SZ170320207

刘健恒

机器视觉

作业三

代码：

///////////////////////////////////////////////////////////////////////////////

// File generated by HDevelop for HALCON/C++ Version 18.11.0.1

// Non-ASCII strings in this file are encoded in local-8-bit encoding (cp936).

// Ensure that the interface encoding is set to locale encoding by calling

// SetHcppInterfaceStringEncodingIsUtf8(false) at the beginning of the program.

//

// 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"

# if defined(\_\_linux\_\_) && (defined(\_\_i386\_\_) || defined(\_\_x86\_64\_\_)) \

&& !defined(NO\_EXPORT\_APP\_MAIN)

# include <X11/Xlib.h>

# endif

#else

# ifndef HC\_LARGE\_IMAGES

# include <HALCONCpp/HalconCpp.h>

# include <HALCONCpp/HDevThread.h>

# else

# include <HALCONCppxl/HalconCpp.h>

# include <HALCONCppxl/HDevThread.h>

# endif

# include <stdio.h>

# include <HALCON/HpThread.h>

# include <CoreFoundation/CFRunLoop.h>

#endif

using namespace HalconCpp;

#ifndef NO\_EXPORT\_MAIN

// Main procedure

void action()

{

// Local iconic variables

HObject ho\_Image;

// Local control variables

HTuple hv\_AcqHandle;

//Image Acquisition 01: Code generated by Image Acquisition 01

//Image Acquisition 01: Attention: The initialization may fail in case parameters need to

//Image Acquisition 01: be set in a specific order (e.g., image resolution vs. offset).

OpenFramegrabber("DirectShow", 1, 1, 0, 0, 0, 0, "default", 8, "rgb", -1, "false",

"default", "[0] ", 0, -1, &hv\_AcqHandle);

SetFramegrabberParam(hv\_AcqHandle, "brightness", 116);

SetFramegrabberParam(hv\_AcqHandle, "contrast", 81);

SetFramegrabberParam(hv\_AcqHandle, "hue", 8);

SetFramegrabberParam(hv\_AcqHandle, "saturation", 50);

SetFramegrabberParam(hv\_AcqHandle, "sharpness", 4);

SetFramegrabberParam(hv\_AcqHandle, "gamma", 118);

SetFramegrabberParam(hv\_AcqHandle, "video\_gain", 4);

GrabImageStart(hv\_AcqHandle, -1);

HWindow w(0, 0, 1024, 1024);

while (0 != 1)

{

GrabImageAsync(&ho\_Image, hv\_AcqHandle, -1);

ho\_Image.DispObj(w);

//Image Acquisition 01: Do something

}

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");

#elif defined(\_\_linux\_\_) && (defined(\_\_i386\_\_) || defined(\_\_x86\_64\_\_))

XInitThreads();

#endif

// file was stored with local-8-bit encoding

// -> set the interface encoding accordingly

SetHcppInterfaceStringEncodingIsUtf8(false);

// 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;

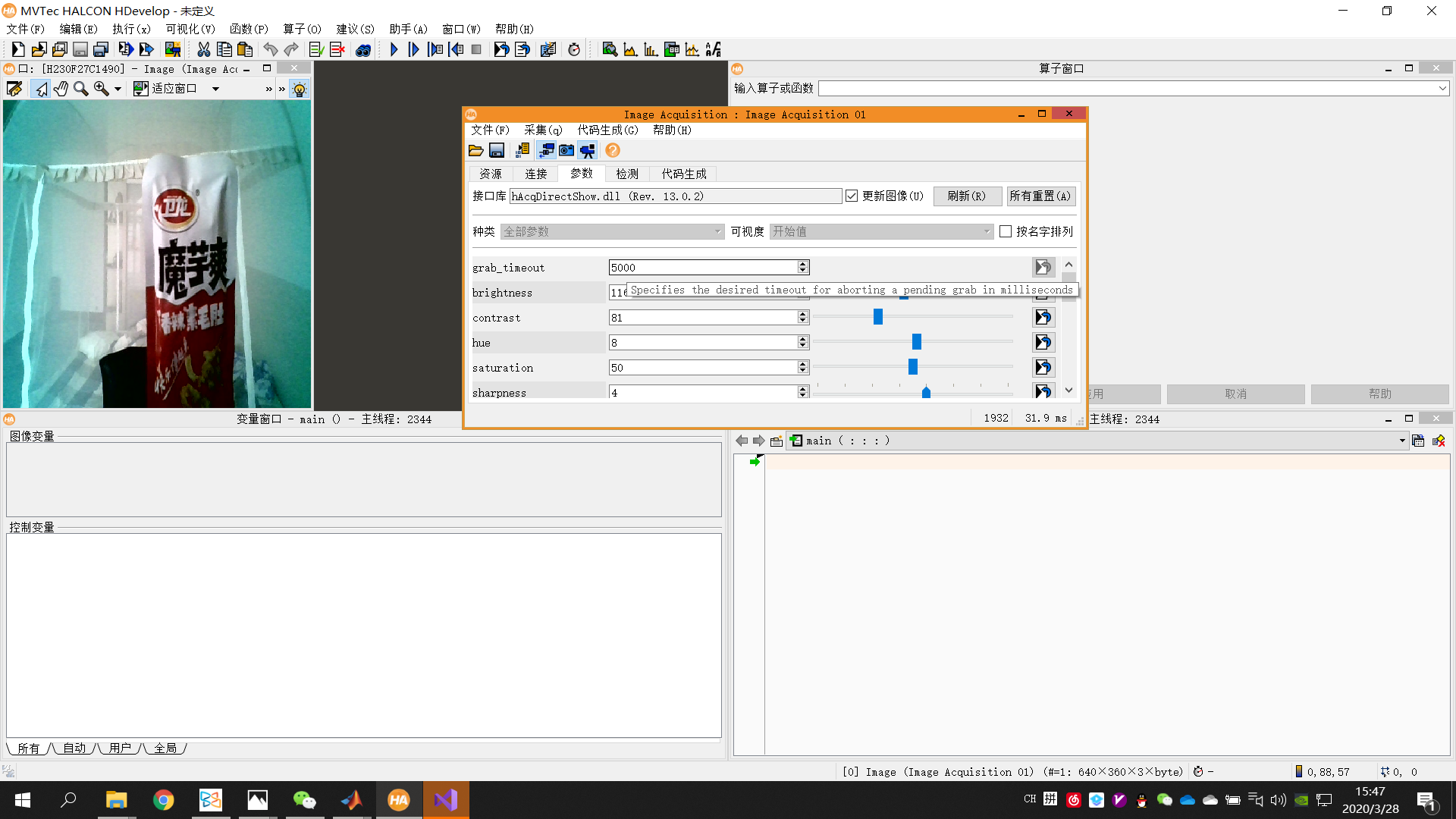
}

#endif

#endif

实验结果：

halcon



Visual studio

