AT.Toolkit.WindowImageH 二次开发教程

📖 概述

AT.Toolkit.WindowImageH 是一个基于 Halcon 的图像显示窗口工具，用于在视觉检测应用中显示图像、绘制几何对象、标注文本等功能。本教程将详细介绍如何在二次开发中使用该模块。

第一章：获取图像窗口实例

1.1 功能说明

在使用图像窗口前，需要先通过容器获取实例。

1.2 代码示例

```
using AT.Librarys.Container;
using AT.Librarys.Container.Toolkit.Vision;

/// <summary>
/// 获取图像窗口实例
/// </summary>
public class WindowImageExample
{
    // 图像窗口实例
    private IWindowImage _windowImage;

    /// <summary>
    /// 初始化图像窗口
    /// </summary>
    /// <param name="instanceName">实例名称，如"主相机图像窗口"</param>
    public void Initialize(string instanceName)
    {
        // 方式一：通过容器获取已创建的实例
        _windowImage = DS.Container.GetToolkit<IWindowImage>(instanceName);
```

```
        // 验证实例是否获取成功
        if (_windowImage == null)
        {
            DS.NotifyG.Error($"获取图像窗口失败: {instanceName}");
            return;
        }

        DS.NotifyG.Info($"图像窗口初始化成功: {instanceName}");
    }
}
```

第二章：显示图像

2.1 功能说明

将相机采集的图像或从文件读取的图像显示到窗口中。

2.2 代码示例

```
using AT.Librarys.Container.Toolkit.Vision;
using HalconDotNet;

/// <summary>
/// 显示图像到窗口
/// </summary>
public class ShowImageExample
{
    private IWindowImage _windowImage;

    /// <summary>
    /// 显示图像（基本方式）
    /// </summary>
    /// <param name="image">Halcon图像对象</param>
    public void ShowImage(HObject image)
    {
        // 将 HObject 转换为 CObject
        CObject cImage = image.ToCObject();

        // 显示图像，自动清除之前的显示内容
        bool success = _windowImage.ShowImage(cImage);

        if (success)
        {
```

```
            DS.NotifyG.Info("图像显示成功");
        }
    }

    /// <summary>
    /// 显示图像（不清除之前内容）
    /// </summary>
    /// <param name="image">Halcon图像对象</param>
    public void ShowImageWithoutClear(HObject image)
    {
        CObject cImage = image.ToCObject();

        // 第二个参数 isClear = false，不清除之前的显示内容
        _windowImage.ShowImage(cImage, false);
    }

    /// <summary>
    /// 从文件读取并显示图像
    /// </summary>
    /// <param name="filePath">图像文件路径</param>
    public void ShowImageFromFile(string filePath)
    {
        // 使用 Halcon 读取图像
        HOperatorSet.ReadImage(out HObject image, filePath);

        // 显示图像
        _windowImage.ShowImage(image.ToCObject());

        // 适应窗口显示（自动缩放）
        _windowImage.WindowFit();
    }
}
```

第三章：显示几何对象

3.1 功能说明

在图像上绘制点、线、圆、矩形等几何图形，用于标注检测结果。

3.2 代码示例

```
using AT.Librarys.Container.Toolkit.Vision;

/// <summary>
```

```
/// 显示几何对象示例
/// </summary>
public class ShowObjectExample
{
    private IWindowImage _windowImage;

    /// <summary>
    /// 显示十字点
    /// </summary>
    /// <param name="row">行坐标</param>
    /// <param name="column">列坐标</param>
    public void ShowPoint(double row, double column)
    {
        // 创建点对象
        CPoint point = new CPoint(row, column);

        // 在指定位置显示十字点，大小为50像素，颜色为绿色
        _windowImage.ShowObject(point, 50, WindowColor.绿色);
    }

    /// <summary>
    /// 显示带角度的十字点
    /// </summary>
    public void ShowPointWithAngle(double row, double column, double angle)
    {
        // 创建带角度的点对象
        CPointAngle pointAngle = new CPointAngle()
        {
            Point = new CPoint(row, column),
            Angle = angle  // 角度，单位：度
        };

        // 显示带角度的十字点
        _windowImage.ShowObject(pointAngle, 80, WindowColor.红色);
    }

    /// <summary>
    /// 显示直线
    /// </summary>
    public void ShowLine(double row1, double col1, double row2, double col2)
    {
        // 创建线对象
        CLine line = new CLine()
        {
            Point1 = new CPoint(row1, col1),  // 起点
            Point2 = new CPoint(row2, col2)   // 终点
        };

        // 显示直线，使用蓝色
        _windowImage.ShowObject(line, WindowColor.蓝色);
    }

    /// <summary>
    /// 显示圆
    /// </summary>
    public void ShowCircle(double centerRow, double centerCol, double radius)
    {
```

```
        // 创建圆对象
        CCircle circle = new CCircle()
        {
            Point = new CPoint(centerRow, centerCol),  // 圆心
            Radius = radius                            // 半径
        };

        // 显示圆，使用黄色
        _windowImage.ShowObject(circle, WindowColor.黄色);
    }

    /// <summary>
    /// 显示矩形1（轴对齐矩形）
    /// </summary>
    public void ShowRectangle1(double row1, double col1, double row2, double col2)
    {
        // 创建轴对齐矩形
        CRectangle1 rect = new CRectangle1()
        {
            Point1 = new CPoint(row1, col1),  // 左上角
            Point2 = new CPoint(row2, col2)   // 右下角
        };

        // 显示矩形
        _windowImage.ShowObject(rect, WindowColor.青色);
    }

    /// <summary>
    /// 显示矩形2（旋转矩形）
    /// </summary>
    public void ShowRectangle2(double centerRow, double centerCol,
                               double phi, double length1, double length2)
    {
        // 创建旋转矩形
        CRectangle2 rect2 = new CRectangle2()
        {
            Point = new CPoint(centerRow, centerCol),  // 中心点
            Phi = phi,                                 // 旋转角度（弧度）
            Length1 = length1,                         // 半长
            Length2 = length2                          // 半宽
        };

        // 显示旋转矩形，带箭头指示方向
        _windowImage.ShowObject(rect2, true, WindowColor.洋红);
    }

    /// <summary>
    /// 显示椭圆
    /// </summary>
    public void ShowEllipse(double centerRow, double centerCol,
                            double angle, double radius1, double radius2)
    {
        // 创建椭圆对象
        CEllipse ellipse = new CEllipse()
        {
            Point = new CPoint(centerRow, centerCol),  // 中心点
            Angle = angle,                             // 旋转角度（度）
```

```
            Radius1 = radius1,                          // 长轴半径
            Radius2 = radius2                           // 短轴半径
        };

        // 显示椭圆
        _windowImage.ShowObject(ellipse, WindowColor.橙色);
    }

    /// <summary>
    /// 显示 Halcon 区域对象
    /// </summary>
    public void ShowHalconObject(HObject region)
    {
        // 直接显示 Halcon 对象（区域、XLD轮廓等）
        _windowImage.ShowObject(region.ToCObject(), WindowColor.绿色);
    }
}
```

第四章：显示文本标注

4.1 功能说明

在图像上添加文字标注，用于显示检测结果、测量数值等信息。

4.2 代码示例

```
using AT.Librarys.Container.Toolkit.Vision;

/// <summary>
/// 显示文本标注示例
/// </summary>
public class ShowTextExample
{
    private IWindowImage _windowImage;

    /// <summary>
    /// 显示自动排列文本
    /// 文本会自动在窗口左上角依次排列
    /// </summary>
    public void ShowAutoText(string text)
    {
        // 自动排列显示文本，使用默认颜色
        _windowImage.ShowText(text);
    }
```

```
/// <summary>
/// 显示带颜色的自动排列文本
/// </summary>
public void ShowAutoTextWithColor(string text, bool isOK)
{
    // 根据检测结果设置颜色
    WindowColor color = isOK ? WindowColor.绿色 : WindowColor.红色;

    // 显示文本
    _windowImage.ShowText(text, color);
}

/// <summary>
/// 显示指定字体大小的文本
/// </summary>
public void ShowTextWithFontSize(string text, int fontSize)
{
    // 显示指定字体大小的文本
    _windowImage.ShowText(text, fontSize, WindowColor.白色);
}

/// <summary>
/// 在指定位置显示文本
/// </summary>
public void ShowTextAtPosition(string text, double row, double column)
{
    // 创建位置点
    CPoint position = new CPoint(row, column);

    // 在图像坐标系的指定位置显示文本
    _windowImage.ShowText(text, position, WindowColor.黄色, WindowImageCoord.Image);
}

/// <summary>
/// 显示测量结果文本
/// </summary>
public void ShowMeasureResult(string itemName, double value, double row, double column)
{
    // 格式化测量结果
    string text = $"{itemName}: {value:F3}";

    // 在测量点附近显示结果
    CPoint position = new CPoint(row + 30, column + 10);

    // 创建自定义字体（宋体，16号，不加粗，不斜体）
    WindowImageFont font = new WindowImageFont("宋体", 16, false, false);

    // 显示文本
    _windowImage.ShowText(text, position, WindowColor.绿色, WindowImageCoord.Image, font);
}

/// <summary>
/// 显示多行检测结果
/// </summary>
public void ShowMultipleResults()
{
```

```
        // 清除之前的显示
        _windowImage.Clear();

        // 显示标题
        _windowImage.ShowText("检测结果:", 20, WindowColor.白色);

        // 显示各项测量值（自动换行排列）
        _windowImage.ShowText("厚度: 10.235 mm", WindowColor.绿色);
        _windowImage.ShowText("宽度: 5.012 mm", WindowColor.绿色);
        _windowImage.ShowText("高度: 3.008 mm", WindowColor.红色);  // NG项用红色
        _windowImage.ShowText("结果: NG", WindowColor.红色);
    }
}
```

第五章：缓存后批量显示

5.1 功能说明

先将多个对象添加到缓存，最后一次性显示，可以提高显示效率。

5.2 代码示例

```
using AT.Librarys.Container.Toolkit.Vision;
using System.Collections.Generic;

/// <summary>
/// 缓存后批量显示示例
/// </summary>
public class BufferDisplayExample
{
    private IWindowImage _windowImage;

    /// <summary>
    /// 批量显示多个检测点
    /// </summary>
    public void ShowMultiplePoints(List<CPoint> points)
    {
        // 清除之前的显示
        _windowImage.Clear();

        // 将所有点添加到缓存（不会立即显示）
        foreach (var point in points)
        {
            _windowImage.AddObject(point, 30, WindowColor.绿色);
```

```
        }

        // 添加统计文本到缓存
        _windowImage.AddText($"检测点数: {points.Count}", WindowColor.白色);

        // 一次性显示所有缓存内容
        _windowImage.ShowBuffer();
    }

    /// <summary>
    /// 批量显示检测结果
    /// 包括检测区域、测量点、标注文本
    /// </summary>
    public void ShowDetectionResult(CCircle circle, CLine line,
                                    double distance, bool isOK)
    {
        // 清除窗口
        _windowImage.Clear();

        // 添加圆到缓存
        _windowImage.AddObject(circle, WindowColor.蓝色);

        // 添加线到缓存
        _windowImage.AddObject(line, WindowColor.黄色);

        // 添加测量结果文本
        WindowColor resultColor = isOK ? WindowColor.绿色 : WindowColor.红色;
        _windowImage.AddText($"距离: {distance:F3} mm", resultColor);
        _windowImage.AddText($"结果: {(isOK ? "OK" : "NG")}", resultColor);

        // 一次性显示
        _windowImage.ShowBuffer();
    }

    /// <summary>
    /// 在指定位置批量添加文本
    /// </summary>
    public void AddTextAtPositions()
    {
        // 在图像上的多个位置添加标注
        _windowImage.AddText("点1", new CPoint(100, 100), WindowColor.白色);
        _windowImage.AddText("点2", new CPoint(200, 200), WindowColor.白色);
        _windowImage.AddText("点3", new CPoint(300, 300), WindowColor.白色);

        // 显示所有标注
        _windowImage.ShowBuffer();
    }
}
```

第六章：清除显示内容

6.1 功能说明

清除窗口中的图像、对象、文本等内容。

6.2 代码示例

```
using AT.Librarys.Container.Toolkit.Vision;

/// <summary>
/// 清除显示内容示例
/// </summary>
public class ClearDisplayExample
{
    private IWindowImage _windowImage;

    /// <summary>
    /// 清除所有显示内容
    /// 包括图像、对象、文本
    /// </summary>
    public void ClearAll()
    {
        _windowImage.Clear();
    }

    /// <summary>
    /// 只清除对象和文本，保留图像
    /// </summary>
    public void ClearObjectsOnly()
    {
        // 清除对象（包括几何图形和文本）
        _windowImage.ClearObject();
    }

    /// <summary>
    /// 清除缓存内容
    /// 在使用 AddObject/AddText 但未调用 ShowBuffer 时使用
    /// </summary>
    public void ClearBuffer()
    {
        _windowImage.ClearBuffer();
    }

    /// <summary>
    /// 重新显示当前图像
    /// 清除叠加对象后重新显示原图
    /// </summary>
    public void RefreshImage()
    {
        // 先清除对象
        _windowImage.ClearObject();

        // 重新显示当前图像
```

```
        _windowImage.ShowImage();

        // 适应窗口
        _windowImage.WindowFit();
    }
}
```

第七章：保存图像

7.1 功能说明

将当前窗口中的图像保存到文件。

7.2 代码示例

```
using AT.Librarys.Container.Toolkit.Vision;
using System;
using System.IO;

/// <summary>
/// 保存图像示例
/// </summary>
public class SaveImageExample
{
    private IWindowImage _windowImage;

    /// <summary>
    /// 保存当前图像
    /// </summary>
    /// <param name="folderPath">保存文件夹路径</param>
    /// <param name="fileName">文件名（不含扩展名）</param>
    public void SaveCurrentImage(string folderPath, string fileName)
    {
        // 确保文件夹存在
        if (!Directory.Exists(folderPath))
        {
            Directory.CreateDirectory(folderPath);
        }

        // 构建完整路径
        string fullPath = Path.Combine(folderPath, $"{fileName}.png");

        // 保存图像
        _windowImage.SaveImage(fullPath);
```

```
        DS.NotifyG.Info($"图像已保存: {fullPath}");
    }

    /// <summary>
    /// 按时间戳保存 NG 图像
    /// </summary>
    public void SaveNGImage(string basePath)
    {
        // 生成带时间戳的文件名
        string timestamp = DateTime.Now.ToString("yyyyMMdd_HHmmss_fff");
        string fileName = $"NG_{timestamp}";

        // 创建 NG 图像子文件夹
        string ngFolder = Path.Combine(basePath, "NG", DateTime.Now.ToString("yyyyMMdd"));

        SaveCurrentImage(ngFolder, fileName);
    }

    /// <summary>
    /// 保存带检测结果的图像
    /// </summary>
    public void SaveResultImage(string basePath, string sn, bool isOK)
    {
        // 根据结果选择文件夹
        string resultFolder = isOK ? "OK" : "NG";
        string dateFolder = DateTime.Now.ToString("yyyyMMdd");

        // 构建保存路径
        string savePath = Path.Combine(basePath, resultFolder, dateFolder);

        // 使用条码作为文件名
        string fileName = $"{sn}_{DateTime.Now:HHmmss}";

        SaveCurrentImage(savePath, fileName);
    }
}
```

第八章：窗口控制

8.1 功能说明

控制图像窗口的显示行为，如适应窗口、置顶显示等。

8.2 代码示例

```
using AT.Librarys.Container.Toolkit.Vision;

/// <summary>
/// 窗口控制示例
/// </summary>
public class WindowControlExample
{
    private IWindowImage _windowImage;

    /// <summary>
    /// 图像适应窗口大小
    /// 自动缩放图像以完整显示在窗口中
    /// </summary>
    public void FitWindow()
    {
        _windowImage.WindowFit();
    }

    /// <summary>
    /// 将窗口显示在最前面
    /// 适用于调试模式下需要观察图像的场景
    /// </summary>
    public void BringToTop()
    {
        _windowImage.ShowWindowTop();
    }

    /// <summary>
    /// 设置像素当量
    /// 用于将像素值转换为实际物理尺寸
    /// </summary>
    /// <param name="mmPerPixel">每像素代表的毫米数</param>
    public void SetPixelEquivalent(double mmPerPixel)
    {
        _windowImage.PixelEquivalent = mmPerPixel;
    }

    /// <summary>
    /// 获取当前图像
    /// 可用于后续处理或保存
    /// </summary>
    public CObject GetCurrentImage()
    {
        return _windowImage.Image;
    }
}
```

第九章：完整使用示例

9.1 功能说明

综合使用图像窗口的各项功能，实现完整的检测结果显示。

9.2 代码示例

```
using AT.Librarys.Container;
using AT.Librarys.Container.Toolkit.Vision;
using HalconDotNet;
using System;
using System.Collections.Generic;

/// <summary>
/// 完整使用示例：视觉检测结果显示
/// </summary>
public class CompleteExample
{
    // 图像窗口实例
    private IWindowImage _windowImage;

    /// <summary>
    /// 初始化
    /// </summary>
    public bool Initialize(string windowName)
    {
        // 获取图像窗口实例
        _windowImage = DS.Container.GetToolkit<IWindowImage>(windowName);

        if (_windowImage == null)
        {
            DS.NotifyG.Error($"图像窗口初始化失败: {windowName}");
            return false;
        }

        return true;
    }

    /// <summary>
    /// 显示检测结果
    /// </summary>
    /// <param name="image">检测图像</param>
    /// <param name="circles">检测到的圆列表</param>
    /// <param name="lines">检测到的线列表</param>
    /// <param name="results">测量结果</param>
    /// <param name="isOK">总体结果</param>
    public void ShowDetectionResult(HObject image,
                                    List<CCircle> circles,
                                    List<CLine> lines,
```

```
                                Dictionary<string, double> results,
                                bool isOK)
{
    try
    {
        // 1. 显示图像（清除之前内容）
        _windowImage.ShowImage(image.ToCObject());

        // 2. 显示检测到的圆
        foreach (var circle in circles)
        {
            _windowImage.AddObject(circle, WindowColor.蓝色);

            // 在圆心显示十字点
            _windowImage.AddObject(circle.Point, 20, WindowColor.黄色);
        }

        // 3. 显示检测到的线
        foreach (var line in lines)
        {
            _windowImage.AddObject(line, WindowColor.绿色);
        }

        // 4. 显示测量结果文本
        _windowImage.AddText("=== 检测结果 ===", WindowColor.白色);

        foreach (var item in results)
        {
            string text = $"{item.Key}: {item.Value:F3}";
            _windowImage.AddText(text, WindowColor.黄色);
        }

        // 5. 显示总体结果
        string resultText = isOK ? "检测结果: OK" : "检测结果: NG";
        WindowColor resultColor = isOK ? WindowColor.绿色 : WindowColor.红色;
        _windowImage.AddText(resultText, 24, resultColor);

        // 6. 一次性显示所有内容
        _windowImage.ShowBuffer();

        // 7. 适应窗口显示
        _windowImage.WindowFit();

        // 8. 如果是 NG，保存图像
        if (!isOK)
        {
            string savePath = $"D:\\NGImages\\{DateTime.Now:yyyyMMdd}\\";
            string fileName = $"NG_{DateTime.Now:HHmmss_fff}";
            _windowImage.SaveImage(savePath + fileName + ".png");
        }
    }
    catch (Exception ex)
    {
        DS.NotifyG.Error($"显示检测结果失败: {ex.Message}");
    }
}
```

```
    /// <summary>
    /// 显示定位结果
    /// </summary>
    /// <param name="image">图像</param>
    /// <param name="position">定位位置</param>
    /// <param name="angle">角度</param>
    /// <param name="region">定位区域</param>
    public void ShowPositionResult(HObject image,
                                   CPoint position,
                                   double angle,
                                   HObject region)
    {
        // 显示图像
        _windowImage.ShowImage(image.ToCObject());

        // 显示定位区域
        _windowImage.AddObject(region.ToCObject(), WindowColor.绿色);

        // 显示定位点（带角度的十字）
        CPointAngle pointAngle = new CPointAngle()
        {
            Point = position,
            Angle = angle
        };
        _windowImage.AddObject(pointAngle, 100, WindowColor.红色);

        // 显示坐标信息
        _windowImage.AddText($"X: {position.Column:F3}", WindowColor.白色);
        _windowImage.AddText($"Y: {position.Row:F3}", WindowColor.白色);
        _windowImage.AddText($"角度: {angle:F3}°", WindowColor.白色);

        // 显示
        _windowImage.ShowBuffer();
    }
}
```

附录A：颜色对照表

颜色名称	说明
`WindowColor.红色`	常用于 NG 标注
`WindowColor.绿色`	常用于 OK 标注
`WindowColor.蓝色`	常用于检测区域
`WindowColor.黄色`	常用于测量结果

颜色名称	说明
`WindowColor.白色`	常用于普通文本
`WindowColor.青色`	辅助颜色
`WindowColor.洋红`	辅助颜色
`WindowColor.橙色`	辅助颜色
`WindowColor.深蓝`	默认对象颜色

附录B：常用几何对象类

类名	说明	主要属性
`CPoint`	点	Row, Column
`CPointAngle`	带角度的点	Point, Angle
`CLine`	直线	Point1, Point2
`CCircle`	圆	Point(圆心), Radius
`CCircleArc`	圆弧	Circle, StartAngle, EndAngle
`CRectangle1`	轴对齐矩形	Point1, Point2
`CRectangle2`	旋转矩形	Point, Phi, Length1, Length2
`CEllipse`	椭圆	Point, Angle, Radius1, Radius2
`CEllipseArc`	椭圆弧	Ellipse, StartAngle, EndAngle
`CContour`	轮廓	Points (点列表)

附录C：坐标系说明

坐标系类型	说明
`WindowImageCoord.Image`	图像坐标系，以图像左上角为原点
`WindowImageCoord.Window`	窗口坐标系，以窗口左上角为原点

技术支持

如有问题，请联系开发团队。