MVE_THRESH_CTRL_S

3.24.1 [Description]

Defines erode control information.

3.24.2 [Syntax]

```
typedef struct _MVE_THRESH_CTRL_S
{
    MVE_THRESH_MODE_E enMode;
    U8 u8LowThr;
```

```
    U8 u8HighThr;
    U8 u8MinVal;
    U8 u8MidVal;
    U8 u8MaxVal;
} MVE_THRESH_CTRL_S;
```

3.24.3 [Member]

Member	Description
enMode	Working 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]

3.24.4 [Note]

- None

3.24.5 [See Also]

- MVE_THRESH_MODE_E

3.25. MVE_THRESH_S16_CTRL_S

3.25.1 [Description]

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

3.25.2 [Syntax]

```
typedef struct _MVE_THRESH_S16_CTRL_S
{
    MVE_THRESH_S16_MODE_E enMode;
    S16 s16LowThr;
    S16 s16HighThr;
    MVE_8BIT_U un8MinVal;
    MVE_8BIT_U un8MidVal;
    MVE_8BIT_U un8MaxVal;
```

```
} MVE_THRESH_S16_CTRL_S;
```

3.25.3 [Member]

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

3.25.4 [Note]

- For details about the related formula, see the Note field of MI_MVE_Thresh_S16

3.25.5 [See Also]

- MVE_THRESH_S16_MODE_E

3.26. MVE_THRESH_U16_CTRL_S

3.26.1 [Description]

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

3.26.2 [Syntax]

```
typedef struct _MVE_THRESH_U16_CTRL_S
{
    MVE_THRESH_U16_MODE_E enMode;
    U16 u16LowThr;
    U16 u16HighThr;
    U8 u8MinVal;
    U8 u8MidVal;
    U8 u8MaxVal;
} MVE_THRESH_U16_CTRL_S;
```

3.26.3 [Member]

Member	Description
enMode	Thresh operation mode
u16LowThr	Low threshold

u16HighThr	High threshold
un8MinVal	Minimum value. Value range: [0, 255]
un8MidVal	Median value. Value range: [0, 255]
un8MaxVal	Maximum value. Value range: [0, 255]

3.26.4 [Note]

- For details about the related formula, see the Note field of MI_MVE_Thresh_U16

3.26.5 [See Also]

- **MVE_THRESH_U16_MODE_E**

3.27. MVE_16BIT_TO_8BIT_CTRL_S

3.27.1 [Description]

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

3.27.2 [Syntax]

```
typedef struct _MVE_16BIT_TO_8BIT_CTRL_S
{
    MVE_16BIT_TO_8BIT_MODE_E enMode;
    U16 u16Denominator;
    U8 u8Numerator;
    S8 s8Bias;
} MVE_16BIT_TO_8BIT_CTRL_S;
```

3.27.3 [Member]

Member	Description
enMode	operation mode
u16Denominator	Denominator during linear transformation Value range: [max{1, u8Numerator}, 65535]
u8Numerator	Numerator during linear transformation Value range: [0, 255]

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

3.27.4 [Note]

- For details about the related formula, see the Note field of MI_MVE_16BitTo8Bit
- The following condition must be met: u8Numerator ≤ u16Denominator and u16Denominator ≠ 0

3.27.5 [See Also]

- MVE_16BIT_TO_8BIT_MODE_E

3.28. MVE_MAG_AND_ANG_CTRL_S

3.28.1 [Description]

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

3.28.2 [Syntax]

```
typedef struct _MVE_MAG_AND_ANG_CTRL_S
{
    MVE_MAG_AND_ANG_OUT_CTRL_E enOutCtrl;
    U16 u16Thr;
    S8   as8Mask[25];
} MVE_MAG_AND_ANG_CTRL_S;
```

3.28.3 [Member]

Member	Description
enOutCtrl	Output format
U16Thr	Threshold for implementing thresh of the magnitude
as8Mask[25]	5x5 template coefficient

3.28.4 [Note]

- None

3.28.5 [See Also]

- **MVE_MAG_AND_ANG_OUT_CTRL_E**

3.29. MVE_SUB_CTRL_S

3.29.1 [Description]

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

3.29.2 [Syntax]

```
typedef struct _MVE_SUB_CTRL_S
{
    MVE_SUB_MODE_E enMode;
} MVE_SUB_CTRL_S;
```

3.29.3 [Member]

Member	Description
enMode	Subtraction mode of two images.

3.29.4 [Note]

- None

3.29.5 [See Also]

- MVE_SUB_MODE_E

3.30. MVE_ADD_CTRL_S

3.30.1 [Description]

Defines weighted addition control parameters for two images.

3.30.2 [Syntax]

```
typedef struct _MVE_ADD_CTRL_S
{
    U0Q16 u0q16X;
    U0Q16 u0q16Y;
} MVE_ADD_CTRL_S;
```

3.30.3 [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}

3.30.4 [Note]

- None

3.30.5 [See Also]

- None

3.31. MVE_ORD_STAT_FILTER_CTRL_S

3.31.1 [Description]

Defines the control parameter for order statistics filter.

3.31.2 [Syntax]

```
typedef struct _MVE_ORD_STAT_FILTER_CTRL_S
{
    MVE_ORD_STAT_FILTER_MODE_E enMode;
} MVE_ORD_STAT_FILTER_CTRL_S;
```

3.31.3 [Member]

Member	Description
enMode	Order statistics filter mode

3.31.4 [Note]

- None

3.31.5 [See Also]

- MVE_ORD_STAT_FILTER_MODE_E

3.32. MVE_BERNSEN_CTRL_S

3.32.1 [Description]

Defines Bernsen thresh control parameters.

3.32.2 [Syntax]

```
typedef struct _MVE_BERNSEN_CTRL_S
{
    MVE_BERNSEN_MODE_E enMode;
    U8 u8WinSize;
    U8 u8Thr;
```

```
} MVE_BERNSEN_CTRL_S;
```

3.32.3 [Member]

Member	Description
enMode	Bernsen thresh mode
u8WinSize	Window size for the local threshold calculation Value range: {3, 5}
u8Thr	Thresh in MVE_BERNSEN_MODE_THRESH mode in which the global threshold is involved Value range: [0, 255]

3.32.4 [Note]

- None

3.32.5 [See Also]

- MVE_BERNSEN_MODE_E

3.33. MVE_MEM_INFO_S

3.33.1 [Description]

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

3.33.2 [Syntax]

```
typedef struct _MVE_MEM_INFO_S
{
    void *pu32PhyAddr;
    U8   *pu8VirAddr;
    U32   u32Size;
} MVE_MEM_INFO_S;
```

3.33.3 [Member]

Member	Description
pu32PhyAddr	Physical address for 1D data
pu8VirAddr	Virtual address for 1D data
u32Size	Number of 1D data bytes

3.33.4 [Note]

- None

3.33.5 [See Also]

- MVE_SRC_MEM_INFO_S
- MVE_DST_MEM_INFO_S

3.34. MVE_SRC_MEM_INFO_S

3.34.1 [Description]

Defines 1D source data.

3.34.2 [Syntax]

typedef MVE_MEM_INFO_S MVE_SRC_MEM_INFO_S

3.34.3 [Member]

- None

3.34.4 [Note]

- None

3.34.5 [See Also]

- MVE_MEM_INFO_S
- MVE_DST_MEM_INFO_S

3.35. MVE_DST_MEM_INFO_S

3.35.1 [Description]

Defines 1D destination data.

3.35.2 [Syntax]

typedef MVE_MEM_INFO_S MVE_DST_MEM_INFO_S

3.35.3 [Member]

- None

3.35.4 [Note]

- None

3.35.5 [See Also]

- MVE_MEM_INFO_S
- MVE_SRC_MEM_INFO_S

3.36. MVE_MAP_LUT_MEM_S

3.36.1 [Description]

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

3.36.2 [Syntax]

```
typedef struct _MVE_MAP_LUT_MEM_S
{
    U8   au8Map[MVE_MAP_NUM];
} MVE_MAP_LUT_MEM_S;
```

3.36.3 [Member]

Member	Description
au8Map[MVE_MAP_NUM]	Map lookup table array.

3.36.4 [Note]

- None

3.36.5 [See Also]

- None

3.37. MVE_NCC_DST_MEM_S

3.37.1 [Description]

Defines the information about the NCC output memory.

3.37.2 [Syntax]

```
typedef struct _MVE_NCC_DST_MEM_S
{
    uint64_t u64Numerator;
    uint64_t u64QuadSum1;
    uint64_t u64QuadSum2;
} MVE_NCC_DST_MEM_S;
```

3.37.3 [Member]

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

	radical sign): $\sum_{i=1}^{w}\sum_{j=1}^{h}(I^2_{src1}(i,j))$
u64QuadSum2	Denominator in the NCC formula (part in the radical sign): $\sum_{i=1}^{w}\sum_{j=1}^{h}(I^2_{src2}(i,j))$

3.37.4 [Note]

- For details about the related formula, see the [Note] field of **MI_MVE_NCC**

3.37.5 [See Also]

- None

3.38. MVE_IMG_INFO

3.38.1 [Description]

Defines input image information for MI_MVE_Init.

3.38.2 [Syntax]

```
typedef struct _MVE_IMG_INFO
{
    S16 width;
    S16 height;
} MVE_IMG_INFO;
```

3.38.3 [Member]

Member	Description
width	Input image width
height	Input image height

3.38.4 [Note]

- None

3.38.5 [See Also]

- None

3.39. MVE_HANDLE

3.39.1 [Description]

Defines MVE handle.

3.39.2 [Syntax]

```
typedef void* MVE_HANDLE;
```

3.39.3 [Member]

- None

3.39.4 [Note]

- None

3.39.5 [See Also]

- None

3.40. MVE_INTEG_CTRL_S

3.40.1 [Description]

Defines the integrogram calculation control parameter.

3.40.2 [Syntax]

```
typedef struct _MVE_INTEG_CTRL_S
{
    MVE_INTEG_OUT_CTRL_E enOutCtrl;
} MVE_INTEG_CTRL_S;
```

3.40.3 [Member]

Member	Description
enOutCtrl	Integrogram output control parameter

3.40.4 [Note]

- None

3.40.5 [See Also]

- MVE_INTEG_OUT_CTRL_E

3.41. MVE_INTEG_OUT_CTRL_E

3.41.1 [Description]

Defines integrogram output control parameters.

3.41.2 [Syntax]

```
typedef enum _MVE_INTEG_OUT_CTRL_E
{
    MVE_INTEG_OUT_CTRL_COMBINE  = 0x0,
```

```
    MVE_INTEG_OUT_CTRL_SUM      = 0x1,
    MVE_INTEG_OUT_CTRL_SQSUM    = 0x2,
    MVE_INTEG_OUT_CTRL_BUTT
} MVE_INTEG_OUT_CTRL_E;
```

3.41.3 [Member]

Member	Description
MVE_INTEG_OUT_CTRL_COMBINE	Combined output of the sum integrogram and squre sum integrogram
MVE_INTEG_OUT_CTRL_SUM	Output of only the sum integrogram
MVE_INTEG_OUT_CTRL_SQSUM	Output of only the square sum integrogram

3.41.4 [Note]

- None

3.41.5 [See Also]

- MVE_INTEG_CTRL_S

3.42. MVE_CCBLOB_S

3.42.1 [Description]

Defines CCL output information.

3.42.2 [Syntax]

```
typedef struct _MVE_CCBLOB_S
{
    U16 u16CurAreaThr;
    S8   s8LabelStatus;
    U8   u8RegionNum;
    MVE_REGION_S astRegion[MVE_MAX_REGION_NUM];
} MVE_CCBLOB_S;
```

3.42.3 [Member]

Member	Description
u16CurAreaThr	Area threshold for valid connected components. The components in **astRegion** whose area is below the threshold are set to **0**.
s16LabelStatus	Connected component labeling status -1: failure 0: success
astRegion[MVE_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.

3.42.4 [Note]

- None

3.42.5 [See Also]

- MVE_REGION_S

3.43. MVE_CCL_CTRL_S

3.43.1 [Description]

Defines CCL control parameters.

3.43.2 [Syntax]

```
typedef struct _MVE_CCL_CTRL_S
{
    U16 u16InitAreaThr;
    U16 u16Step;
} MVE_CCL_CTRL_S;
```

3.43.3 [Member]

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

3.43.4 [Note]

- None

3.43.5 [See Also]

- MVE_CCBLOB_S

3.44. MVE_REGION_S

3.44.1 [Description]

Defines connected component information.

3.44.2 [Syntax]

```
typedef struct _MVE_REGION_S
{
    U32 u32Area;
    U16 u16Left;
    U16 u16Right;
    U16 u16Top;
    U16 u16Bottom;
} MVE_REGION_S;
```

3.44.3 [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

3.44.4 [Note]

- None

3.44.5 [See Also]

- MVE_CCBLOB_S

3.45. MVE_HIST_NUM

3.45.1 [Description]

Defines the number of bins in a histogram.

3.45.2 [Syntax]

```
#define MVE_HIST_NUM        256
```

3.45.3 [Member]

- None

3.45.4 [Note]

- None

3.45.5 [See Also]

- None

3.46. MVE_HIST_TABLE_MEM_S

3.46.1 [Description]

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

3.46.2 [Syntax]

```
typedef struct _MVE_HIST_MEM_S
{
    U32   au32Hist[MVE_HIST_NUM];
} MVE_HIST_TABLE_MEM_S;
```

3.46.3 [Member]

Member	Description
au32Hist[MVE_HIST_NUM]	Hist lookup table array.

3.46.4 [Note]

- None

3.46.5 [See Also]

- None

3.47. MVE_EQUALIZE_HIST_CTRL_S

3.47.1 [Description]

Defines the histogram equalization control parameter.

3.47.2 [Syntax]

```
typedef struct _MVE_EQUALIZE_HIST_CTRL_S
{
```

```
    MVE_MEM_INFO_S stMem;
} MVE_EQUALIZE_HIST_CTRL_S;
```

3.47.3 [Member]

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

3.47.4 [Note]

- None

3.47.5 [See Also]

- MVE_EQUALIZE_HIST_CTRL_MEM_S

3.48. MVE_EQUALIZE_HIST_CTRL_MEM_S

3.48.1 [Description]

Defines the histogram equalization auxiliary memory.

3.48.2 [Syntax]

```
typedef struct _MVE_EQUALIZE_HIST_CTRL_MEM_S
{
    U32 au32Hist[MVE_HIST_NUM];
    U8 au8Map[MVE_MAP_NUM];
} MVE_EQUALIZE_HIST_CTRL_MEM_S;
```

3.48.3 [Member]

Member	Description
au32Hist[MVE_HIST_NUM]	Histogram statistics output
au8Map[MVE_MAP_NUM]	Map lookup table calculated based on the histogram statistics

3.48.4 [Note]

- None

3.48.5 [See Also]

- MVE_EQUALIZE_HIST_CTRL_S

3.49. MVE_SAD_OUT_CTRL_E

3.49.1 [Description]

Defines the SAD output control information.

3.49.2 [Syntax]

```
typedef enum _MVE_SAD_OUT_CTRL_E
{
    MVE_SAD_OUT_CTRL_16BIT_BOTH   = 0x0,
    MVE_SAD_OUT_CTRL_8BIT_BOTH    = 0x1,
    MVE_SAD_OUT_CTRL_16BIT_SAD    = 0x2,
    MVE_SAD_OUT_CTRL_8BIT_SAD     = 0x3,
    MVE_SAD_OUT_CTRL_THRESH       = 0x4,
    MVE_SAD_OUT_CTRL_BUTT
} MVE_SAD_OUT_CTRL_E;
```

3.49.3 [Member]

Member	Description
MVE_SAD_OUT_CTRL_16BIT_BOTH	Output 16-bit SAD value and 8-bit thresh value.
MVE_SAD_OUT_CTRL_8BIT_BOTH	Output 8-bit SAD value and 8-bit thresh value.
MVE_SAD_OUT_CTRL_16BIT_SAD	Output 16-bit SAD value only.
MVE_SAD_OUT_CTRL_8BIT_SAD	Output 8-bit SAD value only.
MVE_SAD_OUT_CTRL_THRESH	Output 8-bit thresh value only.

3.49.4 [Note]

- None

3.49.5 [See Also]

- MVE_SAD_CTRL_S
- MVE_SAD_MODE_E

3.50. MVE_SAD_CTRL_S

3.50.1 [Description]

Defines SAD control parameters.

3.50.2 [Syntax]

```
typedef struct _MVE_SAD_CTRL_S
{
    MVE_SAD_MODE_E enMode;
```

```
    MVE_SAD_OUT_CTRL_E enOutCtrl;
    U16 u16Thr;
    U8 u8MinVal;
    U8 u8MaxVal;
} MVE_SAD_CTRL_S;
```

3.50.3 [Member]

Member	Description
enMode	SAD operating mode. For detail, see MVE_SAD_MODE_E
enOutCtrl	SAD output format. For detail, see MVE_SAD_OUT_CTRL_E
u16Thr	Thresh value for thresh output. Value range: [0, 65535]
u8MinVal	Smaller value of thresh output. Value range: [0, 255]
u8MaxVal	Bigger value of thresh output. Value range: [0, 255]

3.50.4 [Note]

- u16Thr, u8MinVal and u8MaxVal are used for thresh output.
- If SAD(x,y) <= u16Thr, Thr(x,y) = u8MinVal; Otherwise, Thr(x,y) = u8MaxVal.

3.50.5 [See Also]

- **MVE_SAD_MODE_E**
- **MVE_SAD_OUT_CTRL_E**

3.51. MVE_SAD_MODE_E

3.51.1 [Description]

Defines SAD block size.

3.51.2 [Syntax]

```
typedef enum _MVE_SAD_MODE_E
{
    MVE_SAD_MODE_MB_4X4     = 0x0,
    MVE_SAD_MODE_MB_8X8     = 0x1,
    MVE_SAD_MODE_MB_16X16   = 0x2,

    MVE_SAD_MODE_BUTT
```

```
} MVE_SAD_MODE_E;
```

3.51.3 [Member]

Member	Description
MVE_SAD_MODE_MB_4X4	SAD block size is 4x4
MVE_SAD_MODE_MB_8X8	SAD block size is 8x8
MVE_SAD_MODE_MB_16X16	SAD block size is 16x16

3.51.4 [Note]

- None.

3.51.5 [See Also]

- **MVE_SAD_CTRL_S**
- **MVE_SAD_OUT_CTRL_E**

3.52. MVE_NORM_GRAD_CTRL_S

3.52.1 [Description]

Defines control parameters for the normalized gradient calculation.

3.52.2 [Syntax]

```
typedef struct _MVE_NORM_GRAD_CTRL_S
{
    MVE_NORM_GRAD_OUT_CTRL_E enOutCtrl;
    S8 as8Mask[25];
    U8 u8Norm;
} MVE_NORM_GRAD_CTRL_S;
```

3.52.3 [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]

3.52.4 [Note]

- None

3.52.5 [See Also]

- MVE_NORM_GRAD_OUT_CTRL_E

3.53. MVE_NORM_GRAD_OUT_CTRL_E

3.53.1 [Description]

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

3.53.2 [Syntax]

```
typedef enum _MVE_NORM_GRAD_OUT_CTRL_E
{
    MVE_NORM_GRAD_OUT_CTRL_HOR_AND_VER   = 0x0,
    MVE_NORM_GRAD_OUT_CTRL_HOR           = 0x1,
    MVE_NORM_GRAD_OUT_CTRL_VER           = 0x2,
    MVE_NORM_GRAD_OUT_CTRL_COMBINE       = 0x3,
    MVE_ NORM_GRAD_OUT_CTRL _BUTT
} MVE_NORM_GRAD_OUT_CTRL_E;
```

3.53.3 [Member]

Member	Description
MVE_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 MI_MVE_NormGrad.
MVE_NORM_GRAD_OUT_CTRL_HOR	Output of only the H component diagram in the gradient information
MVE_NORM_GRAD_OUT_CTRL_VER	Output of only the V component diagram in the gradient information
MVE_NORM_GRAD_OUT_CTRL_COMBINE	Output of the HV diagram in the gradient information that is stored in package format

3.53.4 [Note]

- None

3.53.5 [See Also]

- MVE_NORM_GRAD_CTRL_S

3.54. MVE_LBP_CTRL_S

3.54.1 [Description]

Defines LBP texture calculation control parameters.

3.54.2 [Syntax]

```
typedef struct _MVE_LBP_CTRL_S
{
    MVE_LBP_CMP_MODE_E enMode;
    MVE_8BIT_U un8BitThr;
} MVE_ LBP_CTRL_S;
```

3.54.3 [Member]

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

3.54.4 [Note]

For details about the related formula, see the Note field of MI_MVE_LBP and Figure 2-7.

3.54.5 [See Also]

- MVE_LBP_CMP_MODE_E
- MVE_8BIT_U

3.55. MVE_LBP_CMP_MODE_E

3.55.1 [Description]

Defines the comparison mode during LBP calculation.

3.55.2 [Syntax]

```
typedef enum _MVE_LBP_CMP_MODE_E
{
    MVE_LBP_CMP_MODE_NORMAL     = 0x0,
    MVE_LBP_CMP_MODE_ABS        = 0x1,
    MVE_ LBP_CMP_MODE _BUTT
} MVE_LBP_CMP_MODE_E;
```

3.55.3 [Member]

Member	Description
MVE_LBP_CMP_MODE_NORMAL	Simple comparison mode
MVE_ LBP_CMP_MODE_ABS	Absolute value comparison mode

3.55.4 [Note]

For details about the related formula, see the Note field of MI_MVE_LBP and Figure 2-7.

3.55.5 [See Also]

- MVE_LBP_CTRL_S

3.56. MVE_8BIT_U

3.56.1 [Description]

Defines an 8-bit data union.

3.56.2 [Syntax]

```
typedef union _MVE_8BIT_U
{
    S8 s8Val;
    U8 u8Val;
} MVE_8BIT_U;
```

3.56.3 [Member]

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

3.56.4 [Note]

- None

3.56.5 [See Also]

- None

3.57. MVE_LOOKUP_TABLE_S

3.57.1 [Description]

Defines the lookup table.

3.57.2 [Syntax]

TBD.

3.57.3 [Member]

Member	Description

3.57.4 [Note]

It is not in use now.

3.57.5 [See Also]

- None.

3.58. MVE_GMM_CTRL_S

3.58.1 [Description]

Defines the control parameters for GMM background modeling.

3.58.2 [Syntax]

```
typedef struct _MVE_GMM_CTRL_S
{
    U22Q10      u22q10NoiseVar;
    U22Q10      u22q10MaxVar;
    U22Q10      u22q10MinVar;
    U0Q16       u0q16LearnRate;
    U0Q16       u0q16BgRatio;
    U8Q8        u8q8VarThr;
    U0Q16       u0q16InitWeight;
    U8          u8ModelNum;
} MVE_GMM_CTRL_S;
```

3.58.3 [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)
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.

3.58.4 [Note]

- None

3.58.5 [See Also]

- None

3.59. MVE_CANNY_STACK_SIZE_S

3.59.1 [Description]

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

3.59.2 [Syntax]

```
typedef struct _MVE_CANNY_STACK_SIZE_S
{
    U32     u32StackSize;
    U8      u8Reserved[12];
} MVE_CANNY_STACK_SIZE_S;
```

3.59.3 [Member]

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

3.59.4 [Note]

- None

3.59.5 [See Also]

- None

3.60. MVE_CANNY_HYS_EDGE_CTRL_S

3.60.1 [Description]

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

3.60.2 [Syntax]

```
typedef struct _MVE_CANNY_HYS_EDGE_CTRL_S
{
    MVE_MEM_INFO_S    stMem;
    U16               u16LowThr;
    U16               u16HighThr;
    S8                as8Mask[25];
} MVE_ CANNY_HYS_EDGE_CTRL_S;
```

3.60.3 [Member]

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

3.60.4 [Note]

- None

3.60.5 [See Also]

- None

3.61. MVE_LINE_FILTER_HOR_CTRL_S

3.61.1 [Description]

Defines control parameters for filtering the horizontal density of binary images.

3.61.2 [Syntax]

```
typedef struct _MVE_LINE_FILTER_HOR_CTRL_S
{
    U8 u8GapMinLen;
    U8 u8DensityThr;
    U8 u8HorThr;
} MVE_LINE_FILTER_HOR_CTRL_S;
```

3.61.3 [Member]

Member	Description
u8GapMinLen	Minimum black line length. For details, see thr1 in the note field of MI_MVE_LineFilterHor. Value range: [1, 20] Reference value: 10
u8DensityThr	Density threshold. For details, see thr2 in the Note field of MI_MVE_LineFilterHor. Value range: [1, 50]

	Reference value: 20
u8HorThr	Horizontal line length threshold. For details, see thr3 in the note field of MI_MVE_LineFilterHor. Value range: [1, 50] Reference value: 20

3.61.4 [Note]

- None.

3.61.5 [See Also]

- None.

3.62. MVE_LINE_FILTER_VER_CTRL_S

3.62.1 [Description]

Defines control parameters for filtering the vertical density of binary images.

3.62.2 [Syntax]

```
typedef struct _MVE_LINE_FILTER_VER_CTRL_S
{
    U8 u8VerThr;
} MVE_LINE_FILTER_VER_CTRL_S;
```

3.62.3 [Member]

Member	Description
u8VerThr	Vertical line length threshold. For details, see thr in the note field of MI_MVE_LineFilterVer. Value range: [1, 64] Reference value: 30

3.62.4 [Note]

- None.

3.62.5 [See Also]

- None.

3.63. MVE_NOISE_REMOVE_HOR_CTRL_S

3.63.1 [Description]

Defines the horizontal noise removal control parameter for the binary image.

3.63.2 [Syntax]

```
typedef struct _MVE_NOISE_REMOVE_HOR_CTRL_S
{
    U8 u8HorThr;
    U8 u8HorThrMax;
} MVE_NOISE_REMOVE_HOR_CTRL_S;
```

3.63.3 [Member]

Member	Description
u8HorThr	Length threshold for determining horizontal noises, see thr1 in the note field of MI_MVE_NoiseRemoveHor. Value range: [1, 100] Reference value: 25
u8HorThrMax	Maximum length threshold for determining horizontal , see thr2 in the note field of MI_MVE_NoiseRemoveHor. Value range: [1, 100] Reference value: 79

3.63.4 [Note]

- u8HorThrMax is always bigger than u8HorThr.

3.63.5 [See Also]

- None.

3.64. MVE_NOISE_REMOVE_VER_CTRL_S

3.64.1 [Description]

Defines the vertical noise removal control parameter for the binary image.

3.64.2 [Syntax]

```
typedef struct _MVE_NOISE_REMOVE_VER_CTRL_S
{
    U8 u8VerThr;
    U8 u8VerThrMax;
} MVE_NOISE_REMOVE_VER_CTRL_S;
```

3.64.3 [Member]

Member	Description
u8VerThr	Length threshold for determining vertical noises, see thr1 in the note field of MI_MVE_NoiseRemoveVer. Value range: [1, 100]

	Reference value: 25
u8VerThrMax	Maximum Length threshold for determining vertical noises, see thr2 in the note field of MI_MVE_NoiseRemoveVer. Value range: [1, 100] Reference value: 79

3.64.4 [Note]

- u8VerThrMax is always bigger than u8VerThr.

3.64.5 [See Also]

- None.

3.65. MVE_ADP_THRESH_CTRL_S

3.65.1 [Description]

Defines adaptive thresh control parameters.

3.65.2 [Syntax]

```
typedef struct _MVE_ADP_THRESH_CTRL_S
{
    U8 u8RateThr;
    U8 u8HalfMaskx;
    U8 u8HalfMasky;
    S16 s16Offset;
    U8 u8ValueThr;
} MVE_ADP_THRESH_CTRL_S;
```

3.65.3 [Member]

Member	Description
u8RateThr	Threshold rate for determining adaptive thresh. For details, see RateThr in the formula of MI_MVE_AdpThresh. Value range: [1, 20] Default: 10
u8HalfMaskx	Half of Mask size width for determining adaptive thresh. For details, see w in the formula of MI_MVE_AdpThresh. Value range: [1,40] Default: 10
u8HalfMasky	Half of Mask size height for determining adaptive thresh. For details, see h in the formula of MI_MVE_AdpThresh. Value range: [1,40] Default: 35

s16Offset	Offset for determining adaptive thresh. For details, see Offset in the formula of MI_MVE_AdpThresh. Value range: [-128,127] Default: -100
u8ValueThr	Threshold pixel value for determining adaptive thresh. For details, see ValueThr in the formula of MI_MVE_AdpThresh. Value range: [1,255] Default: 100

3.65.4 [Note]

- None.

3.65.5 [See Also]

- None.

3.66. MVE_LK_OPTICAL_FLOW_CTRL_S

3.66.1 [Description]

Define optical flow tracking control parameters.

3.66.2 [Syntax]

```
typedef struct _MVE_LK_OPTICAL_FLOW_CTRL_S
{
    U16 u16CornerNum;   /* maxCorners, Number of the feature points, [1, 200] */
    U0Q8 u0q8MinEigThr;   /* QualityLevel, minEigenvalue, Minimum eigenvalue
threshold, [1, 255] */
    U8 u8IterCount;   /* maxIterations, Maximum iteration times, [1, 20] */
    U0Q8 u0q8Epsilon;   /* minDisplacement, Threshold of iteration for dx^2 + dy^2 <
u0q8Epsilon, [1, 255] */
    U16   u16PyrLevel;   /* Change nPyramidLevels and subsampling */
}  MVE_LK_OPTICAL_FLOW_CTRL_S;
```

3.66.3 [Member]

Member	Description
u16CornerNum	Number of the feature points, [1, 200] Default: ≦50
u0q8MinEigThr	Minimum eigenvalue threshold, [1, 255] Default: 255
u8IterCount	Maximum iteration times, [1, 20]

	Default: 10
u0q8Epsilon	Threshold of iteration for dx^2 + dy^2 < u0q8Epsilon, [1, 255] Default: 26
u16PyrLevel	Change nPyramidLevels and subsampling. Suggest ≦15 Default: 15

3.66.4 [Note]

- None.

3.66.5 [See Also]

- None.

3.67. MVE_POINT_S25Q7_S

3.67.1 [Description]

Define Point position.

3.67.2 [Syntax]

```
typedef struct _MVE_POINT_S25Q7_S
{
    S25Q7 s25q7X;   /*X coordinate*/
    S25Q7 s25q7Y;   /*Y coordinate*/
}  MVE_POINT_S25Q7_S;
```

3.67.3 [Member]

Member	Description
s25q7X	X coordinate, where 25bits signed integer & 7bits floating.
s25q7Y	Y coordinate, where 25bits signed integer & 7bits floating.

3.67.4 [Note]

- None.

3.67.5 [See Also]

- None.

3.68. MVE_MV_S9Q7_S

3.68.1 [Description]

Define moving distance.

3.68.2 [Syntax]

```
typedef struct _MVE_MV_S9Q7_S
{
    S32 s32Status;   /* Result of tracking: 0-success; -1-failure */
    S9Q7 s9q7Dx;     /* X-direction component of the movement */
    S9Q7 s9q7Dy;     /* Y-direction component of the movement */
}  MVE_MV_S9Q7_S;
```

3.68.3 [Member]

Member	Description
s32Status	Result of tracking: 0-success; -1-failure
s9q7Dx	X-direction component of the movement, where 9bits signed integer & 7bits floating.
s9q7Dy	Y-direction component of the movement, where 9bits signed integer & 7bits floating.

3.68.4 [Note]

- None.

3.68.5 [See Also]

- None.

4. ERROR CODE

4.1. MVE error code table

Error Code	Macro Definition	Description
0x0	MI_RET_SUCCESS	API execution successfully
0x10000101	MI_MVE_RET_INIT_ERROR	MVE init error
0x10000102	MI_MVE_RET_IC_CHECK_ERROR	MVE platform check error
0x10000103	MI_MVE_RET_INVALID_HANDLE	Invalid MVE handle
0x10000104	MI_MVE_RET_INVALID_PARAMETER	Invalid MVE parameter
0x10000105	MI_MVE_RET_INVALID_ADDRESS	Invalid buffer address
0x10000106	MI_MVE_RET_NOT_SUPPORT	Unsupported feature of MVE
0x10000107	MI_MVE_RET_HARDWARE_TIMEOUT	MVE hardware timeout
0x10000108	MI_MVE_RET_HARDWARE_ERROR	MVE hardware error
0x10000109	MI_MVE_RET_HARDWARE_BUSY	MVE hardware busy
0x1000010A	MI_MVE_RET_INVALID_BUFFER_NAME	Invalid buffer name when allocating MVE buffer
0x1000010B	MI_MVE_RET_MALLOC_ERROR	Allocate MVE buffer error