GM8139/GM8136

INTELLIGENT VIDEO ENGINE

User Guide
Rev.: 1.0
Issue Date: October 2014

REVISION HISTORY

GM8139 GM8136 IVS User Guide

Date	Rev.	From	To
Oct. 2014	1.0		Original

Printed in Taiwan 2014

Grain Media, Inc.
5F, No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Grain Media's home page can be found at:
http://www.grain-media.com

TABLE OF CONTENTS

Chapter 1 Introduction 1
1.1 General Description 2
1.2 Related Document 2
Chapter 2 Data Structure 3
2.1 IVS Information Structure 4
2.2 IVS Parameter Structure 4
Chapter 3 User Interface 9
3.1 Description 10
3.2 Basic Function 10
3.2.1 open 10
3.2.2 close 11
3.2.3 mmap 11
3.2.4 munmap 12
3.2.5 ioctl (IVS_INIT) 12
3.2.6 ioctl (IVS_INIT_MEM) 13
3.2.7 ioctl (IVS_HSI_CONVERSION) 13
3.2.8 ioctl (IVS_RGB_CONVERSION) 14
3.2.9 ioctl (IVS_INTEGRAL_IMAGE) 15
3.2.10 ioctl (IVS_SQUARED_INTEGRAL_IMAGE) 16
3.2.11 ioctl (IVS_DE_INTERLEAVING) 17
3.2.12 ioctl (IVS_HISTOGRAM) 18
3.2.13 ioctl (IVS_CONVOLUTION) 19
3.2.14 ioctl (IVS_MORPHOLOGY) 20
3.2.15 ioctl (IVS_SAD) 21
3.2.16 ioctl (IVS_RASTER_OPERATION) 22
3.2.17 ioctl (IVS_CASCADED_CLASSIFIER) 23
Chapter 4 Sample Code 25
4.1 Main File 26
4.1.1 main() 26

4.1.2 ivs_ioctl.h..... 28

LIST OF FIGURES

Figure 2-1. Example of Offset Value 5
Figure 2-2. Example of swap_y_cbcr 6
Figure 2-3. Example of swap_endian 7
Figure 3-1. Example of Integral Image Calculation 16
Figure 3-2. Example of Squared Integral Image Calculation 17
Figure 3-3. Example of Histogram Calculation 18
Figure 3-4. Example of SAD Calculation 22
Figure 3-5. Flowchart of Cascaded Classifier 24

LIST OF TABLES

Table 3-1. Raster Operator 23
Table 3-2. Example of Raster Operation 23

Chapter 1

Introduction

This chapter contains the following sections:

- 1.1 General Description
- 1.2 Related Document

1.1 General Description

This document describes the user interface of the GM8139/GM8136 Intelligent Video engine (IVS). The supported functions of GM8139 IVS include the color space transform (YUV to HSI), color space transform (YUV to RGB), integral image calculation, squared integral image calculation, de-interleave YC, histogram calculation, convolution operation, morphology operation, Sum of Absolute Differences (SAD) calculation, raster operation, and cascaded classifier. The supported input formats of GM8139/GM8136 IVS are the packed format 4:2:2, planar format, or binary format.

1.2 Related Document

There is no related document available at this release.

Chapter 2

Data Structure

This chapter contains the following sections:

- 2.1 IVS Information Structure
- 2.2 IVS Parameter Structure

2.1 IVS Information Structure

```
typedef struct ivs_info_t
{
    int fd;
    unsigned char *mmap_addr;
} ivs_info_t;
```

Element	Description
int fd	Node number
unsigned char *mmap_addr	Memory address after memory mapping

2.2 IVS Parameter Structure

```
typedef struct ivs_ioctl_param_t {
    ivs_ioctl_io_t      input_img_1;
    ivs_ioctl_io_t      input_img_2;

    ivs_ioctl_io_t      output_rlt_1;
    ivs_ioctl_io_t      output_rlt_2;
    ivs_ioctl_io_t      output_rlt_3;

    ivs_img_t           img_info;
    ivs_ioctl_mem_t     mem_info;
    ivs_operation_t     operation_info;
} ivs_ioctl_param_t;
```

Element		Description
ivs_ioctl_io_t	input_img_1	Parameter of the input image 1
ivs_ioctl_io_t	input_img_2	Parameter of the input image 2
ivs_ioctl_io_t	output_rlt_1	Parameter of result 1
ivs_ioctl_io_t	output_rlt_2	Parameter of result 2

Element		Description
ivs_ioctl_io_t	output_rlt_3	Parameter of result 3
ivs_img_t	img_info	Parameter of the input image information
ivs_ioctl_mem_t	mem_info;	Parameter of the memory information
ivs_operation_t	operation_info;	Parameter of the operation information

```
typedef struct ivs_ioctl_io_t {
    int   addr;
    int   size;
    int   offset;
} ivs_ioctl_io_t;
```

Element	Description
int addr	The index of the memory block and the value of the index cannot exceed the total number of the memory blocks.
int size	The input image size (Byte)
int offset	The offset value of the input and output images

offset = image pitch – destination width;

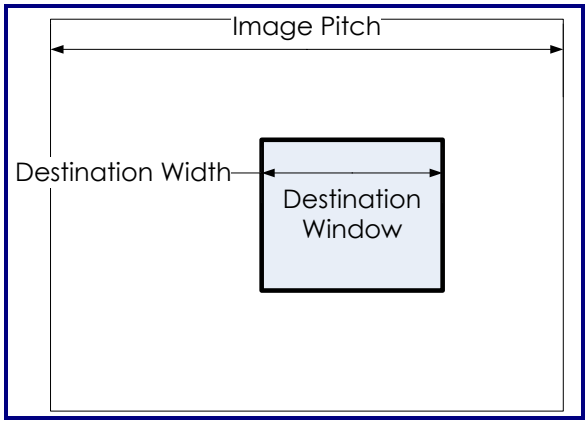

Figure 2-1. Example of Offset Value

```
typedef struct ivs_img_t {
    int   img_width;
    int   img_height;
    int   img_format;
    int   swap_y_cbcr;
    int   swap_endian;
} ivs_img_t;
```

Element	Description
int img_width	Input width Max.: 1920 Min.: 16 Min. binary image: 128
int img_height	Input height
int img_format	Input image format 4: II image or SII image 2: Packed format 1: Planar format 0: Binary format
int swap_y_cbcr	Change the Y CBCR order of the input image with the packed format
int swap_endian	Change the data order of the input image with the planar format

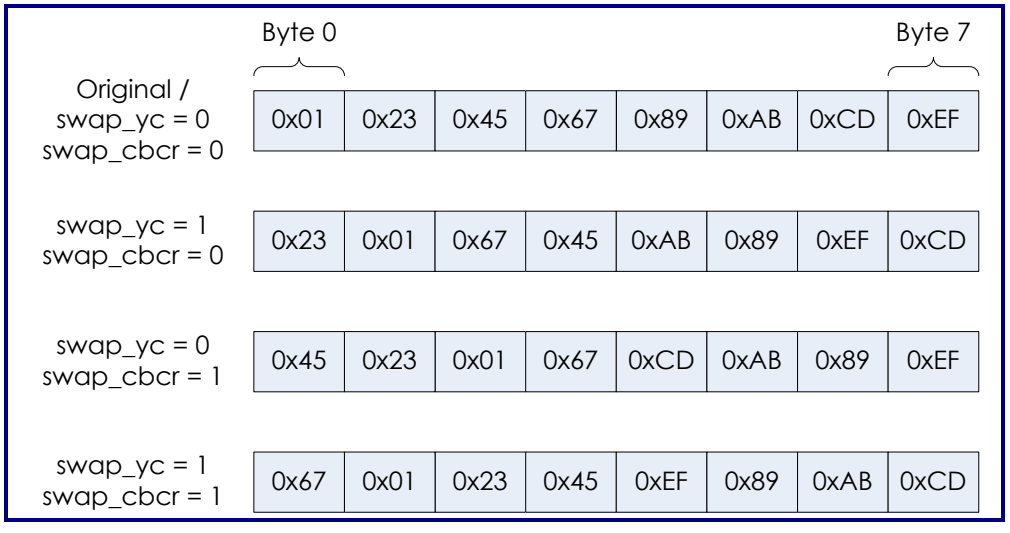

Figure 2-2. Example of swap_y_cbcr

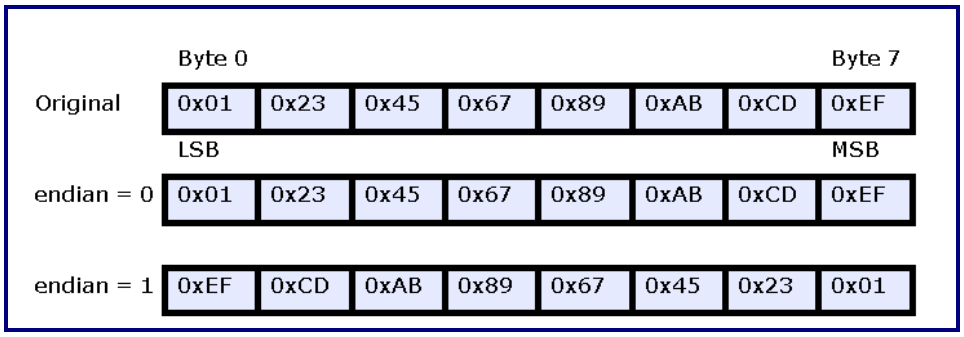

Figure 2-3. Example of swap_endian

```
typedef struct ivs_ioctl_mem_t {
    int   memory_size;
    int   memory_num;
} ivs_ioctl_mem_t;
```

Element	Description
int memory_size	Size of the memory block
int memory_num	Number of the memory blocks

```
typedef struct ivs_operation_t {
    int   operation;
    int   kernel_template_index;
    int   shifted_convolution_sum;
    int   morphology_operator;
    int   select_threshold_output;
    int   threshold;
    int   block_size;
    int   convolution_tile;
    int   raster_operationo_code ;
    int   integral_cascade ;
    int   template_element[25] ;
} ivs_ operation _t;
```

Element		Description
int operation		Category of operation 0: YUV to HSI 1: YUV to RGB 2: Integral image calculation 3: Squared integral image calculation 4: De-interleave YC 5: Histogram calculation 6: Convolution operation 7: Morphology operation 8: Sum of Absolute Differences (SAD) calculation 9: Raster operation 10: Cascaded classifier
int	kernel_template_index;	Index of matrix which is used for the convolution operation and morphology operation
int	shifted_convolution_sum;	Shift convolution result
int	morphology_operator;	Morphology operator
int	select_threshold_output;	Indicates when the morphology result exceeds the threshold, and the output is 1 or original value
int	threshold;	Threshold parameter for the convolution operation, morphology operation, and SAD calculation
int	block_size;	Indicates the block size for the SAD calculation
int	raster_operationo_code ;	Raster operator
int	integral_cascade ;	Integrates the integral image and cascaded classifier for the face detection application
int	template_element[25] ;	Matrix for the morphology operation and convolution operation

Chapter 3

User Interface

This chapter contains the following sections:

- 3.1 Description
- 3.2 Basic Function

3.1 Description

The following table lists the user interfaces of GM8139/GM8136 IVS. Users can initialize IVS, start IVS, and perform certain operations by using the following interfaces.

Function Name	Description
open()	Open driver node
close()	Close driver node
mmap()	Map the user space memory to the kernel space memory
munmap()	Un-map the user space memory to the kernel space memory
ioctl ( );	Initial IVS and perform certain operation by different parameters

3.2 Basic Function

3.2.1 open

open	
Description	Open the driver node
Template	int open(const char *pathname, int flags)
Parameter	**const char *pathname** Node name **int flags** Default value: O_RDWR
Return value	> 0: Successful (Node number) < 0: Failure

3.2.2 close

close	
Description	Close the driver node
Template	int close(int fd)
Parameter	**int fd:** Node number
Return value	0: Successful -1: Failure

3.2.3 mmap

mmap	
Description	Map the user space memory to the kernel space memory
Template	void *mmap (void *addr, size_t length, int prot, int flags, int fd, off_t offset)
Parameter	**void *addr** Default value: 0 **size_t length** Mapping memory size Default value: 1080 * 1920 * 10 **int prot** Default value: PROT_READ | PROT_WRITE **int flags** Default value: MAP_SHARED **int fd** Node number **off_t offset** Default value: 0
Return value	≥ 0: Successful -1: Failure

3.2.4 munmap

munmap	
Description	Un-map the user space memory to the kernel space memory
Template	munmap (void *addr, size_t length)
Parameter	**void *addr** Pointer of the user space memory **size_t length** Mapping memory size Default size: 1080 * 1920 * 10.
Return value	0: Successful -1: Failure

3.2.5 ioctl (IVS_INIT)

ioctl (IVS_INIT)	
Description	Initialize the IVS engine
Template	ioctl(int fd, IVS_INIT, int *API_version)
Parameter	**int fd** Node number **int *API_version** AP version
Return value	0: Successful -1: Failure

3.2.6 ioctl (IVS_INIT_MEM)

ioctl (IVS_INIT_MEM)	
Description	Initialize the IVS memory
Template	ioctl(int fd, IVS_INIT_MEM, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Failure

Before calling ioctl (IVS_INIT_MEM), users must set ivs_param.img_info.img_width and ivs_param.img_info.img_size. After calling ioctl (IVS_INIT_MEM), ivs_param.mem_info.memory_size will record the memory block size(img_width * img_width * 2), ivs_param.mem_info.memory_num will record the memory block number.

3.2.7 ioctl (IVS_HSI_CONVERSION)

ioctl (IVS_HSI_CONVERSION)	
Description	Perform the color space transform (YUV to HIS)
Template	ioctl(int fd, IVS_HSI_CONVERSION, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from user error -3: Copy to user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: (H) output_rlt_2: (S) output_rlt_3: (I)

- Color space transform (YUV to HSI) only supports (Packed format) 4:2:2.

$$R = 1.164(Y - 16) + 1.596(Cr - 128)$$
$$G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.392(Cb - 128)$$
$$B = 1.164(Y - 16) + 2.017(Cb - 128)$$

Formula 3-1. Formula of Color Space Transform (YUV to HSI)

3.2.8 ioctl (IVS_RGB_CONVERSION)

ioctl (IVS_RGB_CONVERSION)	
Description	Perform the color space transform (YUV to RGB)
Template	ioctl(int fd, IVS_RGB_CONVERSION, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from user error -3: Copy to user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: (R) output_rlt_2: (G) output_rlt_3: (B)

- Color space transform (YUV to RGB) only supports the packed format 4:2:2.

$$R = 1.164(Y - 16) + 1.596(Cr - 128)$$
$$G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.392(Cb - 128)$$
$$B = 1.164(Y - 16) + 2.017(Cb - 128)$$

$$H = \begin{cases} \theta & \text{if } B \le G \\ 2\pi - \theta & \text{if } B > G \end{cases}$$

$$\theta = \cos^{-1}\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}} \right)$$

S = 128(1 – 3xmin(R,G,B)/(R + G + B))

I = (R + G + B)/3

Formula 3-2. Formula of Color Space Transform (YUV to RGB)

3.2.9 ioctl (IVS_INTEGRAL_IMAGE)

ioctl (IVS_INTEGRAL_IMAGE)	
Description	Perform the integral image calculation
Template	ioctl(int fd, IVS_INTEGRAL_IMAGE, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from user error -3: Copy to user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: Integral image

- Integral image calculation supports the packed format 4:2:2 and planar format.

$ii(x, y) = \Sigma\, i(x', y'), \quad x' \le x, y' \le y$

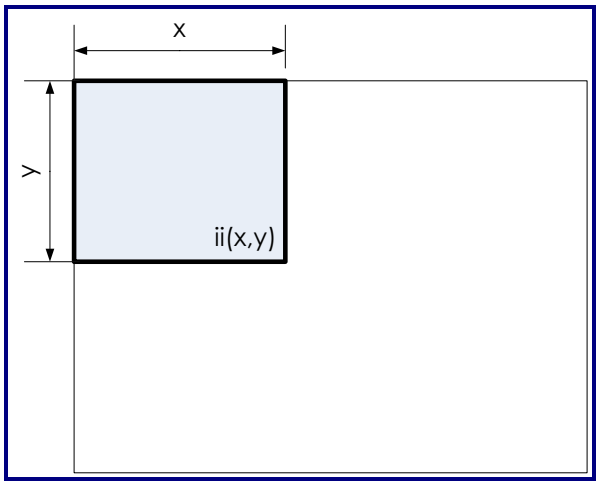

Figure 3-1. Example of Integral Image Calculation

3.2.10 ioctl (IVS_SQUARED_INTEGRAL_IMAGE)

ioctl (IVS_SQUARED_INTEGRAL_IMAGE)	
Description	Perform the squared integral image calculation
Template	ioctl(int fd, IVS_SQUARED_INTEGRAL_IMAGE, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from user error -3: Copy to user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: Squared integral image

- Squared integral image calculation supports the packed format 4:2:2 and planar format.

$ii(x, y) = \Sigma (i(x', y') \times i(x', y')), \quad x' \leq x, y' \leq y$

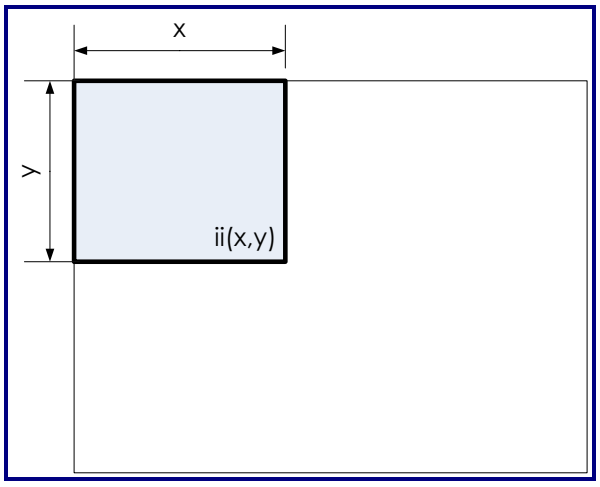

Figure 3-2. Example of Squared Integral Image Calculation

3.2.11 ioctl (IVS_DE_INTERLEAVING)

ioctl (IVS_DE_INTERLEAVING)	
Description	Perform de-interleave YC
Template	ioctl(int fd, IVS_DE_INTERLEAVING, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from user error -3: Copy to user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: (Y) output_rlt_2: (CbCr)

- De-interleave YC only supports the packed format 4:2:2.

3.2.12 ioctl (IVS_HISTOGRAM)

ioctl (IVS_HISTOGRAM)	
Description	Perform the histogram calculation
Template	ioctl(int fd, IVS_HISTOGRAM, ivs_ioctl_param_t *ivs_param)
Parameter	int fd Node number ivs_ioctl_param_t *ivs_param IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from the user error -3: Copy to the user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: Histogram

- Histogram calculation supports the packed format 4:2:2 and planar format.

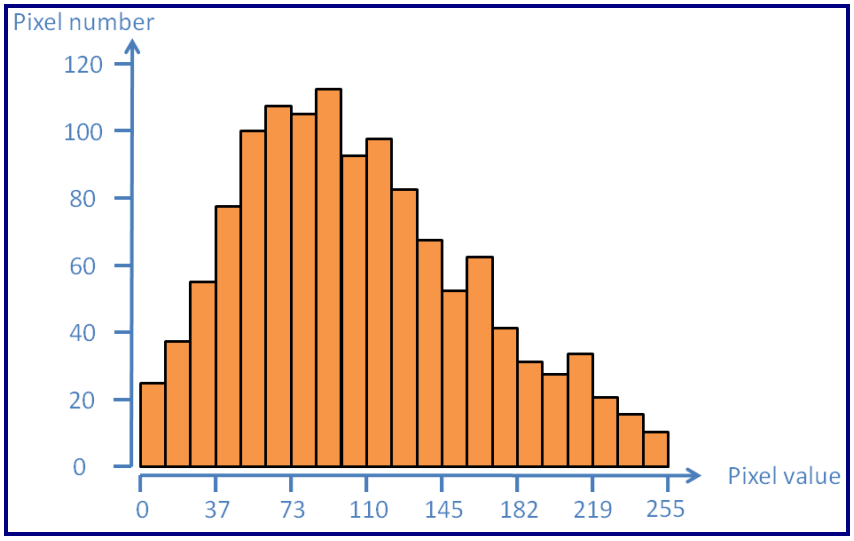

Figure 3-3. Example of Histogram Calculation

3.2.13 ioctl (IVS_CONVOLUTION)

ioctl (IVS_CONVOLUTION)	
Description	Perform the convolution operation
Template	ioctl(int fd, IVS_CONVOLUTION, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from the user error -3: Copy to the user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: Magnitude of gradient vector output_rlt_2: Y component of gradient vector output_rlt_3: X component of gradient vector (shift = 0) or convolution sum (shift != 0)

1. Convolution operation only supports the planar format.
2. Setting the shifted_convolution_sum value x would shift the convolution sum x bit, if the value is larger than 255 after shifting, please replace it by 255.
3. Template_element will store the used mask, and kernel_template_index will indicate the mask position.
4. Setting the threshold value can binary magnitude of the gradient vector, if the magnitude of the gradient vector is large, the threshold output will be 1; else the threshold output will be 0.

$$sum = \left(\sum_{j=-2}^{2} \sum_{i=-2}^{2} i(x+i, y+j) \times mask(i+2, j+2) \right) >> shift$$

$$G_x = \sum_{j=-2}^{2} \sum_{i=-2}^{2} i(x+i, y+j) \times mask(i+2, j+2)$$

$$G_y = \sum_{j=-2}^{2} \sum_{i=-2}^{2} i(x+i, y+j) \times mask(j+2, i+2)$$

$$magnitude = |G_x| + |G_y|$$

Formula 3-3. Formula of Convolution Operation

3.2.14 ioctl (IVS_MORPHOLOGY)

ioctl (IVS_MORPHOLOGY)	
Description	Perform the morphology operation
Template	ioctl(int fd, IVS_MORPHOLOGY, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Bode number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from user error -3: Copy to user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: Result of the morphology operation

1. Morphology operation supports the planar format and binary image.
2. Set select_threshold_output to 0, and the output is 1 if the morphology operation is larger than the threshold, else the output is 0.
3. Template_element will store the used mask, and kernel_template_index indicates the mask position.
4. Set morphology_operator to choose the operator:
   0: Dilation
   1: Close
   2: Erosion
   3: Open
   7: Binary

3.2.15 ioctl (IVS_SAD)

ioctl (IVS_SAD)	
Description	Perform the SAD calculation
Template	ioctl(int fd, IVS_SAD, ivs_ioctl_param_t *ivs_param)
Parameter	int fd Node number ivs_ioctl_param_t *ivs_param IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from the user error -3: Copy to the user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: SAD value

1. SAD calculation only supports the planar format.
2. Setting threshold value can binary the SAD result, if the SAD result exceeds the threshold, the output is 1, else the output is 0.
3. Users can identify the block size.
   Block_size:
   0: 1x1
   1: 3x3
   2: 5x5

$$SAD(x,y)=\sum_{j=-n}^{n}\sum_{i=-n}^{n}\left|A(x+i,y+j)-B(x+i,y+j)\right|$$

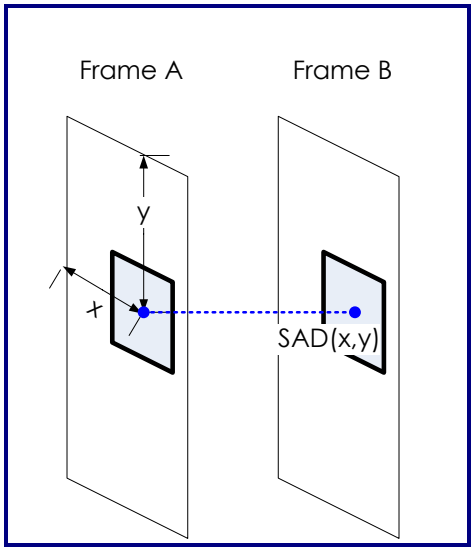

Figure 3-4. Example of SAD Calculation

3.2.16 ioctl (IVS_RASTER_OPERATION)

ioctl (IVS_RASTER_OPERATION)	
Description	Perform the raster operation
Template	ioctl(int fd, IVS_RASTER_OPERATION, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from the user error -3: Copy to the user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: Raster operation

1. Raster operation only supports the planar format.
2. Users can set the raster_operationo_code value (0 ~ 15) to decide the raster operator.

Table 3-1. Raster Operator

Channel 1	Channel 0	Operation Bit
0	0	Bit 0
0	1	Bit 1
1	0	Bit 2
1	1	Bit 3

Table 3-2. Example of Raster Operation

Operation Result	Operator
Channel 0	0b1010
Channel 1	0b1100
Channel 0 and Channel 1	0b1000
Channel 0 or Channel 1	0b1110

3.2.17 ioctl (IVS_CASCADED_CLASSIFIER)

ioctl (IVS_CASCADED_CLASSIFIER)	
Description	Perform the cascaded classifier for the face detection application
Template	ioctl(int fd, IVS_CASCADED_CLASSIFIER, ivs_ioctl_param_t *ivs_param)
Parameter	**int fd** Node number **ivs_ioctl_param_t *ivs_param** IVS parameter
Return value	0: Successful -1: Unknown ioctl -2: Copy from the user error -3: Copy to the user error -4: IVS engine error -5: Software time out
Output	output_rlt_1: Face detection result

1. Cascaded classifier supports the packed format 4:2:2 and planar format.
2. By setting integral_cascade = 1, IVS will first perform the integral image calculation squared integral

image, than perform the cascaded classifier for the face detection application.

3. If integral_cascade = 0, the input image must be the integral image and squared integral image.

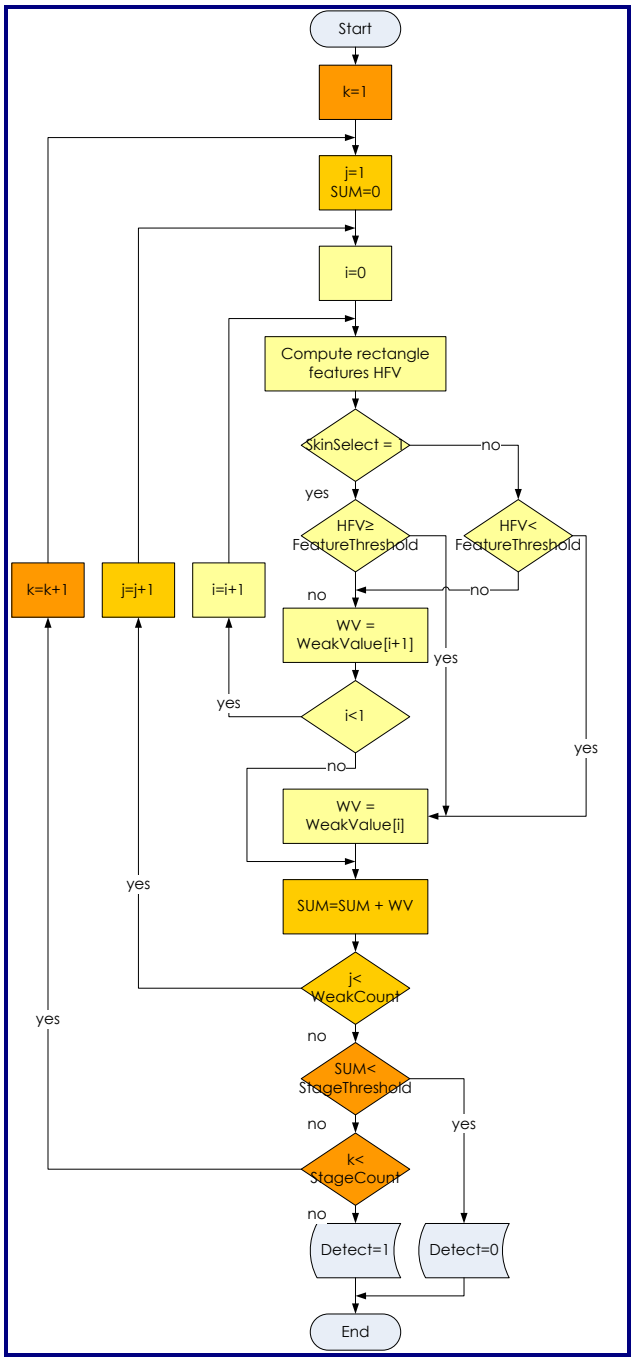

Figure 3-5. Flowchart of Cascaded Classifier

Chapter 4

Sample Code

This chapter contains the following sections:

- 4.1 Main File
- 4.2 ivs_ioctl.h

4.1 Main File

The sample code describes how to start GM8139/GM8136 IVS engine and perform certain operations.

4.1.1 main()

```
#include "ivs_ioctl.h"

int main(int argc, char ** argv)
{
    /*variables*/
    int i = 0;
       int ret = 0;
       int offset = 0;
    char *input_fn_1=”test_img_1”,  *input_fn_2=”test_img_2”;

    //  output pointer
    unsigned char *result_1 = NULL, *result_2 = NULL,
                                                    *result_3 = NULL;

       //  ivs_driver parameter
    ivs_info_t ivs_info;
    ivs_ioctl_param_t ivs_param;

    //  initial_driver
    ivs_param.img_info.img_width  = max_width;
    ivs_param.img_info.img_height = max_height;
    ret = driverInit(&ivs_info,&ivs_param);
    if (ret < 0)
        return ret;
    ivs_param.img_info.img_width  = img_width;
    ivs_param.img_info.img_height = img_height;

    //  open & read input image
    ivs_param.input_img_1.addr = 0;
    ret = readImg(ivs_info, &ivs_param, input_fn_1, 1);
    if (ret < 0)
        goto error;

    ivs_param.input_img_2.addr = 1;
    ret = readImg(ivs_info, &ivs_param, input_fn_2, 2);
    if (ret < 0)
        goto error;


    ivs_param.input_img_1.offset = offset;
    ivs_param.input_img_2.offset = offset;
    ivs_param.output_rlt_1.offset = offset;
    ivs_param.output_rlt_2.offset = offset;
    ivs_param.output_rlt_3.offset = offset;

    ivs_param.output_rlt_1.addr = 2;
    ivs_param.output_rlt_2.addr = 3;
```

```
    ivs_param.output_rlt_3.addr = 4;

    switch (operation) {
             case 0:
             ret = ioctl(ivs_info.fd, IVS_HSI_CONVERSION, ivs_param);
         break;

         case 1:
                  ret = ioctl(ivs_info.fd, IVS_RGB_CONVERSION, ivs_param);
         break;

         case 2:
             ret = ioctl(ivs_info.fd, IVS_INTEGRAL_IMAGE, ivs_param);
         break;

         case 3:
             ret = ioctl(ivs_info.fd, IVS_SQUARED_INTEGRAL_IMAGE, ivs_param);
         break;

         case 4:
             ret = ioctl(ivs_info.fd, IVS_DE_INTERLEAVING, ivs_param);
         break;

         case 5:
             ret = ioctl(ivs_info.fd, IVS_HISTOGRAM, ivs_param);
         break;

         case 6:
             ret = ioctl(ivs_info.fd, IVS_CONVOLUTION, ivs_param);
         break;

         case 7:
             ret = ioctl(ivs_info.fd, IVS_MORPHOLOGY, ivs_param);
         break;

         case 8:
             ret = ioctl(ivs_info.fd, IVS_SAD, ivs_param);
         break;

         case 9:
             ret = ioctl(ivs_info.fd, IVS_RASTER_OPERATION, ivs_param);
         break;

         case 10:
             ret = ioctl(ivs_info.fd, IVS_CASCADED_CLASSIFIER, ivs_param);
         break;
    }

/*intelligent video engine end*/
ivs_close(ivs_info);
return ret;
}
```

4.1.2 ivs_ioctl.h

```
#ifndef _IOCTL_IVS_H_

    #define _IOCTL_IVS_H_

    #define IVS_VER    0x00200100
    //                              HIIFFFBB
    //                              H           huge change
    //                                II            interface change
    //                                  FFF           functional modified or bug fixed
    //                                       BB           branch for customer request

    #define IVS_INIT                                    0x4170
    #define IVS_INIT_MEM                            0x4171
    #define IVS_HSI_CONVERSION                  0x4172
    #define IVS_RGB_CONVERSION                0x4173
    #define IVS_INTEGRAL_IMAGE                  0x4174
    #define IVS_SQUARED_INTEGRAL_IMAGE      0x4175
    #define IVS_DE_INTERLEAVING                 0x4176
    #define IVS_HISTOGRAM                           0x4177
    #define IVS_CONVOLUTION                         0x4178
    #define IVS_MORPHOLOGY                      0x4179
    #define IVS_SAD                                     0x4180
    #define IVS_RASTER_OPERATION                0x4181
    #define IVS_CASCADED_CLASSIFIER             0x4182

    #define IVS_DEV                             "/dev/ivs"

    typedef struct ivs_ioctl_io_t {
        int addr;
        int size;
        int offset;
    } ivs_ioctl_io_t;

    typedef struct ivs_ioctl_mem_t {
        int memory_size;
        int memory_num;
    } ivs_ioctl_mem_t;

    typedef struct ivs_io_t {
        int size;
        int faddr;
        int offset;
        unsigned char *vaddr;
    } ivs_io_t;

    typedef struct ivs_img_t {
        int img_format;
        int img_height;
        int img_width;
        int swap_y_cbcr;
        int swap_endian;
    } ivs_img_t;

    typedef struct ivs_operation_t{
```

```
        int operation ;
        int kernel_template_index ;
        int shifted_convolution_sum ;
        int morphology_operator ;
        int select_threshold_output ;
        int threshold ;
        int block_size ;
       int convolution_tile;
        int raster_operationo_code ;
        int integral_cascade ;
        int template_element[25] ;
    } ivs_operation_t;

    typedef struct ivs_ioctl_param_t {
        ivs_ioctl_io_t  input_img_1;
        ivs_ioctl_io_t  input_img_2;

        ivs_ioctl_io_t  output_rlt_1;
        ivs_ioctl_io_t  output_rlt_2;
        ivs_ioctl_io_t  output_rlt_3;

        ivs_img_t       img_info;
        ivs_ioctl_mem_t mem_info;
        ivs_operation_t   operation_info;
    } ivs_ioctl_param_t ;

    typedef struct ivs_param_t {
        ivs_io_t  input_img_1;
        ivs_io_t  input_img_2;

        ivs_io_t  output_rlt_1;
        ivs_io_t  output_rlt_2;
        ivs_io_t  output_rlt_3;

        ivs_img_t       img_info;
        ivs_operation_t   operation_info;
    } ivs_param_t ;

    typedef struct ivs_info_t{
        int fd;
        unsigned char *mmap_addr;
    } ivs_info_t;

#endif
```