**机器视觉第三次作业**

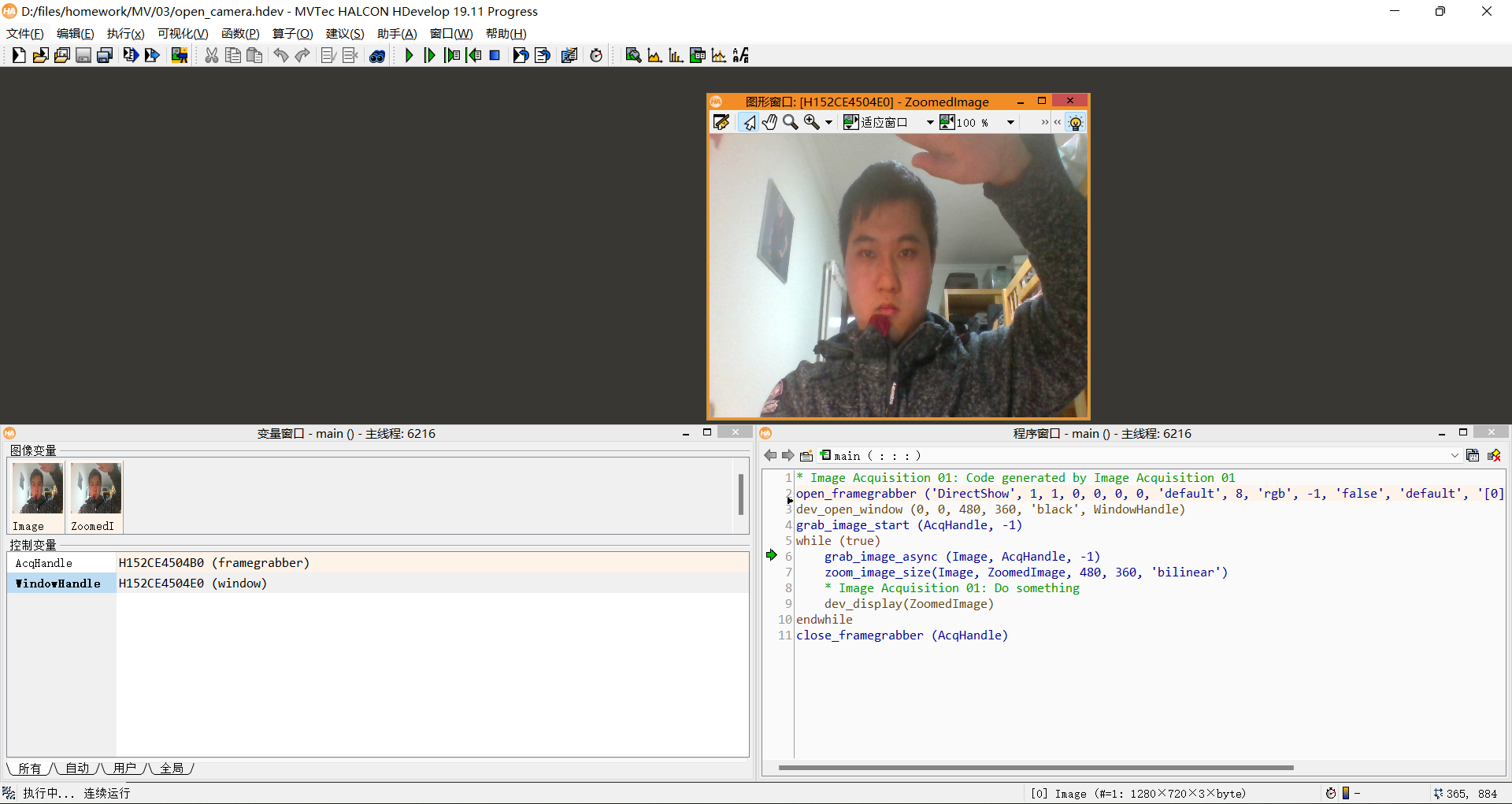
190320225 王福栋

**一、使用HALCON实现摄像头访问**

代码：

\* Image Acquisition 01: Code generated by Image Acquisition 01  
open\_framegrabber('DirectShow',1, 1, 0, 0, 0, 0, 'default', 8, 'rgb',\  
 -1, 'false', 'default', '[0] Integrated Camera', 0, -1, AcqHandle)  
dev\_open\_window (0, 0, 480, 360, 'black', WindowHandle)  
grab\_image\_start (AcqHandle, -1)  
while (true)  
    grab\_image\_async (Image, AcqHandle, -1)   
    \* Image Acquisition 01: Do something  
    \* 为了让C程序也能顺利显示出图像，添了个窗口创建和图像resize  
    zoom\_image\_size(Image, ZoomedImage, 480, 360, 'bilinear')  
    dev\_display(ZoomedImage)  
endwhile  
close\_framegrabber (AcqHandle)

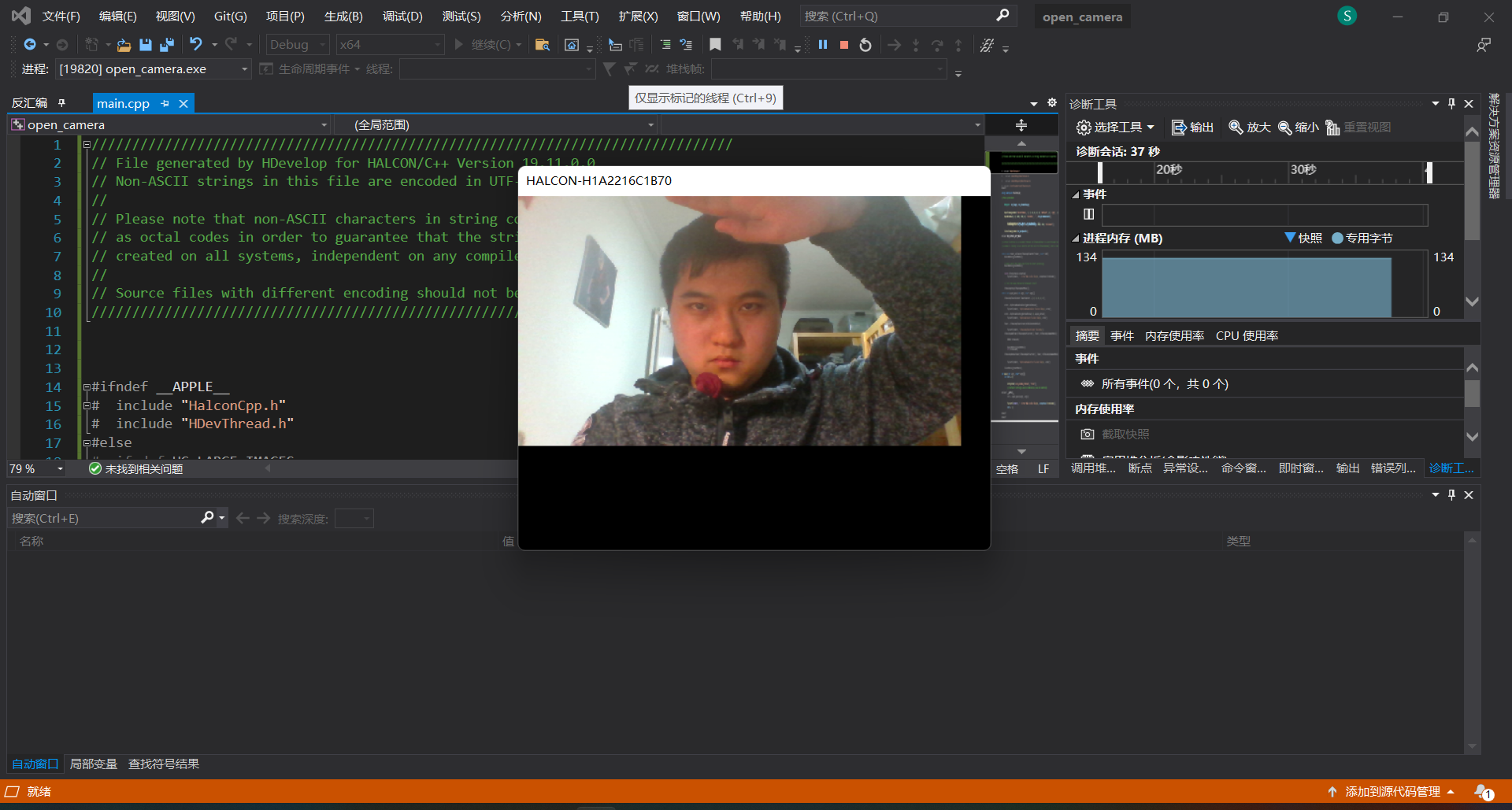
效果：



**二、导出C++程序并运行**

代码较长，见文档最下方。

代码运行效果：



（发现VS编译出来的图像缩放与HALCON内运行出来的图像缩放效果不太一样）

HALCON生成的C程序代码：

///////////////////////////////////////////////////////////////////////////////  
// File generated by HDevelop for HALCON/C++ Version 19.11.0.0  
// Non-ASCII strings in this file are encoded in UTF-8.  
//   
// Please note that non-ASCII characters in string constants are exported  
// as octal codes in order to guarantee that the strings are correctly  
// created on all systems, independent on any compiler settings.  
//   
// Source files with different encoding should not be mixed in one project.  
///////////////////////////////////////////////////////////////////////////////  
  
  
  
#ifndef \_\_APPLE\_\_  
#  include "HalconCpp.h"  
#  include "HDevThread.h"  
#else  
#  ifndef HC\_LARGE\_IMAGES  
#    include <HALCONCpp/HalconCpp.h>  
#    include <HALCONCpp/HDevThread.h>  
#    include <HALCON/HpThread.h>  
#  else  
#    include <HALCONCppxl/HalconCpp.h>  
#    include <HALCONCppxl/HDevThread.h>  
#    include <HALCONxl/HpThread.h>  
#  endif  
#  include <stdio.h>  
#  include <CoreFoundation/CFRunLoop.h>  
#endif  
  
  
  
using namespace HalconCpp;  
  
  
#ifndef NO\_EXPORT\_MAIN  
// Main procedure   
void action()  
{  
  
    // Local iconic variables  
    HObject  ho\_Image, ho\_ZoomedImage;  
  
    // Local control variables  
    HTuple  hv\_AcqHandle, hv\_WindowHandle;  
  
    //Image Acquisition 01: Code generated by Image Acquisition 01  
    OpenFramegrabber("DirectShow", 1, 1, 0, 0, 0, 0, "default", 8, "rgb", -1, "false",  
        "default", "[0] Integrated Camera", 0, -1, &hv\_AcqHandle);  
    SetWindowAttr("background\_color", "black");  
    OpenWindow(0, 0, 480, 360, 0, "visible", "", &hv\_WindowHandle);  
    HDevWindowStack::Push(hv\_WindowHandle);  
    GrabImageStart(hv\_AcqHandle, -1);  
    while (0 != 1)  
    {  
        GrabImageAsync(&ho\_Image, hv\_AcqHandle, -1);  
        ZoomImageSize(ho\_Image, &ho\_ZoomedImage, 480, 360, "bilinear");  
        //Image Acquisition 01: Do something  
        if (HDevWindowStack::IsOpen())  
            DispObj(ho\_ZoomedImage, HDevWindowStack::GetActive());  
    }  
    CloseFramegrabber(hv\_AcqHandle);  
}  
  
  
#ifndef NO\_EXPORT\_APP\_MAIN  
  
#ifdef \_\_APPLE\_\_  
// On OS X systems, we must have a CFRunLoop running on the main thread in  
// order for the HALCON graphics operators to work correctly, and run the  
// action function in a separate thread. A CFRunLoopTimer is used to make sure  
// the action function is not called before the CFRunLoop is running.  
// Note that starting with macOS 10.12, the run loop may be stopped when a  
// window is closed, so we need to put the call to CFRunLoopRun() into a loop  
// of its own.  
HTuple      gStartMutex;  
H\_pthread\_t gActionThread;  
HBOOL       gTerminate = FALSE;  
  
static void timer\_callback(CFRunLoopTimerRef timer, void\* info)  
{  
    UnlockMutex(gStartMutex);  
}  
  
static Herror apple\_action(void\*\* parameters)  
{  
    // Wait until the timer has fired to start processing.  
    LockMutex(gStartMutex);  
    UnlockMutex(gStartMutex);  
  
    try  
    {  
        action();  
    }  
    catch (HException& exception)  
    {  
        fprintf(stderr, "  Error #%u in %s: %s\n", exception.ErrorCode(),  
            (const char\*)exception.ProcName(),  
            (const char\*)exception.ErrorMessage());  
    }  
  
    // Tell the main thread to terminate itself.  
    LockMutex(gStartMutex);  
    gTerminate = TRUE;  
    UnlockMutex(gStartMutex);  
    CFRunLoopStop(CFRunLoopGetMain());  
    return H\_MSG\_OK;  
}  
  
static int apple\_main(int argc, char\* argv[])  
{  
    Herror                error;  
    CFRunLoopTimerRef     Timer;  
    CFRunLoopTimerContext TimerContext = { 0, 0, 0, 0, 0 };  
  
    CreateMutex("type", "sleep", &gStartMutex);  
    LockMutex(gStartMutex);  
  
    error = HpThreadHandleAlloc(&gActionThread);  
    if (H\_MSG\_OK != error)  
    {  
        fprintf(stderr, "HpThreadHandleAlloc failed: %d\n", error);  
        exit(1);  
    }  
  
    error = HpThreadCreate(gActionThread, 0, apple\_action);  
    if (H\_MSG\_OK != error)  
    {  
        fprintf(stderr, "HpThreadCreate failed: %d\n", error);  
        exit(1);  
    }  
  
    Timer = CFRunLoopTimerCreate(kCFAllocatorDefault,  
        CFAbsoluteTimeGetCurrent(), 0, 0, 0,  
        timer\_callback, &TimerContext);  
    if (!Timer)  
    {  
        fprintf(stderr, "CFRunLoopTimerCreate failed\n");  
        exit(1);  
    }  
    CFRunLoopAddTimer(CFRunLoopGetCurrent(), Timer, kCFRunLoopCommonModes);  
  
    for (;;)  
    {  
        HBOOL terminate;  
  
        CFRunLoopRun();  
  
        LockMutex(gStartMutex);  
        terminate = gTerminate;  
        UnlockMutex(gStartMutex);  
  
        if (terminate)  
            break;  
    }  
  
    CFRunLoopRemoveTimer(CFRunLoopGetCurrent(), Timer, kCFRunLoopCommonModes);  
    CFRelease(Timer);  
  
    error = HpThreadHandleFree(gActionThread);  
    if (H\_MSG\_OK != error)  
    {  
        fprintf(stderr, "HpThreadHandleFree failed: %d\n", error);  
        exit(1);  
    }  
  
    ClearMutex(gStartMutex);  
    return 0;  
}  
#endif  
  
int main(int argc, char\* argv[])  
{  
    int ret = 0;  
  
    try  
    {  
#if defined(\_WIN32)  
        SetSystem("use\_window\_thread", "true");  
#endif  
  
        // Default settings used in HDevelop (can be omitted)  
        SetSystem("width", 512);  
        SetSystem("height", 512);  
  
#ifndef \_\_APPLE\_\_  
        action();  
#else  
        ret = apple\_main(argc, argv);  
#endif  
    }  
    catch (HException& exception)  
    {  
        fprintf(stderr, "  Error #%u in %s: %s\n", exception.ErrorCode(),  
            (const char\*)exception.ProcName(),  
            (const char\*)exception.ErrorMessage());  
        ret = 1;  
    }  
    return ret;  
}  
  
#endif  
  
  
#endif