BECKHOFF

TwinCAT Vision 应用案例框架
—— V1.0

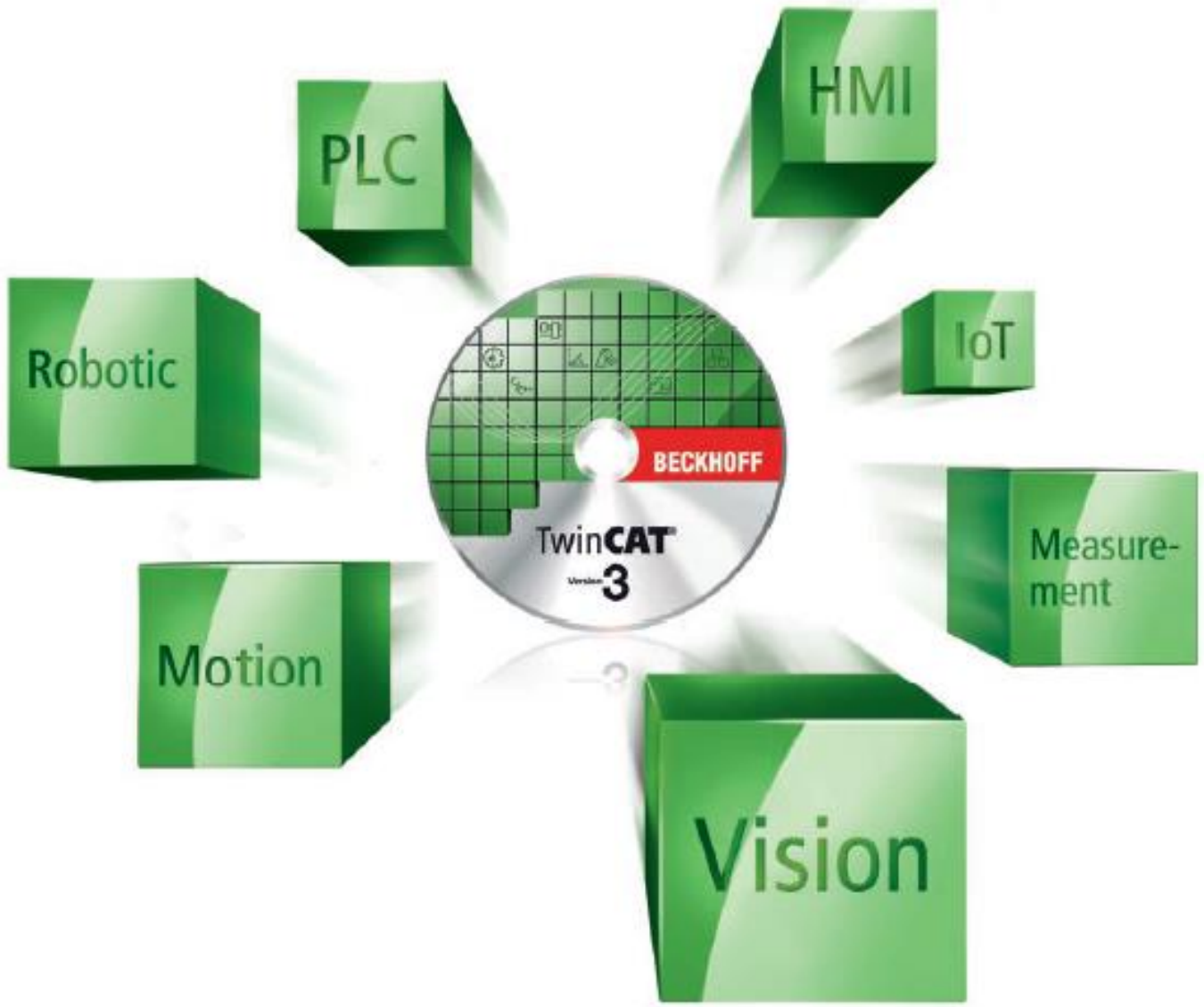
产品经理 杨煜敏
技术支持 陈志光

目录

BECKHOFF

1. **项目案例一：旋转角度检测**
   - 项目介绍
   - 程序框架
   - C# 调试界面
   - 总结

2. **项目案例二：模板匹配定位**
   - 项目介绍
   - 程序框架
   - C# 调试界面
   - 总结



项目案例一：旋转角度检测

BECKHOFF

- **需求：**

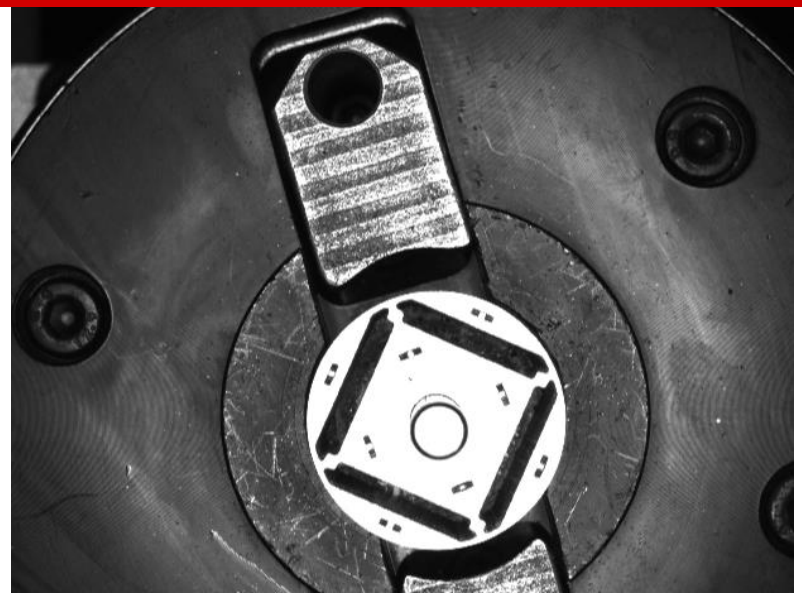
图片是一个电动机的转子，卡在固定的卡槽中，检测其旋转角度，并调整正确的角度，给后续工序正确装配永磁体。

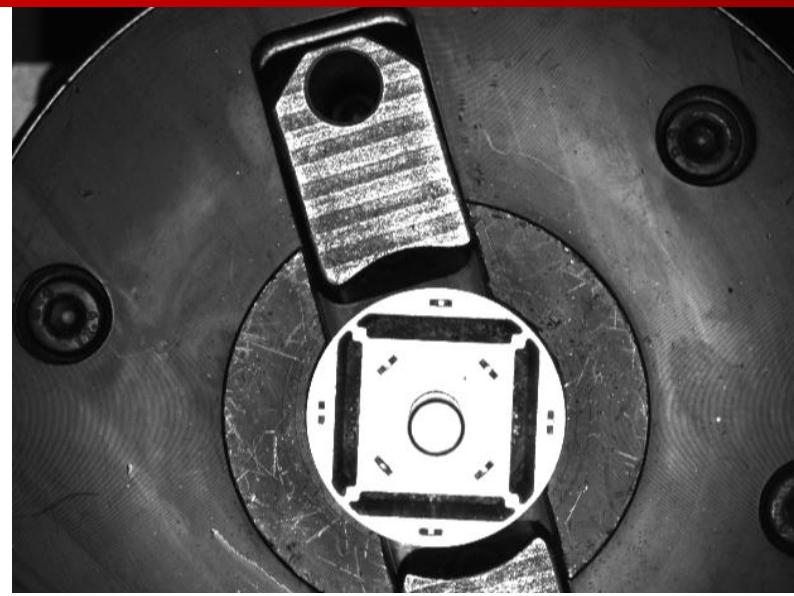
- **实现方式：**

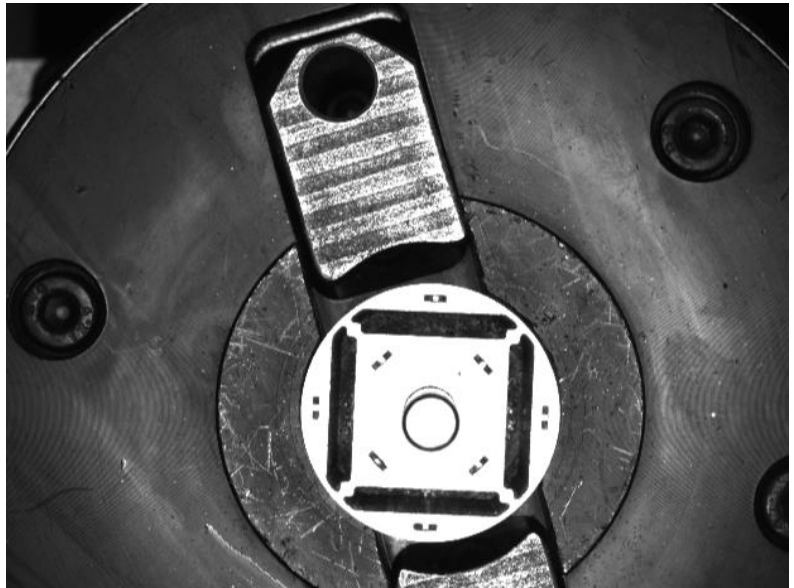
此需求可以利用两种方式寻找角度

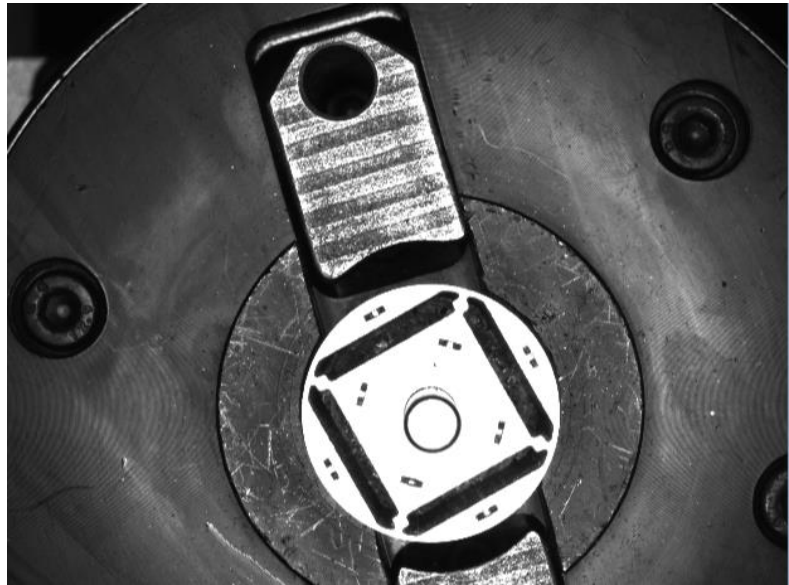
方式一：
（找轮廓）外界矩形获取角度信息
授权：***TF7100 TC3 Vision Base***

方式二：
（寻边）寻找边界计算角度信息
授权：***TF7100 TC3 Vision Base***
TF7300 TC3 Vision Metrology 2D

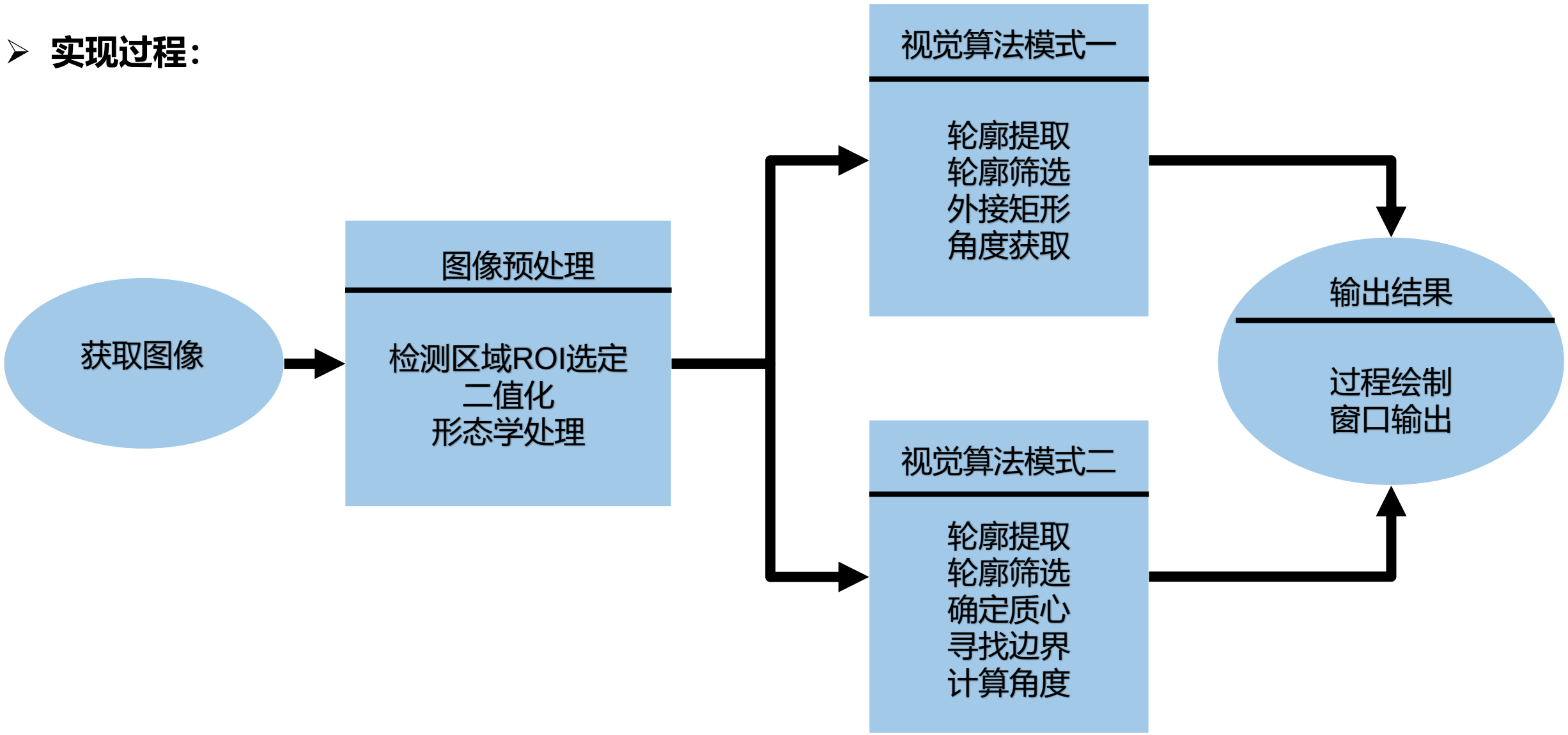








- **实现过程：**


获取图像
图像预处理
检测区域ROI选定
二值化
形态学处理
视觉算法模式一
轮廓提取
轮廓筛选
外接矩形
角度获取
视觉算法模式二
轮廓提取
轮廓筛选
确定质心
寻找边界
计算角度
输出结果
过程绘制
窗口输出

项目案例一：旋转角度检测

BECKHOFF

➢ 获取图像

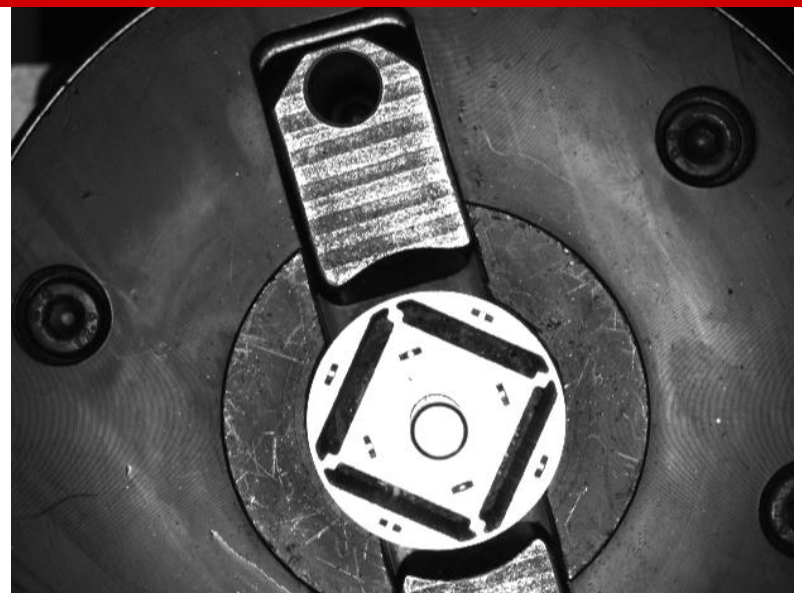
初始化程序代码

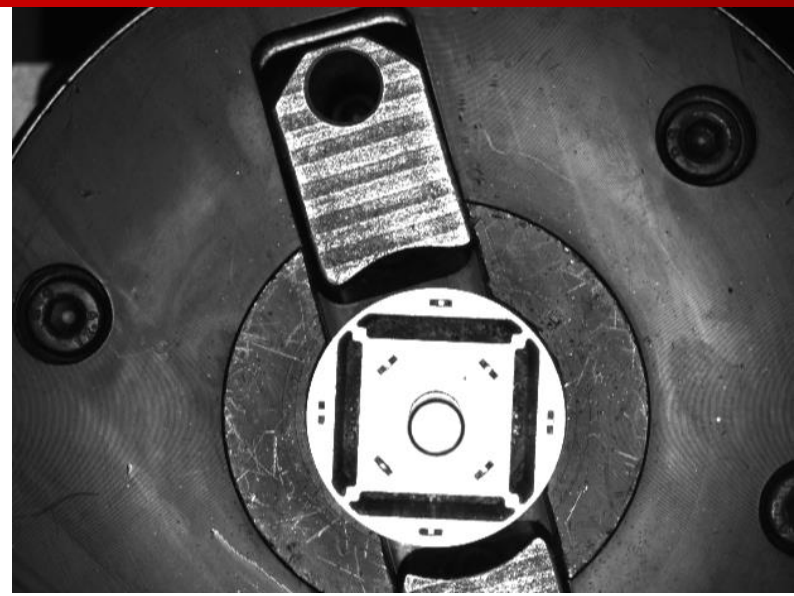
变量声明：

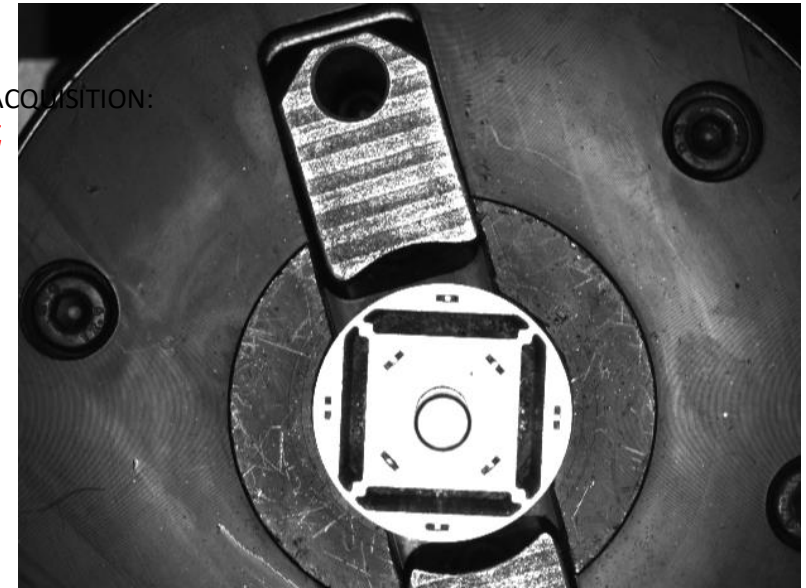
```
PROGRAM MAIN
VAR
hr : HRESULT;                          // 函数返回值
fbCamera: FB_VN_SimpleCameraControl;   // 视觉功能块
eState: ETcVnCameraState;              // TcVision 状态机
ipImageIn : ITcVnImage;                // 输入图像
ipImageInDisp : ITcVnDisplayableImage; // 输入图像显示窗口
END_VAR
```

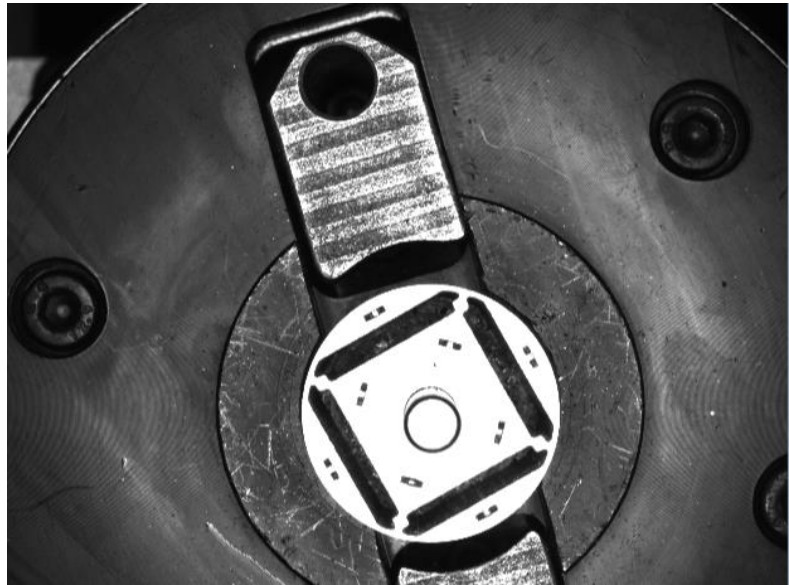
基础代码：

```
eState := fbCamera.GetState();
// 获取当前 TcVision 状态机
CASE eState OF
TCVN_CS_INITIAL,TCVN_CS_OPENING,TCVN_CS_OPENED,TCVN_CS_STARTACQUISITION:
hr := fbCamera.StartAcquisition();          // 切换 TcVision 状态至工作模式
TCVN_CS_ACQUIRING:
hr := fbCamera.GetCurrentImage(ipImageIn);  // 获取当前图片
IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN    // 判断是否正确获取图片
// 视觉算法部分
(*.....*)
END_IF
TCVN_CS_ERROR:
hr := fbCamera.Reset();                     // TcVision 报错复位
END_CASE
```





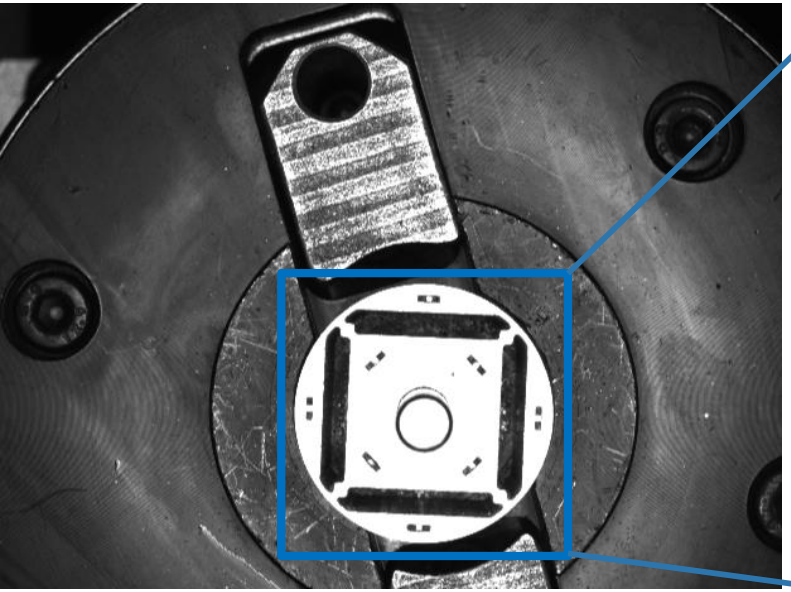




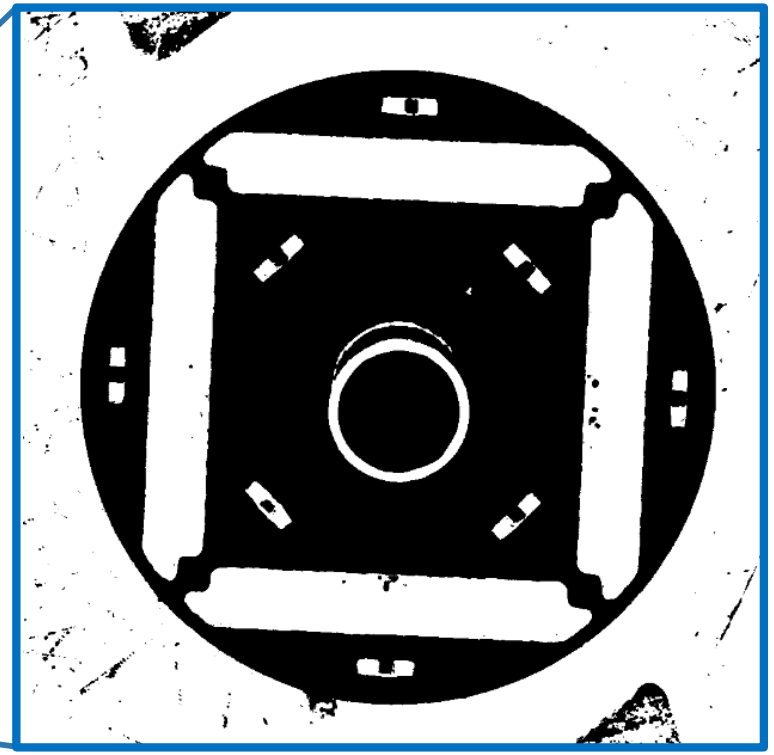
项目案例一：旋转角度检测

BECKHOFF

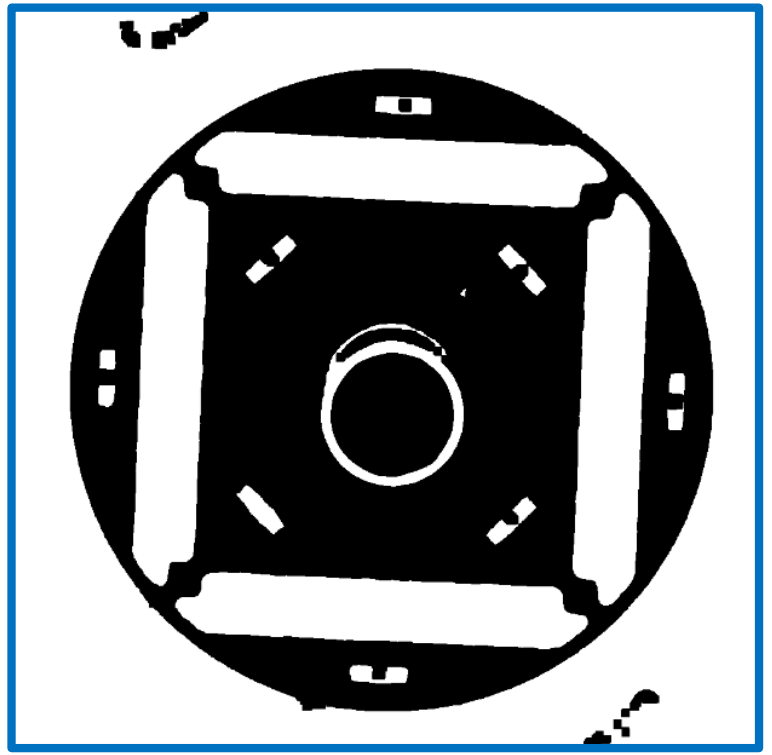
- 图像预处理



检测区域ROI选定
F_VN_CopyImageRegion



二值化
F_VN_Threshold



形态学处理（闭运算）
F_VN_CreateStructuringElement
F_VN_MorphologicalOperator

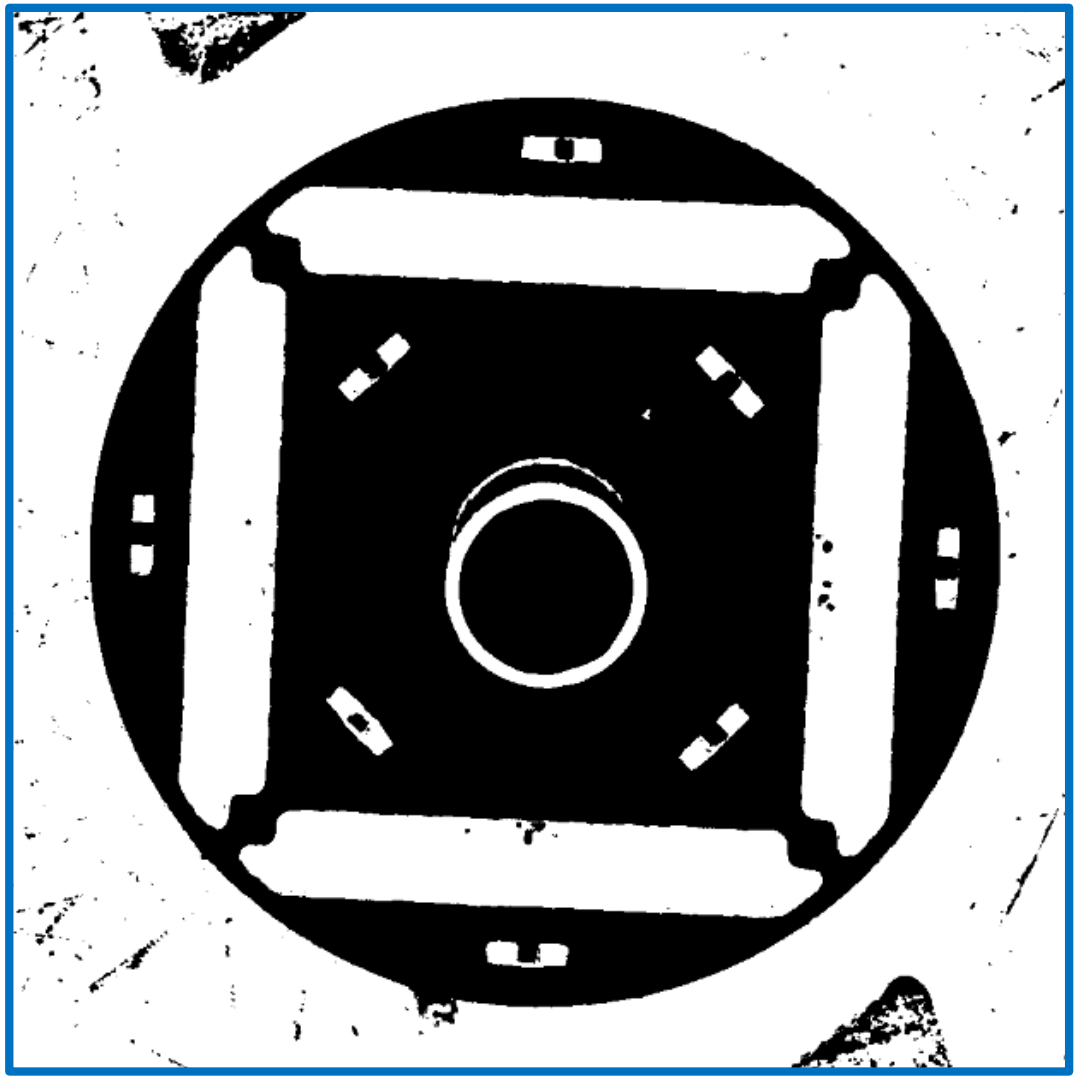
项目案例一：旋转角度检测

BECKHOFF

形态学处理：
图像闭运算(先腐蚀，后膨胀)

此案例适合闭运算

形态学处理：
图像开运算(先腐蚀，后膨胀)



项目案例一：旋转角度检测

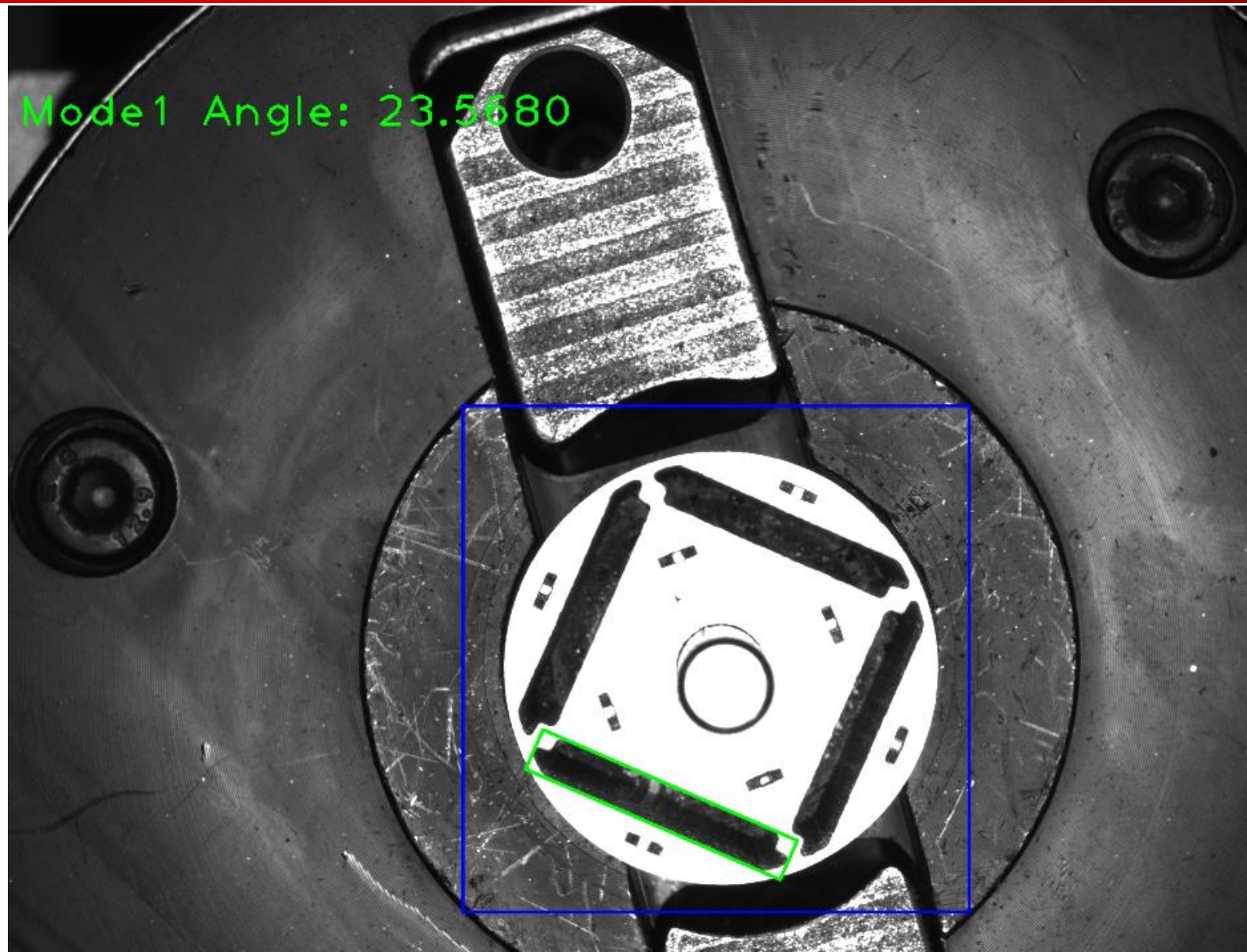
BECKHOFF

视觉算法模式一：找轮廓

轮廓提取
F_VN_FindContoursExp

轮廓筛选（面积，圆度）
F_VN_GetNumberOfElements
F_VN_GetAt_ITcVnContainer
F_VN_ContourArea
F_VN_ContourCircularity

外接矩形、获取角度
F_VN_EnclosingRectangle



项目案例一：旋转角度检测

BECKHOFF

视觉算法模式二：寻边

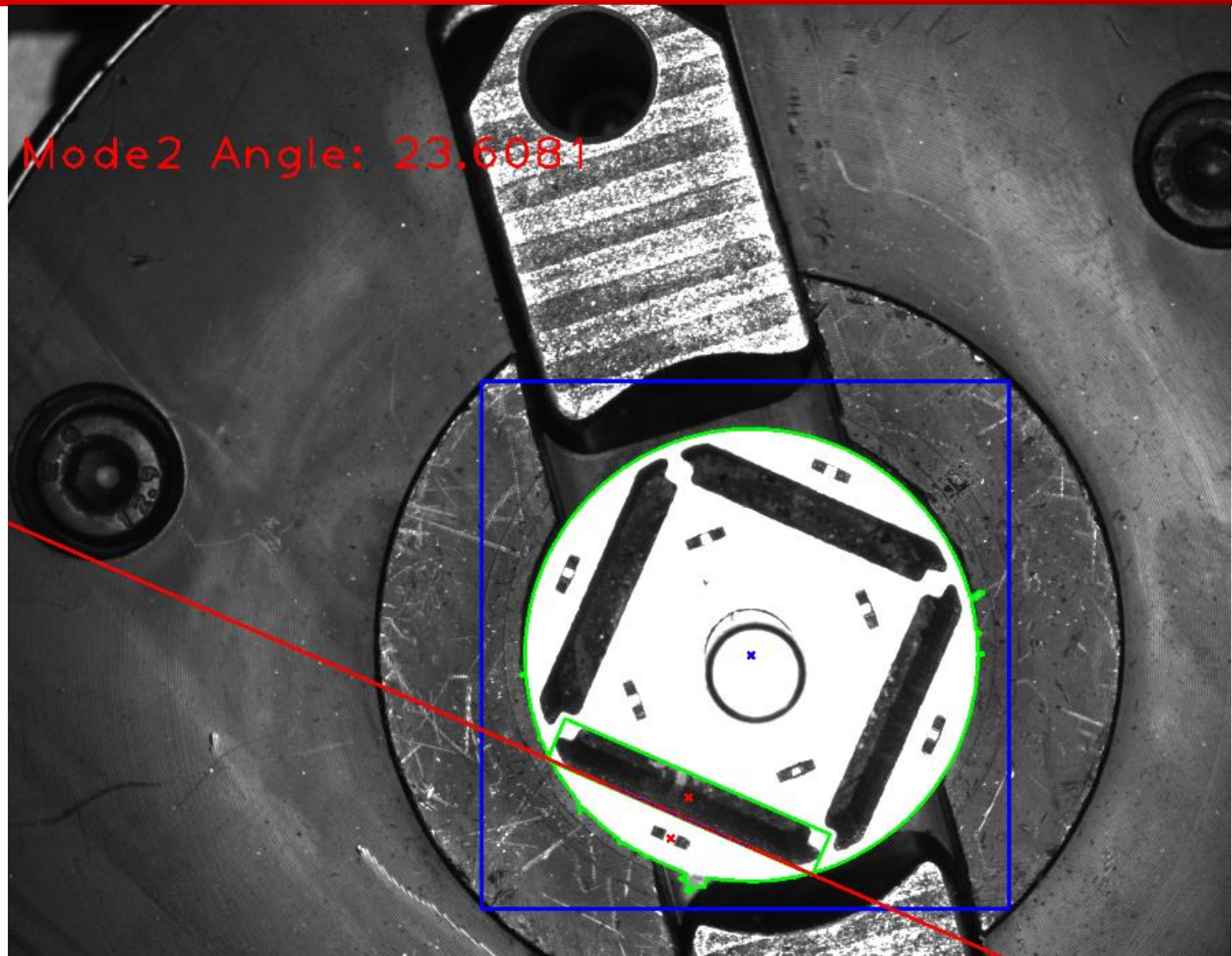
轮廓提取
F_VN_FindContoursExp

轮廓筛选（面积，圆度）
F_VN_GetNumberOfElements
F_VN_GetAt_ITcVnContainer
F_VN_ContourArea
F_VN_ContourCircularity

确定质心
F_VN_ContourCenterOfMass

寻找边界
F_VN_LocateEdgeExp

拟合直线、获取角度
F_VN_FitLine



项目案例一：旋转角度检测

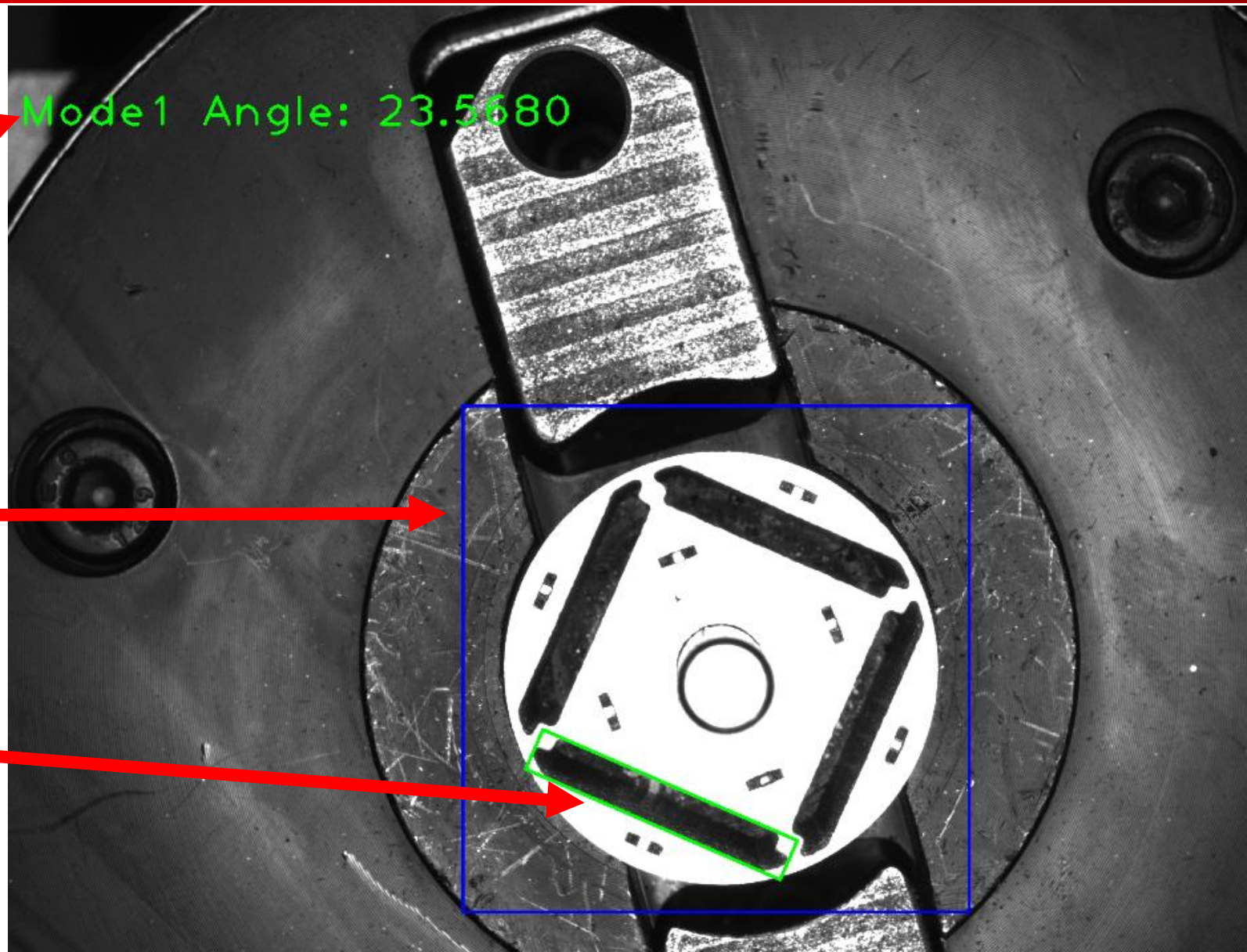
BECKHOFF

- 输出结果
  - 过程绘制

绘制角度结果：
F_VN_PutTextExp

绘制检测区域选定框：
F_VN_DrawRectangle_TcVnRectangle_UDINT

绘制外接矩形框：
F_VN_DrawRotatedRectangle



项目案例一：旋转角度检测

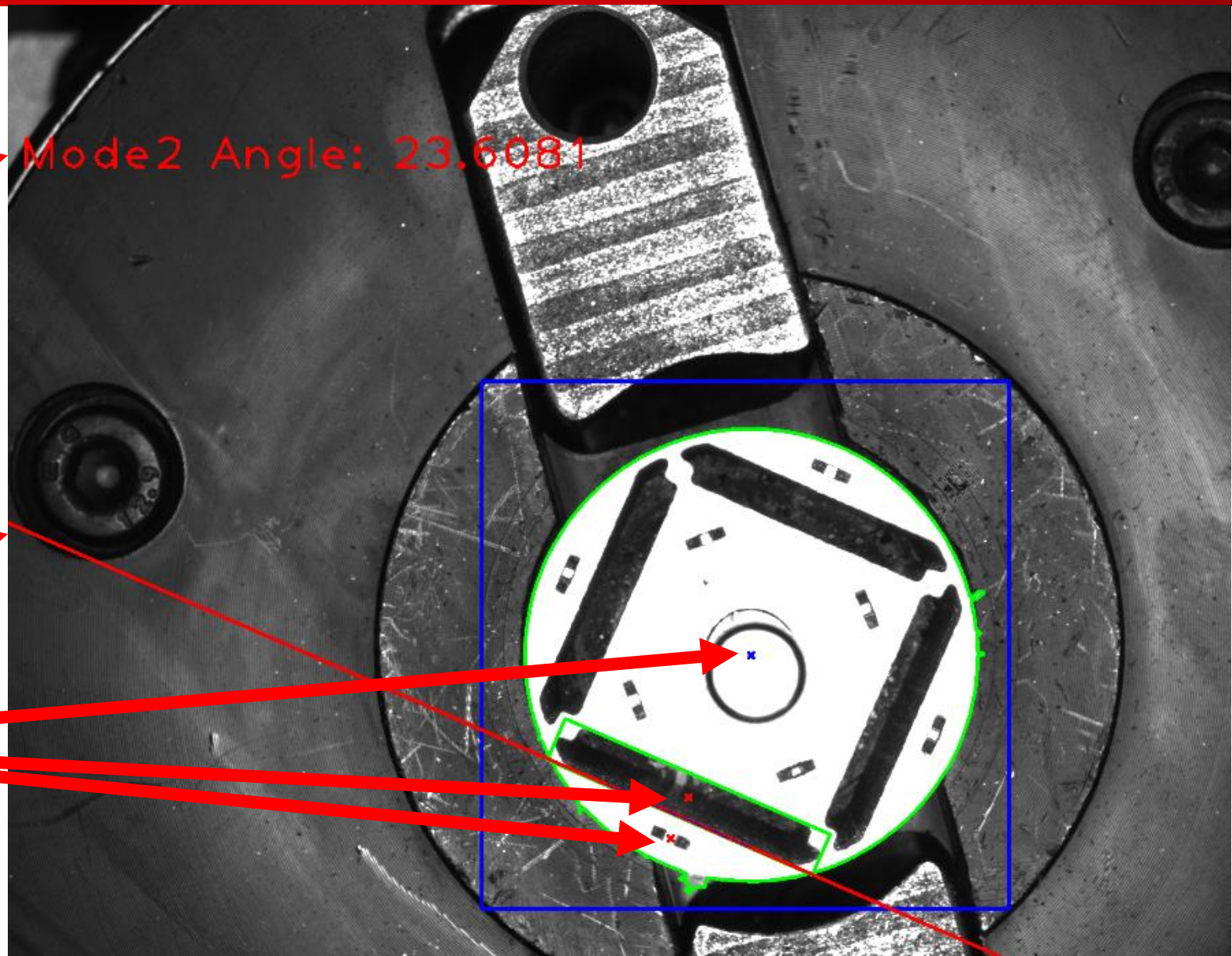
BECKHOFF

- 输出结果
  - 过程绘制

绘制角度结果：
F_VN_PutTextExp

绘制寻边拟合的直线：
F_VN_DrawLine_TcVnVector4_LREAL

绘制坐标点：
F_VN_DrawPointsExp

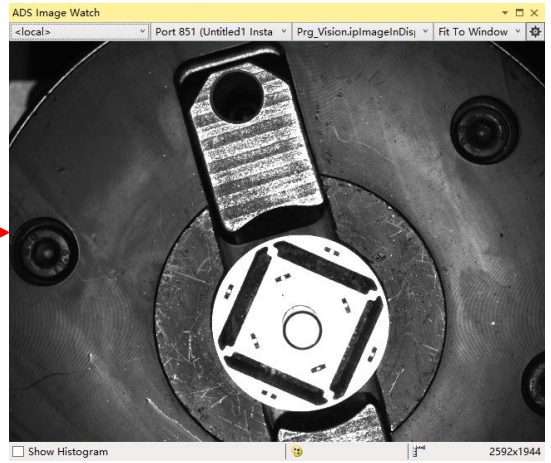


项目案例一：旋转角度检测

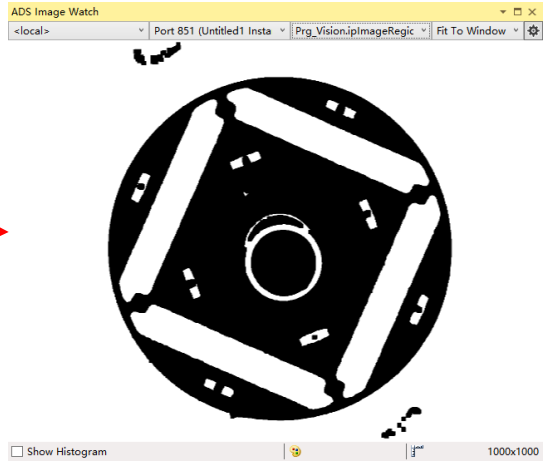
BECKHOFF

- 输出结果
  - 窗口输出

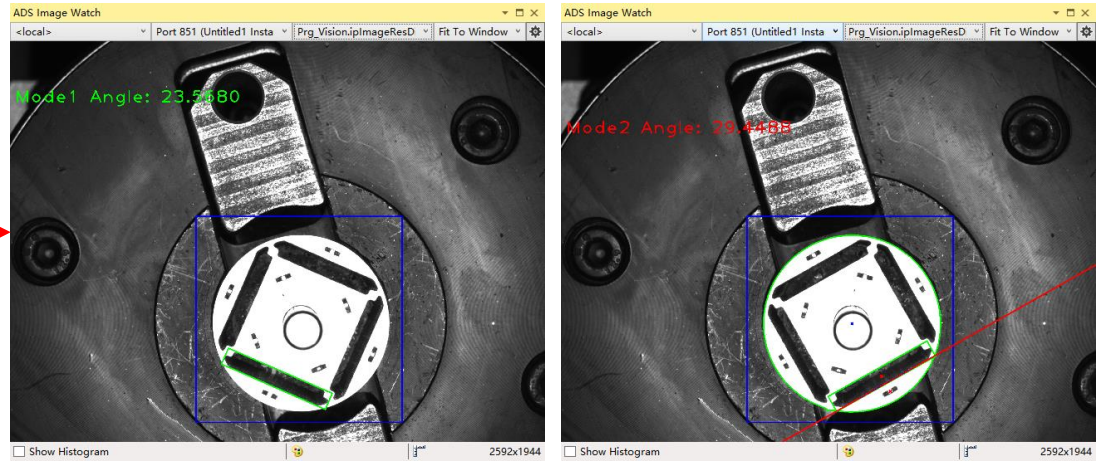
原始图像：
hr :=F_VN_TransformIntoDisplayableImage (ipImageIn,ipImageInDisp,S_OK);



预处理图像：
hr :=F_VN_TransformIntoDisplayableImage (ipImageRegion,ipImageRegionDisp,S_OK);



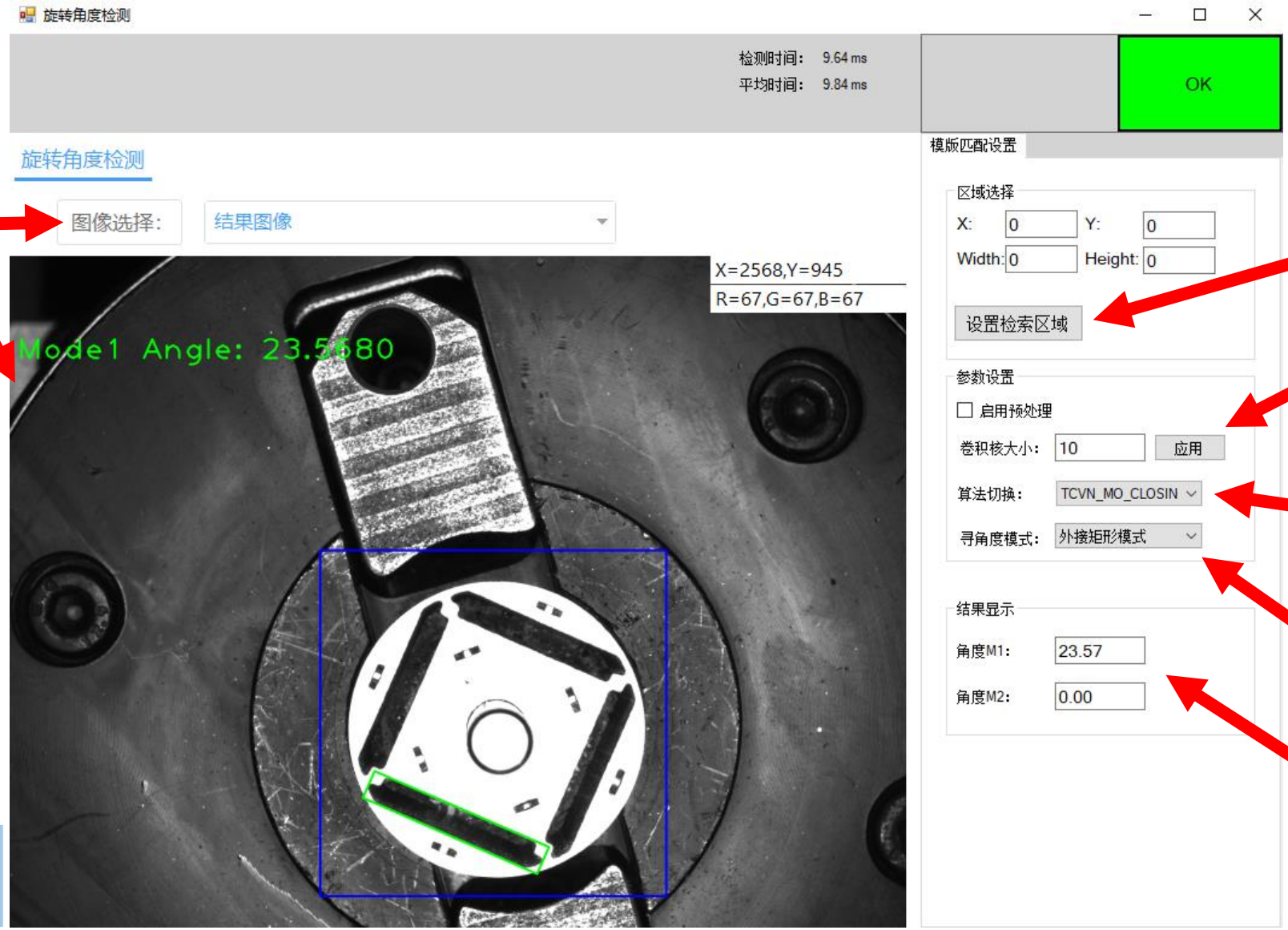
结果图像：
hr :=F_VN_TransformIntoDisplayableImage (ipImageRes,ipImageResDisp, S_OK);



项目案例一：旋转角度检测

BECKHOFF

C# 调试界面总览



图像窗口选择和输出

检测区域ROI选定

过滤卷积核大小

形态学算法切换

寻找角度模式切换

角度结果实时显示

区域选定都可以直接在C#中通过鼠标拖动选择

项目案例一：旋转角度检测

BECKHOFF

C# 调试界面预处理

检测区域窗口选择和输出

开启形态学预处理

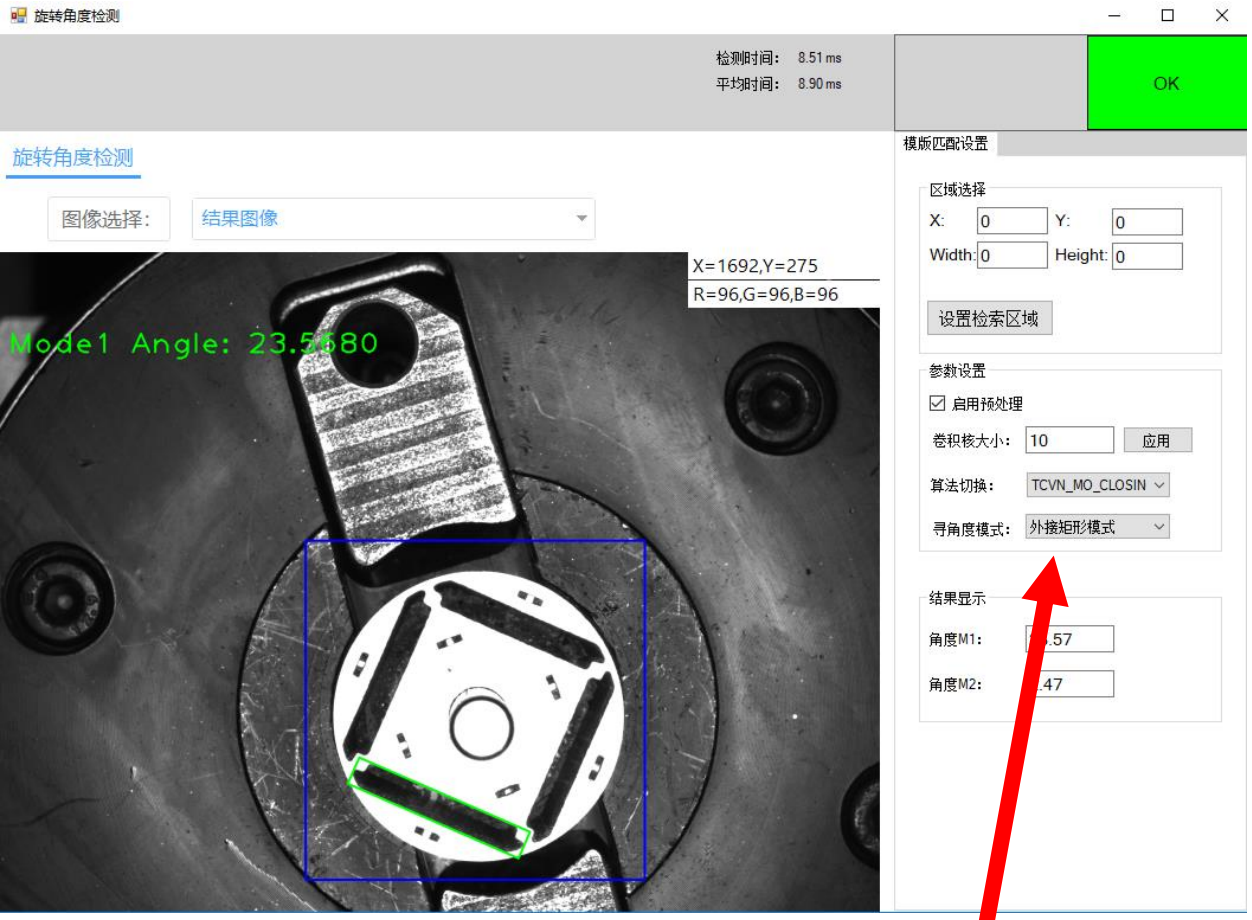
过滤卷积核大小

开闭算法切换

项目案例一：旋转角度检测

BECKHOFF

C# 调试界面模式切换



外界矩形（找轮廓）模式

寻边模式

总结：

- 关于打光方案还是很重要的，案例中没有做详细介绍，但尽量不要用算法弥补成像的缺陷
- 形态学处理的重要性，简单的利用ROI和开闭算法可以指定检测区域和过滤不必要的干扰，提高检测稳定性和效率
- 通过找轮廓 + 面积筛选是最简单有效的找到目标物的方法
- 利用最小外接矩形（F_VN_EnclosingRectangle）可以获取特定的参数，比如角度和长宽，同样的：
  - ✓ 最小外接圆（F_VN_EnclosingCircle）可以获取圆心坐标和半径
  - ✓ 最小外接三角形（F_VN_EnclosingTriangle）可以获取三个顶点坐标
- 通过两点寻边（F_VN_LocateEdgeExp）需要注意函数中几个参数的调整，以达到最稳定的边界定位：
  - ✓ eEdgeDirection：搜索的边缘方向，由亮到暗还是由暗到亮
  - ✓ fMinStrength：两点间最小灰度差寻找边缘
  - ✓ nSearchLines：搜索窗口的宽度（线数）
  - ✓ fSearchLineDist：窗口中搜索线之间的距离
  - ✓ nMaxThickness：搜索的边缘的最大厚度
- 此项目可以应用于哪些场景：角度检测、瑕疵/缺陷检测、寻边算平行度/夹角、寻边定位

项目案例二：模板匹配定位

BECKHOFF

- **需求：**

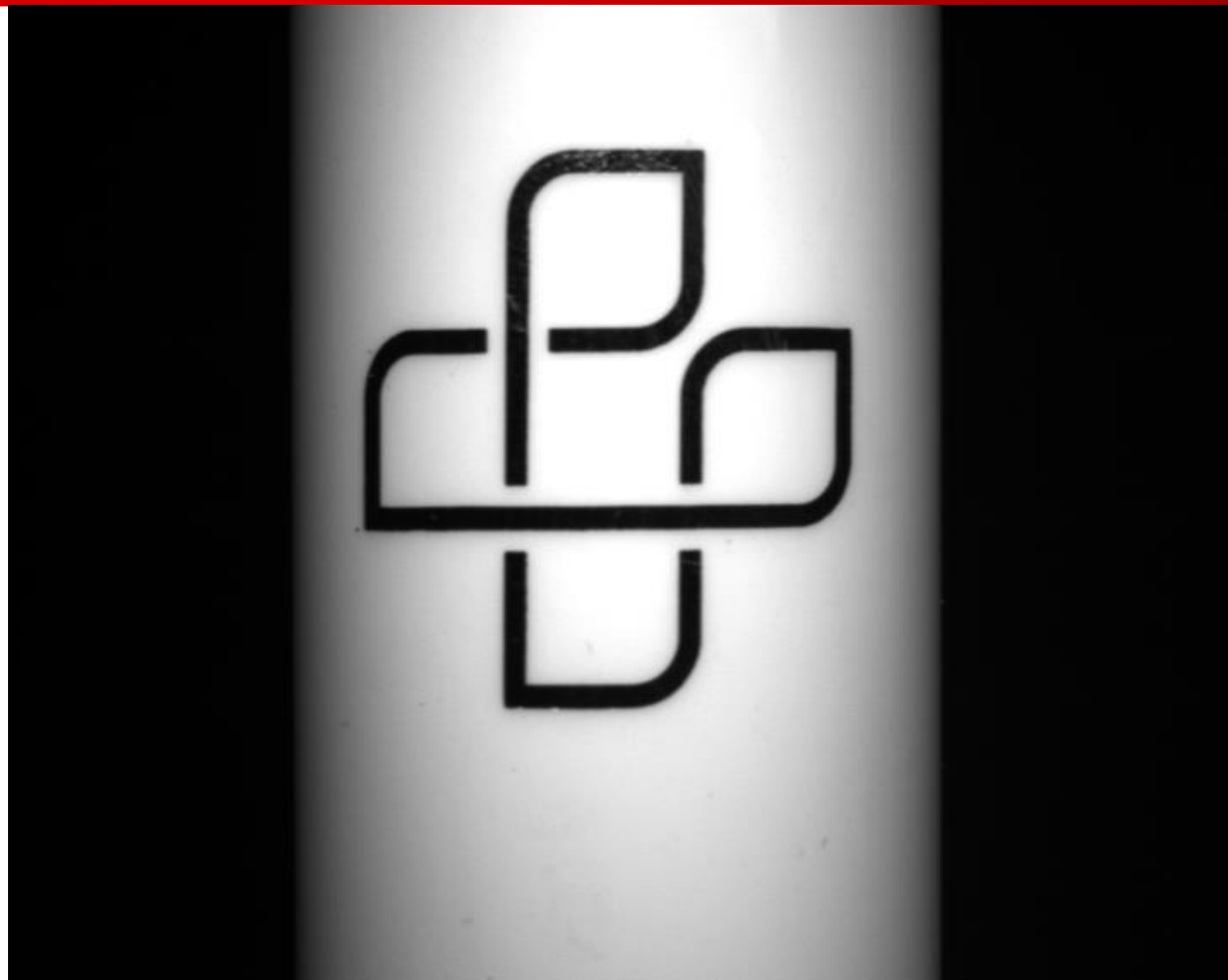
此项目是一个印刷套色项目，图片是一个瓶身图像，通过瓶身的任意特征定位其所处的位置，配合基准位置进行修正，给予后续套色提供准确位置信息。

- **实现方式：**

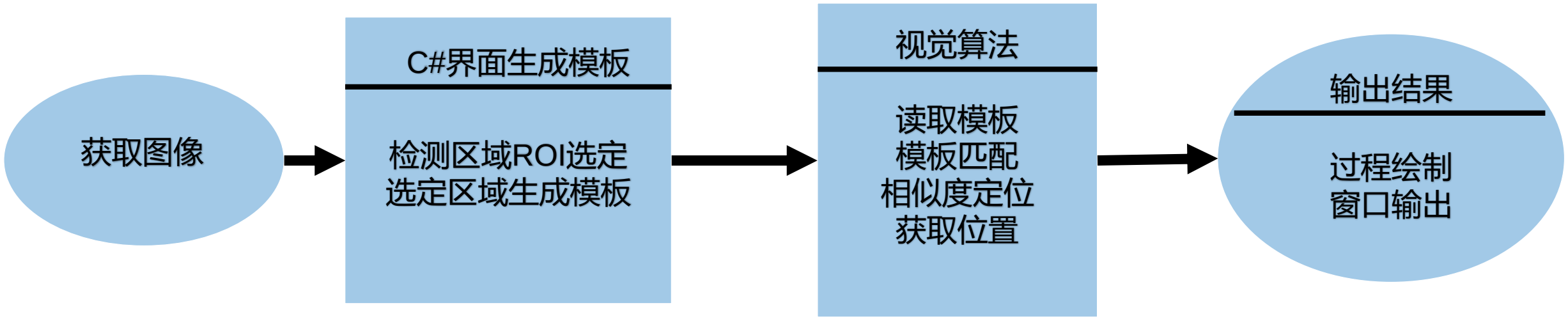
C# 调试界面生成模板
任意特征图像模板匹配

- **TwinCAT Vision 授权：**

TF7100 TC3 Vision Base
TF7200 TC3 Vision Matching 2D



- **实现过程：**


获取图像
C#界面生成模板
检测区域ROI选定
选定区域生成模板
视觉算法
读取模板
模板匹配
相似度定位
获取位置
输出结果
过程绘制
窗口输出

项目案例二：模板匹配定位

BECKHOFF

➢ 获取图像

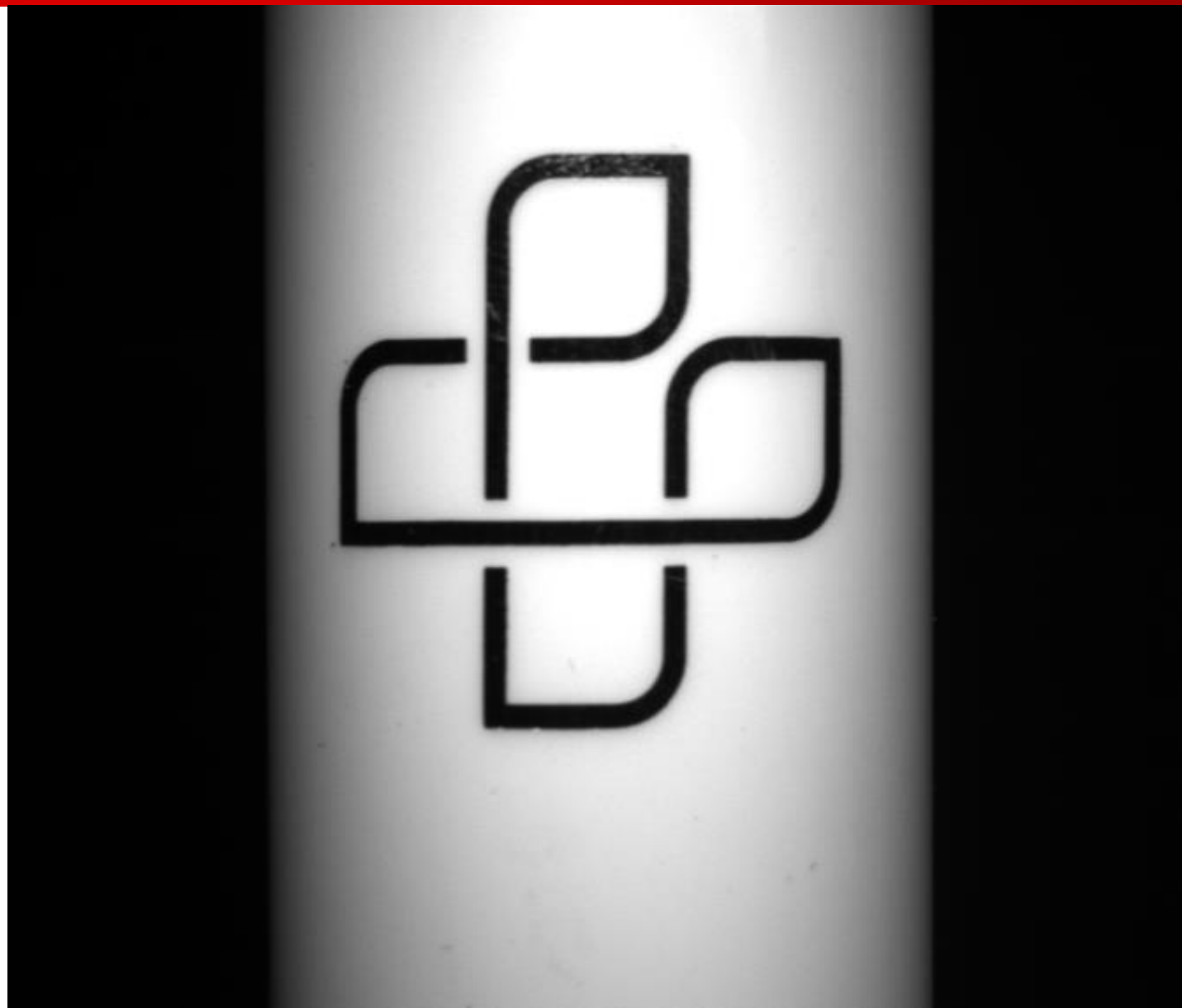
初始化程序代码

变量声明：

```
PROGRAM MAIN
VAR
hr : HRESULT;                          // 函数返回值
fbCamera: FB_VN_SimpleCameraControl;   // 视觉功能块
eState: ETcVnCameraState;              // TcVision 状态机
ipImageIn : ITcVnImage;                // 输入图像
ipImageInDisp : ITcVnDisplayableImage; // 输入图像显示窗口
END_VAR
```

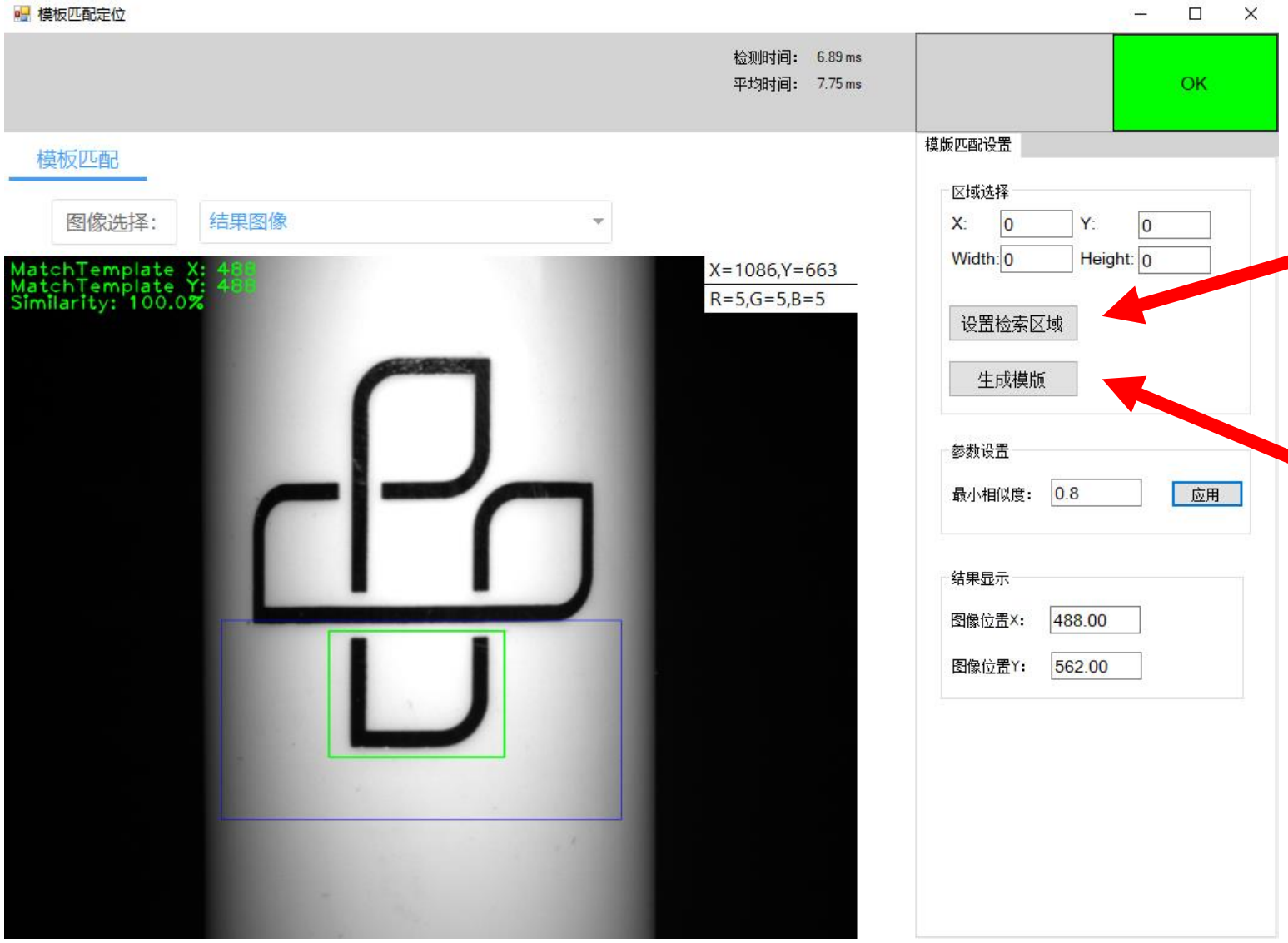
基础代码：

```
eState := fbCamera.GetState();
// 获取当前 TcVision 状态机
CASE eState OF
TCVN_CS_INITIAL,TCVN_CS_OPENING,TCVN_CS_OPENED,TCVN_CS_STARTACQUISITION:
hr := fbCamera.StartAcquisition();          // 切换 TcVision 状态至工作模式
TCVN_CS_ACQUIRING:
hr := fbCamera.GetCurrentImage(ipImageIn);  // 获取当前图片
IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN    // 判断是否正确获取图片
// 视觉算法部分
(*.....*)
END_IF
TCVN_CS_ERROR:
hr := fbCamera.Reset();                     // TcVision 报错复位
END_CASE
```



项目案例二：模板匹配定位

BECKHOFF

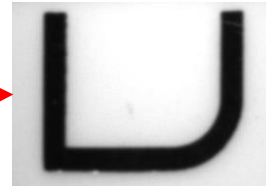


- C# 调试界面生成模板

检测区域ROI选定
F_VN_CopyImageRegion

选定区域生成模板
F_VN_CopyImageRegion
FB_VN_WriteImage

C:\Ref\Ref_1.bmp



所有区域选定都可以直接在C#中通过鼠标拖动选择

项目案例二：模板匹配定位

BECKHOFF

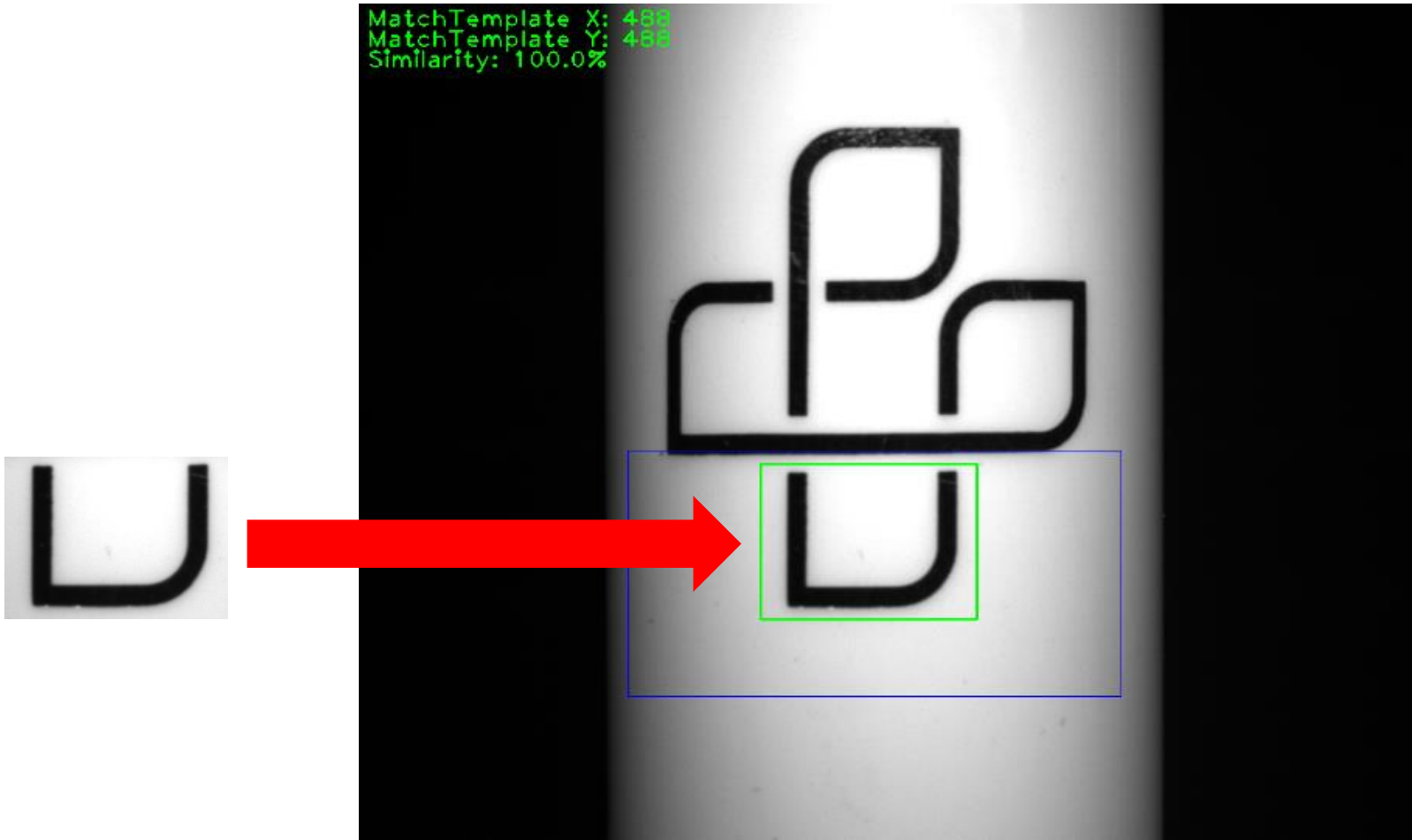
视觉算法

读取模板
FB_VN_ReadImage

模板匹配
F_VN_MatchTemplateExp

相似度定位
F_VN_MaxPixelValue. aMaxValue

获取位置
F_VN_MaxPixelValue. aMaxPos1



项目案例二：模板匹配定位

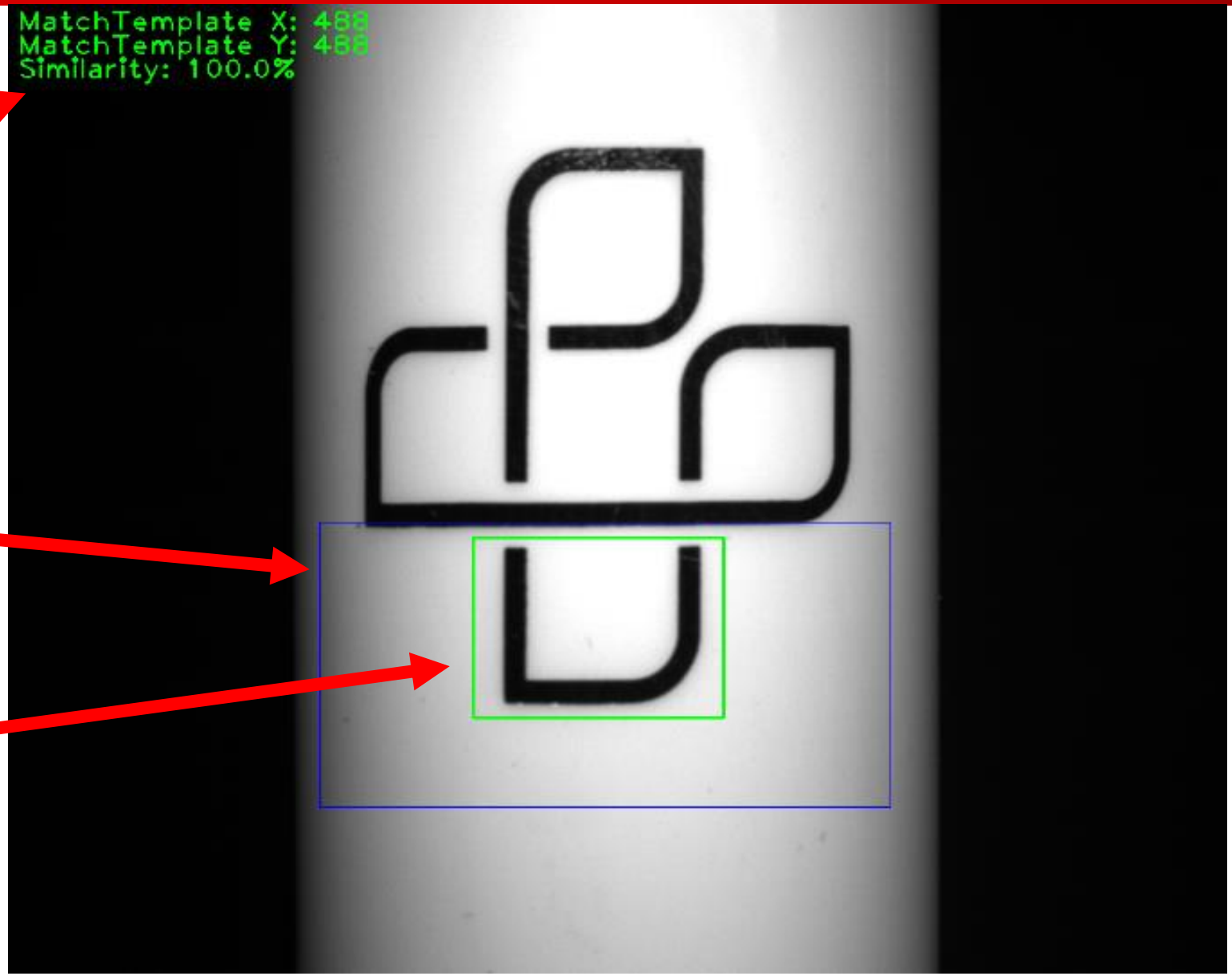
BECKHOFF

- 输出结果
  - 过程绘制

绘制位置和相似度结果：
F_VN_PutTextExp

绘制检测区域选定框：
F_VN_DrawRectangle_TcVnRectangle_UDINT

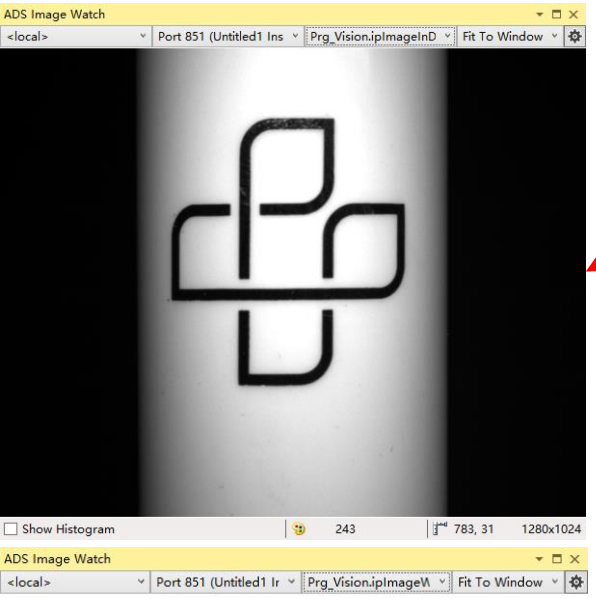
绘制匹配成功的模板框：
F_VN_DrawRotatedRectangle



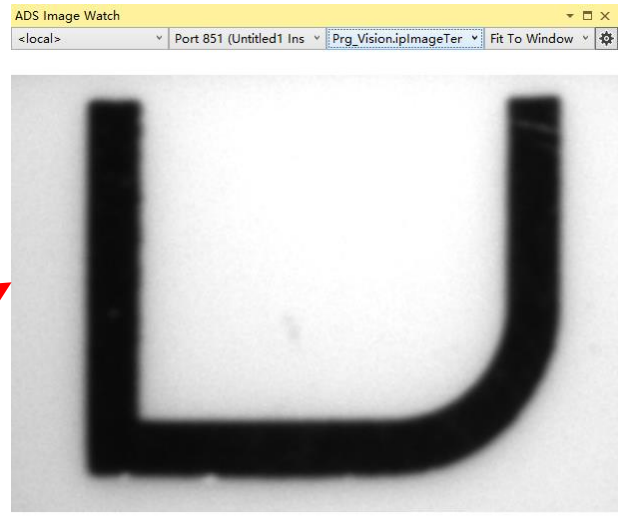
项目案例二：模板匹配定位

BECKHOFF

- 输出结果
  - 窗口输出



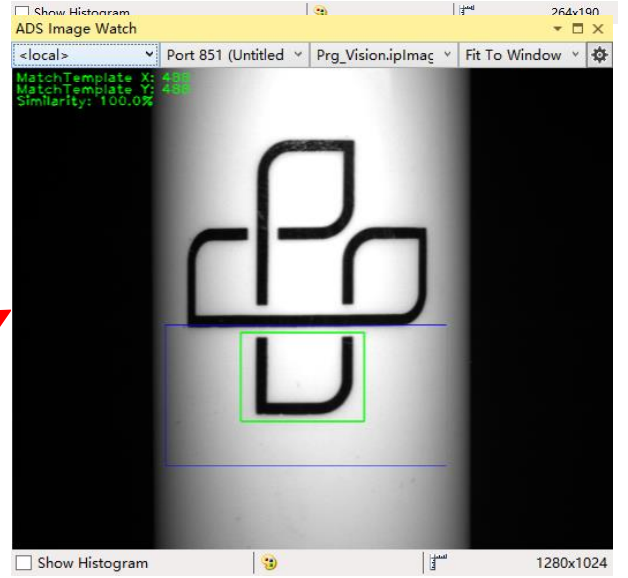
原始图像输出窗口
hr :=F_VN_TransformIntoDisplayableImage (ipImageIn,ipImageInDisp, S_OK);



模板图像输出窗口(无需释放内存)
hr :=F_VN_CopyIntoDisplayableImage (ipImageTemplate1 , ipImageTemplate1Disp , S_OK);

检测区域输出窗口
hr :=F_VN_TransformIntoDisplayableImage (ipImageWork,ipImageWorkDisp , S_OK);

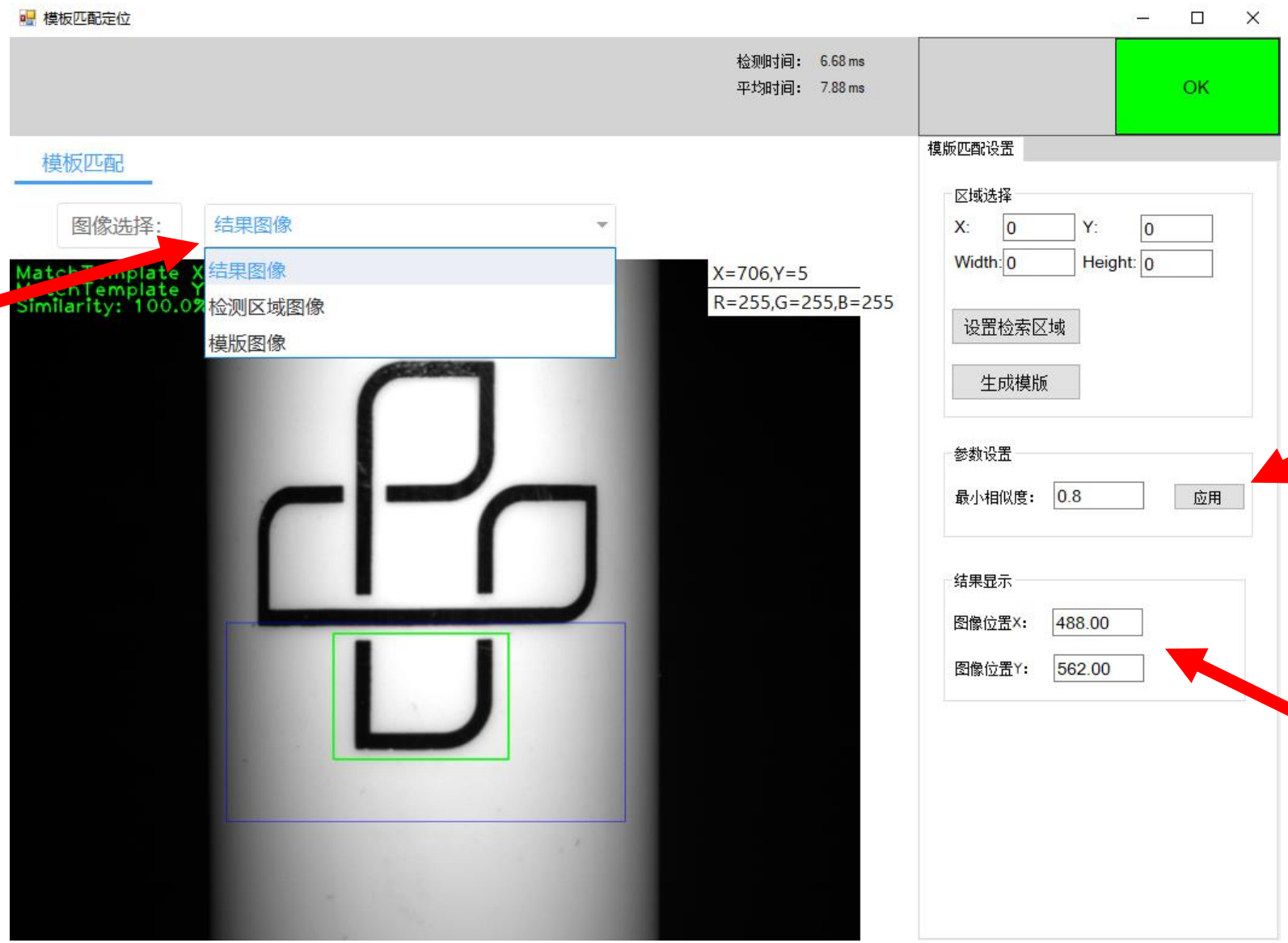
结果信息输出窗口
hr :=F_VN_TransformIntoDisplayableImage (ipImageRes,ipImageResDisp, S_OK);



项目案例二：模板匹配定位

BECKHOFF

C# 调试界面其他信息和操作



输出图像选择

相似度参数修改

模板匹配位置结果

总结：

- 模板匹配定位可以支持匹配目标的移动和适当的伸缩，但不支持模板旋转匹配
- 搭配C# 调试界面可以很方便的生成模板和定位检测区域，默认路径（可修改）是C:\Ref\
- 模板匹配相是像素点的逐一比较，因此相比较找轮廓和寻边耗时较长，不建议全画幅匹配，利用ROI缩小检测区域，尽可能提高检测效率
- 此案例可以应用于哪些场景：
  - ✓ 所有非旋转的特征匹配定位和有无、例如：图案匹配、文字匹配、标识匹配、轮廓匹配等
  - ✓ 适用行业：印刷行业、包装行业、3C行业

Thank you!