大恒 水星 linux 水星相机结合 opencv 控制台例程

1. 开发环境

Ubuntu16.04
Pc 64
Opencv 3.2
Opencv_contrib3.2
Mer-502-79U3M
Mer-131-210U3C

2. 程序功能

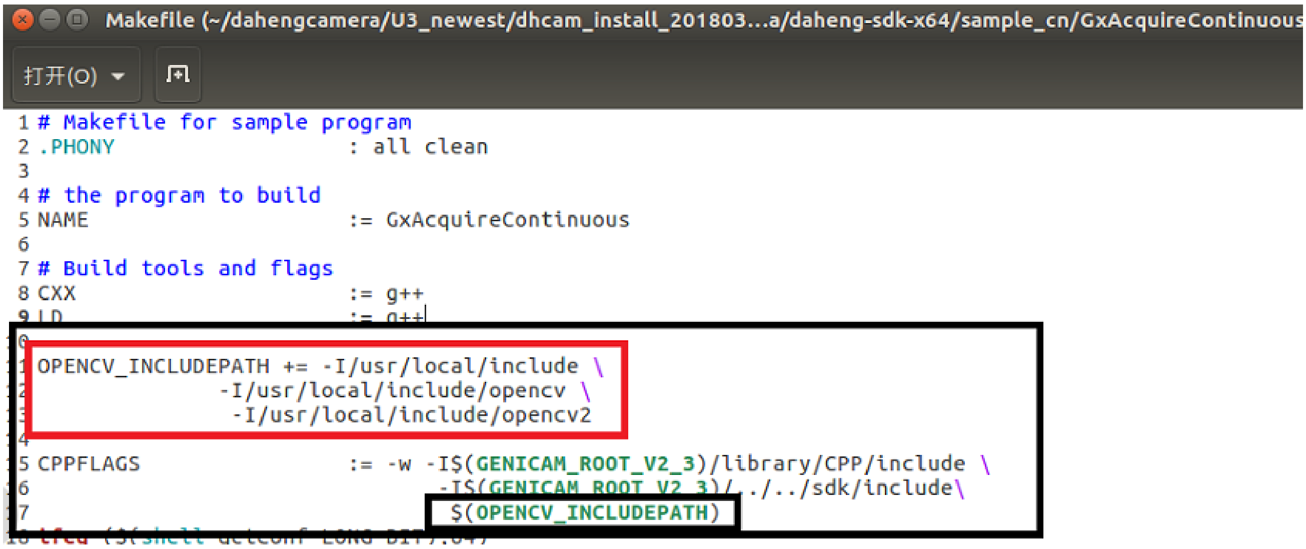
Linux 水星 sdk 采集图像 之后用 opencv 显示

3. 环境配置

3.1. 配置 makefile

3.1.1. 添加 opencv 头文件

使用 sudo 权限打开文件 Makefile,操作代码 sudo gedit Makefile



Makefile (~/dahengcamera/U3_newest/dhcam_install_201803...a/daheng-sdk-x64/sample_cn/GxAcquireContinuous
打开(O)

```
# Makefile for sample program
.PHONY                          : all clean

# the program to build
NAME                            := GxAcquireContinuous

# Build tools and flags
CXX                             := g++
LD                              := g++

OPENCV_INCLUDEPATH += -I/usr/local/include \
                   -I/usr/local/include/opencv \
                    -I/usr/local/include/opencv2

CPPFLAGS                        := -w -I$(GENICAM_ROOT_V2_3)/library/CPP/include \
                                   -I$(GENICAM_ROOT_V2_3)/../../sdk/include\
                                     $(OPENCV_INCLUDEPATH)
```

a:#添加 opencv 的头文件，二处需要修改

I:新增变量 OPENCV_INCLUDEPATH ， 路径如下：

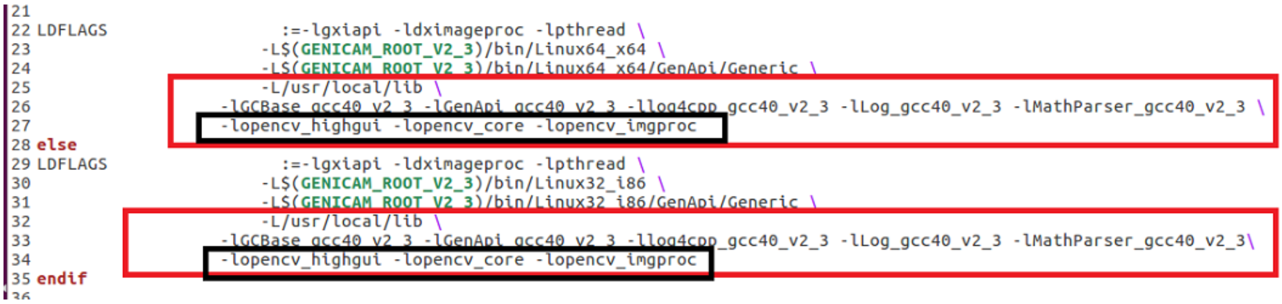
```
OPENCV_INCLUDEPATH += -I/usr/local/include \
                   -I/usr/local/include/opencv \
                    -I/usr/local/include/opencv2
```

II：在变量 CPPFLAGS 最后添加 opencv 头文件

```
CPPFLAGS        := -w -I$(GENICAM_ROOT_V2_3)/library/CPP/include \
                      -I$(GENICAM_ROOT_V2_3)/../../sdk/include\
```

$(OPENCV_INCLUDEPATH)

3.1.2. 添加 opencv lib



```
LDFLAGS                     :=-lgxiapi -ldximageproc -lpthread \
                          -L$(GENICAM_ROOT_V2_3)/bin/Linux64_x64 \
                          -L$(GENICAM_ROOT_V2_3)/bin/Linux64_x64/GenApi/Generic \
                          -L/usr/local/lib \
                      -lGCBase_gcc40_v2_3 -lGenApi_gcc40_v2_3 -llog4cpp_gcc40_v2_3 -lLog_gcc40_v2_3 -lMathParser_gcc40_v2_3 \
                      -lopencv_highgui -lopencv_core -lopencv_imgproc
else
LDFLAGS                     :=-lgxiapi -ldximageproc -lpthread \
                          -L$(GENICAM_ROOT_V2_3)/bin/Linux32_i86 \
                          -L$(GENICAM_ROOT_V2_3)/bin/Linux32_i86/GenApi/Generic \
                          -L/usr/local/lib \
                      -lGCBase_gcc40_v2_3 -lGenApi_gcc40_v2_3 -llog4cpp_gcc40_v2_3 -lLog_gcc40_v2_3 -lMathParser_gcc40_v2_3\
                      -lopencv_highgui -lopencv_core -lopencv_imgproc
endif
```

在 makefile 的变量 LDFLGAS 中添加 opencv lib 环境。具体位置参考上图。

添加内容参考如下代码(标注红字部分)

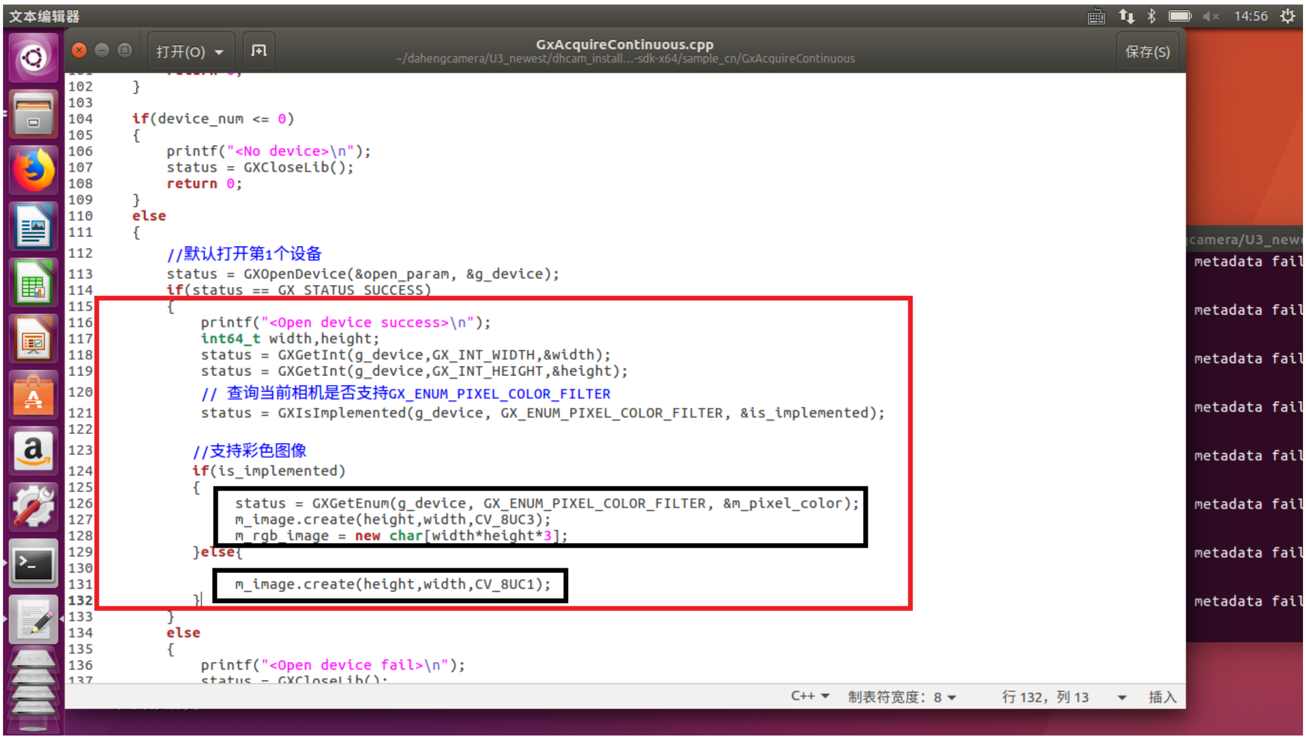
```
ifeq ($(shell getconf LONG_BIT),64)
LDFLAGS                 :=-lgxiapi -ldximageproc -lpthread \
                            -L$(GENICAM_ROOT_V2_3)/bin/Linux64_x64 \
                            -L$(GENICAM_ROOT_V2_3)/bin/Linux64_x64/GenApi/Generic \
                   -L/usr/local/lib \
                        -lGCBase_gcc40_v2_3      -lGenApi_gcc40_v2_3      -llog4cpp_gcc40_v2_3
-lLog_gcc40_v2_3 -lMathParser_gcc40_v2_3 \
              -lopencv_highgui -lopencv_core -lopencv_imgproc
else
LDFLAGS                     :=-lgxiapi -ldximageproc -lpthread \
                            -L$(GENICAM_ROOT_V2_3)/bin/Linux32_i86 \
                            -L$(GENICAM_ROOT_V2_3)/bin/Linux32_i86/GenApi/Generic \
                   -L/usr/local/lib \
                        -lGCBase_gcc40_v2_3      -lGenApi_gcc40_v2_3      -llog4cpp_gcc40_v2_3
-lLog_gcc40_v2_3 -lMathParser_gcc40_v2_3\
              -lopencv_highgui -lopencv_core -lopencv_imgproc
Endif
```

3.1.3. OpenCV 代码实现

1:添加头文件，及命名空间

```
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include "DxImageProc.h"
 using namespace cv;
Mat m_image;
bool        is_implemented = false;
int64_t                  m_pixel_color = 0;          ///< Bayer格?式？
char*m_rgb_image = NULL;
```

2:初始化 mat



```
        return 0;
    }

    if(device_num <= 0)
    {
        printf("<No device>\n");
        status = GXCloseLib();
        return 0;
    }
    else
    {
        //默认打开第1个设备
        status = GXOpenDevice(&open_param, &g_device);
        if(status == GX_STATUS_SUCCESS)
        {
            printf("<Open device success>\n");
            int64_t width,height;
            status = GXGetInt(g_device,GX_INT_WIDTH,&width);
            status = GXGetInt(g_device,GX_INT_HEIGHT,&height);
            // 查询当前相机是否支持GX_ENUM_PIXEL_COLOR_FILTER
            status = GXIsImplemented(g_device, GX_ENUM_PIXEL_COLOR_FILTER, &is_implemented);

           //支持彩色图像
           if(is_implemented)
           {
              status = GXGetEnum(g_device, GX_ENUM_PIXEL_COLOR_FILTER, &m_pixel_color);
              m_image.create(height,width,CV_8UC3);
              m_rgb_image = new char[width*height*3];
           }else{

              m_image.create(height,width,CV_8UC1);
           }
        }
        else
        {
            printf("<Open device fail>\n");
            status = GXCloseLib();
```

针对相机，对 Mat 进行初始化

```
#下面是初始化 Mat 的相关代码
if(status == GX_STATUS_SUCCESS)
    {
            printf("<Open device success>\n");
            int64_t width,height;
            status = GXGetInt(g_device,GX_INT_WIDTH,&width);
            status = GXGetInt(g_device,GX_INT_HEIGHT,&height);
            // 查询当前相机是否支持 GX_ENUM_PIXEL_COLOR_FILTER
            status=GXIsImplemented(g_device,GX_ENUM_PIXEL_COLOR_FILTER, &is_implemented);
           //支持彩色图像
           if(is_implemented)
           {
            status= GXGetEnum(g_device, GX_ENUM_PIXEL_COLOR_FILTER, &m_pixel_color);
            m_image.create(height,width,CV_8UC3);//彩色相机
            m_rgb_image = new char[width*height*3];
       }else{

            m_image.create(height,width,CV_8UC1);//黑白相机
       }
    }
```

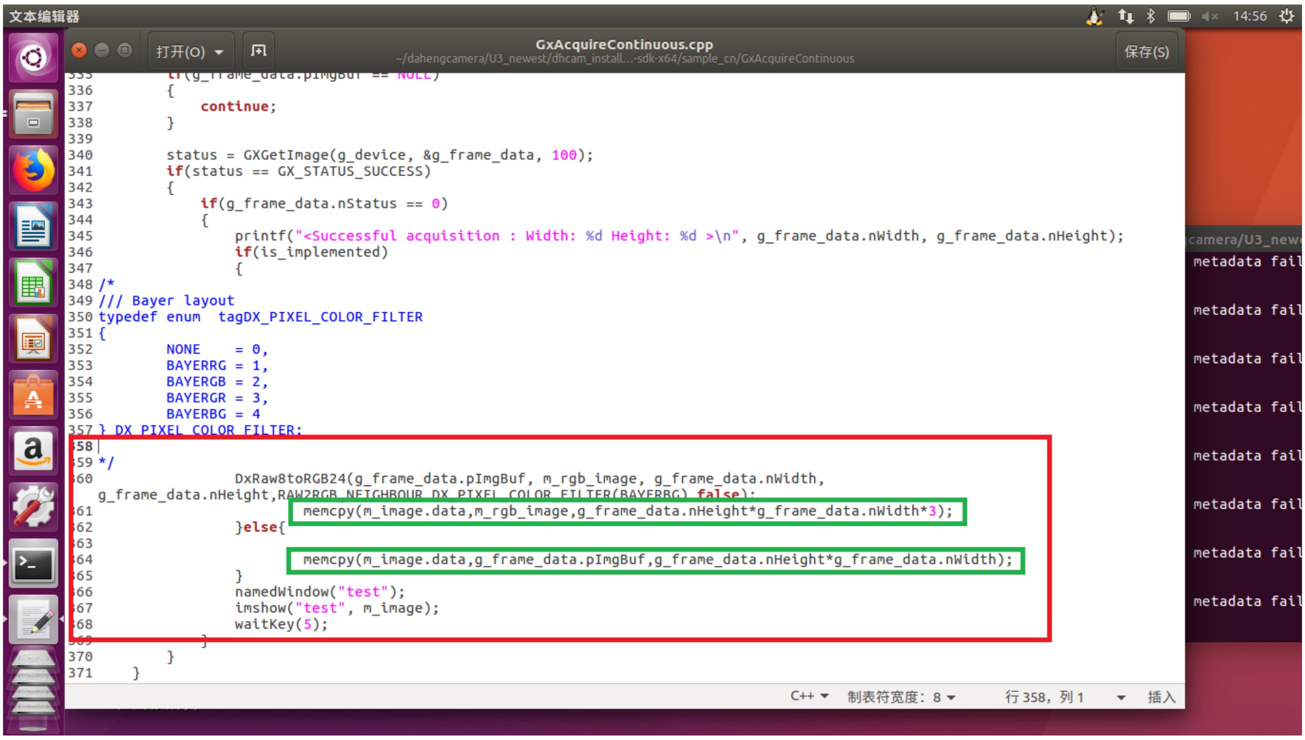
3.图像拷贝

根据相机类型 Mono 或者彩色，进行相应的拷贝动作。

A:针对黑白相机

memcpy(m_image.data,g_frame_data.pImgBuf,g_frame_data.nHeight*g_frame_data.nWidth);

B：针对彩色相机



```
        if(g_frame_data.pImgBuf == NULL)
        {
            continue;
        }

        status = GXGetImage(g_device, &g_frame_data, 100);
        if(status == GX_STATUS_SUCCESS)
        {
            if(g_frame_data.nStatus == 0)
            {
                printf("<Successful acquisition : Width: %d Height: %d >\n", g_frame_data.nWidth, g_frame_data.nHeight);
                if(is_implemented)
                {
/*
/// Bayer layout
typedef enum  tagDX_PIXEL_COLOR_FILTER
{
        NONE    = 0,
        BAYERRG = 1,
        BAYERGB = 2,
        BAYERGR = 3,
        BAYERBG = 4
} DX_PIXEL_COLOR_FILTER;

*/
                DxRaw8toRGB24(g_frame_data.pImgBuf, m_rgb_image, g_frame_data.nWidth,
g_frame_data.nHeight,RAW2RGB_NEIGHBOUR,DX_PIXEL_COLOR_FILTER(BAYERBG),false);
                    memcpy(m_image.data,m_rgb_image,g_frame_data.nHeight*g_frame_data.nWidth*3);
                }else{

                    memcpy(m_image.data,g_frame_data.pImgBuf,g_frame_data.nHeight*g_frame_data.nWidth);
                }
                namedWindow("test");
                imshow("test", m_image);
                waitKey(5);
            }
        }
    }
```

注：

如果是彩色相机，在数据拷贝时，需要修改 bayer 转换的一个参数 DX_PIXEL_COLOR_FILTER，

该参数有 4 种格式:

```
typedef enum   tagDX_PIXEL_COLOR_FILTER
{
      NONE      = 0,
      BAYERRG = 1,
      BAYERGB = 2,
      BAYERGR = 3,
      BAYERBG = 4
} DX_PIXEL_COLOR_FILTER;
```

针对不同的相机，该参数需要进行实际的测试，暂时无经验值.

DxRaw8toRGB24(g_frame_data.pImgBuf,m_rgb_image,g_frame_data.nWidth, g_frame_data.nHeight,RAW2RGB_NEIGHBOUR,DX_PIXEL_COLOR_FILTER(BAYERBG),false);

memcpy(m_image.data,m_rgb_image,g_frame_data.nHeight*g_frame_data.nWidth*3);

4：图像显示

当数据拷贝成功后，执行代码显示图像。

新建显示窗口：namedWindow(“test”);

显示代码：imshow(“test”,m_image);

waitKey(5   );

4. 版本历史

序号	修订版本号	所做改动	作者	发布日期
1	V1.0.0	1.初始发布	严立群	2018-06