机器视觉作业五

SZ170320207

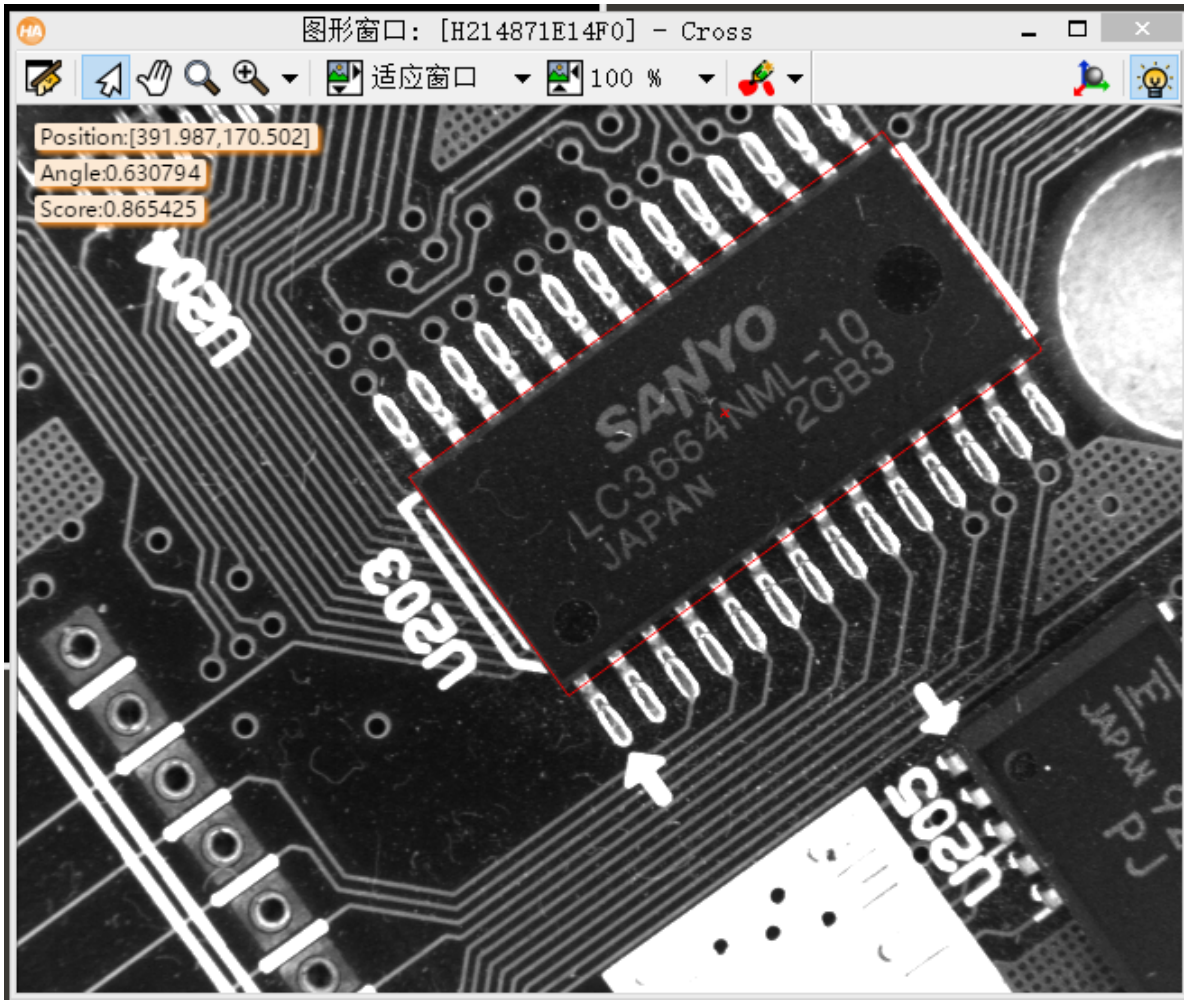
刘健恒

1.相关性模板匹配

halcon

```
* Image Acquisition 01: Code generated by Image Acquisition 01

list_files ('D:/OneDrive -
stu.hit.edu.cn/Lessons/Machine_Vision/Homeworks/5/Imgdata',
['files','follow_links'], ImageFiles)
tuple_regexp_select (ImageFiles, ['\\.
(tif|tiff|gif|bmp|jpg|jpeg|jp2|png|pcx|pgm|ppm|pbm|xwd|ima|hobj)$','ignore_case'
], ImageFiles)
read_image (Image, ImageFiles[0])
get_image_size (Image, Width, Height)
dev_open_window (0, 0, Width, Height, 'black', WindowHandle)
gen_rectangle1 (ROI_0, 170.973, 140.886, 320.027, 465.07)
reduce_domain (Image, ROI_0, ImageReduced)
create_ncc_model (ImageReduced, 'auto', 0, rad(360), 'auto', 'use_polarity',
ModelID1)
for Index := 0 to |ImageFiles| - 1 by 1
    read_image (Image, ImageFiles[Index])
    dev_display (Image)
    find_ncc_model (Image, ModelID1, 0, rad(360), 0.8, 1, 0.5, 'true', 0, Row,
Column, Angle, Score)
    dev_display_ncc_matching_results (ModelID1, 'red', Row, Column, Angle, 0)
    dev_disp_text ('Position:'+'['+Column+','+Row+']', 'window', 10, 10,
'black', [], [])
    dev_disp_text ('Angle:'+Angle, 'window', 30, 10, 'black', [], [])
    dev_disp_text ('Score:'+Score, 'window', 50, 10, 'black', [], [])
    stop()
    * Image Acquisition 01: Do something
endfor
clear_shape_model(ModelID)
```

测试截图



C++

```
//////////////////////////////////////////////////////////////////////////////
// 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
// Chapter: Matching / Shape-Based
// Short Description: Display the results of Shape-Based Matching.
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);

// Procedures
// Chapter: Matching / Shape-Based
// Short Description: Display the results of Shape-Based Matching.
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)
{

    // Local iconic variables
    HObject  ho_ModelContours, ho_ContoursAffinTrans;

    // Local control variables
    HTuple  hv_NumMatches, hv_Index, hv_Match, hv_HomMat2DIdentity;
    HTuple  hv_HomMat2DScale, hv_HomMat2DRotate, hv_HomMat2DTranslate;

    //This procedure displays the results of Shape-Based Matching.
    //
    hv_NumMatches = hv_Row.TupleLength();
    if (0 != (hv_NumMatches > 0))
    {
        if (0 != ((hv_ScaleR.TupleLength()) == 1))
        {
            TupleGenConst(hv_NumMatches, hv_ScaleR, &hv_ScaleR);
        }
        if (0 != ((hv_ScaleC.TupleLength()) == 1))
        {
            TupleGenConst(hv_NumMatches, hv_ScaleC, &hv_ScaleC);
        }
        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_val15 = (hv_ModelID.TupleLength()) - 1;
```

```
            HTuple step_val15 = 1;
            for (hv_Index = 0; hv_Index.Continue(end_val15, step_val15);
hv_Index += step_val15)
            {
                GetShapeModelContours(&ho_ModelContours,
HTuple(hv_ModelID[hv_Index]), 1);
                if (HDevWindowStack::IsOpen())
                    SetColor(HDevWindowStack::GetActive(),
HTuple(hv_Color[hv_Index % (hv_Color.TupleLength())]));
                {
                    HTuple end_val18 = hv_NumMatches - 1;
                    HTuple step_val18 = 1;
                    for (hv_Match = 0; hv_Match.Continue(end_val18, step_val18);
hv_Match += step_val18)
                    {
                        if (0 != (hv_Index == HTuple(hv_Model[hv_Match])))
                        {
                            HomMat2dIdentity(&hv_HomMat2DIdentity);
                            HomMat2dScale(hv_HomMat2DIdentity,
HTuple(hv_ScaleR[hv_Match]), HTuple(hv_ScaleC[hv_Match]),
                                0, 0, &hv_HomMat2DScale);
                            HomMat2dRotate(hv_HomMat2DScale,
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());
                        }
                    }
                }
            }
        }
    }
    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_Scale, hv_Score;

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

    ListFiles("D:/OneDrive -
stu.hit.edu.cn/Lessons/Machine_Vision/Homeworks/5/Imgdata",
        (HTuple("files").Append("follow_links")), &hv_ImageFiles);
```

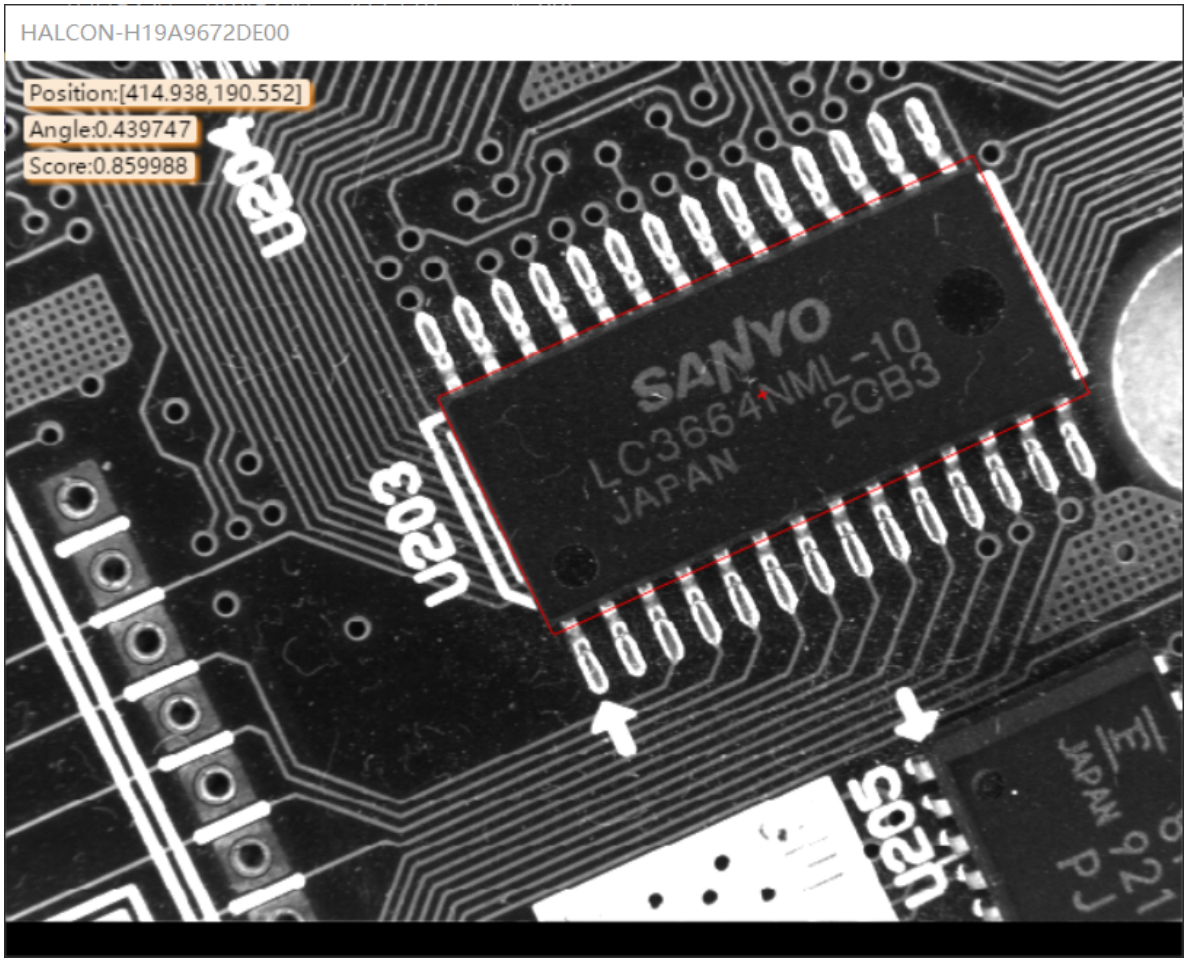
```
    TupleRegexpSelect(hv_ImageFiles, (HTuple("\\.
(tif|tiff|gif|bmp|jpg|jpeg|jp2|png|pcx|pgm|ppm|pbm|xwd|ima|hobj)$").Append("igno
re_case")),
        &hv_ImageFiles);
    ReadImage(&ho_Image, HTuple(hv_ImageFiles[0]));
    GetImageSize(ho_Image, &hv_Width, &hv_Height);
    SetWindowAttr("background_color", "black");
    OpenWindow(0, 0, hv_Width, hv_Height, 0, "visible", "", &hv_WindowHandle);
    HDevWindowStack::Push(hv_WindowHandle);
    GenRectangle1(&ho_ROI_0, 170.973, 140.886, 320.027, 465.07);
    ReduceDomain(ho_Image, ho_ROI_0, &ho_ImageReduced);
    CreateShapeModel(ho_ImageReduced, "auto", 0, HTuple(360).TupleRad(), "auto",
"auto",
        "use_polarity", "auto", "auto", &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());
            FindScaledShapeModel(ho_Image, hv_ModelID, 0,
HTuple(360).TupleRad(), 0.9, 1.1,
                0.5, 1, 0.5, "least_squares", 0, 0.9, &hv_Row, &hv_Column,
&hv_Angle, &hv_Scale,
                &hv_Score);
            dev_display_shape_matching_results(hv_ModelID, "red", hv_Row,
hv_Column, hv_Angle,
                1, 1, 0);
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), ((((HTuple("Position:") +
"[") + hv_Column) + HTuple(",")) + hv_Row) + "]",
                    "window", 10, 10, "black", HTuple(), HTuple());
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), "Angle:" + hv_Angle,
"window", 30, 10,
                    "black", HTuple(), HTuple());
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), "Score:" + hv_Score,
"window", 50, 10,
                    "black", HTuple(), HTuple());
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), "Scale:" + hv_Scale,
"window", 70, 10,
                    "black", HTuple(), HTuple());

            // stop(...); only in hdevelop
            //Image Acquisition 01: Do something
        }
    }
    ClearShapeModel(hv_ModelID);
}


#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
```

测试截图



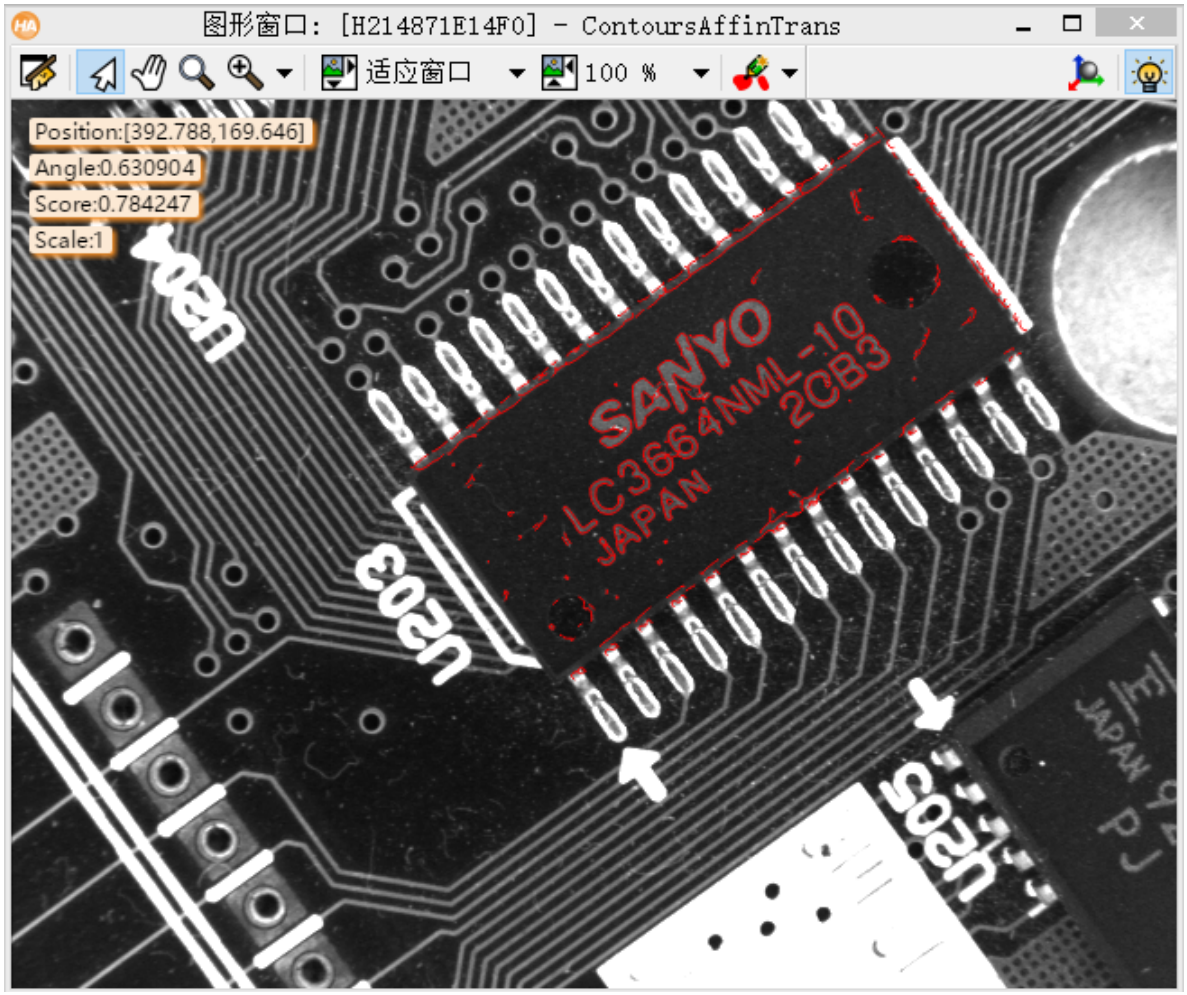
2.形状模板匹配

halcon

```
* Image Acquisition 01: Code generated by Image Acquisition 01

list_files ('D:/OneDrive -
stu.hit.edu.cn/Lessons/Machine_Vision/Homeworks/5/Imgdata',
['files','follow_links'], ImageFiles)
tuple_regexp_select (ImageFiles, ['\\.
(tif|tiff|gif|bmp|jpg|jpeg|jp2|png|pcx|pgm|ppm|pbm|xwd|ima|hobj)$','ignore_case'
], ImageFiles)
read_image (Image, ImageFiles[0])
get_image_size (Image, Width, Height)
dev_open_window (0, 0, Width, Height, 'black', WindowHandle)
gen_rectangle1 (ROI_0, 170.973, 140.886, 320.027, 465.07)
reduce_domain (Image, ROI_0, ImageReduced)
create_shape_model (ImageReduced, 'auto', 0, rad(360), 'auto', 'auto',
'use_polarity', 'auto', 'auto', ModelID)
for Index := 0 to |ImageFiles| - 1 by 1
    read_image (Image, ImageFiles[Index])
    dev_display (Image)
    find_scaled_shape_model (Image, ModelID, 0, rad(360), 0.9, 1.1, 0.5, 1, 0.5,
'least_squares', 0, 0.9, Row, Column, Angle, Scale, Score)
    dev_display_shape_matching_results (ModelID, 'red', Row, Column, Angle, 1,
1, 0)
    dev_disp_text ('Position:'+'['+Column+','+Row+']', 'window', 10, 10,
'black', [], [])
```

```
    dev_disp_text ('Angle:'+Angle, 'window', 30, 10, 'black', [], [])
    dev_disp_text ('Score:'+Score, 'window', 50, 10, 'black', [], [])
    dev_disp_text ('Scale:'+Scale, 'window', 70, 10, 'black', [], [])
    stop()
    * Image Acquisition 01: Do something
endfor
clear_shape_model(ModelID)
```

测试截图



C++

```
//////////////////////////////////////////////////////////////////////////////
// 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
// Chapter: Matching / Shape-Based
// Short Description: Display the results of Shape-Based Matching.
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);

// Procedures
// Chapter: Matching / Shape-Based
// Short Description: Display the results of Shape-Based Matching.
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)
{

    // Local iconic variables
    HObject  ho_ModelContours, ho_ContoursAffinTrans;

    // Local control variables
    HTuple  hv_NumMatches, hv_Index, hv_Match, hv_HomMat2DIdentity;
    HTuple  hv_HomMat2DScale, hv_HomMat2DRotate, hv_HomMat2DTranslate;

    //This procedure displays the results of Shape-Based Matching.
    //
    hv_NumMatches = hv_Row.TupleLength();
    if (0 != (hv_NumMatches > 0))
    {
        if (0 != ((hv_ScaleR.TupleLength()) == 1))
        {
            TupleGenConst(hv_NumMatches, hv_ScaleR, &hv_ScaleR);
        }
        if (0 != ((hv_ScaleC.TupleLength()) == 1))
        {
            TupleGenConst(hv_NumMatches, hv_ScaleC, &hv_ScaleC);
        }
```

```
        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_val15 = (hv_ModelID.TupleLength()) - 1;
            HTuple step_val15 = 1;
            for (hv_Index = 0; hv_Index.Continue(end_val15, step_val15);
hv_Index += step_val15)
            {
                GetShapeModelContours(&ho_ModelContours,
HTuple(hv_ModelID[hv_Index]), 1);
                if (HDevWindowStack::IsOpen())
                    SetColor(HDevWindowStack::GetActive(),
HTuple(hv_Color[hv_Index % (hv_Color.TupleLength())]));
                {
                    HTuple end_val18 = hv_NumMatches - 1;
                    HTuple step_val18 = 1;
                    for (hv_Match = 0; hv_Match.Continue(end_val18, step_val18);
hv_Match += step_val18)
                    {
                        if (0 != (hv_Index == HTuple(hv_Model[hv_Match])))
                        {
                            HomMat2dIdentity(&hv_HomMat2DIdentity);
                            HomMat2dScale(hv_HomMat2DIdentity,
HTuple(hv_ScaleR[hv_Match]), HTuple(hv_ScaleC[hv_Match]),
                                0, 0, &hv_HomMat2DScale);
                            HomMat2dRotate(hv_HomMat2DScale,
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());
                        }
                    }
                }
            }
        }
    }
    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_Scale, hv_Score;

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

    ListFiles("D:/OneDrive -
stu.hit.edu.cn/Lessons/Machine_Vision/Homeworks/5/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, 170.973, 140.886, 320.027, 465.07);
    ReduceDomain(ho_Image, ho_ROI_0, &ho_ImageReduced);
    CreateShapeModel(ho_ImageReduced, "auto", 0, HTuple(360).TupleRad(), "auto",
"auto",
        "use_polarity", "auto", "auto", &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());
            FindScaledShapeModel(ho_Image, hv_ModelID, 0,
HTuple(360).TupleRad(), 0.9, 1.1,
                0.5, 1, 0.5, "least_squares", 0, 0.9, &hv_Row, &hv_Column,
&hv_Angle, &hv_Scale,
                &hv_Score);
            dev_display_shape_matching_results(hv_ModelID, "red", hv_Row,
hv_Column, hv_Angle,
                1, 1, 0);
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), ((((HTuple("Position:") +
"[") + hv_Column) + HTuple(",")) + hv_Row) + "]",
                    "window", 10, 10, "black", HTuple(), HTuple());
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), "Angle:" + hv_Angle,
"window", 30, 10,
                    "black", HTuple(), HTuple());
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), "Score:" + hv_Score,
"window", 50, 10,
                    "black", HTuple(), HTuple());
            if (HDevWindowStack::IsOpen())
                DispText(HDevWindowStack::GetActive(), "Scale:" + hv_Scale,
"window", 70, 10,
                    "black", HTuple(), HTuple());
```

```
            // stop(...); only in hdevelop
            //Image Acquisition 01: Do something
        }
    }
    ClearShapeModel(hv_ModelID);
}


#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
```

测试截图

HALCON-H17B33D9DE00