第 3 章 图像

在前面的章节中，已经知道了图像是如何从硬件设备中获取的，也了解了 HALCON 软件环境以及相应的语法知识，本章将逐步进入 2D 视觉中图像处理的知识。首先，介绍图像的基础知识，包括对像素、图像通道、域的介绍；其次，介绍图像的预处理，包括去噪和图像增强；最后，介绍常见的图像分割方法，包括阈值分割、区域生长和分水岭分割。

3.1 图像基础知识


视频讲解

图像是指人在视觉系统中产生视觉印象的客观对象。计算机处理的图像称为数字图像，分为位图和矢量图，位图由数字阵列组成，常见的格式有 BMP、JPG、GIF；矢量图由矢量数据库表示，常见的格式有 PNG。在图像处理中，可以将图像概括为由像素、通道、域组成。

▶ 3.1.1 像素

像素是分辨率的单位，是构成图像的基本单元，可以用来表示各种信息，多数情况下采用点或者方块显示。而分辨率是单位英寸内的像素点个数，单位为 PPI(Pixels Per Inch)。假设一幅图像的分辨率是 1920×1080，那么可以简单理解为横向有 1920px，纵向有 1080px。不同像素类型以及对应的标准图像类型如表 3.1 所示。

表 3.1 不同像素类型以及对应的标准图像类型

像 素 类 型	标准图像类型
灰度图像(Gray Values)	byte，uint2
差异图(Difference)	int1，int2
2D 直方图(2D Histogram)	int4
边缘方向图(Edge Directions)	direction
导数图(Derivatives)	real
傅里叶变换(Fourier Transformer)	complex
色相值(Hue Values)	cyclic
矢量场(Vector Field)	vector_field

(1) 灰度图像：由 byte(8 位无符号，0～255)或者 uint2(16 位无符号，0～65 535)组成，通常是代表光在传感器上的局部光强度。

(2) 差异图：表示两幅图之间的差异，由 int1(8 位有符号，－127～128)和 int2(16 位有符号，－32 767～32 768)组成。

(3) 2D 直方图：使用 2D 直方图可以根据两幅图像中灰度值的出现情况来检查图像特征，其类型为 int4(32 位带符号，－2 147 483 637～2 147 483 648)。2D 直方图的每个轴表示的是输入图像的灰度值，特定灰度值的组合频率越高，输出图像中的灰度值就越高。

(4) 边缘方向图：表示边缘梯度的方向，图像类型为 direction(8 位无符号)，是一个存储

方向的数据类型，用来存储角度方向，上限是 180°，保存的角度信息一般是该角度的一半。

（5）导数图：32 位浮点型的实数数据类型，代表导数图像，用于提取边缘。

（6）傅里叶变换：图像类型为 complex（复数），每个像素有两个真实值，包含实部和虚部。使用傅里叶变换检查图像的频域，频率的幅值和相位都用复数表示。

（7）色相值：由循环数据类型 cyclic（8 位无符号）编码，像素表示色相值，当一个像素值超过 255 时，会被移动到光谱的另一端，即 255＋1＝0。

（8）矢量场：表示绝对/相对光流的一种特殊图像类型，可以表示像素的相对运动。

在 HALCON 中，像素类型是可以通过 convert_image_type()算子转换的，算子原型如下。

```
convert_image_type(Image : ImageConverted : NewType :)
```

- Image：输入的图像。
- ImageConverted：输出的转换后的图像。
- NewType：所需要的图像类型。

小技巧 在 HALCON 中，可以在右下角看到当前显示图像的图像类型，如图 3.1 所示。

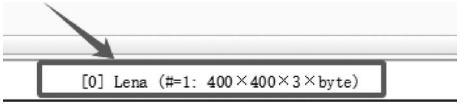


图 3.1 HALCON 软件中当前图像类型

经过上述介绍，已经认识了图像像素，而像素的灰度值可以认为是当前像素值，在 HALCON 中可以使用 get_grayval()算子获得指定坐标处的像素值，使用 set_grayval()算子设置指定坐标处的像素值，也可以使用 min_max_gray()算子获取一张图像内指定区域灰度的最大值和最小值，使用 intensity()算子计算图像灰度值的均值和标准差。其算子原型分别如下。

```
get_grayval(Image : : Row, Column : Grayval)
```

- Image：输入图像。
- Row：指定位置的纵坐标。
- Grayval：输出指定坐标的灰度值。

```
set_grayval(Image : : Row, Column, Grayval :)
```

- Image：输入图像。
- Column：指定位置的横坐标。
- Grayval：设定指定位置的灰度值。

```
min_max_gray(Regions, Image : : Percent : Min, Max, Range)
```

- Regions：需要计算的指定区域。
- Image：输入图像。
- Percent：低于（高于）绝对最大值（最小值）的百分比。
- Min、Max：最小、最大灰度值。
- Range：最大、最小灰度值之间的范围。

```
intensity(Regions, Image : : : Mean, Deviation)
```

- Regions：输入区域。

- Image：输入图像。
- Mean：输出均值。
- Deviation：输出标准差。

▶ 3.1.2 图像通道

除了不同的像素类型，还可以使用不同的通道来存储特定的信息。依据图像通道数的不同，可以分为单通道图像、三通道图像、多光谱图像、多通道图像。

单通道图像包含灰度图像和二值图像。灰度图像只能使用灰、白、黑来描述，像素值一般表示局部光照强度的值，即表示图像像素明暗程度的数值。灰度图像中点的颜色深度，范围一般为0～255，黑色为0，白色为255。二值图像只有黑白颜色，0表示黑色，1表示白色。

RGB图像即三通道图像，RGB代表的是光学三原色，R代表红色(Red)，G代表绿色(Green)，B代表蓝色(Blue)，RGB图像也称为彩色图像。三通道图像一般存储三个通道的数值，分别表示红色、绿色和蓝色的强度。RGB颜色模型(彩色图片见文前彩插)如图3.2所示，水平 x 轴表示红色，红色程度向左加深；y 轴表示蓝色，蓝色程度向右加深；z 轴表示绿色，绿色程度向上加深；原点为黑色。

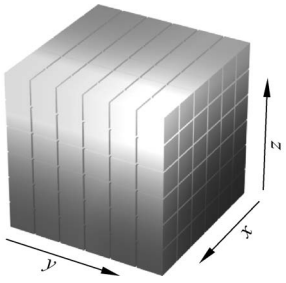


图3.2 RGB颜色模型(见彩插)

在机器视觉图像处理中，一般来说，当检测物的颜色对检测结果影响不大或者三原色中单个颜色对检测结果影响很大时，为了减少计算量，一般会将RGB图像转进行灰度化或者拆分成三通道的单通道图。在HALCON中，可以使用decompose3()算子将RGB图像拆分成三张图，算子原型如下。

```
decompose3(MultiChannelImage : Image1, Image2, Image3 : :)
```

- MultiChannelImage：多通道图像。
- Image1：红色分量上的图。
- Image2：绿色分量上的图。
- Image3：蓝色分量上的图。

也可以使用rgb1_to_gray()或rgb3_to_gray()算子将RGB图像转换成灰度图。算子原型如下。

```
rgb1_to_gray(RGBImage : GrayImage : :)
```

- RGBImage：输入的彩色图像。
- GrayImage：输出的灰度图像。

```
rgb3_to_gray(ImageRed, ImageGreen, ImageBlue : ImageGray: :)
```

- ImageRed：输入的红色通道图像。
- ImageGreen：输入的绿色通道图像。
- ImageBlue：输入的蓝色通道图像。
- ImageGray：输出的灰度图像。

多光谱图像，是一种多通道图像。利用特殊的相机，可以在一幅图像中存储多种光谱波段，包括来自可见光光谱之外的波段。例如，卫星相机通常会获取多光谱数据，并将其分别存储在多个不同的通道中。

多通道图，除了可以表示光照强度的通道之外，还可以表示其他信息的通道。例如，可以将灰度图与包含图像像素各自深度值的额外通道相结合，那么就可以在 3D 场景中将其进行可视化。

【例 3-1】 RGB 图像二值化与灰度化处理示例。

```
read_image (image, 'image/patras')
get_image_size (image, Width, Height)
* 1.直接灰度化图像
rgb1_to_gray (image, GrayImage)
* 2.将 RGB 图像拆分成三种单通道图像,再转换成灰度图像
* 2.1 将 RGB 图像转换成三通道的单通道图像
decompose3 (image, ImageR, ImageG, ImageB)
* 2.2 将三个通道的图像转换成灰度图像
rgb3_to_gray (ImageR, ImageG, ImageB, ImageGrayB)
write_image (GrayImage, 'jpg', 0, 'image/patras_grayImage')
* 二值化图像
binary_threshold (GrayImage, Region, 'max_separability', 'light', UsedThreshold)
region_to_bin (Region, BinImage, 255, 0, Width, Height)
write_image (BinImage, 'jpg', 0, 'image/patras_binImage')
```

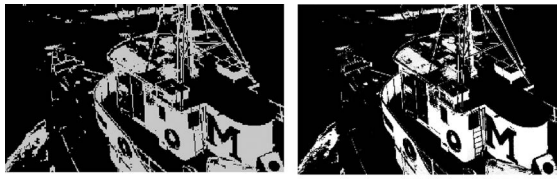
程序执行的结果如图 3.3 所示。

(a) 原图　　(b) 灰度图　　(c) 二值图

图 3.3　RGB 图像二值化与灰度化处理执行结果图

▶ 3.1.3　域

域，也称区域，决定了图像在后续操作中使用的图像的面积。为了将处理重点放在感兴趣的区域或加快操作速度，可以将域缩小到图像的相关部分。假设只对图像中的灰度值为 128～200 的图像域感兴趣，则可以选取灰度值为 128～200 的图像域作为研究对象，使其余域的灰度值为 0，如图 3.4 所示。



(a) 阈值图　　(b) 相应图像区域

图 3.4　灰度值为 128～200 的图像域

域中有一个重要概念——ROI(Region Of Interest，感兴趣区域)，是指从被处理的图像中以方框、圆等方式选中需要处理的区域，这个区域就是图像分析所关注的重点。通过这样的方式可以减少计算量，提高效率。

【例 3-2】 ROI 提取示例。

```
read_image(Clip, 'image/clip.png')
* 画一个矩形区域
```

```
gen_rectangle1(ROI_0, 39.5585, 62.8354, 211.81, 307.959)
* 将 ROI 之外的区域屏蔽(不改变图像大小)
reduce_domain(Clip, ROI_0, ImageReduced)
* 显示 ROI
dev_display(ImageReduced)
* 将 ROI 裁剪成一幅新的图像(改变了图像大小)
crop_domain(ImageReduced, ImagePart)
```

3.2 图像的预处理


视频讲解

图像的预处理就是在提取目标物之前对图像进行处理,使得提取目标物更为容易。预处理一般包含两个方向:去噪和图像增强。

▶ 3.2.1 去噪

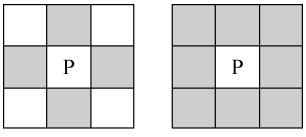
由于受到诸如系统噪声、曝光不足等内外部因素的影响,图片中易存在噪声或者不清晰,加大了图像信息处理的难度,因此有必要对图像进行去噪处理。简单地说,去噪就是在尽可能地保留目标特征的条件下对噪声进行抑制。去噪方法分为两大类:空间域方法和频域方法。空间域方法在空间域对图像进行处理,频域方法在频域空间中对图像进行处理。空间域就是把图像看成二维矩阵形式[nWidth,nHeight],或者用二维函数 $f(x,y)$ 表示。空间域也称为时域或图像空间。空间域方法以对图像的像素值直接进行处理为基础,包含均值滤波、中值滤波、高斯滤波等。频域是以空间域频率为自变量描述图像的特征,将一幅图像的像素值在空间上的变化分解成具有不同振幅、空间频率和相位进行表示。频域方法需要将空间域进行傅里叶变换到傅里叶变换空间,再对其值进行处理,最后转换回空间域,包含低通滤波、高通滤波等。

1. 空间域滤波

根据不同的功能可以将空间域方法分为两大类:一类是图像平滑,本质上是通过模糊来消除噪声,包含均值滤波、中值滤波;另一类是图像锐化,本质上是增强被模糊的图像细节信息,包含高斯滤波。

1) 均值滤波

均值滤波是一种线性平滑滤波,其基本原理是在图像中选择一个邻域,用邻域范围内的像素灰度的平均值来代替该邻域中心点像素的灰度值。经过均值滤波可以减小图像灰度的尖锐变化。在认识均值滤波之前,先了解一下邻域的概念。简单地说,邻域就是目标对象的邻居,如图 3.5 所示,P 的邻域为阴影的方格,分别为四邻域和八邻域。



(a) 四邻域　(b) 八邻域

图 3.5　邻域示意图

在理解邻域之后,假设一幅图像 $f(x,y)$ 为 $N\times N$ 阵列,均值滤波后的图像为 $g(x,y)$,滤波后每个像素的灰度值由包含 (x,y) 点邻域的几个像素的灰度值的平均值所决定,因此有

$$g(x,y)=\frac{1}{M}\sum_{(i,j\in S)}f(x,y) \tag{3-1}$$

式(3-1)中,$x,y=0,1,\cdots,N-1$,S 是 $h(x,y)$ 中以点 (x,y) 为中心的邻域的集合,M 是 S 内坐标点的总数。如图 3.6 所示,这里的点 (x,y) 的值为 90,M 为 9(邻域半径 3×3 的矩阵),那么通过均值滤波之后对应的值为 37。

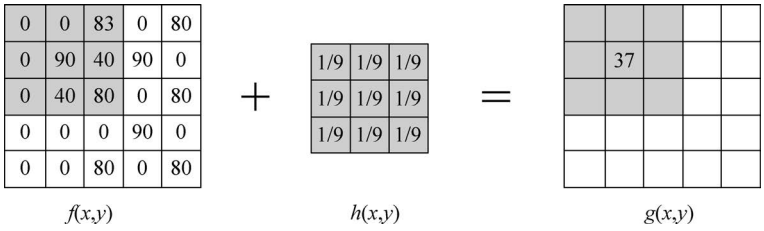


图 3.6　均值滤波示意图

均值滤波的优点是速度快；缺点是均值滤波处理后的图像会变得相对模糊，尤其是边缘和细节部分，其模糊程度与邻域半径有关，半径越大，模糊程度也越大。

在 HALCON 中 mean_image()算子用于实现均值滤波，其原型如下。

```
mean_image(Image : ImageMean : MaskWidth, MaskHeight :)
```

- Image：输入的带噪声的图像。
- ImageMean：输出的均值滤波的图像。
- MaskWidth，MaskHeight：掩膜宽度，掩膜高度。即邻域 S 中包含像素的横纵坐标的尺寸，一般选用奇数，如 3、5、7、9、11 等，奇数可以保证中心像素处于邻域的中心。

2）高斯滤波

高斯滤波就是对整幅图像进行加权平均的过程，每个像素点的值都由其本身和邻域的其他像素值进行加权平均计算后得到，适用于消除高斯噪声。HALCON 中 gauss_filter()算子用于实现高斯滤波，其原型为

```
gauss_filter(Image : ImageGauss : Size :)
```

- Image：输入的图像。
- ImageGauss：输出经过高斯滤波的图像。
- Size：高斯的尺寸，推荐 3、5、7、9、11。

3）中值滤波

均值滤波和高斯滤波都属于邻域平均法，在对噪声抑制的同时会致使图像边缘模糊。中值滤波属于统计排序滤波方法，其本质是将邻域内的像素灰度值进行排序，用其中值代替中心点像素灰度值的图像平滑方法，适用于处理图像中噪声为孤立点或线段的情况（如椒盐噪声），而且对图像边缘能较好地保护。HALCON 中 median_image()算子用于实现中值滤波的功能，其原型为

```
median_image(Image : ImageMedian : MaskType, Radius, Margin :)
```

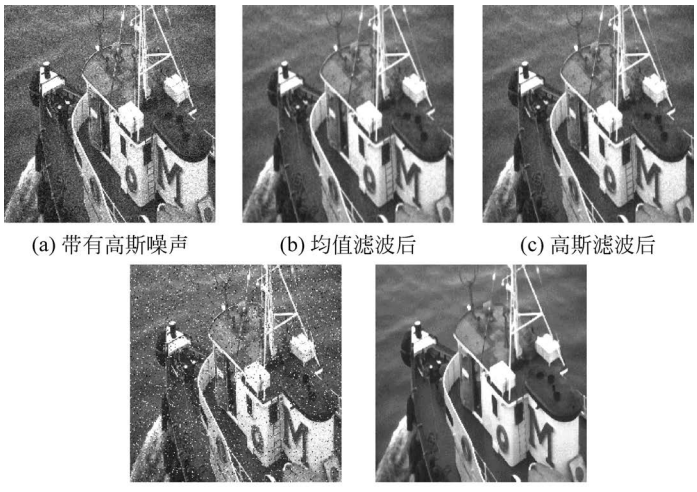
- Image：输入图像。
- ImageMedian：中值滤波后的图像。
- MaskType：掩膜类。
- Radius：掩膜尺寸。
- Margin：边界处理。

【例 3-3】　空间域滤波示例。

```
dev_close_window()
dev_open_window(0, 0, 400, 400, 'black', WindowHandle)
*读取图像
```

```
read_image(image, 'image/patras')
* 将图像灰度化
rgb1_to_gray(image, GrayImage)
* 添加高斯噪声
gauss_distribution(20, Gs_Distribution)
add_noise_distribution(GrayImage, Gs_ImageNoise, Gs_Distribution)
* 均值滤波
mean_image(Gs_ImageNoise, ImageMean, 5, 5)
* 高斯滤波
gauss_filter(Gs_ImageNoise, ImageGauss, 5)
* 添加椒盐噪声
sp_distribution(3, 3, Sp_Distribution)
add_noise_distribution(GrayImage, Sp_ImageNoise, Sp_Distribution)
* 中值滤波
median_image(Sp_ImageNoise, ImageMedian, 'circle', 3, 'mirrored')
* 关闭窗口,打开窗口并对比显示滤波后的图片
dev_close_window()
dev_open_window(0, 0, 400, 400, 'black', WindowHandle1)
dev_display(ImageMean)
dev_open_window(0, 400, 400, 400, 'black', WindowHandle2)
dev_display(ImageGauss)
dev_open_window(0, 800, 400, 400, 'black', WindowHandle3)
dev_display(ImageMedian)
```

程序执行结果如图 3.7 所示。



(a) 带有高斯噪声　(b) 均值滤波后　(c) 高斯滤波后

(d) 带有椒盐噪声图　(e) 中值滤波后

图 3.7　空间域滤波

2. 频域滤波

一幅数字图像可以定义为 $f(x,y)$,其中,x、y 为平面空间中的坐标值。如果 $f(x,y)$的值表示图像在该点的灰度值或者强度值,那么这幅数字图像称为时域图。而在频域中,$f(x,y)$的值表示的是图像在该点的灰度变化的剧烈程度,因此,图像灰度均匀的平滑区域主要对应低频部分,而图像中的噪声、边缘、细节等则对应高频部分。在频域中,通过滤除高频或低频成分来达到图像增强的效果。当对图像进行频域滤波时,需要先将图像中时域转换到频域,再在频域中做处理,最后将处理的结果转换回时域。HALCON 中通过 fft_image()算子将图像从时域转换到频域,通过 fft_image_inv()算子将频域转换到时域,还可以使用 fft_generic()算子实现从时域至频域或者从频域至时域的变换,其原型如下。

```
fft_generic(Image : ImageFFT : Direction, Exponent, Norm, Mode, ResultType :)
```

- Image：输入待变换的图像。
- ImageFFT：输出傅里叶变换之后的图像。
- Direction：变换方向，'to_freq'为时域到频域变换；'from_freq'为频域到时域变换。
- Exponent：指数符号。
- Norm：归一化方法。
- Mode：直流分量在频谱中的位置，'dc_center'为频谱中间；'dc_edge'为频谱边缘。
- ResultType：输出的图像类型。

下面介绍常见的低通滤波和高通滤波。

1）低通滤波

噪声主要集中在高频部分，低通滤波的本质是通过低频成分抑制高频成分，进而实现去噪。设原图像为 $f(x,y)$，选择低通滤波器传递函数 $H(u,v)$，通过衰减 $F(u,v)$ 的高频部分来生成 $G(u,v)$，最后进行傅里叶反变换，即可得到低通滤波后的图像 $g(x,y)$。其核心工作原理可表示为式(3-2)：

$$G(u,v)=H(u,v)F(u,v) \tag{3-2}$$

对于同一幅图像来说，采用不同的 $H(u,v)$ 平滑效果也不同，下面介绍三种常见的低通滤波器。

(1) 理想低通滤波器：最简单的低通滤波器，直接“截断”傅里叶变换中所有与变换原点距离比指定距离远 D_0 的高频成分，其变换函数为式(3-3)：

$$H(u,v)=\begin{cases}1, & D(u,v)\leqslant D_0\\ 0, & D(u,v)>D_0\end{cases} \tag{3-3}$$

其中，D_0 为截止频率，$D(u,v)$ 是点 (u,v) 到频率域原点的距离，即

$$D(u,v)=\sqrt{u^2+v^2} \tag{3-4}$$

其幅频特性曲线如图 3.8 所示。在图像处理中，用理想低通滤波器对图像进行滤波处理，会使滤波后的图像产生“振铃”现象，即在图像灰度剧烈变化处产生振荡。

在 HALCON 中，可以使用 gen_lowpass()算子创建理想的低通滤波器，其具体原型如下。

```
gen_lowpass(: ImageLowpass : Frequency, Norm, Mode, Width, Height :)
```

- ImageLowpass：输入的图像。
- Frequency：截断频率。
- Norm：滤波器归一化因子。
- Mode：直流分量在频域的位置。
- Width：输入图像的宽。
- Height：输入图像的高。

(2) 巴特沃斯低通滤波器：巴特沃思低通滤波器被称为最大平坦滤波器，一个 n 阶巴特沃斯低通滤波器的传递函数为

$$H=\frac{1}{1+\left(\dfrac{D(u,v)}{D_0}\right)^{2n}} \tag{3-5}$$

式中，n 为滤波器的阶数，其大小决定了衰减率。与理想低通滤波器直接截断不同，巴特沃斯低通滤波器带阻和带通之间有一个平滑的过渡带，因此传递函数是平滑过渡的，其幅频特

性曲线如图 3.9 所示。n 越大，该滤波器越接近理想滤波器，“振铃”现象也越明显。

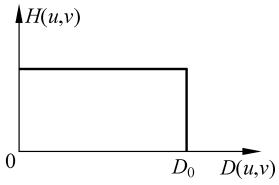


图 3.8 理想低通滤波器幅频特性曲线

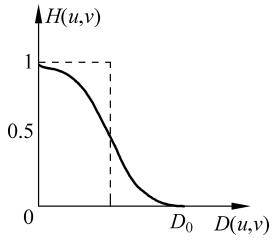


图 3.9 巴特沃斯低通滤波器幅频特性曲线

(3) 高斯低通滤波器：高斯低通滤波器的变换函数可以表示为

$$H(u,v)=\mathrm{e}^{\frac{-D^2(u,v)}{2D_0^2}} \tag{3-6}$$

高斯函数的傅里叶变换仍然是高斯函数，所以高斯型滤波器不会产生“振铃”现象。图 3.10 为高斯低通滤波器的幅频特性曲线。

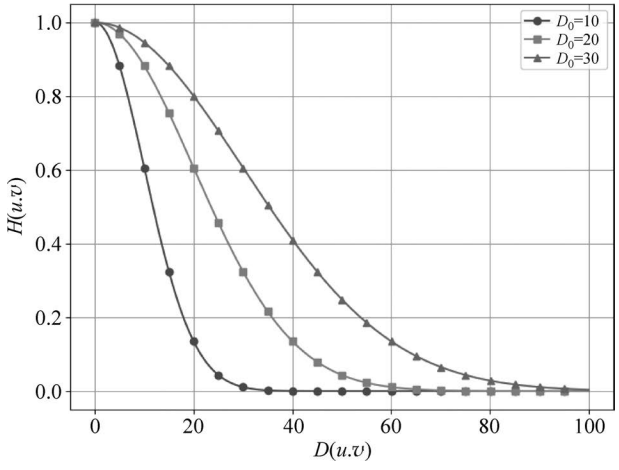


图 3.10 高斯低通滤波器的幅频特性曲线

在 HALCON 中可以通过 gen_gauss_filter()算子来生成高斯滤波器，其算子原型如下。

```
gen_gauss_filter(: ImageGauss : Sigma1, Sigma2, Phi, Norm, Mode, Width, Height :)
```

- ImageGauss：输出的高斯滤波器。
- Sigma1：高斯滤波器在空间域主方向的标准差。
- Sigma2：与主方向垂直方向的标准差。
- Phi：空间域的主方向。
- Norm：滤波器的归一化因子。
- Mode：直流分量在频域的位置。
- Width，Height：输入图像的宽和高。

【例 3-4】 低通滤波示例。

```
dev_close_window()
dev_open_window(0, 0, 400, 400, 'black', WindowHandle)
*读取图片
read_image(GsImagenoise, 'image/Gs_ImageNoise.jpg')
```

```
* 获取图像尺寸大小
get_image_size(GsImagenoise, Width, Height)
********** 对图像进行低通滤波处理 **********
* 生成高斯低通滤波器
gen_gauss_filter(ImageGauss, 2, 2, 0, 'none', 'rft', Width, Height)
* 对图像进行傅里叶变换
rft_generic(GsImagenoise, ImageFFT, 'to_freq', 'none', 'complex', Width)
* 用高斯低通滤波器对图像进行平滑滤波
convol_fft(ImageFFT, ImageGauss, ImageConvol)
* 将图像进行傅里叶反变换回时域图
rft_generic(ImageConvol, ImageFFT1, 'from_freq', 'sqrt', 'real', Width)
```

2）高通滤波

图像的边缘信息对应图像频谱的高频部分，高频部分的削弱会致使图像模糊，所以采用高通滤波法使图像中高频分量顺利通过，同时抑制低频分量，就可以突出图像的边缘信息，实现图像的锐化。在频域中实现高通滤波法的表达式与低通滤波法一致。下面介绍三种经典的高通滤波器：理想高通滤波器、巴特沃斯高通滤波器、高斯高通滤波器。

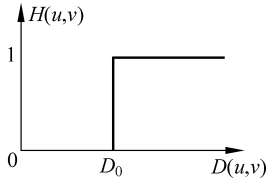


图 3.11　理想高通滤波器幅频特性曲线

（1）理想高通滤波器：理想高通滤波器的变换函数为式(3-7)，其中，D_0 为截止频率，$D(u,v)$是点(u,v)到频率域原点的距离，其幅频特性曲线如图 3.11 所示。

$$H(u,v)=\begin{cases}0, & D(u,v)\leqslant D_0\\ 1, & D(u,v)>D_0\end{cases} \tag{3-7}$$

HALCON 中使用 gen_highpass()算子生成理想高通滤波器，其算子原型如下。

```
gen_highpass(: ImageHighpass : Frequency, Norm, Mode, Width, Height :)
```

- ImageHighpass：输入的图像。
- Frequency：截断频率。
- Norm：滤波器归一化因子。
- Mode：直流分量在频域的位置。
- Width：输入图像的宽。
- Height：输入图像的高。

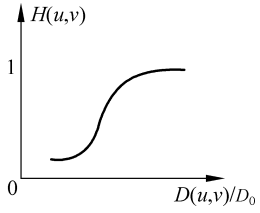


图 3.12　巴特沃斯高通滤波器幅频特性曲线

（2）巴特沃斯高通滤波器：巴特沃斯高通滤波器的转移函数为式(3-8)，其幅频特性曲线如图 3.12 所示。

$$H=\frac{1}{1+\left(\dfrac{D_0}{D(u,v)}\right)^{2n}} \tag{3-8}$$

（3）高斯高通滤波器：高斯高通滤波器的变换函数如式(3-9)所示，其幅频特性曲线如图 3.13 所示。

$$H(u,v)=1-\mathrm{e}^{\frac{-D^2(u,v)}{2D_0^2}} \tag{3-9}$$

在 HALCON 中，没有直接生成高斯高通滤波器的算子，但是从式(3-9)中可以发现，高斯高通滤波器就是一幅图像像素值为 1 的图与高斯低通滤波器相减的结果，具体应用可以参考案例 3-5。

【例 3-5】 高斯高通滤波示例。

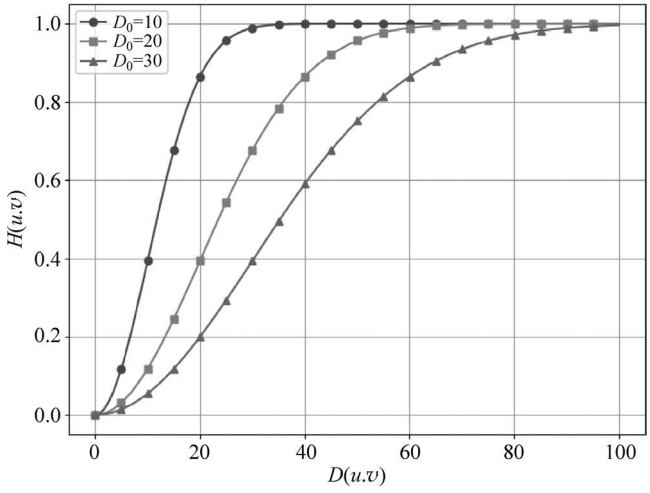


图 3.13 高斯高通滤波器幅频特性曲线

```
dev_close_window ()
* 读取图像
read_image(Image, 'image/brake_disk_bike_01.png')
* 获取图像尺寸大小
get_image_size(Image, Width, Height)
dev_open_window(0, 0, Width, Height, 'black', WindowHandle)
dev_display(Image)
* 构造一个高斯低通滤波器
gen_gauss_filter(ImageGauss, 0.1, 0.1, 0, 'none', 'dc_center', Width, Height)
* 构造一个值为 1 的实数型图
gen_image_const(Image1, 'real', Width, Height)
paint_region(Image1, Image1, ImageResult, 1, 'fill')
* 实数型图与高斯低通滤波器相减得到高斯高通滤波器
sub_image(ImageResult, ImageGauss, ImageSub, 1, 0)
* 傅里叶变换,得到所需要高斯高通处理图像在频域里的图
fft_generic(Image, ImageFFT, 'to_freq', -1, 'none', 'dc_center', 'complex')
* 用高通滤波器实现滤波
convol_fft(ImageFFT, ImageSub, ImageConvol)
* 从频域反变换回时域
fft_generic(ImageConvol, ImageFFT1, 'from_freq', 1, 'sqrt', 'dc_center', 'byte')
```

程序执行结果如图 3.14 所示。

▶ 3.2.2 图像增强

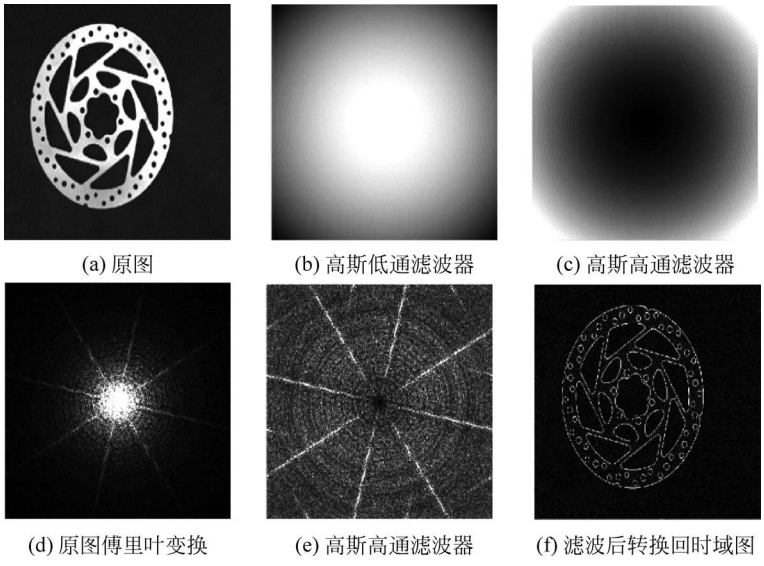
图像增强的目的是让检测物与非检测物有更为明显的区别,在 HALCON 中有以下几种方法可以实现图像增强。

1. 灰度线性变换

在 HALCON 中,scale_image()算子用于实现灰度线性变换,其算子原型为

```
scale_image(Image : ImageScaled : Mult, Add :)
```

- Image:输入的图像。
- ImageScaled:经过灰度线性变换后输出的图像。
- Mult:缩放因子。
- Add:偏移量。



(a) 原图　(b) 高斯低通滤波器　(c) 高斯高通滤波器

(d) 原图傅里叶变换　(e) 高斯高通滤波器　(f) 滤波后转换回时域图

图 3.14　高斯高通滤波检测过程

实现原理如式(3-10)所示。

$$g' = g \times \text{Mult} + \text{Add} \tag{3-10}$$

实际上就是将输入图像的像素值缩放 Mult 倍，然后偏移一定量，在缩放和偏移的过程中会放大像素之间的差异性，进而达到增强的效果。

HALCON 也可以使用 scale_image_max()算子计算图像像素的最大值和最小值，按照最大值比例化各个像素，将灰度值拉伸到 0～255。

在 HALCON 中，也可以使用 invert_image()算子反转图像像素值，如像素值 0 为黑色反转为白色，像素值 255 为白色反转为黑色。

2. 灰度非线性变换

对图像进行灰度非线性变换主要有两种方法，一种是进行对数变换，另一种是进行指数变换。在 HALCON 中，可以使用 log_image()算子对图像进行对数变换，用于提高暗部的像素值；使用 exp_image()对图像进行指数变换，用于提高亮部的像素值。其算子原型分别为

```
log_image(Image : LogImage : Base :)
```

- Image：输入图像。
- LogImage：输出图像。
- Base：底数，默认为'e'，还可以是 2、10。

```
exp_image(Image : ExpImage : Base :)
```

- Image：输入图像。
- ExpImage：输出图像。
- Base：指数，默认为'e'，还可以是 2、10。

3. 图像对比度与照明度的增强

在 HALCON 中，还可以对图像的高频区域(如拐角和边缘)进行增强，使得图像看起来更为清晰。可以使用 emphasize()算子增强图像的对比度；使用 illuminate()算子增强图像的照明度。它们的算子原型分别为

```
emphasize(Image : ImageEmphasize : MaskWidth, MaskHeight, Factor :)
```

- Image：输入图像。
- ImageEmphasize：输出图像。
- MaskWidth：低通掩模的宽度。
- MaskHeight：高通掩模的宽度。
- Factor：对比度强调的强度。

```
illuminate(Image : ImageIlluminate : MaskWidth, MaskHeight, Factor :)
```

- Image：输入图像。
- ImageIlluminate：输出图像。
- MaskWidth：低通掩模的宽度。
- MaskHeight：低通掩模的高度。
- Factor：缩放添加到原始灰度值的“校正灰度”。

小技巧 emphasize()算子的低通掩模的宽度和高通掩模的宽度设置为背景与目标物灰度值差值的2倍加1，即假设背景与目标物的灰度差值为10，那么低通掩模的宽度和高通掩模的宽度设置为21，可以达到较好的增强效果。

4. 直方图均衡化

通过直方图的线性化，可以增强图像的对比度，HALCON中使用equ_histo_image()算子实现，原理为：

$$f(x)=255\sum_{x=0,\cdots g}h(x) \tag{3-11}$$

算子原型为

```
equ_histo_image(Image : ImageEquHisto : :)
```

- Image：输入图像。
- ImageEquHisto：输出图像。

5. 灰度图像的形态学

形态学的运算主要包含腐蚀、膨胀、开运算、闭运算、顶帽变换等。在HALCON中可以使用gray_erosion_rect()算子进行灰度腐蚀，降低亮度，消除白点；使用gray_dilation_rect()算子进行灰度膨胀，增加亮度，消除黑点；使用gray_opening()算子进行灰度开运算，灰度先腐蚀再膨胀，用于去除孤立的白点；使用gray_closing()算子进行灰度闭运算，灰度先膨胀再腐蚀，用于去除孤立的黑点。

6. 图像之间的加减乘除操作

图像之间的加减乘除操作也可以达到图像增强的效果。HALCON中对图像进行加减乘除的算子原型如下。

```
add_image(Image1, Image2 : ImageResult : Mult, Add :)
```

- Image1：输入图像1，加数。
- Image2：输入图像2，加数。
- ImageResult：加法操作后的输出图像。
- Mult：灰度值适应因子，范围为－255～255。
- Add：灰度值范围自适应值，值为0～512。

```
sub_image(ImageMinuend, ImageSubtrahend : ImageSub : Mult, Add :)
```

- ImageMinuend：输入图像 1，被减数。
- ImageSubtrahend：输入图像 2，减数。
- ImageSub：相减后的输出图像。
- Mult：修正系数，范围为－255～255。
- Add：校正值，值为 0～512。

```
mult_image(Image1, Image2 : ImageResult : Mult, Add :)
```

- Image1：输入图像 1，乘数。
- Image2：输入图像 2，乘数。
- ImageResult：两图像相乘之后的输出图像。
- Mult：灰度值适应因子，范围为－255～255。
- Add：灰度值范围自适应值，值为 0～512。

```
div_image(Image1, Image2 : ImageResult : Mult, Add :)
```

- Image1：输入图像 1，被除数。
- Image2：输入图像 2，除数。
- ImageResult：两图像相除之后的输出图像。
- Mult：灰度值适应因子，范围为－1000～1000。
- Add：灰度值范围自适应值，值为－1000～1000。

【例 3-6】 图像增强示例。

```
dev_close_window ()
read_image(Image, 'image/alpha1.png')
get_image_size(Image, Width, Height)
dev_open_window(0, 0, Width, Height, 'black', WindowHandle)
dev_display(Image)
*灰度线性变换－方式 1
scale_image(Image, ImageScaled, 1.5, 30)
*灰度线性变换－方式 2
scale_image_max(Image, ImageScaleMax)
*反转像素值
invert_image(Image, ImageInvert)
*灰度非线性变换－对数
log_image(ImageInvert, LogImage, 'e')
*灰度非线性变换－指数
exp_image(Image, ExpImage, 2)
*增强图像的对比度和照明度
*增强对比度
emphasize(Image, ImageEmphasize, 101, 101, 10)
*增强照明度
illuminate(ImageInvert, ImageIlluminate, 101, 101, 3)
*直方图均衡化
equ_histo_image(Image, ImageEquHisto)
```

程序执行结果如图 3.15 所示。

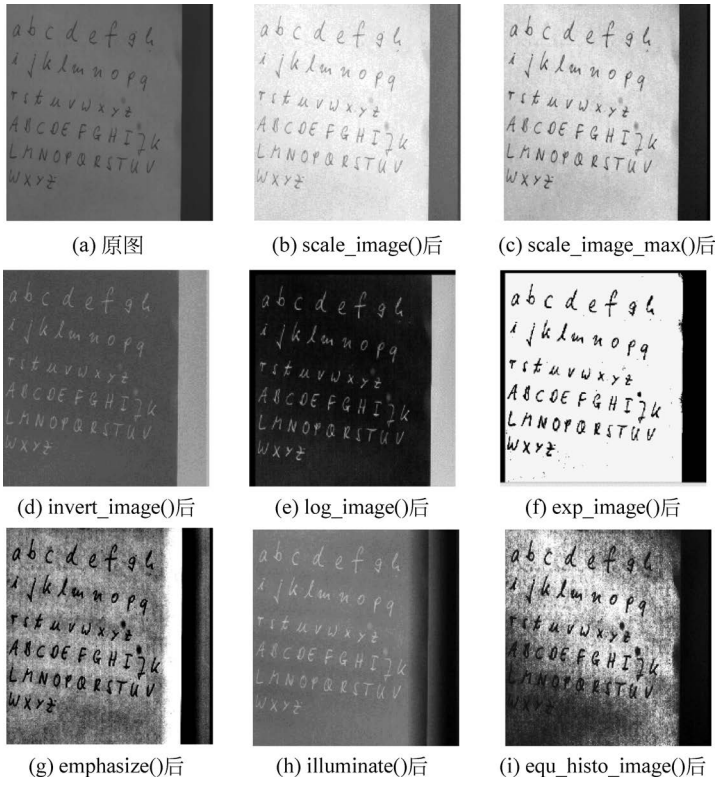


图 3.15 图像增强过程图

3.3 图像分割

视频讲解

一般来说，在图像处理中，并不是一整幅图像的所有像素都是人们感兴趣的像素，人们往往只是对某一部分的区域感兴趣，那么为了得到这部分区域，就需要对图像进行分割。图像分割的方法包含阈值分割、区域生长、分水岭分割。

▶ 3.3.1 阈值分割

阈值分割法是一种传统的图像分割方法，是基于区域的图像分割技术，不仅可以极大地压缩数据量，而且大大简化了分析和处理步骤。

阈值分割本质上是设定一个最大阈值和最小阈值，在这个阈值范围内的像素就被提取，不在范围内的就被忽略。图像的阈值分割主要利用检测目标与背景在灰度上的差异，选取一个或多个灰度阈值，然后把每个像素点的灰度值和确定的阈值相比较，对比比较结果进行分类，用不同的数值分别标记不同类别的像素，从而生成二值图像。

阈值分割操作被定义为

$$S=\{(r,c)\in R \mid g_{\min}\leqslant f_r,\quad c\leqslant g_{\max}\} \tag{3-12}$$

由式(3-12)可知，阈值分割将图像内灰度值处于某一指定灰度值范围内的像素值选中到区域 S 中。如果光照能保持恒定，系统设置好阈值 $g_{\min}$ 和 $g_{\max}$ 后就可以永远不用再进行调整了。

阈值分割可总结为以下三步。

（1）确定阈值。

（2）将阈值与像素灰度值进行比较。

（3）将符合条件的像素进行归类。

阈值分割的优点是计算简单、运算效率较高、速度快。阈值分割的难点主要是阈值的确定，阈值选取过高，容易把部分目标误判为背景；阈值选取过低，又容易将背景误判为目标。

阈值分割可分为全局阈值分割和局部阈值分割。全局阈值分割是对整幅图像进行分割，它适用于每一幅图像的光照都均匀分布，或多幅图像有一致照明的环境。局部阈值分割则基于邻域，通过局部像素灰度对比，为每个像素计算阈值，适用于图像背景灰度复杂或待测目标有阴影的情况。

1. 全局阈值分割

全局阈值分割法包括固定阈值分割和自适应阈值分割两大类。

1）固定阈值分割

固定阈值分割中，阈值选取是关键。在背景和前景灰度差异明显时，这样的图像一般具有明显谷底，可以使用直方图谷底法进行阈值分割。

直方图谷底法是从背景中提取物体的一种方法，它选择两峰之间的谷底对应的灰度值 T 作为阈值进行图像分割。T 值的选取如图 3.16 所示。

分割后的图像 $g(x,y)$ 如式(3-13)所示，$g(x)$ 为阈值运算后的图像。将像素点的灰度值小于 T 的像素点灰度值设为 0，将像素点的灰度值大于或等于 T 的像素点灰度值设为 255。

$$g(x)=\begin{cases}255, & f(x,y)\geqslant T\\ 0, & f(x,y)<T\end{cases} \tag{3-13}$$

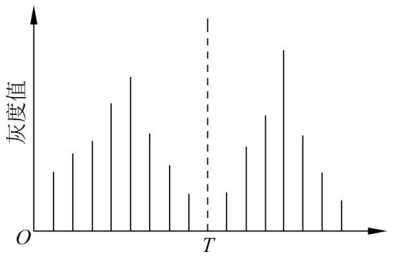


图 3.16　T 值的选取

固定阈值分割简单易操作，是在实际图像处理过程中最常用的方法。但是该方法只有在灰度值不变的情况下才有较好的效果，当光照发生变化时，图像的灰度值就会发生变化，此时使用固定阈值分割的效果就不太理想。

在 HALCON 中，通过 threshold()算子实现该方法，其原型如下。

```
threshold(Image : Region : MinGray, MaxGray :)
```

- Image：输入图像。
- Region：分割后的区域。
- MinGray：最小阈值。
- MaxGray：最大阈值。

2）自适应阈值分割

固定阈值是靠人对图像灰度的感知确定的，当图像灰度值在采集过程中发生轻微变化时，肉眼难以察觉。在连续采集图像时，图像的灰度也是动态变化的。为了能消除人工设定阈值的主观性，并且适应在采集过程中的环境变化，可以运用自适应阈值分割。自适应阈值分割是基于图像的灰度直方图来确定灰度阈值。在 HALCON 中使用 auto_threshold()算子进行自动阈值分割处理，该算子可以对单通道图像进行多重阈值处理，其原理是以直方图出现的谷底为分割点，对灰度直方图的波峰进行处理，算子原型如下。

```
auto_threshold(Image : Regions : Sigma :)
```

- Image：输入图像。
- Regions：分割后的输出区域。
- Sigma：对灰度直方图进行高斯平滑的核的大小。Sigma 的值越大，平滑效果越显著，直方图波峰越少，分割出的区域也越少；反之，Sigma 的值越小，直方图平滑的效果越不明显，分割的次数也越多。一般取 0.0、0.5、1.0、2.0、3.0、4.0、5.0，默认为 2.0。

除了 auto_threshold()算子外，还有 binary_threshold()算子对直方图波峰图像进行自适应阈值分割，适用于在比较亮的背景图像上提取比较暗的目标物。算子原型如下。

```
binary_threshold(Image : Region : Method, LightDark : UsedThreshold)
```

- Image：输入图像。
- Region：分割后的输出区域。
- Method：分割的方法，'max_separability'为最大类间方差法，'smooth_histo'为平滑直方图二分法。
- LightDark：选择提取的是前景还是背景，假如是前景，则选择'light'，否则选择'dark'。
- UsedThreshold：输出自动阈值分割使用的阈值。

2. 局部阈值分割

有时在一幅图像中找不到合适的阈值来全局分割背景和前景，但是在局部区域中，背景和前景又有较为明显的区别。此时，可以通过局部阈值分割来进行区域提取。局部阈值分割又称为局部自适应阈值分割或可变阈值处理。这种方法在像素的某一邻域内以一个或多个指定像素的特性（如灰度范围、方差、均值或标准差）为图像中的每一点计算阈值。一般来说，邻域的尺寸略大于要分割的最小目标即可。

在 HALCON 中通过 dyn_threshold()算子实现局部阈值分割。该算子利用邻域，通过局部灰度对比，找到一个合适的阈值进行分割，适用于灰度背景复杂，前景灰度与背景灰度不能明显区分的情况。

其应用步骤一般分为三步。首先，读取原始图像；然后，使用平滑滤波器对原始图像进行适当平滑；最后，使用 dyn_threshold()算子比较原始图像与均值处理后的图像局部像素差异，将差异大于设定值的点提取出来，如图 3.17 所示。

图 3.17　动态阈值分割流程图

算子原型如下。

```
dyn_threshold(OrigImage, ThresholdImage : RegionDynThresh : Offset, LightDark :)
```

- OrigImage：输入图像。
- ThresholdImage：输入的预处理后的图像，用于做局部灰度对比。
- RegionDynThresh：分割后的输出区域。
- Offset：输入的应用于阈值图像的偏移量，它是将原图与输入的预处理图像做对比后设定的值，灰度差异大于该值的将被提取出来。

- LightDark：确定提取的是哪部分区域的参数，有'dark'、'equal'、'light'和 'not_equal' 4个选择。
- light：原图中大于或等于预处理图像像素点值加上 offset 值的像素被选中。
- dark：原图中小于或等于预处理图像像素点值减去 offset 值的像素被选中。
- equal：原图中像素点大于预处理图像像素点值减去 offset 值，小于预处理图像像素点值加上 offset 值的点被选中。
- not equal：与 equal 相反，它的提取范围在 equal 范围以外。

▶ 3.3.2 区域生长

区域生长法的基本思想是将一幅图像中具有相似性质的像素聚集起来构成区域。首先在图像上选定一个“种子”像素或者“种子”区域，然后以种子像素为生长点，从邻域像素开始搜寻，接着比较种子周围像素与种子相邻像素的相似性，将有相同或相似性质的像素（根据某种事先确定的生长或相似准则来判断）合并到种子像素所在的区域中。最后将这些新像素作为新的种子像素继续进行上述操作，直到没有满足条件的像素为止，以此达到目标物体分割的目的。

区域生长法总结为以下三个步骤：①选择合适的种子像素；②确定区域生长准则；③确定区域生长的终止条件。区域生长法的关键是种子像素的选取，选择不同的种子像素会导致不同的分割结果。在 HALCON 中，使用 regiongrowing()算子和 regiongrowing_mean()算子实现区域生长法。其算子原型分别如下。

```
regiongrowing( Image : Regions : RasterHeight, RasterWidth, Tolerance, MinSize :)
```

- Image：输入图像。
- Regions：分割后的输出区域。
- RasterHeight，RasterWidth：矩形区域的宽、高，一般默认为奇数。
- Tolerance：灰度差值的分割标准，默认为 6.0。Tolerance 是指当像素点的灰度与种子区域的灰度值在该范围内时，则将它们合并为同一区域。
- MinSize：输出区域的最小像素数，默认为 100。

```
regiongrowing_mean( Image : Regions : StartRows, StartColumns, Tolerance, MinSize :)
```

- Image：输入图像。
- Regions：分割后的输出区域。
- StartRows，StartColumns：起始生长点的坐标。
- Tolerance：灰度差值的分割标准，默认为 5.0。
- MinSize：输出区域的最小像素值，默认为 100。

与 regiongrowing 不同，该算子指明了起始生长点坐标(x,y)，其生长终止条件有两种：一是区域边缘的灰度值与当前均值图中对应的灰度值的差小于 Tolerance 参数的值；二是区域包含的像素数应大于 MinSize 参数的值。

【例 3-7】 区域生长法示例。

```
* 读取图像
read_image(Image, 'image/fabrik')
* 对图像进行均值处理，选用 circle 类型的中值滤波器
```

```
median_image(Image, ImageMedian, 'circle', 2, 'mirrored')
* 使用 regiongrowing 算子寻找颜色相近的邻域
regiongrowing(ImageMedian, Regions, 1, 1, 2, 5000)
* 对图像进行区域分割,提取满足各个条件的各个独立区域
shape_trans(Regions, Centers, 'inner_center')
connection(Centers, SingleCenters)
* 计算出初步提取的区域的中心点坐标
area_center(SingleCenters, Area, Row, Column)
* 以均值灰度图像为输入,进行区域增长计算,计算的起始坐标为上一步的各区域中心
regiongrowing_mean(ImageMedian, RegionsMean, Row, Column, 25, 100)
```

程序执行结果(彩色图片见文前彩插)如图 3.18 所示。

(a) 原图　(b) regiongrowing()后　(c) regiongrowing_mean()后

图 3.18　区域生长法程序执行结果图(见彩插)

▶ 3.3.3　分水岭分割

“分水岭”这个名字与一种地貌特点有关,是一种基于拓扑理论的数学形态学的分割方法。它的思想是将图像的灰度看作起伏的地图,图像中的每一点像素的灰度值表示该点的海拔高度,高灰度值代表山脉,低灰度值代表盆地,每一个局部极小值及其影响区域称为集水盆,而集水盆的边界则形成分水岭。分水岭算法是一种典型的基于边缘的图像分割算法,将在空间位置相近并且灰度值相近的像素点互相连接起来构成一个封闭的轮廓,封闭性是分水岭算法的一个重要特征,对微弱的边缘有着良好的响应,但图像中的噪声会使分水岭算法产生过度分割的现象。

HALCON 中使用 watersheds()算子和 watersheds_threshold()算子实现图像的分水岭分割。其算子原型分别如下。

```
watersheds(Image : Basins, Watersheds : :)
```

- Image:输入图像,图像类型只能是 byte、uint2 和 real。
- Basins:盆地区域。
- Watersheds:分水岭区域(至少有一个像素宽度)。

```
watersheds_threshold(Image : Basins : Threshold :)
```

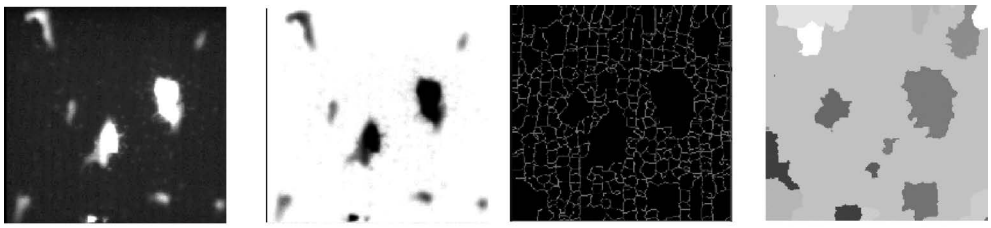
- Image:输入图像。
- Basins:盆地区域。
- Threshold:分割时的阈值。

【例 3-8】 分水岭分割示例。

```
* 获取图像
read_image(Br2, 'image/particle')
```

```
*对单通道图像进行高斯平滑处理,去除噪声
gauss_filter(Br2, ImageGauss, 9)
*将图像颜色进行反转
invert_image(ImageGauss, ImageInvert)
*对高斯平滑后的图像进行分水岭处理与阈值分割,提取出盆地区域
watersheds(ImageInvert, Basins, Watersheds)
watersheds_threshold(ImageInvert, Basins1,30)
```

程序执行结果如图 3.19 所示。



(a) 原图　(b) 反转后的图　(c) 分水岭区域　(d) 分割后的盆地

图 3.19　分水岭算法分割实例图

在线测试

习题

3.1　什么是数字图像？常见的数字图像的格式有哪些？

3.2　什么是 ROI？

3.3　去噪的方法可以分为几类？常见有哪些方法？

3.4　图像分割有哪些方法？