Camera Tampering Detection User Manual

V1.2

REVISION HISTORY

Revision No.	Description	Date
1.0	• Created.	03/21/2017
1.1	• Return value integration	04/27/2017
1.2	• Add MI_OD_SetMotionSensitivity()	09/20/2017

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Purpose

Occlusion detection is a function used for detecting whether someone sabotage the camera. Such behaviors include (1) occlude the camera with an opaque object, (2) alter the lens focus to make it defocus, and (3) displace the camera to change the field of view. The applications for this function are, for example, security monitoring.

2. API REFERENCE

2.1. API Overview

- MI_OD_Init: Initialize OD library.
- MI_OD_Uninit: Release memory of OD library.
- MI_OD_SetAttr: Set OD parameters.
- MI_OD_SetWindowEnable: Set OD enable for certain sub-window.
- MI_OD_GetWindowResult: Get result for certain OD sub-window.
- MI_OD_Run: Run OD library.
- MI_OD_SetMotionSensitivity: Set the sensitivity of moving lens alarm.

2.2. API Lists

MI_OD_Init

Purpose

Initialize OD library.

Function Prototype

OD_HANDLE MI_OD_Init(S32 inImgW, S32 inImgH, ODColor_e nClrType, ODWindow_e div) ;

Arguments

Name	**Description**
inImgW	Width of source image.
inImgH	Height of source image.
nClrType	Image type
div	Window type

Return value

Return value	**Description**
OD_HANDLE	Handle
NULL	Error

Requirement

Header files: mi_od.h

Library files: libMTE_LINUX.so

Note

Enter low resolution to avoid high CPU usage, the recommended value is 320X180.

MI_OD_Uninit

Purpose

Release memory of OD library.

Function Prototype

void MI_OD_Uninit(OD_HANDLE odHandle) ;

Arguments

Name	**Description**
odHandle	Handle

Return value

None

Requirement

Header files: mi_od.h

Library files: libMTE_LINUX.so

Note

None

MI_OD_SetAttr

Purpose

Set OD parameters.

Function Prototype

MI_RET MI_OD_SetAttr(OD_HANDLE odHandle, S32 thd_tamper, S32 tamper_blk_thd, S32 min_duration, S32 alpha, S32 M);

Arguments

Name	Description
odHandle	Handle
thd_tamper	If the ratio of certain sub-window of current image to the same sub-window of reference image is smaller than this threshold, then this sub-window is considered as tampered.
tamper_blk_thd	If total block numbers classified as tampered exceeds this amount, this frame is considered as tampered.
min_duration	If the frame considered as tampered elapses over this amount, then the final result returns true.
alpha	Control learning rate for generating reference image.
M	How many frames will update reference image.

Return value

Return value	Description
MI_OD_RET_SUCCESS	Success
MI_OD_RET_INVALID_PARAMETER	Parameter error.

Requirement

Header files: mi_od.h

Library files: libMTE_LINUX.so

Note

- thd_tamper : 0~10.
- tamper_blk_thd : The set value can not be exceeded number of windows divided.
- If the parameter of window type has been set to OD_WINDOW_3X3(total 9 sub-windows) in MI_OD_Init. When tamper_blk_thd=4, MI_OD_Run return 1 if more than 4 occluded sub-windows.

- alpha : 0~10.
- The larger of min_duration the time to detect occluded longer.
- The sensitivity of MI_OD_Run can be adjusted through set parameters tamper_blk_thd and min_duration.Three sets of parameters recommended as table:

Sensitivity	High	Medium	Low
tamper_blk_thd	2	4	8
min_duration	5	15	30

MI_OD_SetWindowEnable

Purpose

Set OD enable for certain sub-window.

Function Prototype

MI_RET MI_OD_SetWindowEnable(OD_HANDLE odHandle, S32 col, S32 row, S32 bEnable);

Arguments

Name	Description
odHandle	odHandle
col	Horizontal ndex of sub-window.
row	Vertical index of sub-window.
bEnable	Set 1 to enable, otherwise disable.

Return value

Return value	Description
MI_RET_SUCCESS	Success
MI_OD_RET_INVALID_HANDLE	null handle
MI_OD_RET_INVALID_WINDOW	Set window error.

Requirement

Header files: mi_od.h
Library files: libMTE_LINUX.so

Note

Preset all sub-windows are enabled. When all sub-windows are in the disable state, OD does not work.

MI_OD_GetWindowResult

Purpose

Get result for certain OD sub-window.

Function Prototype

MI_OD_WIN_STATE MI_OD_GetWindowResult(OD_HANDLE odHandle, S32 col, S32 row);

Arguments

Name	Description
odHandle	odHandle
col	Horizontal index of sub-window.
row	Vertical index of sub-window.

Return value

Return value	Description
MI_OD_WIN_STATE_TAMPER	window occlusion
MI_OD_WIN_STATE_NON_TAMPER	window non occlusion
MI_OD_WIN_STATE_NO_FEATURE	not enough features
MI_OD_WIN_STATE_FAIL	function fail

Requirement

Header files: mi_od.h

Library files: libMTE_LINUX.so

Note

The final result of OD is based on the return value of MI_OD_RUN.

MI_OD_Run

Purpose

Run OD library.

Function Prototype

S32 MI_OD_Run(OD_HANDLE odHandle, const U8 * yImage);

Arguments

Name	Description
odHandle	odHandle
yImage	Preview buffer address.

Return value

Return value	Description
-1	Function fail
1	Occlusion detected.
0	Did not detect occlusion.

Requirement

Header files: mi_od.h

Library files: libMTE_LINUX.so

Note

Run at low frame rate to avoid high CPU usage, the recommended value is 3~5.

MI_OD_SetMotionSensitivity

Purpose

Set the sensitivity of moving lens alarm.

Function Prototype

MI_RET MI_OD_SetMotionSensitivity(OD_HANDLE odHandle, U8 level);

Arguments

Name	**Description**
odHandle	odHandle
level	The sensitivity of moving lens alarm.

Return value

Return value	**Description**
MI_RET_SUCCESS	Success
MI_OD_RET_INVALID_HANDLE	null handle
MI_OD_RET_INVALID_WINDOW	Set window error.

Requirement

Header files: mi_od.h

Library files: libMTE_LINUX.so

Note

- This is a optional function.
- The level value is set as a percentage, range 0 to 100, the larger of level value the more sensitivity to detect occluded by moving lens.

3. DATA TYPE

3.1. Overview

ODColor_e	OD image type.
ODWindow_e	OD windows type.
MI_OD_WIN_STATE	Result of OD windows.
MI_RET	OD function return state.

3.2. Struct Lists

ODColor_e

Description
OD image type.

Syntax
```
typedef enum
{
        OD_Y = 1,
        OD_COLOR_MAX
} ODColor_e;
```

Member

Member	Description
OD_Y	Y component of source image YUV.
OD_COLOR_MAX	Maximum of input image type.

ODWindow_e

Description
Type of OD windows, the recommended value is OD_WINDOW_3X3, for test.

Syntax
```
typedef enum
{
        OD_WINDOW_1X1 = 0,
        OD_WINDOW_2X2,
        OD_WINDOW_3X3,
        OD_WINDOW_MAX
} ODWindow_e;
```

Member

Member	Description
OD_WINDOW_1X1	1 sub-window.
OD_WINDOW_2X2	4 sub-windows.
OD_WINDOW_3X3	9 sub-windows.
OD_WINDOW_MAX	Maximum of window type.

MI_OD_WIN_STATE

Description
Result of OD windows.

Syntax
```
typedef enum _MI_OD_WIN_STATE
{
        MI_OD_WIN_STATE_TAMPER = 0,
        MI_OD_WIN_STATE_NON_TAMPER = 1,
        MI_OD_WIN_STATE_NO_FEATURE = 2,
        MI_OD_WIN_STATE_FAIL = -1,
} MI_OD_WIN_STATE;
```

Member

Member	Description
MI_OD_WIN_STATE_TAMPER	window occlusion
MI_OD_WIN_STATE_NON_TAMPER	window non occlusion
MI_OD_WIN_STATE_NO_FEATURE	not enough features
MI_OD_WIN_STATE_FAIL	function fail

MI_RET

Description
OD function return state.

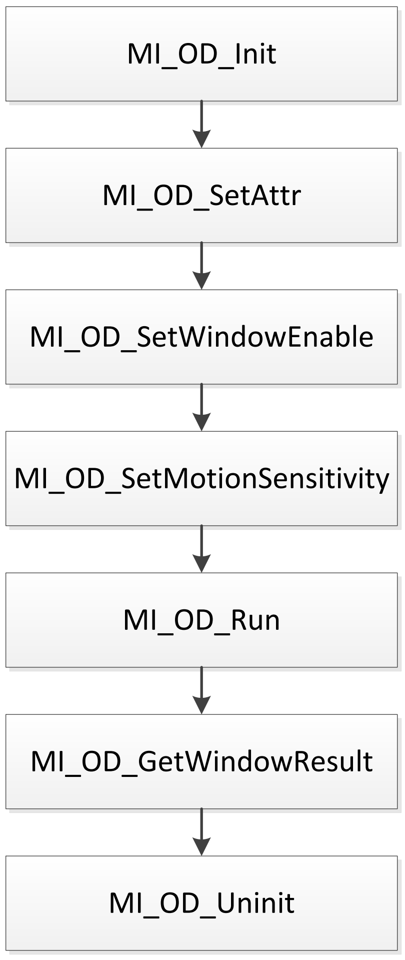
Syntax
```
typedef enum _MI_RET_E
```

```
{
    MI_RET_SUCCESS                          = 0x00000000,
    MI_OD_RET_INVALID_HANDLE                = 0x10000503,   /*Invalid OD handle*/
    MI_OD_RET_INVALID_PARAMETER             = 0x10000504,   /*Invalid OD parameter*/
    MI_OD_RET_INVALID_WINDOW                = 0x10000505,   /*Invalid window*/
} MI_RET;
```

Member

Member	**Description**
MI_RET_SUCCESS	Function success
MI_OD_RET_INVALID_HANDLE	OD handle is null.
MI_OD_RET_INVALID_PARAMETER	Parameter error.
MI_OD_RET_INVALID_WINDOW	Set window error.

4. OD FLOW


MI_OD_Init
MI_OD_SetAttr
MI_OD_SetWindowEnable
MI_OD_SetMotionSensitivity
MI_OD_Run
MI_OD_GetWindowResult
MI_OD_Uninit

5. EXAMPLE

Sample code: \IE\video\MTE\I3\sample\OD\mi_sample_od.c