机器视觉第五次作业

1.基于相关性

Halcon代码：

```
read_image (chipImage, 'chip.png')
read_image (boardImage, 'board.png')
rgb1_to_gray(chipImage, chipImage)
rgb1_to_gray(boardImage, boardImage)

dev_close_window ()
dev_open_window_fit_image (chipImage, 0, 0, -1, -1, WindowHandleforChip)
set_display_font (WindowHandleforChip, 16, 'mono', 'true', 'false')
dev_update_off ()
dev_display (chipImage)

create_ncc_model (chipImage, 'auto', 0, 360, 'auto', 'use_polarity', ModelID)

get_image_size(chipImage, Width, Height)
gen_rectangle1(Rect, 2, 2, Height-4, Width-4)

find_ncc_model (boardImage, ModelID, 0, 360, 0.5, 1, 0.5, 'true', 0, Row,
Column, Angle, Score)

dev_open_window_fit_image (boardImage, 0, 0, -1, -1, WindowHandleforBoard)
set_display_font (WindowHandleforChip, 16, 'mono', 'true', 'false')
dev_update_off ()
dev_display (boardImage)

dev_display_ncc_matching_results (ModelID, 'green', Row, Column, Angle, 0)
disp_message (WindowHandleforBoard, 'Found NCC model, Score: '+Score, 'window',
12, 12, 'black', 'true')
disp_message(WindowHandleforBoard, 'Position x: '+Column, 'window', 30, 12,
'black', 'true')
disp_message(WindowHandleforBoard, 'Position y: '+Row, 'window', 45, 12,
'black', 'true')
disp_message(WindowHandleforBoard, 'Angle: '+Angle, 'window', 60, 12, 'black',
'true')

while (1)
    * Without this loop, the displaying window will disappear once showing
endwhile
```

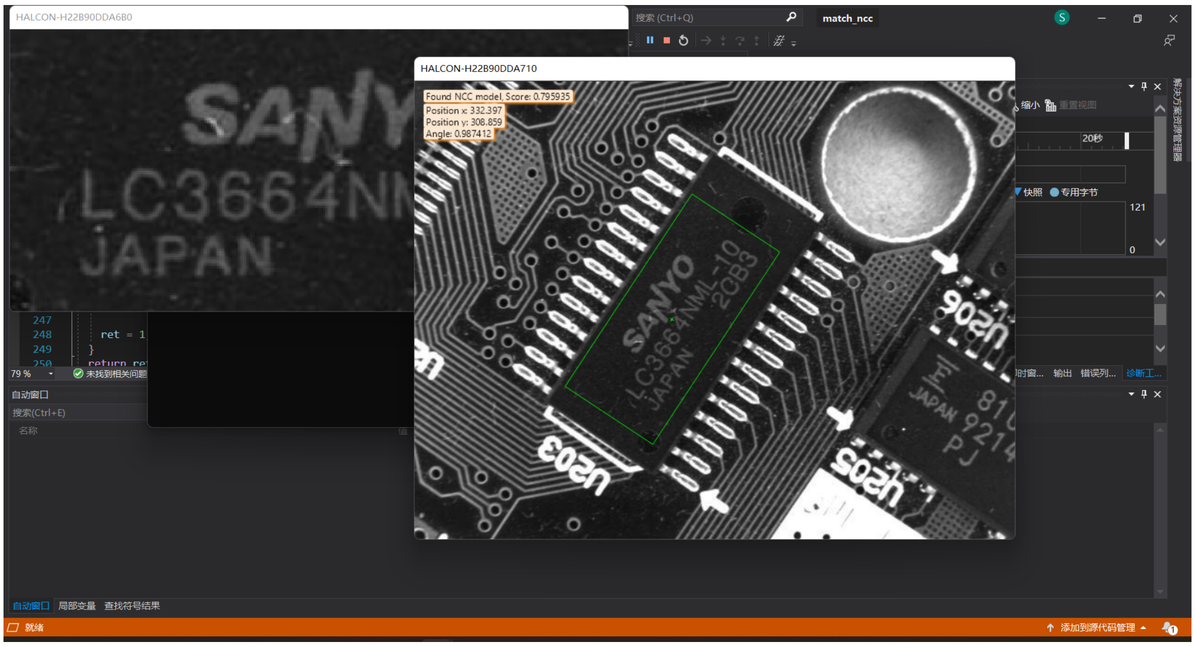
导出的C++代码：

```
///////////////////////////////////////////////////////////////////////////////
// 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;

// Procedure declarations
// External procedures
// Chapter: Matching / Correlation-Based
// Short Description: Display the results of Correlation-Based Matching.
extern void dev_display_ncc_matching_results (HTuple hv_ModelID, HTuple
hv_Color,
    HTuple hv_Row, HTuple hv_Column, HTuple hv_Angle, HTuple hv_Model);
// Chapter: Develop
// Short Description: Open a new graphics window that preserves the aspect ratio
of the given image.
extern void dev_open_window_fit_image (HObject ho_Image, HTuple hv_Row, HTuple
hv_Column,
    HTuple hv_WidthLimit, HTuple hv_HeightLimit, HTuple *hv_WindowHandle);
// Chapter: Develop
// Short Description: Switch dev_update_pc, dev_update_var and dev_update_window
to 'off'.
extern void dev_update_off ();
// Chapter: Graphics / Text
// Short Description: This procedure writes a text message.
extern void disp_message (HTuple hv_WindowHandle, HTuple hv_String, HTuple
hv_CoordSystem,
    HTuple hv_Row, HTuple hv_Column, HTuple hv_Color, HTuple hv_Box);
// Chapter: Graphics / Text
// Short Description: Set font independent of OS
extern void set_display_font (HTuple hv_WindowHandle, HTuple hv_Size, HTuple
hv_Font,
    HTuple hv_Bold, HTuple hv_Slant);

// Procedures
#ifndef NO_EXPORT_MAIN
// Main procedure
```

```
void action()
{

  // Local iconic variables
  HObject  ho_chipImage, ho_boardImage, ho_Rect;

  // Local control variables
  HTuple  hv_WindowHandleforChip, hv_ModelID, hv_Width;
  HTuple  hv_Height, hv_Row, hv_Column, hv_Angle, hv_Score;
  HTuple  hv_WindowHandleforBoard;

  ReadImage(&ho_chipImage, "chip.png");
  ReadImage(&ho_boardImage, "board.png");
  Rgb1ToGray(ho_chipImage, &ho_chipImage);
  Rgb1ToGray(ho_boardImage, &ho_boardImage);

  if (HDevWindowStack::IsOpen())
    CloseWindow(HDevWindowStack::Pop());
  dev_open_window_fit_image(ho_chipImage, 0, 0, -1, -1,
&hv_WindowHandleforChip);
  set_display_font(hv_WindowHandleforChip, 16, "mono", "true", "false");
  dev_update_off();
  if (HDevWindowStack::IsOpen())
    DispObj(ho_chipImage, HDevWindowStack::GetActive());

  CreateNccModel(ho_chipImage, "auto", 0, 360, "auto", "use_polarity",
&hv_ModelID);

  GetImageSize(ho_chipImage, &hv_Width, &hv_Height);
  GenRectangle1(&ho_Rect, 2, 2, hv_Height-4, hv_Width-4);

  FindNccModel(ho_boardImage, hv_ModelID, 0, 360, 0.5, 1, 0.5, "true", 0,
&hv_Row,
      &hv_Column, &hv_Angle, &hv_Score);

  dev_open_window_fit_image(ho_boardImage, 0, 0, -1, -1,
&hv_WindowHandleforBoard);
  set_display_font(hv_WindowHandleforChip, 16, "mono", "true", "false");
  dev_update_off();
  if (HDevWindowStack::IsOpen())
    DispObj(ho_boardImage, HDevWindowStack::GetActive());

  dev_display_ncc_matching_results(hv_ModelID, "green", hv_Row, hv_Column,
hv_Angle,
      0);
  disp_message(hv_WindowHandleforBoard, HTuple("Found NCC model, Score:
")+hv_Score,
      "window", 12, 12, "black", "true");
  disp_message(hv_WindowHandleforBoard, "Position x: "+hv_Column, "window", 30,
12,
      "black", "true");
  disp_message(hv_WindowHandleforBoard, "Position y: "+hv_Row, "window", 45, 12,
      "black", "true");
  disp_message(hv_WindowHandleforBoard, "Angle: "+hv_Angle, "window", 60, 12,
"black",
      "true");

  while (0 != 1)
  {
    //Without this loop, the displaying window will disappear once showing
  }

}


#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
```

运行结果：



```
Found NCC model, Score: 0.795935
Position x: 332.397
Position y: 308.859
Angle: 0.987412
```

2.基于形状

HALCON代码：

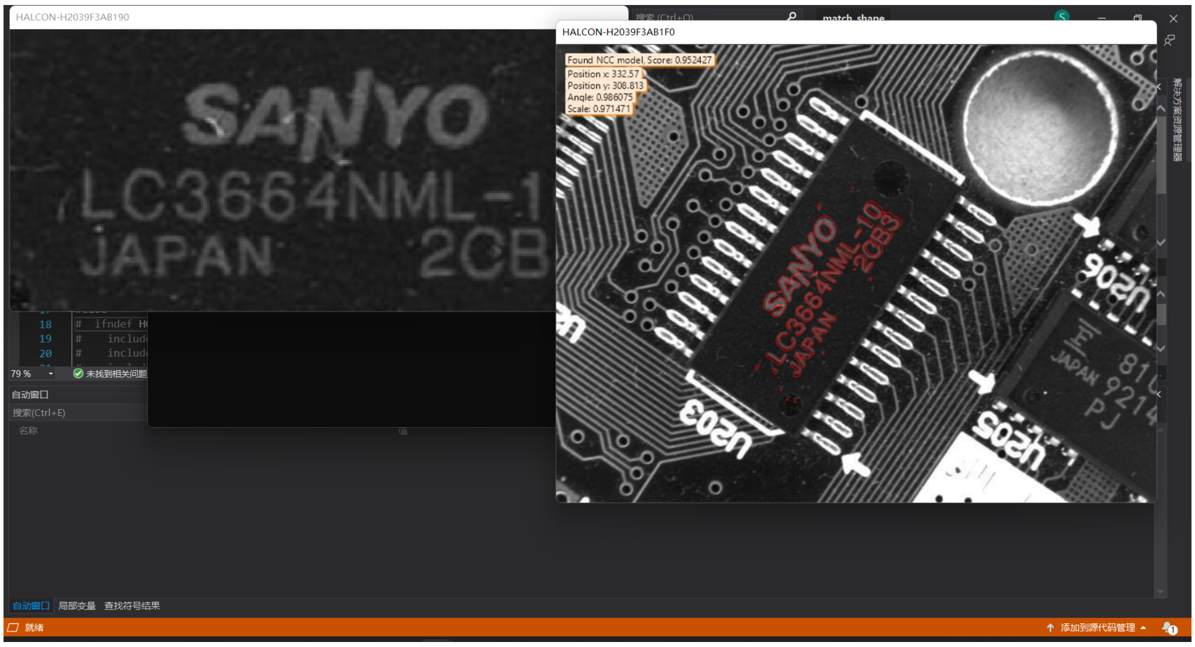
```
read_image (chipImage, 'chip.png')
read_image (boardImage, 'board.png')
rgb1_to_gray(chipImage, chipImage)
rgb1_to_gray(boardImage, boardImage)

dev_close_window ()
dev_open_window_fit_image (chipImage, 0, 0, -1, -1, WindowHandleforChip)
set_display_font (WindowHandleforChip, 16, 'mono', 'true', 'false')
dev_update_off ()
dev_display (chipImage)

create_scaled_shape_model(chipImage, 'auto', 0, 3.14, 'auto', 0.5, 2.0, 'auto',
'auto', 'use_polarity', 'auto', 'auto', ModelID)

find_scaled_shape_model(boardImage, ModelID, 0, 3.14, 0.5, 2.0, 0.8, 1, 0.5,
'least_squares', 0.1, 0.4, Row, Column, Angle, Scale, Score)

dev_open_window_fit_image (boardImage, 0, 0, -1, -1, WindowHandleforBoard)
set_display_font (WindowHandleforChip, 16, 'mono', 'true', 'false')
dev_update_off ()
dev_display (boardImage)

dev_display_shape_matching_results(ModelID, 'red', Row, Column, Angle, 1, 1, 0)
disp_message (WindowHandleforBoard, 'Found NCC model, Score: '+Score, 'window',
12, 12, 'black', 'true')
disp_message(WindowHandleforBoard, 'Position x: '+Column, 'window', 30, 12,
'black', 'true')
disp_message(WindowHandleforBoard, 'Position y: '+Row, 'window', 45, 12,
'black', 'true')
disp_message(WindowHandleforBoard, 'Angle: '+Angle, 'window', 60, 12, 'black',
'true')
disp_message(WindowHandleforBoard, 'Scale: '+Scale, 'window', 75, 12, 'black',
'true')

while (1)
    * Without this loop, the displaying window will disappear once showing
endwhile
```

C++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;

// Procedure declarations
// External procedures
// Chapter: Matching / Shape-Based
// Short Description: Display the results of Shape-Based Matching.
extern void dev_display_shape_matching_results (HTuple hv_ModelID, HTuple
hv_Color,
    HTuple hv_Row, HTuple hv_Column, HTuple hv_Angle, HTuple hv_ScaleR, HTuple
hv_ScaleC,
    HTuple hv_Model);
// Chapter: Develop
// Short Description: Open a new graphics window that preserves the aspect ratio
of the given image.
extern void dev_open_window_fit_image (HObject ho_Image, HTuple hv_Row, HTuple
hv_Column,
    HTuple hv_WidthLimit, HTuple hv_HeightLimit, HTuple *hv_WindowHandle);
// Chapter: Develop
// Short Description: Switch dev_update_pc, dev_update_var and dev_update_window
to 'off'.
extern void dev_update_off ();
// Chapter: Graphics / Text
// Short Description: This procedure writes a text message.
extern void disp_message (HTuple hv_WindowHandle, HTuple hv_String, HTuple
hv_CoordSystem,
    HTuple hv_Row, HTuple hv_Column, HTuple hv_Color, HTuple hv_Box);
// Chapter: Matching / Shape-Based
// Short Description: Calculate the transformation matrix for results of Shape-
Based Matching.
extern void get_hom_mat2d_from_matching_result (HTuple hv_Row, HTuple hv_Column,
    HTuple hv_Angle, HTuple hv_ScaleR, HTuple hv_ScaleC, HTuple *hv_HomMat2D);
// Chapter: Graphics / Text
// Short Description: Set font independent of OS
extern void set_display_font (HTuple hv_WindowHandle, HTuple hv_Size, HTuple
hv_Font,
    HTuple hv_Bold, HTuple hv_Slant);

// Procedures
#ifndef NO_EXPORT_MAIN
// Main procedure
void action()
{
  // Local iconic variables
  HObject  ho_chipImage, ho_boardImage;

  // Local control variables
  HTuple  hv_WindowHandleforChip, hv_ModelID, hv_Row;
  HTuple  hv_Column, hv_Angle, hv_Scale, hv_Score, hv_WindowHandleforBoard;

  ReadImage(&ho_chipImage, "chip.png");
  ReadImage(&ho_boardImage, "board.png");
  Rgb1ToGray(ho_chipImage, &ho_chipImage);
  Rgb1ToGray(ho_boardImage, &ho_boardImage);

  if (HDevWindowStack::IsOpen())
    CloseWindow(HDevWindowStack::Pop());
  dev_open_window_fit_image(ho_chipImage, 0, 0, -1, -1,
&hv_WindowHandleforChip);
  set_display_font(hv_WindowHandleforChip, 16, "mono", "true", "false");
  dev_update_off();
  if (HDevWindowStack::IsOpen())
    DispObj(ho_chipImage, HDevWindowStack::GetActive());

  CreateScaledShapeModel(ho_chipImage, "auto", 0, 3.14, "auto", 0.5, 2.0,
"auto",
      "auto", "use_polarity", "auto", "auto", &hv_ModelID);

  FindScaledShapeModel(ho_boardImage, hv_ModelID, 0, 3.14, 0.5, 2.0, 0.8, 1,
0.5,
      "least_squares", 0.1, 0.4, &hv_Row, &hv_Column, &hv_Angle, &hv_Scale,
&hv_Score);

  dev_open_window_fit_image(ho_boardImage, 0, 0, -1, -1,
&hv_WindowHandleforBoard);
  set_display_font(hv_WindowHandleforChip, 16, "mono", "true", "false");
  dev_update_off();
  if (HDevWindowStack::IsOpen())
    DispObj(ho_boardImage, HDevWindowStack::GetActive());

  dev_display_shape_matching_results(hv_ModelID, "red", hv_Row, hv_Column,
hv_Angle,
      1, 1, 0);
  disp_message(hv_WindowHandleforBoard, HTuple("Found NCC model, Score:
")+hv_Score,
      "window", 12, 12, "black", "true");
  disp_message(hv_WindowHandleforBoard, "Position x: "+hv_Column, "window", 30,
12,
      "black", "true");
  disp_message(hv_WindowHandleforBoard, "Position y: "+hv_Row, "window", 45, 12,
      "black", "true");
  disp_message(hv_WindowHandleforBoard, "Angle: "+hv_Angle, "window", 60, 12,
"black",
      "true");
  disp_message(hv_WindowHandleforBoard, "Scale: "+hv_Scale, "window", 75, 12,
"black",
      "true");

  while (0 != 1)
  {
    //Without this loop, the displaying window will disappear once showing
  }

}


#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
```

运行结果：



```
Found NCC model, Score: 0.952427
Position x: 332.57
Position y: 308.813
Angle: 0.986075
Scale: 0.971471
```