**MILESTONE 4**

Step 1: Read Image

### Step 2: Use Morphological Opening to Estimate the Background

### Step 3: View the Background Approximation as a Surface

### Step 4: Subtract the Background Image from the Original Image

### Step 5: Increase the Image Contrast

### Step 6: Threshold the Image

### Step 7: Identify Objects in the Image

### Step 8: Examine One Object

### Step 9: View All Objects

### Step 10: Compute Area of Each Object

### Step 11: Compute Area-based Statistics

### Step 12: Create Histogram of the Area

**Introduction**

Using an image of rice grains, this example illustrates how you can enhance an image to correct for nonuniform illumination, and then use the enhanced image to identify individual grains. This enables you to learn about the characteristics of the grains and easily compute statistics for all the grains in the image.

**Step 1: Read Image**

Read and display the grayscale image rice.png.

I = imread('rice.png');

imshow(I)

I=rgb2gray(I);

I = rgb2gray(RGB) converts the truecolor image RGB to the grayscale intensity image I. rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

**Grayscale Image rice.png**



### Step 2: Use Morphological Opening to Estimate the Background

In the sample image, the background illumination is brighter in the center of the image than at the bottom. In this step, the example uses a morphological opening operation to estimate the background illumination. Morphological opening is an erosion followed by a dilation, using the same structuring element for both operations. The opening operation has the effect of removing objects that cannot completely contain the structuring element. For more information about morphological image processing, see [Morphological Operations](jar:file:///C:/Program%20Files%20%28x86%29/MATLAB/R2010a/help/toolbox/images/help.jar%21/f18-9530.html).

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

The example calls the imopen function to perform the morphological opening operation. Note how the example calls the strel function to create a disk-shaped structuring element with a radius of 15. 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.

### Step 3: View the Background Approximation as a Surface

Use the surf command to create a surface display of the background (the background approximation created in Step 2). The surf command creates colored parametric surfaces that enable you to view mathematical functions over a rectangular region. However, the surf function requires data of class double, so you first need to convert background using the double command:

figure, surf(double(background(1:8:end,1:8:end))),zlim([0 255]);

set(gca,'ydir','reverse');

The example uses MATLAB 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 and are represented in the surface plot by the lowest part of the curve.

The surface plot is a Handle Graphics® object. You can use object properties to fine-tune its appearance. For more information, see the [surface](jar:file:///C:/Program%20Files%20%28x86%29/MATLAB/R2010a/help/techdoc/help.jar%21/ref/surface.html) reference page.

**Surface Plot**

### Step 4: Subtract the Background Image from the Original Image

To create a more uniform background, subtract the background image, background, from the original image, I, and then view the image:

I2 = I - background;

figure, imshow(I2)

**Image with Uniform Background**



**Step 5: Increase the Image Contrast**

After subtraction, the image has a uniform background but is now a bit too dark. Use imadjust to adjust the contrast of the image. imadjust increases the contrast of the image by saturating 1% of the data at both low and high intensities of I2 and by stretching the intensity values to fill the uint8 dynamic range. See the [imadjust](jar:file:///C:/Program%20Files%20%28x86%29/MATLAB/R2010a/help/toolbox/images/help.jar%21/imadjust.html) reference page for more information.

The following example adjusts the contrast in the image created in the previous step and displays it:

I3 = imadjust(I2);

figure, imshow(I3);

**Image After Intensity Adjustment**



### Step 6: Threshold the Image

Create a binary version of the image so you can use toolbox functions to count the number of rice grains. 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 bwareaopen:

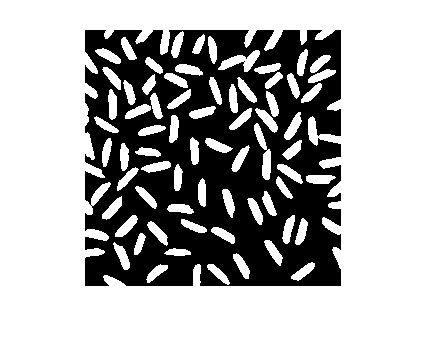
level = graythresh(I3);

bw = im2bw(I3,level);

bw = bwareaopen(bw, 50);

figure, imshow(bw)

**Binary Version of the Image**



### Step 7: Identify Objects in the Image

The function bwconncomp finds 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 bw are touching.

Enter the following at the command line:

cc = bwconncomp(bw, 4)

cc.NumObjects

You will receive the output shown below:

cc =

Connectivity: 4

ImageSize: [256 256]

NumObjects: 95

PixelIdxList: {1x95 cell}

ans = 95

**Step 8: Examine One Object**

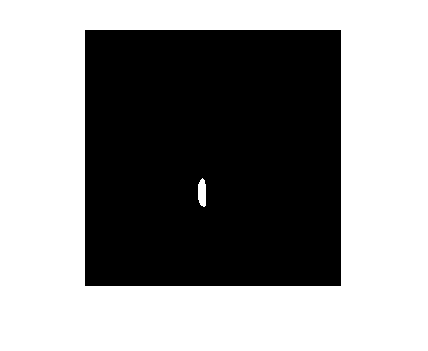
Each distinct object is labeled with the same integer value. Show the grain that is the 50th connected component:

grain = false(size(bw));

grain(cc.PixelIdxList{50}) = true;

figure, imshow(grain);

**The 50th Connected Component**



### Step 9: View All Objects

One way to visualize connected components is to create a label matrix, and then display it as a pseudo-color 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:

labeled = labelmatrix(cc);

whos labeled

Name Size Bytes Class Attributes

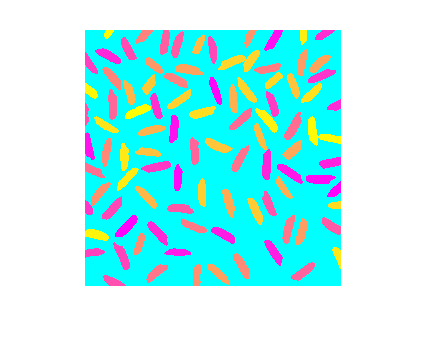
labeled 256x256 65536 uint8

In the pseudo-color image, the label identifying each object in the label matrix maps to a different color in the 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:

RGB\_label = label2rgb(labeled, @spring, 'c', 'shuffle');

figure, imshow(RGB\_label)

**Label Matrix Displayed as Pseudocolor Image**



### Step 10: Compute Area of Each Object

Each rice grain is one connected component in the cc structure. Use regionprops on cc to compute the area.

graindata = regionprops(cc, 'basic')

MATLAB responds with

graindata =

95x1 struct array with fields:

Area

Centroid

BoundingBox

To find the area of the 50th component, use dot notation to access the Area field in the 50th element of graindata structure array:

graindata(50).Area

ans =

194

### Step 11: Compute Area-based Statistics

Create a new vector allgrains to hold the area measurement for each grain:

grain\_areas = [graindata.Area];

Find the grain with the smallest area:

[min\_area, idx] = min(grain\_areas)

grain = false(size(bw));

grain(cc.PixelIdxList{idx}) = true;

figure, imshow(grain);

min\_area =

61

idx =

16

**Smallest Grain**



### 

### Step 12: Create Histogram of the Area

Use hist to create a histogram of the rice grain area:

nbins = 20;

figure, hist(grain\_areas, nbins)

title('Histogram of Rice Grain Area');

