实验六 matlab中的图像操作

1. 实验目的
2. 熟悉图像滤波、图像变化和图像增强等数学原理
3. 了解matlab实现图像滤波、图像变化和图像增强等操作方法
4. 编写并尝试matlab中的图像增强、边缘检测等算法
5. 实验平台

MATLAB 2022b

Image Processing Toolbox库需自行安装

三、实验步骤

1. 对比度增强方法

此示例说明如何使用以下方法增强灰度和彩色图像的对比度：强度值映射、直方图均衡化和对比度受限的自适应直方图均衡化。

以下三个函数特别适用于对比度增强：

imadjust 将输入强度图像的值映射到新值，以对输入数据中强度最低和最高的 1%（默认值）数据进行饱和处理，从而提高图像的对比度。

histeq 执行直方图均衡化。它变换强度图像中的值，以使输出图像的直方图近似匹配指定的直方图（默认情况下为均匀分布），从而增强图像的对比度。

adapthisteq 执行对比度受限的自适应直方图均衡化。与 histeq 不同，它对小数据区域（图块）而不是整个图像执行运算。它会增强每个图块的对比度，使得每个输出区域的直方图近似匹配指定的直方图（默认情况下为均匀分布）。可以限制对比度增强，以避免放大图像中可能存在的噪声。

1. 增强灰度图像

将一个对比度差的灰度图像读入工作区中。使用采用默认设置的三种对比度调整方法增强图像。

pout = imread("pout.tif");

pout\_imadjust = imadjust(pout);

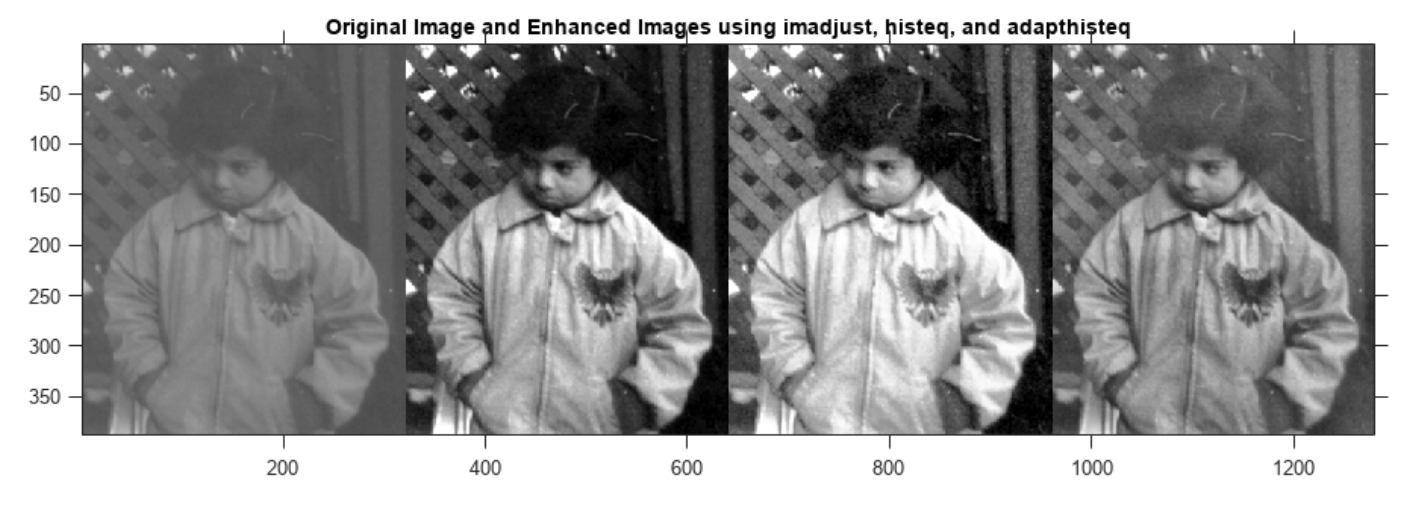
pout\_histeq = histeq(pout);

pout\_adapthisteq = adapthisteq(pout);

以蒙太奇方式显示原始图像和三个对比度调整后的图像。

montage({pout,pout\_imadjust,pout\_histeq,pout\_adapthisteq},"Size",[1 4])

title("Original Image and Enhanced Images using imadjust, histeq, and adapthisteq")



将另一个灰度图像读入工作区，并使用三种对比度调整方法增强该图像。

tire = imread("tire.tif");

tire\_imadjust = imadjust(tire);

tire\_histeq = histeq(tire);

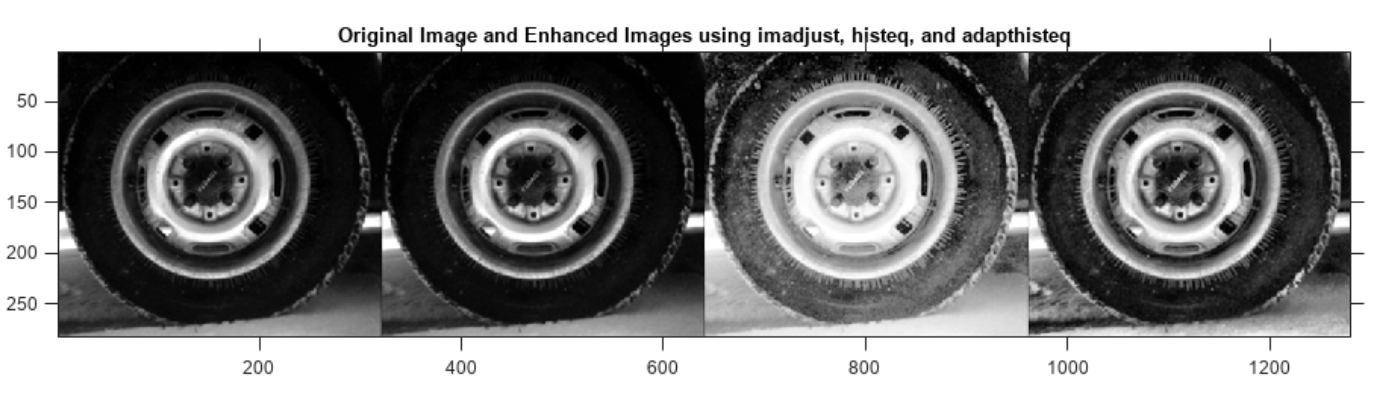
tire\_adapthisteq = adapthisteq(tire);

以蒙太奇方式显示原始图像和三个对比度调整后的图像。

montage({tire,tire\_imadjust,tire\_histeq,tire\_adapthisteq},"Size",[1 4])

title("Original Image and Enhanced Images using " + ...

"imadjust, histeq, and adapthisteq")



请注意，imadjust 对轮胎的图像影响不大，但在 pout 图中产生了明显变化。绘制 pout.tif 和 tire.tif 的直方图表明，第一个图像中的大部分像素集中在直方图的中心，而对于 tire.tif，这些值已分布在最小值 0 和最大值 255 之间，从而阻止 imadjust 有效地调整图像的对比度。

figure

subplot(1,2,1)

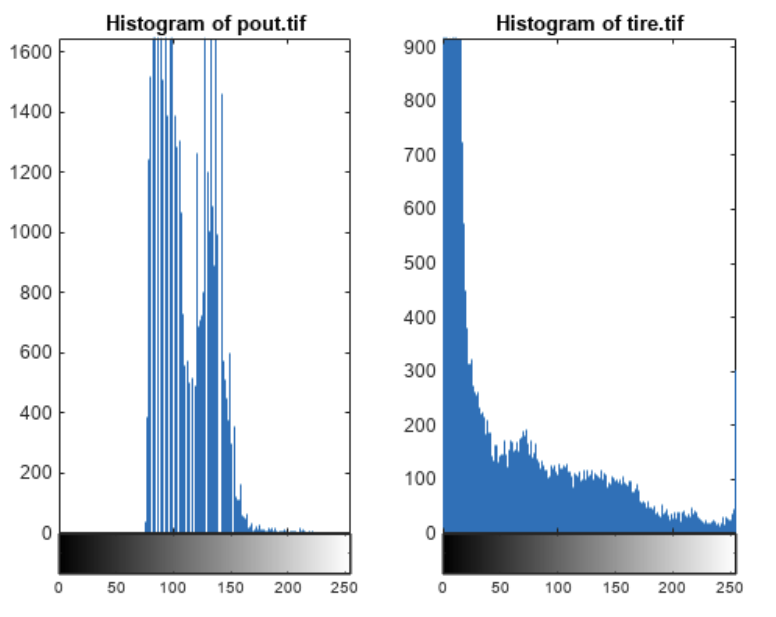
imhist(pout)

title("Histogram of pout.tif")

subplot(1,2,2)

imhist(tire)

title("Histogram of tire.tif");



而直方图均衡化明显改变了两个图像。许多先前隐藏的特征暴露出来，尤其是轮胎上的碎屑颗粒。遗憾的是，这种增强同时使两个图像的几个区域都出现过饱和。注意轮胎的中心、孩子脸部的一部分和外套都泛白。

对于轮胎图像的处理，车轮的中心最好保持大约相同的亮度，同时增强图像其他区域的对比度。为了实现此点，必须对图像的不同部分应用不同变换。在 adapthisteq 中实现的对比度受限自适应直方图均衡化方法可以实现这一点。该算法分析图像的各部分并计算适当的变换。它还可以设置对比度增强水平的限制，从而防止由 histeq 的基本直方图均衡化方法引起的过饱和。这是此示例中最复杂的方法。

1. 增强彩色图像

彩色图像的对比度增强通常是通过将图像转换为以图像亮度作为其分量之一的颜色空间（例如 L\*a\*b\* 颜色空间）来完成的。对比度调整仅对亮度层 L\* 执行，然后将图像转换回 RGB 颜色空间。操作亮度只影响像素的强度，会保留原始颜色。

将一个对比度差的图像读入工作区中。然后，将图像从 RGB 颜色空间转换为 L\*a\*b\* 颜色空间。

shadow = imread("lowlight\_1.jpg");

shadow\_lab = rgb2lab(shadow);

亮度值的范围是从 0 到 100。将值缩放到范围 [0 1]，这是数据类型为 double 的图像的预期范围。

max\_luminosity = 100;

L = shadow\_lab(:,:,1)/max\_luminosity;

对亮度通道执行三种类型的对比度调整，并保持 a\* 和 b\* 通道不变。将图像转换回 RGB 颜色空间。

shadow\_imadjust = shadow\_lab;

shadow\_imadjust(:,:,1) = imadjust(L)\*max\_luminosity;

shadow\_imadjust = lab2rgb(shadow\_imadjust);

shadow\_histeq = shadow\_lab;

shadow\_histeq(:,:,1) = histeq(L)\*max\_luminosity;

shadow\_histeq = lab2rgb(shadow\_histeq);

shadow\_adapthisteq = shadow\_lab;

shadow\_adapthisteq(:,:,1) = adapthisteq(L)\*max\_luminosity;

shadow\_adapthisteq = lab2rgb(shadow\_adapthisteq);

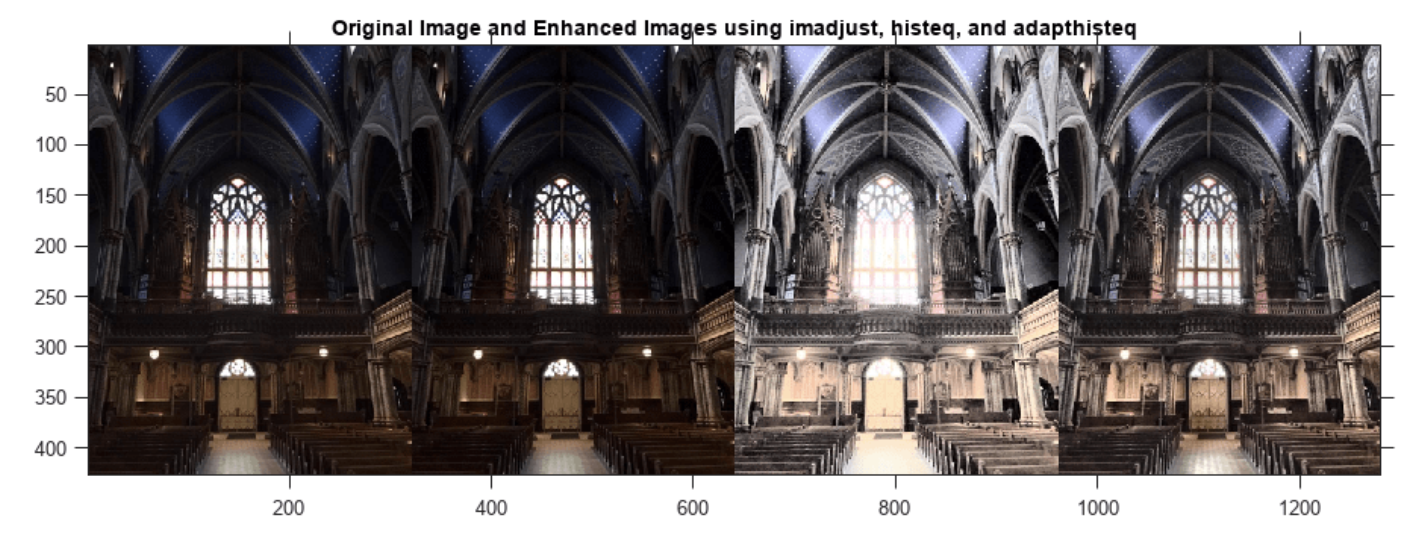
以蒙太奇方式显示原始图像和三个对比度调整后的图像。

figure

montage({shadow,shadow\_imadjust,shadow\_histeq,shadow\_adapthisteq},"Size",[1 4])

title("Original Image and Enhanced Images using " + ...

"imadjust, histeq, and adapthisteq")



1. 使用 L\*a\*b\* 颜色空间实现基于颜色的分割

此示例说明如何通过分析 L\*a\*b\* 颜色空间来识别织物中的不同颜色。

1. 采集图像

在 fabric.png 图像中读取，这是一个彩色织物图像。

fabric = imread("fabric.png");

imshow(fabric)

title("Fabric")



1. 基于 L\*a\*b\* 颜色空间计算每个区域的样本颜色

可以在图像中看到六种主要颜色：背景色、红色、绿色、紫色、黄色和洋红色。请注意，你可以很轻松地在视觉上区分这些颜色。L\*a\*b\* 颜色空间（也称为 CIELAB 或 CIE L\*a\*b\*）使你能够量化这些视觉差异。

L\*a\*b\* 颜色空间是从 CIE XYZ 三色值派生的。L\*a\*b\* 空间包含亮度层 'L\*'、色度层 'a\*'（表示颜色落在沿红-绿轴的位置）和色度层 'b\*'（表示颜色落在沿蓝-黄轴的位置）。

方法是为每种颜色选择一个小样本区域，并计算每个样本区域的基于 'a\*b\*' 空间的平均颜色。将使用这些颜色标记对每个像素进行分类。

为了简化此示例，请加载存储在 MAT 文件中的区域坐标。

load regioncoordinates;

nColors = 6;

sample\_regions = false([size(fabric,1) size(fabric,2) nColors]);

for count = 1:nColors

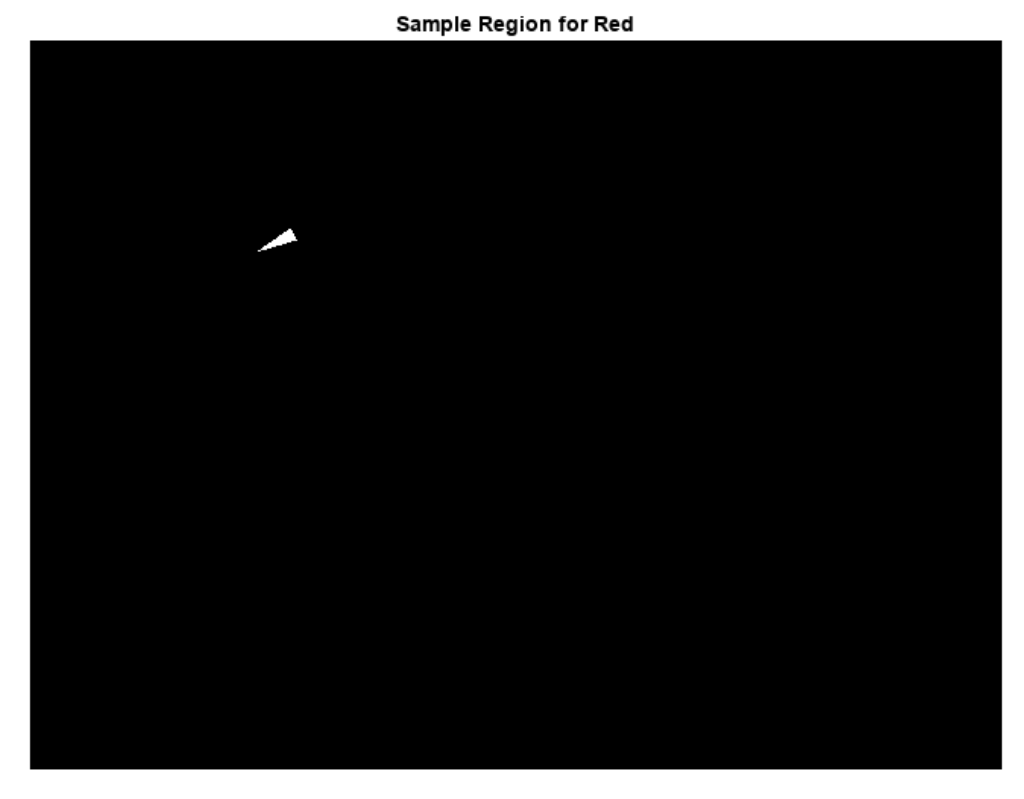
sample\_regions(:,:,count) = roipoly(fabric,region\_coordinates(:,1,count), ...

region\_coordinates(:,2,count));

end

imshow(sample\_regions(:,:,2))

title("Sample Region for Red")



使用 rgb2lab 函数将您的织物 RGB 图像转换为 L\*a\*b\* 图像。

lab\_fabric = rgb2lab(fabric);

计算用 roipoly 提取的每个区域的 'a\*' 和 'b\*' 均值。这些值用作基于 'a\*b\*' 空间的颜色标记。

a = lab\_fabric(:,:,2);

b = lab\_fabric(:,:,3);

color\_markers = zeros([nColors, 2]);

for count = 1:nColors

color\_markers(count,1) = mean2(a(sample\_regions(:,:,count)));

color\_markers(count,2) = mean2(b(sample\_regions(:,:,count)));

end

例如，基于 a\*b\* 空间的红色样本区域平均颜色为

disp([color\_markers(2,1), color\_markers(2,2)]);

1. 使用最近邻规则对每个像素进行分类

现在每个颜色标记都有一个 a\* 和一个 b\* 值。可以通过计算 lab\_fabric 图像中每个像素与每个颜色标记之间的欧几里德距离对该像素进行分类。最小距离表示像素最接近该颜色标记。例如，如果某像素和红色标记之间的距离最小，则该像素将被标记为红色像素。

创建一个包含颜色标签的数组：0 表示背景，1 表示红色，2 表示绿色，3 表示紫色，4 表示洋红色，5 表示黄色。

color\_labels = 0:nColors-1;

初始化用于最近邻分类的矩阵。

a = double(a);

b = double(b);

distance = zeros([size(a), nColors]);

执行分类

for count = 1:nColors

distance(:,:,count) = ( (a - color\_markers(count,1)).^2 + ...

(b - color\_markers(count,2)).^2 ).^0.5;

end

[~,label] = min(distance,[],3);

label = color\_labels(label);

clear distance;

1. 显示最近邻分类的结果

标签矩阵包含织物图像中每个像素的颜色标签。使用标签矩阵按颜色分离原始织物图像中的对象。

rgb\_label = repmat(label,[1 1 3]);

segmented\_images = zeros([size(fabric), nColors],"uint8");

for count = 1:nColors

color = fabric;

color(rgb\_label ~= color\_labels(count)) = 0;

segmented\_images(:,:,:,count) = color;

end

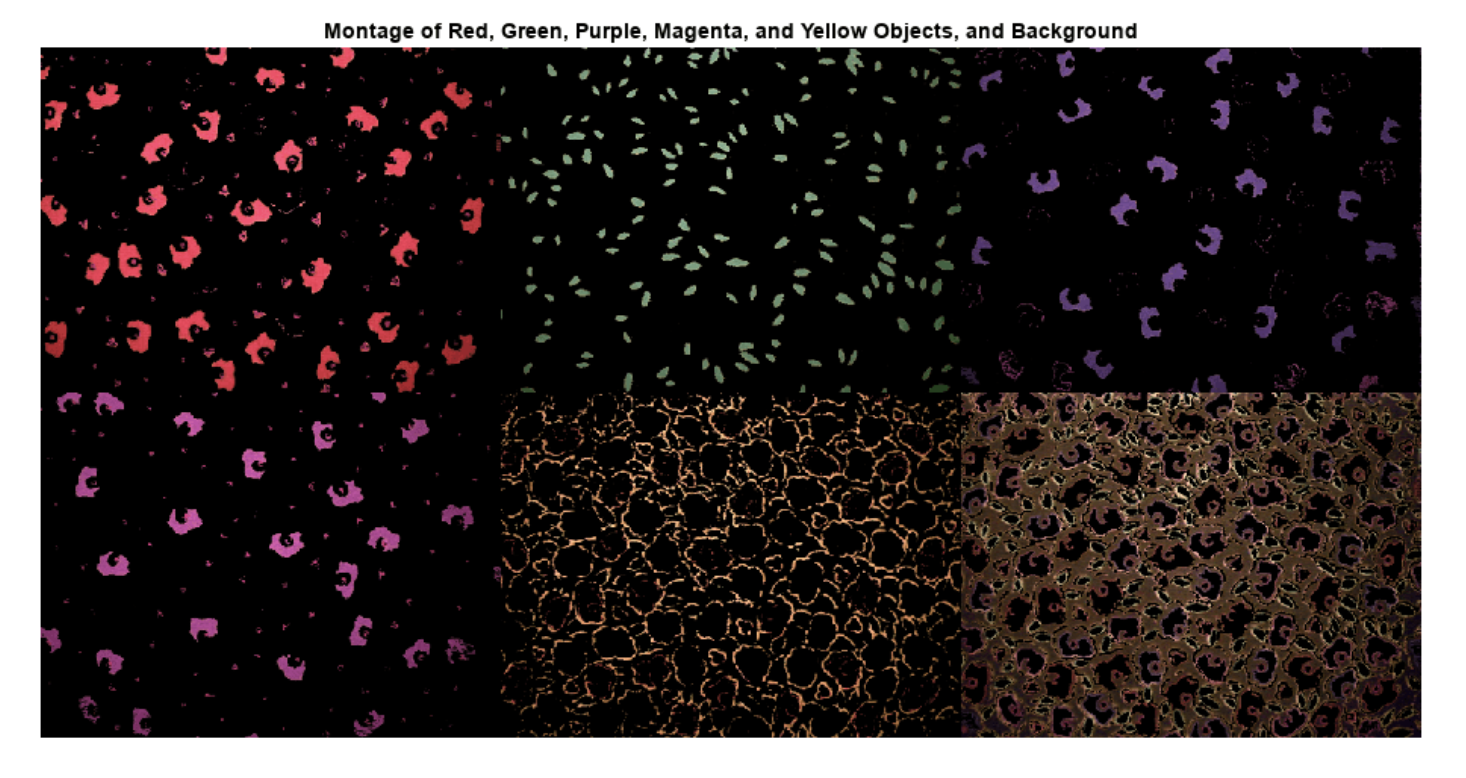
以蒙太奇方式显示五种分割颜色。同时显示图像中未归类为颜色的背景像素。

montage({segmented\_images(:,:,:,2),segmented\_images(:,:,:,3) ...

segmented\_images(:,:,:,4),segmented\_images(:,:,:,5) ...

segmented\_images(:,:,:,6),segmented\_images(:,:,:,1)});

title("Montage of Red, Green, Purple, Magenta, and Yellow Objects, and Background")



1. 显示已标注颜色的 a\* 和 b\* 值

通过绘制分类为不同颜色的像素的 a\* 和 b\* 值，可以看到最近邻分类对不同颜色总体的区分情况。出于显示目的，用颜色标签标注每个点。紫色不是已命名颜色值，因此使用具有十六进制颜色代码的字符串来指定紫色。

purple = "#774998";

plot\_labels = ["k", "r", "g", purple, "m", "y"];

figure

for count = 1:nColors

plot\_label = plot\_labels(count);

plot(a(label==count-1),b(label==count-1),".", ...

MarkerEdgeColor=plot\_label,MarkerFaceColor=plot\_label);

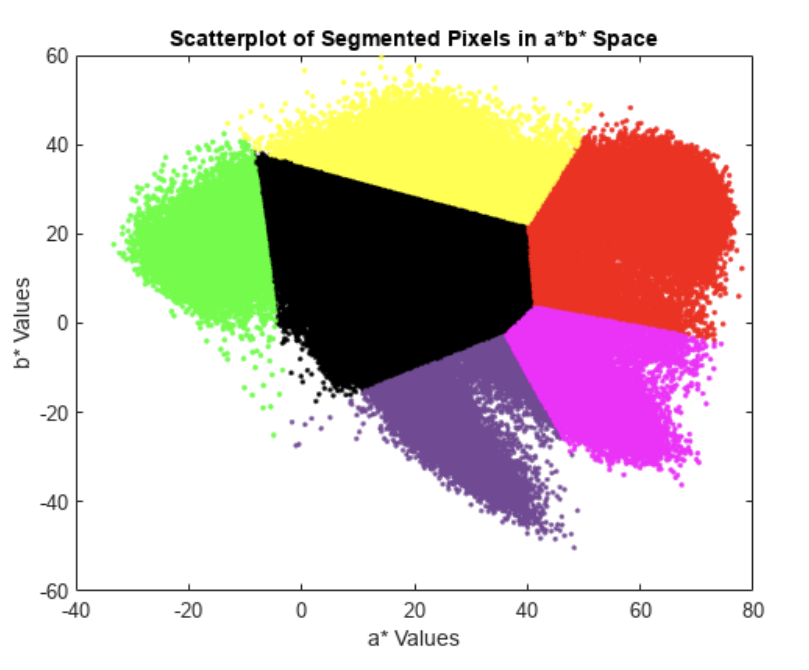
hold on

end

title("Scatterplot of Segmented Pixels in a\*b\* Space");

xlabel("a\* Values");

ylabel("b\* Values");



1. 使用边缘检测和形态学检测细胞

此示例说明如何使用边缘检测和基本形态学检测细胞。如果某对象与背景有足够的对比度，则可以在图像中轻松检测到该对象。

1. 读取图像

在 cell.tif 图像中读取，这是一个前列腺癌细胞的图像。此图像中存在两个细胞，但只有一个细胞完整显示。目标是检测或分割完整显示的细胞。

I = imread('cell.tif');

imshow(I)

title('Original Image');

text(size(I,2),size(I,1)+15, ...

'Image courtesy of Alan Partin', ...

'FontSize',7,'HorizontalAlignment','right');

text(size(I,2),size(I,1)+25, ....

'Johns Hopkins University', ...

'FontSize',7,'HorizontalAlignment','right');



1. 检测整个细胞

要分割的对象与背景图像的对比度相差很大。计算图像梯度的算子可以检测到对比度的变化。要创建包含分割后的细胞的二值掩膜，请计算梯度图像并应用一个阈值。

使用 edge 和索贝尔算子计算阈值。调整阈值，再次使用 edge 获得包含分割后的细胞的二值掩膜。

[~,threshold] = edge(I,'sobel');

fudgeFactor = 0.5;

BWs = edge(I,'sobel',threshold \* fudgeFactor);

显示生成的二元梯度掩膜。

imshow(BWs)

title('Binary Gradient Mask')



1. 膨胀图像

二元梯度掩膜显示图像中高对比度的线条。这些线条没有很好地描绘出感兴趣的对象的轮廓。与原始图像相比，梯度掩膜中对象周围的线条有间隙。如果使用线性结构元素膨胀索贝尔图像，这些线性间隙将消失。使用 strel 函数创建两个垂直线性结构元素。

se90 = strel('line',3,90);

se0 = strel('line',3,0);

先后使用垂直结构元素和水平结构元素，来膨胀二元梯度掩膜。使用 imdilate 函数膨胀图像。

BWsdil = imdilate(BWs,[se90 se0]);

imshow(BWsdil)

title('Dilated Gradient Mask')



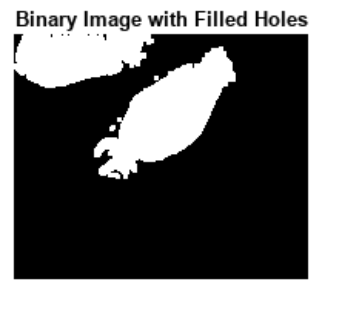
1. 填补内部间隙

膨胀的梯度掩膜很好地显示了细胞的轮廓，但细胞内部仍有小孔。要填充这些孔洞，请使用 imfill 函数。

BWdfill = imfill(BWsdil,'holes');

imshow(BWdfill)

title('Binary Image with Filled Holes')



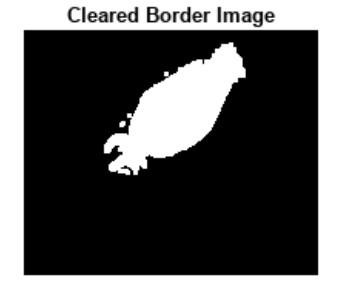
1. 删除边界上的连通对象

感兴趣的细胞已成功分割，但它不是被发现的唯一对象。可以使用 imclearborder 函数删除任何与图像边界连通的对象。要删除对角线连通，请将 imclearborder 函数中的连通性设置为 4。

BWnobord = imclearborder(BWdfill,4);

imshow(BWnobord)

title('Cleared Border Image')



1. 平滑处理对象

最后，为了使分割后的对象看起来自然，用菱形结构元素对图像腐蚀两次来平滑处理对象。使用 strel 函数创建菱形结构元素。

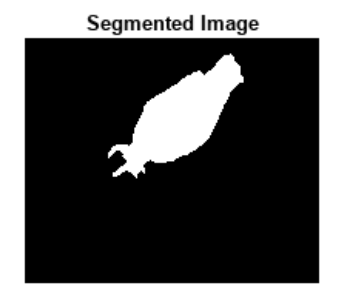
seD = strel('diamond',1);

BWfinal = imerode(BWnobord,seD);

BWfinal = imerode(BWfinal,seD);

imshow(BWfinal)

title('Segmented Image');



1. 可视化分割

您可以使用 labeloverlay 函数在原始图像上显示掩膜。

imshow(labeloverlay(I,BWfinal))

title('Mask Over Original Image')



显示分割后的对象的另一种方法是在分割的细胞周围绘制轮廓。使用 bwperim 函数绘制轮廓。

BWoutline = bwperim(BWfinal);

Segout = I;

Segout(BWoutline) = 255;

imshow(Segout)

title('Outlined Original Image')

