CKVISION SDK 使用说明

Ver. 5.3.0.1

CKVision SDK

1 Bin

1.1 DLL 库文件

CKBase.dll 基础库

CKBlob.dll 斑点分析

CKCalibration.dll 标定与校准

CKColor.dll 颜色识别

CKContour.dll 轮廓提取

CKGDI.dll 图形显示

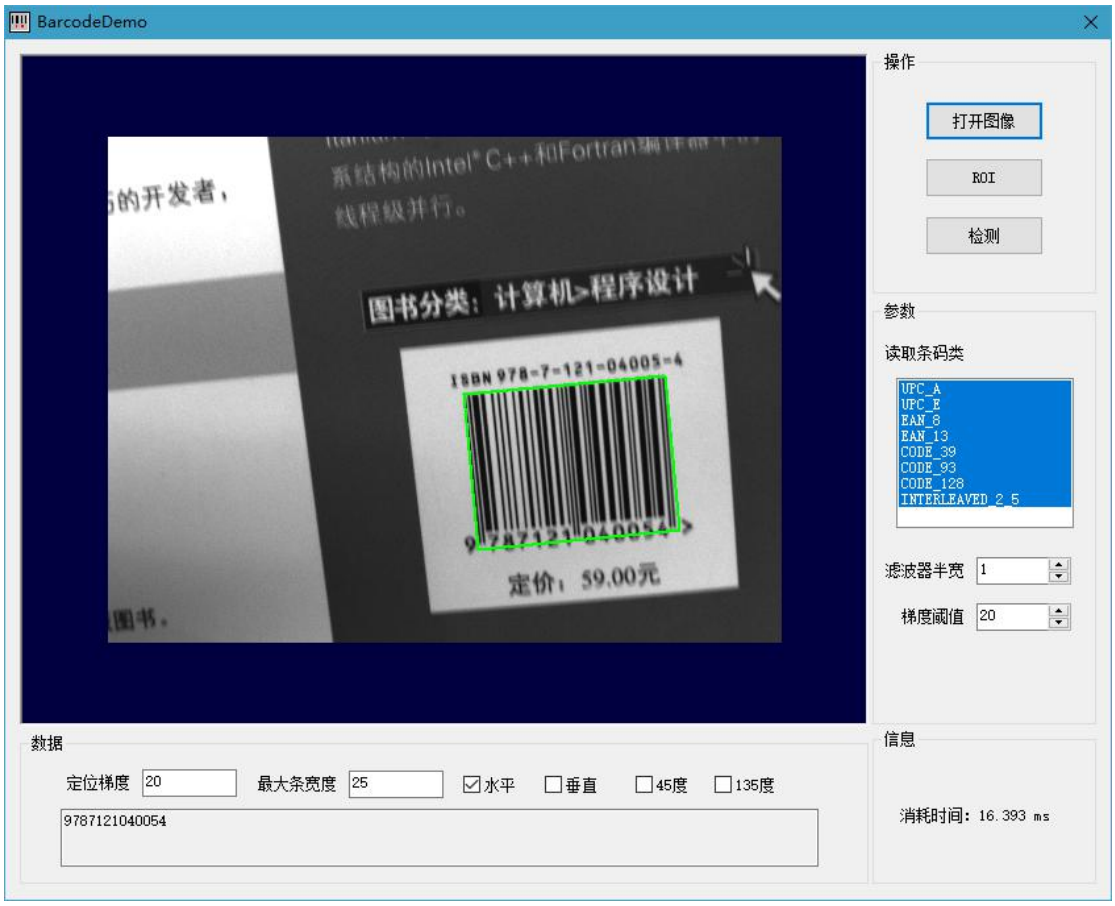
CKLocate.dll 形状匹配（识别定位）

CKMeasure.dll 测量（点、线、圆）

CKReader.dll 条码识别（一维码、二维码）

1.2 功能实例 exe 执行文件

BarcodeDemo.exe 一维码检测



打开图像：打开一张 8 位 bmp 的灰度图像。

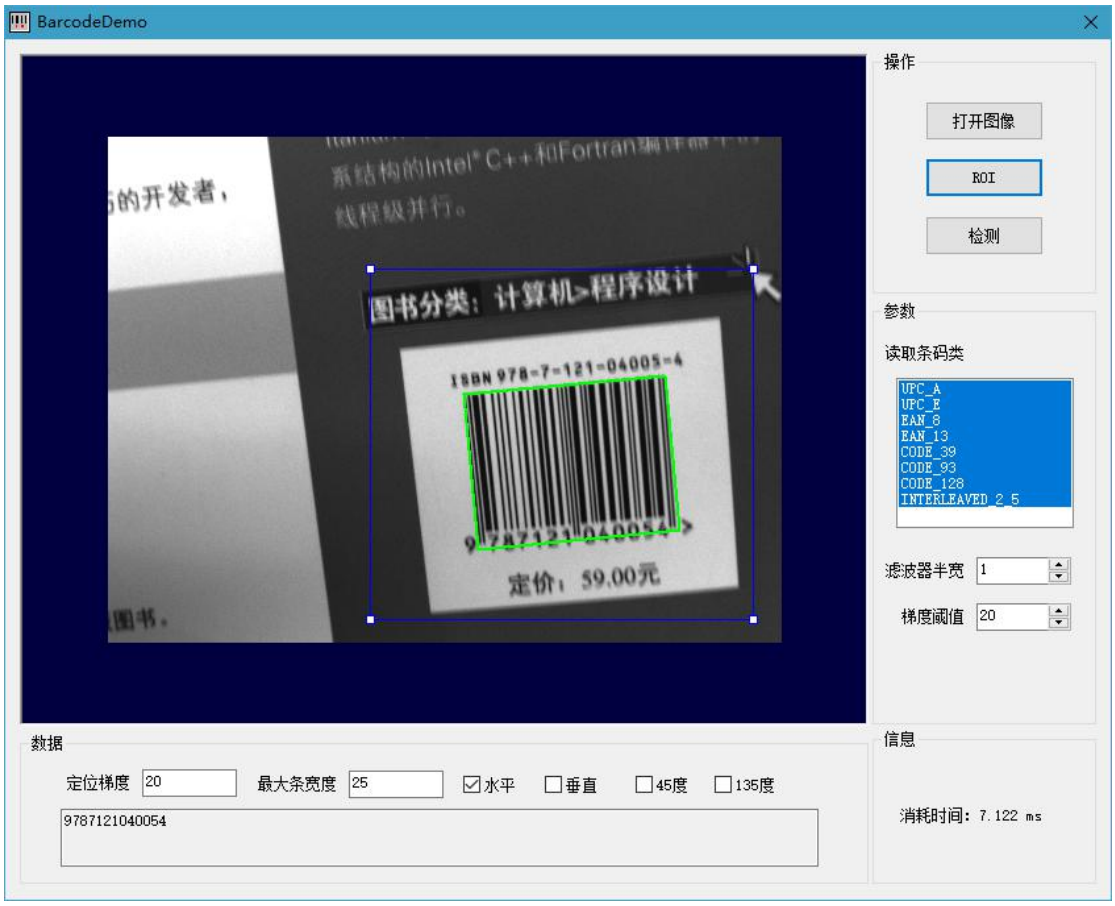
ROI：检测区域，指定区域内进行检测，可减少处理的时间。

检测：自动定位识别条码所在的位置并读取出条码的值。

条码类型：选择当前需要读取的条码类型，可以同时选择多个（自动识别类型）。

滤波器半宽：用于增强边缘提取功能，去除图像上的噪音干扰；最小值为 1，当边缘模糊不清晰或有噪音干扰时可以增大滤波半宽值，这样可以使得检测结果更加稳定，但如果边缘和边缘之间挨得太近（距离小于滤波半宽值）时反而会影响边缘位置的精度甚至丢失边缘，所有要根据实际情况来设置。

梯度阈值：取值范围 0 到 255，只有梯度值大于该值的边缘点才被检测到，梯度值是度量图像边缘的清晰度或对比度。



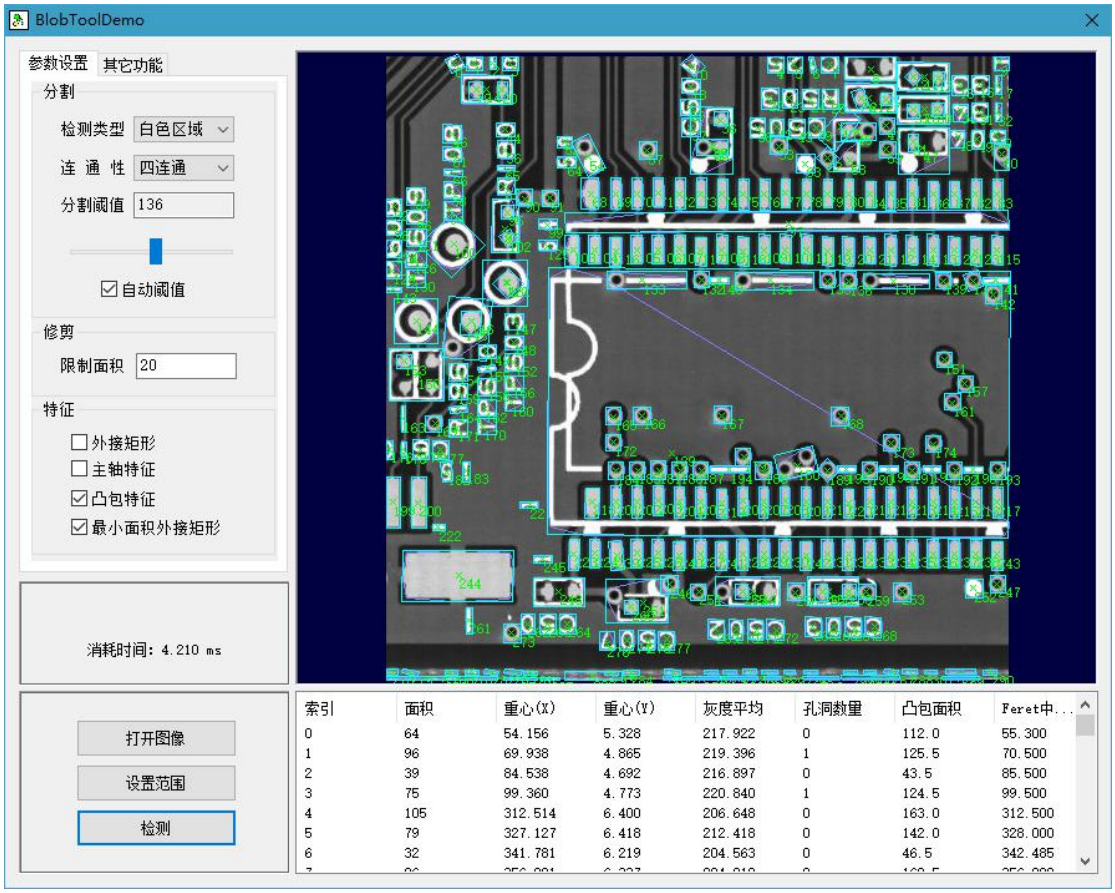
定位梯度：
梯度表示边缘的强度（清晰度），取值范围 0~255，当梯度大于该值的边缘点才会被识别。
最大调宽度：条码的黑条或白条可能出现的最大宽度值。

扫描方向：

水平，仅识别水平方向大约±30 度范围内的条码；
垂直，仅识别垂直方向大约±30 度范围内的条码；
45 度，仅识别 45° 左右范围的条码；
135 度，仅识别 135° 左右范围的条码。

BlobToolDemo.exe 斑点分析

检测图像中目标的数量和几何特征（面积、位置、方位、长轴和短轴），目标对象的定义为二值图像中黑色或白色像素连通区域，该功能要求图像背景均匀并亮度和目标的亮度区别比较明显。



分割阈值：设置二值图像的分割阈值，当像素灰度值大于等于该值为白色，否则为黑色。

自动计算阈值：软件将根据直方图分布自动计算出分割阈值。

二值图像预览：为了方便调整阈值，以二值化效果显示当前图像。

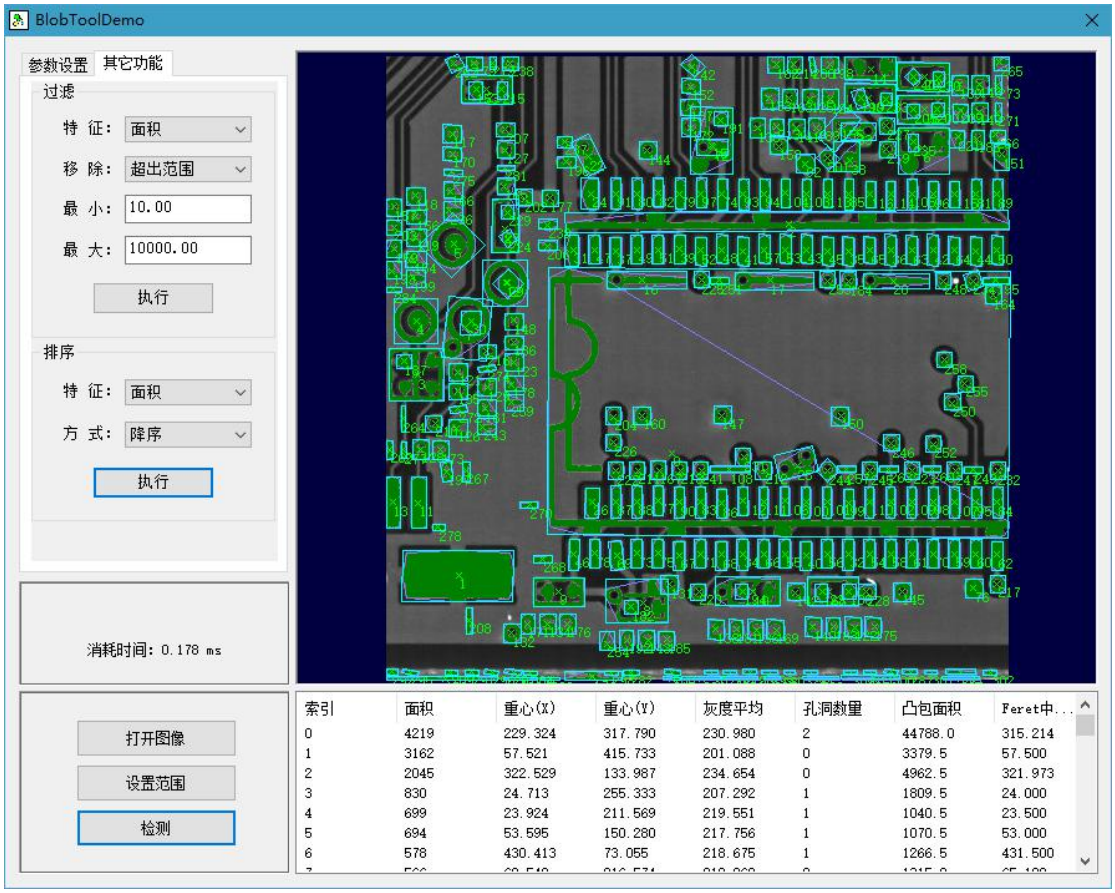
检测类型：可以设置当前需要检测的目标为黑色像素或白色像素区域。
连通性：算法中判定为连通的方法，4 连通表示只在上、下、左和右 4 个方向上相邻才被判定为互相连通，8 连通则在左上、右上、左下和右下方向上也算是互相连通。

限定面积：当目标面积（像素数量）小于该值时会被删除。

主轴椭圆特征：主轴椭圆为以区域重心为中心拟合的椭圆。

凸包特征：分析计算凸包面积。

最小外接矩形特征：计算连通区域的最小面积外接矩形。



过滤：可以设置目标每种特征的最小值和最大值，当某个目标的特征不在该范围值只能则会被移除，列表中被勾选的项目表示使用该特征的过滤功能。

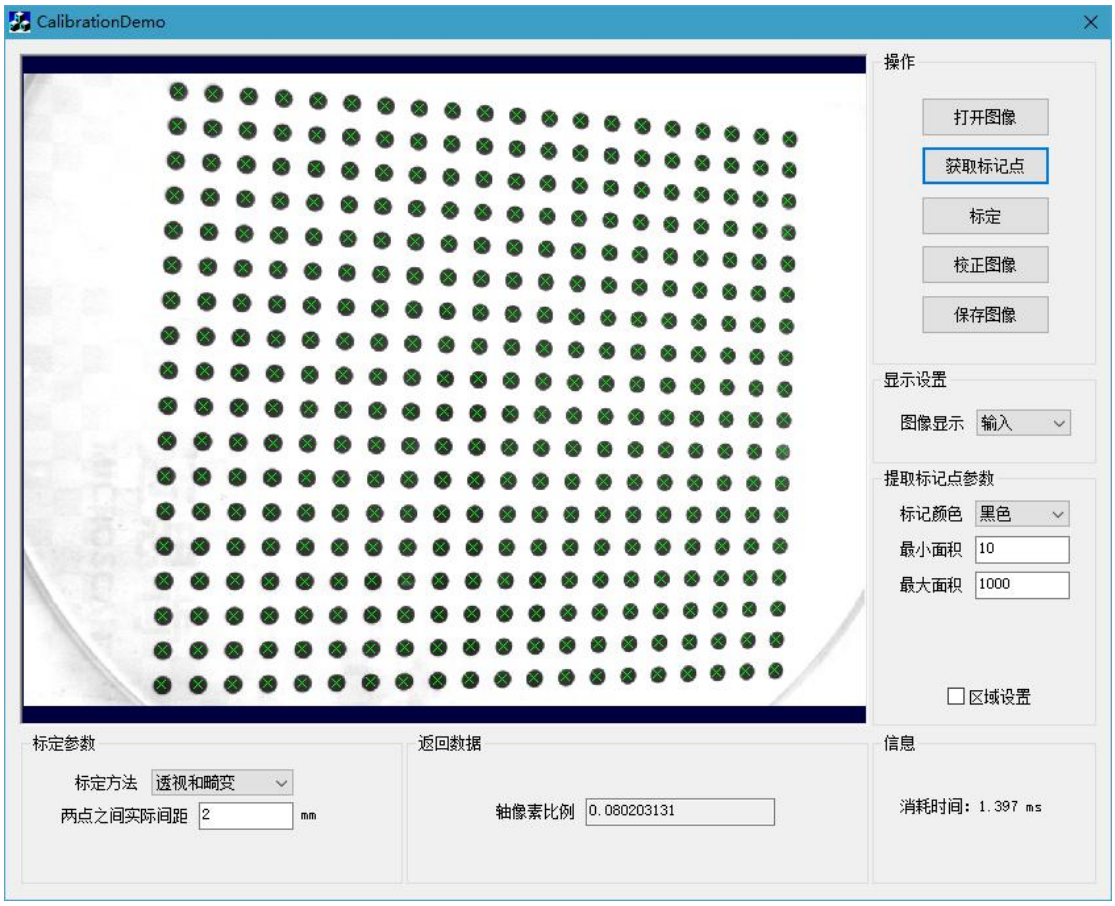
特征：按照指定特征将所有目标对象进行排序。

方式：排序方式，可以选择升序或降序。

激活排序：勾选表示使用排序功能。

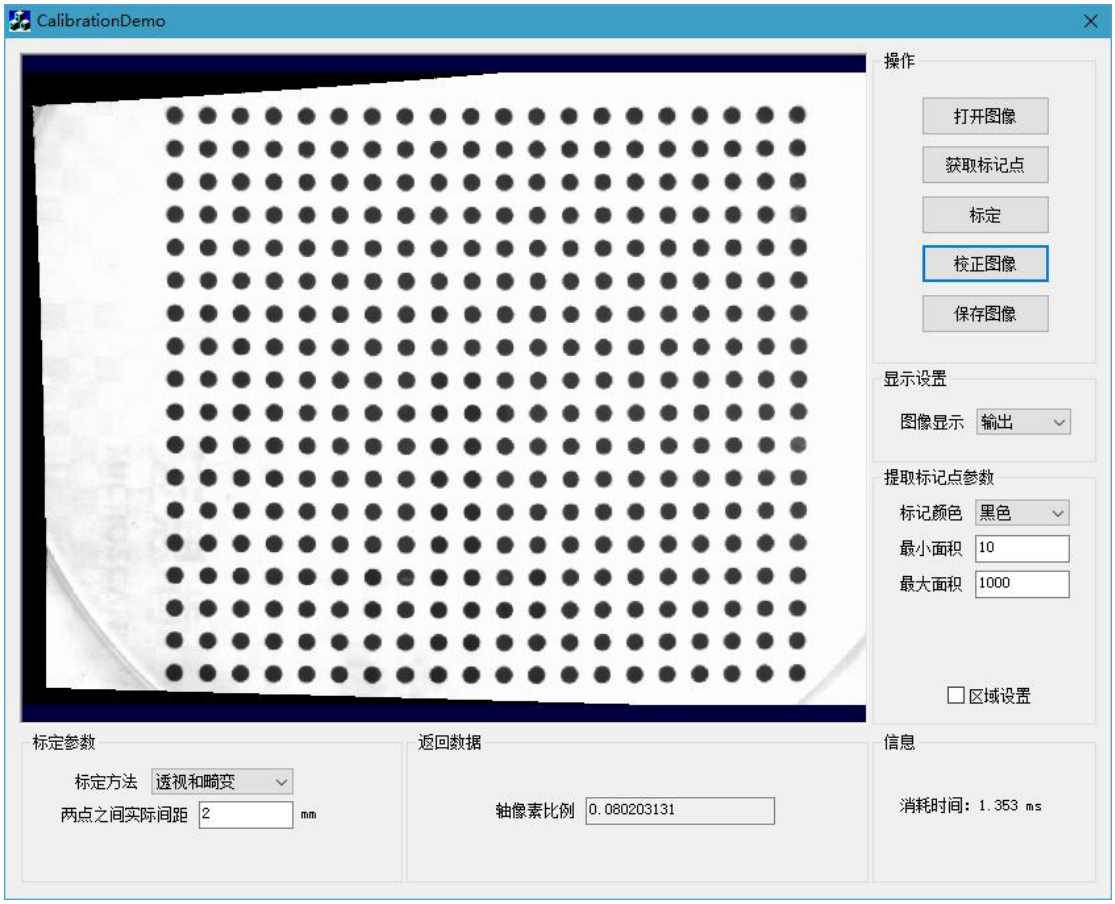
CalibrationDemo.exe 标定校准

通过拍摄圆点矩阵标定板，自动计算出图像矫正系数，并将倾斜拍摄的图像矫正成平面，并输出矫正后的图像。

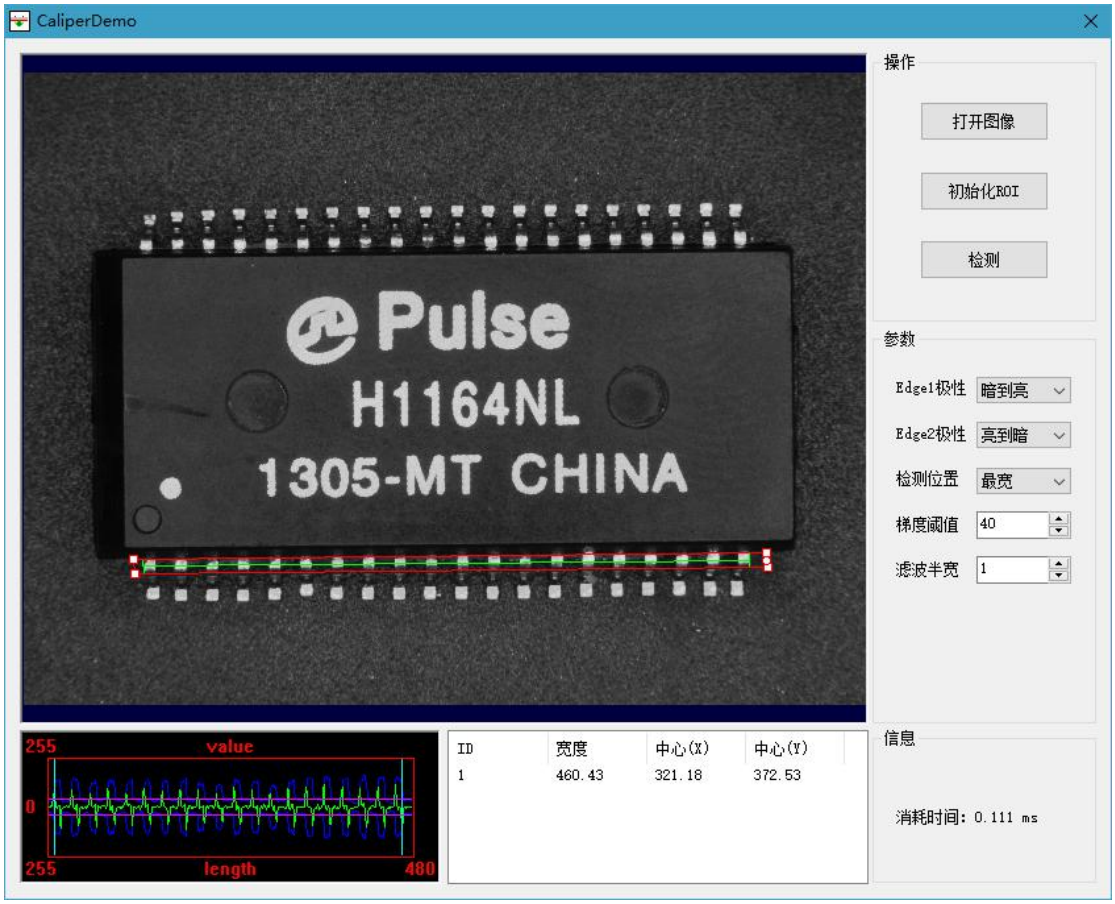


输入图像：链接需要进行校正的图像。
标记颜色：进行标定的圆点的类型为黑色或者白色。
设置区域：设置需要进行校正的图像区域。
最小面积：图像上圆点像素面积。
最大面积：图像上圆点像素面积。
标定方法：有透视与透视和畸变(径向畸变)两种校准类型。
两点之间的距离：圆点与圆点之间的距离，以 mm 为单位。
标定：获取标定结果。

校正图像：按标定方法校准图像。


CalibrationDemo
操作
打开图像
获取标记点
标定
校正图像
保存图像
显示设置
图像显示
输出
提取标记点参数
标记颜色
黑色
最小面积
10
最大面积
1000
区域设置
标定参数
标定方法
透视和畸变
两点之间实际间距
2
mm
返回数据
轴像素比例
0.080203131
信息
消耗时间：1.353 ms

CaliperDemo.exe 卡尺测量、间距检测



边缘极性：有亮到暗、暗到亮和任意 3 种模式；

亮到暗表示从亮度高过度到亮度低的边缘；

暗到亮表示从亮度低过度到亮度高的边缘；

任意模式则亮到暗和暗到亮的边缘都检测。

检测位置：有起始、最后、最宽和全部 4 种模式；

起始表示只检测最靠近扫描起始位置的 1 个间距；

最后表示只检测离扫描起始位置最远的 1 个间距；

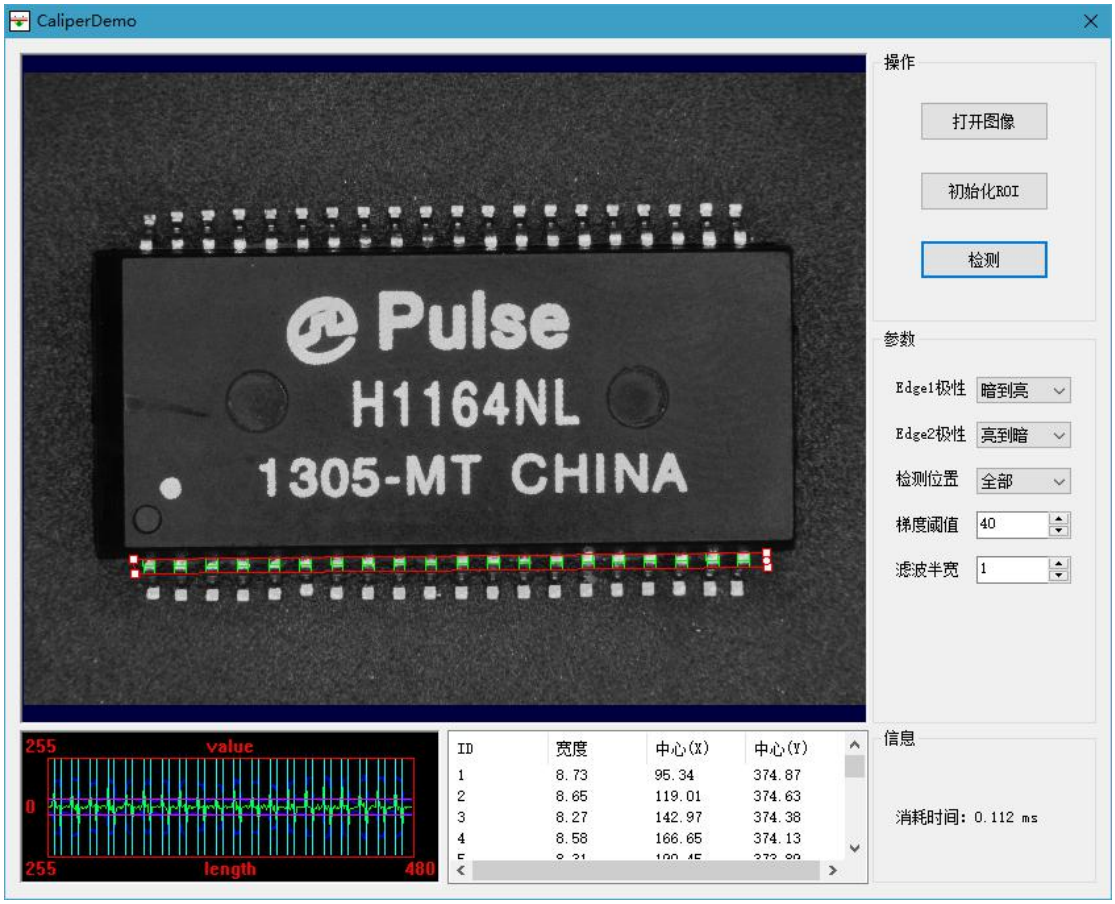
最宽表示只检测扫描范围内距离最远的 1 个间距，

全部表示检测扫描范围所有间距。

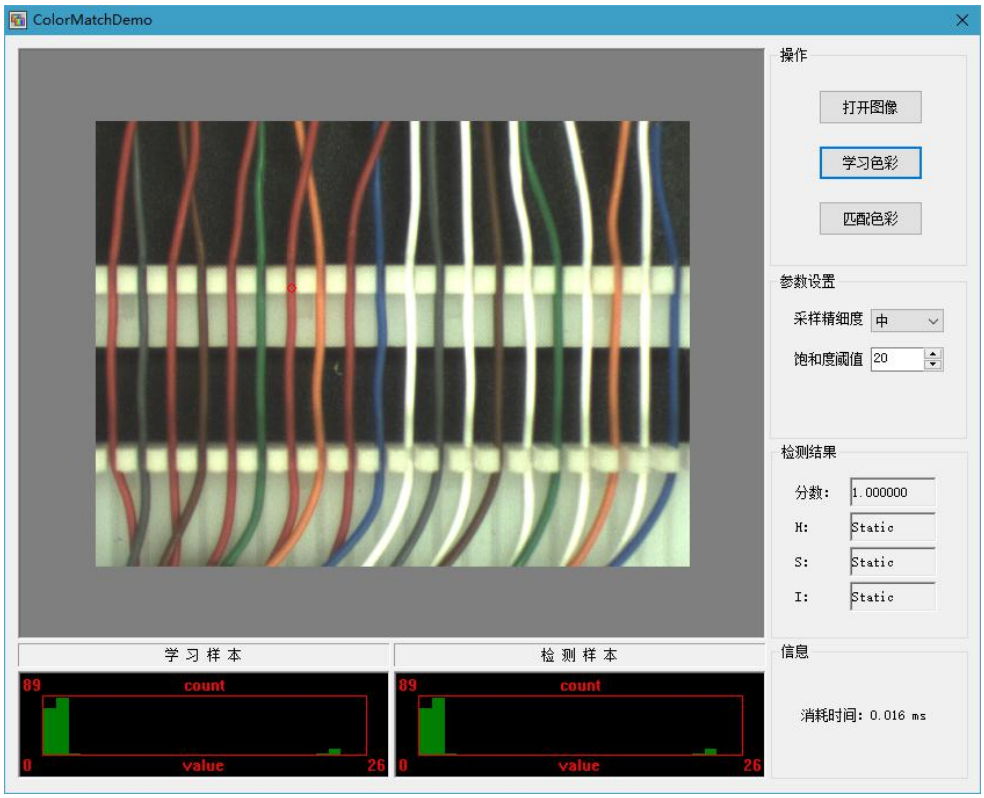
梯度阈值：梯度表示边缘的强度（清晰度），取值范围 0~255，只有梯度值大于该值的边缘点才会被检测到。

滤波半宽：用于增强边缘和抑制噪音干扰，最小值为 1，当边缘模糊不清晰或有噪音干扰时可以增大滤波半宽值，这样可以使得检测结果更加稳定，但如果边缘和边缘之间挨

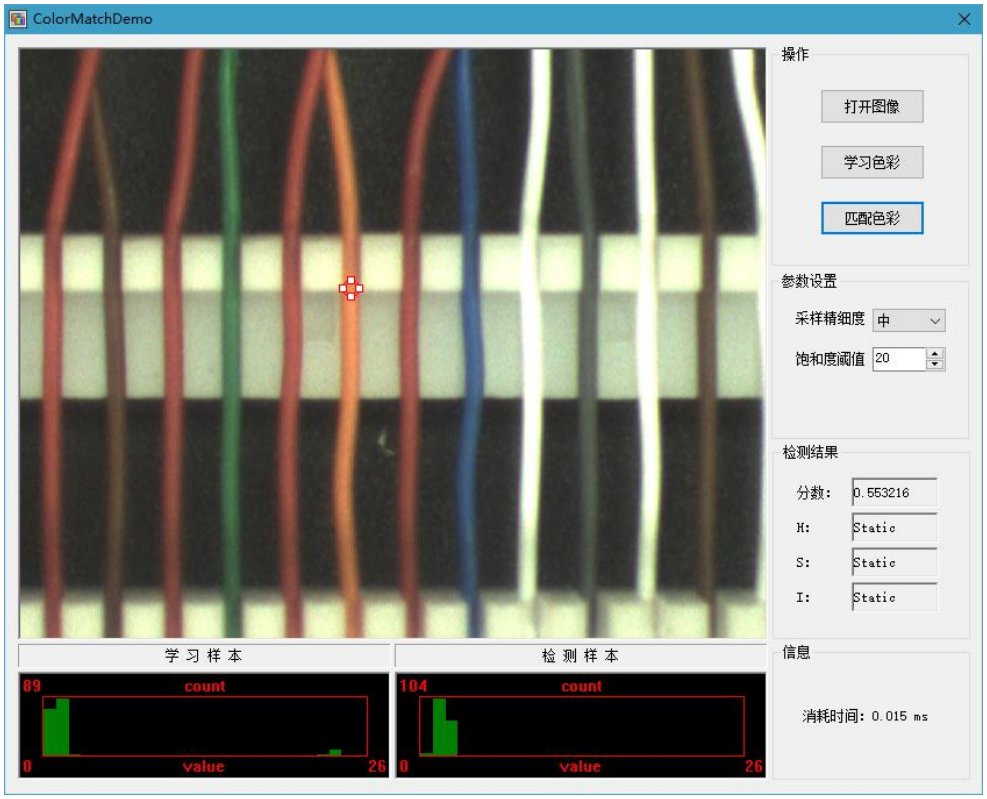
得太近（距离小于滤波半宽值）时反而会影响边缘位置的精度甚至丢失边缘，所有要根据实际情况来设置。



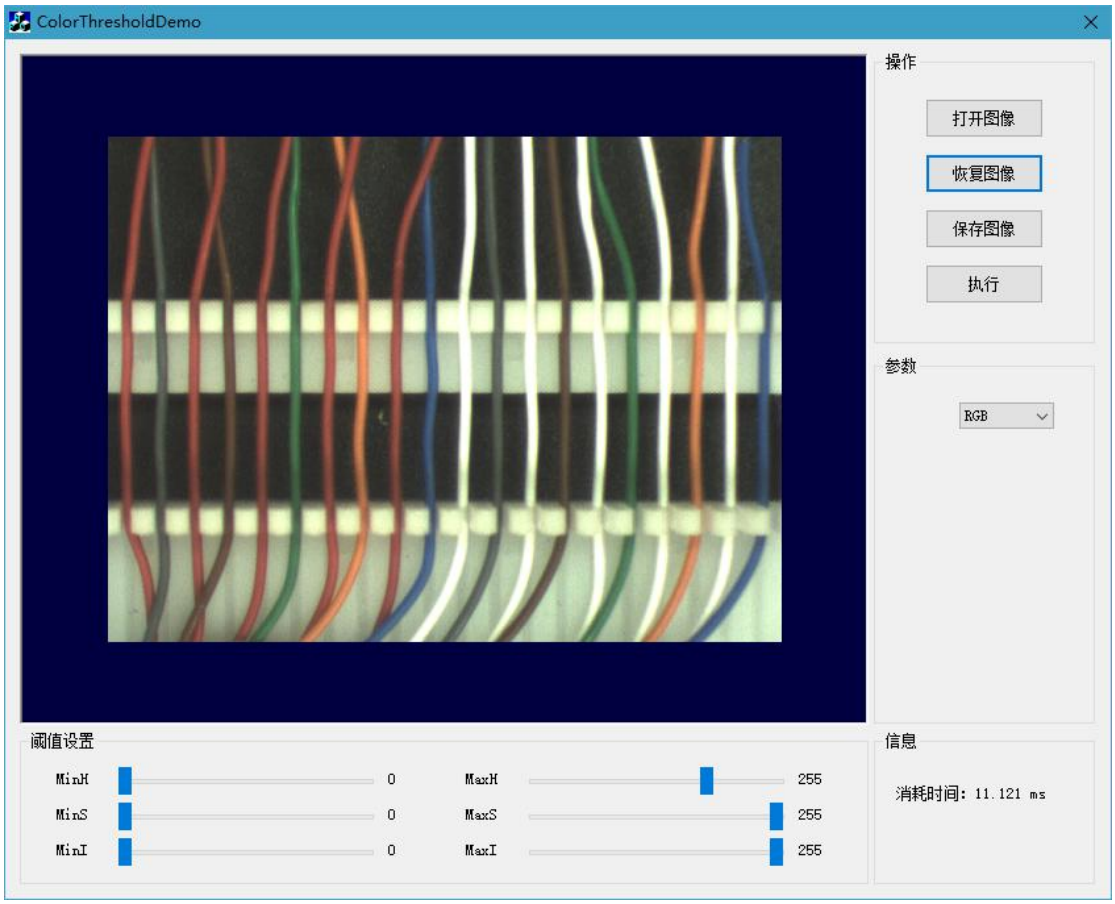
ColorMatchDemo.exe 色彩匹配



预先学习颜色样本，识别出当前颜色属于哪一个样本，并返回匹配程度。



ColorThresholdDemo.exe 彩色二值化

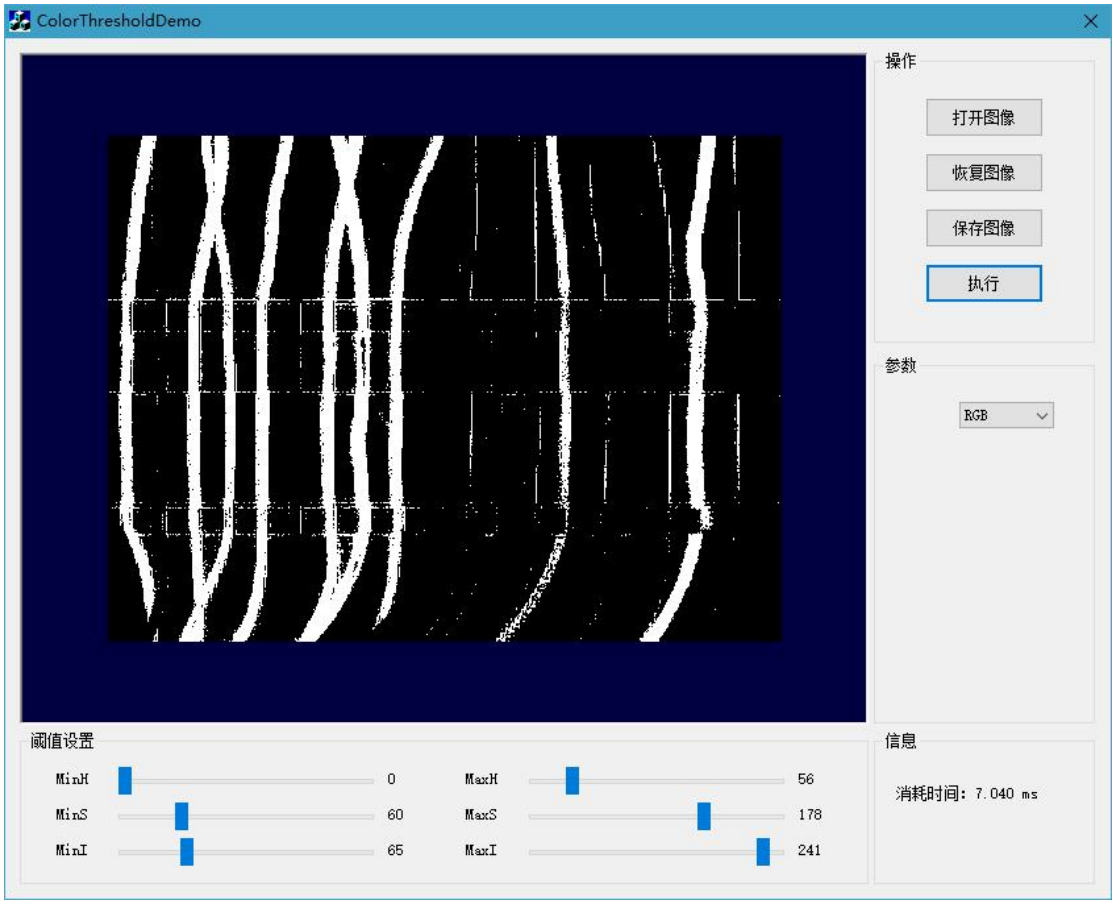


恢复:恢复调节之前的图像。
保存图像 :另存一张调节好的图像。
参数: RGB 、HSI 。
RGB 即是代表红、绿、蓝三个通道的颜色。

色调 H(Hue)：与光波的波长有关，它表示人的感官对不同颜色的感受，如红色、绿色、蓝色等，它也可表示一定范围的颜色，如暖色、冷色等。

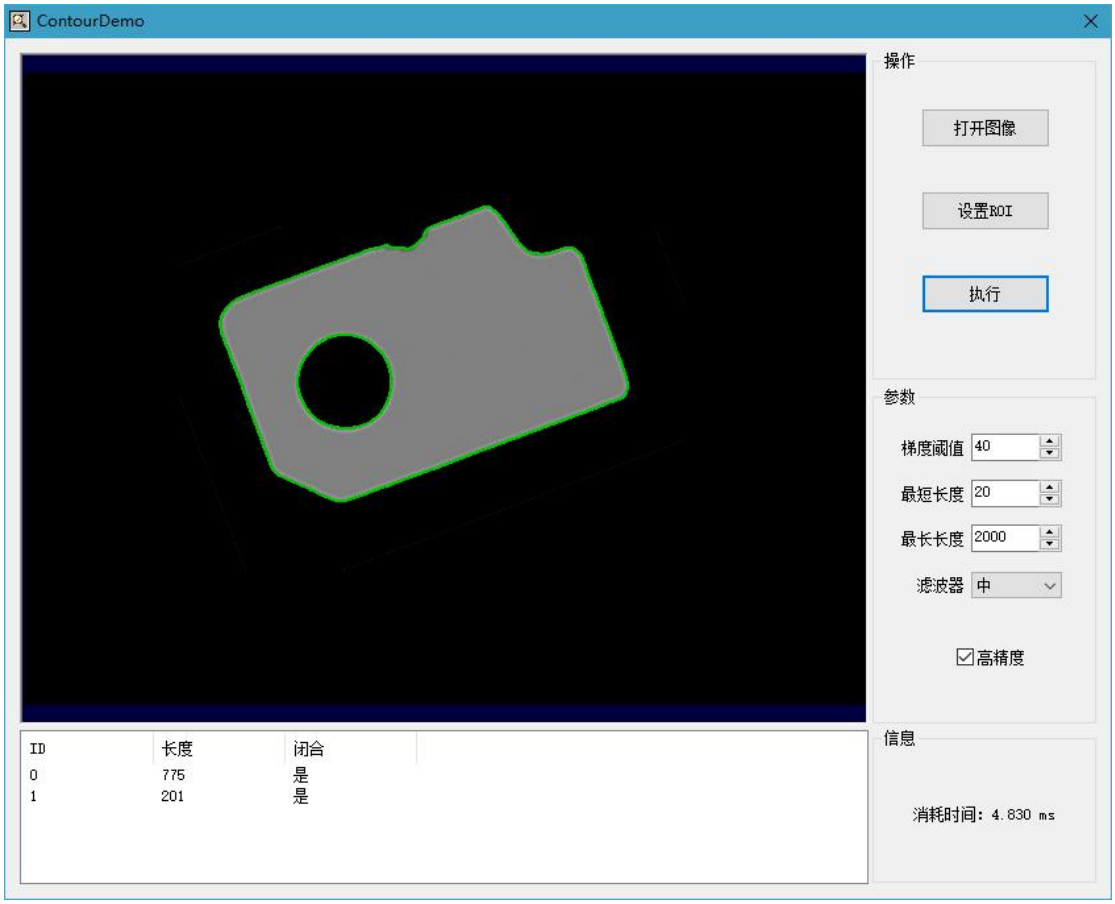
饱和度 S(Saturation)：表示颜色的纯度，纯光谱色是完全饱和的，加入白光会稀释饱和度。饱和度越大，颜色看起来就会越鲜艳，反之亦然。

强度 I(Intensity)：对应成像亮度和图像灰度，是颜色的明亮程度。


ColorThresholdDemo
操作
打开图像
恢复图像
保存图像
执行
参数
RGB
阈值设置
MinH
0
MaxH
56
MinS
60
MaxS
178
MinI
65
MaxI
241
信息
消耗时间：7.040 ms

ContourDemo.exe 轮廓提取

检测产品轮廓，获取轮廓数量以及轮廓长度。



梯度阈值：提取边缘轮廓时使用的参数，当边缘对比度较差时需要降低梯度阈值，如果目标边缘清晰，则可以设置比较高，取值范围 0~255。

最短长度：检测的最短轮廓数据，初始值默认为 20；

最长长度：检测的最长轮廓数据，初始值默认为 2000；

高精度：使用插值功能可以提升定位精度。

滤波器：滤波器可以增强边缘效果，但也会丢失细节，有低、中和高 3 个选项，默认为中。

DataMatrixDemo.exe 二维码读取（DM 码）

读取 DataMatrix 二维码，可以自动定位二维码，并允许二维码图像旋转任意角度。

搜索数量：最多被允许搜索到的目标数量。

滤波级别：用于增强边缘和抑制噪音干扰，分为低、中、高三个级别。

梯度阈值：梯度表示边缘的强度（清晰度），取值范围 0~255，要求二维码的边缘强度大于该值才有可能被定位和读取。

缺失长度：允许检测的二维码定位边缺失部分长度，默认值为 20。

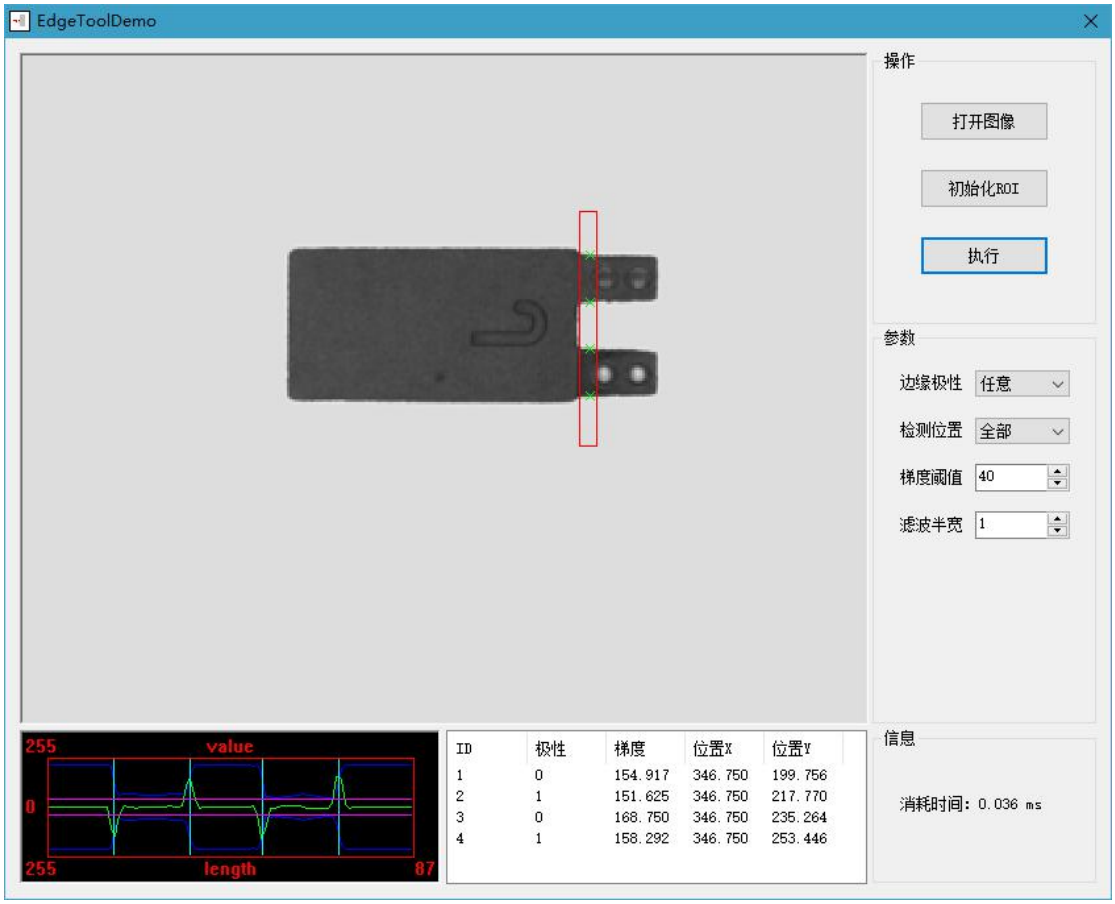
二维码颜色：选择要读取的二维码是黑色、白色或者任意，其中任意包含了黑色和白色。
二维码形状：选择要读取的二维码是正方形、长方形或者任意。

检测结果：显示读取二维码数据结果。

预设尺寸：预先设置所需检测二维码的尺寸大小。

固定单元数：固定设置视野中所需检测二维码的行、列数量。

EdgeToolDemo.exe 边缘点检测



初始化 ROI：

初始 ROI 默认位置与大小，ROI 终点位置（在 ROI 边上中间有圆形旋转调节点的边）。

参数

边缘极性：有亮到暗、暗到亮和任意 3 种模式，从 ROI 起点边开始往结束边的方向上；

亮到暗表示从亮度高过度到亮度低的边缘；

暗到亮表示从亮度低过度到亮度高的边缘；

任意模式则亮到暗和暗到亮的边缘都检测。

检测位置：有起始、最后、最宽和全部 4 种模式；

起始表示只检测最靠近扫描起始位置的 1 个间距；

最后表示只检测离扫描起始位置最远的 1 个间距；

最宽表示只检测扫描范围内距离最远的 1 个间距，

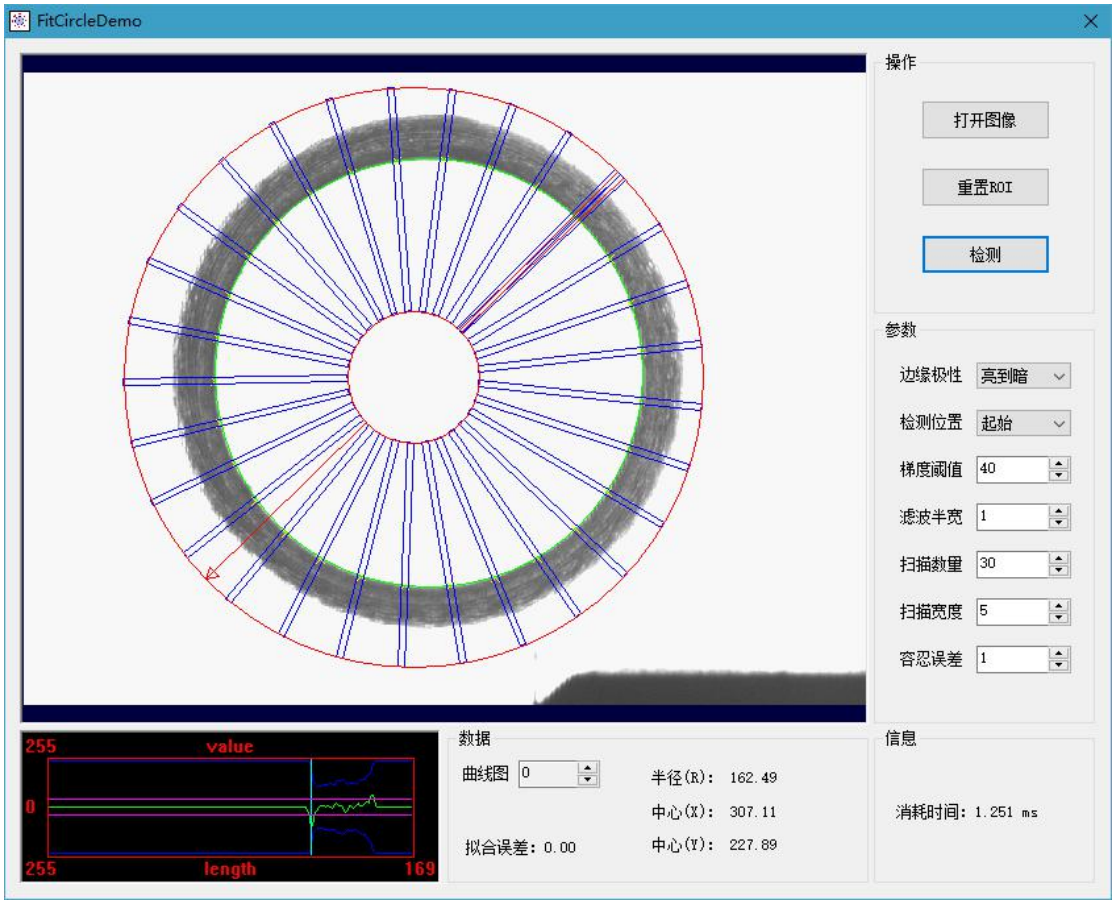
全部表示检测扫描范围所有间距。

梯度阈值：取值范围 0 到 255，只有梯度值大于该值的边缘点才被检测到，梯度值是度量图像边缘的清晰度或对比度。

滤波半宽：用于增强边缘和抑制噪音干扰，最小值为 1，当边缘模糊不清晰或有噪音干扰时可以增大滤波半宽值，这样可以使得检测结果更加稳定，但如果边缘和边缘之间挨得太近（距离小于滤波半宽值）时反而会影响边缘位置的精度甚至丢失边缘，所有要根据实际情况来设置。

FitCircleDemo.exe 圆形测量（拟合圆）

同时检测出多个边缘点并拟合成圆形，可用于测量圆的尺寸和定位。



重置 ROI：

初始 ROI 默认位置与大小。

参数

边缘极性：有亮到暗、暗到亮和任意 3 种模式；

亮到暗表示从亮度高过度到亮度低的边缘；

暗到亮表示从亮度低过度到亮度高的边缘；
任意模式则亮到暗和暗到亮的边缘都检测。

检测位置：有起始、最后、最宽和全部 4 种模式；
起始表示只检测最靠近扫描起始位置的 1 个间距；
最后表示只检测离扫描起始位置最远的 1 个间距；
最宽表示只检测扫描范围内距离最远的 1 个间距，
全部表示检测扫描范围所有间距。

梯度阈值：取值范围 0 到 255，只有梯度值大于该值的边缘点才被检测到，梯度值是度量图像边缘的清晰度或对比度。

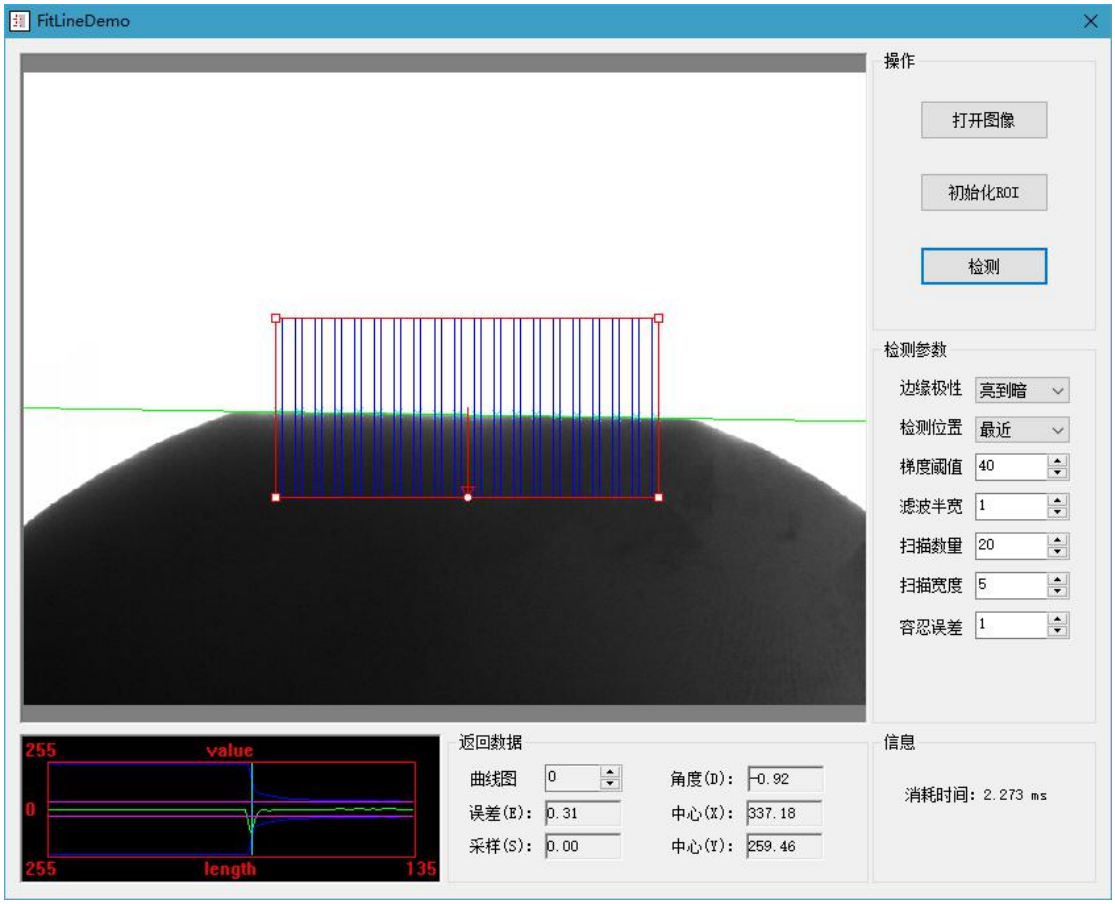
滤波半宽：用于增强边缘和抑制噪音干扰，最小值为 1，当边缘模糊不清晰或有噪音干扰时可以增大滤波半宽值，这样可以使得检测结果更加稳定，但如果边缘和边缘之间挨得太近（距离小于滤波半宽值）时反而会影响边缘位置的精度甚至丢失边缘，所有要根据实际情况来设置。

扫描数量：设置扫描边缘点 ROI（图中蓝色图形区域）的数量

扫描宽度：扫描边缘点 ROI（图中蓝色图形区域）的宽度，增大宽度可以在一定范围内求平均，使得计算结果根据稳定。

容忍误差：容忍拟合的最小误差值，拟合出来出来的圆形误差会小于该值，误差大的点将会被排除。

FitLineDemo.exe 直线测量（拟合直线）



初始化 ROI：

初始 ROI 默认位置与大小。

参数

边缘极性：有亮到暗、暗到亮和任意 3 种模式；

亮到暗表示从亮度高过度到亮度低的边缘；

暗到亮表示从亮度低过度到亮度高的边缘；

任意模式则亮到暗和暗到亮的边缘都检测。

检测位置：有起始、最后、最宽和全部 4 种模式；

起始表示只检测最靠近扫描起始位置的 1 个间距；

最后表示只检测离扫描起始位置最远的 1 个间距；

最宽表示只检测扫描范围内距离最远的 1 个间距，

全部表示检测扫描范围所有间距。

梯度阈值：取值范围 0 到 255，只有梯度值大于该值的边缘点才被检测到，梯度值是度量图像边缘的清晰度或对比度。

滤波半宽：用于增强边缘和抑制噪音干扰，最小值为 1，当边缘模糊不清晰或有噪音干扰时可以增大滤波半宽值，这样可以使得检测结果更加稳定，但如果边缘和边缘之间挨

得太近（距离小于滤波半宽值）时反而会影响边缘位置的精度甚至丢失边缘，所有要根据实际情况来设置。

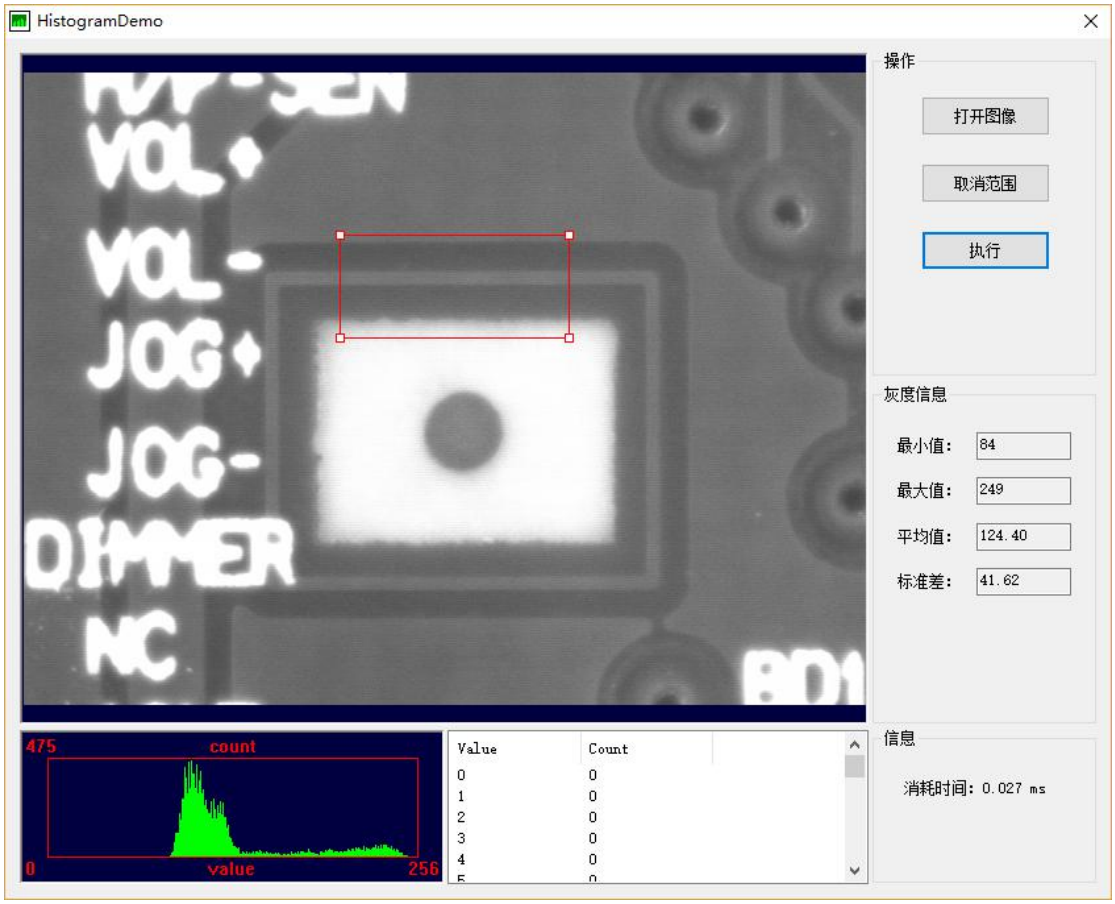
扫描数量：设置扫描边缘点 ROI（图中蓝色图形区域）的数量

扫描宽度：扫描边缘点 ROI（图中蓝色图形区域）的宽度，增大宽度可以在一定范围内求平均，使得计算结果根据稳定。

容忍误差：容忍拟合的最小误差值，拟合出来出来的圆形误差会小于该值，误差大的点将会被排除。

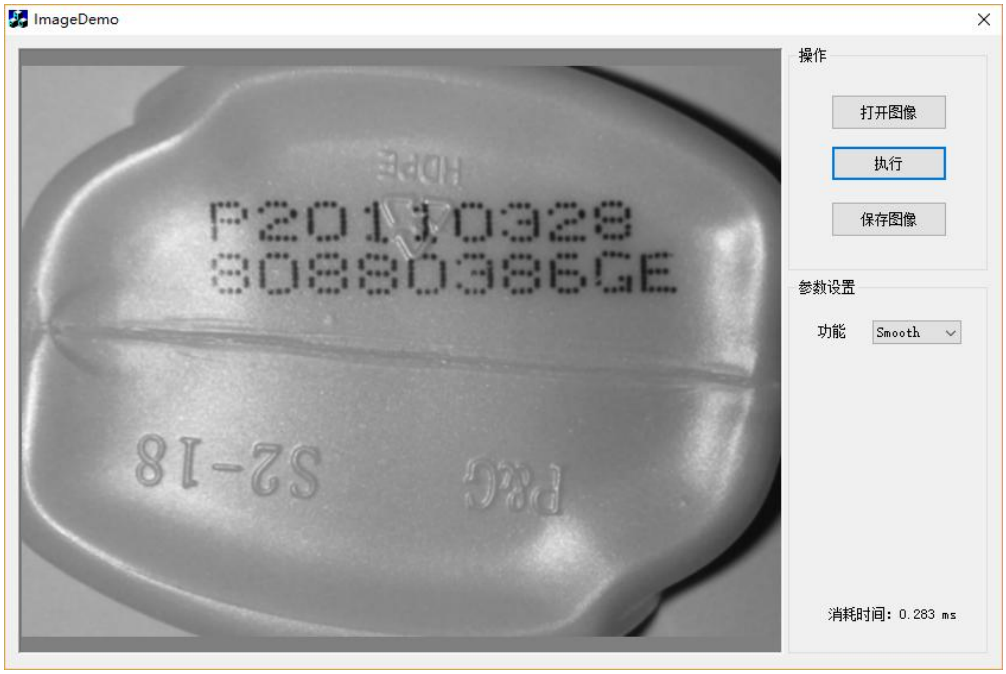
HistogramDemo.exe 灰度直方图、亮度检测、自动二值化阈值

检测图像中指定区域的灰度平均值（亮度）和标准差。



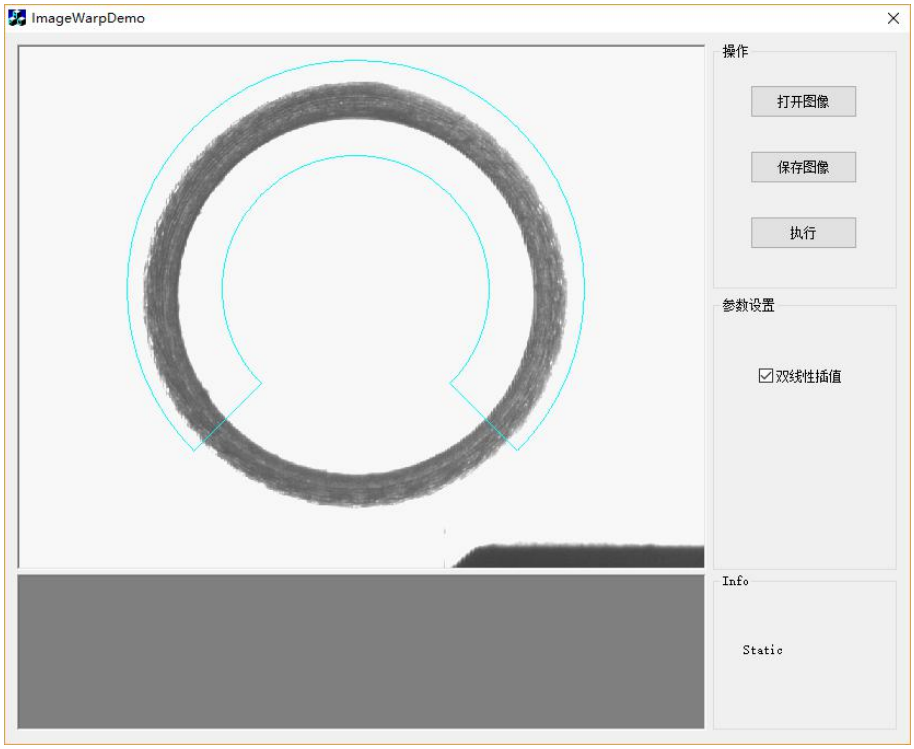
灰度信息：显示检测后的数据，最大值、最小值、平均值和标准差；标准差是一组数据平均值分散程度的一种度量，一个较大的标准差，代表大部分数值和其平均值之间差异较大；一个较小的标准差，代表这些数值较接近平均值。

ImageDemo.exe 图像预处理



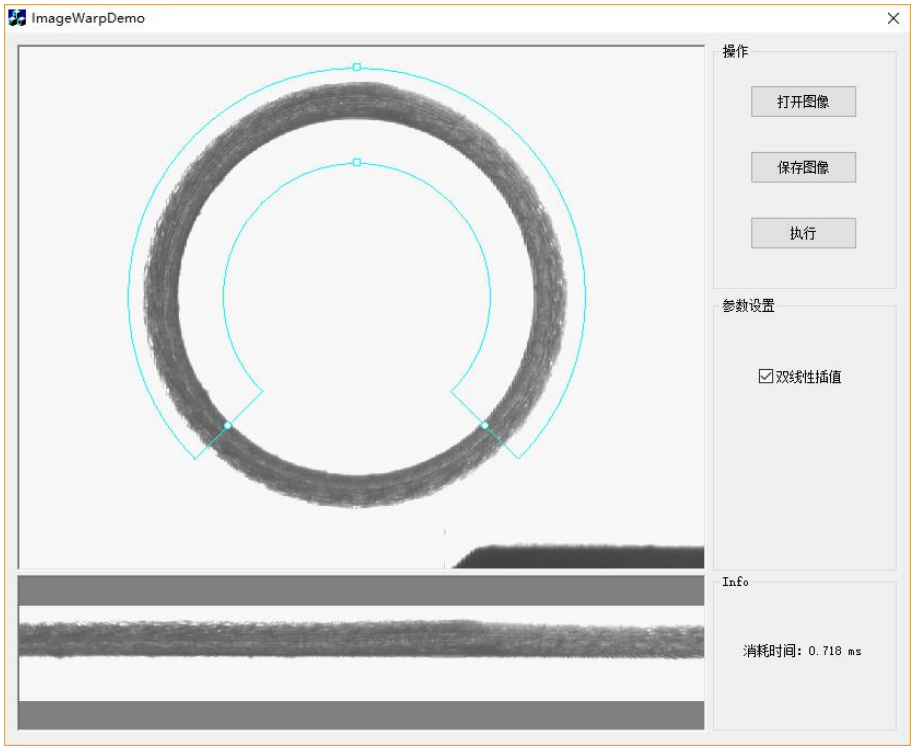
对图像做一些预处理（平滑、锐化、腐蚀和膨胀等功能）后并输出处理后的图像。

ImageWarpDemo.exe 环形展开裁剪图像

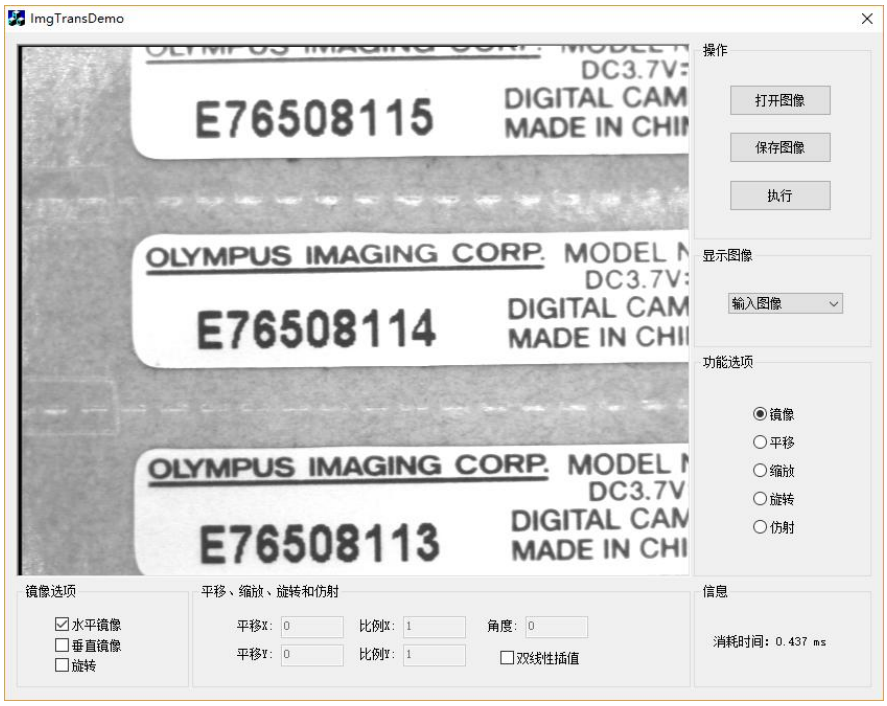


按 ROI 区域裁剪展开。

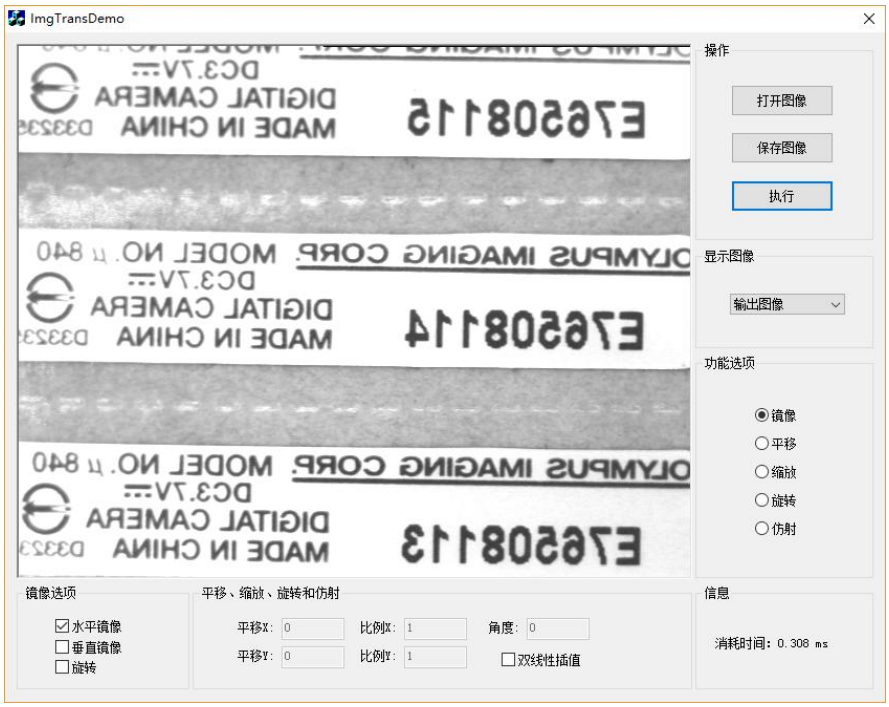
裁剪展开之后效果如下图：

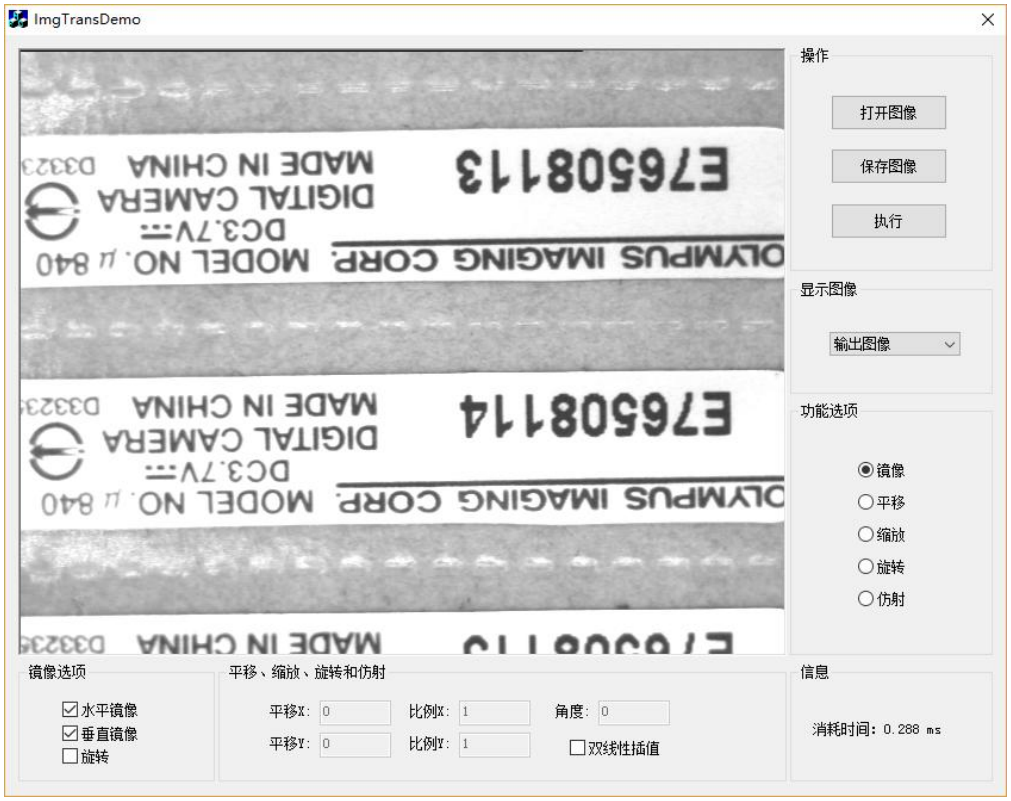


ImgTransDemo.exe 图形变换（镜像、平移、旋转、缩放、仿射）


ImgTransDemo
操作
打开图像
保存图像
执行
显示图像
输入图像
功能选项
镜像
平移
缩放
旋转
仿射
镜像选项
水平镜像
垂直镜像
旋转
平移、缩放、旋转和仿射
平移X: 0
比例X: 1
角度: 0
平移Y: 0
比例Y: 1
双线性插值
信息
消耗时间：0.437 ms

镜像变换：


ImgTransDemo
操作
打开图像
保存图像
执行
显示图像
输出图像
功能选项
镜像
平移
缩放
旋转
仿射
镜像选项
水平镜像
垂直镜像
旋转
平移、缩放、旋转和仿射
平移X: 0
比例X: 1
角度: 0
平移Y: 0
比例Y: 1
双线性插值
信息
消耗时间：0.308 ms


ImgTransDemo
操作
打开图像
保存图像
执行
显示图像
输出图像
功能选项
镜像
平移
缩放
旋转
仿射
镜像选项
水平镜像
垂直镜像
旋转
平移、缩放、旋转和仿射
平移X: 0
比例X: 1
角度: 0
平移Y: 0
比例Y: 1
双线性插值
信息
消耗时间: 0.288 ms

缩放变换：


ImgTransDemo
操作
打开图像
保存图像
执行
显示图像
输出图像
功能选项
镜像
平移
缩放
旋转
仿射
镜像选项
水平镜像
垂直镜像
旋转
平移、缩放、旋转和仿射
平移X: 0
比例X: 0.5
角度: 10
平移Y: 0
比例Y: 0.5
双线性插值
信息
消耗时间: 0.728 ms

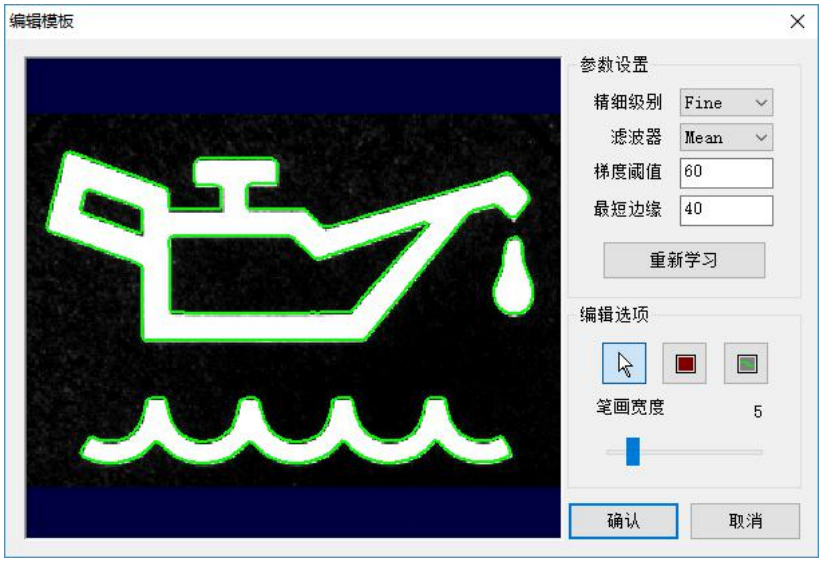
旋转：


ImgTransDemo
操作
打开图像
保存图像
执行
显示图像
输出图像
功能选项
镜像
平移
缩放
旋转
仿射
镜像选项
水平镜像
垂直镜像
旋转
平移、缩放、旋转和仿射
平移X: 0
平移Y: 0
比例X: 1
比例Y: 1
角度: 10
双线性插值
信息
消耗时间：24.121 ms

InspectDemo.exe 图像对比缺陷检测

使用模板图像和当前图像进行对比，将差异部分检测出来，可用于一般的缺陷检测。

设置学习区域：设置学习区域的 ROI。
学习模板：学习定位模板与图像对比模板。
编辑模板：编辑定位模板。



生成:按照(Half)半宽大小重新把学习的模板生成一个掩模图(屏蔽图)。
统计 :把检测测试对比结果图添加到掩模图。

修正模式 ：

亮度修正，对检测结果由于对画面的明亮程度差异造成的影响可消除；

对比度修正，对比度是图像中明暗区域最亮的白和最暗的黑之间不同亮度层级的测量，差异范围越大代表对比越大，差异范围越小代表对比越小；

无修正，不进行修正，直接进行检测。

差异类型:选择哪一种不同的情况进行检测，亮度或对比度或者是全部（亮度和对比度都检测）。

Blob 分析，分析对比之后的差异图像。

阈值：设置二值图像的分割阈值，当像素灰度值大于等于该值为白色，否则为黑色。

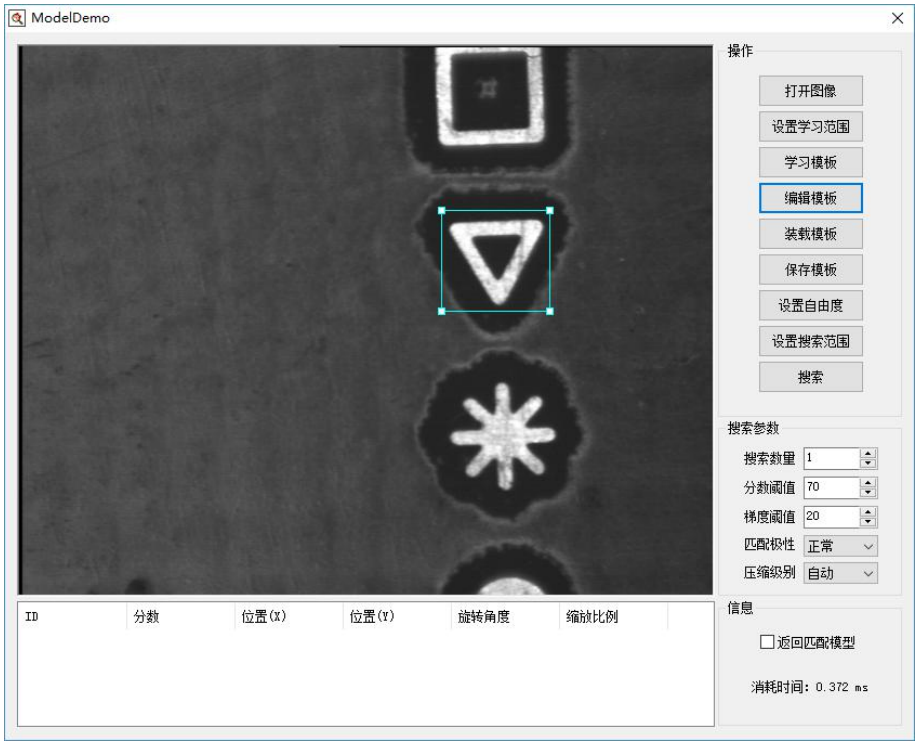
面积：限制最小的面积，只有缺陷大于限制的面积才会被检测出来。


InspectDemo
操作
打开图像
设置学习区域
学习模板
编辑模板
执行
掩摸图操作
Half 1
生成
统计
检查参数
修正模式 Null
差异类型 全部
使用掩摸图像
Blob分析
阈值 60
面积 10
信息
来源图像：
模板图像：
掩膜图像：
差异图像：
消耗时间：13.456 ms
Static


InspectDemo
操作
打开图像
设置学习区域
学习模板
编辑模板
执行
掩摸图操作
Half 1
生成
统计
检查参数
修正模式 Null
差异类型 全部
使用掩摸图像
Blob分析
阈值 60
面积 10
信息
来源图像：
模板图像：
掩膜图像：
差异图像：
消耗时间：11.668 ms
Static

ModelDemo.exe 模板轮廓匹配定位（老版本）

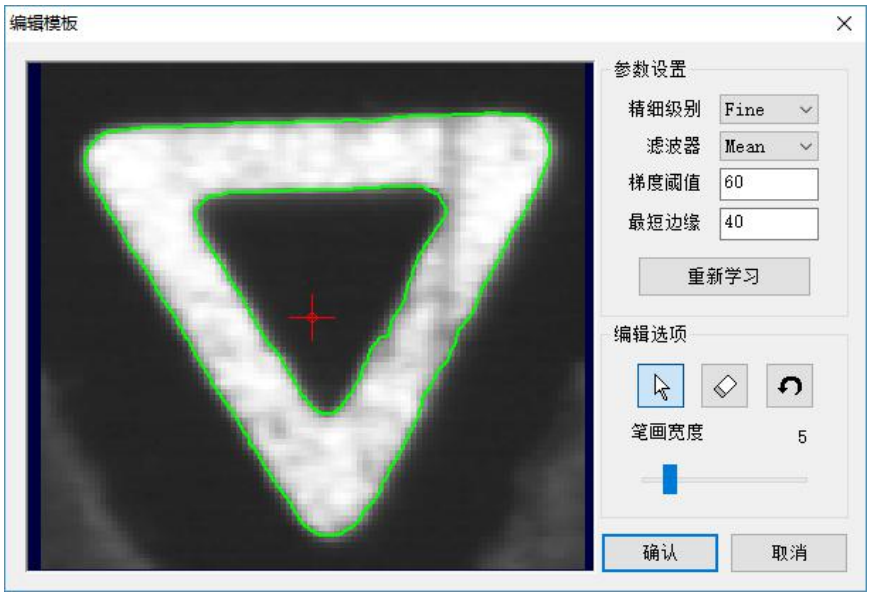
使用图像的边缘轮廓特征作为模板，在图像中搜索形状上相似的目标，可以设置角度和比例范围，可用于定位、计数和判断有无等



设置学习范围：将会在画面中出现一个青蓝色的 ROI 矩形框，点击选取矩形框，再将该矩形移动到需要作为标准模板的图像区域并调整大小（如上图）。

学习模板：按学习 ROI 在图像上学模板轮廓。

编辑模板：学模板之后可查看编辑模板。



精细级别：定义边缘轮廓的细腻程度，有 Fine、Noraml 和 Coarse3 个选项，Fine 模式精度最高，但边缘太模糊时将无法检测到边缘，Noraml 和 Coarse3 模式则会通过压缩方式来提取模糊的边缘，同时会丢失细节部分，并会影响定位进精度。

滤波器：用于增强边缘提取功能，跟精细级别类似，但不会影响定位精度。

梯度阈值：取值范围 0 到 255，只有梯度值大于该值的边缘点才被检测到，梯度值是度量图像边缘的清晰度或对比度。

最短边缘：用于过滤长度小于该值的边缘轮廓。

重新学习：当修改参数后需要点击“重新学习”按钮来获得新的边缘轮廓。

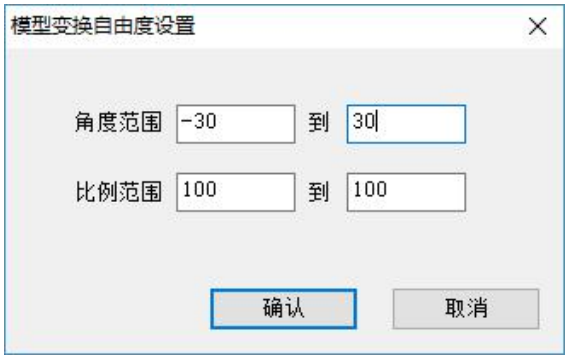
编辑选项：手动编辑模板功能，“指示”可修改当前十字点位置，点击“擦除”或“恢复”按钮后可以使用鼠标在画面中对着边缘轮廓进行擦除或者恢复，上图中蓝色轮廓部分为被擦除，绿色部分为正在使用的边缘轮廓。

画笔大小：设置擦除或者恢复画笔的尺寸大小。

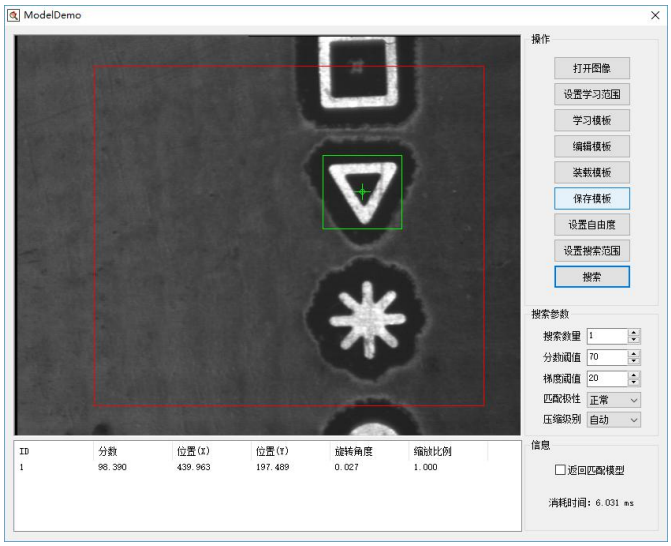
标记点：画面中红色十字标为模板的标记点，在“指示”模式下可以使用鼠标点击选取并拖动调整位置。

装载模板：从文件中加载模板数据。
保存模板：将模板数据保存到文件中。

设置自由度，
角度范围：匹配目标相对于模板可能存在的最小角度，值范围-180 到 180。
比例范围：匹配目标相对于模板可能存在的最小比例，值范围 80 到 120（ 原始比例值 100 ）。



设置搜索范围：在图像上画出一个红色的 ROI 矩形框，该矩形框为搜索目标是的搜索范围，如果是全图搜索可以不用设置。



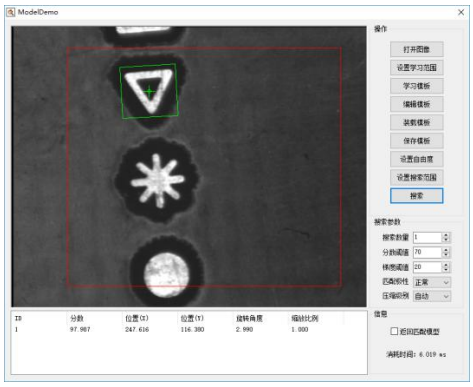
搜索数量：最多被允许搜索到的目标数量。

最小分数：分数表示目标和模板的相似程度，分数越高越相似，最大值 100 表示完全匹配，目标分数必须大于该值才会被搜索到，该参数值将会影响搜索速度。

梯度阈值：提取边缘轮廓时使用的参数，当边缘对比度较差时需要降低梯度阈值，如果目标边缘清晰，则可以设置比较高，取值范围 0~255，一般设为 40 左右，该参数值将会影响搜索速度。

匹配极性：可以设置正常和反转，正常表示目标和模板极性相同，反转则表示相反。

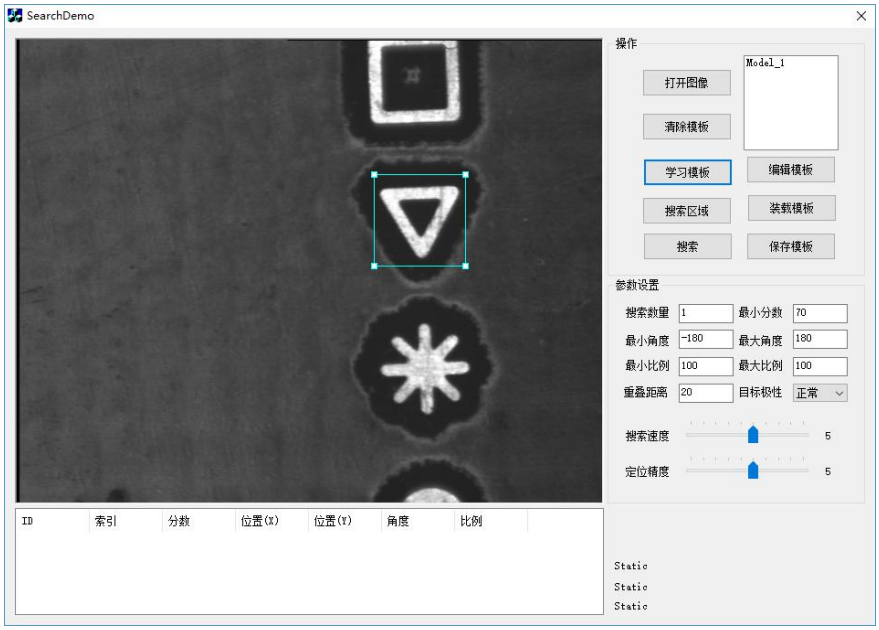
压缩级别：在搜索过程中对图像进行压缩处理可以提升搜索速度，但也会降低识别率（影响程度跟模板和目标图像背景复杂度有关），一般采用“自动”设置。



分数：匹配目标与模板的相似度。
位置：匹配目标相对于当前图像的坐标位置。
角度：匹配目标相对于模板的旋转角度。
比例：匹配目标相对于模板的缩放比例。

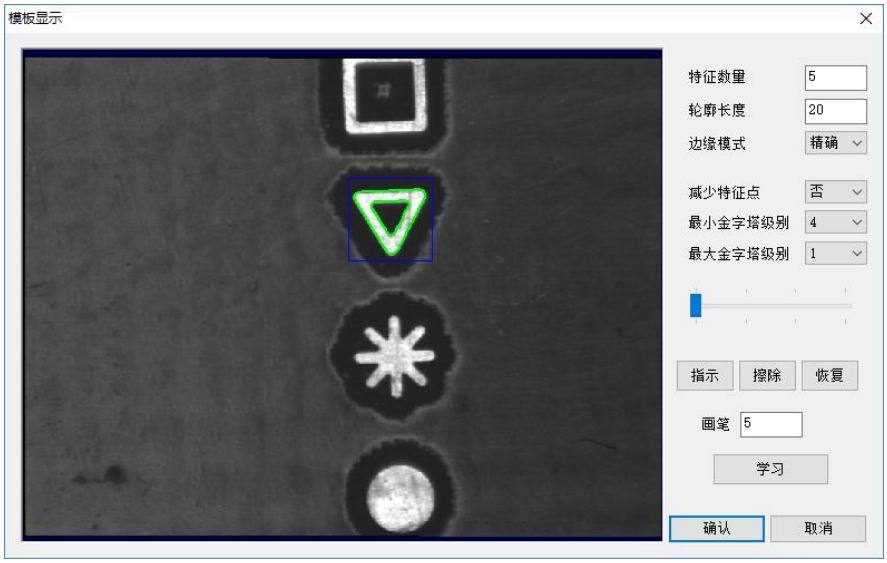
MultiModelDemo.exe 多轮廓匹配定位（新版本）

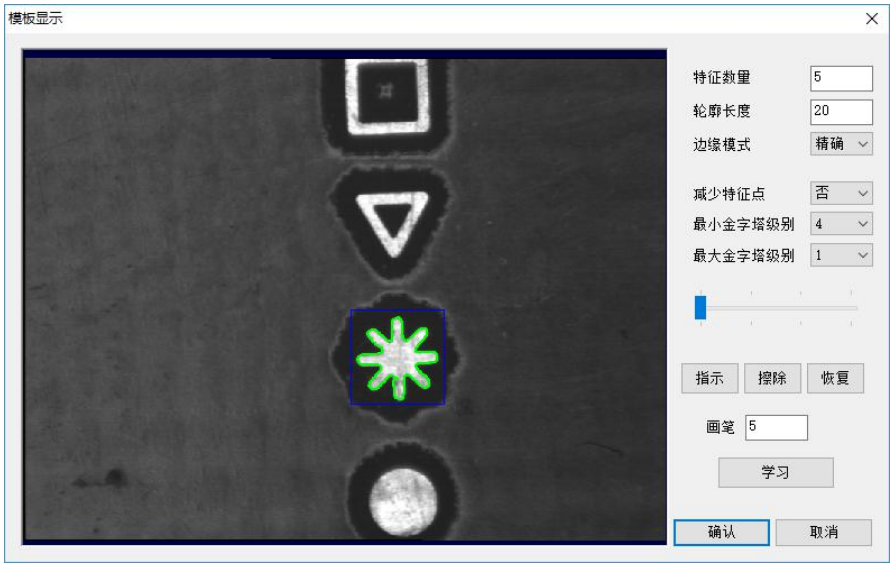
以边缘轮廓特征作为模板，可学习多个目标作为模板，在图像中搜索形状相似的目标。



学习模板：以图像上的 ROI 所在位置学习模板轮廓。

编辑模板：点击“编辑”按钮将会弹出编辑模板对话框，在编辑模板对话框下可以对模板进行编辑。





特征数量：模板特征数量占学习区域特征数量的百分比。

轮廓长度：轮廓长度参数用于过滤，长度小于该值的轮廓将会被删除。

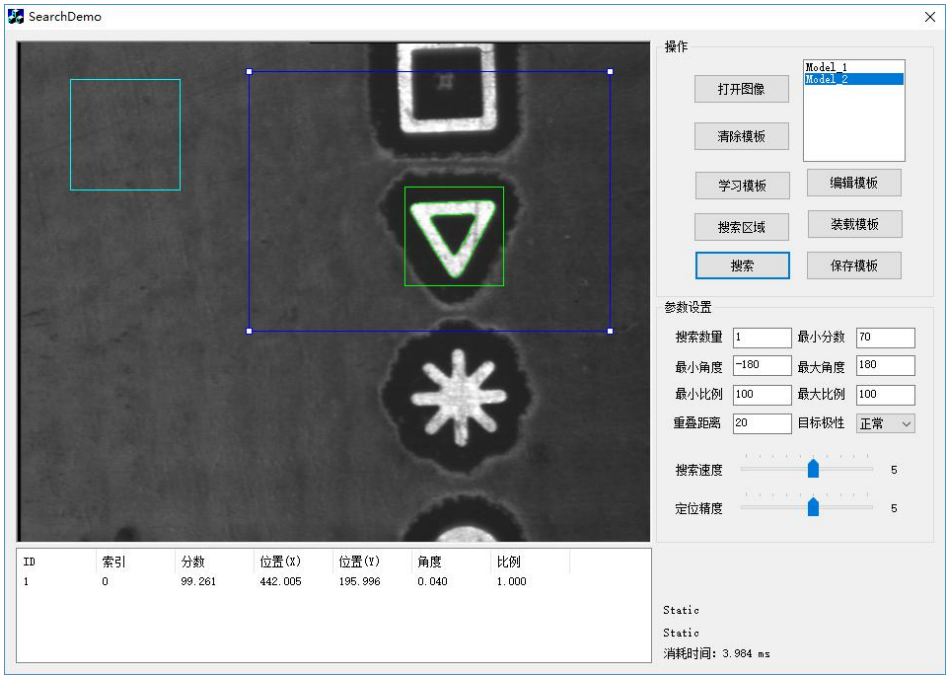
减少特征点：是否减少模版特征点数量。

金字塔级别：可预览当前图像模板层数，点击获取边缘轮廓按钮后会自动获取金字塔级别数值。

掩模编辑：点击“屏蔽”按钮可设置掩模图像屏蔽区域，设置完屏蔽区域后，需要点击“获取边缘轮廓”才能成功屏蔽所选区域，如上图中红色区域部分为被屏蔽，绿色部分为正在使用的边缘轮廓，点击“恢复”按钮可恢复图像中被屏蔽的区域。

画笔大小：设置擦除或者恢复画笔的尺寸大小。

学习：重新学习轮廓模板。



搜索数量：最多被允许搜索到的目标数量。

最小分数：分数表示目标和模板的相似程度，分数越高越相似，最大值 100 表示完全匹配，目标分数必须大于该值才会被搜索到，该参数值将会影响搜索速度。

角度：可以设置被搜索目标可能存在的角度范围，角度为目标相对于模板的角度，取值范围 -180~180。

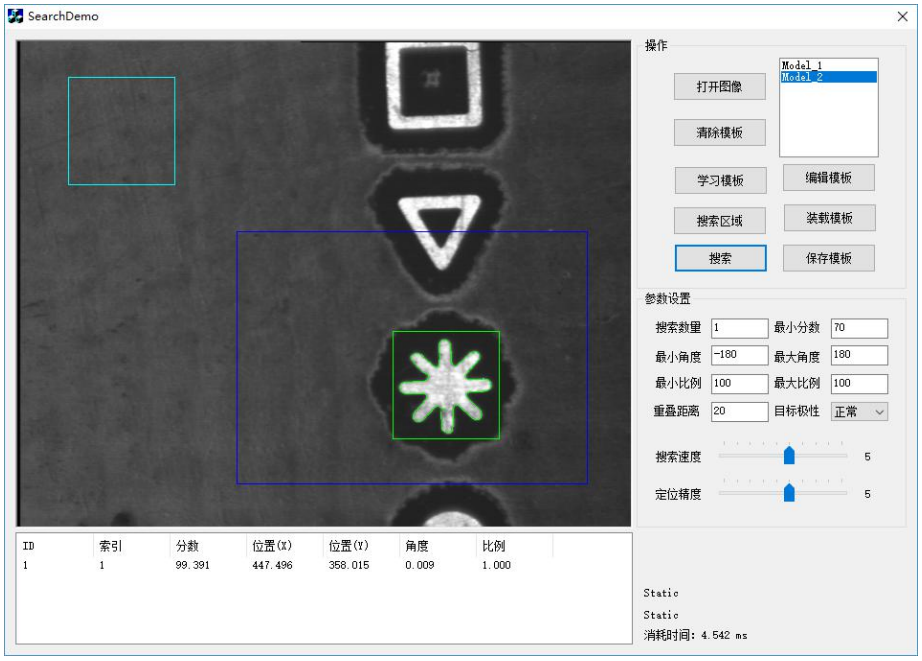
比例：可以设置被搜索目标可能存在的比例范围，比例为目标相对于模板的比例，取值范围 80~120。

重叠距离：匹配多个目标时之间的最小距离不能小于重叠距离，小于则忽略。

匹配极性：可以设置正常和反转，正常表示目标和模板极性相同，反转则表示相反。

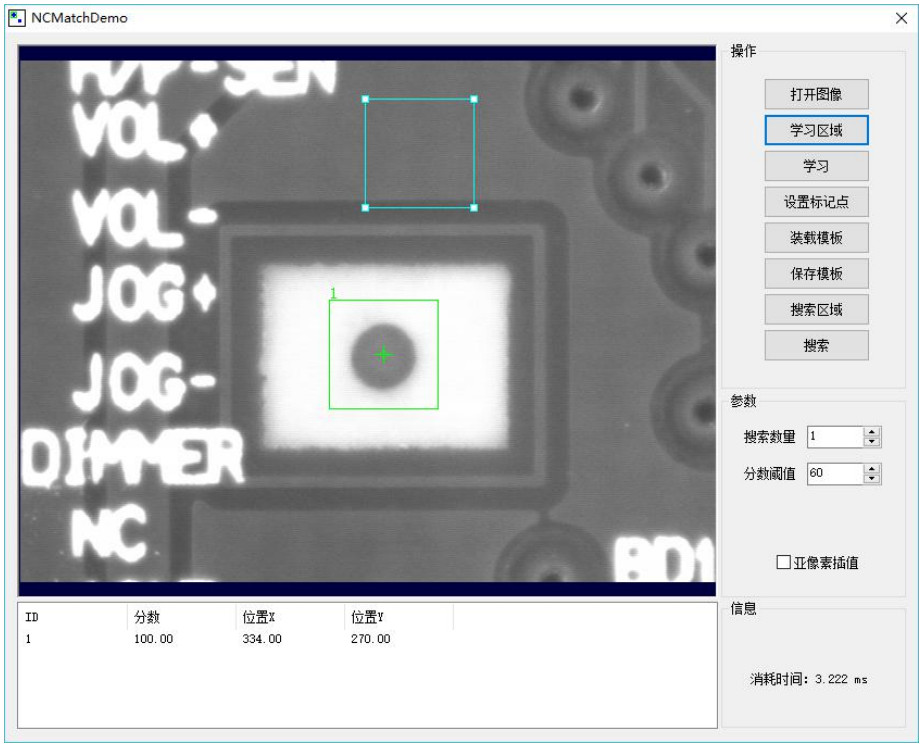
搜索速度：总共有 10 个等级，等级为 0~9,默认为 5，设置的搜素速度级别越高，识别度会有所下降。

定位精度：总共有 10 个等级，等级为 0~9,默认为 5，设置的定位精度级别越高，搜索速度会有所下降。

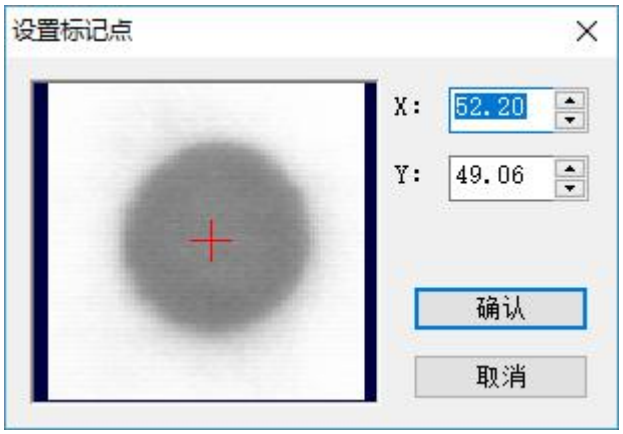


NCMatchDemo.exe 灰度匹配定位

在图像指定区域中搜索跟模板图像相似的目标，使用灰度归一化互相关匹配方法，目标图像和模板图像允之间许存在亮度和对比度变化，可用于定位、计数和判断有无等。



学习：点击“学习”按钮可进入学习模式，在学习模式下将画面中的 ROI 拖动到需要作为模板的图像区域，然后点击“确认”按钮完成学习。



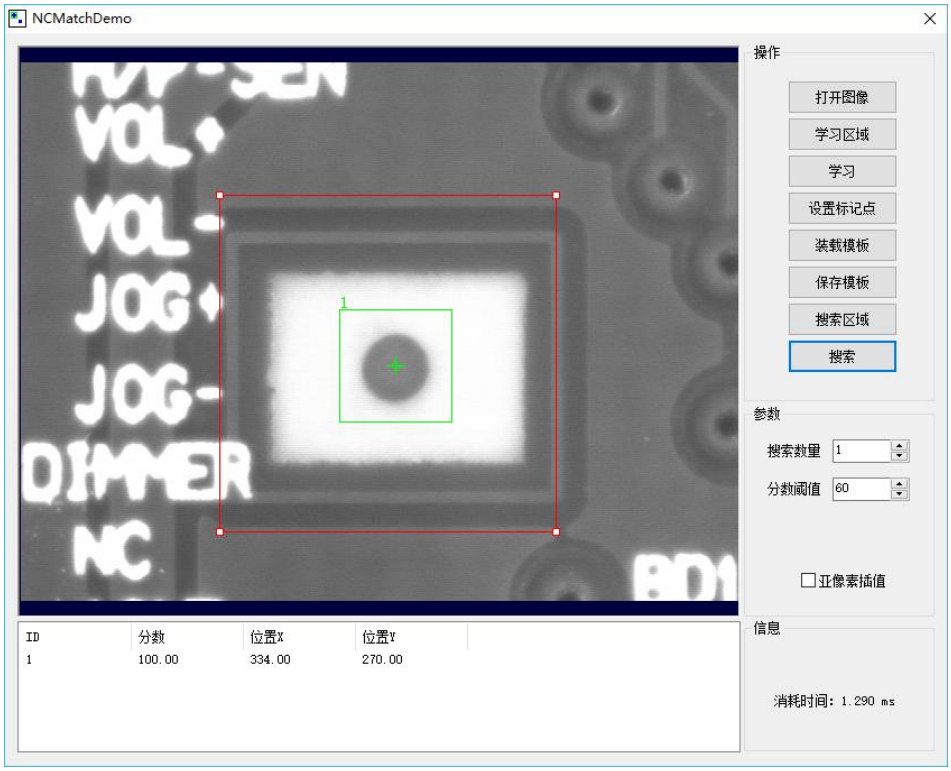
模板：显示模板图像数据和模板的标记点（红色十字标），可以使用鼠标点击选取标记点，然后再拖动可调整标记点位置。
装载模板：从文件中加载模板数据。

保存模板：将模板数据保存到文件中。

搜索数量：最多被允许搜索到的目标数量。

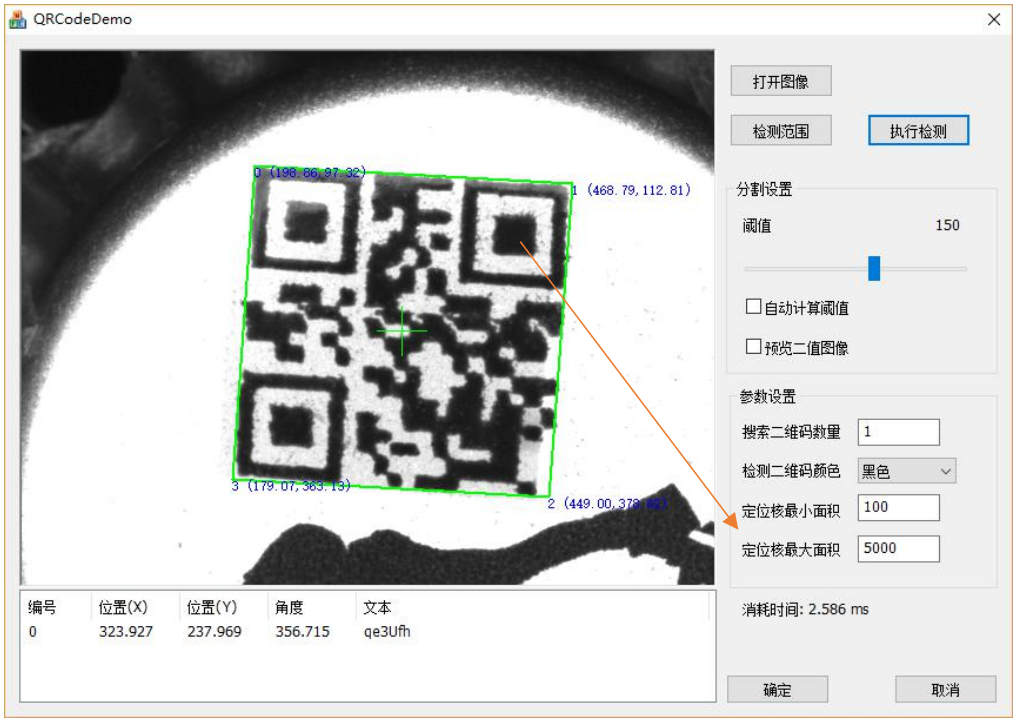
最小分数：分数表示目标和模板的相似程度，分数越高越相似，最大值 100 表示完全匹配，目标分数必须大于该值才会被搜索到，该参数值将会影响搜索速度。

亚像素插值：使用插值功能可以提升定位精度。



QRCodeDemo.exe 二维码检测（QR 码）

读取 QR 码，可以自动定位 QR 码，并允许 QR 图像旋转任意角度。



分割设置

阈值：设置二值图像的分割阈值，当像素灰度值大于等于该值为白色，否则为黑色。

自动计算阈值：软件将根据直方图分布自动计算出分割阈值。

二值图像预览：为了方便调整阈值，以二值化效果显示当前图像。

参数设置

搜索二维码数量：最多被允许搜索到的二维码数量。

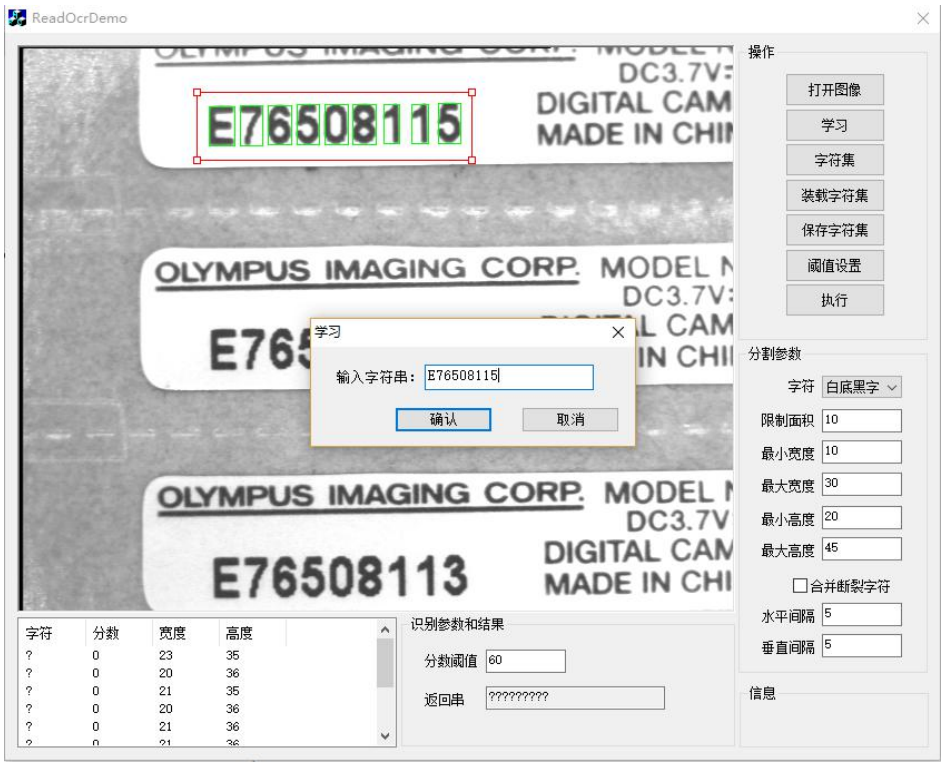
检测二维码颜色：选择要读取的二维码是黑色、白色。

定位核最小面积：定位标记最小面积。

定位核最大面积：定位标记最大面积。

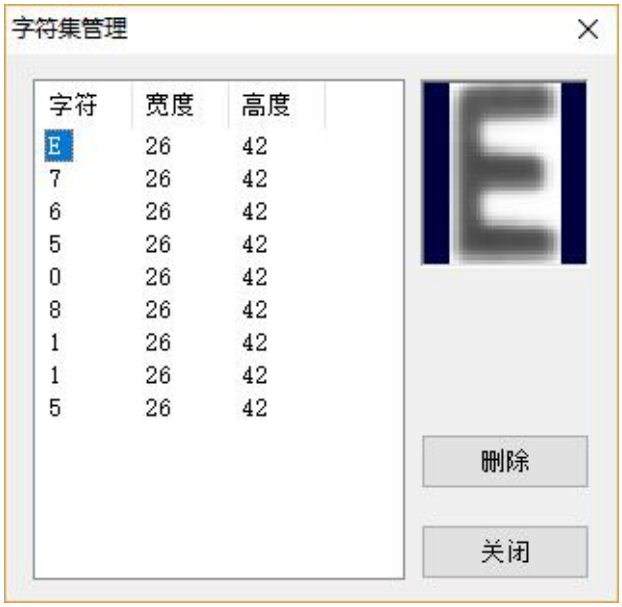
ReadOcrDemo.exe 字符读取

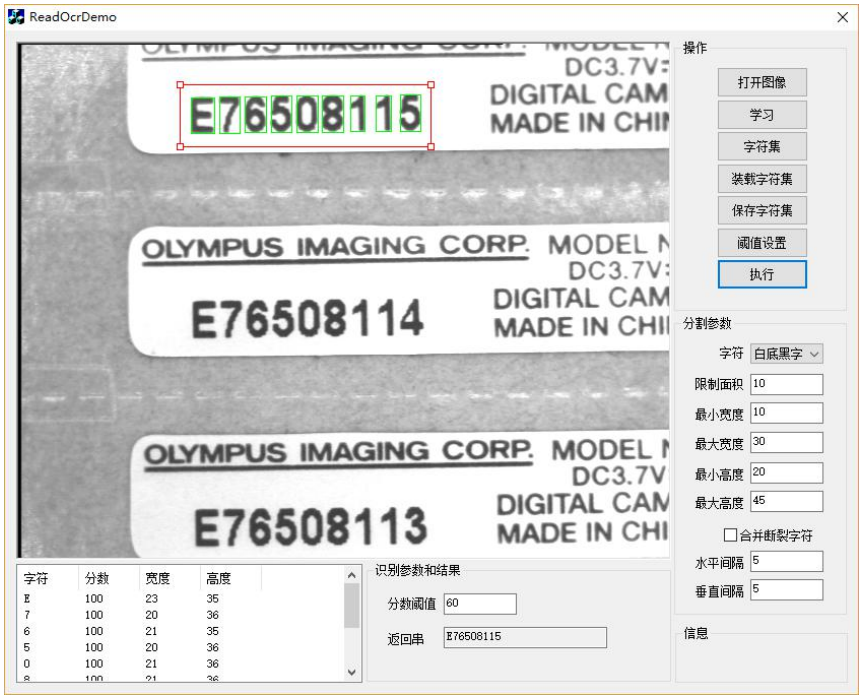
读取标签上的字符文本，需要事先将标准字符录入字符集合中。



学习：先调整 ROI 后点“执行”，然后在点“学习”，出现输入学习字符框，填入对应字符，确认完成字符学习。

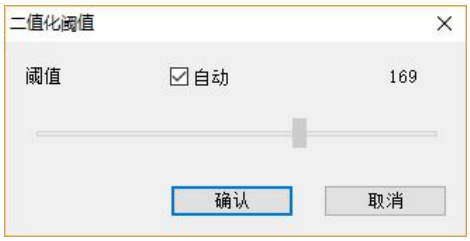
字符集：学习字符后的字符集合。





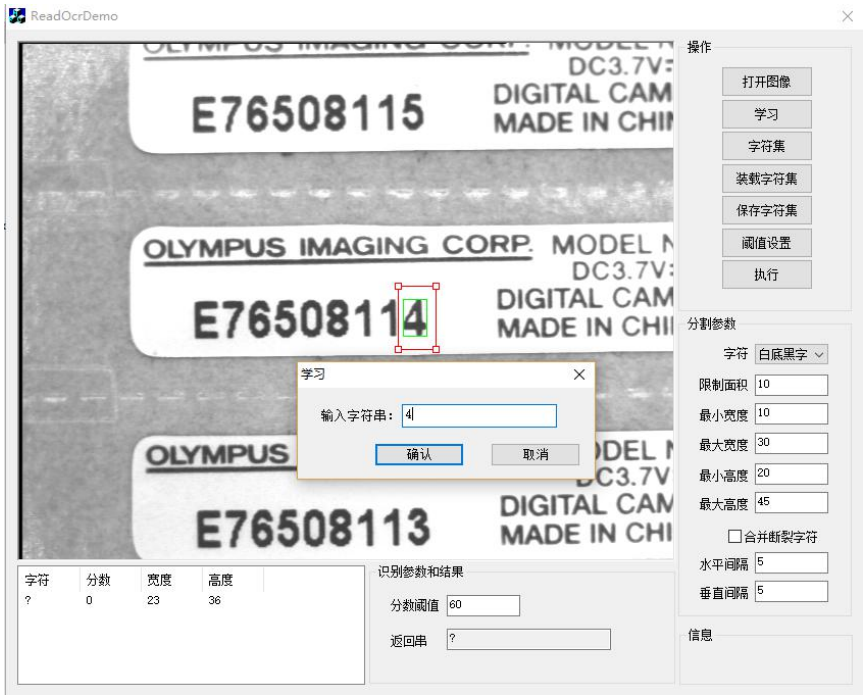
字符集：选择当前使用的字符集工具，点击右边的“...”按钮可以弹出字符集属性对话框。

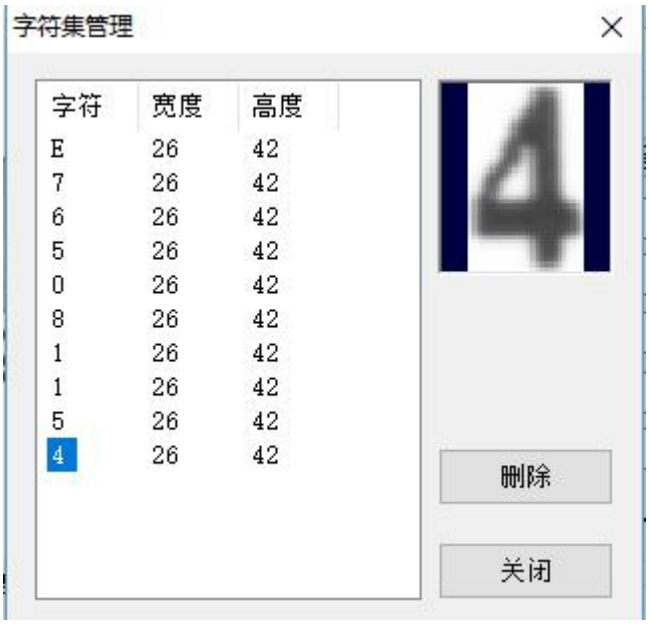
阈值设置：设置字符和背景之间亮度的分界线，用于分割字符区域。

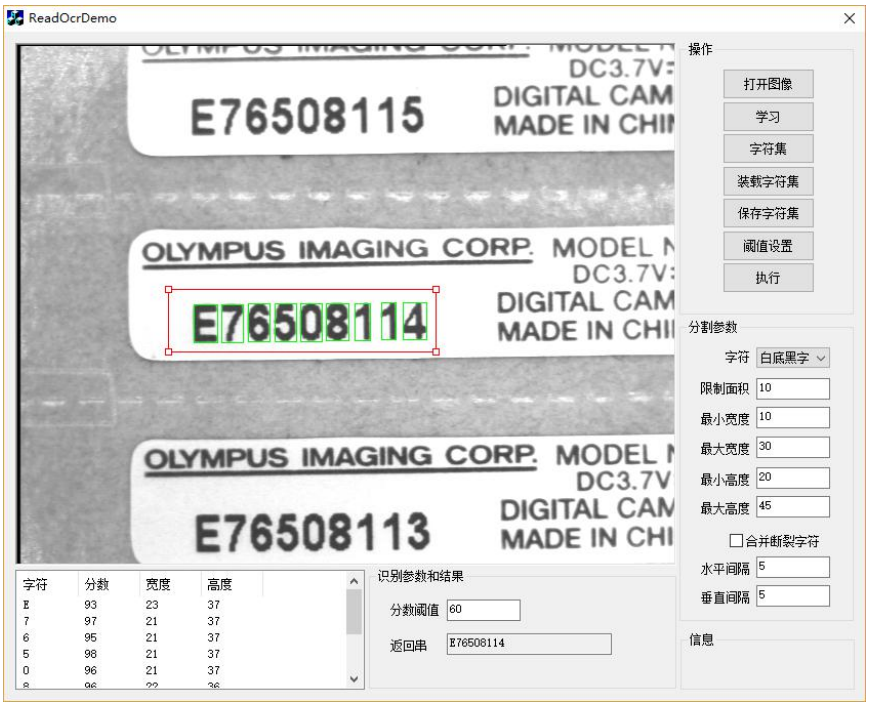


自动阈值：自动计算分割阈值功能。

字符极性：设置字符是黑色还是白色。
限制面积：设置噪声点面积，面积小于该值的区域将被过滤。
字符宽度：设置字符的最小宽度和最大宽度。
字符高度：设置字符的最小高度和最大宽度。
合并断裂字符：如果某个字符断裂成多个部分（点阵字符），可以使用该功能合并。
水平间隔：设置断裂字符水平方向间隔，水平间隔小于该值不能进行合并。
垂直间隔：设置断裂字符垂直方向间隔，垂直间隔小于该值不能进行合并。
分数阈值：设置识别字符的最小分数，小于该分数的字符将不能识别。



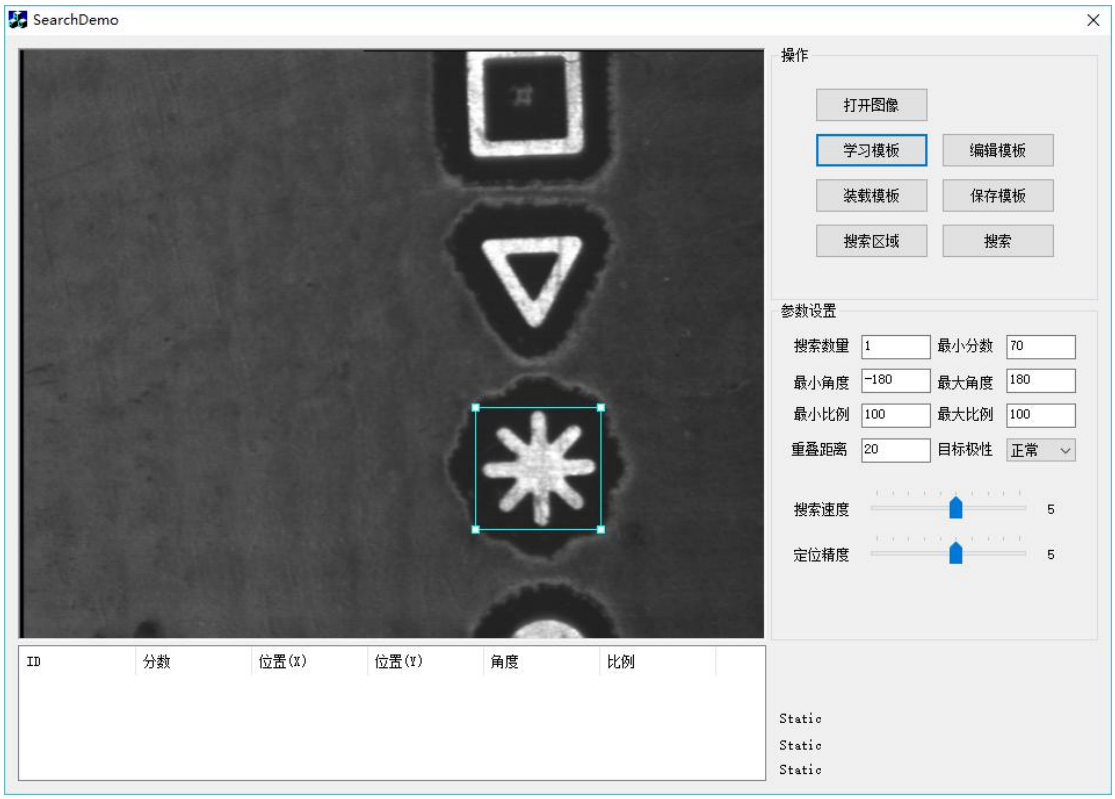




返回串：显示识别结果字符串文本。

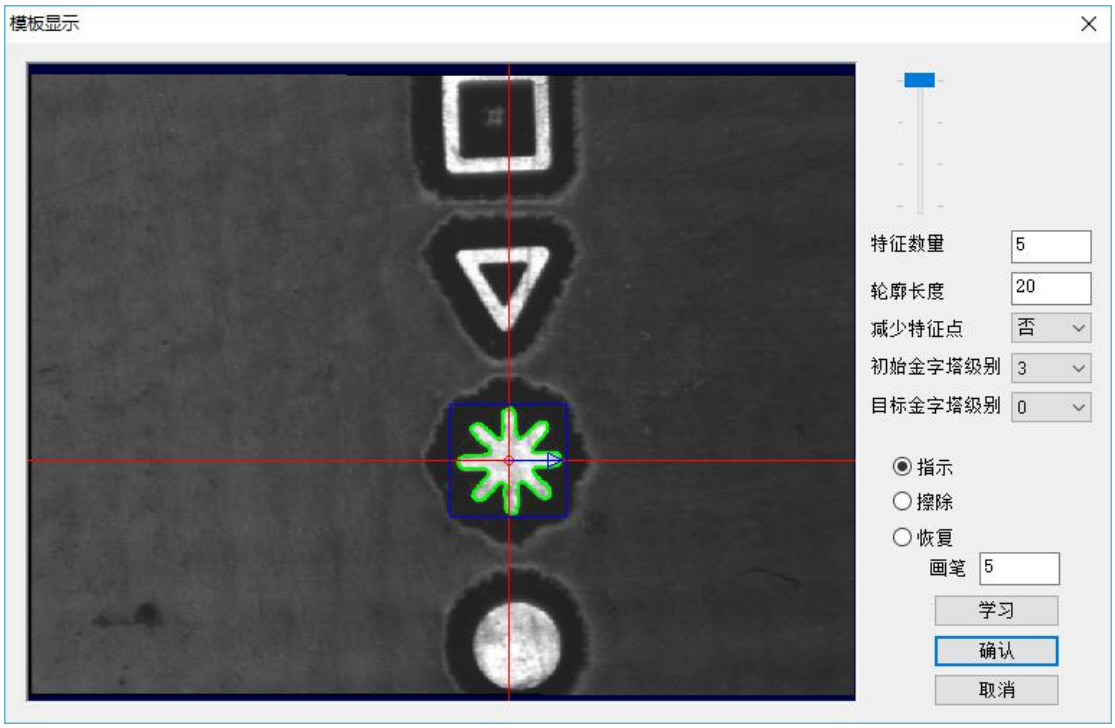
SearchDemo.exe 模板轮廓匹配定位（新版本）

以边缘轮廓特征作为模板，在图像中搜索形状相似的目标。



学习模板：以图像上的 ROI 所在位置学习模板轮廓。

编辑模板：点击“编辑”按钮将会弹出编辑模板对话框，在编辑模板对话框下可以对模板进行编辑。



特征数量：模板特征数量占学习区域特征数量的百分比。

轮廓长度：轮廓长度参数用于过滤，长度小于该值的轮廓将会被删除。

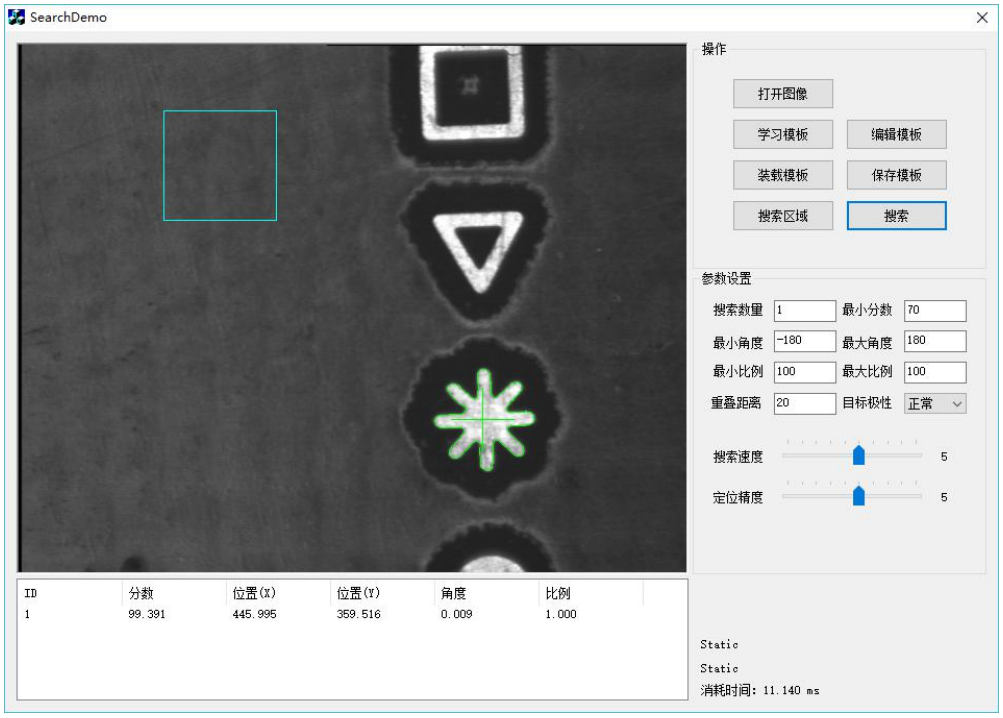
减少特征点：是否减少模版特征点数量。

金字塔级别：可预览当前图像模板层数，点击获取边缘轮廓按钮后会自动获取金字塔级别数值。

掩模编辑：点击“屏蔽”按钮可设置掩模图像屏蔽区域，设置完屏蔽区域后，需要点击“获取边缘轮廓”才能成功屏蔽所选区域，如上图中红色区域部分为被屏蔽，绿色部分为正在使用的边缘轮廓，点击“恢复”按钮可恢复图像中被屏蔽的区域。

画笔大小：设置擦除或者恢复画笔的尺寸大小。

学习：重新学习轮廓模板。



搜索数量：最多被允许搜索到的目标数量。

最小分数：分数表示目标和模板的相似程度，分数越高越相似，最大值 100 表示完全匹配，目标分数必须大于该值才会被搜索到，该参数值将会影响搜索速度。

角度：可以设置被搜索目标可能存在的角度范围，角度为目标相对于模板的角度，取值范围-180~180。

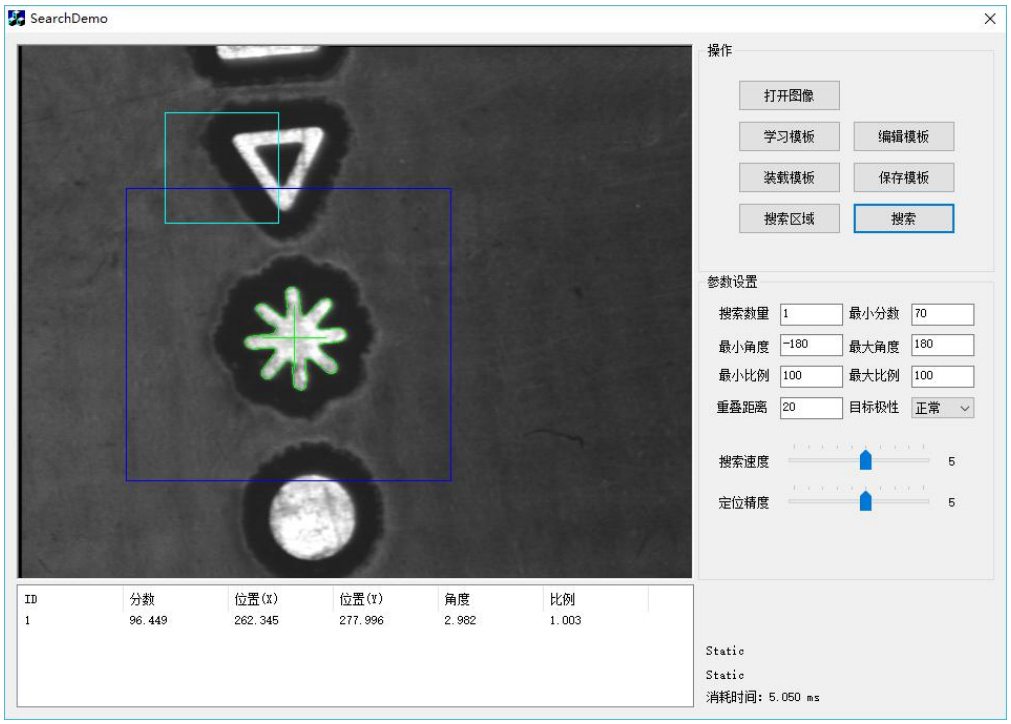
比例：可以设置被搜索目标可能存在的比例范围，比例为目标相对于模板的比例，取值范围80~120。

重叠距离：匹配多个目标时之间的最小距离不能小于重叠距离，小于则忽略。

匹配极性：可以设置正常和反转，正常表示目标和模板极性相同，反转则表示相反。

搜索速度：总共有 10 个等级，等级为 0~9,默认为 5，设置的搜素速度级别越高，识别度会有所下降。

定位精度：总共有 10 个等级，等级为 0~9,默认为 5，设置的定位精度级别越高，搜索速度会有所下降。



分数：匹配目标与模板的相似度。
位置：匹配目标相对于当前图像的坐标位置。
角度：匹配目标相对于模板的旋转角度。
比例：匹配目标相对于模板的缩放比例。

2 Bin_x64

2.1 等同 Bin 文件内功能。

3 Document 文档

CKVision.chm

CKVision 简介.pdf

版本说明.doc

CKVISION SDK 说明

4 Include 开发库头文件

CKAcmeTool.h	顶点测量	(CAcmeTool)	CKMeasure.dll
CKBarcode.h	一维码读取	(CReadBarcode)	CKReader.dll
CKBase.h	基础模块		CKBase.dll
CKBaseDef.h	导出/导入、数据结构定义		
CKBlob.h	斑点分析、图像对比		CKBlob.dll
CKBlobAnalyzer.h	斑点分析	(CBlobAnalyze)	
CKBlobData.h	Blob 数据	(CBlobData)	
CKBlobDef.h	Blob 定义		
CKCalibration.h	标定功能	(CCalibration)	CKCalibration.dll
CKCaliper.h	卡尺、间距测量	(CCaliper)	CKMeasure.dll
CKCharset.h	字符集	(CCharset)	CKReader.dll
CKColor.h	颜色		CKColor.dll
CKColorIdentify.h	颜色颜色识别	(CColorSamples、CColorIdentify)	
CKColorMonitor.h	颜色监测	(CColorMonitor)	
CKColorSample.h	颜色样本	(CColorSample)	
CKContour.h	轮廓检测、轮廓缺陷		CKContour.dll
CKContourDefect.h	轮廓缺陷	(CContourDefect)	
CKContourDetect.h	轮廓检测	(CContourDetect)	
CKDataMatrix.h	读取 DataMatrix 二维码	(CDataMatrix)	CKReader.dll
CKDotMatrix.h	圆形矩阵标定板	(CDotMatrix)	
CKEdgeTool.h	边缘点检测	(CEdgeTool)	CKMeasure.dll
CKFileStore.h	文件存储结构	(CFileStore)	CKBase.dll
CKFindBarcode.h	读取一维码	(CFindBarcode)	CKReader.dll
CKFindModel.h	形状模型搜索	(CFindModel)	CKLocate.dll
CKFitCircle.h	圆拟合工具	(CFitCircle)	CKMeasure.dll
CKFitLine.h	线拟合工具	(CFitLine)	
CKFrameTrans.h	坐标系变换	(CFrameTrans)	CKBase.dll

CKGDI.h	图形显示		CKGDI.dll
CKGdiBoxScan.h	旋转矩形框内扫描线	(CGdiBoxScan)	
CKGdiCircle.h	圆形	(CGdiCircle)	
CKGdiContour.h	轮廓图形	(CGdiContour)	
CKGdiEllipse.h	椭圆图形	(CGdiEllipse)	
CKGdiFigure.h	图形功能(基类)	(CGdiFigure)	
CKGdiFrame.h	坐标系显示	(CGdiFrame)	
CKGdiHistogram.h	直方图	(CGdiHistogram)	
CKGdiLine.h	线段图形	(CGdiLine)	
CKGdiMask.h	掩摸显示	(CGdiMask)	
CKGdiModel.h	模型轮廓显示	(CGdiModel)	
CKGdiPoint.h	点、十字显示	(CGdiPoint)	
CKGdiPolygon.h	多边形图形	(CGdiPolygon)	
CKGdiProfile.h	投影曲线边缘位置	(CGdiProfile)	
CKGdiRect.h	矩形框	(CGdiRect)	
CKGdiRing.h	圆环图形	(CGdiRing)	
CKGdiRingScan.h	圆环内扫描线	(CGdiRingScan)	
CKGdiRotBox.h	旋转矩形	(CGdiRotBox)	
CKGdiText.h	文本显示	(CGdiText)	
CKGdiType.h	模板类显示	(CGdiType)	
CKGdiView.h	图形视图窗口	(CGdiView)	CKGDI.dll
CKGeoMeas.h	基本几何测量	(CKVISION_API)	CKBase.dll
CKHasp.h	校验锁		
CKHistogram.h	直方图、分割阈值	(CHistogram)	
CKHSIThreshold.h	HSI 颜色抽取	(CHSIThreshold)	CKColor.dll

CKImage.h	图像基本功能	(CPrImage)	CKBase.dll
CKImgConve.h	图像转换、高级调整	(CKVISION_API)	CKBase.dll
CKImgFilter.h	图像滤波		
CKImgMorph.h	图像灰度形态学		
CKImgOpera.h	图像算术和逻辑		
CKImgTrans.h	图像变换（镜像、平移、旋转、缩放、等）		
CKLocate.h	形状匹配、识别定位		CKLocate.dll
CKMask.h	图像掩摸	(CMask)	CKBase.dll
CKMeasDef.h	测量定义		CKMeasure.dll
CKMeasure.h	测量		CKMeasure.dll
CKModel.h	模型特征点模板	(CModel)	CKLocate.dll
CKModelContour.h	模型轮廓	(CModelContour)	
CKNCMatch.h	灰度区域匹配	(CNCMatch)	
CKNCPat.h	灰度模板	(CNCPat)	
CKOverlay.h	覆盖图功能	(COverlay)	CKGDI.dll
CKPatInspect.h	基于图像对比缺陷检测	(CPatInspect)	CKBlob.dll
CKPixelStat.h	像素统计功能	(CPixelStat)	CKBase.dll
CKPointVector.h	坐标点容器	(CPointVector)	CKMeasure.dll
CKProfile.h	图像截面投影曲线	(CProfile)	CKMeasure.dll
CKReadDXF.h	读取 DXF 文件生成模板轮廓	(CReadDXF)	CKGDI.dll
CKReader.h	读取条码、字符		CKReader.dll
CKReadOcr.h	字符识别	(CReadOcr)	CKReader.dll
CKReadQRCode.h	读取 QR 码	(CReadQRCode)	
CKScanEdge.h	扫描边缘	(CScanEdge)	CKMeasure.dll
CKScanSpace.h	扫描间距	(CScanSpace)	
CKShapeMatch.h	边缘轮廓形状匹配(新)	(CShapeMatch)	CKLocate.dll
CKShapeModel.h	形状模板(新)	(CShapeModel)	CKLocate.dll
CKSharpAssess.h	图像清晰度评估	(CSharpAssess)	CKBase.dll

5 Install 运行库安装包

5.1 VC2008 运行包

5.2 加密锁驱动

6 Lib 开发库 Lib 文件

CKBase.lib
CKBlob.lib
CKCalibration.lib
CKColor.lib
CKContour.lib
CKGDI.lib
CKLocate.lib
CKMeasure.lib
CKReader.lib

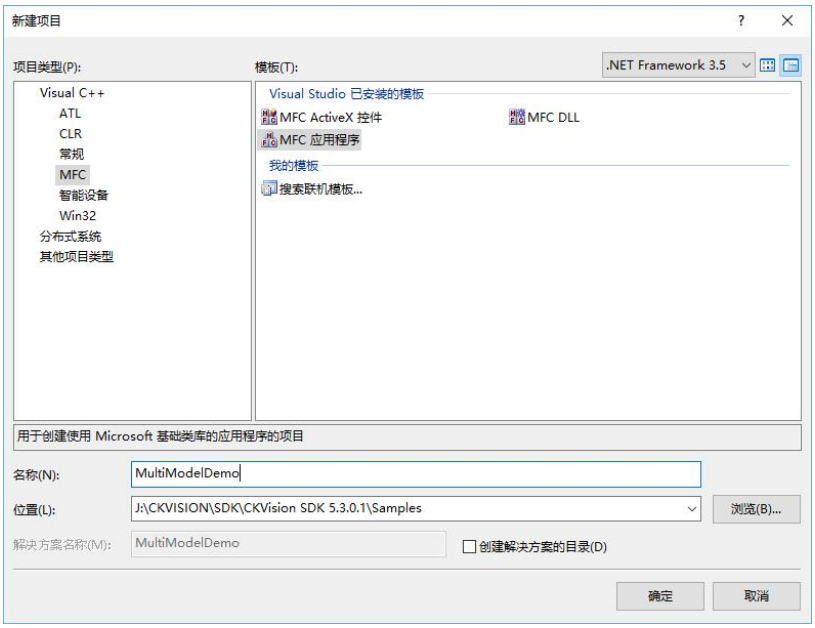
7 Lib_x64 开发库 64 位版本文件

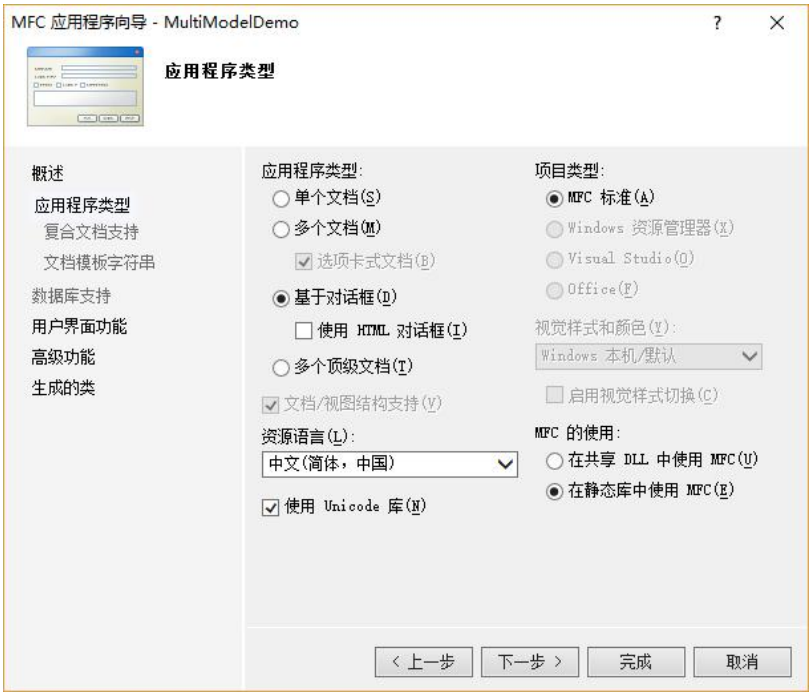
CKBase.lib
CKBlob.lib
CKCalibration.lib
CKColor.lib
CKContour.lib
CKGDI.lib
CKLocate.lib
CKMeasure.lib
CKReader.lib

8 Samples 功能 API 调用实例

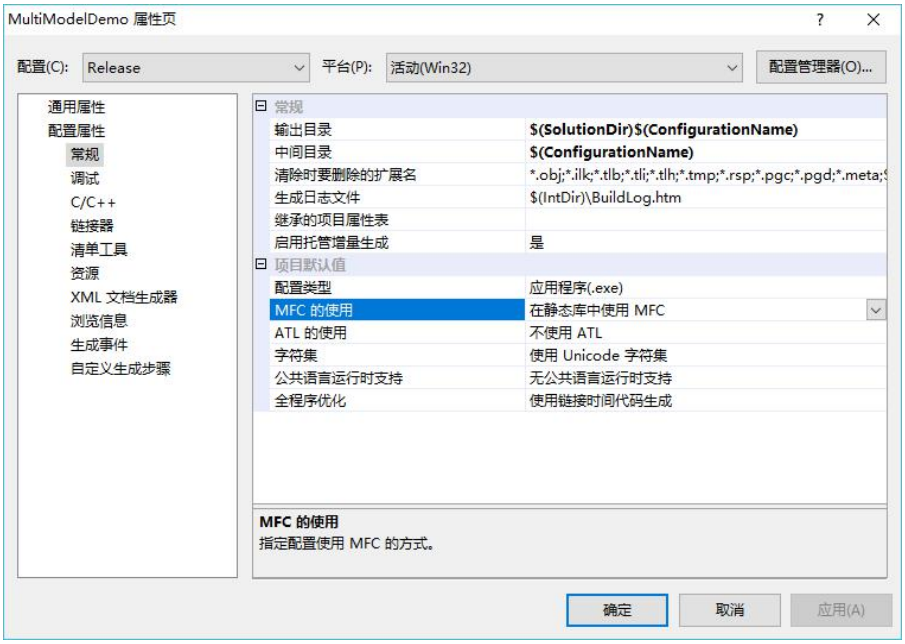
8.1)

以 vs 文件->新建->项目->创建 MFC 应用程式，基于对话框模、在静态库中使用 MFC 的实例。





项目属性



添加当前 CKVISION 开发功能对应的.h 与 Lib 文件。

选择到解决方案管理器->选择到当前的项目，展开列表，找到 Header Files->StdAfx.h 双击打开文件，然后添加:

```
#include "..\\..\\Include\\CKGDI.h"
#include "..\\..\\Include\\CKBase.h"
#include "..\\..\\Include\\CKLocate.h"

#ifdef_WIN64
    #pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
    #pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
    #pragma comment(lib, "..\\..\\Lib_x64\\CKLocate.lib")
#else
    #pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
    #pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
    #pragmacomment(lib, "..\\..\\Lib\\CKLocate.lib")
#endif

usingnamespaceCKVision;  // CKVISION 命名空间
```

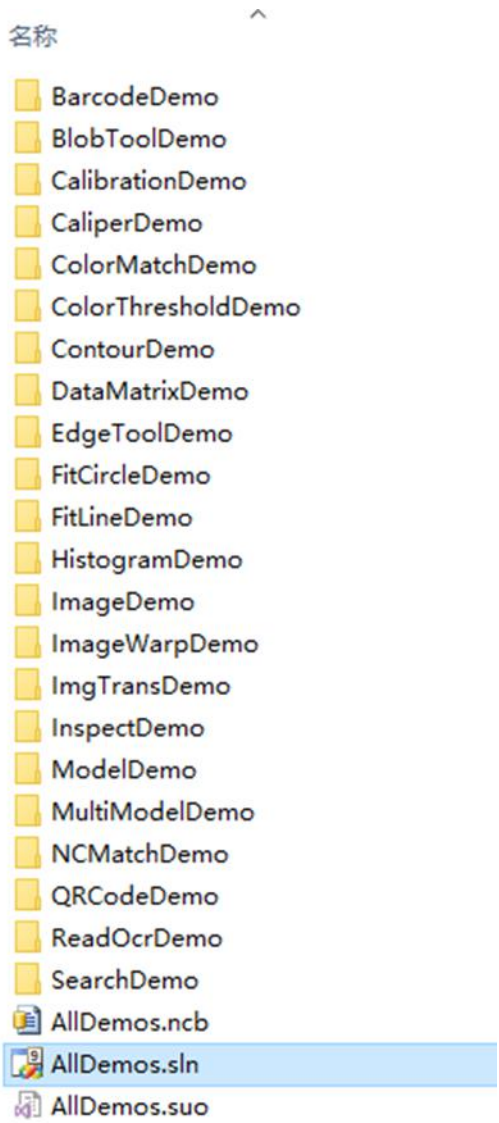
在程序入口处增加 InitLibrary 函数，用于初始化 CKVision 库,只有调用初始函数:

```
CKVision::InitLibrary();    // 初始化CKVision库
```

初始化之后才能正常其它图像处理的功能。

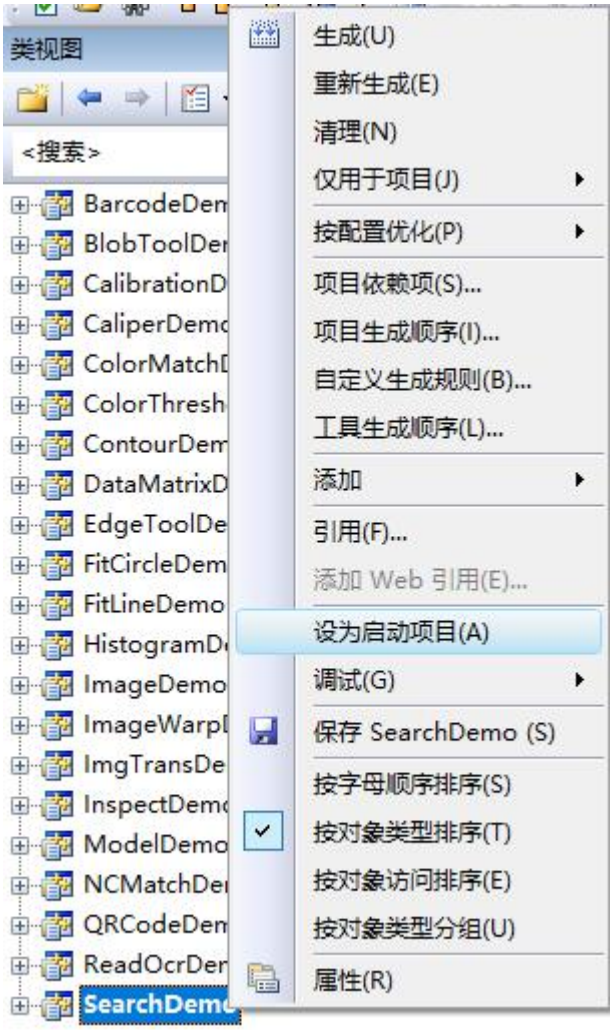
在程序退出终止处增加 ExitLibrary 函数，用于释放 CKVision 库:

```
CKVision::ExitLibrary();    // 退出 CKVision 库
```



以 vs 2008 以上版本打开 AllDemos.sln 加载功能实例文件。

打开工程文件后，可以选择类视图->选择当前需要查看的项目，右键鼠标->弹出菜单设置当前启动项目。

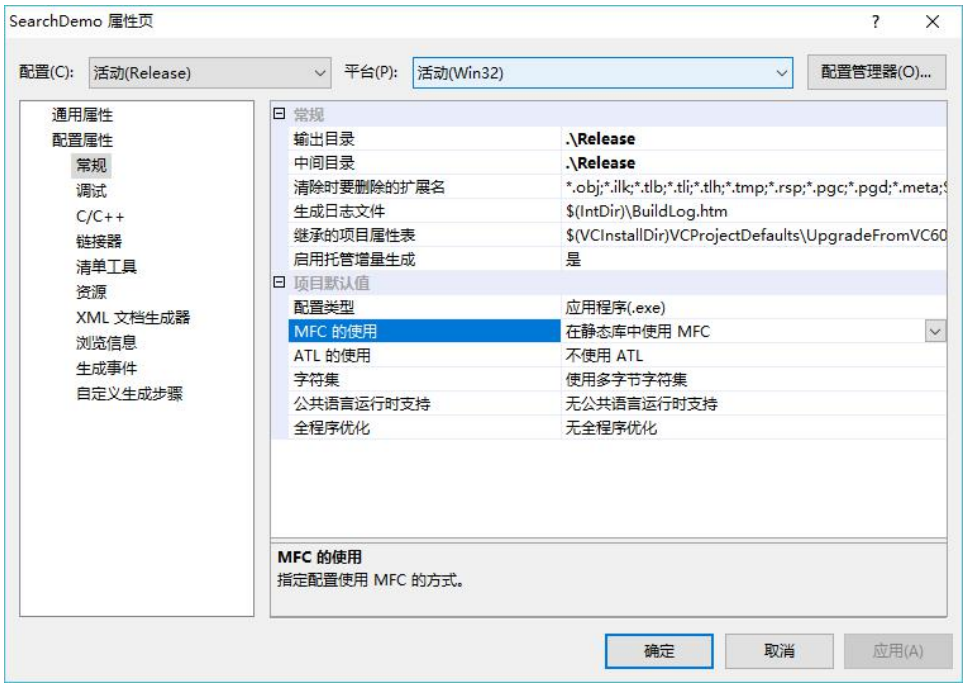


8.2)

->项目属性->配置属性->常规-> MFC 的使用

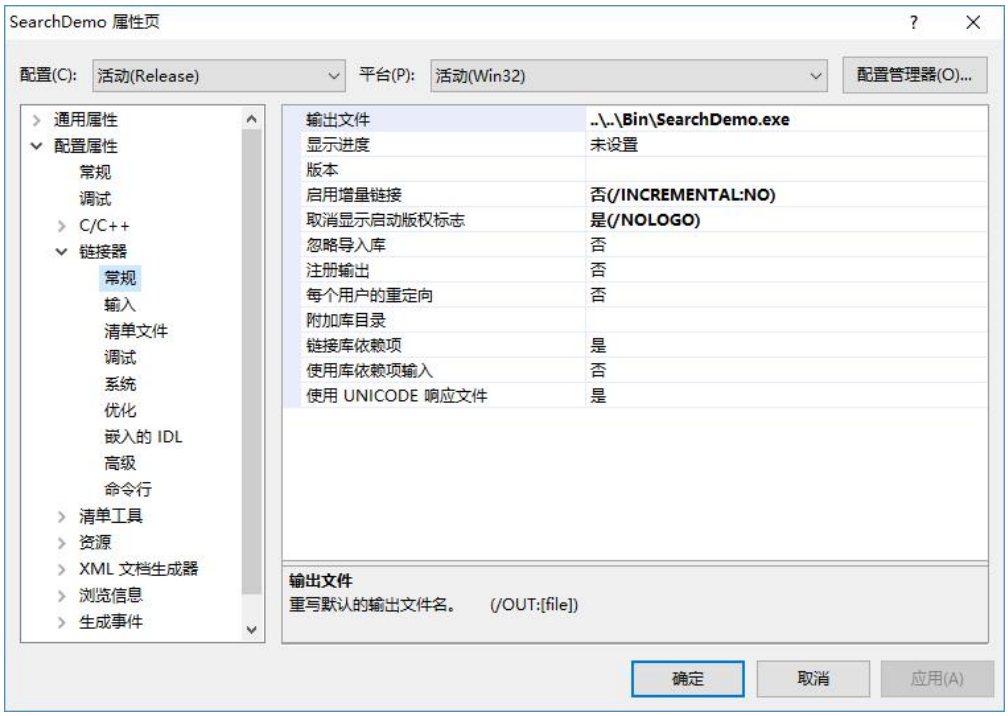
->在静态库中使用 MFC


类视图
<搜索>
Barcodel
BlobToo
Calibrati
CaliperD
ColorMa
ColorThr
ContourD
DataMat
EdgeToc
FitCircleD
FitLineDe
Histogra
ImageDe
ImageW
ImgTran
InspectD
ModelDe
NCMatch
QRCodel
ReadOcr
SearchDemo
生成(U)
重新生成(E)
清理(N)
仅用于项目(J)
按配置优化(P)
项目依赖项(S)...
项目生成顺序(I)...
自定义生成规则(B)...
工具生成顺序(L)...
添加
引用(F)...
添加 Web 引用(E)...
设为启动项目(A)
调试(G)
保存 SearchDemo (S)
按字母顺序排序(S)
按对象类型排序(T)
按对象访问排序(E)
按对象类型分组(U)
属性(R)



8.3)->项目属性->查看编译生成输出的文件路径。

连接器->输出文件.

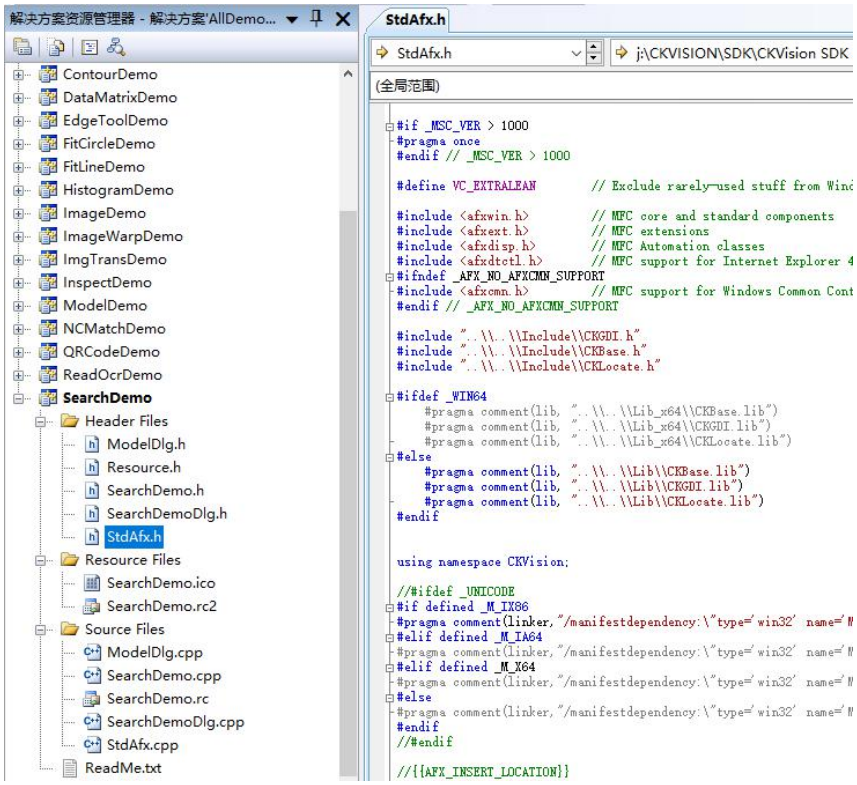


8.4)

添加当前 CKVISION 开发功能对应的.h 与 Lib 文件。

选择到解决方案管理器->选择到当前的项目，展开列表，找到 Header Files->StdAfx.h 双击打开文件，然后添加:

```
#include "..\\..\\Include\\CKGDI.h"
#include "..\\..\\Include\\CKBase.h"
#include "..\\..\\Include\\CKLocate.h"

#ifdef_WIN64
    #pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
    #pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
    #pragma comment(lib, "..\\..\\Lib_x64\\CKLocate.lib")
#else
    #pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
    #pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
    #pragmacomment(lib, "..\\..\\Lib\\CKLocate.lib")
#endif

usingnamespaceCKVision;  // CKVISION 命名空间
```



8.5)

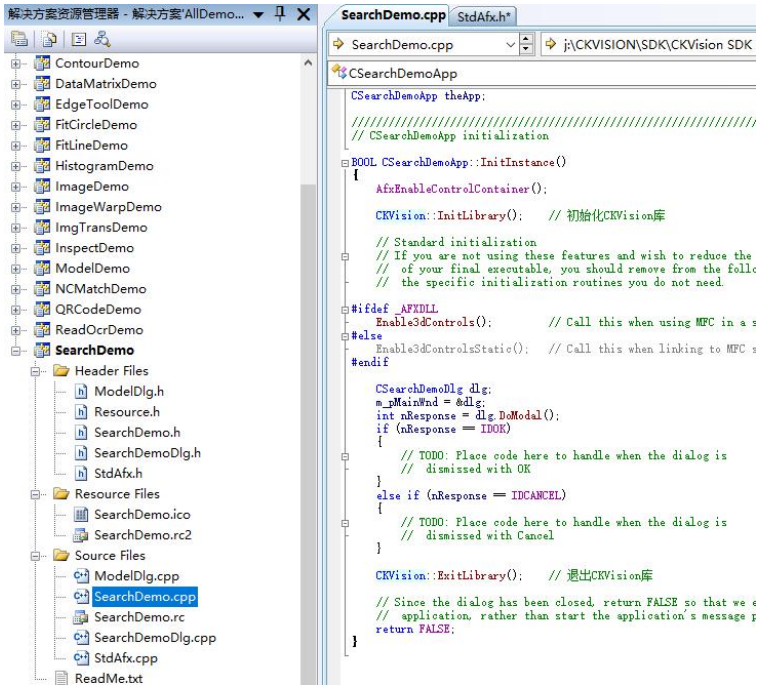
在程序入口处增加 InitLibrary 函数，用于初始化 CKVision 库,只有调用初始函数:

```
CKVision::InitLibrary();    // 初始化CKVision库
```

之后才能正常其它图像处理的功能。

在程序退出终止处增加 ExitLibrary 函数，用于释放 CKVision 库:

```
CKVision::ExitLibrary();    // 退出 CKVision 库
```



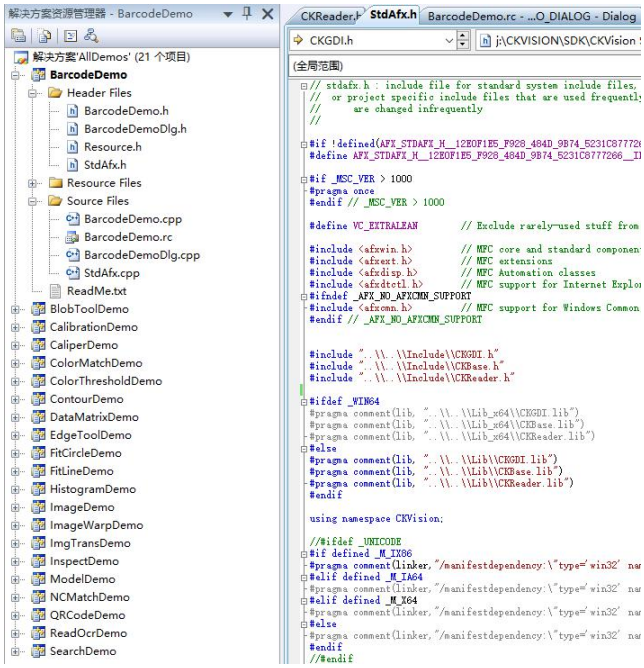
```
BOOL CQRCodeDemoApp::InitInstance()
{
    // 如果一个运行在 Windows XP 上的应用程序清单指定要
    // 使用 ComCtl32.dll 版本 6 或更高版本来启用可视化方式，
    //则需要 InitCommonControlsEx()。否则，将无法创建窗口。
    INITCOMMONCONTROLSEX InitCtrls;
    InitCtrls.dwSize = sizeof(InitCtrls);
    // 将它设置为包括所有要在应用程序中使用的
    // 公共控件类。
    InitCtrls.dwICC = ICC_WIN95_CLASSES;
    InitCommonControlsEx(&InitCtrls);

    CWinApp::InitInstance();

    CKVision::InitLibrary();    // 初始化CKVision库

    AfxEnableControlContainer();

    // 标准初始化
    // 如果未使用这些功能并希望减小
    // 最终可执行文件的大小，则应移除下列
    // 不需要的特定初始化例程
    // 更改用于存储设置的注册表项
    // TODO: 应适当修改该字符串，
    // 例如修改为公司或组织名
    SetRegistryKey(_T("应用程序向导生成的本地应用程序"));

    CQRCodeDemoDlg dlg;
    m_pMainWnd = &dlg;
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: 在此放置处理何时用
        //  “确定”来关闭对话框的代码
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: 在此放置处理何时用
        //  “取消”来关闭对话框的代码
    }

    CKVision::ExitLibrary();    // 退出CKVision库

    // 由于对话框已关闭，所以将返回 FALSE 以便退出应用程序，
    //  而不是启动应用程序的消息泵。
    return FALSE;
}
```

注:详细代码请打开对应的...Demo 功能实例。

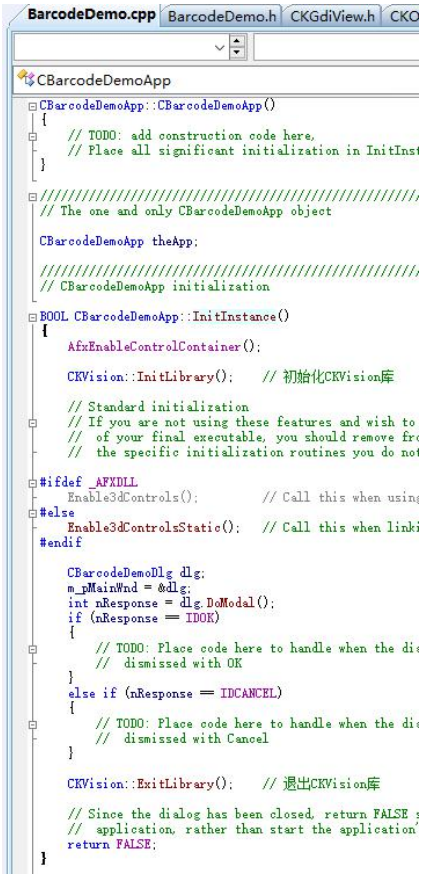
BarcodeDemo 一维码检测

1.) 在 StdAfx.h 的头文件中添加读取条码相关的文件链接。

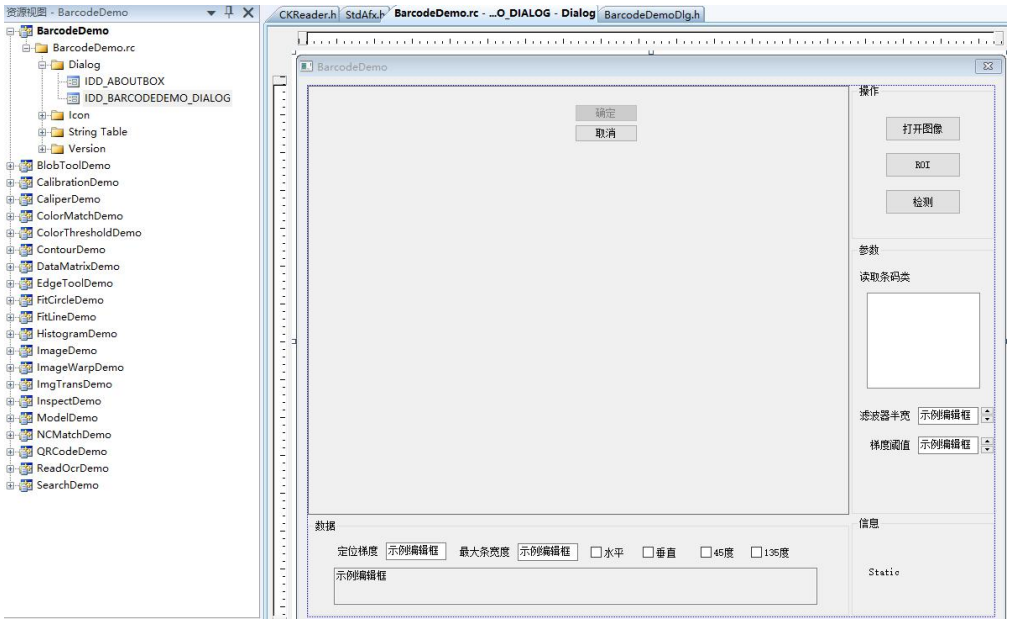
```
// .h 头文件
#include "..\\..\\Include\\CKGDI.h"
#include "..\\..\\Include\\CKBase.h"
#include "..\\..\\Include\\CKReader.h"
//.lib 文件
#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKReader.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKReader.lib")
#endif

usingnamespaceCKVision;
```



//初始化 CKVision 库



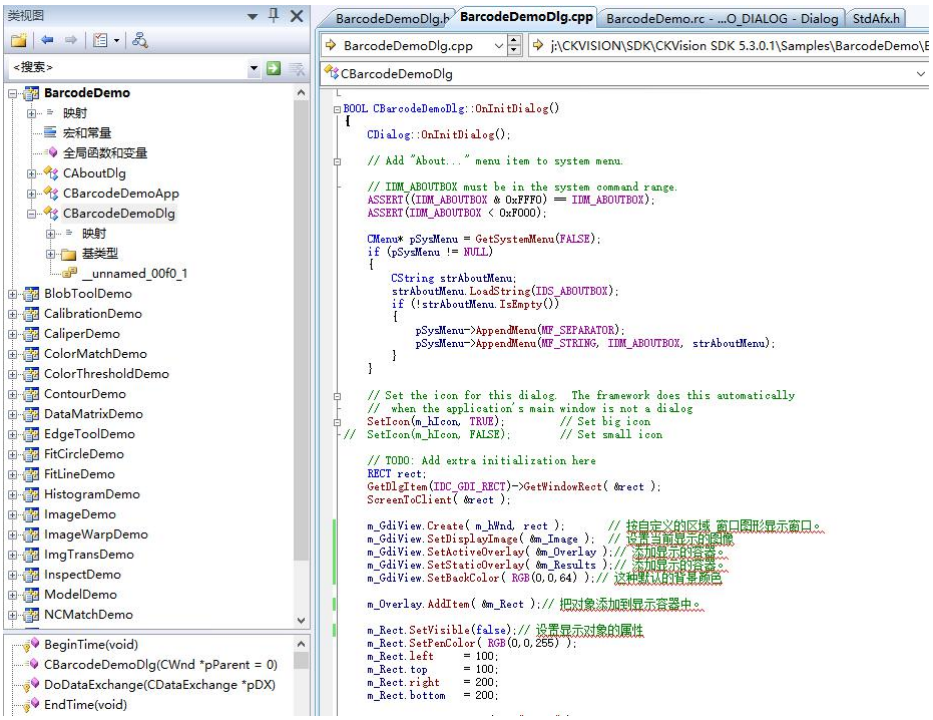
2.) 在资源视图 Dialog 中添加相应的界面操作。



3.) 在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
CPrImage       m_Image; // 基础图像类

CFindBarcode   m_FindBC;// 条码定位

CReadBarcode   m_Barcode;    // 读取条码

CGdiRect       m_Rect;       // 矩形框显示

COverlay       m_Overlay;    // 图像显示表面，前显示的动态图形,主要用于ROI 显示。

COverlay       m_Results;    // 图像显示表面，前显示的静态图形,主要用于检测结果生成图形显示。

CGdiView       m_GdiView;    // 图形视图窗口
```

4.) 在对话框窗口的.cpp 实现文件中添加相应的功能实现。




```
// 执行条码读取
voidCBarcodeDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    m_Barcode.Release();
    Overlay_DeleteAll(m_Results);

    // 一维码类型
    intnBarcodeType = 0;

    if( m_List1.GetSel(0) )
        nBarcodeType |= BARCODE_UPC_A;
    if( m_List1.GetSel(1) )
        nBarcodeType |= BARCODE_UPC_E;
    if( m_List1.GetSel(2) )
        nBarcodeType |= BARCODE_EAN_8;
    if( m_List1.GetSel(3) )
        nBarcodeType |= BARCODE_EAN_13;
    if( m_List1.GetSel(4) )
        nBarcodeType |= BARCODE_CODE_39;
    if( m_List1.GetSel(5) )
        nBarcodeType |= BARCODE_CODE_93;
```

```
if( m_List1.GetSel(6) )
     nBarcodeType |= BARCODE_CODE_128;
if( m_List1.GetSel(7) )
     nBarcodeType |= BARCODE_INTERLEAVED_2_5;

m_Barcode.SetReadType( nBarcodeType );
m_Barcode.SetFilterHalf( GetDlgItemInt(IDC_EDIT1) );
m_Barcode.SetThreshold( GetDlgItemInt(IDC_EDIT2) );

intfx = 0;
if( IsDlgButtonChecked(IDC_CHECK1) )
     fx|=0x01;
if( IsDlgButtonChecked(IDC_CHECK2) )
     fx|=0x02;
if( IsDlgButtonChecked(IDC_CHECK3) )
     fx|=0x04;
if( IsDlgButtonChecked(IDC_CHECK4) )
     fx|=0x08;

m_FindBC.SetMaxCount( 20 );
m_FindBC.SetOrientation( fx );

m_FindBC.SetThreshold( GetDlgItemInt(IDC_EDIT4) );
m_FindBC.SetMaxSpace( GetDlgItemInt(IDC_EDIT3) );

// 开始计算时间
BeginTime();

// 执行条码定位
if( m_Rect.GetVisible() ) {
     m_FindBC.Execute( m_Image, m_Rect );
} else {
     m_FindBC.Execute( m_Image, MaxROI );
}

ROTRECTrc, *pRect;
for( inti=0; i<m_FindBC.GetNumResults(); i++ )      {
     pRect = m_FindBC.GetCodeBorder(i);
     rc.center = pRect->center;
     rc.angle = pRect->angle;
     rc.width = pRect->width+10;
```

```
            rc.height = min(max(pRect->height-50,10),80);
            // 指定位置读取条码
            if( m_Barcode.Execute( m_Image, rc ) ) {
                CGdiRotBox* p1 = newCGdiRotBox(*pRect);
                if( p1!=NULL ) {
                    p1->Offset(0.5,0.5);
                    p1->SetPenWidth( 2 );
                    if( m_Barcode.GetCodeLen()>0 )
                        p1->SetPenColor( RGB(0,255,0) );
                    else
                        p1->SetPenColor( RGB(255,0,0) );
                    m_Results.AddItem(p1);// 添加显示图形到画面上显示
                }
                break;
            }
        }

        // 结束计算时间周期
        EndTime();

        SetDlgItemText( IDC_CODE_TEXT, m_Barcode.GetCodeText() );

        m_GdiView.Redraw();     // 视图刷新显示
}
```

BlobToolDemo 斑点分析

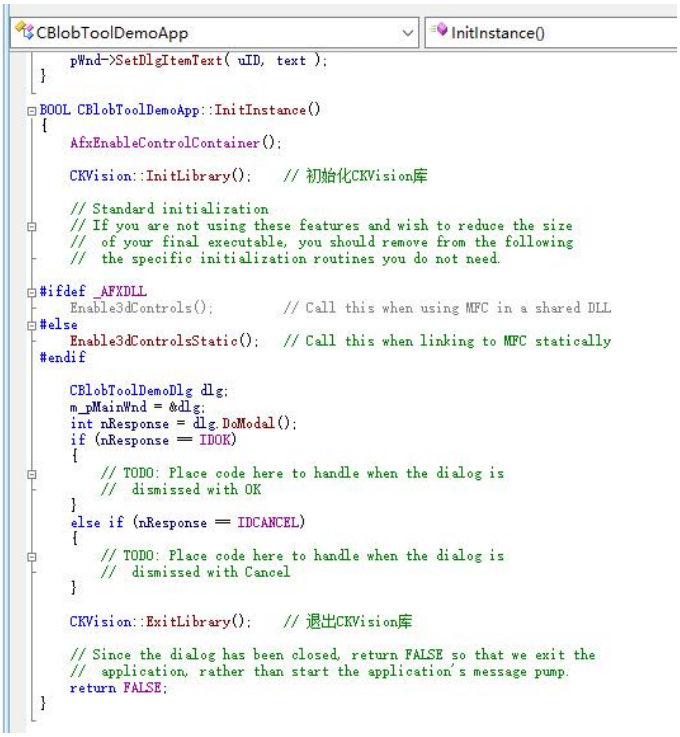
1.)在 StdAfx.h 的头文件中添加读取条码相关的文件链接。

```
#include "..\\..\\Include\\CKGDI.h"
#include "..\\..\\Include\\CKBase.h"
#include "..\\..\\Include\\CKBlobAnalyzer.h"

#ifdef _WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBlob.lib")
#else
#pragma comment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib\\CKBase.lib")
```

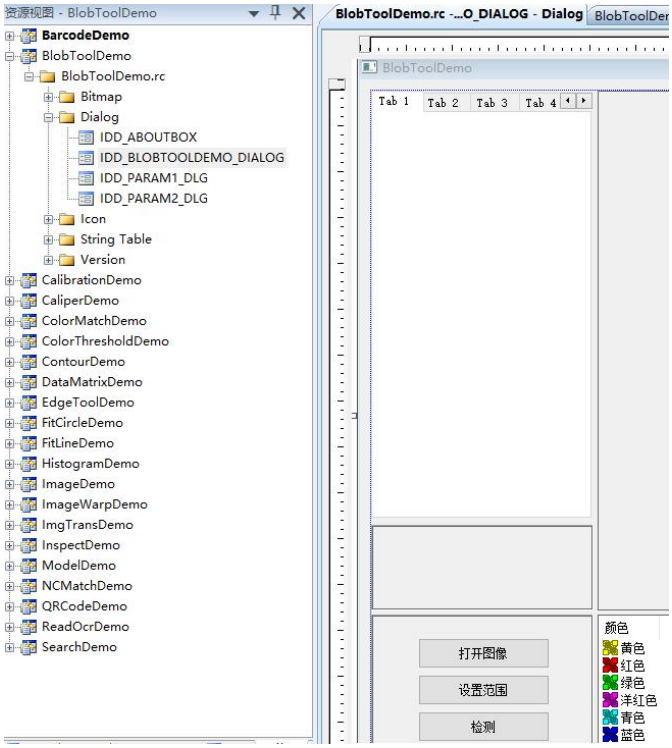
```
#pragma comment(lib, "..\\..\\Lib\\CKBlob.lib")
#endif

using namespace CKVision;
```

//在程式入口和退出的地方增加初始化和释放 CKVISION 库。

```
BOOLCBlobToolDemoApp::InitInstance()

CKVision::InitLibrary();     // 初始化 CKVision 库.
//.....

CKVision::ExitLibrary();     // 退出 CKVision 库
```



2.)在资源视图 Dialog 中添加相应的界面操作。



在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
voidFilter( intnMeasure, doubledMin, doubledMax, BOOLbInvert );// 过滤
voidSort( intnMeasure, intnOrder );// 排序

voidThresholdImage( INTvalue );// 设置二值化阈值显示
voidThresholdDraw( BOOLdraw );// 显示二值化图像

// 数据结果插入到列表中显示
voidInsertBase( int&nColNum );
voidInsertBound( int&nColNum );
voidInsertMoment( int&nColNum );
voidInsertConvex( int&nColNum );
voidInsertMinBox( int&nColNum );

// 基础图像类
CPrImage      m_Image;
CPrImage      m_TImage;

// 掩摸图
CMask         m_Label;
// ROI 检测区域
```

```
    CGdiRect      m_Rect;
    // Blob 分析工具类
    CBlobAnalyzer m_blobTool;

    // 图像显示表面,
    COverlay      m_Overlay;  //前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;
    CGdiView      m_GdiView; // 图形视图窗口

    // 参数设置
    CParam1Dlg         m_Param1Dlg;
    CParam2Dlg         m_Param2Dlg;
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

// 对话框窗口初始化函数中

```
BOOLCBlobToolDemoDlg::OnInitDialog()
{
        // TODO: Add extra initialization here
    CRectdrc;
    m_Tab1.GetWindowRect( &drc );
    ScreenToClient( &drc );
    drc.DeflateRect( 10, 25, 10, 10 );

    m_Tab1.InsertItem( 0, "参数设置" );
    m_Tab1.InsertItem( 1, "其它功能" );
    m_Tab1.SetCurSel( 0 );
    // 创建参数设置窗口
    m_Param1Dlg.Create( IDD_PARAM1_DLG, this );
    m_Param1Dlg.ShowWindow( SW_SHOW );
    m_Param1Dlg.MoveWindow( &drc );

    m_Param2Dlg.Create( IDD_PARAM2_DLG, this );
    m_Param2Dlg.MoveWindow( &drc );

    m_List1.SetExtendedStyle( 0x20 );

    RECTrect;
    GetDlgItem(IDC_GDI_RECT)->GetWindowRect( &rect );
    ScreenToClient( &rect );
    // 窗口图形显示视图
```

```
    m_GdiView.Create( m_hWnd, rect );
    m_GdiView.SetBackColor( RGB(0,0,64) );
    m_GdiView.SetDisplayImage( &m_Image );// 显示当前的图像
    m_GdiView.SetActiveOverlay( &m_Overlay );// ROI 显示
    m_GdiView.SetStaticOverlay( &m_Results );// 结果图形显示

    m_Overlay.AddItem( &m_Rect );//将需要显示的图形添加到覆盖图容器中。

    m_Rect.left        = 100;
    m_Rect.top         = 100;
    m_Rect.right   = 500;
    m_Rect.bottom = 400;
    m_Rect.SetPenColor( RGB(255,0,0) );
    m_Rect.SetVisible( false );

    m_Disp1.SetCheck( 1 );
}

// 在执行按钮中
voidCBlobToolDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    m_blobTool.SetBlobType( m_Param1Dlg.m_Combo1.GetCurSel() );
    m_blobTool.SetConnexity( m_Param1Dlg.m_Combo2.GetCurSel() );
    m_blobTool.SetThreshold( m_Param1Dlg.GetDlgItemInt( IDC_EDIT1 ) );
    m_blobTool.SetLimitArea( m_Param1Dlg.GetDlgItemInt( IDC_EDIT2 ) );
    m_blobTool.SetFeatures( m_Param1Dlg.GetFeature() );

    BeginTime();

    //直方图是否自动分割二值化阈值
    CHistogramhist;
    hist.SetAnalyse(Analyse_Threshold);// 设置分析分割阈值参数

    if( m_Rect.GetVisible() ) {
        if( m_Param1Dlg.m_bAutom.GetCheck() ) {// 自动计算二值化阈值
            hist.Execute( m_Image, m_Rect );
            m_blobTool.SetThreshold( hist.GetThreshold() );
            m_Param1Dlg.UpdateThreshold( hist.GetThreshold() );
        }
        m_blobTool.Execute( m_Image, m_Rect );// 执行Blob 分析
```

```
    } else {
        if( m_Param1Dlg.m_bAutom.GetCheck() ) {
            hist.Execute( m_Image, MaxROI );
            m_blobTool.SetThreshold( hist.GetThreshold() );
            m_Param1Dlg.UpdateThreshold( hist.GetThreshold() );
        }
        m_blobTool.Execute( m_Image, MaxROI );// 执行Blob 分析
    }
    EndTime();

    intnColNum=0;
    m_List1.DeleteAllItems();
    Overlay_DeleteAll(m_Results);
    while(m_List1.DeleteColumn(0));

    // 添加基本特征
    InsertBase( nColNum );

    // 添加外接矩形特征
    if( m_blobTool.GetFeatures()&BLOB_FEATURE_BOUND )
        InsertBound( nColNum );

    // 添加力主轴特征
    if( m_blobTool.GetFeatures()&BLOB_FEATURE_AXIS )
        InsertMoment( nColNum );

    // 添加凸包相关特征
    if( m_blobTool.GetFeatures()&BLOB_FEATURE_CONVEX )
        InsertConvex( nColNum );

    // 添加最小外框特征
    if( m_blobTool.GetFeatures()&BLOB_FEATURE_FETBOX )
        InsertMinBox( nColNum );

    m_GdiView.Redraw();// 刷新视图
}
```

CalibrationDemo 标定校准

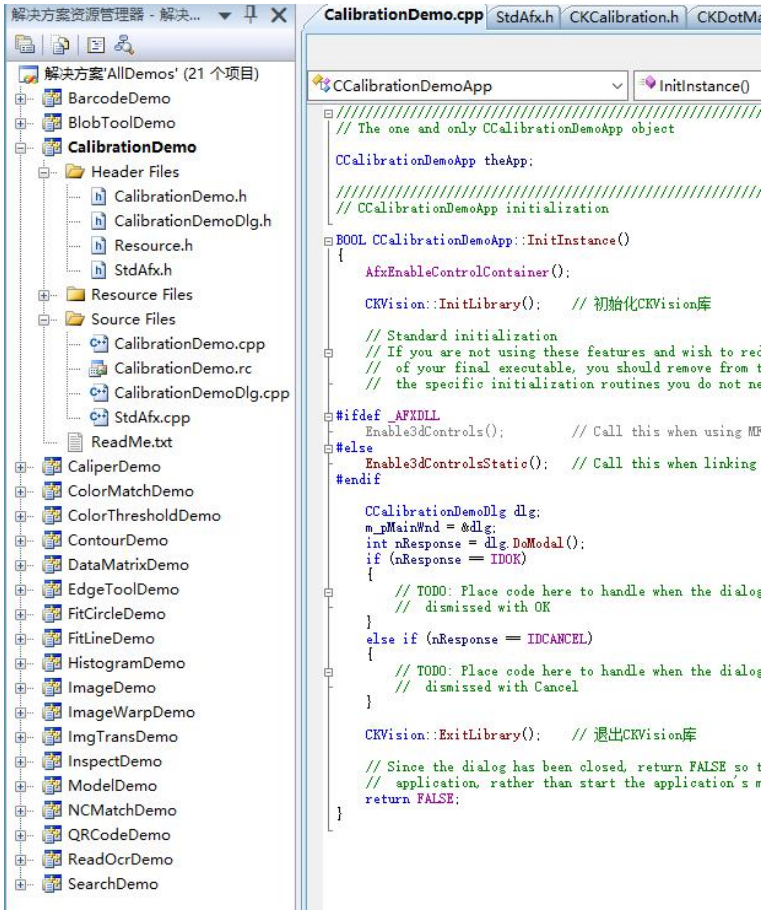
注意：CKVISION SDK 添加 .h 与 lib 文件、初始方法步骤与以上功能相同。

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKDotMatrix.h"
#include"..\\..\\Include\\CKCalibration.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKCalibration.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKCalibration.lib")
#endif

usingnamespaceCKVision;

// CalibrationDemo.cpp : Defines the class behaviors for the application.
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
BOOLCCalibrationDemoApp::InitInstance()

CKVision::InitLibrary();    // 初始化 CKVision 库
// .....
// 退出
CKVision::ExitLibrary();    // 退出 CKVision 库
```



在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

```
// CalibrationDemoDlg.h : header file
```

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
// CCalibrationDemoDlg dialog
    CPvImage      m_ImageSrc;// 来源图像
    CPvImage      m_ImageDst;// 校正后的图像
    COverlay      m_Overlay;// 图像显示表面，前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;// 图像显示表面，前显示的静态图形,主要用于检测结果生成图形显示。
    CGdiView      m_GdiView;// 图形显示视图窗口
    CGdiRect      m_Roi;        // 矩形区域
    CDotMatrix        m_DotMatrix;  //获取圆形矩阵点位置数据
    CCalibration  m_Calibration;     // 执行标定、图像畸变校正、九点标定。
//……


// CalibrationDemoDlg.cpp : implementation file
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
//CCalibrationDemoDlg message handlers
BOOLCCalibrationDemoDlg::OnInitDialog()
{
    //...
        // TODO: Add extra initialization here

    CRectrect;
    GetDlgItem(IDC_VIEW_RECT)->GetWindowRect( &rect );
    ScreenToClient( &rect );

    m_GdiView.Create( m_hWnd, rect );              // 按自定义的区域创建图形显示视图窗口。
    m_GdiView.SetBackColor( RGB(0,0,64) );         // 设置默认的背景颜色
    m_GdiView.SetDisplayImage( &m_ImageDst ); // 设置当前显示的图像
    m_GdiView.SetActiveOverlay( &m_Overlay ); // 设置当前显示的覆盖图ROI 图形显示。
    m_GdiView.SetStaticOverlay( &m_Results ); // 设置当前显示的覆盖图结果图形显示。

    m_Overlay.AddItem( &m_Roi );                   //将需要显示的图形添加到覆盖图容器中。

    //...
}
// 获取图像上的标定点
voidCCalibrationDemoDlg::OnCheckDot()
{
    // TODO: Add your control notification handler code here
    BeginTime();

    m_DotMatrix.SetDotType( m_Combo1.GetCurSel() );
    m_DotMatrix.SetMinArea( GetDlgItemInt(IDC_EDIT1) );
    m_DotMatrix.SetMaxArea( GetDlgItemInt(IDC_EDIT2) );

    CHistogramhistTool;
    histTool.SetAnalyse( Analyse_Threshold );
    if( m_Roi.GetVisible() ) {
        histTool.Execute( m_ImageSrc, m_Roi );
        m_DotMatrix.SetThreshold( histTool.GetThreshold() );
    if( !m_DotMatrix.Execute( m_ImageSrc, m_Roi ) ) {
            AfxMessageBox( _T("获取标定点失败！") );
        }
    } else {
        histTool.Execute( m_ImageSrc, MaxROI );
```

```
        m_DotMatrix.SetThreshold( histTool.GetThreshold() );
        if( !m_DotMatrix.Execute( m_ImageSrc, MaxROI ) ) {
            AfxMessageBox( _T("获取标定点失败！") );
        }
    }

    Overlay_DeleteAll(m_Results);
    DPNT* points = m_DotMatrix.GetDotData();
    for( inti=0; i<m_DotMatrix.GetDotCount(); i++ ) {
        CGdiPoint* p1 = newCGdiPoint(points[i]);
        if( p1!=NULL ) {
            p1->SetStyle(1);
            p1->SetPenColor(RGB(0,255,0));
            m_Results.AddItem( p1 );
        }
    }

    EndTime();

    SelectView( 0 );
}

// 执行标定
voidCCalibrationDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    UpdateData();

    BeginTime();

    m_Calibration.SetActualSpace( m_dUnitSpaceX );
    m_Calibration.SetCalibrationType( m_Combo2.GetCurSel() );

    // 执行标定
    if( !m_Calibration.Execute( m_DotMatrix.GetDotData(), m_DotMatrix.GetDotCount() ) )
    {
        AfxMessageBox( "标定失败！" );
        return;
    }

    // 执行多个点标定功能（九点标定）、坐标系变换
```

```
    // pointSrc    来源点，图像上的坐标点
    // pointDst    目标点，实际单位的坐标点
    // nCount 点的数量
    // bool Execute( const DPNT* pointSrc, const DPNT* pointDst, int nCount );


    if( !m_Calibration.CreateLUTable( m_ImageSrc.GetWidth(), m_ImageSrc.GetHeight() ) )
    {
        AfxMessageBox( "创建图像校正查找表失败！" );
        return;
    }

    EndTime();

    CStringtext;

    text.Format( _T("%0.9f"), m_Calibration.GetPixelScale() );
    SetDlgItemText( IDC_PIXEL_SCALE_X, text );

    SelectView( 0 );
}
```

//……详情请打开实例参考。

CaliperDemo 卡尺测量、间距检测

```
// stdafx.h : include file for standard system include files,or project specific include
// files that are used frequently,butare changed infrequently
```

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKCaliper.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKMeasure.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKMeasure.lib")
```

```
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
// CaliperDemo.cpp : Defines the class behaviors for the
application.BOOLBOOLCCaliperDemoApp::InitInstance()
{
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
}
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
// CaliperDemoDlg.h : header file
// CCaliperDemoDlg dialog

    CPrImage      m_Image;// 基础图像
    CGdiRotBox    m_Rect;// ROI 检测区域
    CCaliper      m_Caliper;//卡尺测量、间距检测、
    COverlay      m_Overlay;// 图像上层显示表面, 前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;// 图像上层显示表面, 前显示的静态图形,主要用于检测结果生成图形
显示。
    CGdiView      m_GdiView;// 图形显示视图窗口
    CGdiProfile   m_Profile;// 投影曲线边缘位置显示
    COverlay      m_ProOver;// 曲线图形显示
    CGdiView      m_ProView;// 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// CalibrationDemoDlg.cpp : implementation file
// CCaliperDemoDlg message handlers

BOOLCCaliperDemoDlg::OnInitDialog()
{
    //...
```

```
    RECTrect;
    GetDlgItem(IDC_GDI_RECT)->GetWindowRect( &rect );
    ScreenToClient( &rect );

    m_GdiView.Create( m_hWnd, rect );              // 按自定义的区域创建图形显示视图窗口。
    m_GdiView.SetBackColor( RGB(0,0,64) );         // 设置默认的背景颜色
    m_GdiView.SetDisplayImage( &m_Image );         // 设置当前显示的图像
    m_GdiView.SetActiveOverlay( &m_Overlay ); // 设置当前显示的覆盖图ROI 图形显示。
    m_GdiView.SetStaticOverlay( &m_Results ); // 设置当前显示的覆盖图结果图形显示。

    m_Overlay.AddItem( &m_Rect );                  //将需要显示的图形添加到覆盖图容器中。
    //...
}
// 执行测量
voidCCaliperDemoDlg::OnExecuct()
{
    // TODO: Add your control notification handler code here
    m_Caliper.SetPolarity1( m_Combo1.GetCurSel() );
    m_Caliper.SetPolarity2( m_Combo2.GetCurSel() );
    m_Caliper.SetLocation( m_Combo3.GetCurSel() );
    m_Caliper.SetThreshold( GetDlgItemInt(IDC_EDIT1) );
    m_Caliper.SetFilterHalf( GetDlgItemInt(IDC_EDIT2) );

    BeginTime();

    // 执行测量
    m_Caliper.Execute( m_Image, m_Rect );

    EndTime();

    // 创建投影曲线
    m_Profile.Create( m_Caliper.GetLength() );
    m_Profile.SetCurve1( m_Caliper.GetProjection() );
    m_Profile.SetCurve2( m_Caliper.GetStrengths() );
    m_Profile.SetThreshold( double(GetDlgItemInt(IDC_EDIT1)) );

    // 建立坐标系变换
    CFrameTranstrans( m_Rect );

    intitem;
    CStringtext;
```

```
CaliperData* data;
m_List1.DeleteAllItems();
Overlay_DeleteAll(m_Results);// 清除已有的结果显示覆盖图形

for( inti=0; i<m_Caliper.GetCaliperCount(); i++ ) {
     data = m_Caliper.GetCaliperData( i );//获取数据
     text.Format( "%d", i+1 );
     item = m_List1.InsertItem( i, text );
     text.Format( "%0.2f", data->Width );
     m_List1.SetItemText( item, 1, text );
     text.Format( "%0.2f", data->Center.x );
     m_List1.SetItemText( item, 2, text );
     text.Format( "%0.2f", data->Center.y );
     m_List1.SetItemText( item, 3, text );
     m_Profile.AddEdge( data->Edge1.Distance );
     m_Profile.AddEdge( data->Edge2.Distance );
     CGdiLine* p1 = newCGdiLine(
          data->Edge1.Distance, 0,
          data->Edge1.Distance,
          m_Rect.height );
     if( p1!=NULL ) {
          p1->Transform( &trans );
          p1->SetPenColor( RGB(0,255,0) );
          m_Results.AddItem( p1 );// 添加到结果显示图形表面
     }

     CGdiLine* p2 = newCGdiLine(
          data->Edge2.Distance, 0,
          data->Edge2.Distance,
          m_Rect.height );
     if( p2!=NULL ) {
          p2->Transform( &trans );
          p2->SetPenColor( RGB(0,255,0) );
          m_Results.AddItem( p2 );
     }

     CGdiLine* p3 = newCGdiLine(
               data->Edge1.Position,
               data->Edge2.Position );
     if( p3!=NULL ) {
          p3->SetPenColor( RGB(0,255,0) );
```

```
                m_Results.AddItem( p3 );
            }
        }

        m_GdiView.Redraw();// 刷新显示视图
        m_ProView.Redraw();
}
```

//……详情请打开实例参考。

ColorMatchDemo 色彩匹配

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKColor.h"

//#include "..\\..\\Include\\CKColorIdentify.h"     // 颜色识别
//#include "..\\..\\Include\\CKColorMonitor.h" // 颜色监测

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKColor.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKColor.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
// CColorMatchDemoApp initialization

BOOLCColorMatchDemoApp::InitInstance()
{
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...
```

```
    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库

}
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
// ColorMatchDemoDlg.h : header file

    CPrImage      m_Image; // 基础图像
    CGdiRect      m_Rect;       // 矩形区域ROI
    CGdiCircle          m_Roi;          // 圆形区域ROI
    COverlay      m_Overlay;    // 图像上层显示表面, 前显示的动态图形,主要用于ROI 显示。
    CGdiView      m_Display;    // 图形显示视图窗口

    CColorSamples m_Sample;// 颜色样本集合
    CColorIdentify     m_Identify;   // 颜色识别

    COverlay      m_sOverlay;
    CGdiView      m_sDisplay;
    CGdiHistogram m_sHistogram; //学习样本直方图显示

    COverlay      m_tOverlay;
    CGdiView      m_tDisplay;
    CGdiHistogram m_tHistogram;//匹配样本直方图显示
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// ColorMatchDemoDlg.cpp : implementation file

// CColorMatchDemoDlg message handlers

BOOLCColorMatchDemoDlg::OnInitDialog()
//...

// 颜色识别
voidCColorMatchDemoDlg::OnExecute()
```

```
{
    // TODO: Add your control notification handler code here
    doubledMatchScore = 0;

    m_Identify.SetSensitivity( m_Combo1.GetCurSel() );
    m_Identify.SetSatThreshold( GetDlgItemInt(IDC_EDIT1) );

    BeginTime();

    // 颜色识别
    m_Identify.Execute( m_Image, m_Roi );

    // 识别当前颜色与颜色样本的区别
    m_Identify.Identify( m_Sample, dMatchScore );

    EndTime();

    CStringtext;
    text.Format( "%f", dMatchScore );
    SetDlgItemText( IDC_DATA1, text );

    m_tHistogram.SetValues(
        m_Identify.GetColorValues(),
        m_Identify.GetColorCount() );

    m_tDisplay.Redraw();//识别样本直方图刷新显示
}
```

//……详情请打开实例参考。

ColorThresholdDemo 彩色二值化

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKColor.h"
//#include "..\\..\\Include\\CKHSIThreshold.h" // 颜色抽取

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
```

```
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKColor.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKColor.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
// ColorThresholdDemo.cpp : Defines the class behaviors for the application.
BOOLCColorThresholdDemoApp::InitInstance()
{
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
}
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
// ColorThresholdDemoDlg.h : header file
    CPrImage      m_Image; // 基础图像
    CPrImage      m_Result;// 处理后的图像
    CGdiView      m_GdiView;    // 图形显示视图窗口
    CHSIThreshold m_hsiThre; // HSI 颜色抽取
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// ColorThresholdDemoDlg.cpp : implementation file
// CColorThresholdDemoDlg message handlers
BOOLCColorThresholdDemoDlg::OnInitDialog()
{
    // TODO: Add extra initialization here

    RECTrect;
```

```
    GetDlgItem(IDC_VIEW_RECT)->GetWindowRect( &rect );
    ScreenToClient( &rect );

    m_GdiView.Create( m_hWnd, rect );              // 按自定义的区域创建图形显示视图窗口。
    m_GdiView.SetBackColor( RGB(0,0,64) );         // 设置默认的背景颜色
    m_GdiView.SetDisplayImage( &m_Image );         // 设置当前显示的图像
}
// 执行彩色二值化
voidCColorThresholdDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    m_hsiThre.SetHueRange( m_Slider1.GetPos(), m_Slider2.GetPos() );
    m_hsiThre.SetSaturationRange( m_Slider3.GetPos(), m_Slider4.GetPos() );
    m_hsiThre.SetIntensityRange( m_Slider5.GetPos(), m_Slider6.GetPos() );

    BeginTime();
    // 执行颜色抽取（彩色二值化）
    m_hsiThre.Execute( m_Image, m_Result, MaxROI );
    EndTime();

    m_GdiView.SetDisplayImage( &m_Result );// 设置当前显示的图像为处理后的图像

    m_GdiView.Redraw();// 刷新显示视图
}
```

//……详情请打开实例参考。

ContourDemo 轮廓提取

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
// 添加自定义删除图形消息
#defineWM_DELETE_FIGURES     WM_USER+123

#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKContour.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
```

```
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKContour.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKContour.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
// CContourDemoApp initialization

BOOLCContourDemoApp::InitInstance()
{
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
}
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image; // 基础图像
    CContourDetect     m_Contour;    // 轮廓检测
    CGdiRect      m_Rect;       // 矩形ROI 检测区域
    COverlay      m_Overlay;    // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;    // 图像显示表面, 前显示的静态图形,主要用于检测结果生成图形
显示。
    CGdiView      m_GdiView;    // 图形显示视图窗口
    // 添加自定义消息删除图形
    afx_msgLRESULTOnDeleteFigures( WPARAMwParam, LPARAMlParam );
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
    //{{AFX_MSG_MAP(CContourDemoDlg)
    ON_MESSAGE( WM_DELETE_FIGURES, OnDeleteFigures )
```

```
// 以消息的方式删除图形
LRESULTCContourDemoDlg::OnDeleteFigures( WPARAMwParam, LPARAMlParam )
{
    if( m_Results.GetCount() )
    {
        for( inti=0; i<m_Results.GetCount(); i++ )
        {
            deletem_Results[i];
        }
        m_Results.RemoveAll();
    }

    return 0L;
}
// 执行轮廓检测
voidCContourDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    m_Contour.SetThreshold( GetDlgItemInt(IDC_EDIT1) );
    m_Contour.SetMinLength( GetDlgItemInt(IDC_EDIT2) );
    m_Contour.SetMaxLength( GetDlgItemInt(IDC_EDIT3) );
    m_Contour.SetFilterSize( m_Combo1.GetCurSel() );
    m_Contour.SetSubPixel( m_SubPixel.GetCheck() );

    BeginTime();
    if( m_Rect.GetVisible() )
        m_Contour.Execute( m_Image, m_Rect );// 执行轮廓检测
    else
        m_Contour.Execute( m_Image, MaxROI );
    EndTime();


    CStringtext;
    MPNT* data=NULL;
    m_List1.DeleteAllItems();

    //Overlay_DeleteAll(m_Results);// 删除所有图形，请注意在线程中调用清除图形时，最好使用发送消息的方式。
    SendMessage( WM_DELETE_FIGURES , 0 , 0);  // 发送消息删除图形

    if( m_Contour.GetPointCount()>0 ) {
```

```
        intnum, n=0, i=0;
        while( i>=0 ) {
            data = m_Contour.GetPointData(i);

            i = m_Contour.GetContourId(i,num);

            text.Format( "%d", n );
            m_List1.InsertItem( n, text );

            text.Format( "%d", num );
            m_List1.SetItemText( n, 1, text );

            text = (data->m&MCP_CLOSE) ? "是" : "否";
            m_List1.SetItemText( n, 2, text );

            n++;
        }

        // 根据结果数据添加到覆盖图显示
        CGdiContour* pp = newCGdiContour(
            m_Contour.GetPointData(0),
            m_Contour.GetPointCount() );
        if( pp!=NULL ) {
            pp->Offset( 0.5, 0.5 );
            pp->SetPenWidth( 2 );
            pp->SetPenColor(RGB(0,200,0));
            m_Results.AddItem(pp);
        }
    }
    // 刷新视图显示
    m_GdiView.Redraw();
}
```

//……详情请打开实例参考。

DataMatrixDemo 二维码读取（DM 码）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKDataMatrix.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKReader.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKReader.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image; // 基础图像
    CGdiRect      m_Rect;       // 矩形ROI 检测区域
    COverlay      m_Overlay;    // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;    // 图像显示表面, 前显示的静态图形,主要用于检测结果生成图形
显示。
    CGdiView      m_GdiView;    // 图形显示视图窗口
    CDataMatrix       m_dm;         // 读取 DataMatrix 二维码
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行读取DataMatrix 二维码
voidCDataMatrixDemoDlg::OnExecute()
{
```

```
// TODO: Add your control notification handler code here
Overlay_DeleteAll(m_Results);

intoption = 0;

if( IsDlgButtonChecked(IDC_CHECK1) )
     option |= 0x10;
if( IsDlgButtonChecked(IDC_CHECK2) )
     option |= 0x20;
if( IsDlgButtonChecked(IDC_CHECK3) )
     option |= 0x01;

m_dm.SetFilterLevel(m_Combo1.GetCurSel());
m_dm.SetPolarity(m_Combo2.GetCurSel());
m_dm.SetShapeType(m_Combo3.GetCurSel());
m_dm.SetOptions(option);

m_dm.SetMaxCount(GetDlgItemInt(IDC_EDIT5));
m_dm.SetThreshold(m_Combo4.GetCurSel());
m_dm.SetDefectLen(GetDlgItemInt(IDC_EDIT8));

m_dm.SetCodeWidth(GetDlgItemInt(IDC_EDIT3));
m_dm.SetCodeHeight(GetDlgItemInt(IDC_EDIT4));

m_dm.SetNumCellX(GetDlgItemInt(IDC_EDIT6));
m_dm.SetNumCellY(GetDlgItemInt(IDC_EDIT7));

BeginTime();

if( m_Rect.GetVisible() )
{
     m_dm.Execute( m_Image, m_Rect );// 执行读取DataMatrix 二维码
}
else
{
     m_dm.Execute( m_Image, MaxROI );
}

EndTime();

CStringtext;
```

```
    DataMatrixResult* data;
    m_List1.DeleteAllItems();
    for( inti=0; i<m_dm.GetResultCount(); i++ )
    {
        data = m_dm.GetResultCode(i);
        text.Format( "%d", i+1 );
        m_List1.InsertItem( i, text );
        text.Format( "%0.2f", data->border[0].x );
        m_List1.SetItemText( i, 1, text );
        text.Format( "%0.2f", data->border[0].y );
        m_List1.SetItemText( i, 2, text );
        m_List1.SetItemText( i, 3, data->codeText );


        CGdiPolygon* p1 = newCGdiPolygon;
        if( p1!=NULL )
        {
            p1->SetMax(4);
            p1->Add(data->border[0]);
            p1->Add(data->border[1]);
            p1->Add(data->border[2]);
            p1->Add(data->border[3]);
            if( data->codeLen>0 )
            {
                p1->SetPenColor(RGB(0,255,0));
            }
            else
            {
                p1->SetPenColor(RGB(255,0,0));
            }
            p1->SetPenWidth(2);
            m_Results.AddItem(p1);// 添加显示图形
        }
    }

    m_GdiView.Redraw(); // 刷新视图，显示图形
}
//……详情请打开实例参考。
```

EdgeToolDemo 边缘点检测

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKEdgeTool.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKMeasure.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKMeasure.lib")
#endif


usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image; // 基础图像
    CGdiRotBox        m_Rect;       // 旋转矩形ROI
    CEdgeTool     m_EdgeTool;   // 边缘点检测

    COverlay      m_Overlay;    // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;    // 图像显示表面, 前显示的静态图形,主要用于检测结果生成图形
```

显示。

```
CGdiView	m_GdiView;	// 图形显示视图窗口

CGdiProfile		m_Profile;	// 投影曲线边缘位置显示
COverlay	m_ProOver;	// 曲线图形显示
CGdiView	m_ProView;	// 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行边缘点检测
voidCEdgeToolDemoDlg::OnExecute()
{
	// TODO: Add your control notification handler code here
	m_EdgeTool.SetPolarity( m_Combo1.GetCurSel() );
	m_EdgeTool.SetLocation( m_Combo2.GetCurSel() );
	m_EdgeTool.SetThreshold( GetDlgItemInt(IDC_EDIT1) );
	m_EdgeTool.SetFilterHalf( GetDlgItemInt(IDC_EDIT2) );
	BeginTime();
	m_EdgeTool.Execute( m_Image, m_Rect );	// 执行区域内的边缘点检测
	EndTime();
	// 创建曲线显示视图窗口
	m_Profile.Create( m_EdgeTool.GetLength() );
	m_Profile.SetCurve1( m_EdgeTool.GetProjection() );
	m_Profile.SetCurve2( m_EdgeTool.GetStrengths() );
	m_Profile.SetThreshold( double(GetDlgItemInt(IDC_EDIT1)) );

	intitem;
	CStringtext;
	EdgeData* data;
	m_List1.DeleteAllItems();
	Overlay_DeleteAll(m_Results);
	for( inti=0; i<m_EdgeTool.GetEdgeCount(); i++ ) {
		data = m_EdgeTool.GetEdgeData( i );
		text.Format( "%d", i+1 );
		item = m_List1.InsertItem( i, text );
		text.Format( "%d", data->Polarity );
		m_List1.SetItemText( item, 1, text );
		text.Format( "%0.3f", data->Strength );
		m_List1.SetItemText( item, 2, text );
		text.Format( "%0.3f", data->Position.x );
		m_List1.SetItemText( item, 3, text );
		text.Format( "%0.3f", data->Position.y );
```

```
            m_List1.SetItemText( item, 4, text );
            m_Profile.AddEdge( data->Distance );

            // 把结果数据添加到覆盖图显示
            CGdiPoint* p = newCGdiPoint(
                  data->Position.x+0.5,
                  data->Position.y+0.5 );
            if( p!=NULL ) {
                  p->SetSize( 3 );
                  p->SetStyle( 1 );
                  p->SetPenColor( RGB(0,255,0) );
                  m_Results.AddItem( p );
            }
      }

      m_GdiView.Redraw();      // 刷新显示视图
      m_ProView.Redraw();      // 刷新曲线视图
}
```

//……详情请打开实例参考。

FitCircleDemo 圆形测量（拟合圆）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKEdgeTool.h"
#include"..\\..\\Include\\CKFitCircle.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKMeasure.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKMeasure.lib")
#endif
```

```
usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image; // 基础图像
    CEdgeTool     m_EdgeTool;   // 边缘点检测
    CFitCircle          m_Fit;        // 圆拟合工具
    CGdiRingScan  m_Ring;       // 圆环内扫描线ROI
    COverlay      m_Overlay;    // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;    // 图像显示表面, 前显示的静态图形,主要用于检测结果生成图形
显示。
    CGdiView      m_GdiView;    // 图形显示视图窗口
    COverlay      m_ProOver;    // 曲线图形显示
    CGdiView      m_ProView;    // 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 圆形测量
voidCFitCircleDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    UpdateData( TRUE );

    Overlay_DeleteAll(m_Results);// 清除原来的显示
    Overlay_DeleteAll(m_ProOver);// 清除原来的显示

    // 边缘点检测设置参数
    m_EdgeTool.SetPolarity( m_Combo1.GetCurSel() );
    m_EdgeTool.SetLocation( m_Combo2.GetCurSel() );
```

```
m_EdgeTool.SetThreshold( GetDlgItemInt(IDC_EDIT1) );
m_EdgeTool.SetFilterHalf( GetDlgItemInt(IDC_EDIT2) );

//拟合圆设置容忍误差
m_Fit.SetTolerance( m_dTolerate );

BeginTime();

intnum=0;
ROTRECTrc;
EdgeData* data;
// 圆环内扫描线设置扫描参数
m_Ring.SetScanCount( GetDlgItemInt(IDC_EDIT3) );
m_Ring.SetScanWidth( GetDlgItemInt(IDC_EDIT4) );
// 坐标点容器
CPointVectorptVector(m_Ring.GetScanCount());

for( inti=0; i<m_Ring.GetScanCount(); i++ ) {
     m_Ring.GetScanRoi( i, rc );
     m_EdgeTool.Execute( m_Image, rc );// 边缘点检测

     CGdiProfile* pProfile = new
          CGdiProfile(m_EdgeTool.GetLength());

     data = m_EdgeTool.GetEdgeData(0);// 获取边缘点数据
     if( data != NULL ) {
          ptVector.Add(data->Position);// 把点添加到容器中

          CGdiPoint* pp = new
               CGdiPoint(data->Position);
          if( pp != NULL ) {
               pp->Offset( 0.5, 0.5 );
               pp->SetStyle( 1 );
               pp->SetPenColor( RGB(255,255,0) );
               m_Results.AddItem( pp );// 显示扫描边缘点
          }
          if( pProfile!=NULL ) {
               pProfile->AddEdge(data->Distance);
          }
     }
     if( pProfile!=NULL ) {
```

```
            pProfile->SetCurve1( m_EdgeTool.GetProjection() );
            pProfile->SetCurve2( m_EdgeTool.GetStrengths() );
            pProfile->SetThreshold( m_EdgeTool.GetThreshold() );
            pProfile->SetVisible( FALSE );
            m_ProOver.AddItem( pProfile );// 曲线图显示
        }
    }
    // 执行拟合圆形
    if( m_Fit.Execute( ptVector ) ) {
        for( intn=0; n<m_Results.GetCount(); n++ ) {
            if( m_Fit.GetUse(n)==false ) {
                m_Results[n]->SetPenColor(RGB(255,0,0));
            }
        }
        CGdiCircle* p = newCGdiCircle;
        if( p != NULL ) {
            p->radius   = m_Fit.GetRadius();
            p->center.x = m_Fit.GetCenterX();
            p->center.y = m_Fit.GetCenterY();
            p->Offset( 0.5, 0.5 );
            p->SetPenColor( RGB(0,255,0) );
            m_Results.AddItem( p );// 显示圆
        }
    }

    CStringtext;

    text.Format( "半径(R)：%0.2f", m_Fit.GetRadius() );
    SetDlgItemText( IDC_DATA1, text );

    text.Format( "中心(X)：%0.2f", m_Fit.GetCenterX() );
    SetDlgItemText( IDC_DATA2, text );

    text.Format( "中心(Y)：%0.2f", m_Fit.GetCenterY() );
    SetDlgItemText( IDC_DATA3, text );

    text.Format( "拟合误差：%0.2f", m_Fit.GetRMSError() );
    SetDlgItemText( IDC_DATA4, text );

    EndTime();
```

```
    if( m_ProOver.GetCount()>0 )
        m_ProOver[0]->SetVisible( TRUE );
    SetDlgItemInt( IDC_EDIT5, 0 );

    m_GdiView.Redraw();// 刷新显示视图
    m_ProView.Redraw();// 刷新显示曲线视图
}
```

//……详情请打开实例参考。

FitLineDemo 直线测量（拟合直线）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKEdgeTool.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKMeasure.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKMeasure.lib")
#endif


usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage        m_Image;        // 基础图像
    CEdgeTool       m_EdgeTool;         // 边缘点检测
    CFitLine        m_Fit;              // 直线拟合工具
    CGdiBoxScan         m_Rect;             // 旋转矩形框内扫描线ROI
    COverlay        m_Results;          // 图像显示表面，前显示的动态图形,主要用于ROI 显示。
    COverlay        m_Overlay;          // 图像显示表面，前显示的静态图形,主要用于检测结果生成
图形显示。
    CGdiView        m_GdiView;          // 图形显示视图窗口
    COverlay        m_CurveOver;    // 曲线图形显示
    CGdiView        m_CurveView;    // 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 拟合直线
voidCFitLineDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    Overlay_DeleteAll(m_Results);    // 清除原来的显示
    Overlay_DeleteAll(m_CurveOver);  // 清除原来的显示
    // 边缘点检测设置参数
    m_EdgeTool.SetPolarity( m_Combo1.GetCurSel() );
    m_EdgeTool.SetLocation( m_Combo2.GetCurSel() );
    m_EdgeTool.SetThreshold( GetDlgItemInt(IDC_EDIT1) );
    m_EdgeTool.SetFilterHalf( GetDlgItemInt(IDC_EDIT2) );

    BeginTime();

    intnum=0;
    ROTRECTrc;
    EdgeData* data;
    //旋转矩形框内扫描线设置
    m_Rect.SetScanCount( GetDlgItemInt(IDC_EDIT3) );
    m_Rect.SetScanWidth( GetDlgItemInt(IDC_EDIT4) );
    // 坐标点容器
    CPointVectorptVector(m_Rect.GetScanCount());
```

```
for( inti=0; i<m_Rect.GetScanCount(); i++ ) {
    m_Rect.GetScanRoi( i, rc );
    m_EdgeTool.Execute( m_Image, rc );// 边缘点检测
    CGdiProfile* pProfile = new
        CGdiProfile(m_EdgeTool.GetLength()); //创建投影曲线边缘位置
    data = m_EdgeTool.GetEdgeData(0);
    if( data != NULL ) {
        ptVector.Add( data->Position ); // 把点添加到容器中
        CGdiPoint* pp = new
            CGdiPoint(data->Position);
        if( pp != NULL ) {
            pp->SetStyle( 1 );
            pp->SetPenColor( RGB(0,255,255) );
            m_Results.AddItem( pp );
        }
        if( pProfile!=NULL ) {
            pProfile->AddEdge( data->Distance );
        }
    }
    if( pProfile!=NULL ) {
        pProfile->SetCurve1( m_EdgeTool.GetProjection() ); // 设置曲线投影数据
        pProfile->SetCurve2( m_EdgeTool.GetStrengths() );  // 设置曲线梯度数据
        pProfile->SetThreshold( m_EdgeTool.GetThreshold() );
        pProfile->SetVisible( FALSE );
        m_CurveOver.AddItem( pProfile ); // 添加曲线到显示容器
    }
}

CStringtext;
GetDlgItemText( IDC_EDIT5, text );
// 拟合直线容忍误差设置
m_Fit.SetTolerance( atof(text) );

//执行拟合直线
if( m_Fit.Execute( ptVector ) ) {
    for( intn=0; n<m_Results.GetCount(); n++ ) {
        if( m_Fit.GetUse(n)==false ) { // 不参与拟合的点
            m_Results[n]->SetPenColor(RGB(255,0,0));  // 设置红色显示
        }
    }
    DLINEline;
```

```
        m_Fit.GetLine(9999,line);
        CGdiLine* pp = newCGdiLine(line);
        if( pp != NULL ) {
            pp->Offset( 0.5, 0.5 );
            pp->SetPenColor( RGB(0,255,0) );
            m_Results.AddItem( pp );
        }

        text.Format( "%0.2f", m_Fit.GetRMSError() );
        SetDlgItemText( IDC_RMSE, text );

        text.Format( "%0.2f", m_Fit.GetSampling() );
        SetDlgItemText( IDC_SAMPL, text );

        text.Format( "%0.2f", m_Fit.GetAngle() );
        SetDlgItemText( IDC_ANGLE, text );

        text.Format( "%0.2f", m_Fit.GetCenterX() );
        SetDlgItemText( IDC_CENTER_X, text );

        text.Format( "%0.2f", m_Fit.GetCenterY() );
        SetDlgItemText( IDC_CENTER_Y, text );

    } else {

        SetDlgItemText( IDC_RMSE, "" );
        SetDlgItemText( IDC_ANGLE, "" );
        SetDlgItemText( IDC_LENGTH, "" );
        SetDlgItemText( IDC_CENTER_X, "" );
        SetDlgItemText( IDC_CENTER_Y, "" );
    }

    EndTime();

    if( m_CurveOver.GetCount()>0 )
        m_CurveOver[0]->SetVisible( TRUE );
    SetDlgItemInt( IDC_EDIT6, 0 );
    //刷新显示
    m_GdiView.Redraw();
    m_CurveView.Redraw();
}
```

//……详情请打开实例参考。

HistogramDemo 灰度直方图、自动二值化阈值

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image;      // 基础图像
    CGdiRect      m_Rect;           // 矩形区域
    CHistogram        m_Histogram;  // 直方图功能
    COverlay      m_Overlay;        // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    CGdiView      m_GdiView;        // 图形显示视图窗口
    COverlay      m_HistOver;       // 直方图显示图    覆盖图
    CGdiView      m_HistView;       // 图形显示视图窗口
```

```
    CGdiHistogram m_GdiHist;          // 直方图显示数据
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行直方图分析
voidCHistogramDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here

    // 设置分析功能
    m_Histogram.SetAnalyse( Analyse_Min_Max|Analyse_Mean_StdDev );

    BeginTime();
    if( m_Rect.GetVisible() )
        m_Histogram.Execute( m_Image, m_Rect ); // 执行直方图分析
    else
        m_Histogram.Execute( m_Image, MaxROI );
    EndTime();

    CStringtext;
    SetDlgItemInt( IDC_DATA1, m_Histogram.GetMin() );
    SetDlgItemInt( IDC_DATA2, m_Histogram.GetMax() );
    text.Format( "%0.2f", m_Histogram.GetMean() );
    SetDlgItemText( IDC_DATA3, text );
    text.Format( "%0.2f", m_Histogram.GetStdDev() );
    SetDlgItemText( IDC_DATA4, text );

    intitem;
    intnLen = m_Histogram.GetLength();
    int* pValues = m_Histogram.GetValues();
    m_List1.DeleteAllItems();
    for( inti=0; i<nLen; i++ ) {
        item = m_List1.InsertItem( i, "" );
        text.Format( "%d", i );
        m_List1.SetItemText( item, 0, text );
        text.Format( "%d", pValues[i] );
        m_List1.SetItemText( item, 1, text );
    }
    m_GdiHist.SetValues( pValues, nLen ); // 设置直方图数据
    m_HistView.Redraw();// 直方图刷新显示
}
```

//……详情请打开实例参考。

ImageDemo 图像预处理

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image; // 基础图像
    CPrImage      m_Result;// 处理后的结果图像
    CGdiView      m_GdiView;    // 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行
voidCImageDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    UpdateData( TRUE );

    BeginTime();

    switch( m_Combo1.GetCurSel() )
    {
    case 0:
        ImgSmooth( m_Image, m_Result ); // 平滑
        break;
    case 1:
        ImgSharp( m_Image, m_Result ); // 锐化
        break;
    case 2:
        ImgSobel( m_Image, m_Result ); // Sobel边缘
        break;
    case 3:
        ImgErode( m_Image, m_Result );// 腐蚀
        break;
    case 4:
        ImgDilate( m_Image, m_Result );// 膨胀
        break;
    }

    EndTime();
    // 把处理结果图复制到当前图像
    m_Image.Copy( m_Result );

    m_GdiView.Redraw();
}
```

//……详情请打开实例参考。

ImageWarpDemo 环形展开裁剪图像

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
```

```
#include"..\\..\\Include\\CKGDI.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image; // 基础图像
    CPrImage      m_Result;// 处理后的结果图像
    CGdiRing      m_Roi;        // 圆环图形ROI
    COverlay      m_Overlay;    // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    CGdiView      m_GdiView1;   // 图形显示视图窗口
    CGdiView      m_GdiView2;   // 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行
voidCImageWarpDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    BeginTime();

    // 环形区域展开，把来源图像按ROI 裁剪，返回结果图像
```

```
    ImgRingWarp( m_Image, m_Result, m_Roi, m_Check1.GetCheck() );

    EndTime();

    m_GdiView2.FitSize();// 图像显示窗口自动适应显示。
    m_GdiView2.Redraw(); // 刷新显示视图窗口。
}
```

//……详情请打开实例参考。

ImgTransDemo 图形变换（镜像、平移、旋转、缩放、仿射）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage        m_Image; // 基础图像
    CPrImage        m_Result;// 处理结果图像
    CGdiView        m_GdiView;    // 图形显示视图窗口


    int                 m_nFunSel;    // 功能选择
    int                 m_nMirFlag;   // 镜像方式
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行
void CImgTransDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);

    BeginTime();

    switch( m_nFunSel ) {
    case 0:
        {
            m_nMirFlag=0;
            if( m_bMir1==TRUE )
                m_nMirFlag |= MirrorHor;
            if( m_bMir2==TRUE )
                m_nMirFlag |= MirrorVer;
            if( m_bMir3==TRUE )
                m_nMirFlag |= MirrorRot;
            ImgMirror( m_Image, m_Result, m_nMirFlag );// 镜像
        }
        break;
    case 1:
        {
            ImgShift( m_Image, m_Result, m_dShiftX, m_dShiftY, m_bInpl );// 平移
        }
        break;
    case 2:
        {
            ImgScale( m_Image, m_Result, m_dScaleX, m_dScaleY, m_bInpl );// 缩放
        }
        break;
```

```
    case 3:
        {
            ImgRotate( m_Image, m_Result, m_dAngle, m_bInpl );// 旋转
        }
        break;
    case 4:
        {
            doublematrix[6];
            doublerd = m_dAngle*PI/180;
            doubleca = cos(rd);
            doublesa = sin(rd);

            matrix[0] = ca*m_dScaleX;
            matrix[1] = sa*m_dScaleY;
            matrix[2] = m_dShiftX*m_dScaleX;
            matrix[3] = -sa*m_dScaleX;
            matrix[4] = ca*m_dScaleY;
            matrix[5] = m_dShiftY*m_dScaleY;

            ImgAffine( m_Image, m_Result, matrix, m_bInpl );// 仿射
        }
        break;
    }

    EndTime();

    m_Combo1.SetCurSel( 1 );

    m_GdiView.SetDisplayImage(&m_Result);// 显示结果图像
    m_GdiView.FitSize();                 // 适应显示
    m_GdiView.Redraw();                      // 刷新显示
}
```

//……详情请打开实例参考。

InspectDemo 基于图像对比缺陷检测

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"
```

```
#include"..\\..\\Include\\CKBlob.h"
#include"..\\..\\Include\\CKFindModel.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBlob.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKLocate.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBlob.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKLocate.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
CKVision::InitLibrary();    // 初始化 CKVision 库
//...

//...End
CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
CPrImage      m_Image;      // 基础图像
CPrImage      m_ImgDst;     // 目标图像
CModel            m_Model;      // 定位模板
CFindModel        m_Find;           //形状模型搜索
CPatInspect       m_Inspect;        // 基于图像对比缺陷检测
CBlobAnalyzer m_Blob;           // 斑点分析
CGdiRect      m_Rect;           // 矩形检测区域
COverlay      m_Overlay;        // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
```

```
    COverlay      m_Results;        // 图像显示表面，前显示的静态图形,主要用于检测结果生成图形显示。
    CGdiView      m_GdiView;        // 图形显示视图窗口
    CGdiView      m_SubView[4]; // 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行
void CInspectDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    Overlay_DeleteAll( m_Results );

    // 斑点分析参数设置
    m_Blob.SetBlobType( BLOB_WHITE );
    m_Blob.SetConnexity( CONNEXITY_8 );
    m_Blob.SetFeatures( BLOB_FEATURE_BASE|BLOB_FEATURE_AXIS );
    m_Blob.SetThreshold( GetDlgItemInt(IDC_THRE) );
    m_Blob.SetLimitArea( GetDlgItemInt(IDC_AREA) );

    // 对比参数设置
    m_Inspect.SetMaskEnabled( m_bMask.GetCheck() );
    m_Inspect.SetEqualizeMode( m_Combo1.GetCurSel() );
    m_Inspect.SetDefectType( m_Combo2.GetCurSel() );

    BeginTime();

    // 模板定位参数设置
    m_Find.SetMinScore( 60 );
    m_Find.SetThreshold( m_Model.GetThreshold()>>1 );
    // 执行识别定位
    m_Find.Execute( m_Image, m_Model, MaxROI );

    FindResult* pData = m_Find.GetResultData(0);
    if( pData!=NULL ) {

        ROTRECTrc;
        CPrImagetempl;

        rc.angle  = pData->Angle;
        rc.center = pData->Center;
        rc.width  = pData->Width;
```

```
        rc.height = pData->Height;

        templ.Cut( m_Image, rc );                    // 根据模板定位裁剪图像
        m_Inspect.Execute( templ, m_ImgDst );// 进行图像对比

        m_Blob.Execute( m_ImgDst, MaxROI );        // 分析对比后的图像

        CBlobData* data=NULL;
        CFrameTransframe( rc );
        for( inti=0; i<m_Blob.GetBlobCount(); i++ ) {
            data = m_Blob.GetBlobData(i);
            //椭圆图形功能
            CGdiEllipse* p1 = newCGdiEllipse(
                data->m_dCenterMassX,
                data->m_dCenterMassY,
                data->m_dMajorAxis/2,
                data->m_dMinorAxis/2,
                data->m_dRotation );
            if( p1!=NULL ) {
                p1->Transform( &frame );
                p1->SetPenColor( RGB(255,0,0) );
                m_Results.AddItem( p1 ); // 添加到显示覆盖图上
            }
        }
        /* 2D 旋转矩形功能*/
        CGdiRotBox* p2 = newCGdiRotBox(rc);
        if( p2!=NULL ) {
            p2->SetPenColor( RGB(0,255,0) );
            m_Results.AddItem( p2 );// 添加到显示覆盖图上
        }
    } else {
        AfxMessageBox( "没有搜索到目标！" );
    }

    EndTime();

    m_GdiView.Redraw();          //刷新显示
    m_SubView[3].Redraw();
}
```

//……详情请打开实例参考。

ModelDemo 模板轮廓匹配定位（老版本）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKLocate.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKLocate.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKLocate.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
CKVision::InitLibrary();    // 初始化 CKVision 库
//...

//...End
CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
CPrImage      m_Image;      // 基础图像
CModel            m_Model;      // 定位模板
CPrImage      m_ModImage;       // 模板图像
CFindModel        m_Find;           // 形状模型搜索
CGdiRect      m_lRoi;           // 学习ROI
CGdiRect      m_sRoi;           // 搜索ROI
```

```
    COverlay      m_Overlay;        // 图像显示表面，前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;        // 图像显示表面，前显示的静态图形,主要用于检测结果生成图形显示。
    CGdiView      m_GdiView;        // 图形显示视图窗口
    CModelDlg     m_ModelDlg;       // 编辑定位模板
    CModVarDlg        m_ModVarDlg;  // 角度、比例设置
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行搜索
voidCModelDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here
    m_lRoi.SetVisible( false );

    // 删除所有图形，请注意在线程中调用清除图形时，最好使用发送消息的方式，
    // 可参考ContourDemo 中的CContourDemoDlg::OnExecute() 。

    Overlay_DeleteAll(m_Results);



    // 设置搜索参数
    m_Find.SetMaxCount( GetDlgItemInt(IDC_EDIT1) );
    m_Find.SetMinScore( GetDlgItemInt(IDC_EDIT2) );
    m_Find.SetThreshold( GetDlgItemInt(IDC_EDIT3) );
    m_Find.SetPolarity( m_Combo1.GetCurSel() );
    m_Find.SetCompressor( m_Combo2.GetCurSel() );
    m_Find.SetModResult( m_Check1.GetCheck() );

    BeginTime();

    if( m_sRoi.GetVisible() )
    {
        m_Find.Execute( m_Image, m_Model, m_sRoi ); // 执行搜索功能
    }
    else
    {
        m_Find.Execute( m_Image, m_Model, MaxROI );
    }

    EndTime();
```

```
CStringtext;
FindResult* data=NULL;
m_List1.DeleteAllItems();

for( inti=0; i<m_Find.GetResultCount(); i++ )
{
    data = m_Find.GetResultData(i);// 获取匹配结果数据

    text.Format( _T("%d"), i+1 );
    m_List1.InsertItem( i, text );

    text.Format( _T("%0.3f"), data->Score );
    m_List1.SetItemText( i, 1, text );

    text.Format( _T("%0.3f"), data->Center.x );
    m_List1.SetItemText( i, 2, text );

    text.Format( _T("%0.3f"), data->Center.y );
    m_List1.SetItemText( i, 3, text );

    text.Format( _T("%0.3f"), data->Angle );
    m_List1.SetItemText( i, 4, text );

    text.Format( _T("%0.3f"), data->Scale );
    m_List1.SetItemText( i, 5, text );

    if( data->Model.count>0 ) {
        CGdiContour* p1 = new
            CGdiContour(data->Model);
        if( p1!=NULL ) {
            p1->Offset(0.5,0.5);
            p1->SetPenWidth( 1 );
            p1->SetPenColor( RGB(0,255,0) );
            m_Results.AddItem(p1);// 添加模板轮廓显示
        }
    }

    CGdiRotBox* p2 = newCGdiRotBox(
        data->Center.x,
        data->Center.y,
```

```
                data->Width,
                data->Height,
                data->Angle );
            if( p2!=NULL ) {
                p2->Offset(0.5,0.5);
                p2->SetPenColor( RGB(0,255,0) );
                m_Results.AddItem(p2);// 添加旋转矩形ROI显示
            }

            CGdiPoint* p3 = new
                CGdiPoint(data->Position);
            if( p3!=NULL ) {
                p3->Offset(0.5,0.5);
                p3->SetSize( 10 );
                p3->SetStyle( 2 );
                p3->SetPenColor( RGB(0,255,0) );
                m_Results.AddItem(p3);// 添加旋转矩形ROI显示
            }
        }

        m_GdiView.Redraw();// 刷新显示
    }
```

//……详情请打开实例参考。

MultiModelDemo 多轮廓匹配定位（新版本）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

//……详情请打开实例参考。

NCMatchDemo 灰度匹配定位

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKLocate.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKLocate.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKLocate.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
CKVision::InitLibrary();    // 初始化 CKVision 库
//...

//...End
CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
CNCPat          m_Pat;          // 匹配模板
CNCMatch    m_Match;    // 基于灰度区域匹配功能
CPrImage    m_Image;    // 基础图像
CGdiRect    m_lRect;    // 学习ROI
```

```
    CGdiRect      m_sRect;      // 搜索ROI
    COverlay      m_Overlay;          // 图像显示表面，前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;          // 图像显示表面，前显示的静态图形,主要用于检测结果生成
图形显示。
    CGdiView      m_GdiView;          // 图形显示视图窗口
    CPatternDlg       m_PatternDlg; // 编辑模板窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行匹配
voidCNCMatchDemoDlg::OnSearch()
{
    // TODO: Add your control notification handler code here

    m_lRect.SetVisible( false );     // 隐藏学习ROI
    // 设置匹配参数
    m_Match.SetMaxCount( GetDlgItemInt(IDC_EDIT1) );
    m_Match.SetMinScore( GetDlgItemInt(IDC_EDIT2) );
    m_Match.SetSubPixel( m_subPixel.GetCheck() );

    BeginTime();
    if( m_sRect.GetVisible() )
        m_Match.Execute( m_Image, m_Pat, m_sRect ); //执行匹配功能
    else
        m_Match.Execute( m_Image, m_Pat, MaxROI );
    EndTime();

    intitem;
    CStringtext;
    constMatchData* data;
    m_List1.DeleteAllItems();
    for( intn=0; n<m_Results.GetCount(); n++ )
        deletem_Results[n];
    m_Results.RemoveAll();
    for( inti=0; i<m_Match.GetMatchCount(); i++ ) {
        data = m_Match.GetMatchData( i );
    // 添加数据到列表
        text.Format( "%d", i+1 );
        item = m_List1.InsertItem( i, text );
        text.Format( "%0.2f", data->Score );
        m_List1.SetItemText( item, 1, text );
```

```
            text.Format( "%0.2f", data->Position.x );
            m_List1.SetItemText( item, 2, text );
            text.Format( "%0.2f", data->Position.y );
            m_List1.SetItemText( item, 3, text );
        // 添加中心点
            CGdiPoint* p1 = new
                CGdiPoint(data->Position);
            if( p1!=NULL ) {
                p1->SetPenColor( RGB(0,255,0) );
                p1->SetSize( 8 );
                p1->SetStyle( 2 );
                m_Results.AddItem( p1 );
            }
        // 添加矩形框
            CGdiRect* p2 = newCGdiRect;
            if( p2!=NULL ) {
                p2->left  = data->Center.x-data->Width/2+0.5;
                p2->top       = data->Center.y-data->Height/2+0.5;
                p2->right= data->Center.x+data->Width/2+0.5;
                p2->bottom    = data->Center.y+data->Height/2+0.5;
                p2->SetPenColor( RGB(0,255,0) );
                m_Results.AddItem( p2 );
            }
        // 添加标号
            CGdiText* p3 = newCGdiText;
            if( p3!=NULL ) {
                text.Format( "%d", i+1 );
                p3->SetText( text );
                p3->SetFontSize( 15 );
                p3->SetPenColor( RGB(0,255,0) );
                p3->SetPosition( p2->left+0.5, p2->top+0.5 );
                p3->SetSpace( 0, -15 );
                m_Results.AddItem( p3 );
            }
        }
        m_GdiView.Redraw();// 刷新显示
}
```

//……详情请打开实例参考。

QRCodeDemo 二维码检测（QR 码）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKReadQRCode.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKReader.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKReader.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image;      // 基础图像
    CGdiRect      m_Rect;           // 搜索ROI
    COverlay      m_Overlay;        // 图像显示表面，前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;        // 图像显示表面，前显示的静态图形,主要用于检测结果生成
图形显示。
    CGdiView      m_GdiView;        // 图形显示视图窗口

    CReadQRCode        m_QRCode;     // 二维码 QRCode 读取
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行
voidCQRCodeDemoDlg::OnBnClickedExecute()
{
    // TODO: 在此添加控件通知处理程序代码

    // 删除所有图形，请注意在线程中调用清除图形时，最好使用发送消息的方式，
    // 可参考ContourDemo 中的CContourDemoDlg::OnExecute() 。

    Overlay_DeleteAll(m_Results);

    BOOLisCheck1 = m_Check1.GetCheck();
    if(isCheck1){
        AutoThreshold();
    }

    intnMaxCount = GetDlgItemInt(IDC_EDIT1);
    intnPos = m_Slider1.GetPos();
    // 设置读取参数
    m_QRCode.SetMaxCount(nMaxCount);
    m_QRCode.SetThreshold(nPos);
    m_QRCode.SetPolarity(m_Combo1.GetCurSel());
    m_QRCode.SetMinArea(GetDlgItemInt(IDC_EDIT2));
    m_QRCode.SetMaxArea(GetDlgItemInt(IDC_EDIT3));

    BeginTime();

    if (m_Rect.GetVisible()){
        m_QRCode.Execute(m_Image, m_Rect);    // // 执行读取二维码功能
    } else {
        m_QRCode.Execute(m_Image, MaxROI);
    }

    EndTime();


    m_List1.DeleteAllItems();

    for (inti=0; i<m_QRCode.GetResultCount(); i++)
    {
```

```
// 获取二维码数据
QRCodeResult* pResult = m_QRCode.GetResultItem(i);
if (pResult)
{
    intnItem = m_List1.InsertItem(i,_T(""));
    CStringstr;
    str.Format( _T("%d"), i);
    m_List1.SetItemText(nItem, 0, str);

    DPNTctPos;
    Center2P( pResult->border[0], pResult->border[2], ctPos );
    doubledAngle = 0;
    dAngle = Angle2P( pResult->border[0], pResult->border[1] );

    str.Format( _T("%0.3f"), ctPos.x);
    m_List1.SetItemText(nItem, 1, str);
    str.Format( _T("%0.3f"), ctPos.y);
    m_List1.SetItemText(nItem, 2, str);

    str.Format( _T("%0.3f"), dAngle);
    m_List1.SetItemText(nItem, 3, str);

    m_List1.SetItemText(nItem, 4,  CString(pResult->codeText));

    // 多边形图形
    CGdiPolygon* p1 = newCGdiPolygon;
    if( p1!=NULL )
    {
        p1->SetMax(4);
        p1->Add(pResult->border[0]);
        p1->Add(pResult->border[1]);
        p1->Add(pResult->border[2]);
        p1->Add(pResult->border[3]);
        if( pResult->codeLen>0 )
        {
            p1->SetPenColor(RGB(0,255,0));
        }
        else
        {
            p1->SetPenColor(RGB(255,0,0));
        }
```

```
                p1->SetPenWidth(2);
                m_Results.AddItem(p1); // 添加到显示
            }

            CGdiPoint* p2 = newCGdiPoint(ctPos);
            if (p2!=NULL)
            {
                p2->SetSize(21);
                p2->SetStyle(0);
                p2->SetPenWidth(1);
                p2->SetPenColor(RGB(0,255,0));
                m_Results.AddItem(p2); // 添加到显示
            }

            for (intn=0; n<4; n++)
            {
                CStringstr;
                str.Format( _T("%d (%0.2f,%0.2f)"), n, pResult->border[n].x,
pResult->border[n].y);
                CGdiText* pText = newCGdiText(CStringA(str));
                if (pText!=NULL)
                {
                    pText->SetPosition(pResult->border[n].x, pResult->border[n].y);
                    pText->SetPenColor(RGB(0,0,255));
                    m_Results.AddItem(pText);// 添加到显示
                }
            }

        }
    }
    m_GdiView.Redraw();     // 刷新显示

}
```

//……详情请打开实例参考。

ReadOcrDemo 字符读取

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
```

```
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKReader.h"

#ifdef_WIN64
#pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
#pragma comment(lib, "..\\..\\Lib_x64\\CKReader.lib")
#else
#pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
#pragmacomment(lib, "..\\..\\Lib\\CKReader.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作

//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image;          // 基础图像
    CGdiRect      m_Rect;           // 范围ROI
    COverlay      m_Overlay;        // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;        // 图像显示表面, 前显示的静态图形,主要用于检测结果生成
图形显示。
    CGdiView      m_GdiView;        // 图形显示视图窗口

    CCharset      m_Charset;        // 字符集功
    CReadOcr      m_ReadOcr;        // 字符识别

    BOOL          m_bAutThers;      // 自动二值化
```

```
    CLearnDlg     m_LearnDlg;        // 学习字符窗口
    COcrListDlg   m_OcrListDlg;      // 编辑字符集
    CThresholdDlg m_ThresholdDlg;    // 二值化窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行字符读取
voidCReadOcrDemoDlg::OnExecute()
{
    // TODO: Add your control notification handler code here

    Overlay_DeleteAll(m_Results);

    if( m_bAutThers==TRUE ) {// 自动二值化阈值
        CHistogramhist;
        hist.SetAnalyse( Analyse_Threshold ); // 设置参数分析二值化阈值
        hist.Execute( m_Image, m_Rect );

        m_ReadOcr.SetThreshold( hist.GetThreshold() ); // 设置分割阈值
    }
    // 字符识别参数设置
    m_ReadOcr.SetPolarity( m_Combo1.GetCurSel() );
    m_ReadOcr.SetNoiseArea( GetDlgItemInt(IDC_EDIT2) );
    m_ReadOcr.SetMinWidth( GetDlgItemInt(IDC_EDIT3) );
    m_ReadOcr.SetMaxWidth( GetDlgItemInt(IDC_EDIT4) );
    m_ReadOcr.SetMinHeight( GetDlgItemInt(IDC_EDIT5) );
    m_ReadOcr.SetMaxHeight( GetDlgItemInt(IDC_EDIT6) );
    m_ReadOcr.SetMinScore( GetDlgItemInt(IDC_EDIT7) );

    m_ReadOcr.SetUniteSpaceX( GetDlgItemInt(IDC_EDIT9) );
    m_ReadOcr.SetUniteSpaceY( GetDlgItemInt(IDC_EDIT10) );

    m_ReadOcr.SetUniteEnabled( m_Check1.GetCheck() );

    m_ReadOcr.Execute( m_Image, m_Charset, m_Rect );// 执行字符识别

    m_List1.DeleteAllItems();

    OcrResult* data;
    CString  str, text;
    for( inti=0; i<m_ReadOcr.GetResultCount(); i++ ) {
```

```
        data = m_ReadOcr.GetResultData(i); // 返回单个字符数据

        str = data->Text;
        m_List1.InsertItem( i, str );
        text += str;

        str.Format( _T("%0.0f"), data->Score );
        m_List1.SetItemText( i, 1, str );

        str.Format( _T("%d"), RECT_WIDTH(data->Rect) );
        m_List1.SetItemText( i, 2, str );

        str.Format( _T("%d"), RECT_HEIGHT(data->Rect) );
        m_List1.SetItemText( i, 3, str );

        CGdiRect* p1 = new
            CGdiRect( data->Rect ); //矩形框
        if( p1!=NULL ) {

            p1->SetPenColor(RGB(0,255,0));
            m_Results.AddItem(p1); // 添加到显示
        }
    }

    SetDlgItemText( IDC_EDIT8, text );

    m_GdiView.Redraw(); // 刷新显示
}
```

//……详情请打开实例参考。

SearchDemo 模板轮廓匹配定位（新版本）

在 StdAfx.h 的头文件中添加 CKVISION 相关定义

```
#include"..\\..\\Include\\CKGDI.h"
#include"..\\..\\Include\\CKBase.h"
#include"..\\..\\Include\\CKLocate.h"
```

```
#ifdef_WIN64
    #pragma comment(lib, "..\\..\\Lib_x64\\CKBase.lib")
    #pragma comment(lib, "..\\..\\Lib_x64\\CKGDI.lib")
    #pragma comment(lib, "..\\..\\Lib_x64\\CKLocate.lib")
#else
    #pragmacomment(lib, "..\\..\\Lib\\CKBase.lib")
    #pragmacomment(lib, "..\\..\\Lib\\CKGDI.lib")
    #pragmacomment(lib, "..\\..\\Lib\\CKLocate.lib")
#endif

usingnamespaceCKVision;
```

在应用程序入口和退出的地方增加初始化和释放 CKVISION 库。

```
    CKVision::InitLibrary();    // 初始化 CKVision 库
    //...

    //...End
    CKVision::ExitLibrary();    // 退出 CKVision 库
```

在资源视图 Dialog 中添加相应的界面操作
//……详情请打开实例参考。

在对话框窗口的 .h 头文件中定义相应的图像处理功能:

```
    CPrImage      m_Image;      // 基础图像

    CMask         m_Mask;           // 图像掩摸
    CPrImage      m_Pat;            // 模板图像
    CShapeModel   m_Model;      // 形状模板
    CShapeMatch   m_Match;      // 基于边缘轮廓特征的形状匹配
    IRECT         m_lRect;      // 学习区域记录

    CGdiRect      m_lRoi;           // 学习ROI
    CGdiRect      m_sRoi;           // 搜索范围ROI
    COverlay      m_Overlay;        // 图像显示表面, 前显示的动态图形,主要用于ROI 显示。
    COverlay      m_Results;        // 图像显示表面, 前显示的静态图形,主要用于检测结果生成
图形显示。
    CGdiView      m_GdiView;        // 图形显示视图窗口
```

在对话框窗口的.cpp 实现文件中添加相应的功能实现。

```
// 执行匹配
voidCSearchDemoDlg::OnSearch()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);

    m_nSel = -1;

    // 设置搜索参数
    m_Match.SetMaxCount( GetDlgItemInt(IDC_EDIT1) );
    m_Match.SetMinScore( GetDlgItemInt(IDC_EDIT2) );
    m_Match.SetMinAngle( GetDlgItemInt(IDC_EDIT3) );
    m_Match.SetMaxAngle( GetDlgItemInt(IDC_EDIT4) );
    m_Match.SetMinScale( GetDlgItemInt(IDC_EDIT5) );
    m_Match.SetMaxScale( GetDlgItemInt(IDC_EDIT6) );
    m_Match.SetMaxOverlap( GetDlgItemInt(IDC_EDIT7) );
    m_Match.SetMaxSpeed( m_Slider1.GetPos() );
    m_Match.SetAccuracy( m_Slider2.GetPos() );
    m_Match.SetPolarity( m_Combo1.GetCurSel() );

    CStringtext;
    CModelContourtempl;

    m_List1.DeleteAllItems();

    // 删除所有图形，请注意在线程中调用清除图形时，最好使用发送消息的方式，
    // 可参考ContourDemo 中的CContourDemoDlg::OnExecute() 。

    Overlay_DeleteAll(m_Results);

    BeginTime();

    if( m_sRoi.GetVisible() )
    {
        m_Match.Execute( m_Image, m_Model, m_sRoi );   // 执行搜索功能
    }
    else
    {
        m_Match.Execute( m_Image, m_Model, MaxROI );
    }
```

```
EndTime();
SMatchData* data;
for( inti=0; i<m_Match.GetNumMatchs(); i++ )
{
    data=m_Match.GetMatchData(i);    // 获取匹配数据

    text.Format( "%d", i+1 );
    m_List1.InsertItem( i, text );

    text.Format( "%0.3f", data->score );
    m_List1.SetItemText( i, 1, text );

    text.Format( "%0.3f", data->center.x );
    m_List1.SetItemText( i, 2, text );

    text.Format( "%0.3f", data->center.y );
    m_List1.SetItemText( i, 3, text );

    text.Format( "%0.3f", data->angle );
    m_List1.SetItemText( i, 4, text );

    text.Format( "%0.3f", data->scale );
    m_List1.SetItemText( i, 5, text );

/*  CGdiRotBox* p1 = new CGdiRotBox(
            data->center.x,
            data->center.y,
            data->width,
            data->height,
            data->angle );
    if( p1!=NULL )
    {
        p1->Offset(0.5,0.5);
        p1->SetPenColor( RGB(0,255,0) );
        m_Results.AddItem(p1); // 把旋转矩形添加到显示
    }*/

    CGdiContour* p2 = newCGdiContour(data->model);
    if( p2!=NULL )
    {
```

```
            p2->Offset(0.5,0.5);
            p2->SetPenWidth( 1 );
            p2->SetPenColor( RGB(0,255,0) );
            m_Results.AddItem(p2); // 把模板轮廓添加到显示
        }

        CGdiPoint* p3 = newCGdiPoint(data->frame.point);

        if (p3 != NULL)
        {
            p3->SetSize(30);
            p3->SetPenWidth( 1 );
            p3->SetPenColor( RGB(0,255,0) );
            m_Results.AddItem(p3);// 把匹配点添加到显示
        }

    }

    m_GdiView.Redraw(); // 刷新显示

}
```

//.....详情请打开实例参考。

9 附 1.CKVISION API 功能分类

基础库			
	CKImage.h	图像	
	CKMask.h	图像掩膜	
	/* 图像处理 */		
	CKImgOpera.h	算术和逻辑	
	CKImgFilter.h	滤波处理	
CKBase.dll	CKImgMorph.h	形态学	
CKBase.lib	CKImgTrans.h	几何变换	
CKBase.h	CKImgConve.h	转换功能	
	/* 检测功能 */		
	CKHistogram.h	直方图和灰度分析	
	CKPixelStat.h	像素统计	
	CKSharpAssess.h	清晰度评测	
	/* 其它功能 */		
	CKGeoMeas.h	几何测量	
	CKFrameTrans.h	坐标系变换	

斑点分析			
	CKPatInspect.h		
	模板对比		
	CKBlobAnalyzer.h	CKMask.h	
CKBlob.dll	Blob 分析	图形掩摸	
CKBlob.lib			
CKBlob.h		CKBlobDef.h	
		Blob 定义	
		CKBlobData.h	
		Blob 数据	

标定与校准			
	CKDotMatrix.h		
CKCalibration.dll	圆形矩阵标定板		
CKCalibration.lib			
CKCalibration.h			
	CKCalibration.h		
	标定功能		

颜色识别			
	CKColorMonitor.h	颜色监测	
CKColor.dll			
CKColor.lib	CKColorIdentify.h	颜色识别	
CKColor.h			
	CKHSIThreshold.h	颜色抽取	

图形显示			
	CKGdiView.h	图形视图窗口功能	
	CKGdiType.h	GDI 模板类	
	CKGdiText.h	文本显示功能	
	CKGdiPoint.h	GDI 点显示类	
	CKGdiFrame.h	坐标系显示	
	CKGdiLine.h	线段图形功能	
	CKGdiRect.h	矩形框功能	
	CKGdiRotBox.h	旋转矩形功能	
CKGDI.dll	CKGdiBoxScan.h	旋转矩形框内扫描线	
CKGDI.lib	CKGdiCircle.h	圆形功能	
CKGDI.h	CKGdiRing.h	圆环图形	
	CKGdiRingScan.h	圆环内扫描线	
	CKGdiEllipse.h	椭圆图形	

	CKGdiContour.h	轮廓图形显示	
	CKGdiPolygon.h	多边形图形	
	CKGdiProfile.h	投影曲线边缘位置显示	
	CKGdiHistogram.h	直方图图形	
	CKGdiMask.h	掩摸显示	
	CKGdiModel.h	模型轮廓显示	

形状匹配			
	CKNCMatch.h	灰度匹配	
CKLocate.dll			
CKLocate.lib	CKFindModel.h	形状匹配	
CKLocate.h			
	CKShapeMatch.h	新形状匹配	
	CKModelContour.h	模型轮廓	

测量（点、线、圆）			
	CKEdgeTool.h	边缘点检测	
	CKCaliper.h	卡尺工具	
CKMeasure.dll	CKAcmeTool.h	顶点测量工具	
CKMeasure.lib	CKFitCircle.h	圆拟合工具	
CKMeasure.h	CKFitLine.h	线拟合工具	
	CKScanEdge.h	扫描边缘工具	
	CKScanSpace.h	扫描间距工具	

条码识别（一维码、二维码）			
CKReader.dll	CKBarcode.h	条码读取	
CKReader.lib			
CKReader.h	CKReadOcr.h	读取字符	
	CKFindBarcode.h	条码定位	
	CKDataMatrix.h	DataMatrix 二维码读取	
	CKReadQRCode.h	QRCode 二维码读取	