meerecompany

meerecompany Inc.

Cube Eye MTF_API Manual

Version 1.2 - 03.31.2016

This manual is written for using meerecompany's Cube Eye API.

cube eye
3D RANGE CAMERA meerecompany

meerecompany Inc., 69-12, Jeongmunsongsan-ro, Yanggam-myeon, Hwaseong-si, Gyounggi-do, 445-933, Korea
Tel. +82.31.350.9999, Fax. +82.31.352.8871
www.cube-eye.co.kr / www.meerecompany.com

Table of contents

1. Introduction

This programming is written to explain APIs to activate and use meerecompany 3D Range Camera [Cube-Eye].

This software package described in this manual includes meerecompany's intellectual property thus should not be copied, reproduced, or modified for any use without the permission of meerecompany.

1.1 System Requirements

◎ **For Windows:**

- Microsoft Windows XP/7/8 (Both of x86, x64)
- Microsoft Visual Studio C++ 2010/2012

◎ **For Linux:**

- Linux Operating System(Ubuntu 10.04, 12.04, 14.04 (Both of x86, x64))
- Qt version 5.2(Tool program)

◎ **For Arm Board:**

- Test Arm CPU: Exynos4412(A9 Core)
- Compiler Version: arm-none-linux-gnueabi, arm-linux-gnueabihf
- Kernel V4L2(Video4Linux2) & UVC(USB VIDEO CLASS)

◎ **For Android:**

- root permission required
- Test Version : Exynos4412(Ice Cream Sandwich, Jelly Bean)
- Android Sutdio(SampleRead, Cube Eye Viewer)

1.2 Revision History

◎ **Version 1.0:**

06. 01. 2015 , Initial distribution.

◎ **Version 1.1:**

12. 04. 2015 , Add "mtfSetMultiSyncMode()" on Page 15 ,
Add "mtfGetMultiSyncMode ()" on Page 16 ,
Add "mtfSetFpsDelay()" on Page 16 ,
Add "mtfGetFpsDelay()" on Page 16 ,
Add "mtfSetUndistortion()" on Page 17 .

◎ **Version 1.2:**

03. 31. 2016 , Add "Android application build" on page 03,
03. 31. 2016 , Add "Android Studio Project Guide" on page 20.

2. Application build

2.1 Windows application build

MTF_API library is necessary for Windows application developments.

◎ MTF_API.h: This file contains the declarations of all available MTF_API functions. This must be included in the application source code.
(C:\Cube Eye\include)

◎ MTF_API.lib: The import library to be linked to the application.
(C:\Cube Eye\x86[x64]\debug[release])

◎ MTF_API.dll: This is the library itself.(This file be included in system32 folder)

2.2 Linux application build

Linux version MTF_API library is necessary for linux application developments.

◎ MTF_API.h: This file contains the declarations of all available MTF_API functions. This must be included in the application source code.
(mtf_linux_1.0-API\include, arm version: mtf_arm_1.0-API\include)

◎ libMTF_API.so: This is the library.
(mtf_linux_1.0-API\lib, arm version: mtf_arm_1.0-API\lib)

2.3 Android application build

Android version MTF_API library is necessary for Android application developments.

◎ MTF_API.java: This file contains the declarations of all available MTF_API functions. This must be included in the application source code.
(Package path : kr.co.cubeeye.mtflib)

◎ mtfDeviceinfo.java:Device information file. Used in mtfGetDeviceList function.
(Package path : kr.co.cubeeye.mtflib)

◎ Mtflib.jar: This is the library. Copy Path [Project/app/libs] – Android Studio Project based)

3. API function description & examples

3.1 Function List

(1) mtfGetDeviceList(mtfDeviceInfo *pDevInfo, int *nDevCount);
(2) mtfDeviceOpen(mtfHandle hnd, unsigned char nRGBCam_No = 0);
(3) mtfDeviceClose(mtfHandle hnd);
(4) mtfDeviceIsOpen(mtfHandle hnd);
(5) mtfSetIntegrationTime(mtfHandle hnd, unsigned short nIntegrationTime);
(6) mtfGetIntegrationTime(mtfHandle hnd);
(7) mtfSetModuleFrequency(mtfHandle hnd, unsigned short nModuleFrequency);
(8) mtfGetModuleFrequency(mtfHandle hnd);
(9) mtfSetOffset(mtfHandle hnd, unsigned short nOffset);
(10) mtfGetOffset(mtfHandle hnd);
(11) mtfSetCheckThreshold(mtfHandle hnd, unsigned short nThreshold);
(12) mtfGetCheckThreshold(mtfHandle hnd);
(13) mtfReadBufInit(mtfHandle hnd);
(14) mtfGrabStart(mtfHandle hnd);
(15) mtfGrabStop(mtfHandle hnd);
(16) mtfReadFromDevice(mtfHandle hnd, unsigned short** wRecvData);
(17) mtfReadFromDevice_RGB888(mtfHandle hnd, char* nRGBData);
(18) mtfSetDepthRange(mtfHandle hnd, int nMinDepth, int nMaxDepth);
(19) mtfGetDepthRange(mtfHandle hnd, int *nMinDepth, int *nMaxDepth);
(20) mtfSetFlipHorizontal(mtfHandle hnd, bool bEnable);
(21) mtfGetFlipHorizontal(mtfHandle hnd);
(22) mtfSetFlipVertical(mtfHandle hnd, bool bEnable);
(23) mtfGetFlipVertical(mtfHandle hnd);
(24) mtfGetTemperature(mtfHandle hnd);
(25) mtfSetFilltering(mtfHandle hnd, mtfFilter nFilterType);
(26) mtfGetFiltering(mtfHandle hnd);
(27) mtfSetMultiSyncMode(mtfHandle hnd, bool bEnable);
(28) mtfGetMultiSyncMode(mtfHandle hnd);
(29) mtfSetFpsDelay(mtfHandle hnd, int nDelay);
(30) mtfGetFpsDelay (mtfHandle hnd);
(31) mtfSetUndistortion(mtfHandle hnd, bool bEnable, float fK1 = -0.23f, float fFx = 300.0f, float fFy = 300.0f, float fCx = 160.0f, float fCy = 120.0f);

```
//Device Information structure
typedef struct
{
        mtfHandle mtfHnd;               //Device Handle
        char szVendor[256];             //Vendor Name
        char szName[256];               //Name
        char szSerialNum[256];          //Sereial Number
        unsigned short nVendorId;       //Vendor ID
        unsigned short nProductId;      //Product ID
        unsigned short nDeviceType;     //Device Type(0:Only Depth, 1:Depth+RGB)
        long nWidth;                    //Image Width
        long nHeight;                   //Image Height
} mtfDeviceInfo;
```

3.2 Function Description

(1) mtfGetDeviceList

Get the information & number of the connected cameras.

Each camera is assigned corresponding handle number between 0 and 9.

Syntax

```
int mtfGetDeviceList(mtfDeviceInfo *pDevInfo, int *nDevCount);
```

Parameter

pDevInfo – Device information(pDevInfo[0] ~ pDevInfo[9])

nDevCount – Number of connected devices

Return Value

ERROR_NO(true) or Error number.

Example

```
mtfDeviceInfo m_pDevInfo[MAX_DEVICE];//MAX_DEVICE=10
int m_nDeviceNum;//Connected Camera Number
int GetDeviceList()
{
        mtfGetDeviceList(m_pDevInfo, &m_nDeviceNum)
}
//m_pDevInfo[0] => device 0 information. (mtfHnd: handle…)
```

(2) mtfDeviceOpen

Connect to ToF Camera. If connection fails, check USB cable connection and power supply status.

Syntax

```
int mtfDeviceOpen(mtfHandle hnd, unsigned char nRGBCam_No = 0);
```

Parameter

nRGBCam_No – RGB camera number. (Number can be changed depending on connecting-port. Default = 0. If there is any other devices such as webcam, assign a correct device number.)

Return Value

ERROR_NO(true) or Error number.

Example

```
int DeviceOpen()
{
        mtfHandle hnd = pDevInfo[0] .mtfHnd; //get device 0 handle
        unsigned char nRGBCamNo = 0;

        if(mtfDeviceOpen(hnd, nRGBCamNo) == TRUE)
                return true;
        else
                return false;
}
```

(3) mtfDeviceClose

Disconnect and close the device.

Syntax

```
void mtfDeviceClose(mtfHandle hnd);
```

Return Value

Void

Example

```
void DeviceClose(){
        mtfDeviceClose(hnd);
}
```

(4) mtfDeviceIsOpen

Check the connection status of device.

Syntax

```
int mtfDeviceIsOpen(mtfHandle hnd);
```

Return Value

true (Open) or false (Close).

Example

```
int DeviceIsOpen(){
        return mtfDeviceIsOpen(hnd);
}
```

(5) mtfSetIntegrationTime

Set the integration time of the camera. (3000, 4000, 5000, Unit: us)

Syntax

```
int mtfSetIntegrationTime(mtfHandle hnd, unsigned short nIntegrationTime)
```

Parameter

nIntegrationTime – Value of Integration time.(3000/4000/ 5000)

Return Value

ERROR_NO(true) or Error number.

Example

```
//device 0 set the integration 4000ms
mtfSetIntegrationTime(hnd, 4000);
```

(6) mtfGetIntegrationTime

Get the integration time of the camera.

Syntax

```
int mtfGetIntegrationTime(mtfHandle hnd);
```

Return Value

Value of the integration time.

Example

```
int nIntegrationTime = mtfGetIntegrationTime(hnd);
```

(7) mtfSetModuleFrequency

Set the Modulation Frequency of the camera.

Syntax

```
int mtfSetModulationFrequency(mtfHandle hnd, unsigned short
nModuleFrequency)
```

Parameter

nModuleFrequency – Value of Modulation Frequency.
(long range = 10Mhz, middle range = 20Mhz/30Mhz Fixed)

Return Value

Return Value – ERROR_NO(true) or Error number.

Example

```
//device 0 set the integration 10Mhz
mtf SetModulationFrequency(hnd, 10);
```

(8) mtfGetModuleFrequency

Get the Modulation Frequency of the camera.

Syntax

```
int mtfGetModuleFrequency(mtfHandle hnd);
```

Return Value

Value of the Modulation Frequency.

Example

```
int nModuleFrequency = mtfGetModuleFrequency(hnd);
```

(9) mtfSetOffset

Set the distance offset of the camera.

Syntax

```
int mtfSetOffsset(mtfHandle hnd, unsigned short nOffset)
```

Parameter

nOffset – Value of distance offset.(unit: mm)

Return Value

Return Value – ERROR_NO(true) or Error number.

Example

```
//device 0 set the offset 1960mm
mtfSetOffet(hnd, 1960);
```

(10) mtfGetOffset

Get the distance offset of the camera.

Syntax

```
int mtfGetOffset(mtfHandle hnd);
```

Return Value

Value of the distance offset.

Example

```
int nOffset = mtfGetOffset(hnd);
```

(11) mtfSetCheckThreshold

Set the threshold for depth signal strength check.

(If the amplitude value of IR image's each pixel is smaller than this check threshold, this pixel is regarded as invalid pixel. Invalid pixel will be shown as "0". This 'Function' can be used to reduce background noise.)

Syntax

```
int mtfSetCheckThreshold(mtfHandle hnd, unsigned short nThreshold)
```

Parameter

nThreshold – Value of Signal check threshold.

Return Value

Return Value – ERROR_NO(true) or Error number.

Example

```
//device 0 set the Signal Check threshold 20
mtfSetCheckThreshold(hnd, 20);
```

(12) mtfGetCheckThreshold

Get the threshold for depth signal strength check.

Syntax

```
int mtfGetCheckThreshold(mtfHandle hnd);
```

Return Value

Value of the signal strength check threshold.

Example

```
int nCheckThreshold = mtfGetCheckThreshold(hnd);
```

(13) mtfReadBufInit

Initialize the read buffer. (Call this function once, before starting data reading thread.)

Syntax

```
int mtfReadBufInit(mtfHandle hnd);
```

Return Value

Return Value – ERROR_NO(true) or Error number.

Example

```
mtfReadBufInit(hnd);
```

(14) mtfGrabStart

Initiate the image grab. (Image acquisition is started, if function is called once after data reading thread is initiated)

Syntax

```
int mtfGrabStart(mtfHandle hnd);
```

Return Value

true (Success) or false (Fail).

Example

mtfGrabStart(hnd);

(15) mtfGrabStop

Stop the image grab.

(Call this function once, before ending data reading thread)

Syntax

int mtfGrabStop(mtfHandle hnd);

Return Value

Return Value – ERROR_NO(true) or Error number.

Example

```
//hnd =device 0 handle

mtfGrabStop(hnd);
```

(16) mtfReadFromDevice

Get the camera Depth/IR data from the current frame.

([0]:IR, [1]:depth / 320x240, 16bit)

Syntax

void mtfReadFromDevice(mtfHandle hnd, unsigned short** wRecvData);

Parameter

```
//wRecvData – Read Data Buffer.
wRecvData[0][320*240] - Amplitude Data
wRecvData[1][320*240] - Depth Data
```

Return Value – void

(17) mtf ReadFromDevice _RGB888

Get the camera RGB data from the current frame.(640x480, RGB888)

Syntax

void mtfReadFromDevice_RGB888(mtfHandle hnd, unsigned char *nRGBData);

Parameter

nRGBData - RGB Data Buffer

Return Value - void

(18) mtfSetDepthRange

Set the Min & Max depth range. (Depth data out of the pre-set range is set 0)
Max depth limit depends on the camera model.

Syntax

void mtfSetDepthRange(mtfHandle hnd, int nMinDepth, int nMaxDepth)

Parameter

nMinDepth – Mini depth range.

nMaxDepth – Max depth range

Return Value – ERROR_NO(true) or Error number.

Example

```
//device 0 set the depth range 0~5000mm
mtfSetDepthRange(hnd, 0, 5000); //unit=mm
```

(19) mtfGetDepthRange

Get the Min & Max depth range.

Syntax

void mtfSetDepthRange(mtfHandle hnd, int *nMinDepth, *int nMaxDepth)

Parameter

nMinDepth – Mini depth range.

nMaxDepth – Max depth range

Return Value – ERROR_NO(true) or Error number.

Example

```
int nMinDepth, nMaxDepth;
mtfSetDepthRange(hnd, &nMinDepth, & nMaxDepth);
```

(20) mtfSetFlipHorizontal

Set or cancel image horizontal flip.

Syntax

```
int mtfSetFlipHorizontal(mtfHandle hnd, bool bEnable);
```

Parameter

bEnable – Image Flip Horizontal status

Return Value

true (Success) or false (Fail).

Example

```
bool bEnalbe = true;
mtfSetFlipHorizontal(hnd, bEnable);
```

(21) mtfGetFlipHorizontal

Get image horizontal flip status.

Syntax

```
int mtfGetFlipHorizontal(mtfHandle hnd);
```

Return Value

true(Enable) or false(Disable)

Example

```
int nStatus = mtfGetFlipHorizontal(hnd);
```

(22) mtfSetFlipVertical

Set or cancel image vertical flip.

Syntax

```
int mtfSetFlipVertical(mtfHandle hnd, bool bEnable);
```

Parameter

bEnable – Image Flip Vertical status

Return Value

true (Success) or false (Fail).

Example

```
bool bEnalbe = true;
mtfSetFlipHorizontal (hnd, bEnable);
```

(23) mtfGetFlipVertical

Get image vertical flip status.

Syntax

```
int mtfGetFlipVertical(mtfHandle hnd);
```

Return Value

true(Enable) or false(Disable)

Example

```
int nStatus = mtfGetFlipVertical(hnd);
```

(24) mtfGetTemperature

Get the camera sensor temperature

Syntax

```
double mtfGetTemperature(mtfHandle hnd);
```

Return Value

Sensor temperature

Example

```
double dTemperature = mtfGetTemperature(hnd);
```

(25) mtfSetFiltering

Set the hardware filtering type. (Using this function does not affect frame rate)

Syntax

```
int mtfSetFlipVertical(mtfHandle hnd, bool bEnable);
```

Parameter

bEnable – Image Flip Vertical status

Return Value

true (Success) or false (Fail).

Example

```
mtfSetFlipHorizontal (hnd, true);
```

(26) mtfGetFiltering

Get the current hardware filtering type.

Syntax

```
int mtfGetFlipVertical(mtfHandle hnd);
```

Return Value

true(Enable) or false(Disable)

Example

```
int nStatus = mtfGetFlipVertical(hnd);
```

(27) mtfSetMultiSyncMode

Set the multi camera synchronize mode. (Frame rate is decreased.)
If you want to using multi cameras without non light interference,
You have to enable through this function.

Syntax

```
int mtfSetMultiSyncMode(mtfHandle hnd, bool bEnable);
```

Parameter

bEnable – Multi-Camera Synchronize Mode status

Return Value

true (Success) or false (Fail).

Example

```
mtfSetMultiSyncMode(hnd, true);
```

(28) mtfGetMultiSyncMode

Get the Multi Camera Synchronize mode status.

Syntax

```
int mtfGetMultiSyncMode(mtfHandle hnd);
```

Return Value

true(Enable) or false(Disable)

Example

```
int nStatus = mtfGetMultiSyncMode(hnd);
```

(29) mtfSetFpsDelay

Set the Frame rate delay time. (If set the non-zero value, the frame rate is decreased)

Syntax

```
int mtfSetFpsDelay(mtfHandle hnd, int nDelay);
```

Parameter

nDelay – Delay time value.(0~30)

Return Value

true (Success) or false (Fail).

Example

```
mtfSetFpsDelay(hnd, 0);
```

(30) mtfGetFpsDelay

Get the Frame rate delay time.

Syntax

```
int mtfGetFpsDelay(mtfHandle hnd);
```

Return Value

Value of the Frame rate delay time.

Example

```
int nDelayTime = mtfGetFpsDelay (hnd);
```

(31) mtfSetUndistortion

Set the 'lens image distortion remove' functionality.

Syntax

int mtfSetUndistortion(mtfHandle hnd, bool bEnable, float fK1 = -0.23f, float fFx = 300.0f, float fFy = 300.0f, float fCx = 160.0f, float fCy = 120.0f)

Return Value

true(Enable) or false(Disable)

Parameter

bEnable – undistortion status
fK1 - radial distortion value
fFx - focal length x value
fFy - focal length y value
fCx - principal point x value
fCy - principal point y value

Example

```
mtfSetUndistortion(hnd, true);
```

<Caution>

All set functions can only be applied during reading thread run.
(reading thread continuously call 'ReadFromDevice' function)

*Please refer to the sample code in SDK folder

3.3 Image Data Acquisition Sample

*Sample - Data Reading Process Code

```
//** start data acquired process **//
//1.Read buffer init
mtfReadBufInit(hnd);

//2.create for data read thread
bool bstart = true;
int nStatus;
pthread_t tidRead;
nRet= pthread_create(&tidRead, NULL, ThreadFunc, NULL);

//3.Grab start
mtfGrabStart(hnd);//*after run reading thread


//** Data Read Thread function **//
static void *ThreadFunc(void *arg)
{
        unsigned short *m_pCameraBuf[2];//320 x 240 16bit
        int nImageBuffSize = sizeof((unsigned short) * 320 * 240);

        //Amplitude data buffer
        m_pCameraBuf[0] = (unsigned short *)malloc(nImageBuffSize);
        //Depth data buffer
        m_pCameraBuf[1] = (unsigned short *)malloc(nImageBuffSize);
```

```
        unsigned short *m_pRGBBuf;//640 x 480 x 8bit
        int nRGBBuffSize = sizeof((unsigned char) * 640 * 480 * 3);

        //R data buffer
        m_pRGBBuf = (char short *)malloc(nRGBBuffSize);

        while(bstart)
        {
                //reading function
                mtfReadFromDevice(hnd, m_pCameraBuf);
                mtfReadFromDevice_RGB888(hnd, m_pRGBBuf);
                 usleep(1);
        }

        free(m_pCameraBuf[0]);
        free(m_pCameraBuf[1]);
        free(pRGBBuf);
}


//** stop data acquired process **//
//1.Grab stop
mtfGrabStop(hnd);

//2.stop for data read thread
bstart = false;
pthread_join(tidRead, (void**)nStatus);
```

3.4 Android Studio Project Guide(SimpleRead)

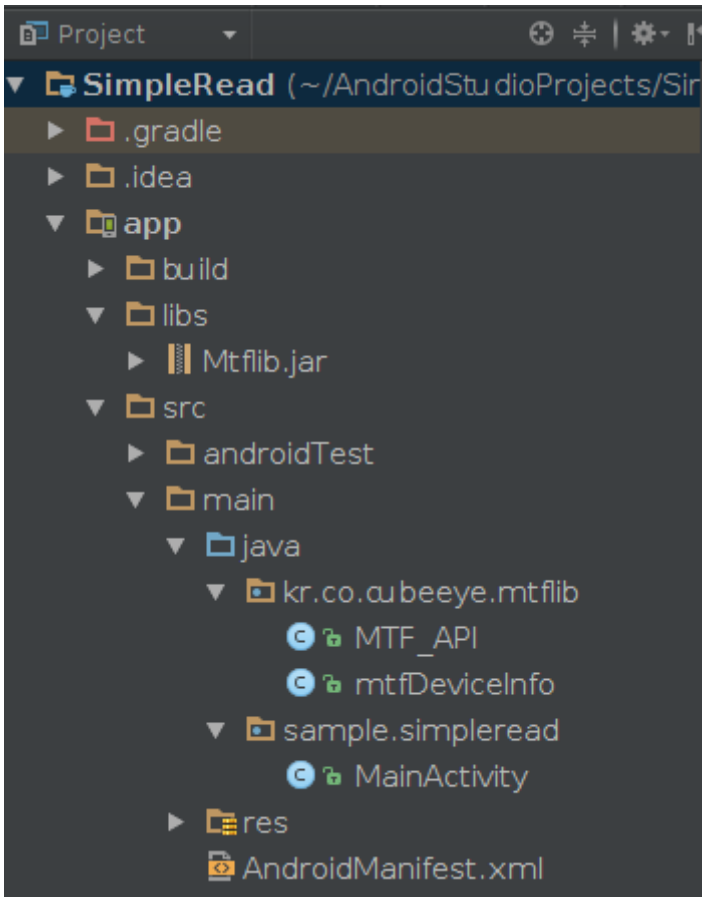
1. Create Android Project

2. Copy to jar file(Project/app/libs)

3. Create Package(kr.co.cubeeye.mtflib)

4. Copy the file to the generated package.

   (MTF_API.java, mtfDeviceinfo.java)



5. Source Coding(MainActivity)

- Library setting.
- Import mtfDeviceInfo.

```
public class MainActivity extends AppCompatActivity {
    static{
        System.loadLibrary("MTF_Android");
    }
    kr.co.cubeeye.mtflib.mtfDeviceInfo m_stDevInfo = new kr.co.cubeeye.mtflib.mtfDeviceInfo();
```