# 二维测量

HALCON提供多种方法，适用于不同的2D测量任务。图像测量需要提取物体的特定特征，通常提取的2D特征包括:

•测量对象的面积，即表示它的像素数;•测量对象的方向;•测量对象之间的角度;•测量对象的位置;•测量对象的尺寸，即其直径、宽度、高度和对象与对象部分之间的距离;•测量对象的数量关系。

选择使用哪些特征取决于测量任务的目标、所需精度以及物体在图像中的表示方式。

步骤1：创建区域并提取基本特征阈值

首先通过区域处理来提取一些基本特征。通过阈值算子寻找适当的灰度阈值，用一个或几个阈值将图像的灰度级分为几个部分，认为属于同一个部分的像素是同一个物体。如果一个区域中含有多个组件，建议将其分离处理。对于单独的组件，使用area\_center和orientation\_region运算符计算面积、位置和方向。

area\_center (Region, areareregion, RowCenterRegion, ColumnCenterRegion)   
orientation\_region (Region, OrientationRegion)   
dev\_display (Region)

第2步：提取边缘

利用边缘提取算子edges\_sub\_pix 来提取物体的轮廓。

第3步：分割轮廓

segment\_contours\_xld使用参数'lines\_circles'将轮廓分割成线性和圆形段。可选参数值有' lines '，用于只分割为直线，' lines\_ellipses '用于分割为直线和椭圆。

步骤4:将轮廓段划分为直线段和圆形段。

从轮廓段列表中选择圆形段。“cont\_approx”该变量的值可以通过操作get\_contour\_global\_attrib\_xld查询。它确定线段是代表一条线(' cont\_approx ' =-1)，椭圆弧(' cont\_approx ' =0)还是圆弧(' cont\_approx ' =1)。当我们在segment\_contours\_xld中选择参数“lines\_circles”时，“cont\_approx”可以是-1或1。根据这个值，每个轮廓可以用一个圆或一条线来近似。

步骤5:提取圆形轮廓段半径

fit\_circle\_contour\_xld(Contours : : Algorithm, MaxNumPoints, \  
MaxClosureDist, ClippingEndPoints, Iterations, ClippingFactor \  
: Row, Column, Radius, StartPhi, EndPhi, PointOrder)

算子fit\_circle\_contour\_xld用圆逼近值为1的段，即确定描述最能拟合到所选轮廓段中的圆的参数。参数StartPhi和EndPhi决定了圆中属于实际轮廓线段的部分。参数“半径”、“行”和“列”描述了圆的半径和位置，并用作生成相应圆的算子gen\_circle\_contour\_xld的输入。

步骤6:提取圆心之间的距离

distance\_pp( : : Row1, Column1, Row2, Column2 : Distance)

最后，distance\_pp是一个简单的几何运算，根据以下公式计算点对之间的距离:



为了提高检测的精度，就必须选择合适的工具来提取感兴趣的目标特征。在这里，我们简要介绍几种可用的基本工具，

•区域处理，主要对应于blob分析方法

•轮廓处理，其中包括边缘滤波方法，边缘和线条提取方法以及轮廓处理方法。

对于由相似灰度值、颜色或纹理的区域表示的对象或对象部分，blob分析是一种快速、简单的提取对象及其特征的方法。在此方法中，图像被分割成所谓的blobs，这些blobs是图像中包含特定范围或像素值行为的区域。

2.1.1图像或区域预处理

如果图像采集过程中的条件不理想，建议进行预处理，有时采集的图像是嘈杂的，杂乱的，或者对象被小范围的对象干扰或重叠，以至于小点或细线干扰了均匀区域描述感兴趣的实际对象这一过程。通常应用的预处理方法可以使用mean\_image或binomial\_filter消除噪声或者使用median\_image抑制小点或细线。进一步常用的预处理整个图像的算子有，gray\_opening\_shape和gray\_closing\_shape；使用smooth\_image可以实现图像的平滑。如果想平滑图像，但又想保留边缘，可以使用各向异性扩散的方法。对于区域，孔洞可以使用fill\_up或形态学运算符填充。修改区域以抑制小区域，opening\_circle和opening\_rectangle1抑制噪声，而closing\_circle和closing\_rectangle1填充空白。使用阴影校正补偿不均匀背景造成的影响。

预处理后，需要将图像分割成感兴趣的目标区域。阈值算子可以根据灰度值的分布对灰度值图像或多通道图像中的单个通道进行分割。常用的阈值算子有threshold、binary\_threshold、auto\_threshold、dyn\_threshold、fast\_threshold和local\_threshold。在手动选择阈值时，获取图像灰度值分布的信息可能会有所帮助。例如，gray\_histo, histo\_to\_thresh和intensity。此外，也可以使用HDevelop中的在线灰度直方图检查，显示设置为“阈值”，以交互式搜索合适的阈值。用阈值算子对图像进行分割后，感兴趣的图像部分作为几个区域进行使用。对于蜂窝结构的物体，分水岭算子比阈值算子更适合，因为分水岭算子基于拓扑而不是灰度值的分布对图像进行分割。如果希望获得具有相同强度的区域，可以应用区域生长的方法。对于这两种操作方法，建议使用低通滤波器(如binomial\_filter)进行预处理。

2.1.3选择和修改区域

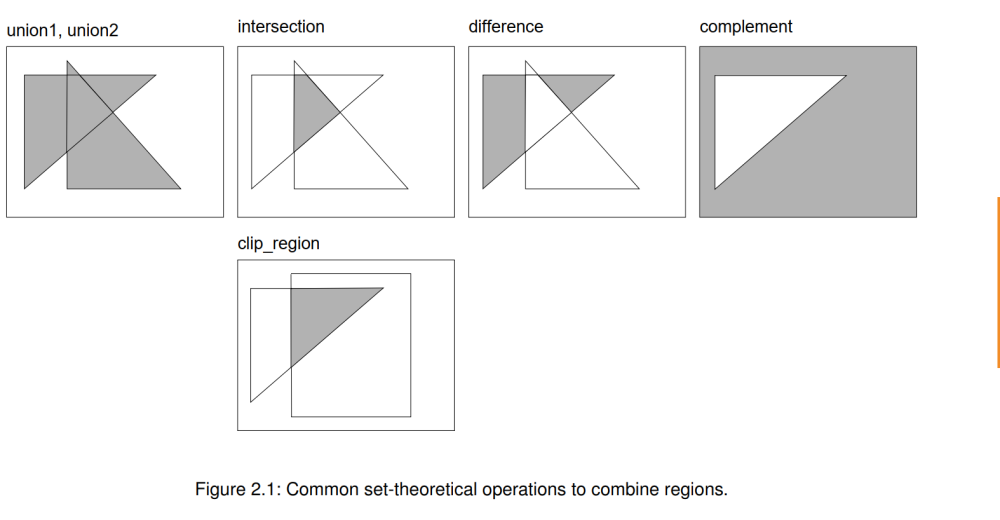
将图像分割成一组区域后，可以使用select\_shape或select\_gray等算子选择具有特定特征的区域。常见的特征有，一个特定的面积范围，一个特定的形状，或一个特定的灰度值。对于可用于选择的所有特性的列表，可以参见select\_shape的可选择条目。在许多情况下，需要对区域进行修改，这时可以使用形态学运算符(如opening\_circle或dilation\_rectangle)来消除小间隙或小连接。此外，不同的区域可以通过图2.1所示的集合理论算子进行组合。

•union1或union2合并区域

•intersection返回区域的交集

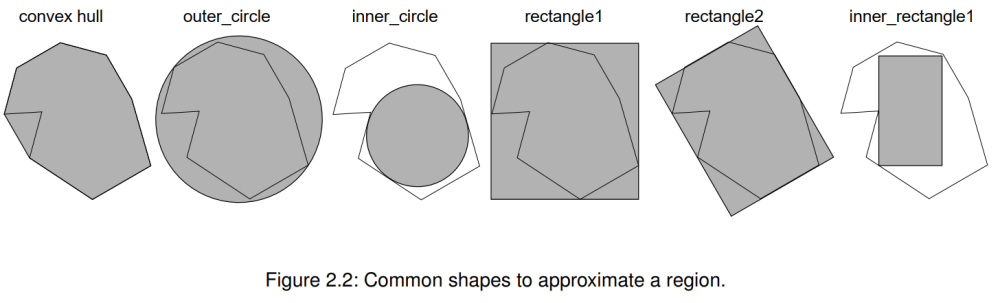
•difference从第一个区域减去两个区域的重叠部分

•complement获得一个区域的补。



如果你需要一个矩形区域的交集，例如，由gen\_rectangle1创建，你应该使用clip\_region。它的工作原理与交集（intersection ）相似，但效率更高。另一种修改区域的方法是对其进行变换，使用shape\_trans算子，通过特定的形状来近似它。如果想要获得目标对象的最大宽度，则可以通过其最小的封闭矩形或圆形近似该形状，并提取其宽度或半径。图2.2展示了一组常见的形状。可能的形状包括:

•凸壳(' convex ')、•最小的封闭圆(' outer\_circle ')、•适合该区域的最大圆(' inner\_circle ')、•平行于坐标轴的最小封闭矩形(' rectangle1 ')、•具有任意方向的最小封闭矩形(' rectangle2 ')、•平行于坐标轴且完全适合该区域的最大矩形(' inner\_rectangle1 ')、•与输入区域具有相同力矩的椭圆(' ellipse ')，•输入区域骨架上与输入区域重心距离最小的点(' inner\_center ')。骨架描述输入区域的中轴，可以通过算子骨架获得。此外，区域可以通过算子sort\_region、partition\_dynamic和rank\_region进行排序处理。



2.1.4提取特征

获得待测量对象所代表的区域后，即可提取待测量对象的特征，即实际测量结果。HALCON提供了一些算子来计算区域的面积、位置、方向或维度等特征。常见的算子有:

•area\_center计算任意形状区域的面积和中心位置，

•smallst\_rectangle1和smallst\_rectangle2计算最小的封闭矩形。特别是，smallst\_rectangle1计算与图像坐标轴平行的最小包围矩形的角坐标，smallst\_rectangle2计算具有任意方向的最小包围矩形的半径(半长)、位置和方向。

•inner\_rectangle1计算与坐标轴平行且完全适合区域的最大矩形的角坐标。

•inner\_circle确定适合区域的最大圆的半径和位置。

•diameter\_region 获得区域两个边界点之间的最大距离。

•orientation\_region用于获得区域的方向。orientation\_region和smallst\_rectangle2都计算对象的方向，但两者的结果可能不同，这取决于对象的形状。

如果想使用轮廓处理而不是区域处理，并且像素级精确的测量足够了，也可以从区域处理开始，使用gen\_contour\_region\_xld将区域转换为轮廓。如果可以用一个简单的轮廓形状来描述对象，但轮廓有很大的变形，那么使用轮廓加工是必要的。轮廓处理适用于高精度测量，适用于在图像中不是以均匀区域表示而是以清晰的灰度值或颜色过渡(边缘)表示的物体，和适用于没有封闭轮廓边界的物体部分。

轮廓处理从轮廓的创建开始，获取轮廓的常用方法是提取边缘。边缘是图像中暗区和亮区之间的过渡，在数学上确定，通过计算图像梯度，也可以计算边缘幅度和边缘方向，通过选取边缘幅值高或边缘方向特定的像素点，提取区域间的轮廓。

2.2.1.1像素级精确的边缘和线条

如果像素级精确的边缘提取是足够的，可以应用边缘滤波器。它产生一幅或两幅边缘图像，通过使用阈值算子选择具有给定最小边缘幅度的像素来提取边缘区域。如果需要得到厚度为1像素的边缘，就必须对得到的轮廓进行薄化，可以通过使用算子骨架获取之。常见的像素精度边缘滤波器有sobel\_amp算子，它是快速的。edges\_image算子，它不是那么快，但它包含了一个迟滞阈值和一个细化，并产生比sobel\_amp更准确的结果。edges\_image和它对应的彩色图像edges\_color可以通过Filter参数设置为' sobel\_fast '来应用，但这个参数只建议用于噪点或纹理小、边缘锐利的图像。

2.2.1.2亚像素级的边缘和线条

如果像素级的边缘和线条提取不精确，可以使用亚像素级的边缘和线条提取算子。这些会立即返回XLD轮廓。提取亚像素级精确边缘的常用算子有edges\_sub\_pix(用于一般边缘提取)、edges\_color\_sub\_pix(用于彩色图像边缘提取)和zero\_crossing\_sub\_pix(用于提取图像中的零交叉)。用于提取亚像素级精确线条(即具有一定宽度的细线性结构)的常用算子有lines\_gauss(用于一般线条提取)，lines\_facet(用于使用facet模型提取线条)和lines\_color(用于提取彩色图像中的线条)。

2.2.2.1抑制不相关轮廓

通过只选择那些满足特定约束的轮廓来抑制不相关轮廓。算子select\_shape\_xld可用于选择具有特定形状特征的封闭轮廓，轮廓的凹凸度，圆度或面积，大约有30种不同的形状特征可供选择；也可以根据典型的线条特征，如长度、曲率或方向来选择开放和封闭的轮廓，算子select\_xld\_point可以与捕捉函数结合使用，以交互方式选择轮廓。

2.2.2.2合并轮廓

当有多个轮廓近似于同一物体部分时，可以通过轮廓合并进一步减少线段的数量。针对以下情况提供了合适的运算符:

•近似位于同一直线上(union\_collinear\_contours\_xld)，

•位于同一圆上(union\_cocircular\_contours\_xld)，

•相邻(union\_adjacent\_contours\_xld)，

•共切(union\_cotangential\_contours\_xld)。

对于封闭轮廓或多边形，还可以使用集合理论算子将不同封闭轮廓或多边形的封闭区域组合起来。可用的运算符有:

•intersection\_closed\_contours\_xld和intersection\_closed\_polygons\_xld用于计算被封闭轮廓或多边形包围的区域的相交，

•difference\_closed\_contours\_xld和difference\_closed\_polygons\_xld用于计算被封闭轮廓或多边形包围的区域之间的差。

•symm\_difference\_closed\_contours\_xld和symm\_difference\_closed\_polygons\_xld用于计算被封闭轮廓或多边形包围的区域之间的对称差，

•union2\_closed\_contours\_xld和union2\_closed\_polygons\_xld用于合并被封闭轮廓或多边形包围的区域。

也可以通过直接将轮廓转换成形状来简化轮廓，但是在轮廓上而不是区域上操作的，通过使用shape\_trans\_xld可以将轮廓转换为最小的封闭圆，具有相同力矩的椭圆，凸壳，或最小的封闭矩形(平行于坐标轴或具有任意方向)。

2.2.4用已知形状近似提取轮廓段的特征

在选择和分割之后，通常的步骤是对轮廓或轮廓段进行形状基元拟合，以获得其特定的形状参数。可用的形状基元有线、圆、椭圆和矩形。所获得的特征可以是，线的端点或圆的中心和半径。如果你在前面的步骤中应用了一个分段，对于每个分段，“cont\_approx”的值，即段的形状，可以用算子get\_contour\_global\_attrib\_xld查询。根据其值，可以使用相应的拟合方法将最适合的形状基元拟合到轮廓段中:

•对于线段(' cont\_approx ' = -1)， fit\_line\_contour\_xld获取每个线段的参数，例如，两个端点的坐标。

•对于圆弧(' cont\_approx ' = 1)， fit\_circle\_contour\_xld

•对于椭圆弧(' cont\_approx ' = 0)，使用fit\_ellipse\_contour\_xld计算中心位置，半径以及被轮廓段覆盖的圆或椭圆的部分(由起点和终点的角度决定)。

•矩形由纯(未分割的)轮廓或已合并的线性轮廓组成，如通过union\_adjacent\_contours\_xld。对于它们，提供了操作符fit\_rectangle2\_contour\_xld。

利用得到的参数，可以生成相应的轮廓，进行可视化或进一步处理。线条可以用gen\_contour\_polygon\_xld生成，圆可以用gen\_circle\_contour\_xld生成，椭圆可以用gen\_ellipse\_contour\_xld生成，矩形可以用gen\_rectangle2\_contour\_xld生成。对于可视化，使用dev\_display等常见的可视化算子。

通过二维测量，可以测量用特定几何体表示的物体的尺寸。例如测量的几何形状包括圆圈、椭圆、矩形和线条。进而获得测量对象的位置、方向和尺寸的近似值。然后，通过观察图像中对象的实际边缘位置位不位于于近似对象的边界附近，对几何形状的参数进行优化，以更好地适应图像数据，并作为测量结果返回。

\*这个程序得到的偏差之间  
\*圆形轮廓和他们最适合的圆  
\*   
dev\_update\_var ('off')  
dev\_update\_off ()  
\*   
\* 步骤:获取图像并初始化可视化  
\*   
read\_image (Image, 'pumpe')  
get\_image\_size (Image, Width, Height)  
dev\_close\_window ()  
dev\_open\_window (0, 0, Width, Height, 'light gray', WindowID)  
dev\_set\_part (0, 0, Height - 1, Width - 1)  
dev\_set\_line\_width (1)  
dev\_set\_color ('red')  
dev\_set\_draw ('margin')  
dev\_display (Image)  
set\_display\_font (WindowID, 16, 'mono', 'true', 'false')  
disp\_continue\_message (WindowID, 'black', 'true')  
stop ()  
\*   
\* 步骤:创造ROI  
\*   
fast\_threshold (Image, Region, 0, 70, 150)  
dev\_set\_colored (3)  
connection (Region, ConnectedRegions)  
select\_shape (ConnectedRegions, SelectedRegions, \  
['outer\_radius', 'anisometry', 'area'], 'and', [5, 1, 100], [50, 1.8, 99999])  
shape\_trans (SelectedRegions, RegionTrans, 'outer\_circle')  
dilation\_circle (RegionTrans, RegionDilation, 5.5)  
union1 (RegionDilation, RegionUnion)  
reduce\_domain (Image, RegionUnion, ImageReduced)  
dev\_clear\_window ()  
dev\_display (ImageReduced)  
disp\_continue\_message (WindowID, 'black', 'true')  
stop ()  
\*   
\* 步骤:创建并选择相关轮廓  
\*   
threshold\_sub\_pix (ImageReduced, Border, 80)  
select\_shape\_xld (Border, SelectedXLD, ['contlength', 'outer\_radius'], 'and',\  
[70, 15], [99999, 99999])  
segment\_contours\_xld (SelectedXLD, ContoursSplit, 'lines\_circles', 4, 2, 2)  
select\_shape\_xld (ContoursSplit, SelectedXLD3, ['outer\_radius', 'contlength'], \  
'and', [15, 30], [45, 99999])  
union\_cocircular\_contours\_xld (SelectedXLD3, UnionContours2, 0.5, 0.1, 0.2, 2, \  
10, 10, 'true', 1)  
sort\_contours\_xld (UnionContours2, SortedContours, 'upper\_left', 'true', 'column')  
dev\_clear\_window ()  
dev\_set\_color ('white')  
dev\_display (ImageReduced)  
dev\_display (SortedContours)  
disp\_continue\_message (WindowID, 'black', 'true')  
stop ()  
\*   
\* 步骤:将圆圈放入轮廓中，并得到  
\* 等高线与圆的平均偏差  
\*   
count\_obj (SortedContours, NumSegments)  
dev\_display (Image)  
dev\_display (SortedContours)  
NumCircles := 0  
disp\_message (WindowID, 'Circle radius and average distance', 'window', \  
10, 10, 'white', 'false')  
disp\_message (WindowID, 'between circle and contour:', 'window', 30, 10, 'white', 'false')  
for i := 1 to NumSegments by 1  
 select\_obj (SortedContours, SingleSegment, i)  
 NumCircles := NumCircles + 1  
 fit\_circle\_contour\_xld (SingleSegment, 'atukey', -1, 2, 0, 5, 2, Row, \  
 Column, Radius, StartPhi, EndPhi, PointOrder)  
 gen\_circle\_contour\_xld (ContCircle, Row, Column, Radius, 0, rad(360), 'positive', 1)  
 dev\_display (ContCircle)  
 dist\_ellipse\_contour\_xld (SingleSegment, 'algebraic', -1, 0, Row, Column,\  
 0, Radius, Radius, MinDist, MaxDist, AvgDist, SigmaDist)  
 disp\_message (WindowID, NumCircles, 'window', Row - 10, Column - 5, 'white', 'false')  
 disp\_message (WindowID, 'R' + NumCircles + ': ' + Radius$'.3', 'window',\  
 (i - 1) \* 50 + 30, 450, 'white', 'false')  
 disp\_message (WindowID, 'D\_avg: ' + AvgDist$'.3', 'window', \  
 ((i - 1) \* 50) + 50, 450, 'white', 'false')  
endfor  
\*   
dev\_update\_window ('on')