机器视觉作业——模板匹配

基于形状

代码

```
//////////////////////////////////////////////////////////////////////////////
// File generated by HDevelop for HALCON/C++ Version 18.11.0.1
// 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"
#  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, ho_ModelRegion, ho_TemplateImage;
    HObject  ho_ModelContours, ho_TransContours;
```

```
    // Local control variables
    HTuple  hv_w, hv_h, hv_WindowHandle, hv_ModelID;
    HTuple  hv_ModelRegionArea, hv_RefRow, hv_RefColumn, hv_HomMat2D;
    HTuple  hv_TestImages, hv_T, hv_Row, hv_Column, hv_Angle;
    HTuple  hv_Score, hv_I;

    //
    //Matching 01: ************************************************
    //Matching 01: BEGIN of generated code for model initialization
    //Matching 01: ************************************************
    SetSystem("border_shape_models", "false");
    //
    //Matching 01: Obtain the model image
    ReadImage(&ho_Image, "C:/Users/94925/Desktop/Lesson06-software
handout1.jpg");
    //
    GetImageSize(ho_Image, &hv_w, &hv_h);
    SetWindowAttr("background_color", "black");
    OpenWindow(0, 0, hv_w, hv_h, 0, "visible", "", &hv_WindowHandle);
    HDevWindowStack::Push(hv_WindowHandle);
    //Matching 01: Build the ROI from basic regions
    GenRectangle1(&ho_ModelRegion, 14.417, 23.4648, 100.251, 229.632);
    //
    //Matching 01: Reduce the model template
    ReduceDomain(ho_Image, ho_ModelRegion, &ho_TemplateImage);
    //
    //Matching 01: Create the shape model
    CreateShapeModel(ho_TemplateImage, 3, HTuple(0).TupleRad(),
HTuple(90).TupleRad(),
        HTuple(1.0849).TupleRad(),
(HTuple("point_reduction_low").Append("no_pregeneration")),
        "use_polarity", ((HTuple(26).Append(30)).Append(9)), 6, &hv_ModelID);
    //
    //Matching 01: Get the model contour for transforming it later into the
image
    GetShapeModelContours(&ho_ModelContours, hv_ModelID, 1);
    //
    //Matching 01: Get the reference position
    AreaCenter(ho_ModelRegion, &hv_ModelRegionArea, &hv_RefRow, &hv_RefColumn);
    VectorAngleToRigid(0, 0, 0, hv_RefRow, hv_RefColumn, 0, &hv_HomMat2D);
    AffineTransContourXld(ho_ModelContours, &ho_TransContours, hv_HomMat2D);
    //
    //Matching 01: Display the model contours
    if (HDevWindowStack::IsOpen())
        DispObj(ho_Image, HDevWindowStack::GetActive());
    if (HDevWindowStack::IsOpen())
        SetColor(HDevWindowStack::GetActive(), "green");
    if (HDevWindowStack::IsOpen())
        SetDraw(HDevWindowStack::GetActive(), "margin");
    if (HDevWindowStack::IsOpen())
        DispObj(ho_ModelRegion, HDevWindowStack::GetActive());
    if (HDevWindowStack::IsOpen())
        DispObj(ho_TransContours, HDevWindowStack::GetActive());
    // stop(...); only in hdevelop
    //
    //Matching 01: END of generated code for model initialization
    //Matching 01:  * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

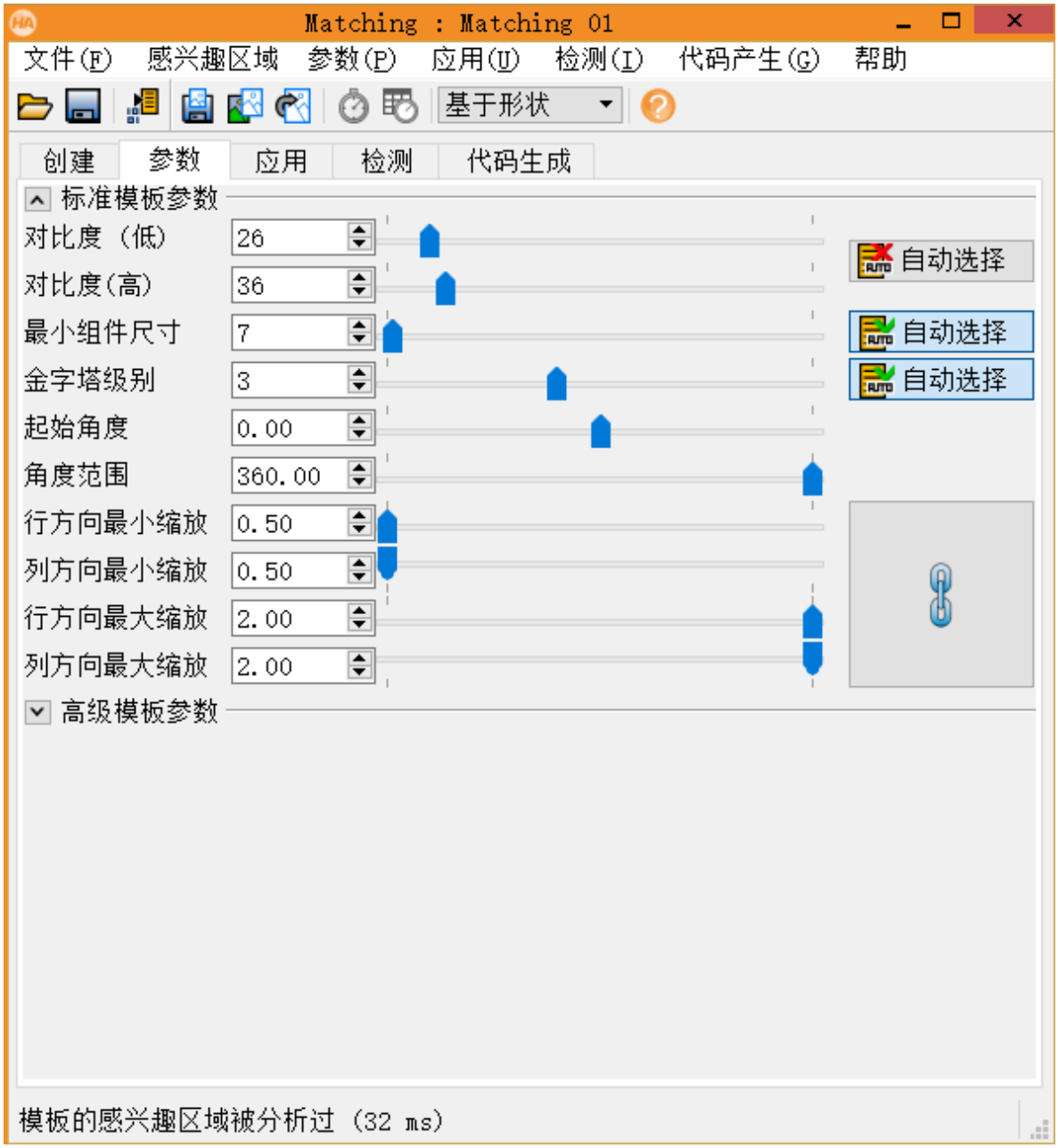
```
    //Matching 01: BEGIN of generated code for model application
    //
    //Matching 01: Loop over all specified test images
    hv_TestImages = "C:/Users/94925/Desktop/Lesson06-software handout.jpg";
    for (hv_T = 0; hv_T <= 0; hv_T += 1)
    {
        //
        //Matching 01: Obtain the test image
        ReadImage(&ho_Image, HTuple(hv_TestImages[hv_T]));
        //
        //Matching 01: Find the model
        FindShapeModel(ho_Image, hv_ModelID, HTuple(0).TupleRad(),
HTuple(90).TupleRad(),
            0.5, 1, 0.5, "least_squares", (HTuple(3).Append(1)), 0.75, &hv_Row,
&hv_Column,
            &hv_Angle, &hv_Score);
        //
        //Matching 01: Transform the model contours into the detected positions
        if (HDevWindowStack::IsOpen())
            DispObj(ho_Image, HDevWindowStack::GetActive());
        {
            HTuple end_val52 = (hv_Score.TupleLength()) - 1;
            HTuple step_val52 = 1;
            for (hv_I = 0; hv_I.Continue(end_val52, step_val52); hv_I +=
step_val52)
            {
                HomMat2dIdentity(&hv_HomMat2D);
                HomMat2dRotate(hv_HomMat2D, HTuple(hv_Angle[hv_I]), 0, 0,
&hv_HomMat2D);
                HomMat2dTranslate(hv_HomMat2D, HTuple(hv_Row[hv_I]),
HTuple(hv_Column[hv_I]),
                    &hv_HomMat2D);
                AffineTransContourXld(ho_ModelContours, &ho_TransContours,
hv_HomMat2D);
                if (HDevWindowStack::IsOpen())
                    SetColor(HDevWindowStack::GetActive(), "green");
                if (HDevWindowStack::IsOpen())
                    DispObj(ho_TransContours, HDevWindowStack::GetActive());
                // stop(...); only in hdevelop
                while (1) {}
            }
        }
    }
    //
    //Matching 01: ************************************************
    //Matching 01: END of generated code for model application
    //Matching 01: ************************************************
    //
}


#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

        // Default settings used in HDevelop (can be omitted)
        SetSystem("width", 512);
        SetSystem("height", 512);

#ifndef __APPLE__
```

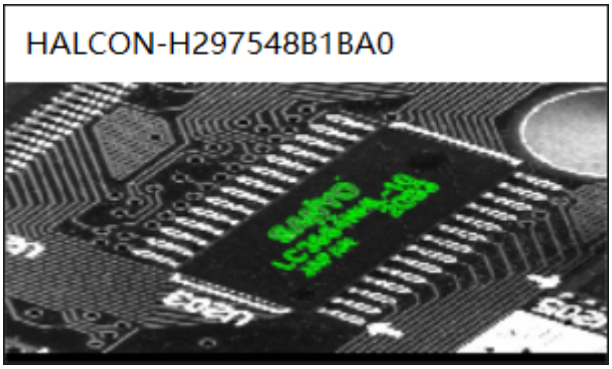
```
        action();
#else
        ret = apple_main(argc, argv);
#endif
    }
    catch (HException& exception)
    {
        fprintf(stderr, "  Error #%u in %s: %s\n", exception.ErrorCode(),
            (const char*)exception.ProcName(),
            (const char*)exception.ErrorMessage());
        ret = 1;
    }
    return ret;
}

#endif


#endif
```

参数设置如下

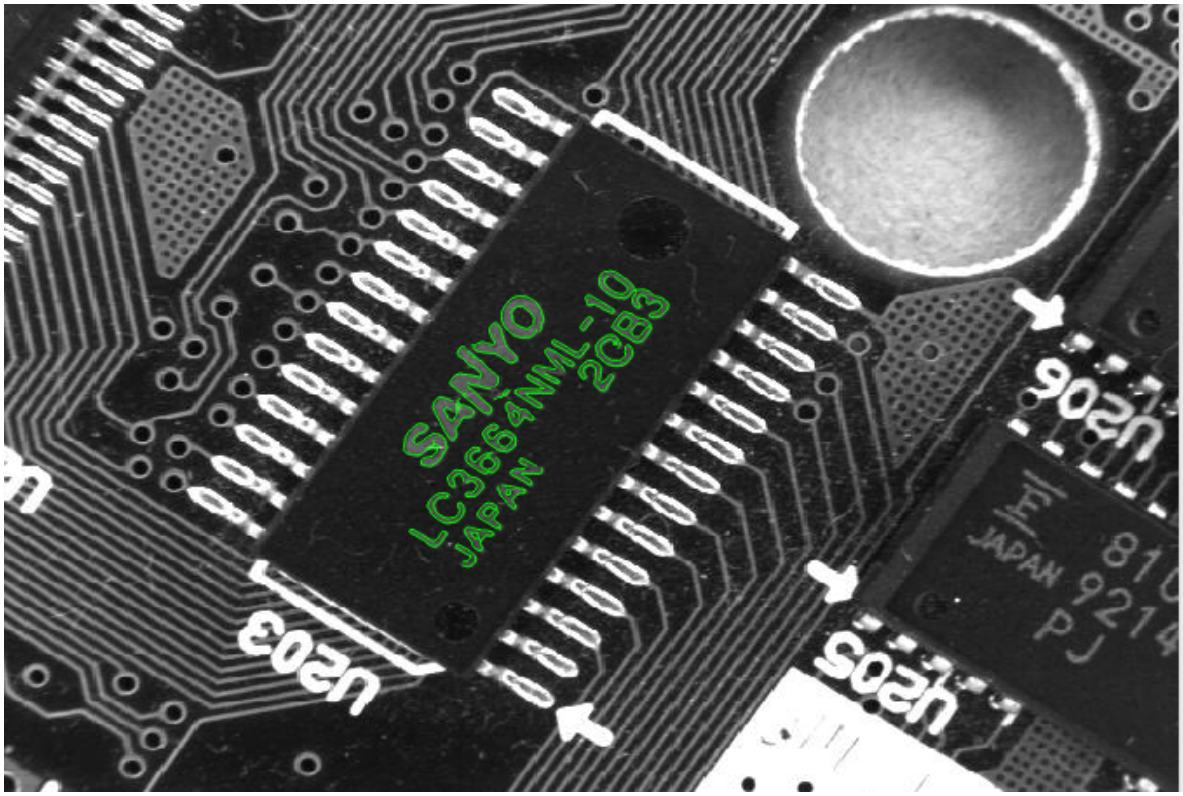


现象

用Visual Studio运行得到的窗口很小很小，而且比例不太对，找了一下原因没有找到



再贴一张halcon里面的运行结果吧



基于相关性

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;

// Procedure declarations
// External procedures
// Chapter: Matching / Correlation-Based
// Short Description: Display the results of Correlation-Based Matching.
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: Graphics / Text
// Short Description: This procedure writes a text message.
void disp_message(HTuple hv_WindowHandle, HTuple hv_String, HTuple
hv_CoordSystem,
    HTuple hv_Row, HTuple hv_Column, HTuple hv_Color, HTuple hv_Box);

// Procedures
// External procedures
// Chapter: Matching / Correlation-Based
// Short Description: Display the results of Correlation-Based Matching.
void dev_display_ncc_matching_results(HTuple hv_ModelID, HTuple hv_Color, HTuple
hv_Row,
    HTuple hv_Column, HTuple hv_Angle, HTuple hv_Model)
{

    // Local iconic variables
    HObject  ho_ModelRegion, ho_ModelContours, ho_ContoursAffinTrans;
    HObject  ho_Cross;

    // Local control variables
    HTuple  hv_NumMatches, hv_Index, hv_Match, hv_HomMat2DIdentity;
    HTuple  hv_HomMat2DRotate, hv_HomMat2DTranslate, hv_RowTrans;
    HTuple  hv_ColTrans;

    //This procedure displays the results of Correlation-Based Matching.
    //
    hv_NumMatches = hv_Row.TupleLength();
    if (0 != (hv_NumMatches > 0))
    {
        if (0 != ((hv_Model.TupleLength()) == 0))
        {
            TupleGenConst(hv_NumMatches, 0, &hv_Model);
        }
        else if (0 != ((hv_Model.TupleLength()) == 1))
        {
            TupleGenConst(hv_NumMatches, hv_Model, &hv_Model);
        }
        {
            HTuple end_val9 = (hv_ModelID.TupleLength()) - 1;
            HTuple step_val9 = 1;
```

```
            for (hv_Index = 0; hv_Index.Continue(end_val9, step_val9); hv_Index
+= step_val9)
            {
                GetNccModelRegion(&ho_ModelRegion,
HTuple(hv_ModelID[hv_Index]));
                GenContourRegionXld(ho_ModelRegion, &ho_ModelContours,
"border_holes");
                if (HDevWindowStack::IsOpen())
                    SetColor(HDevWindowStack::GetActive(),
HTuple(hv_Color[hv_Index % (hv_Color.TupleLength())]));
                {
                    HTuple end_val13 = hv_NumMatches - 1;
                    HTuple step_val13 = 1;
                    for (hv_Match = 0; hv_Match.Continue(end_val13, step_val13);
hv_Match += step_val13)
                    {
                        if (0 != (hv_Index == HTuple(hv_Model[hv_Match])))
                        {
                            HomMat2dIdentity(&hv_HomMat2DIdentity);
                            HomMat2dRotate(hv_HomMat2DIdentity,
HTuple(hv_Angle[hv_Match]), 0, 0, &hv_HomMat2DRotate);
                            HomMat2dTranslate(hv_HomMat2DRotate,
HTuple(hv_Row[hv_Match]), HTuple(hv_Column[hv_Match]),
                                &hv_HomMat2DTranslate);
                            AffineTransContourXld(ho_ModelContours,
&ho_ContoursAffinTrans, hv_HomMat2DTranslate);
                            if (HDevWindowStack::IsOpen())
                                DispObj(ho_ContoursAffinTrans,
HDevWindowStack::GetActive());
                            AffineTransPixel(hv_HomMat2DTranslate, 0, 0,
&hv_RowTrans, &hv_ColTrans);
                            GenCrossContourXld(&ho_Cross, hv_RowTrans,
hv_ColTrans, 6, HTuple(hv_Angle[hv_Match]));
                            if (HDevWindowStack::IsOpen())
                                DispObj(ho_Cross, HDevWindowStack::GetActive());



                        }
                    }
                }
            }
        }
    }
    return;
}

// Chapter: Graphics / Text
// Short Description: This procedure writes a text message.
void disp_message(HTuple hv_WindowHandle, HTuple hv_String, HTuple
hv_CoordSystem,
    HTuple hv_Row, HTuple hv_Column, HTuple hv_Color, HTuple hv_Box)
{

    // Local iconic variables

    // Local control variables
    HTuple  hv_GenParamName, hv_GenParamValue;
```

```
    //This procedure displays text in a graphics window.
    //
    //Input parameters:
    //WindowHandle: The WindowHandle of the graphics window, where
    //   the message should be displayed
    //String: A tuple of strings containing the text message to be displayed
    //CoordSystem: If set to 'window', the text position is given
    //   with respect to the window coordinate system.
    //   If set to 'image', image coordinates are used.
    //   (This may be useful in zoomed images.)
    //Row: The row coordinate of the desired text position
    //   A tuple of values is allowed to display text at different
    //   positions.
    //Column: The column coordinate of the desired text position
    //   A tuple of values is allowed to display text at different
    //   positions.
    //Color: defines the color of the text as string.
    //   If set to [], '' or 'auto' the currently set color is used.
    //   If a tuple of strings is passed, the colors are used cyclically...
    //   - if |Row| == |Column| == 1: for each new textline
    //   = else for each text position.
    //Box: If Box[0] is set to 'true', the text is written within an orange box.
    //     If set to' false', no box is displayed.
    //     If set to a color string (e.g. 'white', '#FF00CC', etc.),
    //       the text is written in a box of that color.
    //     An optional second value for Box (Box[1]) controls if a shadow is
displayed:
    //       'true' -> display a shadow in a default color
    //       'false' -> display no shadow
    //       otherwise -> use given string as color string for the shadow color
    //
    //It is possible to display multiple text strings in a single call.
    //In this case, some restrictions apply:
    //- Multiple text positions can be defined by specifying a tuple
    //  with multiple Row and/or Column coordinates, i.e.:
    //  - |Row| == n, |Column| == n
    //  - |Row| == n, |Column| == 1
    //  - |Row| == 1, |Column| == n
    //- If |Row| == |Column| == 1,
    //  each element of String is display in a new textline.
    //- If multiple positions or specified, the number of Strings
    //  must match the number of positions, i.e.:
    //  - Either |String| == n (each string is displayed at the
    //                          corresponding position),
    //  - or     |String| == 1 (The string is displayed n times).
    //
    //
    //Convert the parameters for disp_text.
    if (0 != (HTuple(hv_Row == HTuple()).TupleOr(hv_Column == HTuple())))
    {
        return;
    }
    if (0 != (hv_Row == -1))
    {
        hv_Row = 12;
    }
    if (0 != (hv_Column == -1))
```

```
    {
        hv_Column = 12;
    }
    //
    //Convert the parameter Box to generic parameters.
    hv_GenParamName = HTuple();
    hv_GenParamValue = HTuple();
    if (0 != ((hv_Box.TupleLength()) > 0))
    {
        if (0 != (HTuple(hv_Box[0]) == HTuple("false")))
        {
            //Display no box
            hv_GenParamName = hv_GenParamName.TupleConcat("box");
            hv_GenParamValue = hv_GenParamValue.TupleConcat("false");
        }
        else if (0 != (HTuple(hv_Box[0]) != HTuple("true")))
        {
            //Set a color other than the default.
            hv_GenParamName = hv_GenParamName.TupleConcat("box_color");
            hv_GenParamValue = hv_GenParamValue.TupleConcat(HTuple(hv_Box[0]));
        }
    }
    if (0 != ((hv_Box.TupleLength()) > 1))
    {
        if (0 != (HTuple(hv_Box[1]) == HTuple("false")))
        {
            //Display no shadow.
            hv_GenParamName = hv_GenParamName.TupleConcat("shadow");
            hv_GenParamValue = hv_GenParamValue.TupleConcat("false");
        }
        else if (0 != (HTuple(hv_Box[1]) != HTuple("true")))
        {
            //Set a shadow color other than the default.
            hv_GenParamName = hv_GenParamName.TupleConcat("shadow_color");
            hv_GenParamValue = hv_GenParamValue.TupleConcat(HTuple(hv_Box[1]));
        }
    }
    //Restore default CoordSystem behavior.
    if (0 != (hv_CoordSystem != HTuple("window")))
    {
        hv_CoordSystem = "image";
    }
    //
    if (0 != (hv_Color == HTuple("")))
    {
        //disp_text does not accept an empty string for Color.
        hv_Color = HTuple();
    }
    //
    DispText(hv_WindowHandle, hv_String, hv_CoordSystem, hv_Row, hv_Column,
hv_Color,
        hv_GenParamName, hv_GenParamValue);
    return;
}

#ifndef NO_EXPORT_MAIN
// Main procedure
void action()
```

```
{

    // Local iconic variables
    HObject  ho_Image, ho_ROI_0, ho_ImageReduced;

    // Local control variables
    HTuple  hv_ImageFiles, hv_width, hv_height, hv_WindowHandle;
    HTuple  hv_ModelID, hv_Index, hv_Row, hv_Column, hv_Angle;
    HTuple  hv_Score, hv_Message;

    //Image Acquisition 01: Code generated by Image Acquisition 01
    ListFiles("C:/Users/94925/Desktop/Imgdata",
(HTuple("files").Append("follow_links")),
        &hv_ImageFiles);
    TupleRegexpSelect(hv_ImageFiles, (HTuple("\\.
(tif|tiff|gif|bmp|jpg|jpeg|jp2|png|pcx|pgm|ppm|pbm|xwd|ima|hobj)$").Append("igno
re_case")),
        &hv_ImageFiles);
    ReadImage(&ho_Image, HTuple(hv_ImageFiles[0]));
    GetImageSize(ho_Image, &hv_width, &hv_height);
    SetWindowAttr("background_color", "black");
    OpenWindow(0, 0, hv_width, hv_height, 0, "visible", "", &hv_WindowHandle);
    HDevWindowStack::Push(hv_WindowHandle);
    GenRectangle1(&ho_ROI_0, 195, 188, 290, 399);
    ReduceDomain(ho_Image, ho_ROI_0, &ho_ImageReduced);
    //create_scaled_shape_model (ImageReduced, 'auto', 0, rad(360), 'auto', 0.9,
1.1, 'auto', 'auto', 'use_polarity', 'auto', 'auto', ModelID)
    CreateNccModel(ho_ImageReduced, "auto", 0, HTuple(360).TupleRad(), "auto",
"use_polarity",
        &hv_ModelID);
    {
        HTuple end_val10 = (hv_ImageFiles.TupleLength()) - 1;
        HTuple step_val10 = 1;
        for (hv_Index = 0; hv_Index.Continue(end_val10, step_val10); hv_Index +=
step_val10)
        {
            ReadImage(&ho_Image, HTuple(hv_ImageFiles[hv_Index]));
            if (HDevWindowStack::IsOpen())
                DispObj(ho_Image, HDevWindowStack::GetActive());
            //Image Acquisition 01: Do something
            //find_scaled_shape_model (Image, ModelID, 0, rad(360), 0.8, 1.2,
0.5, 1, 0.5, 'least_squares', 0, 0.9, Row, Column, Angle, Scale, Score)
            FindNccModel(ho_Image, hv_ModelID, 0, HTuple(360).TupleRad(), 0.8,
1, 0.5, "true",
                0, &hv_Row, &hv_Column, &hv_Angle, &hv_Score);
            //dev_display_shape_matching_results (ModelID, 'green', Row, Column,
Angle, 1, 1, 0)
            dev_display_ncc_matching_results(hv_ModelID, "red", hv_Row,
hv_Column, hv_Angle,
                0);
            hv_Message = (((((((//'(x,y,θ,score):\n'
                HTuple("(x,y,\246\310,score):\n") + hv_Column) + "\n") + hv_Row)
+ "\n") + ((360 * hv_Angle) / (HTuple(360).TupleRad()))) +//'°'
                "\241\343") + "\n") + hv_Score;
            disp_message(hv_WindowHandle, hv_Message, "window", hv_Row,
hv_Column, "black",
                "true");
            // stop(...); only in hdevelop
```

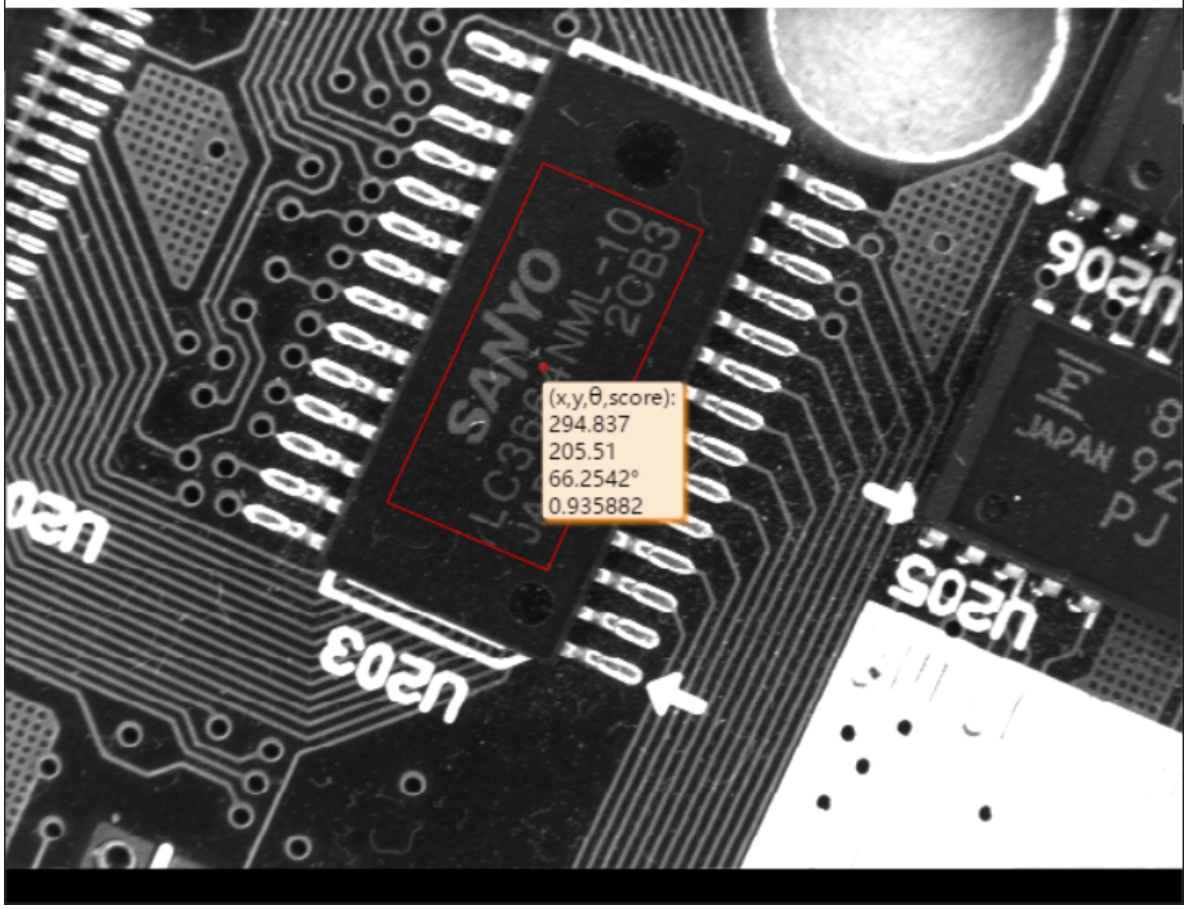
```
            for (int i = 0; i < 1e9; ++i);
        }
    }
    ClearNccModel(hv_ModelID);
    if (HDevWindowStack::IsOpen())
        CloseWindow(HDevWindowStack::Pop());


}


#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
```

现象

HALCON-H2595CC1D700



HALCON-H2595CC1D700