ArcSoft MobileCv Lib

Developer's Guide

ArcSoft Corporation
46601 Fremont Blvd.
Fremont, CA 94538
http://www.arcsoft.com

Trademark or Service Mark Information

ArcSoft Inc. and ArcWare are registered trademarks of ArcSoft Inc.

Other product and company names mentioned herein may be trademarks and/or service marks of their respective owners. The absence of a trademark or service mark from this list does not constitute a waiver of ArcSoft Inc.'s trademark or other intellectual property rights concerning that trademark or service mark.

The information contained in this document is for discussion purposes only. None of the information herein shall be interpreted as an offer or promise to any of the substance herein nor as an agreement to contract or license, or as an implication of a transfer of rights. Any and all terms herein are subject to change at the discretion of ArcSoft. Copying, distributing, transferring or any other reproduction of these documents or the information contained herein is expressly prohibited, unless such activity is expressly permitted by an authorized representative of ArcSoft, Inc.

Version	Date	Modifier	Summary of changes
ArcSoft_MobileCV_0.1.0.1	05/02/2013	ArcSoft	Initial version

Chapter 1: Introduction

1.1. Overview

The ArcSoft® MobileCv Lib provides a set of efficient APIs for mobile platform. It contains several categories of functions such as resize, color space conversion, motion related functions ,etc. It is written by C language, and supports NEON instructions. To be added….

1.2. Deliverables

- Header files
- API reference document
- Library file(s)

1.3. Platforms

- ADS/RVDS
- WIN32
- Android
- Others

1.4. Peak Memory

- Memory Needed

It requires 3000KB runtime memory for this version library. To be added….

- Space for Program Data memory

1700K or above

- Space for Program Binary Code

1000K or above ROM space is needed for this version of the library.

Chapter 2: Structures and Constants

2.1. Macro definition

2.1.1. Pixel array format definition

Definition	Description
ASVL_PAF_YUYV	y0,u0,y1,v0,y2,u1,y3,v1...
ASVL_PAF_LPI422H	y0,y1,y2,y3,y4,y5,y6,y7...u0,v0,u1,v1,u2,v2,u3,v3...
ASVL_PAF_NV12	y0,y1,y2,y3,y4,y5,y6,y7...u0,v0,u1,v1...
ASVL_PAF_I420	y0, y1, y2, y3... u0, u1... v0, v1... 8 bit Y plane followed by 8 bit 2x2 subsampled U and V planes
ASVL_PAF_YVYU2	y1, v0, y0, u0, y3, v1, y2, u1...
ASVL_PAF_I422H	y0, y1, y2, y3... u0, u1... v0, v1... 8 bit Y plane followed by 8 bit 2x1 subsampled U and V planes

2.1.2. Error definition

Definition	Description	Value
MCV_OK	No error	0
MCV_NULL_POINTER	Null pointer	-1
MCV_INVALID_PARAM	Invalid in/out parameters	-2
MCV_INVALID_CALL	Invalid function call method	-3
MCV_QUEUE_OVERFLOW	Queue is full when trying to add new element to the queue.	-4

2.1.3. Other definition

Definition	Description

2.2. General data structure

2.2.1. ASVLOFFSCREEN

Description

This structure defines information of image. We think an image as an offscreen.

Definition

```
typedef struct __tag_ASVL_OFFSCREEN
{
	MUInt32		u32PixelArrayFormat;
	MInt32		i32Width;
	MInt32		i32Height;
	MUInt8*		ppu8Plane[4];
	MInt32		pi32Pitch[4];
}ASVLOFFSCREEN, *LPASVLOFFSCREEN;
```

Member description

`i32Width`	Off-screen width
`i32Height`	Off-screen height
`u32PixelArrayFormat`	Format of pixel array
`pi32Pitch[4]`	The line bytes for each separate image data
`ppu8Plane[4]`	The pointer for each separate image data

2.2.2. MCV_Version

Description

This structure defines version information of the lib.

Definition

```
typedef struct
{
	MLong lCodebase;
	MLong lMajor;
	MLong lMinor;
	MLong lBuild;
	const MChar *Version;
	const MChar *BuildDate;
	const MChar *CopyRight;
} MCV_Version;
```

Member description

lCodebase	Codebase version number
lMajor	major version number
lMinor	minor version number
lBuild	Build version number, increasable only
Version	version in string form
BuildDate	latest build Date
CopyRight	copyright

2.2.3. TobeAdded

Chapter 3: Supporting API Reference

3.1. Resize API

These functions used for Resize a image.

3.1.1. mcvResizeYUYVToYUYVBilinear

Description

Resize one frame in ASVL_PAF_YUYV format for both input and output.
Use bilinear interpolation for Y component and neighbor interpolation for CbCr.

Prototype

```
MInt32 mcvResizeYUYVToYUYVBilinear(MInt32 *plTmpWidthBuf,MInt32 buflength,
             MUInt8 *pYUYVData, MInt32 lSrcWidth, MInt32 lSrcHeight,
             MInt32 lSrcLineStep, MUInt8 *pDstYUYVData,
             MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstLineStep);
```

Parameters

`plTmpWidthBuf`	[in]	The tmp allocated memory. Should align memory
`buflength`	[in]	The size of plTmpWidthBuf in [Byte] Unit
`pYUYVData`	[in]	The buffer of input YUYV frame
`lSrcWidth`	[in]	The width(columns) of input YUYV frame
`lSrcHeight`	[in]	The height(rows) of input YUYV frame
`lSrcLineStep`	[in]	The line step of pYUYVData in [Byte] Unit
`pDstYUYVData`	[out]	The buffer of output YUYV frame
`lDstWidth`	[in]	The width(columns) of output YUYV frame
`lDstHeight`	[in]	The height(rows) of output YUYV frame
`lDstLineStep`	[in]	The line step of pDstYUYVData in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of plTmpWidthBuf should be at least lDstWidth*3*sizeof(MInt32) + lDstWidth*2*sizeof(MUInt16).
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.
4. lSrcWidth, lDstWidth should be even.

3.1.2. mcvResizeYUYVToLPI422HBilinear

Description

Resize one frame. Input frame is ASVL_PAF_YUYV format and output frame is ASVL_PAF_LPI422H format.

Use bilinear interpolation for Y component and neighbor interpolation for CbCr.

Prototype

```
MInt32 mcvResizeYUYVToLPI422HBilinear(MUInt16 *plTmpBuf,MInt32 buflength,
                        LPASVLOFFSCREEN srcImage,LPASVLOFFSCREEN dstImage);
```

Parameters

`plTmpBuf`	[in]	The tmp allocated memory. Should align memory
`buflength`	[in]	The size of plTmpWidthBuf in [Byte] Unit
`srcImage`	[in]	input image
`dstImage`	[in]	output image

Return value

MCV_OK

MCV_NULL_POINTER

MCV_INVALID_PARAM

Notes

1. The size of plTmpBuf should be at least lDstWidth*5*sizeof(MUInt16).
2. All buffer pointers should not be NULL.
3. Width and height of both input and output image should be greater than 2 and be a multiple of 2.

3.1.3. mcvResizeYUYVToI422HBilinearY

Description

Resize one frame. Input frame is ASVL_PAF_YUYV format and output frame is ASVL_PAF_I422H format.

Use bilinear interpolation for Y component and neighbor interpolation for Cb and Cr.

Prototype

```
MInt32 mcvResizeYUYVToI422HBilinearY(MUInt16 *plTmpBuf,MInt32 buflength,
             MUInt8 *pSrcYUYV, MInt32 lSrcWidth, MInt32 lSrcHeight,
             MInt32 lSrcLineStep, MUInt8 *pDstY, MUInt8 *pDstCb, MUInt8 *pDstCr,
             MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStrideY,
             MInt32 lDstStrideCb,MInt32 lDstStrideCr);
```

Parameters

`plTmpBuf`	[in]	The tmp allocated memory. Should align memory

buflength	[in]	The size of plTmpBuf in [Byte] Unit
pSrcYUYV	[in]	The buffer of input YUYV frame
lSrcWidth	[in]	The width(columns) of input YUYV frame
lSrcHeight	[in]	The height(rows) of input YUYV frame
lSrcLineStep	[in]	The line step of pSrcYUYV in [Byte] Unit
pDstY	[out]	The buffer of output Y frame
pDstCb	[out]	The buffer of output Cb frame
pDstCr	[out]	The buffer of output Cr frame
lDstWidth	[in]	The width(columns) of output YUYV frame
lDstHeight	[in]	The height(rows) of output YUYV frame
lDstStrideY	[in]	The line step of pDstY in [Byte] Unit
lDstStrideCb	[in]	The line step of pDstCb in [Byte] Unit
lDstStrideCr	[in]	The line step of pDstCr in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of plTmpBuf should be at least sizeof(MInt16)*(lDstWidth * 4 + lDstWidth/2).
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.
4. lSrcWidth, lDstWidth should be even.

3.1.4. mcvResizeSingleComponentBilinear

Description

Resize one frame. Input frame is Y, or U, or V format and output frame is the same as intput.
Use bilinear interpolation.

Prototype

```
MInt32 mcvResizeSingleComponentBilinear(MUInt16 *plTmpBuf,MInt32 buflength,
                    MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight,
                    MInt32 lSrcStride, MUInt8 *pDst,MInt32 lDstWidth,
                    MInt32 lDstHeight, MInt32 lDstStride);
```

Parameters

plTmpBuf	[in]	The tmp allocated memory. Should align memory

buflength	[in]	The size of plTmpBuf in [Byte] Unit
pSrc	[in]	The buffer of input frame
lSrcWidth	[in]	The width(columns) of input frame
lSrcHeight	[in]	The height(rows) of input frame
lSrcStride	[in]	The line step of pSrc in [Byte] Unit
pDst	[out]	The buffer of output frame
lDstWidth	[in]	The width(columns) of output frame
lDstHeight	[in]	The height(rows) of output frame
lDstStride	[in]	The line step of pDst in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of plTmpBuf should be at least sizeof(MInt16)*(lDstWidth * 4).
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.

3.1.5. mcvResizeYUYVToYBilinear

Description

Resize one frame in ASVL_PAF_YUYV format for input and output with Y format.
Use bilinear interpolation for Y component.

Prototype

```
MInt32 mcvResizeYUYVToYBilinear(MInt32 *plTmpWidthBuf,MInt32 buflength,
                    MUInt8 *pYUYVData, MInt32 lSrcWidth, MInt32 lSrcHeight,
                    MInt32 lSrcLineStep, MUInt8 *pDstYData,
                    MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstLineStep);
```

Parameters

plTmpWidthBuf	[in]	The tmp allocated memory. Should align memory
buflength	[in]	The size of plTmpWidthBuf in [Byte] Unit
pYUYVData	[in]	The buffer of input YUYV frame
lSrcWidth	[in]	The width(columns) of input YUYV frame
lSrcHeight	[in]	The height(rows) of input YUYV frame
lSrcLineStep	[in]	The line step of pYUYVData in [Byte] Unit

pDstYData	[out]	The buffer of output Y frame
lDstWidth	[in]	The width(columns) of output YUYV frame
lDstHeight	[in]	The height(rows) of output YUYV frame
lDstLineStep	[in]	The line step of pDstYUYVData in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of plTmpWidthBuf should be at least lDstWidth*2*sizeof(MInt32) + lDstWidth*2*sizeof(MUInt16).
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.
4. lSrcWidth, lDstWidth should be even.

3.1.6. mcvResizeYUYVtoI422HDownSampleby2

Description

Resize one frame down by two in YUYV format for input and output with I422H format.

Prototype

```
MInt32 mcvResizeYUYVtoI422HDownSampleby2(MByte* pSrc, MLong lSrcStep,
       MLong lSrcWidth, MLong lSrcHeight, MByte* pDstY,MLong lDstYStep,
       MByte* pDstU, MLong lDstUStep, MByte* pDstV, MLong lDstVStep)
```

Parameters

pSrc	[in]	The YUYV format src image buffer
lSrcStep	[in]	The line step of pSrc in [Byte] Unit
lSrcWidth	[in]	The width(columns) of output YUYV frame
lSrcHeight	[in]	The height(rows) of output YUYV frame
pDstY	[out]	The buffer of output Y frame
lDstYStep	[in]	The line step of pDstY in [Byte] Unit
pDstU	[out]	The buffer of output U frame
lDstUStep	[in]	The line step of pDstU in [Byte] Unit
pDstV	[out]	The buffer of output V frame
lDstVStep	[in]	The line step of pDstV in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER

Notes

1. The size of pSrc should be lSrcWidth*lSrcHeight*2*sizeof(MByte).
2. The lheight should be double size of 2.
3. Buffer pointers pSrc pDstY pDstU pDstV should not be NULL.
4. lDstUStep, lDstVStep should be even. lDstYStep should be double size of lDstUStep.
5. The buffer size of pDstY should be lSrcWidth* lSrcHeight/4, buffer size of pDstU and pDstV should be lSrcWidth* lSrcHeight/8.

3.1.7. mcvResizeYUYVtoI422HDownSampleby2WithRect

Description

Resize the rectangle of one frame down by two in YUYV format for input and output with I422H format.

Prototype

```
MInt32 mcvResizeYUYVtoI422HDownSampleby2WithRect(MByte* pSrc, MLong lSrcStep,
       MLong lSrcWidth, MLong lSrcHeight, MByte* pDstY,MLong lDstYStep,
       MByte* pDstU, MLong lDstUStep, MByte* pDstV, MLong lDstVStep, MRECT *roi)
```

Parameters

`pSrc`	[in]	The YUYV format src image buffer
`lSrcStep`	[in]	The line step of pSrc in [Byte] Unit
`lSrcWidth`	[in]	The width(columns) of output YUYV frame
`lSrcHeight`	[in]	The height(rows) of output YUYV frame
`pDstY`	[out]	The buffer of output Y frame
`lDstYStep`	[in]	The line step of pDstY in [Byte] Unit
`pDstU`	[out]	The buffer of output U frame
`lDstUStep`	[in]	The line step of pDstU in [Byte] Unit
`pDstV`	[out]	The buffer of output V frame
`lDstVStep`	[in]	The line step of pDstV in [Byte] Unit
`roi`	[in]	The rectangle of image you want to resize

Return value

MCV_OK

MCV_NULL_POINTER

Notes

1. The size of pSrc should be lSrcWidth*lSrcHeight*2*sizeof(MByte).
2. Buffer pointers pSrc pDstY pDstU pDstV should not be NULL.
3. lDstUStep, lDstVStep should be even. lDstYStep should be double size of lDstUStep.
4. The size of pDstY should be (roi->right- roi->left)*(roi->bottom- roi->top)*sizeof(Byte)/8.

3.1.8. mcvResizeNV21ToLPI422HBilinear

Description

Resize the an image of ASVL_PAF_NV21 format and convert to ASVL_PAF_LPI422H format. Use Bilinear Interpolation for Y, and Nearest Interpolation for UV.

Prototype

```
MInt32 mcvResizeNV21ToLPI422HBilinear(MUInt16 *plTmpBuf,MInt32 buflength,
                                      LPASVLOFFSCREEN srcImage,
                                      LPASVLOFFSCREEN dstImage)
```

Parameters

`plTmpBuf`	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
`buflength`	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
`srcImage`	[in]	The descriptor of input image
`dstImage`	[out]	The descriptor of output image

Return value

MCV_OK

MCV_NULL_POINTER

MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least ((lDstWidth<<2) + (lDstWidth>>1))*sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. Width, Height of both input image and output image should be greater than 2.
4. Width of both input image and output image should be even.

3.1.9. mcvResizeNV21Bilinear

Description

Resize the an image of ASVL_PAF_NV21 format. Use Bilinear Interpolation for Y, and Nearest Interpolation for UV.

Prototype

```
MInt32 mcvResizeNV21Bilinear(MUInt16 *plTmpBuf,MInt32 buflength,
      MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight, MInt32 lSrcStride,
      MUInt8 *pDst, MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStride)
```

Parameters

`plTmpBuf`	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
`buflength`	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
`pSrc`	[in]	The buffer of input NV21 frame

lSrcWidth	[in]	The width(columns) of input NV21 frame
lSrcHeight	[in]	The height(rows) of input NV21 frame
lSrcStride	[in]	The line step of *pSrc* in [Byte] Unit
pDst	[out]	The buffer of output NV21 frame
lDstWidth	[in]	The width(columns) of output NV21 frame
lDstHeight	[in]	The height(rows) of output NV21 frame
lDstStride	[in]	The line step of *pDst* in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least ((lDstWidth<<2) + (lDstWidth>>1))*sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.
4. lSrcWidth, lSrcHeight ,lDstWidth, lDstHeight should be even.

3.1.10. mcvResizeNV12Bilinear

Description

Resize the an image of ASVL_PAF_NV12 format. Use Bilinear Interpolation for Y, and Nearest Interpolation for UV.

Prototype

```
MInt32 mcvResizeNV12Bilinear(MUInt16 *plTmpBuf,MInt32 buflength,
       MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight, MInt32 lSrcStride,
       MUInt8 *pDst, MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStride)
```

Parameters

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrc	[in]	The buffer of input NV12 frame
lSrcWidth	[in]	The width(columns) of input NV12 frame
lSrcHeight	[in]	The height(rows) of input NV12 frame
lSrcStride	[in]	The line step of *pSrc* in [Byte] Unit
pDst	[out]	The buffer of output NV12 frame
lDstWidth	[in]	The width(columns) of output NV12 frame
lDstHeight	[in]	The height(rows) of output NV12 frame

lDstStride	[in]	The line step of *pDst* in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least ((lDstWidth<<2) + (lDstWidth>>1))*sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.
4. lSrcWidth, lSrcHeight ,lDstWidth, lDstHeight should be even.

3.1.11. mcvResizeI420Bilinear

Description

Resize the an image of ASVL_PAF_I420 format. Use Bilinear Interpolation for Y, and Nearest Interpolation for UV.

Prototype

```
MInt32 mcvResizeI420Bilinear(MUInt16 *plTmpBuf,MInt32 buflength,
       MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight, MInt32 lSrcStride,
       MUInt8 *pDst, MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStride)
```

Parameters

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrc	[in]	The buffer of input I420 frame
lSrcWidth	[in]	The width(columns) of input I420 frame
lSrcHeight	[in]	The height(rows) of input I420 frame
lSrcStride	[in]	The line step of *pSrc* in [Byte] Unit
pDst	[out]	The buffer of output I420 frame
lDstWidth	[in]	The width(columns) of output I420 frame
lDstHeight	[in]	The height(rows) of output I420 frame
lDstStride	[in]	The line step of *pDst* in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least

((lDstWidth<<2) + (lDstWidth>>1))*sizeof(MUInt16) bytes.

2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.
4. lSrcWidth, lSrcHeight ,lDstWidth, lDstHeight should be even.

3.1.12. McvResizeRGB888Bilinear

Description

Resize the an image of ASVL_PAF_RGB24_R8G8B8 or ASVL_PAF_RGB24_ B8G8R8 format. Use Bilinear Interpolation.

Prototype

```
MInt32 mcvResizeRGB888Bilinear(MUInt16 *plTmpBuf,MInt32 buflength,
    MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight, MInt32 lSrcStride,
    MUInt8 *pDst, MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStride)
```

Parameters

`plTmpBuf`	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
`buflength`	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
`pSrc`	[in]	The buffer of input image
`lSrcWidth`	[in]	The width(columns) of input frame
`lSrcHeight`	[in]	The height(rows) of input frame
`lSrcStride`	[in]	The line step of *pSrc* in [Byte] Unit
`pDst`	[out]	The buffer of output frame
`lDstWidth`	[in]	The width(columns) of output frame
`lDstHeight`	[in]	The height(rows) of output frame
`lDstStride`	[in]	The line step of *pDst* in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least (lDstWidth<<3) *sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.

3.1.13. mcvResizeNV21toYUYVBilinear

Description

Resize the an image of ASVL_PAF_NV21 to ASVL_PAF_YUYV format. Use Bilinear Interpolation.

Prototype

```
MInt32 mcvResizeNV21toYUYVBilinear(LPASVLOFFSCREEN pSrcNV21,
                                   LPASVLOFFSCREEN pDstYUYV,
                                   MUInt16 *plTmpBuf, MInt32 buflength)
```

Parameters

`pSrcNV21`	[in]	The structure of input image
`pDstYUYV`	[out]	The structure of output frame
`plTmpBuf`	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
`buflength`	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least sizeof(MUInt16)*((lDstWidth << 2)+(lDstWidth>>1)) bytes.
2. All buffer pointers should not be NULL.
3. Width of both input image and output image should be >=2..

3.1.14. mcvResizeYUYVToI420BilinearY

Description

Resize one frame. Input frame is ASVL_PAF_YUYV format and output frame is ASVL_PAF_I420 format. Use bilinear interpolation for Y component and neighbor interpolation for Cb and Cr.

Prototype

```
MInt32 mcvResizeYUYVToI420BilinearY(MUInt16 *plTmpBuf,MInt32 buflength,
             MUInt8 *pSrcYUYV, MInt32 lSrcWidth, MInt32 lSrcHeight,
             MInt32 lSrcLineStep, MUInt8 *pDstY, MUInt8 *pDstCb, MUInt8 *pDstCr,
             MInt32 lDstWidth, MInt32 lDstHeight,
             MInt32 lDstStrideY,MInt32 lDstStrideCb,MInt32 lDstStrideCr);
```

Parameters

`plTmpBuf`	[in]	The tmp allocated memory. Should align memory
`buflength`	[in]	The size of plTmpBuf in [Byte] Unit
`pSrcYUYV`	[in]	The buffer of input YUYV frame
`lSrcWidth`	[in]	The width(columns) of input YUYV frame
`lSrcHeight`	[in]	The height(rows) of input YUYV frame
`lSrcLineStep`	[in]	The line step of pSrcYUYV in [Byte] Unit

pDstY	[out]	The buffer of output Y frame
pDstCb	[out]	The buffer of output Cb frame
pDstCr	[out]	The buffer of output Cr frame
lDstWidth	[in]	The width(columns) of output I420 frame
lDstHeight	[in]	The height(rows) of output I420 frame
lDstStrideY	[in]	The line step of pDstY in [Byte] Unit
lDstStrideCb	[in]	The line step of pDstCb in [Byte] Unit
lDstStrideCr	[in]	The line step of pDstCr in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of plTmpBuf should be at least sizeof(MInt16)*(lDstWidth * 4 + lDstWidth/2).
2. All buffer pointers should not be NULL.
3. lSrcWidth, lSrcHeight, lDstWidth, lDstHeight should be greater than 2.
4. lSrcWidth, lDstWidth should be even.

3.1.15. mcvResizeNV21ToI420Bilinear

Description

Resize the an image of ASVL_PAF_NV21 format and convert to ASVL_PAF_LPI420 format. Use Bilinear Interpolation for Y, and Nearest Interpolation for UV.

Prototype

```
MUInt32 mcvResizeNV21ToI420Bilinear(MUInt16 *plTmpBuf, MUInt32 buflength,
        MUInt8 *pSrcY, MUInt32 lSrcStrideY,  MUInt8* pSrcUV,
        MUInt32 lSrcStrideUV, MUInt32 lSrcWidth,MUInt32 lSrcHeight,
        MUInt8 *pDstY, MUInt32 lDstStrideY, MUInt8 *pDstU,
        MUInt32 lDstStrideU, MUInt8 *pDstV, MUInt32 lDstStrideV,
        MUInt32 lDstWidth,MUInt32 lDstHeight);
```

Parameters

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrcY	[in]	The buffer of input Y frame
lSrcStrideY	[in]	The line step of pSrcY in [Byte] Unit
pSrcUV	[in]	The buffer of input UV frame
lSrcStrideUV	[in]	The line step of pSrcUV in [Byte] Unit

lSrcWidth	[in]	The width(columns) of input NV21 frame
lSrcHeight	[in]	The height(rows) of input NV21 frame
pDstY	[out]	The buffer of output Y frame
lDstStrideY	[in]	The line step of pDstY in [Byte] Unit
pDstU	[out]	The buffer of output U frame
lDstStrideU	[in]	The line step of pDstU in [Byte] Unit
pDstV	[out]	The buffer of output V frame
lDstStrideV	[in]	The line step of pDstV in [Byte] Unit
lDstWidth	[in]	The width(columns) of output I420 frame
lDstHeight	[in]	The height(rows) of output I420 frame

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least sizeof(MUInt16)*((lDstWidth<<2) + (lDstWidth>>1)) bytes.
2. All buffer pointers should not be NULL.
3. Width, Height of both input image and output image should be greater than 2.

3.1.16. mcvResizeLPI422HToI420Bilinear

Description

Resize the an image of ASVL_PAF_LPI422H format and convert to ASVL_PAF_LPI420 format. Use Bilinear Interpolation for Y, and Nearest Interpolation for UV.

Prototype

```
MUInt32 mcvResizeLPI422HToI420Bilinear (
        MUInt16 *plTmpBuf, MUInt32buflength,
        MUInt8 *pSrcY, MUInt32 lSrcStrideY,  MUInt8* pSrcUV,
        MUInt32 lSrcStrideUV, MUInt32 lSrcWidth,MUInt32 lSrcHeight,
        MUInt8 *pDstY, MUInt32 lDstStrideY, MUInt8 *pDstU,
        MUInt32 lDstStrideU, MUInt8 *pDstV, MUInt32 lDstStrideV,
        MUInt32 lDstWidth,MUInt32 lDstHeight);
```

Parameters

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrcY	[in]	The buffer of input Y frame

lSrcStrideY	[in]	The line step of pSrcY in [Byte] Unit
pSrcUV	[in]	The buffer of input UV frame
lSrcStrideUV	[in]	The line step of pSrcUV in [Byte] Unit
lSrcWidth	[in]	The width(columns) of input frame
lSrcHeight	[in]	The height(rows) of input frame
pDstY	[out]	The buffer of output Y frame
lDstStrideY	[in]	The line step of pDstY in [Byte] Unit
pDstU	[out]	The buffer of output U frame
lDstStrideU	[in]	The line step of pDstU in [Byte] Unit
pDstV	[out]	The buffer of output V frame
lDstStrideV	[in]	The line step of pDstV in [Byte] Unit
lDstWidth	[in]	The width(columns) of output I420 frame
lDstHeight	[in]	The height(rows) of output I420 frame

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least ((lDstWidth<<2))*sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. Width, Height of both input image and output image should be greater than 2.

3.1.17. mcvResizeRGBA8888Bilinear

Description

Resize the an image of RGBA8888 using Bilinear Method.

Prototype

```
MInt32 mcvResizeRGBA8888Bilinear (
        MUInt16 *plTmpBuf,MInt32 buflength,
        MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight,
        MInt32 lSrcStride, MUInt8 *pDst,
        MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStride);
```

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrc	[in]	The buffer of input RGBA8888 frame

lSrcWidth	[in]	The width(columns) of input frame
lSrcHeight	[in]	The height(rows) of input frame
lSrcStride	[in]	The line step of input RGBA8888 in [Byte] Unit
pDst	[out]	The buffer of output RGBA8888 frame
lDstWidth	[in]	The width(columns) of output frame
lDstHeight	[in]	The height(rows) of output frame
lDstStride	[in]	The line step of output RGBA8888 in [Byte] Unit

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least (lDstWidth*10)*sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. Width, Height of both input image and output image should be greater than 2.

3.1.18. mcvResizeRGBA8888BilinearFromRegion

Description

Resize the an image of RGBA8888 form region using Bilinear Method.

Prototype

```
MInt32 mcvResizeRGBA8888BilinearFromRegion(
        MUInt16 *plTmpBuf,MInt32 buflength,
        MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight,
        MInt32 lSrcStride, MUInt8 *pDst,
        MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStride,
        MInt32 lRegionPositionX,MInt32 lRegionPositionY,
        MInt32 lRegionWidth,MInt32 lRegionHeight);
```

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrc	[in]	The buffer of input RGBA8888 frame
lSrcWidth	[in]	The width(columns) of input frame
lSrcHeight	[in]	The height(rows) of input frame
lSrcStride	[in]	The line step of input RGBA8888 in [Byte] Unit
pDst	[out]	The buffer of output RGBA8888 frame
lDstWidth	[in]	The width(columns) of output frame
lDstHeight	[in]	The height(rows) of output frame

lDstStride	[in]	The line step of output RGBA8888 in [Byte] Unit
lRegionPositionX	[in]	the x position of region in src image to be resized
lRegionPositionY	[in]	the y position of region in src image to be resized
lRegionWidth	[in]	the region width
lRegionHeight	[in]	the region height

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least (lDstWidth*10)*sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. Width, Height of both input image and output image should be greater than 2.
4. Width of both input image and output image should be even.

3.1.19. mcvResizeRGBA8888NearestFromRegion

Description

Resize the an image of RGBA8888 form region using Nearest Method.

Prototype

```
MInt32 mcvResizeRGBA8888NearestFromRegion (
        MUInt16 *plTmpBuf,MInt32 buflength,
        MUInt8 *pSrc, MInt32 lSrcWidth, MInt32 lSrcHeight,
        MInt32 lSrcStride, MUInt8 *pDst,
        MInt32 lDstWidth, MInt32 lDstHeight, MInt32 lDstStride,
        MInt32 lRegionPositionX,MInt32 lRegionPositionY,
        MInt32 lRegionWidth,MInt32 lRegionHeight);
```

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrc	[in]	The buffer of input RGBA8888 frame
lSrcWidth	[in]	The width(columns) of input frame
lSrcHeight	[in]	The height(rows) of input frame
lSrcStride	[in]	The line step of input RGBA8888 in [Byte] Unit
pDst	[out]	The buffer of output RGBA8888 frame
lDstWidth	[in]	The width(columns) of output frame

lDstHeight	[in]	The height(rows) of output frame
lDstStride	[in]	The line step of output RGBA8888 in [Byte] Unit
lRegionPositionX	[in]	the x position of region in src image to be resized
lRegionPositionY	[in]	the y position of region in src image to be resized
lRegionWidth	[in]	the region width
lRegionHeight	[in]	the region height

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. The size of *plTmpBuf* should be at least (lDstWidth*10)*sizeof(MUInt16) bytes.
2. All buffer pointers should not be NULL.
3. Width, Height of both input image and output image should be greater than 2.
4. Width of both input image and output image should be even.

3.1.20. mcvResizeSingleComponentBicubicu8

Description

Bicubic interpolation for SingleComponent..

Prototype

```
MUInt32 mcvResizeSingleComponentBicubicu8(
            MUInt32 *plTmpBuf,MInt32 buflength,
            MUInt8 *pSrc, MUInt32 lSrcW, MUInt32 lSrcH,
            MUInt32 lSrcLB,MUInt8 *pDst, MUInt32 lDstW,
            MUInt32 lDstH, MUInt32 lDstLB);
```

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrc	[in]	The source image.
lSrcW	[in]	The src image width in [pixel] unit.
lSrcH	[in]	The src image height in [pixel] unit.
lSrcLB	[in]	The src data line stride.
pDst	[out]	The dst img
lDstW	[in]	The dst image width in [pixel] unit.
lDstH	[in]	The dst image height in [pixel] unit.

lDstLB	[in]	The dst data line stride.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. pSrc,plTmpBuf,pDst should not be null.
2. lSrcW,lSrcH,lDstW,lDstW should be greater than 4.
3. The size of plTmpBuf should not be less than lDstW*(6*sizeof(MInt32))

3.1.21. mcvResizeNV21Bicubicu8

Description

Resize a NV21 image.

Bicubic interpolation for Y. Nearest interpolation for UV.

Prototype

```
MUInt32 mcvResizeNV21Bicubicu8(
        MUInt32 *plTmpBuf,MInt32 buflength, MUInt8 *pSrcY,
        MUInt32 lSrcStrideY,  MUInt8* pSrcUV, MUInt32 lSrcStrideUV,
        MUInt32 lSrcWidth, MUInt32 lSrcHeight,MUInt8 *pDstY,
        MUInt32 lDstStrideY, MUInt8 *pDstUV, MUInt32 lDstStrideUV,
        MUInt32 lDstWidth, MUInt32 lDstHeight);
```

plTmpBuf	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
buflength	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
pSrcY	[in]	The src Y data.
lSrcStrideY	[in]	The src Y line stride.
pSrcUV	[in]	The src UV data.
lSrcStrideUV	[in]	The src UV line stride.
lSrcWidth	[in]	The src image width in [pixel] unit.
lSrcHeight	[in]	The src image height in [pixel] unit.
pDstY	[out]	The dst Y data.
lDstStrideY	[in]	The dst Y line stride.
pDstUV	[out]	The dst UV data.
lDstStrideUV	[in]	The dst UV line stride.
lDstWidth	[in]	The dst image width in [pixel] unit.
lDstHeight	[in]	The dst image height in [pixel] unit.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. pSrcY,pSrcUV,plTmpBuf,pDstY,pDstUV should not be null..
2. lSrcWidth,lSrcHeight,lDstWidth,lDstHeight should be greater than 4.
3. The size of plTmpBuf should not be less than lDstW*(6*sizeof(MInt32))

3.1.22. mcvResizeI420Bicubicu8

Description

Resize an I420image.

Bicubic interpolation for Y. Nearest interpolation for UV.

Prototype

```
MUInt32 mcvResizeI420Bicubicu8(MUInt32 *plTmpBuf,MInt32 buflength,
        MUInt8 *pSrcY, MUInt32 lSrcStrideY, MUInt8* pSrcU,
        MUInt32 lSrcStrideU,MUInt8* pSrcV, MUInt32 lSrcStrideV,
        MUInt32 lSrcWidth, MUInt32 lSrcHeight,MUInt8 *pDstY,
        MUInt32 lDstStrideY, MUInt8 *pDstU, MUInt32 lDstStrideU,
        MUInt8 *pDstV, MUInt32 lDstStrideV,
        MUInt32 lDstWidth, MUInt32 lDstHeight);
```

`plTmpBuf`	[in]	The tmp allocated memory, used for store coordinates. Should align memory.
`buflength`	[in]	The size of *plTmpBuf* in [Byte] Unit, used for internal checking.
`pSrcY`	[in]	The src Y data.
`lSrcStrideY`	[in]	The src Y line stride.
`pSrcU`	[in]	The src U data.
`lSrcStrideU`	[in]	The src U line stride.
`pSrcV`	[in]	The src V data.
`lSrcStrideV`	[in]	The src V line stride.
`lSrcWidth`	[in]	The src image width in [pixel] unit.
`lSrcHeight`	[in]	The src image height in [pixel] unit.
`pDstY`	[out]	The dst Y data.
`lDstStrideY`	[in]	The dst Y line stride.
`pDstU`	[out]	The dst U data.
`lDstStrideU`	[in]	The dst U line stride.
`pDstV`	[out]	The dst V data.
`lDstStrideV`	[in]	The dst V line stride.

lDstWidth	[in]	The dst image width in [pixel] unit.
lDstHeight	[in]	The dst image height in [pixel] unit.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. pSrcY,pSrcU,pSrcV,plTmpBuf,pDstY,pDstU,pDstV should not be null..
2. lSrcWidth,lSrcHeight,lDstWidth,lDstHeight should be greater than 4.
3. The size of plTmpBuf should not be less than lDstW*(6*sizeof(MInt32)).

3.1.23. mcvWarpAffineSingleComponentu8

Description

WarpAffine an SingleComponent Image.

Prototype

```
MUInt32 mcvWarpAffineSingleComponentu8(
        MUInt8 *pSrc,MUInt32 lSrcWidth,MUInt32 lSrcHeight,
        MUInt32 lSrcLineStep, MUInt8 *pDst,MUInt32 lDstWidth,
        MUInt32 lDstHeight,MUInt32 lDstLineStep,MFloat *rotMat);
```

pSrc	[in]	The source image.
lSrcWidth	[in]	The src image width in [pixel] unit.
lSrcHeight	[in]	The src image height in [pixel] unit.
lSrcLineStep	[in]	The src data line stride.
pDst	[out]	The dst img
lDstWidth	[in]	The dst image width in [pixel] unit.
lDstHeight	[in]	The dst image height in [pixel] unit.
lDstLineStep	[in]	The dst data line stride.
rotMat	[in]	The transformation matrix.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. pSrc,pDst,rotMat should not be null..
2. lSrcWidth,lSrcHeight,lDstWidth,lDstHeight should be greater than 4.
3. The rotMat should be set as follows:

rotMat_[0] = (MFloat)cos(angle*0.017453292519943295769236907684886l);
rotMat_[1] = (MFloat)(-sin(angle*0.017453292519943295769236907684886l));
rotMat_[2] = xoffset;
rotMat_[3] = (MFloat)sin(angle*0.017453292519943295769236907684886l);
rotMat_[4] = (MFloat)cos(angle*0.017453292519943295769236907684886l);
rotMat_[5] = yoffset;
angle: rotation angle(clockwise).
xoffset,yoffset：translation offset.

3.1.24. mcvWarpAffineNV21u8

Description

WarpAffine a NV21 image.

Prototype

```
MUInt32 mcvWarpAffineNV21u8(MUInt8 *pSrcY, MUInt32 lSrcStrideY,
        MUInt8* pSrcUV, MUInt32 lSrcStrideUV, MUInt32 lSrcWidth,
        MUInt32 lSrcHeight, MUInt8 *pDstY, MUInt32 lDstStrideY,
        MUInt8 *pDstUV, MUInt32 lDstStrideUV,  MUInt32 lDstWidth,
        MUInt32 lDstHeight,MFloat *rotMat);
```

`pSrcY`	[in]	The src Y data.
`lSrcStrideY`	[in]	The src Y line stride.
`pSrcUV`	[in]	The src UV data.
`lSrcStrideUV`	[in]	The src UV line stride.
`lSrcWidth`	[in]	The src image width in [pixel] unit.
`lSrcHeight`	[in]	The src image height in [pixel] unit.
`pDstY`	[out]	The dst Y data.
`lDstStrideY`	[in]	The dst Y line stride.
`pDstUV`	[out]	The dst UV data.
`lDstStrideUV`	[in]	The dst UV line stride.
`lDstWidth`	[in]	The dst image width in [pixel] unit.
`lDstHeight`	[in]	The dst image height in [pixel] unit.
`rotMat`	[in]	The transformation matrix.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. pSrcY,pSrcUV,pDstY,pDstUV,rotMat should not be null..
2. lSrcWidth,lSrcHeight,lDstWidth,lDstHeight should be greater than 4.

3. The rotMat should be set as follows:

rotMat_[0] = (MFloat)cos(angle*0.017453292519943295769236907684886l);

rotMat_[1] = (MFloat)(-sin(angle*0.017453292519943295769236907684886l));

rotMat_[2] = xoffset;

rotMat_[3] = (MFloat)sin(angle*0.017453292519943295769236907684886l);

rotMat_[4] = (MFloat)cos(angle*0.017453292519943295769236907684886l);

rotMat_[5] = yoffset;

angle: rotation angle(clockwise).

xoffset,yoffset：translation offset.

3.1.25. mcvWarpAffineI420u8

Description

WarpAffine an I420 Image.

Prototype

```
MUInt32 mcvWarpAffineI420u8(MUInt8 *pSrcY, MUInt32 lSrcStrideY,
        MUInt8* pSrcU, MUInt32 lSrcStrideU, MUInt8* pSrcV,
        MUInt32 lSrcStrideV,  MUInt32 lSrcWidth,MUInt32 lSrcHeight,
        MUInt8 *pDstY, MUInt32 lDstStrideY,  MUInt8 *pDstU,
        MUInt32 lDstStrideU, MUInt8 *pDstV, MUInt32 lDstStrideV,
        MUInt32 lDstWidth, MUInt32 lDstHeight, MFloat *rotMat);
```

`pSrcY`	[in]	The src Y data.
`lSrcStrideY`	[in]	The src Y line stride.
`pSrcU`	[in]	The src U data.
`lSrcStrideU`	[in]	The src U line stride.
`pSrcV`	[in]	The src V data.
`lSrcStrideV`	[in]	The src V line stride.
`lSrcWidth`	[in]	The src image width in [pixel] unit.
`lSrcHeight`	[in]	The src image height in [pixel] unit.
`pDstY`	[out]	The dst Y data.
`lDstStrideY`	[in]	The dst Y line stride.
`pDstU`	[out]	The dst U data.
`lDstStrideU`	[in]	The dst U line stride.
`pDstV`	[out]	The dst V data.
`lDstStrideV`	[in]	The dst V line stride.
`lDstWidth`	[in]	The dst image width in [pixel] unit.
`lDstHeight`	[in]	The dst image height in [pixel] unit.
`rotMat`	[in]	The transformation matrix.

Return value

MCV_OK

MCV_NULL_POINTER

MCV_INVALID_PARAM

Notes

1. pSrcY,pSrcU,pSrcV,pDstY,pDstU,pSrcV,rotMat should not be null..
2. lSrcWidth,lSrcHeight,lDstWidth,lDstHeight should be greater than 4.
3. The rotMat should be set as follows:

rotMat_[0] = (MFloat)cos(angle*0.017453292519943295769236907684886l);

rotMat_[1] = (MFloat)(-sin(angle*0.017453292519943295769236907684886l));

rotMat_[2] = xoffset;

rotMat_[3] = (MFloat)sin(angle*0.017453292519943295769236907684886l);

rotMat_[4] = (MFloat)cos(angle*0.017453292519943295769236907684886l);

rotMat_[5] = yoffset;

angle: rotation angle(clockwise).

xoffset,yoffset： translation offset.

3.2. Resize Multi-thread API

3.2.1. mcvResizeMultiThreadsInit

Description

Initialize The Resize Engine.

Prototype

```
MHandle mcvResizeMultiThreadsInit();
```

Parameters

Return value

A Handle of the engine : valid if non-zero.

Notes

3.2.2. mcvResizeMultiThreadsProcess

Description

Do The Resize.

Prototype

```
MInt32 mcvResizeMultiThreadsProcess(MHandle mcvResizeHandle,LPASVLOFFSCREEN
srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`mcvResizeHandle`	[in]	Engine handle returned by mcvResizeMultiThreadsInit ().
`srcImg`	[in]	Source image to be resize.
`dstImg`	[out]	converted output image.

Return value

MCV_NULL_POINTER
MCV_INVALID_PARAM
MCV_OK.

Notes

1. Supported color format list:

image format
ASVL_PAF_NV21

2. src img and dst img uv plane should be continuously followed by y plane in memory

3.2.3. mcvResizeMultiThreadsUninit

Description

UnInitialize the Resize Engine

Prototype

```
MInt32 mcvResizeMultiThreadsUninit(MHandle mcvResizeHandle);
```

Parameters

`mcvResizeHandle`	[in]	Engine handle returned by mcvResizeMultiThreadsInit ()

Return value

MCV_NULL_POINTER
MCV_UNEXPECTED_ERR
MCV_OK.

Notes

3.3. AbsDiff API

3.3.1. mcvAbsDiffu32

Description

Computes the per-element absolute difference between two uint32 image.

Prototype

```
MInt32 mcvAbsDiffu32(MUInt32* pSrc1, MUInt32* pSrc2, MUInt32* pDst, MInt32
lWidth, MInt32 lHeight, MInt32 lLineBytes);
```

Parameters

pSrc1	[in]	First input image
pSrc2	[in]	Second input image, must has the same size as pSrc1.
pDst	[out]	Output image, must has the same size as pSrc1.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.
lLineBytes	[in]	Image length of a row in bytes.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pSrc2 and pDst must has the same size as pSrc1.
2. lWidth, lHeight and lLineBytes must be a multiple of 2.

3.3.2. mcvAbsDiffs32

Description

Computes the per-element absolute difference between two int32 image.

Prototype

```
MInt32 mcvAbsDiff32(MInt32* pSrc1, MInt32* pSrc2, MInt32* pDst, MInt32 lWidth,
MInt32 lHeight, MInt32 lLineBytes);
```

Parameters

pSrc1	[in]	First input image
pSrc2	[in]	Second input image, must has the same size as pSrc1.
pDst	[out]	Output image, must has the same size as pSrc1.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.
lLineBytes	[in]	Image length of a row in bytes.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pSrc2 and pDst must has the same size as pSrc1.
2. lWidth, lHeight and lLineBytes must be a multiple of 2.

3.3.3. mcvAbsDiffu8

Description

Computes the per-element absolute difference between two byte image.

Prototype

```
MInt32 mcvAbsDiffu8(MByte* pSrc1, MByte* pSrc2, MByte* pDst, MInt32 lWidth,
MInt32 lHeight, MInt32 lLineBytes);
```

Parameters

`pSrc1`	[in]	First input image
`pSrc2`	[in]	Second input image, must has the same size as pSrc1.
`pDst`	[out]	Output image, must has the same size as pSrc1.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.
`lLineBytes`	[in]	Image length of a row in bytes.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. `pSrc2 and pDst must has the same size as pSrc1.`
2. `lWidth, lHeight and lLineBytes must be a multiple of 2.`

3.3.4. mcvAbsDiffVs32

Description

Computes the per-element absolute difference between one int32 image and one value.

Prototype

```
MInt32 mcvAbsDiffVs32(MInt32* pSrc, MInt32 lValue, MInt32* pDst, MInt32 lWidth,
MInt32 lHeight, MInt32 lLineBytes);
```

Parameters

`pSrc`	[in]	Input image
`lValue`	[in]	Input Value
`pDst`	[out]	Output image, must has the same size as pSrc1.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.
`lLineBytes`	[in]	Image length of a row in bytes.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst must has the same size as pSrc.
2. lWidth, lHeight and lLineBytes must be a multiple of 2.

3.3.5. mcvAbsDiffVf32

Description

Computes the per-element absolute difference between one float image and one value.

Prototype

```
MInt32 mcvAbsDiffVf32(MFloat* pSrc, MFloat lValue, MFloat* pDst, MInt32 lWidth,
MInt32 lHeight, MInt32 lLineBytes);
```

Parameters

pSrc	[in]	Input image
lValue	[in]	Input Value
pDst	[out]	Output image, must has the same size as pSrc1.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.
lLineBytes	[in]	Image length of a row in bytes.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst must has the same size as pSrc.
2. lWidth, lHeight and lLineBytes must be a multiple of 2.

3.4. Filter API

3.4.1. mcvFilterThresholdu8

Description

Binarizes a grayscale image based on a threshold value.

Prototype

```
MInt32 mcvFilterThresholdu8(MByte* pSrc, MByte* pDst, MInt32 lThreshold, MInt32
lWidth, MInt32 lHeight, MInt32 lLineBytes);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
lThreshold	[in]	Input threshold.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.
lLineBytes	[in]	Image length of a row in bytes.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst must has the same size as pSrc.
2. lWidth, lHeight and lLineBytes must be a multiple of 2.

3.4.2. mcvFilterDilate3x3u8

Description

Dilate a grayscale image by taking the local maxima of 3x3 neighborhood window.

Prototype

```
MInt32 mcvFilterDilate3x3u8(MByte* pSrc, MByte* pDst, MByte* pTmp, MInt32
lWidth, MInt32 lHeight);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
pTmp	[in]	Temporary buffer for use, must has the same size as pSrc.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst and pTmp must has the same size as pSrc.
2. lWidth and lHeight must be a multiple of 2.

3.4.3. mcvFilterGaussian5x5u8

Description

Blur a greyscale image with 5x5 Gaussian filter.

Prototype

```
MInt32 mcvFilterGaussian5x5u8(MByte* pSrc, MByte* pDst, MByte* pTmp, MInt32 lWidth, MInt32 lHeight);
```

Parameters

`pSrc`	[in]	Input image
`pDst`	[out]	Output image, must has the same size as pSrc.
`pTmp`	[in]	Temporary buffer for use, must has the same size as pSrc.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. `pDst and pTmp must has the same size as pSrc.`
2. `lWidth and lHeight must be a multiple of 2.`

3.4.4. mcvFilterGaussian7x7f32

Description

Blur a greyscale image with 7x7 Gaussian filter(Separate mode). Kernel:
`0.0126f, 0.0788f, 0.2373f, 0.3426f, 0.2373f, 0.0788f, 0.0126f`

Prototype

```
MInt32 mcvFilterGaussian7x7f32(MFloat* pSrc, MFloat* pDst, MFloat* pTmp, MInt32 lWidth, MInt32 lHeight);
```

Parameters

`pSrc`	[in]	Input image
`pDst`	[out]	Output image, must has the same size as pSrc.
`pTmp`	[in]	Temporary buffer for use, must has the same size as pSrc.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst and pTmp must has the same size as pSrc.
2. lWidth and lHeight must be >= 6.

3.4.5. mcvFilterGaussian7x7f32_2D

Description

Blur a greyscale image with 7x7 Gaussian filter(2D mode). Kernel:

```
0.000158,0.000990,0.002980,0.004304,0.002980,0.000990,0.000158,
0.000990,0.006214,0.018706,0.027009,0.018706,0.006214,0.000990,
0.002980,0.018706,0.056309,0.081305,0.056309,0.018706,0.002980,
0.004304,0.027009,0.081305,0.117396,0.081305,0.027009,0.004304,
0.002980,0.018706,0.056309,0.081305,0.056309,0.018706,0.002980,
0.000990,0.006214,0.018706,0.027009,0.018706,0.006214,0.000990,
0.000158,0.000990,0.002980,0.004304,0.002980,0.000990,0.000158
```

Prototype

```
MInt32 mcvFilterGaussian7x7f32_2D(MFloat* pSrc, MFloat* pDst, MInt32 lWidth,
MInt32 lHeight);
```

Parameters

`pSrc`	[in]	Input image
`pDst`	[out]	Output image, must has the same size as pSrc.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst and pTmp must has the same size as pSrc.
   lWidth and lHeight must be >= 6.

3.4.6. mcvFilterGaussian7x7u16

Description

Blur a greyscale image with 7x7 Gaussian filter(Separate mode). Kernel:

```
0.0126, 0.0788, 0.2373, 0.3426, 0.2373, 0.0788, 0.0126
```

Prototype

```
MInt32 mcvFilterGaussian7x7u16(MUInt16* pSrc, MUInt16* pDst, MUInt16* pTmp,
MInt32 lWidth, MInt32 lHeight);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
pTmp	[in]	Temporary buffer for use, must has the same size as pSrc.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst and pTmp must has the same size as pSrc.
2. lWidth and lHeight must be >= 6.

3.4.7. mcvFilterGaussian7x7u16_2D

Description

Blur a greyscale image with 7x7 Gaussian filter(2D mode). Kernel:

```
0.000158,0.000990,0.002980,0.004304,0.002980,0.000990,0.000158,
0.000990,0.006214,0.018706,0.027009,0.018706,0.006214,0.000990,
0.002980,0.018706,0.056309,0.081305,0.056309,0.018706,0.002980,
0.004304,0.027009,0.081305,0.117396,0.081305,0.027009,0.004304,
0.002980,0.018706,0.056309,0.081305,0.056309,0.018706,0.002980,
0.000990,0.006214,0.018706,0.027009,0.018706,0.006214,0.000990,
0.000158,0.000990,0.002980,0.004304,0.002980,0.000990,0.000158
```

Prototype

```
MInt32 mcvFilterGaussian7x7u16_2D(MUInt16* pSrc, MUInt16* pDst, MInt32 lWidth,
MInt32 lHeight);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst and pTmp must has the same size as pSrc.
2. lWidth and lHeight must be >= 6.

3.4.8. mcvFilterErode3x3u8

Description

Erode a grayscale image by taking the local minima of 3x3 neighborhood window.

Prototype

```
MInt32 mcvFilterErode3x3u8(MByte* pSrc, MByte* pDst, MByte* pTmp, MInt32 lWidth,
MInt32 lHeight);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
pTmp	[in]	Temporary buffer for use, must has the same size as pSrc.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst and pTmp must has the same size as pSrc.
2. lWidth and lHeight must be a multiple of 2.

3.4.9. mcvFilterMedian3x3u8

Description

Blur a greyscale image with 3x3 Median filter.

Prototype

```
MInt32 mcvFilterMedian3x3u8(MByte* pSrc, MByte* pDst, MByte* pTmp, MInt32
lWidth, MInt32 lHeight);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
pTmp	[in]	Temporary buffer for use, must has the same size as pSrc.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. `pDst and pTmp must has the same size as pSrc.`
2. `lWidth and lHeight must be a multiple of 2.`

3.4.10. mcvFilterBox3x3u8

Description

Smooth a greyscale image with a 3x3 box filter.

Prototype

```
MInt32 mcvFilterBox3x3u8(MByte* pSrc, MByte* pDst, MByte* pTmp, MInt32 lWidth,
MInt32 lHeight);
```

Parameters

`pSrc`	[in]	Input image
`pDst`	[out]	Output image, must has the same size as pSrc.
`pTmp`	[in]	Temporary buffer for use, must has the same size as pSrc.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. `pDst and pTmp must has the same size as pSrc.`
2. `lWidth and lHeight must be a multiple of 2.`
3. `pSrc can be the same with pDst.`

3.4.11. mcvFilterBox3x3u8_2D

Description

Smooth a greyscale image with a 3x3 box filter (2D mode).

Prototype

```
MInt32 mcvFilterBox3x3u8_2D(MByte* pSrc, MByte* pDst,MInt32 lWidth,MInt32
lHeight,MInt32 lStrideSrc,MInt32 lStrideDst);
```

Parameters

`pSrc`	[in]	Input image

pDst	[out]	Output image, must has the same size as pSrc.
lWidth	[in]	Src Image width.
lHeight	[in]	Dst Image Height.
lStrideSrc	[in]	Src Image line stride.
lStrideDst	[in]	Dst Image line stride.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst and pTmp must has the same size as pSrc.
2. lWidth and lHeight must be > 2.
3. lStrideSrc >= lWidth, lStrideDst>= lWidth

3.4.12. mcvFilterBoxu8

Description

Smooth a greyscale (or other component)image with a box filter .

Prototype

```
MInt32 mcvFilterBoxu8(MByte* pSrc,MByte* pDst,MUInt16* pRowBuf,MUInt32
kernelSize,MUInt32 lWidth,MUInt32 lHeight,MUInt32 lStride);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
pRowBuf	[in]	Tmp buffer for internal usage.
kernelSize	[in]	Box filter size,3 for 3x3,5for 5x5,…
lWidth	[in]	Image width.
lHeight	[in]	Image Height.
lStride	[in]	Image buffer line stride.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst == pSrc supported.
2. pRowBuf: must be 4bytes alligned and size of the buffer is >= lWidth*sizeof(MUInt32) + kernelSize *lWidth*sizeof(MUInt16)
3. lWidth and lHeight must be >= 4.
4. kernelSize >= 3 and is odd.

3.4.13. mcvFilterBoxYUYV

Description

Smooth a YUYV image with a (1 << lSmoothLenShift * 1 << lSmoothLenShift) box using integral.

Prototype

```
MInt32 mcvFilterBoxYUYV(MUInt8* pu8YUYV, MUInt32 *pYSum, MUInt32 *pCbSum,
                        MUInt32 *pCrSum, MLong lPitch, MLong lWidth,
                        MLong lHeight, MLong lSmoothLenShift);
```

Parameters

`pu8YUYV`	[in/out]	Input/output YUYV image
`pYSum`	[in]	Temp buffer for Y integral.
`pCbSum`	[in]	Temp buffer for Cb integral.
`pCrSum`	[in]	Temp buffer for Cr integral.
`lPitch`	[in]	Image pitch.
`lWidth`	[in]	Image width
`lHeight`	[in]	Image height
`lSmoothLenShift`	[in]	Smooth Box Length.(if lSmoothLenShift is 2 box is 4x4, 3->9x9 and so on)

Return value

MCV_OK

MCV_INVALID_PARAM

MCV_NULL_POINTER

Notes

1. `The buffer size of pYSum should be (alignBy4(lWidth+1)*(lHeight+1)* sizeof(MUInt32)) bytes.`
2. `The buffer size of pCbSum should be (alignBy4(lWidth/2+1)*(lHeight+1)* sizeof(MUInt32)) bytes.`
3. `The buffer size of pCrSum should be (alignBy4(lWidth/2+1)*(lHeight+1)* sizeof(MUInt32)) bytes.`
4. `alignBy4 mentioned above is ceil the number to 4x(eg. alignBy4(3)= 4, alignBy4(4) = 4, alignBy4(5) = 8).`
5. `Only Pure C version exists, NEON version is not implemented.`

3.4.14. mcvFilterBoxYUYVInplaceLuma

Description

Smooth the Y component of a YUYV image with a (kernelSize * kernelSize) box filter.

Prototype

```
MInt32 mcvFilterBoxYUYVInplaceLuma(MByte* pSrc,MUInt16* pRowBuf,MUInt32
                        kernelSize,MUInt32 lWidth,MUInt32 lHeight,MUInt32
                        lStride);
```

Parameters

pSrc	[in/out]	Input/output YUYV image
pRowBuf	[in]	Temp buffer internal usage.
kernelSize	[in]	The size of box filter,say, 3 for 3x3, 5for 5x5.
lWidth	[in]	Image width.
lHeight	[in]	Image height.
lStride	[in]	Image buffer line stride.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. lWidth and lHeight must be >= 4 and lWidth must be even.
2. kernelSize >= 3 and kernelSize is odd.
3. The size of pRowBuf >=
   (kernelSize) * lWidth * sizeof(MUInt16) + lWidth * sizeof(MUInt32)

3.4.15. mcvFilterSobel3x3u8

Description

Apply Sobel filter on a greyscale image.

Prototype

```
MInt32 mcvFilterSobel3x3u8(MByte* pSrc, MByte* pDst, MInt32 lWidth, MInt32
lHeight);
```

Parameters

pSrc	[in]	Input image
pDst	[out]	Output image, must has the same size as pSrc.
lWidth	[in]	Image width.
lHeight	[in]	Image Height.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. pDst must has the same size as pSrc.
2. lWidth and lHeight must be a multiple of 2.

3.4.16. mcvPyrDownGauss5x5u8c1

Description

Pyramid down a single channel image using guass 5x5, use 1/16[1 4 6 4 1] for both horizontal and vertical directions.

Prototype

```
MInt32 mcvPyrDownGauss5x5u8c1(MByte *pTmp,MByte *pSrc,MInt32 srcWidth,
                      MInt32 srcHeight,MInt32 srcStep,
                       MByte *pDst,MInt32 dstStep);
```

Parameters

pTmp	[in]	Tmp line buffer(5 lines), for internal use.
pSrc	[in]	Input image.
srcWidth	[in]	Input image width.
srcHeight	[in]	Input image height.
srcStep	[in]	Input image buffer line stride in [BYTE] unit.
pDst	[out]	Output Image.
dstStep	[in]	Output Image buffer line stride in [BYTE] unit.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. The size of buffer *pTmp* is 5 * *srcStep* * sizeof(char).
   The size of buffer *pSrc* is *srcStep* * *srcHeight* * sizeof(char).
   The size of buffer *pDst* is *dstStep* * ((*srcHeight* + 1)/2) * sizeof(char).
2. *srcStep* >= *srcWidth* and is a multiple of 2, *pSrc* must be at least 2Bytes aligned.
   Valid area of *pDst* is ((*srcWidth* + 1)/2) x ((*srcHeight* + 1)/2),
   dstStep >= ((*srcWidth* + 1)/2).
3. *srcWidth and srcHeight* must >= 32.

3.4.17. mcvConv_32_5_i32

Description

Convolution an 32x32 matrix by 5x5 filter matrix.

Prototype

```
MUInt32 mcvConv_32_5_i32(MInt32 input[32][32], MInt32 output[28][28],
                 MInt32 filter[5][5]);
```

Parameters

input	[in]	Input 32x32 matrix.
output	[in]	Output 28x28 matrix.

`filter`	[in]	5x5 filter matrix.

Return value

MCV_OK
MCV_NULL_POINTER

3.4.18. mcvConv_14_5_i32

Description

Convolution an 14x14 matrix by 5x5 filter matrix.

Prototype

```
MUInt32 mcvConv_14_5_i32(MInt32 input[14][14], MInt32 output[10][10],
                    MInt32 filter[5][5]);
```

Parameters

`input`	[in]	Input 14x14 matrix.
`output`	[in]	Output 10x10 matrix.
`filter`	[in]	5x5 filter matrix.

Return value

MCV_OK
MCV_NULL_POINTER

3.5. SetElements API

3.5.1. mcvSetElementsu8

Description

Sets every element of an uint8_t array to a given value.

Prototype

```
MInt32 mcvSetElementsu8(MByte* pSrc, MInt32 lWidth, MInt32 lHeight, MInt32
lChannel, MByte lValue);
```

Parameters

`pSrc`	[in/out]	Input image
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.
`lChannel`	[in]	Channel of one pixel, such as RGB has 3 channel.
`lValue`	[in]	The input uint8_t value.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. `lWidth and lHeight must be a multiple of 2.`

3.5.2. mcvSetElementss32

Description

Sets every element of an int32_t array to a given value.

Prototype

```
MInt32 mcvSetElementss32(MInt32* pSrc, MInt32 lWidth, MInt32 lHeight, MInt32 lChannel, MInt32 lValue);
```

Parameters

`pSrc`	[in/out]	Input image
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height.
`lChannel`	[in]	Channel of one pixel, such as RGB has 3 channel.
`lValue`	[in]	The input int32_t value.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. `lWidth and lHeight must be a multiple of 2.`

3.6. Math API

3.6.1. mcvFastSqrts64

Description

Calulate the sqrt of a int64 value.

Prototype

```
MInt32 mcvFastSqrts64(MInt64 a);
```

Parameters

`a`	[in]	The input value.

Return value

MCV_INVALID_PARAM	if	a < 0;
The sqrt of a	if	a>= 0.

Notes

1. 1% error more or less compared with (int)sqrt(a);.

3.6.2. mcvFastSqrts32

Description

Calulate the sqrt of a int32 value.

Prototype

```
MInt32 mcvFastSqrts32(MInt32 x);
```

Parameters

x	[in]	The input value.

Return value

MCV_NULL_POINTER	if	x < 0;
The sqrt of x	if	x>= 0.

Notes

1. 1% error more or less compared with (int)sqrt(x);.

3.6.3. mcvDotProducts8

Description

Dot product of two vectors.

Prototype

```
MLong mcvDotProducts8(MUInt8* a,MUInt8*  b,MUInt32 absize );
```

Parameters

a	[in]	Vector a.
b	[in]	Vector b.
absize	[in]	The length of vectors.

Return value

Dot product value of vector{a[0],a[1],a[2],...} and vector{b[0],b[1],b[2],...} ;
MCV_NULL_POINTER;

Notes

1. Array a and b are of the same length.
2. absize ranges from 1 to 30000 inclusively.
3. All buffer pointers should not be NULL.
4. Calculation formula: return value = Σ a[i]*b[i].

3.6.4. mcvBitCountu8

Description

Get the bit ‘1’s of a bit stream.

Prototype

```
MLong mcvBitCountu8 (MUInt8*  src,MLong  srcLength);
```

Parameters

`src`	[in]	Input bit stream.
`srcLength`	[in]	Stream length in [byte] unit.

Return value

Number of bit ‘1’s in stream *src* ;
MCV_NULL_POINTER;

Notes

1. srcLength ranges from 1 to 0x00ffffff inclusively.
2. All buffer pointers should not be NULL.

3.6.5. mcvBitwiseOru8

Description

Performs per-element bitwise-OR operation on two 8-bit single channel images.

Prototype

```
MInt32 mcvBitwiseOru8(MByte* pSrc1, MByte* pSrc2, MByte* pDst,
            MInt32 iwidth,MInt32 iheight,
            MInt32 iStrideSrc1,MInt32 iStrideSrc2,MInt32 iStrideDst);
```

Parameters

`pSrc1`	[in]	First input image.
`pSrc2`	[in]	Second input image.
`pDst`	[out]	Output image.
`iWidth`	[in]	Image width.
`iHeight`	[in]	Image height.
`iStrideSrc1`	[in]	First input image line stride.
`iStrideSrc2`	[in]	Second input image line stride.
`iStrideDst`	[in]	Output image line stride.

Return value

MCV_OK
MCV_INVALID_PARAM

MCV_NULL_POINTER

Notes

1. All images pointers should not be NULL.
2. All images have the same size.

3.6.6. mcvSqrtf32

Description

Fast sqrt approx.

Prototype

```
MFloat mcvSqrtf32(MFloat fX);
```

Parameters

`fX`	[in]	Input number.

Return value

The result of sqrt.

Notes

1. fx should be >= 0.

3.6.7. mcvSqrtVectorf32

Description

Fast vector sqrt approx.

Prototype

```
MInt32 mcvSqrtVectorf32(MFloat* pSrc, MFloat* pDst, MInt32 lLength);
```

Parameters

`pSrc`	[in]	Input singel precision numbers.
`pDst`	[out]	Output singel precision numbers.
`lLength`	[in]	The length of the numbers.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. Length should be >= 0.
3. The result of C and NEON version maybe different. NEON 版本计算的近似平方根。C 语言版本用的 C 库函数，仅供参考。

3.6.8. mcvInvSqrtf32

Description

Fast reciprocal sqrt approx.

Prototype

```
MFloat mcvInvSqrtf32(MFloat fX);
```

Parameters

`fX`	[in]	Input number.

Return value

The result of reciprocal sqart.

Notes

1. fx should be >= 0.
2. The result of C and NEON version maybe different. NEON 版本计算的近似平方根倒数。C 语言版本用的 C 库函数，仅供参考。

3.6.9. mcvInvSqrtVectorf32

Description

Fast vector reciprocal sqrt approx.

Prototype

```
MInt32 mcvInvSqrtVectorf32(MFloat* pSrc, MFloat* pDst, MInt32 lLength);
```

Parameters

`pSrc`	[in]	Input singel precision numbers.
`pDst`	[out]	Output singel precision numbers.
`lLength`	[in]	The length of the numbers.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. Length should be >= 0.
3. The result of C and NEON version maybe different. NEON 版本计算的近似平方根倒数。C 语言版本用的 C 库函数，仅供参考。

3.6.10. mcvDivf32

Description

Fast div approx.

Prototype

```
MFloat mcvDivf32(MFloat fDividend, MFloat fDivisor);
```

Parameters

`fDividend`	[in]	Input dividend number.
`fDivisor`	[in]	Input divisor number.

Return value

The result of div.

Notes

3.6.11. mcvVectorDivf32

Description

Fast vector div approx.

Prototype

```
MInt32 mcvVectorDivf32(MFloat* pDividend, MFloat* pQuotient, MFloat fDivisor,
                       MInt32 lLength)
```

Parameters

`pDividend`	[in]	Input singel precision dividend numbers.
`pQuotient`	[out]	Output singel precision quotient numbers.
`fDivisor`	[in]	Input single precision divisor numbers.
`lLength`	[in]	The length of the numbers.

Return value

MCV_OK

MCV_INVALID_PARAM

MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. Length should be >= 0.
3. The result of C and NEON version maybe different. NEON 版本计算使用近似平方根倒数的方法。C 语言版本用/，仅供参考。

3.6.12. mcvVectorDiffNorm2s32

Description

Calculate the 2-norm value of the difference between two vectors. The elements of both vectors are signed 32bit .

Output = sqrt(Σ (vec1[x]-vec2[x]) (vec1[x]-vec2[x]))

Prototype

```
MDouble mcvVectorDiffNorm2s32(MInt32 *vec1,MInt32 *vec2,MInt32 len)
```

Parameters

`vec1`	[in]	Vector1.
`Vec2`	[in]	Vector2.
`len`	[in]	Vector length.

Return value

The 2-norm of the difference between two vectors.

Notes

1. Length should be >= 0.

3.6.13. mcvVectorDiffNorm2u32

Description

Calculate the 2-norm value of the difference between two vectors. The elements of both vectors are unsigned 32bit .

Output = sqrt(Σ (vec1[x]-vec2[x]) (vec1[x]-vec2[x]))

Prototype

```
MDouble mcvVectorDiffNorm2u32(MUInt32 *vec1,MUInt32 *vec2,MInt32 len)
```

Parameters

`vec1`	[in]	Vector1.
`Vec2`	[in]	Vector2.
`len`	[in]	Vector length.

Return value

The 2-norm of the difference between two vectors.

Notes

1. Length should be >= 0.

3.6.14. mcvVectorDiffNorm2f32

Description

Calculate the 2-norm value of the difference between two vectors. The elements of both vectors are 32bit float.

Output = sqrt(Σ (vec1[x]-vec2[x]) (vec1[x]-vec2[x]))

Prototype

```
MDouble mcvVectorDiffNorm2f32(MFloat *vec1, MFloat *vec2,MInt32 len)
```

Parameters

`vec1`	[in]	Vector1.
`Vec2`	[in]	Vector2.
`len`	[in]	Vector length.

Return value

The 2-norm of the difference between two vectors.

Notes

1. Length should be >= 0.

3.6.15. mcvVectorDiffNorm2Fasts16

Description

Calculate the 2-norm value of the difference between two vectors. The elements of both vectors are signed 16bit .

Output = sqrt(Σ (vec1[x]-vec2[x]) (vec1[x]-vec2[x]))

Prototype

```
MDouble mcvVectorDiffNorm2Fasts16(MInt16 *vec1,MInt16 *vec2,MInt32 len)
```

Parameters

`vec1`	[in]	Vector1.
`Vec2`	[in]	Vector2.
`len`	[in]	Vector length.

Return value

The 2-norm of the difference between two vectors.

Notes

1. Length should be >= 0.

3.6.16. mcvVectorDiffNorm2Fastu16

Description

Calculate the 2-norm value of the difference between two vectors. The elements of both vectors are unsigned 16bit .

Output = sqrt(Σ (vec1[x]-vec2[x]) (vec1[x]-vec2[x]))

Prototype

```
MDouble mcvVectorDiffNorm2Fastu16(MUInt16 *vec1,MUInt16 *vec2,MInt32 len)
```

Parameters

`vec1`	[in]	Vector1.
`Vec2`	[in]	Vector2.

`len`	[in]	Vector length.

Return value

The 2-norm of the difference between two vectors.

Notes

1. Length should be >= 0.

3.6.17. mcvVectorDiffNorm2Fasts8

Description

Calculate the 2-norm value of the difference between two vectors. The elements of both vectors are signed 8bit .

Output = sqrt(Σ (vec1[x]-vec2[x]) (vec1[x]-vec2[x]))

Prototype

```
MDouble mcvVectorDiffNorm2Fasts8(MInt8 *vec1, MInt8 *vec2,MInt32 len)
```

Parameters

`vec1`	[in]	Vector1.
`Vec2`	[in]	Vector2.
`len`	[in]	Vector length.

Return value

The 2-norm of the difference between two vectors.

Notes

1. Length should be >= 0.
2. The result is not guaranteed because of overflow when Length > 66051 and big difference exists between the two vectors.

3.6.18. mcvVectorDiffNorm2Fastu8

Description

Calculate the 2-norm value of the difference between two vectors. The elements of both vectors are unsigned 8bit .

Output = sqrt(Σ (vec1[x]-vec2[x]) (vec1[x]-vec2[x]))

Prototype

```
MDouble mcvVectorDiffNorm2Fastu8(MUInt8 *vec1,MUInt8 *vec2,MInt32 len)
```

Parameters

`vec1`	[in]	Vector1.
`Vec2`	[in]	Vector2.
`len`	[in]	Vector length.

Return value

The 2-norm of the difference between two vectors.

Notes

1. Length should be >= 0.
2. The result is not guaranteed because of overflow when Length > 66051 and big difference exists between the two vectors.

3.6.19. mcvMatrixAddMatrix_f32

Description

Calculate the matrix add operation: M1 + M2. The element is float32.

Prototype

```
MInt32 mcvMatrixAddMatrix_f32(MFloat *matrixOut,MFloat *matrix1,MFloat
                        *matrix2,MInt32 row,MInt32 column)
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrix1`	[in]	Matrix1.
`Matrix2`	[in]	Matrix2.
`row`	[in]	Number of rows of both Matrixs.
`column`	[in]	Number of columns of both Matrixs.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. Inplace add supported
2. Row >= 0; column >= 0;

3.6.20. mcvMatrixSubMatrix_f32

Description

Calculate the matrix substraction operation: M1 - M2. The element is float32.

Prototype

```
MInt32 mcvMatrixSubMatrix_f32(MFloat *matrixOut,MFloat *matrix1,MFloat
                        *matrix2,MInt32 row,MInt32 column)
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrix1`	[in]	Matrix1.
`Matrix2`	[in]	Matrix2.

`row`	[in]	Number of rows of both Matrixs.
`column`	[in]	Number of columns of both Matrixs.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. Inplace sub supported
2. Row >= 0; column >= 0;

3.6.21. mcvMatrixMulScalar_f32

Description

Calculate the matrix scalar multiply operation: λ M1. The element is float32.

Prototype

```
MInt32 mcvMatrixMulScalar_f32(MFloat *matrixOut,MFloat *matrixIn,MFloat
                        lamda,MInt32 row,MInt32 column)
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrixIn`	[in]	Input Matrix.
`lamda`	[in]	The scalar.
`row`	[in]	Number of rows of Matrixs.
`column`	[in]	Number of columns of Matrixs.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. Inplace multiply supported
2. Row >= 0; column >= 0;

3.6.22. mcvMatrixMulMatrixRowMajor_f32

Description

Calculate the matrix multiply operation: M1 x M2. The element is float32.

Mout[j + i* column2] = Σ(M1[k + i * column1] * M2[j + k * column2])

Prototype

```
MInt32 mcvMatrixMulMatrixRowMajor_f32(MFloat *matrixOut,MFloat *matrix1,MFloat
                      *matrix2,MInt32 row1,MInt32 column1,MInt32 column2)
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrix1`	[in]	Input Matrix 1(M1).
`matrix2`	[in]	Input Matrix 1(M2)..
`row1`	[in]	The number of rows of M1.
`column1`	[in]	The number of columns of M1(also the number of rows of M2).
`column2`	[in]	The number of columns of M2.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. row1>= 0; column1>= 0; column2>=0

3.6.23. mcvMatrixMulMatrixRowMajor_s32

Description

Calculate the matrix multiply operation: M1 x M2. The element is int32.

Mout[j + i* column2] = Σ(M1[k + i * column1] * M2[j + k * column2])

Prototype

```
MInt32 mcvMatrixMulMatrixRowMajor_f32(MInt32 *matrixOut, MInt32 *matrix1,
                      MInt32 *matrix2,MInt32 row1,MInt32 column1,MInt32 column2)
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrix1`	[in]	Input Matrix 1(M1).
`matrix2`	[in]	Input Matrix 1(M2)..
`row1`	[in]	The number of rows of M1.
`column1`	[in]	The number of columns of M1(also the number of rows of M2).
`column2`	[in]	The number of columns of M2.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. row1>= 0; column1>= 0; column2>=0

3.6.24. mcvMatrixMulMatrixColMajor_f32

Description

Calculate the matrix multiply operation: M1 x M2. The element is float32.

- Column major

Prototype

```
MInt32 mcvMatrixMulMatrixColMajor_f32(MFloat *matrixOut,MFloat *matrix1,MFloat
                      *matrix2,MInt32 row1,MInt32 column1,MInt32 column2)
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrix1`	[in]	Input Matrix 1(M1).
`matrix2`	[in]	Input Matrix 1(M2)..
`row1`	[in]	The number of rows of M1.
`column1`	[in]	The number of columns of M1(also the number of rows of M2).
`column2`	[in]	The number of columns of M2.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. row1>= 0; column1>= 0; column2>=0
2. Column major: that is to say, the elements in the buffer is arranged by the order of column: a00,a10,a20,a30,…,a01,a11,a21,…
   Notice that aij indicates the element on the i*th* row and j*th* column.

3.6.25. mcvMatrixMulMatrixColMajor_s32

Description

Calculate the matrix multiply operation: M1 x M2. The element is int32.

- Column major

Prototype

```
MInt32 mcvMatrixMulMatrixColMajor_s32(MInt32 *matrixOut, MInt32 *matrix1,
                      MInt32 *matrix2,MInt32 row1,MInt32 column1,MInt32 column2)
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrix1`	[in]	Input Matrix 1(M1).
`matrix2`	[in]	Input Matrix 1(M2)..
`row1`	[in]	The number of rows of M1.
`column1`	[in]	The number of columns of M1(also the number of rows of M2).

column2	[in]	The number of columns of M2.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. row1>= 0; column1>= 0; column2>=0
2. Column major: that is to say, the elements in the buffer is arranged by the order of column: a00,a10,a20,a30,…,a01,a11,a21,…
   Notice that aij indicates the element on the i*th* row and j*th* column.

3.6.26. mcvMatrixMulAddRowMajor_f32

Description

Calculate the matrix multiply add operation: M1 x M2 + M3. The element is float32.

Prototype

```
MInt32 mcvMatrixMulAddRowMajor_f32(MFloat *matrixOut,MFloat *matrix1,MFloat
                      *matrix2,MFloat *matrix3,MInt32 row1,MInt32
                      column1,MInt32 column2)
```

Parameters

matrixOut	[out]	The result matrix.
matrix1	[in]	Input Matrix 1(M1).
matrix2	[in]	Input Matrix 2(M2)..
matrix3	[in]	Input Matrix 3(M3)..
row1	[in]	The number of rows of M1.
column1	[in]	The number of columns of M1(also the number of rows of M2).
column2	[in]	The number of columns of M2.

Return value

MCV_NULL_POINTER.

MCV_OK

Notes

1. matrix3 is the same size as matrixOut
2. row1>= 0; column1>= 0; column2>=0

3.6.27. mcvMatrixMulMatrixRowMajor_s64

Description

Calculate the matrix multiply operation: M1 x M2. The element is int64.

Mout[j + i* column2] = Σ(M1[k + i * column1] * M2[j + k * column2])

Prototype

```
MInt32 mcvMatrixMulMatrixRowMajor_s64(MInt64 *matrixOut,MInt16 *matrix1,MInt32
                      *matrix2,MInt32 row1,MInt32 column1,MInt32 column2);
```

Parameters

`matrixOut`	[out]	The result matrix.
`matrix1`	[in]	Input Matrix 1(M1).
`matrix2`	[in]	Input Matrix 1(M2)..
`row1`	[in]	The number of rows of M1.
`column1`	[in]	The number of columns of M1(also the number of rows of M2).
`column2`	[in]	The number of columns of M2.

Return value

MCV_NULL_POINTER.
MCV_OK

Notes

1. row1>= 0; column1>= 0; column2>=0

3.7. Scale API

3.7.1. mcvScaleDownBy2u8

Description

Down-scale the image to half width and height by averaging 2x2 pixels into one..

Prototype

```
MInt32 mcvScaleDownBy2u8(  MUInt8*  src,MLong  srcWidth,MLong  srcHeight,
                           MUInt8* dst );
```

Parameters

`src`	[in]	The input buffer.
`srcWidth`	[in]	The width(columns) of input frame.
`srcHeight`	[in]	The height(rows) of input frame.
`dst`	[out]	The output buffer.

Return value

MCV_OK
MCV_NULL_POINTER.
MCV_INVALD_PARAM

Notes

1. src is srcWidth * srcHeight big. dst is srcWidth/2 * srcHeight/2 big.

2. srcWidth >= 4, srcHeight >= 4.

3.8. Motion API

3.8.1. mcvGetMotionCue

Description

Calculate the absolute difference of two images.

Prototype

```
MInt32 mcvGetMotionCue(MUInt8 *pCurGreyImage,MUInt8 *pPreGreyImage,
                    MLong lImgWidth, MLong lImgHeight, MLong lLineStep,
                    MUInt8 *pFrameDiffImage);
```

Parameters

`pCurGreyImage`	[in]	The buffer of current frame.
`pPreGreyImage`	[in]	The buffer of previous frame.
`lImgWidth`	[in]	The width(columns) of input frames.
`lImgHeight`	[in]	The height(rows) of input frames.
`lLineStep`	[in]	The line step of the three image buffers.
`pFrameDiffImage`	[out]	The absolute difference of two image buffers.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The size of *pCurGreyImage*, *pPreGreyImage*, *pFrameDiffImage* is lLineStep * lImgHeight *sizeof(MUInt8) .
2. All the buffers should not be NULL.
3. lImgWidth, lImgHeight should not be less than 0.

3.8.2. mcvDetectMotion

Description

Motion detect in three images. Absolute difference will be calculated inside.

Prototype

```
MInt32 mcvDetectMotion(
       ASVLOFFSCREEN* prevprev_colors[3], ASVLOFFSCREEN* prev_colors[3],
       ASVLOFFSCREEN* curr_colors[3], ASVLOFFSCREEN* motionimage);
```

Parameters

prevprev_colors	[in/out]	The buffer pointers of previous previous frame.
prev_colors	[in]	The buffer pointers of previous frame.
curr_colors	[in]	The buffer pointers of cur frame.
motionimage	[out]	The buffer of output motion data(absolute difference).

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The data in prevprev_colors[0/1/2] will be changed, will be used as tmp buffer after being used as frame buffer .
2. The width and height of the three images should be the same, doesn't check this inside.

3.8.3. mcvDetectMotion3FrameDiffY

Description

Motion detect for three images. Absolute difference will be calculated inside.

Prototype

```
MInt32 mcvDetectMotion3FrameDiffY(MByte *pPrevPrevData,
       MLong lPrevPrevStep, MByte *pPrevData, MLong lPrevStep,
       MByte *pCurrData,MLong lCurrStep, MLong lWidth, MLong lHeight,
       MByte *pDstData, MLong lDstStep)
```

Parameters

pPrevPrevData	[in]	The source image pPrevPrevData
lPrevPrevStep	[in]	The line step of image pPrevPrevData
pPrevData	[in]	The source image pPrevData
lPrevStep	[in]	The line step of image pPrevData
pCurrData	[in]	The source image pCurrData
lCurrStep	[in]	The line step of image pCurrData
lWidth	[in]	The width of three images
lHeight	[in]	The height of three images
pDstData	[out]	The source image pDstData
lDstStep	[in]	The line step of destination image pDstData

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. pPrevPrevData、pPrevData、pCurrData should not be null, they stand for three source image grey data.

2. This function only calculate the absolute difference of three images' grey data.
3. pDstData stand for absolute difference data, its buffer size should be lwidth * lheight.

The width and height of the three images should be the same, three steps should be the width of image, doesn't check inside function.

3.8.4. mcvDetectMotion3FrameDiffYWithRect

Description

Motion detect in the rectangle of three images. Absolute difference will be calculated inside.

Prototype

```
MInt32 mcvDetectMotion3FrameDiffYWithRect(MByte *pPrevPrevData,
       MLong lPrevPrevStep, MByte *pPrevData, MLong lPrevStep,
       MByte *pCurrData,MLong lCurrStep, MLong lWidth, MLong lHeight,
       MByte* pDstData, MLong lDstStep, MRECT *roi)
```

Parameters

pPrevPrevData	[in]	The source image pPrevPrevData
lPrevPrevStep	[in]	The line step of image pPrevPrevData
pPrevData	[in]	The source image pPrevData
lPrevStep	[in]	The line step of image pPrevData
pCurrData	[in]	The source image pCurrData
lCurrStep	[in]	The line step of image pCurrData
lWidth	[in]	The width of three images
lHeight	[in]	The height of three images
pDstData	[out]	The source image pDstData
lDstStep	[in]	The line step of destination image pDstData
roi	[in]	The rectangle of image you want detect motion

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. pPrevPrevData、pPrevData、pCurrData should not be null, they stand for three source image grey data.
2. This function only calculate the absolute difference of the rectangle of three images' grey data.
3. pDstData stand for absolute difference data, its buffer size should be lwidth * lheight, the data of rectangle position is absolute difference of images other position is filled by 0.

The width and height of the three images should be the same, three steps should be the width of image, doesn't check inside function.

3.9. Sum API

3.9.1. mcvIntegral

Description

Calculate the Integral value of an image.

Prototype

```
MInt32 mcvIntegral(MUInt8* pSrc, MUInt32 lSrcWidth, MUInt32 lSrcHeight,
            MUInt32 lSrcPitch,MUInt32* sum,MUInt32 lSumPitch);
```

Parameters

`pSrc`	[in]	The buffer of input frame.
`lSrcWidth`	[in]	The width(columns) of input frame.
`lSrcHeight`	[in]	The height(rows) of input frame.
`lSrcPitch`	[in]	The line step of *pSrc* in [byte] unit.
`sum`	[out]	The output buffer.
`lSumPitch`	[in]	The line step of *sum* in [byte] unit.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The size of *sum* is lSumPitch*(lSrcHeight+1) *sizeof(MUInt8) and the valid area is (lSrcHeight+1)* (lSrcWidth +1)*sizeof(MUInt32).
2. The value of lSumPitch must no more less than (lSrcWidth +1)*sizeof(MUInt32).
3. The 1'st row and the 1'st column of *sum* will always be 0.
4. All the buffers should not be NULL.

3.9.2. mcvIntegralWithRect

Description

Calculate the Integral value of the rectangle of an image.

Prototype

```
MInt32 mcvIntegral(MUInt8* pSrc, MUInt32 lSrcWidth, MUInt32 lSrcHeight,
            MUInt32 lSrcPitch, MUInt32* sum, MUInt32 lSumPitch, MRECT *roi);
```

Parameters

`pSrc`	[in]	The buffer of input frame.
`lSrcWidth`	[in]	The width(columns) of input frame.
`lSrcHeight`	[in]	The height(rows) of input frame.

`lSrcPitch`	[in]	The line step of *pSrc* in [byte] unit.
`sum`	[out]	The output buffer.
`lSumPitch`	[in]	The line step of *sum* in [byte] unit.
`roi`	[in]	The rectangle of image you want to calculate integral

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The valid area is (lSrcHeight+1)* (lSrcWidth+1)*sizeof(MUInt32).
2. The value of lSumPitch must no more less than (lSrcWidth+1)*sizeof(MUInt32).
3. The point which is in left or top or bottom side of roi, its integral value will always be 0.
4. The point(x,y) which is in right side of roi, its integral is the same as the point(y,roi->right) .

Buffers pSrc and sum should not be NULL.

3.9.3. mcvImgIntegralu8

Description

Calculate the Integral value of an image and the Integral of square of the image.

Prototype

```
MInt32 mcvImgIntegralu8(MUInt8*  src,MLong  srcWidth,MLong  srcHeight,MLong
*pIntegralImg, MLong *pIntegralImg2);
```

Parameters

`src`	[in]	The buffer of input frame.
`srcWidth`	[in]	The width(columns) of input frame.
`srcHeight`	[in]	The height(rows) of input frame.
`pIntegralImg`	[out]	The integral buffer.
`pIntegralImg2`	[out]	The integral of square buffer.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The size of *src* is srcWidth * srcHeight *sizeof(MUInt8).
2. *pIntegralImg* and *pIntegralImg2* can not be NULL in the same time.
3. The size of pIntegralImg is (srcWidth + 1) * (srcHeight + 1) *sizeof(MLong).
4. The size of pIntegralImg2 is (srcWidth + 1) * (srcHeight + 1) *sizeof(MLong)

3.9.4. mcvCalcSurfIntegralImage_Detect_Surf

Description

Calculate the Integral value of rectangle of an image.

Prototype

```
MInt32 mcvCalcSurfIntegralImage_Detect_Surf(MByte *pGraySrc,MInt32
       lWidth,MInt32 lHeight,MVoid *surf_int_image,MRECT *pRect);
```

Parameters

pGraySrc	[in]	The buffer of input gray image.
lWidth	[in]	The width(columns) of input frame.
lHeight	[in]	The height(rows) of input frame.
surf_int_image	[out]	The integral buffer of 8 channels.
pRect	[out]	The rectangle of image you want to calculate integral.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The size of pGraySrc is lWidth * lHeight *sizeof(MByte).
2. pGraySrc and pRect can not be NULL in the same time.

3.10. Optical flow API

3.10.1. mcvICmCalc_Bx_By

Description

Calculate the B image in Patch I and J. If you want to know the specific definition, Please Check the optical method in detail.

Prototype

```
MInt32 mcvICmCalc_Bx_By(MUInt8*pSrcI,MUInt8 *pSrcJ,MInt32 nPitch,
       MInt32 nWidth,MInt32 nHeight,MInt32 *pIxIy,MInt32 nIPitch,
       MInt32 nDx,MInt32 nDy,MInt32 *pBx,MInt32 *pBy);
```

Parameters

pSrcI	[in]	The buffer patch I.
pSrcJ	[in]	The buffer patch J.
nPitch	[in]	The line step of patch I and J in [byte] unit.
nWidth	[in]	The width(columns of pixels) of pSrcI and pSrcJ.

nHeight	[in]	The height(rows of pixels) of pSrcI and pSrcJ.
pIxIy	[in]	The buffer of output motion data(absolute difference).
nIPitch	[in]	The line step of *pIxIy* in [MInt32] unit.
nDx	[in]	Distance data, ranges from 0 to 256, inclusively.
nDy	[in]	Distance data, ranges from 0 to 256, inclusively.
pBx	[out]	The output pointer of B value.
pBy	[out]	The output pointer of B value.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The size of *pSrcI* is *nHeight* nPitch*sizeof(MUint8)*.
2. The size of *pSrcJ* is *(nHeight+1)* nPitch*sizeof(MUint8)*.
3. The size of *pIxIy* is *nHeight *nIPitch*sizeof(MInt32)*, including dI/dx and dI/dy, that is Ix, Iy.
4. *nIPitch* is in MInt32 unit, that is 4 bytes unit.
5. *pBx* and *pBy* each point to a single value of MInt32 type.
6. *nWidth* and *nHeight* should be greater than 0 and *nWidth* be even.
7. *nPitch >= nWidth+1, nIPitch >= nWidth*2.*

3.10.2. mcvIcmCalc_Bx_By_Gxx_Gxy_Gyy

Description

Calculate the B image and G matrix in Patch I and J. If you want to know the specific definition, Please Check the optical method in detail.

Prototype

```
MInt32 mcvIcmCalc_Bx_By_Gxx_Gxy_Gyy(MUInt8 *pSrcI,MUInt8 *pSrcJ,
       MInt32 nPitch,MInt32 nWidth,MInt32 nHeight, MInt32 *pIxIy,
       MInt32 nIPitch,MInt32 nDx,MInt32 nDy,MInt32 *pBx,MInt32 *pBy,
       MInt32 *pGxx,MInt32 *pGxy,MInt32 *pGyy);
```

Parameters

pSrcI	[in]	The buffer patch I.
pSrcJ	[in]	The buffer patch J.
nPitch	[in]	The line step of patch I and J in [byte] unit.
nWidth	[in]	The width(columns of pixels) of pSrcI and pSrcJ.
nHeight	[in]	The height(rows of pixels) of pSrcI and pSrcJ.
pIxIy	[in]	The buffer of output motion data(absolute difference).
nIPitch	[in]	The line step of *pIxIy* in [MInt32] unit.
nDx	[in]	Distance data, ranges from 0 to 256, inclusively.

nDy	[in]	Distance data, ranges from 0 to 256, inclusively.
pBx	[out]	The output pointer of B value.
pBy	[out]	The output pointer of B value.
pGxx	[out]	The output pointer of G value.
pGxy	[out]	The output pointer of G value.
pGyy	[out]	The output pointer of G value.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. The size of *pSrcI* is *nHeight* nPitch*sizeof(MUint8)*.
2. The size of *pSrcJ* is (*nHeight+1)* nPitch*sizeof(MUint8)*.
3. The size of *pIxIy* is *nHeight *nIPitch**sizeof(Mint32), including dI/dx and dI/dy, that is Ix, Iy.
4. *nIPitch* is in MInt32 unit, that is 4 bytes unit.
5. Each of *pBx*, *pBy*, *pGxx*, *pGxy*, *pGyy* points to a single value of MInt32 type.
6. *nWidth* and *nHeight* should be greater than 0 and *nWidth* be even.
7. *nPitch >= nWidth+1*, *nIPitch >= nWidth*2*.

3.11. Format conversion API

3.11.1. mcvExtract_Y_From_YUYV

Description

Extract Y component from an image of ASVL_PAF_YUYV format.

Prototype

```
MInt32 mcvExtract_Y_From_YUYV(MUInt8 *pSrc,MUInt8 *pDst,MUInt32 width,
       MUInt32 height,MUInt32 lSrcStride,MUInt32 lDstStride);
```

Parameters

pSrc	[in]	The input YUYV image buffer.
pDst	[out]	The output Y component buffer.
width	[in]	The width(columns) of input image.
height	[in]	The height(rows) of input image.
lSrcStride	[in]	The line step of *pSrc* in [byte] unit.
lDstStride	[in]	The line step of *pDst* in [byte] Unit

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. Buffer *pSrc* is lSrcStride* height*sizeof(MUInt8) big, and is *width*height* valid.
2. Buffer *pDst* is lDstStride * height*sizeof(MUInt8) big, and is (*width*/2)**height* valid.
3. All buffer pointers should not be NULL.
4. *width* should not be less than 8.
5. *width* should be even.

3.11.2. mcvYUYVToOrgData

Description

Split Y and CbCr from an image of ASVL_PAF_YUYV format.

Prototype

```
MInt32 mcvYUYVToOrgData(MUInt8 *pYUYVData, MLong lImgWidth, MLong lImgHeight,
                        MUInt8 *pGreyData, MUInt8 *pCbCrData);
```

Parameters

`pYUYVData`	[in]	The input YUYV image buffer.
`lImgWidth`	[in]	The width(columns) of input image.
`lImgHeight`	[in]	The height(rows) of input image.
`pGreyData`	[out]	The output Y component buffer.
`pCbCrData`	[out]	The output CbCr component buffer.

Return value

MCV_OK
MCV_NULL_POINTER

Notes

1. Buffer pYUYVData is (lImgWidth *2)* lImgHeight *sizeof(MUInt8) big.
2. Buffer pGreyData is lImgWidth * lImgHeight *sizeof(MUInt8) big.
2. Buffer pCbCrData is lImgWidth * lImgHeight *sizeof(MUInt8) big.
3. All buffer pointers should not be NULL.
4. lImgWidth and lImgHeight should not be less than 2;
5. lImgWidth should be even.

3.11.3. mcvColorRGB888toYUV420u8

Description

Convert Image of RGB888 to YUV420.

Prototype

```
MInt32 mcvColorRGB888toYUV420u8(MByte *rgbSrc,MByte *yuvDst,MLong width,
                                MLong height);
```

Parameters

`rgbSrc`	[in]	Input RGB888 format image.
`yuvDst`	[out]	Output YUV420 format image.
`width`	[in]	Image width. **NOTE** : must be a multiple of 2.
`height`	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.

3.11.4. mcvColorRGB888toBGR565u8

Description

Convert Image of RGB888 to BGR565.

Prototype

```
MInt32 mcvColorRGB888toBGR565(MByte *pSrc,MByte *pDst,MLong lWidth,
                              MLong lHeight);
```

Parameters

`pSrc`	[in]	Input RGB888 format image.
`pDst`	[out]	Output BGR565 format image.
`lWidth`	[in]	Image width. **NOTE** : must be a multiple of 2.
`lHeight`	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.

3.11.5. mcvColorRGB888toNV21u8

Description

Convert Image of RGB888 to YUV420.

Prototype

```
MInt32 mcvColorRGB888toNV21u8(MByte *rgbSrc,MByte *yuvDst,MLong width,
                              MLong height);
```

Parameters

`rgbSrc`	[in]	Input RGB888 format image.
`yuvDst`	[out]	Output NV21 format image.
`width`	[in]	Image width. **NOTE** : must be a multiple of 2.
`height`	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK

MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.

3.11.6. mcvColorRGB888toYUYVu8

Description

Convert Image of RGB888 to YUV420.

Prototype

```
MInt32 mcvColorRGB888toYUYVu8(MByte *rgbSrc,MByte *yuvDst,MLong width,
                              MLong height);
```

Parameters

`rgbSrc`	[in]	Input RGB888 format image.
`yuvDst`	[out]	Output YUYV format image.
`width`	[in]	Image width. **NOTE** : must be a multiple of 2.
`height`	[in]	Image Height.

Return value

MCV_OK

MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. The width should be a multiple of 2.

3.11.7. mcvColorRGB888toBGR888u8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_RGB24_B8G8R8.

Prototype

```
MInt32 mcvColorRGB888toBGR888u8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.

3.11.8. mcvColorRGB888toRGB565u8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_RGB16_R5G6B5.

Prototype

```
MInt32 mcvColorRGB888toRGB565u8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.

3.11.9. mcvColorRGB888toYVYUu8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_YVYU.

Prototype

```
MInt32 mcvColorRGB888toYVYUu8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
   Width is a multiple of 2.

3.11.10. mcvColorRGB888toUYVYu8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_UYVY.

Prototype

```
MInt32 mcvColorRGB888toUYVYu8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER

MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
   Width is a multiple of 2.

3.11.11. mcvColorRGB888toVYUYu8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_VYUY.

Prototype

```
MInt32 mcvColorRGB888toVYUYu8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
   Width is a multiple of 2.

3.11.12. mcvColorRGB888toYV24u8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_YV24.

Prototype

```
MInt32 mcvColorRGB888toYV24u8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.

3.11.13. mcvColorRGB888toI422Hu8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_I422H.

Prototype

```
MInt32 mcvColorRGB888toI422Hu8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Width must be a multiple of 2.

3.11.14. mcvColorRGB888toNV12u8

Description

Convert Image of ASVL_PAF_RGB24_R8G8B8 to ASVL_PAF_NV12.

Prototype

```
MInt32 mcvColorRGB888toNV12u8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Width and height must be a multiple of 2.

3.11.15. mcvColorBGR888toRGB565u8

Description

Convert Image of BGR888 to RGB565.

Prototype

```
MInt32 mcvColorBGR888toRGB565u8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.
`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Width and height must be a multiple of 2.

3.11.16. mcvColorBGR888toARGB8888u8

Description

Convert Image of BGR888 to ARGB8888

Prototype

```
MInt32 mcvColorBGR888toARGB8888u8(MByte* pSrc, MByte* pDst, MLong width,MLong
                        height);
```

Parameters

`pSrc`	[in]	Input image.
`pDst`	[out]	Output image.

`width`	[in]	Image width.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Width and height must be a multiple of 2.

3.11.17. mcvColorYUYVtoRGB888u8

Description

Convert Image of YUYV to RGB888.

Prototype

```
MInt32 mcvColorYUYVtoRGB888u8(MByte *yuvSrc,MByte *rgbDst,MLong width,
                              MLong height);
```

Parameters

`yuvSrc`	[in]	Input YUYV format image.
`rgbDst`	[out]	Output RGB888 format image.
`width`	[in]	Image width. **NOTE** : must be a multiple of 2.
`height`	[in]	Image Height.

Return value

MCV_OK
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. The width should be a multiple of 2.

3.11.18. mcvColorYUYVtoYUV420u8

Description

Convert Image of YUYV to YUV420.

Prototype

```
MInt32 mcvColorYUYVtoYUV420u8(MByte *yuvSrc,MByte *yuvDst,MLong width,
                              MLong height);
```

Parameters

`yuvSrc`	[in]	Input YUYV format image.
`yuvDst`	[out]	Output YUV420 format image.
`width`	[in]	Image width. **NOTE** : must be a multiple of 2.
`height`	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.

3.11.19. mcvColorYUYVtoNV21u8

Description

Convert Image of YUYV to NV21.

Prototype

```
MInt32 mcvColorYUYVtoNV21u8(MByte *pSrc,MByte *pDst,MLong lWidth,
                            MLong lHeight);
```

Parameters

`pSrc`	[in]	Input YUYV format image.
`pDst`	[out]	Output NV21 format image.
`lWidth`	[in]	Image width. **NOTE** : must be a multiple of 2.
`lHeight`	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.

3.11.20. mcvColorYUYVtoNV12u8

Description

Convert Image of YUYV to NV12.

Prototype

```
MInt32 mcvColorYUYVtoNV12u8(MByte *pSrc,MByte *pDst,MLong lWidth,
                            MLong lHeight);
```

Parameters

`pSrc`	[in]	Input YUYV format image.
`pDst`	[out]	Output NV12 format image.
`lWidth`	[in]	Image width. **NOTE** : must be a multiple of 2.
`lHeight`	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.

3.11.21. mcvColorNV21toRGB888u8

Description

Convert Image of ASVL_PAF_NV21 to ASVL_PAF_RGB24_R8G8B8.

Prototype

```
MInt32 mcvColorNV21toRGB888u8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input NV21 format image.
`dstImg`	[out]	Output RGB888 format image.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Image width and height must be a multiple of 2 and >= 2.

3.11.22. mcvColorNV21toBGR888u8

Description

Convert Image of ASVL_PAF_NV21 to ASVL_PAF_RGB24_B8G8R8.

Prototype

```
MInt32 mcvColorNV21toBGR888u8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input NV21 format image.
`dstImg`	[out]	Output BGR888 format image.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Image width and height must be a multiple of 2 and >= 2.

3.11.23. mcvColorNV21toRGBA8888u8

Description

Convert Image of ASVL_PAF_NV21 to ASVL_PAF_RGB32_A8R8G8B8.

Prototype

```
MInt32 mcvColorNV21toRGBA8888u8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg,
            MUInt8 alpha);
```

Parameters

`srcImg`	[in]	Input NV21 format image.
`dstImg`	[out]	Output RGBA8888 format image.
`alpha`	[in]	Alpha value.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. dstImg->ppu8Plane[0] should be 4 bytes allign.
3. Image width and height must be a multiple of 2 and >= 2

3.11.24. mcvColorNV12toRGBA8888u8

Description

Convert Image of ASVL_PAF_NV12 to ASVL_PAF_RGB32_A8R8G8B8.

Prototype

```
MInt32 mcvColorNV12toRGBA8888u8 (LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg,
            MUInt8 alpha);
```

Parameters

`srcImg`	[in]	Input NV12 format image.
`dstImg`	[out]	Output RGBA8888 format image.
`alpha`	[in]	Alpha value.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. dstImg->ppu8Plane[0] should be 4 bytes allign.
3. Image width and height must be a multiple of 2 and >= 2.

3.11.25. mcvColorI420toRGBA8888u8

Description

Convert Image of ASVL_PAF_I420 to ASVL_PAF_RGB32_A8R8G8B8.

Prototype

```
MInt32 mcvColorI420toRGBA8888u8 (LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg,
            MUInt8 alpha);
```

Parameters

`srcImg`	[in]	Input NV12 format image.
`dstImg`	[out]	Output RGBA8888 format image.
`alpha`	[in]	Alpha value.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. dstImg->ppu8Plane[0] should be 4 bytes allign.
3. Image width and height must be a multiple of 2 and >= 2.

3.11.26. mcvColorRGBA8888toNV21u8

Description

Convert Image of ASVL_PAF_RGB32_A8R8G8B8 to ASVL_PAF_NV21.

Prototype

```
MInt32 mcvColorRGBA8888toNV21u8 (LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input RGBA8888 format image.
`dstImg`	[out]	Output ASVL_PAF_NV21format image.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. srcImg->ppu8Plane[0] should be 4 bytes allign.
3. Image width and height must be a multiple of 2 and >= 2.

3.11.27. mcvColorRGBA8888toNV12u8

Description

Convert Image of ASVL_PAF_RGB32_A8R8G8B8 to ASVL_PAF_NV12.

Prototype

```
MInt32 mcvColorRGBA8888toNV12u8 (LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input RGBA8888 format image.
`dstImg`	[out]	Output ASVL_PAF_NV12 format image.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. srcImg->ppu8Plane[0] should be 4 bytes allign.
3. Image width and height must be a multiple of 2 and >= 2.

3.11.28. mcvColorRGBA8888toI420u8

Description

Convert Image of ASVL_PAF_RGB32_A8R8G8B8 to ASVL_PAF_NV21.

Prototype

```
MInt32 mcvColorRGBA8888toNV21u8 (LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input RGBA8888 format image.
`dstImg`	[out]	Output ASVL_PAF_I420 format image.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. srcImg->ppu8Plane[0] should be 4 bytes allign.
3. Image width and height must be a multiple of 2 and >= 2.

3.11.29. mcvColorNV21toI420u8

Description

Convert Image of ASVL_PAF_NV21 to ASVL_PAF_I420.

Prototype

```
MInt32 mcvColorNV21toI420u8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input NV21 format image.
`dstImg`	[out]	Output I420 format image.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. Image width and height must be a multiple of 2, and >= 2

3.11.30. mcvColorNV12toI420u8

Description

Convert Image of ASVL_PAF_NV12 to ASVL_PAF_I420.

Prototype

```
MInt32 mcvColorNV12toI420u8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input NV12 format image.
`dstImg`	[out]	Output I420 format image.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. Image width and height must be a multiple of 2, and >= 2

3.11.31. mcvColorI420toYUYVu8

Description

Convert Image of ASVL_PAF_I420 to ASVL_PAF_YUYV.

Prototype

```
MInt32 mcvColorI420toYUYVu8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input I420 format image.
`dstImg`	[out]	Output YUYV format image.

Return value

MCV_OK
MCV_INVALID_PARAM
MCV_NULL_POINTER

Notes

1. All buffer pointers should not be NULL.
2. Image width and height must be a multiple of 2, and >= 2
3. Image buffer pointer should 4Bytes align.

3.11.32. mcvColorI420toRGB888u8

Description

Convert Image of ASVL_PAF_I420 to ASVL_PAF_RGB24_R8G8B8.

Prototype

```
MInt32 mcvColorI420toRGB888u8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input I420 format image.
`dstImg`	[out]	Output RGB888 format image.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Iamge width and height must be a multiple of 2 and >= 2.

3.11.33. mcvColorI420toNV21u8

Description

Convert Image of ASVL_PAF_I420 to ASVL_PAF_NV21.

Prototype

```
MInt32 mcvColorI420toNV21u8(LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg);
```

Parameters

`srcImg`	[in]	Input I420 format image.
`dstImg`	[out]	Output ASVL_PAF_NV21 format image.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Iamge width and height must be a multiple of 2 and >= 2.

3.11.34. mcvColorYV12toRGB888u8

Description

Convert Image of ASVL_PAF_YV12 to ASVL_PAF_RGB24_R8G8B8.

Prototype

```
MInt32 mcvColorYV12toRGB888u8(MByte* pSrc, MByte *pDst,
                              MLong width, MLong height);
```

Parameters

pSrc	[in]	Input YV12 format image.
pDst	[out]	Output RGB888 format image.
width	[in]	Image width. **NOTE** : must be a multiple of 2.
height	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

All buffer pointers should not be NULL.

3.11.35. mcvColorYV12toNV21u8

Description

Convert Image of ASVL_PAF_YV12 to ASVL_PAF_NV21.

Prototype

```
MInt32 mcvColorYV12toNV21u8(MByte* pSrc, MByte *pDst,
                            MLong width, MLong height);
```

Parameters

pSrc	[in]	Input YV12 format image.
pDst	[out]	Output NV21 format image.
width	[in]	Image width. **NOTE** : must be a multiple of 2.
height	[in]	Image Height. **NOTE** : must be a multiple of 2.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

All buffer pointers should not be null.

3.11.36. mcvColorBGR888toHSL888u8

Description

Convert Image of BGR888 to HSL888.

Prototype

```
MInt32 mcvColorBGR888toHSL888u8(MUInt8 *bgr, MUInt8 *hsl, MUInt32 lHeight,
       MUInt32 lWidth, MUInt32 lineBytesBgr, MUInt32 lineBytesHsl);
```

Parameters

`bgr`	[in]	Input BGR888 format image
`hsl`	[out]	Output HSL888 format image.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height..
`lineBytesBgr`	[in]	Input BGR888 line step.
`lineBytesHsl`	[in]	Output HSL888 line step.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Image width and height should be larger than 2.

3.11.37. mcvColorI420toHSL888u8

Description

Convert Image of I420 to HSL888.

Prototype

```
MInt32 mcvColorI420toHSL888u8(MUInt8 *pY,MUInt8 *pCb, MUInt8 *pCr,
        MUInt32 lWidth, MUInt32 lHeight, MUInt8 *pHSL,
        MUInt32 y_step1,MUInt32 cbcr_step1,MUInt32 hsl_step1);
```

Parameters

`pY`	[in]	Input I420 format image Y frame.
`pCb`	[in]	Input YUYV format image Cb frame.
`pCr`	[in]	Input YUYV format image Cr frame.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height..
`pHSL`	[out]	Output HSL888 format image.
`y_step1`	[in]	Input Y frame line step.
`cbcr_step1`	[in]	Input Cb and Cr frame line step
`hsl_step1`	[in]	Output HSL888 line step

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Image width and height should be larger than 2.

3.11.38. mcvColorBGR565toHSL888u8

Description

Convert Image of BGR565to HSL888.

Prototype

```
MInt32  mcvColorBGR565toHSL888u8(MUInt8 *pBgr565, MUInt8 *pHsl,
        MUInt32 lHeight,MUInt32 lWidth,
        MUInt32 lineBytesBgr, MUInt32 lineBytesHsl);
```

Parameters

`pBgr565`	[in]	Input BGR565 format image
`pHsl`	[out]	Output HSL888 format image.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height..
`lineBytesBgr`	[in]	Input BGR565 line step.
`lineBytesHsl`	[in]	Output HSL888 line step.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Image width and height should be larger than 2.

3.11.39. mcvColorRGB565toHSL888u8

Description

Convert Image of RGB565HSL888.

Prototype

```
MInt32  mcvColorRGB565toHSL888u8(MUInt8 *pBgr565, MUInt8 *pHsl,
        MUInt32 lHeight,MUInt32 lWidth,
        MUInt32 lineBytesBgr, MUInt32 lineBytesHsl);
```

Parameters

`pBgr565`	[in]	Input RGB565format image
`pHsl`	[out]	Output HSL888 format image.
`lWidth`	[in]	Image width.
`lHeight`	[in]	Image Height..
`lineBytesBgr`	[in]	Input RGB565line step.
`lineBytesHsl`	[in]	Output HSL888 line step.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_PARAM

Notes

1. All buffer pointers should not be NULL.
2. Image width and height should be larger than 2.

3.12. Format Convert Multi-threads API

3.12.1. mcvColorCvtInit_MultiThreads

Description

Initialize the Format Convert Engine.

Prototype

```
MHandle mcvColorCvtInit_MultiThreads();
```

Parameters

Return value

A Handle of the engine : valid if non-zero.

Notes

3.12.2. mcvColorCvtProcess_MultiThreads

Description

Do Format Convert for a big image.

Prototype

```
MInt32 mcvColorCvtProcess_MultiThreads(MHandle
mcvColorCvtHandle,LPASVLOFFSCREEN srcImg,LPASVLOFFSCREEN dstImg, MVoid
*extParam);
```

Parameters

mcvColorCvtHandle	[in]	Engine handle returned by mcvColorCvtInit_MultiThreads ().
srcImg	[in]	Source image to be converted.
dstImg	[out]	converted output image.
extParam	[in]	Extended parameter buffer

Return value

MCV_NULL_POINTER

MCV_INVALID_PARAM

MCV_OK.

Notes

3. Generally, extParam is NULL. It offers extra information beyond the information contained in srcImg and dstImg.
4. Supported color format list:

Src image format	to	Dst image format	Notes
ASVL_PAF_NV21	→	ASVL_PAF_RGB32_R8G8B8	extParam : the pointer of typedef struct { MInt32 alpha; }mcvColorExtParam_ALPHA_t;
ASVL_PAF_NV21	→	ASVL_PAF_RGB24_R8G8B8	extParam : Ignored
ASVL_PAF_NV21	→	ASVL_PAF_RGB24_B8G8R8	extParam : Ignored
ASVL_PAF_NV21	→	ASVL_PAF_I420	extParam : Ignored
ASVL_PAF_I420	→	ASVL_PAF_RGB24_R8G8B8	extParam : Ignored
ASVL_PAF_I420	→	ASVL_PAF_NV21	extParam : Ignored
ASVL_PAF_I420	→	ASVL_PAF_YUYV	extParam : Ignored
ASVL_PAF_NV12	→	ASVL_PAF_I420	extParam : Ignored

5. mcvColorCvtHandle should not be NULL.

3.12.3. mcvColorCvtUnInit_MultiThreads

Description

UnInitialize the Format Convert Engine

Prototype

```
MInt32 mcvColorCvtUnInit_MultiThreads(MHandle mcvColorCvtHandle);
```

Parameters

`mcvColorCvtHandle`	[in]	Engine handle returned by mcvColorCvtInit_MultiThreads ()

Return value

MCV_NULL_POINTER
MCV_UNEXPECTED_ERR
MCV_OK.

Notes

3.12.4. Format Convert Engine Instance

Notes:

1. Buffer pointer must be 4 bytes align.
2. Each row of the buffer should be 4 bytes align.(that is to say, row stride should be a multiple of 4)

Usage:

Step1:

```
Mhandle cvtHandle = mcvColorCvtInit_MultiThreads ();
if(cvtHandle == MNULL)
{
        Return -1;
}
```

Step 2:

```
typedef struct
 {
    MInt32 alpha;
 }mcvColorExtParam_ALPHA_t;
/***************NV21 TO RGBA8888***************/
ASVLOFFSCREEN srcImg, dstImg;
mcvColorExtParam_ALPHA_t  extParam;

extParam.alpha = 80;

srcImg.u32PixelArrayFormat = ASVL_PAF_NV21;
srcImg.xxx = xxx;….
dstImg.u32PixelArrayFormat = ASVL_PAF_RGB32_R8G8B8;
dstImg.xxx = xxx;…
mcvColorCvtProcess_MultiThreads (cvtHandle,&srcImg,&dstImg,& extParam);

/***************xxx TO xxx***************/
ASVLOFFSCREEN srcImg, dstImg;
mcvColorExtParam_ALPHA_t  extParam;
```

```
extParam.alpha = alpha;

srcImg.u32PixelArrayFormat = xxx;
srcImg.xxx = xxx;….
dstImg.u32PixelArrayFormat = xxx;
dstImg.xxx = xxx;…
mcvColorCvtProcess_MultiThreads (cvtHandle,&srcImg,&dstImg,NULL);

……
```

Step 3:

```
mcvColorCvtUnInit_MultiThreads (cvtHandle);
```

3.13. Gradient API

3.13.1. mcviCalcGradientMagAngle_I422H_FixPoint

Description

Calculate the Gradient of an image of ASVL_PAF_I422H format, then calculate the magnitude and angle of the Gradient. The template is [-1 0 1] or [1 0 -1] in horizontal direction, and $[-1\ 0\ 1]^T$ in vertical direction.

Prototype

```
MInt32 mcviCalcGradientMagAngle_I422H_FixPoint(MUInt8* pChns[3], MInt32 step[3],
       MInt32* magptr, MInt32 magstep, MUInt8* angleptr, MInt32 anglestep,
       MInt32 width, MInt32 height, MInt32 nBins, MInt32 lDirectionX);
```

Parameters

`pChns`	[in]	The buffer pointers of Y and U and V.
`step`	[in]	The buffer line step of Y and U and V in [byte] unit.
`magptr`	[out]	The buffer of magnitude output.
`magstep`	[in]	The line step of *magptr* in [byte] unit.
`angleptr`	[out]	The buffer of angle output.
`anglestep`	[in]	The line step of *angleptr* in [byte] unit.
`width`	[in]	The width(columns of pixels) of input image.
`height`	[in]	The height(rows of pixels) of input image.
`nBins`	[in]	the number of zones to classify the angle data.
`lDirectionX`	[in]	Indicates the horizontal template when calculate the gradient.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. 256 > nBins > 0: typical value :6
2. lDirectionX can only be 1 or -1, the template is [-1 0 1]when lDirectionX = 1 and [1 0 -1] when lDirectionX = -1.
3. magptr: Restore magnitude of the gradient of the image for output, this buffer is magstep * height big, and is (width*2)*height*sizeof(MInt32) valid, namely, each gradient contributes 2 magnitude data;

   angleptr: Restore angle of the gradient of the image for output, this buffer is anglestep * height big, and is (width*2)*height*sizeof(MUInt8) valid,namely, each gradient contributes 2 angle data.
4. All buffer pointers should not be NULL.
5. width and height should not be less than 2;
6. width should be even.

3.13.2. mcviCalcGradientMagAngle_I422H_left

Description

Calculate the Gradient of an image of ASVL_PAF_I422H format, then calculate the magnitude and angle of the Gradient. The template is [-1 0 1] in horizontal direction, and $[-1\ 0\ 1]^T$ in vertical direction.

This function is the same as function *mcviCalcGradientMagAngle_I422H_FixPoint* if you set lDirectionX = 1 in *mcviCalcGradientMagAngle_I422H_FixPoint*.

Prototype

```
MInt32 mcviCalcGradientMagAngle_I422H _left(MUInt8* pChns[3], MInt32 step[3],
       MInt32* magptr, MInt32 magstep, MUInt8* angleptr, MInt32 anglestep,
       MInt32 width, MInt32 height, MInt32 nBins);
```

Parameters

`pChns`	[in]	The buffer pointers of Y and U and V.
`step`	[in]	The buffer line step of Y and U and V in [byte] unit.
`magptr`	[out]	The buffer of magnitude output.
`magstep`	[in]	The line step of *magptr* in [byte] unit.
`angleptr`	[out]	The buffer of angle output.
`anglestep`	[in]	The line step of *angleptr* in [byte] unit.
`width`	[in]	The width(columns of pixels) of input image.
`height`	[in]	The height(rows of pixels) of input image.
`nBins`	[in]	the number of zones to classify the angle data.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. 256 > nBins > 0: typical value :6.
2. magptr: Restore magnitude of the gradient of the image for output, this buffer is magstep * height big, and is (width*2)*height*sizeof(MInt32) valid, namely, each gradient contributes 2 magnitude data;

   angleptr: Restore angle of the gradient of the image for output, this buffer is anglestep * height big, and is (width*2)*height*sizeof(MUInt8) valid,namely, each gradient contributes 2 angle data.
3. All buffer pointers should not be NULL.
4. width and height should not be less than 2;
5. width should be even.

3.13.3. mcviCalcGradientMagAngle_I422H_right

Description

Calculate the Gradient of an image of ASVL_PAF_I422H format, then calculate the magnitude and angle of the Gradient. The template is [1 0 -1] in horizontal direction, and $[-1\ 0\ 1]^T$ in vertical direction.

This function is the same as function *mcviCalcGradientMagAngle_I422H_FixPoint* if you set lDirectionX = -1 in *mcviCalcGradientMagAngle_I422H_FixPoint*.

Prototype

```
MInt32 mcviCalcGradientMagAngle_I422H _right(MUInt8* pChns[3], MInt32 step[3],
       MInt32* magptr, MInt32 magstep, MUInt8* angleptr, MInt32 anglestep,
       MInt32 width, MInt32 height, MInt32 nBins);
```

Parameters

`pChns`	[in]	The buffer pointers of Y and U and V.
`step`	[in]	The buffer line step of Y and U and V in [byte] unit.
`magptr`	[out]	The buffer of magnitude output.
`magstep`	[in]	The line step of *magptr* in [byte] unit.
`angleptr`	[out]	The buffer of angle output.
`anglestep`	[in]	The line step of *angleptr* in [byte] unit.
`width`	[in]	The width(columns of pixels) of input image.
`height`	[in]	The height(rows of pixels) of input image.
`nBins`	[in]	the number of zones to classify the angle data.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. 256 > nBins > 0: typical value :6.

2. magptr: Restore magnitude of the gradient of the image for output, this buffer is magstep * height big, and is (width*2)*height*sizeof(MInt32) valid, namely, each gradient contributes 2 magnitude data;

   angleptr: Restore angle of the gradient of the image for output, this buffer is anglestep * height big, and is (width*2)*height*sizeof(MUInt8) valid,namely, each gradient contributes 2 angle data.
3. All buffer pointers should not be NULL.
4. width and height should not be less than 2;
5. width should be even.

3.13.4. mcvCalcGradientMagAngleFix_Gray

Description

Calculate the Gradient of a grey image(Y component only), then calculate the magnitude and angle of the Gradient. The template is [-1 0 1] or [1 0 -1] in horizontal direction, and $[-1\ 0\ 1]^T$ in vertical direction.

Prototype

```
MInt32 mcvCalcGradientMagAngleFix_Gray(MUInt8* pSrc, MInt32 lSrcStep,
       MInt32* magptr, MInt32 magstep, MUInt8* angleptr, MInt32 anglestep,
       MInt32 width, MInt32 height, MInt32 nBins, MInt32 lDirectionX);
```

Parameters

`pSrc`	[in]	The buffer of Y.
`lSrcStep`	[in]	The buffer line step of Y.
`magptr`	[out]	The buffer of magnitude output.
`magstep`	[in]	The line step of *magptr* in [byte] unit.
`angleptr`	[out]	The buffer of angle output.
`anglestep`	[in]	The line step of *angleptr* in [byte] unit.
`width`	[in]	The width(columns of pixels) of input image.
`height`	[in]	The height(rows of pixels) of input image.
`nBins`	[in]	the number of zones to classify the angle data.
`lDirectionX`	[in]	Indicates the horizontal template when calculate the gradient.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. 256 > nBins > 0: typical value :6
2. lDirectionX can only be 1 or -1, the template is [-1 0 1]when lDirectionX = 1 and [1 0 -1] when lDirectionX = -1.
3. magptr: Restore magnitude of the gradient of the image for output, this buffer is magstep * height big, and is (width*2)*height*sizeof(MInt32) valid, namely, each gradient contributes 2 magnitude data;

angleptr: Restore angle of the gradient of the image for output, this buffer is anglestep * height big, and is (width*2)*height*sizeof(MUInt8) valid,namely, each gradient contributes 2 angle data.

4. All buffer pointers should not be NULL.
5. width and height should not be less than 2;
6. width should be even.

3.13.5. mcvCalcGradientMagAngleFix_Gray_left

Description

Calculate the Gradient of a grey image(Y component only), then calculate the magnitude and angle of the Gradient. The template is [-1 0 1] in horizontal direction, and $[-1\ 0\ 1]^T$ in vertical direction. This function is the same as function *mcviCalcGradientMagAngleFix_Gray* if you set lDirectionX = 1 in *mcviCalcGradientMagAngleFix_Gray*.

Prototype

```
MInt32 mcvCalcGradientMagAngleFix_Gray_left(MUInt8* pSrc, MInt32 lSrcStep,
       MInt32* magptr, MInt32 magstep, MUInt8* angleptr, MInt32 anglestep,
       MInt32 width, MInt32 height, MInt32 nBins);
```

Parameters

`pSrc`	[in]	The buffer of Y.
`lSrcStep`	[in]	The buffer line step of Y in [byte] unit.
`magptr`	[out]	The buffer of magnitude output.
`magstep`	[in]	The line step of *magptr* in [byte] unit.
`angleptr`	[out]	The buffer of angle output.
`anglestep`	[in]	The line step of *angleptr* in [byte] unit.
`width`	[in]	The width(columns of pixels) of input image.
`height`	[in]	The height(rows of pixels) of input image.
`nBins`	[in]	the number of zones to classify the angle data.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. 256 > nBins > 0: typical value :6.
2. magptr: Restore magnitude of the gradient of the image for output, this buffer is magstep * height big, and is (width*2)*height*sizeof(MInt32) valid, namely, each gradient contributes 2 magnitude data;

   angleptr: Restore angle of the gradient of the image for output, this buffer is anglestep * height big, and is (width*2)*height*sizeof(MUInt8) valid,namely, each gradient contributes 2 angle data.
3. All buffer pointers should not be NULL.

4. width and height should not be less than 2;
5. width should be even.

3.13.6. mcvCalcGradientMagAngleFix_Gray_right

Description

Calculate the Gradient of a grey image(Y component only), then calculate the magnitude and angle of the Gradient. The template is [1 0 -1] in horizontal direction, and $[-1\ 0\ 1]^T$ in vertical direction. This function is the same as function *mcviCalcGradientMagAngleFix_Gray* if you set lDirectionX = -1 in *mcviCalcGradientMagAngleFix_Gray*.

Prototype

```
MInt32 mcvCalcGradientMagAngleFix_Gray_right(MUInt8* pSrc, MInt32 lSrcStep,
       MInt32* magptr, MInt32 magstep, MUInt8* angleptr, MInt32 anglestep,
       MInt32 width, MInt32 height, MInt32 nBins);
```

Parameters

`pSrc`	[in]	The buffer of Y.
`lSrcStep`	[in]	The buffer line step of Y in [byte] unit.
`magptr`	[out]	The buffer of magnitude output.
`magstep`	[in]	The line step of *magptr* in [byte] unit.
`angleptr`	[out]	The buffer of angle output.
`anglestep`	[in]	The line step of *angleptr* in [byte] unit.
`width`	[in]	The width(columns of pixels) of input image.
`height`	[in]	The height(rows of pixels) of input image.
`nBins`	[in]	the number of zones to classify the angle data.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

1. 256 > nBins > 0: typical value :6.
2. magptr: Restore magnitude of the gradient of the image for output, this buffer is magstep * height big, and is (width*2)*height*sizeof(MInt32) valid, namely, each gradient contributes 2 magnitude data;

   angleptr: Restore angle of the gradient of the image for output, this buffer is anglestep * height big, and is (width*2)*height*sizeof(MUInt8) valid,namely, each gradient contributes 2 angle data.
3. All buffer pointers should not be NULL.
4. width and height should not be less than 2;
5. width should be even.

3.14. Parallel Engine API

3.14.1. mcvParallelInit

Description

Initialize parallel engine.

Prototype

```
MInt32 mcvParallelInit(MHandle hContext,MUInt32 iCoreNumHint);
```

Parameters

`hContext`	[in]	Memory handle returned by *MMemMgrCreate*.
`iCoreNumHint`	[in]	Number of cores to be used.

Return value

Actually used core number >= 0 ;
MCV_NULL_POINTER;
MCV_INVALID_CALL.

Notes

1. *iCoreNumHint* ranges from 0 to 0xffffffff inclusively , if *iCoreNumHint* == 0, use default cpus. This value is just a hint, actually used core number is limited by the platform.
2. Call it only once.
3. *hContext* is returned by *MMemMgrCreate*, which belongs to the *mpbase* lib
   if *hContext* == NULL, malloc will be used to allocate memory inside.
4. Less than 2K Bytes will be allocated inside;

3.14.2. mcvAddTask

Description

Add a task to the parallel engine.

Prototype

```
MInt32 mcvAddTask(MVoid *process, MVoid *arg);
```

Parameters

`process`	[in]	Function pointer.
`arg`	[in]	Function parameter pointer.

Return value

An unique taskId, which can be used in function "mcvWaitTask" ;
MCV_NULL_POINTER;
MCV_QUEUE_OVERFLOW.

Notes

1. *process*: should be *void (*f)(void *)*.

2. After your program finished, the times you call *mcvAddTask* should be the same as you call *mcvWaitTask*.
3. If you have called *mcvAddTask n* times , and *mcvWaitTask m* times so far, (*n* – *m*) should be less than 32(*n* should not be less than *m*, of course).
4. *arg* will be used in a child thread, so, make sure it is valid until mcvWaitTask(taskId) is called.

3.14.3. mcvWaitTask

Description

Wait a task to finish.

Prototype

```
MInt32 mcvWaitTask(MInt32 taskId);
```

Parameters

`taskId`	[in]	The Id of the task.

Return value

MCV_OK ;

MCV_NULL_POINTER.

Notes

1. *taskid* is the value returned by function *mcvAddTask*.
2. This function is a blocking function.

3.14.4. mcvParallelUninit

Description

Destroy the parallel engine.

Prototype

```
MInt32 mcvParallelUninit();
```

Parameters

Return value

MCV_OK ;

MCV_INVALID_CALL;

Notes

1. Call it only once.

3.14.5. Parallel Engine Instance

Parallel Engine Instance:

```
Typdef struct
{
        MUInt8 * pSrc;
        MInt32 lSrcWidth;
        MInt32 lSrcHeight;
        …
}yourParam_t;
```

Step 1: Call mcvParallelInit at the very beginning of your program(call it only once):

```
        if(mcvParallelInit(Memhandle, 1000) < 0)//you wanna use 1000 cores to do your tasks.
        {
                printf("Failed to start parallel engine!!\n");
        }
```

Step 2: Package your function like this :

```
        MVoid yourFunction1(MVoid *pHaha)
        {
                yourParam_t *pParam = (yourParam_t *) pHaha;
                if(pParam!= MNull)
                {
                        //this is what you will actually do
                        resize(pParam-> pSrc, pParam->width, pParam->height,...);
                }
        }
        ….
```

Step 3: Execute tasks:

```
        task 1:
                yourParam_t *pParam1 = (yourParam_t *)malloc(...);
                yourParam_t *pParam2 = (yourParam_t *)malloc(...);
                MInt32 taskId1,taskId2,...;

                pParam1 -> pSrc = pSrc;
                pParam1 -> lSrcWidth = lSrcWidth;
                pParam1 -> lSrcHeight = lSrcHeight;
                .....
                pParam2->…

                taskId1 = mcvAddTask(yourFunction1, pParam1);//keep pParam1 valid
        task 2:
                ....
                taskId2 = mcvAddTask(yourFunction2, pParam2); //keep pParam2 valid
                ...
        task 3:
```

....

Step 4: Wait tasks to finish:

mcvWaitTask(taskId1); // after calling this function, pParam1 is no longer used by the child thread.

mcvWaitTask(taskId2); // after calling this function, pParam2 is no longer used by the child thread

....

Step 5: goto Step3 to do more tasks. Goto Step6 till there is no more task to do.

Step 6: Shutdown parallel engine when your program is coming to an end(call it only once):

```
if(mcvParallelUninit() < 0)
{
        printf("Failed to shut down parallel engine!!\n");
}
```

- Please Pay attention to the remarks in red.

.

3.15. Algorithm

3.15.1. mcvCalcHistBackProject_I422H

Description

Calculate histogram projection of image. The format of input image should be I422H.

Prototype

```
MInt32 mcvCalcHistBackProject_I422H (LPASVLOFFSCREEN pOffScreen,
                                     MLong* table)
```

Parameters

`pOffScreen`	[in]	The structure describe source and destination image
`table`	[in]	The table stores histogram projection information

Return value

MCV_OK

MCV_NULL_POINTER

Notes

1. The size of table should no less than 8K(8192) in [MLong] Unit, index start from 0.
2. image format (pOffScreen->u32PixelArrayFormat) should be ASVL_PAF_I422H.
3. pOffScreen stand for input I422H image.
4. pOffScreen-> i32Width should pOffScreen-> i32Height should be double size by 2.
5. pOffScreen->pi32Pitch[0](y step) should be double size pOffScreen-> pi32Pitch[1] (u step)and pOffScreen-> pi32Pitch[2](v step).

6. For usage details please refer to the description of mcvCalcHistBackProject_I422HWithRect in mobilecv.h

3.15.2. mcvCalcHistBackProject_I422HWithRect

Description

Calculate histogram projection of the rectangle of one frame in I422H format.

Prototype

```
MInt32 mcvCalcHistBackProject_I422HWithRect(LPASVLOFFSCREEN pOffScreen,
                                            MLong* table, MRECT *roi)
```

Parameters

`pOffScreen`	[in]	The structure describe source and destination image
`table`	[in]	The table stores histogram projection information
`roi`	[in]	The rectangle you want to calculate histogram projection

Return value

MCV_OK

MCV_NULL_POINTER

Notes

1. The size of table should no less than 8K(8192) in [MLong] Unit, index start from 0.
2. image format (pOffScreen->u32PixelArrayFormat) should be ASVL_PAF_I422H.
3. The buffer size of pOffScreen-> pi32Pitch[3](dst) should be sizeof(MByte)*dstStep*height
4. The width and height of image should be larger than 1.
5. For usage details please refer to the description of mcvCalcHistBackProject_I422HWithRect in mobilecv.h

3.16. Version API

3.16.1. MCV_GetVersion

Description

Version string.

Prototype

```
const MCV_Version *MCV_GetVersion(MVoid);
```

Parameters

Return value

The pointer of MCV_Version.

Notes

Chapter 4: GPU OpenCL API Reference

4.1. Common API

4.1.1. mcvOCLInit

Description

Initialize GPU OpenCL.

Prototype

```
MHandle mcvOCLInit(MHandle hContext);
```

Parameters

`hContext`	[in]	Memory handle returned by *MMemMgrCreate*.

Return value

A Handle of OCL.

Notes

1. Return NULL if failed.

4.1.2. mcvOCLUnInit

Description

Un-initialize GPU OpenCL.

Prototype

```
MInt32 mcvOCLUnInit(MHandle mcvOCLHandle);
```

Parameters

`mcvOCLHandle`	[in]	A OCL handle returned by mcvOCLInit.

Return value

MCV_NULL_POINTER

MCV_OK.

Notes

.

4.1.3. mcvOCLWaitGpu

Description

Wait for GPU to be finished. (clFinish)

Prototype

```
MInt32 mcvOCLWaitGpu(MHandle mcvOCLHandle);
```

Parameters

`mcvOCLHandle`	[in]	A OCL handle returned by mcvOCLInit.

Return value

MCV_NULL_POINTER

MCV_OK.

Notes

1. Blocking wait.

4.2. Matrix Operation API

4.2.1. mcvOCLMatrixMulInit

Description

Initialize Matrix Multiply(Mout = M1 x M2) on GPU.

Prototype

```
MHandle mcvOCLMatrixMulInit(MHandle mcvOCLHandle, MInt32 m1Rows,  MInt32 m1Cols,
MInt32 m2Cols);
```

Parameters

`mcvOCLHandle`	[in]	A OCL handle returned by mcvOCLInit.
`m1Rows`	[in]	The number of rows of M1.
`m1Cols`	[in]	The number of columns of M1 (also the number of rows of M2).
`m2Cols`	[in]	The number of columns of M2

Return value

A handle.

Notes

.

4.2.2. mcvOCLMatrixMulUnInit

Description

Un-initialize Matrix Multiply(Mout = M1 x M2) on GPU.

Prototype

```
MInt32 mcvOCLMatrixMulUnInit(MHandle mcvOCLMatrixHandle);
```

Parameters

mcvOCLMatrixHandle	[in]	A matrix multiply handle returned by mcvOCLMatrixMulInit.

Return value

MCV_OK

MCV_NULL_POINTER.

Notes

.

4.2.3. mcvOCLMatrixMul_RowMajor_f32_begin

Description

Update the matrix size.

Prototype

```
MInt32 mcvOCLMatrixMul_RowMajor_f32_begin(MHandle mcvOCLMatrixHandle, MFloat
*M_out,MFloat *M1_in,MFloat *M2_in,MInt32 m1Rows,  MInt32 m1Cols, MInt32
m2Cols);
```

Parameters

mcvOCLMatrixHandle	[in]	A matrix multiply handle returned by mcvOCLMatrixMulInit.
M_out	[in]	The result matrix.
M1_in	[in]	M1.
M2_in	[in]	M2.
m1Rows	[in]	The number of rows of M1.
m1Cols	[in]	The number of columns of M1 (also the number of rows of M2).
m2Cols	[in]	The number of columns of M2

Return value

MCV_OK

MCV_NULL_POINTER.

MCV_INVALID_CALL.

Notes

1. Start GPU calculation. You can call *mcvOCLMatrixMul_RowMajor_f32_end* to check if GPU has finished calculating Matrix Multiply or use *mcvOCLWaitGpu* to make sure GPU has finished all the work including Matrix Multiply.
2. If the m1Rows, m1Cols, m2Cols are not the same as the ones passed to *mcvOCLMatrixMulInit*, this function will automatically update the parameters before doing GPU calculating.

4.2.4. mcvOCLMatrixMul_RowMajor_f32_end

Description

Update the matrix size.

Prototype

```
MInt32 mcvOCLMatrixMul_RowMajor_f32_end(MHandle mcvOCLMatrixHandle);
```

Parameters

`mcvOCLMatrixHandle` [in] A matrix multiply handle returned by mcvOCLMatrixMulInit.

Return value

MCV_OK
MCV_NULL_POINTER
MCV_INVALID_CALL.

Notes

1. Use event to wait GPU.

4.2.5. Matrix Multiply Instance

1. GPU init

```
MHandle mcvGpuHandle = mcvOCLInit(0);
if(mcvGpuHandle == 0)
{
   printf("haha\n");
}
```

- GPU 初始化函数

2. Init matrix multiply instance

```
mcvOCLMatrixHandle = mcvOCLMatrixMulInit(mcvGpuHandle,row1,column1,column2,0);
if(mcvOCLMatrixHandle == 0)
{
   printf("haha1!!!\n");
}
```

- 矩阵相乘的 GPU 初始化函数(在步骤 1 之下，可以允许初始化多个 GPU 的功能模块如矩阵相乘、金字塔融合等等)

3. Start GPU to do calculating.

mcvOCLMatrixMul_RowMajor_f32_begin (mcvOCLMatrixHandle,M3,M1,M2,row1,column1,column2);

4. Wait GPU to finish calculating

mcvOCLMatrixMul_RowMajor_f32_end(mcvOCLMatrixHandle);

- Or you can call this later, do something else which is independent of the GPU calculation.
- Also, you can call
  mcvOCLWaitGpu(mcvGpuHandle);
  instead.

It is the same as calling *mcvOCLMatrixMul_RowMajor_f32_end* if there is only Matrix Multiply module in your program.

5. Goto step 3 if any other matrix multiply operations exist .

6. Uninit Matrix multiply instance
   mcvOCLMatrixMulUnInit(mcvOCLMatrixHandle);
7. GPU Uninit
   mcvOCLUnInit(mcvGpuHandle);

Chapter 5: Lib Version API Reference

5.1.1. MCV_GetVersion

Description

Version string.

Prototype

```
const MCV_Version *MCV_GetVersion(MVoid);
```

Parameters

Return value

The pointer of MCV_Version.

Notes

.

Chapter 6: Performance Testing Data

Speed up:

performance of android NEON version / performance of android C version.

Or time of android C version / time of android NEON version

Notes:

1. Currently, we support ads1.2/win32/android C /android NEON versions. ads1.2/ win32/android C are compiled with C code, and android NEON is compiled with C&NEON code.
2. More functions will be added in future, please pay your sustained attention to MCV lib.
3. We will add more features such as OpenCL and Qualcomm aDSP in future. They will coming soon.

QA test results, for reference only :

Function name	Speed Up	Notes
mcvAbsDiffu32	1.74	
mcvAbsDiffs32	1.75	
mcvAbsDiffu8	5.73	
mcvAbsDiffVs32	1.91	
mcvAbsDiffVf32	3.03	
mcvColorYUYVtoRGB888u8	6.66	
mcvColorYUYVtoNV21u8	1.57	
mcvColorYUYVtoYUV420u8	2.54	
mcvColorRGB888toYUYVu8	6.86	
mcvColorRGB888toBGR565u8	2.27	
mcvColorRGB888toYUV420u8	9.32	
mcvColorRGB888toNV21u8	8.60	
mcvColorYUV420toYUYVu8	2.21	
mcvColorYUV420toRGB888u8	11.97	
mcvColorNV21toYUV420u8	3.91	
mcvColorNV21toRGB888u8	8.18	
mcvYUYVToOrgData	3.22	
mcvExtract_Y_From_YUYV	2.20	
mcvResizeYUYVToYUYVBilinear	1.30	
mcvResizeYUYVToI422HBilinearY	1.58	
mcvResizeYUYVToLPI422HBilinear	1.25	
mcvResizeYUYVtoI422HDownSampleby2	1.91	
mcvResizeNV21Bilinear	1.42	Newly added
mcvResizeNV12Bilinear	1.38	Newly added
mcvResizeI420Bilinear	1.40	Newly added
McvResizeRGB888Bilinear	1.47	Newly added

mcvResizeSingleComponentBilinear	1.45	
mcvScaleDownBy2u8	4.52	
mcvICmCalc_Bx_By	3.52	
mcvIcmCalc_Bx_By_Gxx_Gxy_Gyy	2.69	
mcviCalcGradientMagAngle_I422H_left	1.27	
mcviCalcGradientMagAngle_I422H_right	1.07	
mcvCalcGradientMagAngleFix_Gray_left	1.11	
mcvCalcGradientMagAngleFix_Gray_right	1.07	
mcvCalcHistBackProject_I422H	2.03	
mcvDetectMotion3FrameDiffY	14.75	
mcvGetMotionCue	2.48	
mcvIntegral	1.17	
mcvBitwiseOru8	3.34	
mcvFastSqrts64	1.39	compare with stdc library
mcvFastSqrts32	1.55	compare with stdc library
mcvDivf32	1.19	
mcvVectorDivf32	27.26	
mcvSqrtf32	2.00	
mcvSqrtVectorf32	36.34	
mcvInvSqrtf32	2.62	
mcvInvSqrtVectorf32	103.43	
mcvVectorDiffNorm2s32	7.22	Newly added
mcvVectorDiffNorm2u32	4.36	Newly added
mcvVectorDiffNorm2f32	3.78	Newly added
mcvVectorDiffNorm2Fasts16	2.58	Newly added
mcvVectorDiffNorm2Fastu16	2.37	Newly added
mcvVectorDiffNorm2Fasts8	2.86	Newly added
mcvVectorDiffNorm2Fastu8	2.65	Newly added
mcvFilterDilate3x3u8	3.11	
mcvFilterErode3x3u8	3.28	
mcvFilterMedian3x3u8	12.93	
mcvFilterBox3x3u8	2.66	
mcvFilterBox3x3u8_2D	2.07	
mcvFilterGaussian5x5u8	12.44	
mcvFilterBoxYUYV	1	Newly added,仅 C 语言版本
mcvFilterThresholdu8	3.95	
mcvSetElementsu8	4.88	
mcvSetElementss32	1.28	