Basic Image Enhancement and Analysis Techniques

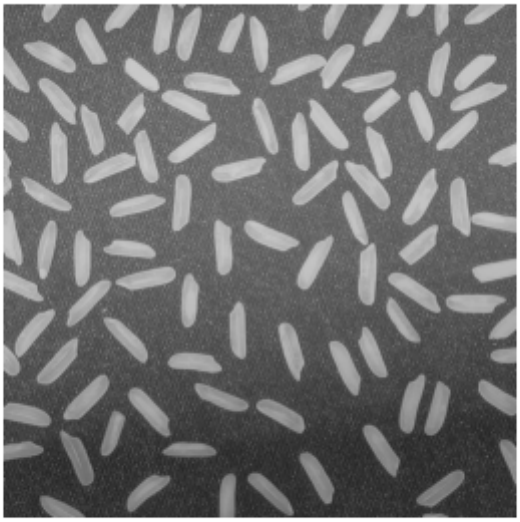
This example shows how to enhance an image as a preprocessing step before analysis. In this example, you correct the nonuniform background illumination and convert the image into a binary image so that you can perform analysis of the image foreground objects.

Open Script

Step 1: Read the Image into the Workspace

Read and display the grayscale image `rice.png`.

```
I = imread('rice.png');
imshow(I)
```



Step 2: Preprocess the Image to Enable Analysis

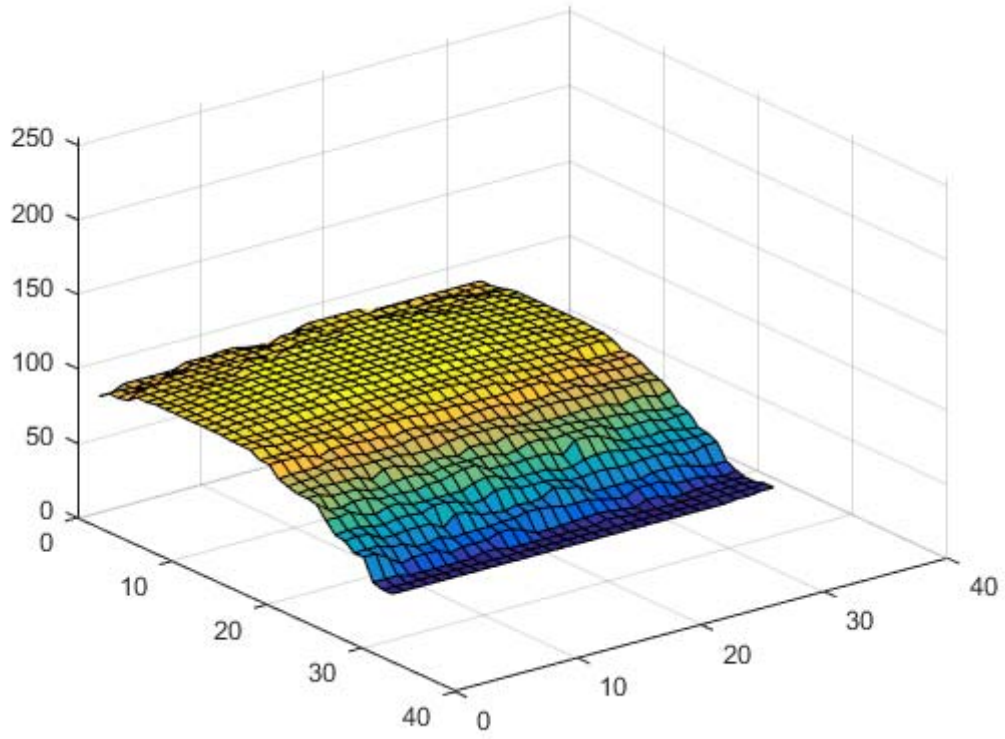
In the sample image, the background illumination is brighter in the center of the image than at the bottom. As a preprocessing step before analysis, make the background uniform and then convert the image into a binary image. To make the background illumination more uniform, create an approximation of the background as a separate image and then subtract this approximation from the original image.

As a first step to creating a background approximation image, remove all the foreground (rice grains) using morphological opening. The opening operation has the effect of removing objects that cannot completely contain the structuring element. To remove the rice grains from the image, the structuring element must be sized so that it cannot fit entirely inside a single grain of rice. The example calls the `strel` function to create a disk-shaped structuring element with a radius of 15.

```
background = imopen(I,strel('disk',15));
```

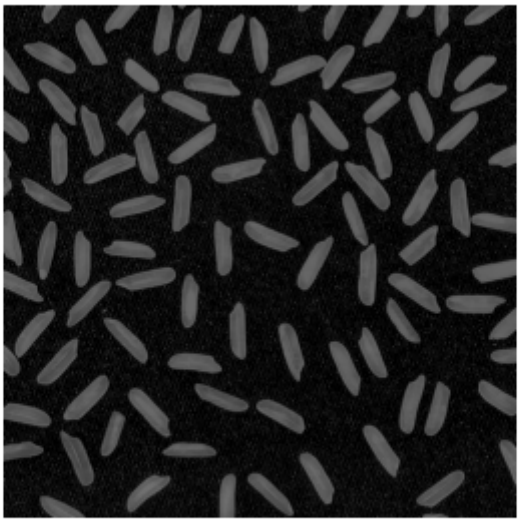
View the background approximation image as a surface to see where illumination varies. The `surf` command creates colored parametric surfaces that enable you to view mathematical functions over a rectangular region. Because the `surf` function requires data of class `double`, you first need to convert `background` using the `double` command. The example uses indexing syntax to view only 1 out of 8 pixels in each direction; otherwise, the surface plot would be too dense. The example also sets the scale of the plot to better match the range of the `uint8` data and reverses the y-axis of the display to provide a better view of the data. (The pixels at the bottom of the image appear at the front of the surface plot.) In the surface display, [0, 0] represents the origin, or upper-left corner of the image. The highest part of the curve indicates that the highest pixel values of background (and consequently `rice.png`) occur near the middle rows of the image. The lowest pixel values occur at the bottom of the image.

```
figure
surf(double(background(1:8:end,1:8:end))),zlim([0 255]);
set(gca,'ydir','reverse');
```



Subtract the background approximation image, `background`, from the original image, `I`, and view the resulting image. After subtracting the adjusted background image from the original image, the resulting image has a uniform background but is now a bit dark for analysis.

```
I2 = I - background;
imshow(I2)
```



Use `imadjust` to increase the contrast of the processed image `I2` by saturating 1% of the data at both low and high intensities and by stretching the intensity values to fill the `uint8` dynamic range.

```
I3 = imadjust(I2);
imshow(I3);
```

Create a binary version of the processed image so you can use toolbox functions for analysis. Use the `im2bw` function to convert the grayscale image into a binary image by using thresholding. The function `graythresh` automatically computes an appropriate threshold to use to convert the grayscale image to binary. Remove background noise with the `bwareaopen` function.

```
level = graythresh(I3);
bw = im2bw(I3,level);
bw = bwareaopen(bw, 50);
imshow(bw)
```

Step 3: Perform Analysis of Objects in the Image

Now that you have created a binary version of the original image you can perform analysis of objects in the image.

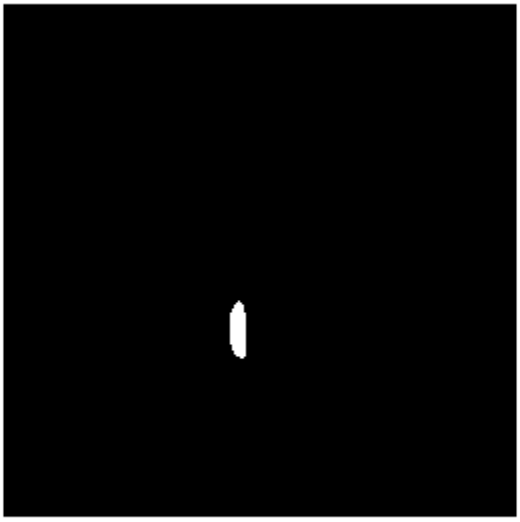
Find all the connected components (objects) in the binary image. The accuracy of your results depends on the size of the objects, the connectivity parameter (4, 8, or arbitrary), and whether or not any objects are touching (in which case they could be labeled as one object). Some of the rice grains in the binary image `bw` are touching.

```
cc = bwconncomp(bw, 4)
cc.NumObjects
```

```
cc =

  struct with fields:

    Connectivity: 4
       ImageSize: [256 256]
      NumObjects: 95
    PixelIdxList: {1×95 cell}


ans =

    95
```

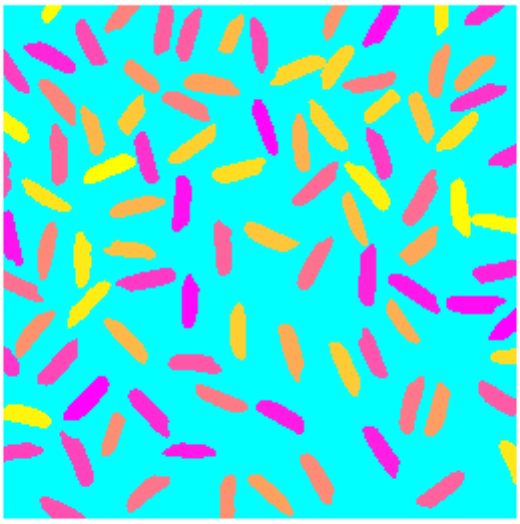
View the rice grain that is labeled 50 in the image.

```
grain = false(size(bw));
grain(cc.PixelIdxList{50}) = true;
imshow(grain);
```



Visualize all the connected components in the image. First, create a label matrix, and then display the label matrix as a pseudocolor indexed image. Use `labelmatrix` to create a label matrix from the output of `bwconncomp`. Note that `labelmatrix` stores the label matrix in the smallest numeric class necessary for the number of objects. Since `bw` contains only 95 objects, the label matrix can be stored as `uint8` . In the pseudocolor image, the label identifying each object in the label matrix maps to a different color in an associated colormap matrix. Use `label2rgb` to choose the colormap, the background color, and how objects in the label matrix map to colors in the colormap.

```
labeled = labelmatrix(cc);
RGB_label = label2rgb(labeled, @spring, 'c', 'shuffle');
imshow(RGB_label)
```



Compute the area of each object in the image using `regionprops`. Each rice grain is one connected component in the `cc` structure.

```
graindata = regionprops(cc, 'basic')
```

```
graindata =

  95×1 struct array with fields:

    Area
    Centroid
    BoundingBox
```

Find the area of the 50th component, using dot notation to access the Area field in the 50th element of `graindata`.

```
graindata(50).Area
```

```
ans =

   194
```

Create a vector `grain_areas` to hold the area measurement of each object (rice grain).

```
grain_areas = [graindata.Area];
```

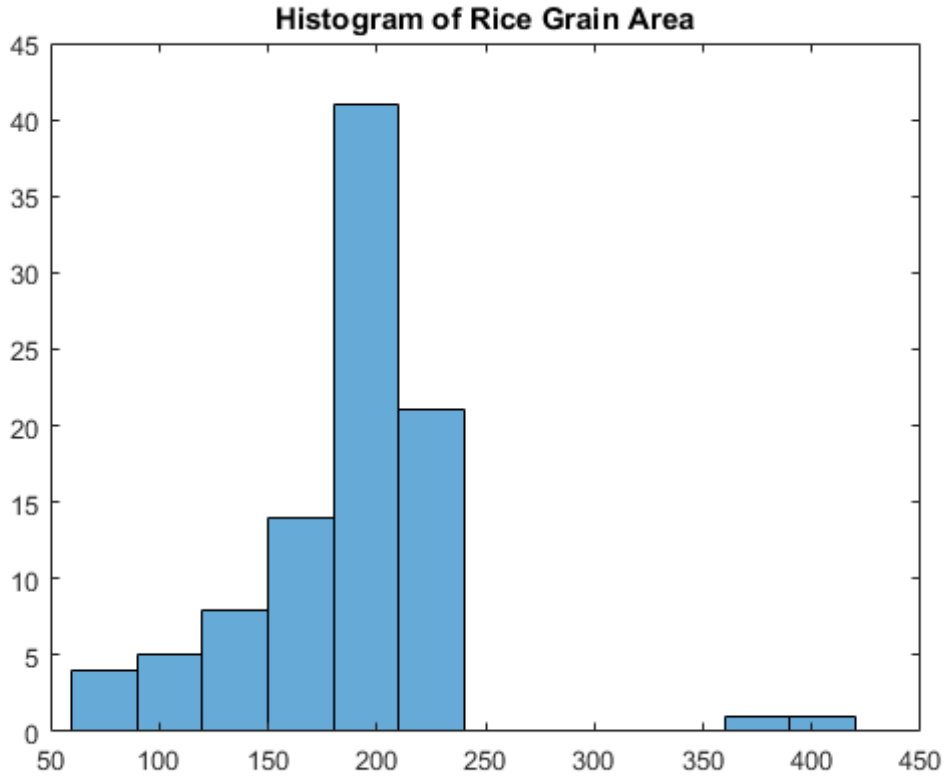
Find the rice grain with the smallest area.

```
[min_area, idx] = min(grain_areas)
grain = false(size(bw));
grain(cc.PixelIdxList{idx}) = true;
imshow(grain);
```

```
min_area =

    61


idx =

    16
```

Use the `histogram` command to create a histogram of rice grain areas.

```
figure
histogram(grain_areas)
title('Histogram of Rice Grain Area');
```


Histogram of Rice Grain Area
45
40
35
30
25
20
15
10
5
0
50
100
150
200
250
300
350
400
450