Table of Contents

Theoretical Background

Types of images

Digital images are stored in several different formats and resolutions. Image processing toolbox provides the support functions for conversion between different type of images. There are many different types of images used which have different color distribution, few are listed as follow:

Binary images

The type of images which have only two levels and their intensity values are represented by a single bit (0 or 1) are called binary images. The function 'im2bw'converts given image to binary image.

e.g., p=im2bw(A);

Grayscale images

The type of images which is composed of shades of gray are called grayscale images. These images are also called single channel images.

RGB images

The type of images which is composed of the combination of three colors namely red, green and blue are called RGB images (3 channel) or color images. The function rgb2gray is used to convert RGB images to grayscale.

e.g.,colorimage=imread('filename');
a=rgb2gray(colorimage);

Matlab data types

The variables used for storing matrices in MATLAB may have different numeric data types. They are listed in the below table:

Data type	Description	Range
int8	8-bit integer	−128 — 127
uint8	8-bit unsigned integer	0 — 255
int16	16-bit integer	−32768 — 32767
uint16	16-bit unsigned integer	0 — 65535
double	Double precision real number	Machine specific

Image Complement

In the complement of a binary image, zeros become ones and ones become zeros. Black and white are reversed.

In the complement of a grayscale or color image, each pixel value is subtracted from the maximum pixel value supported by the class (or 1.0 for double-precision images). The difference is used as the pixel value in the output image. In the output image, dark areas become lighter and light areas become darker. For color images, reds become cyan, greens become magenta, blues become yellow, and vice versa.

Boundary Tracing in Images

The bwtraceboundary function returns the row and column coordinates of all the pixels on the border of an object in an image. You must specify the location of a border pixel on the object as the starting point for the trace.

The bwboundaries function returns the row and column coordinates of border pixels of all the objects in an image.

For both functions, the nonzero pixels in the binary image belong to an object, and pixels with the value 0 (zero) constitute the background.

Edge detection methods for finding object boundaries in images

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods.

Edge Detection

Edge Detection is simply a case of trying to find the regions in an image where we have a sharp change in intensity or a sharp change in color, a high value indicates a steep change and a low value indicates a shallow change.

Holes Finding

Labeling of connected components and calculating the Euler number in a binary image are two fundamental operations in image analysis, pattern recognition, computer (robot) vision, and machine intelligence. Labeling is necessary whenever independent objects (connected components) are to be recognized in a binary image. On the other hand, the Euler number, which is defined as the difference between the number of connected components and that of holes in a binary image, is a basic topologic property of a binary image and is used for processing cell images in medical diagnosis, document image processing, shadow detection, reflectance-based object recognition, and robot vision.

Two of the commonly used are:

Highly Connected Subgraphs (HCS)

The HCS (Highly Connected Subgraphs) clustering algorithm [1] (also known as the HCS algorithm, and other names such as Highly Connected Clusters/Components/Kernels) is an algorithm based on graph connectivity for cluster analysis. It works by representing the similarity data in a similarity graph, and then finding all the highly connected subgraphs. It does not make any prior assumptions on the number of the clusters. This algorithm was published by Erez Hartuv and Ron Shamir in 1998.

The HCS algorithm gives a clustering solution, which is inherently meaningful in the application domain, since each solution cluster must have diameter 2 while a union of two solution clusters will have diameter 3.

Machine Learning Algorithms

Image classification can be accomplished by any machine learning algorithms (logistic regression, random forest and SVM). But all the machine learning algorithms required proper features for doing the classification. If you feed the raw image into the classifier, it will fail to classify the images properly and the accuracy of the classifier would be less.

CNN (convolution neural network) extract the features from the images and it handles the entire feature engineering part. In normal CNN architecture, beginning layers are extracting the low-level features and end level layers extract high-level features from the image.

Before CNN, we need to spend time on selecting the proper features for classifying the image. There are so many handcrafted features available (local feature, global feature), but it will take so much time to select the proper features for a solution (image classification) and selecting the proper classification model. CNN handles all these problems and the accuracy of the CNN is higher compared with the normal classifier.

MATLAB coding with results:

```
clc;
clear;
close all;
```

Read the Input Image

```
InputImage=imread('analysis.jpg');
% Display the Input Image
figure(1)
imshow(InputImage);title('InputImage');
```

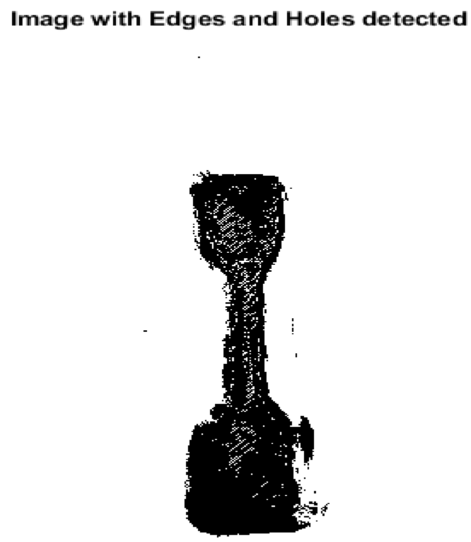
```
Warning: Image is too big to fit on screen; displaying at 33%
```

InputImage

Convert the Gray Scale Image to Binary Image

```
BinaryImage=im2bw(InputImage);
%Display Binary Image
figure(2)
imshow(BinaryImage);
title('Image with Edges and Holes detected')
```

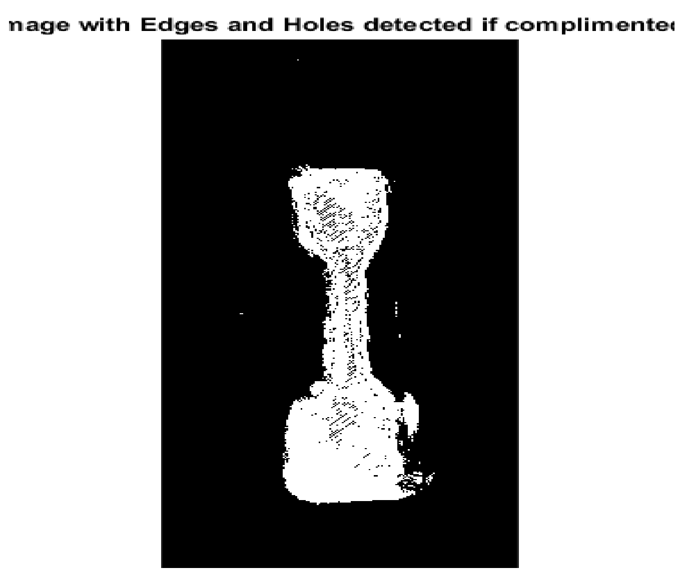
```
Warning: Image is too big to fit on screen; displaying at 33%
```



Complement the Binary Image

```
ComplementImage=imcomplement(BinaryImage);
figure(3)
imshow(ComplementImage);
title('Image with Edges and Holes detected if complimented')
```

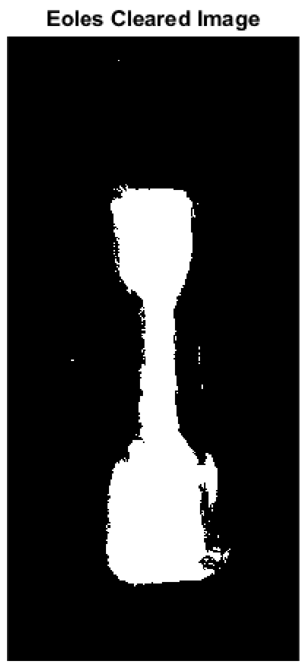
```
Warning: Image is too big to fit on screen; displaying at 33%
```



Perform morphological operations

```
%Fill in the holes of a binary image
HolesClearedImage = imfill(ComplementImage,'holes');
%Display Holes Cleared Image
figure(4)
imshow(HolesClearedImage);title('Eoles Cleared Image');
%Apply labels for each binary connected component
%Apply Labels for each  connected components in 2-D binary image
[L,Num] = bwlabel(HolesClearedImage);
```

```
Warning: Image is too big to fit on screen; displaying at 33%
```



Calculation of Area marked by yellow line

```
[r,c] = size(ComplementImage);
num_elements = numel(ComplementImage);
region = 0;
for i = 1:r
    for j = 1:c
        if ComplementImage(i,j) == 1
            region = region + 1;
        end
    end
end
disp('Area covered by the outlined box is ')
area_covered = (region/num_elements)*100
```

```
Area covered by the outlined box is

area_covered =

   13.1676
```

Drawing the boundary line

Calculate boundaries.

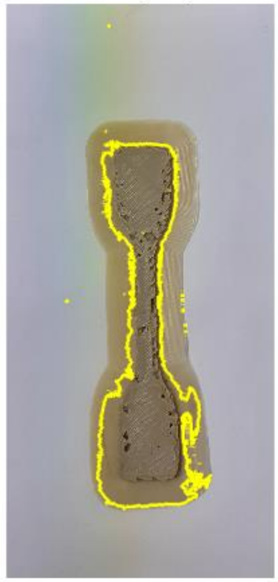
```
figure(5)
BW = imread('analysis.jpg');
[B,L,N] = bwboundaries(im2bw(BW));
```

Display object boundaries in yellow!

```
imshow(BW); hold on;
for k=1:length(B),
   boundary = B{k};
   if(k > N)
     plot(boundary(:,2), boundary(:,1), 'y','LineWidth',2);
     title('Outlined Portion having Edges and Boundary')
   end
end
```

```
Warning: Image is too big to fit on screen; displaying at 33%
```

Outlined Portion having Edges and Boundary



Displaying holes!

```
BW = imread('analysis.jpg');
figure(6)
%Calculate boundaries of regions in the image.
[B,L,N,A] = bwboundaries(im2bw(BW));

%Display the image with the boundaries overlaid.
%Add the region number next to every boundary (based on the label matrix).
%Use the zoom tool to read individual labels.
imshow(BW); hold on;
colors=['b' 'g' 'r' 'c' 'm' 'y'];
for k=1:length(B)
  boundary = B{k};
  cidx = mod(k,length(colors))+1;
  plot(boundary(:,2), boundary(:,1),...
       colors(cidx),'LineWidth',2);
   title('Holes detected with random Colours')
end
```

```
Warning: Image is too big to fit on screen; displaying at 33%
```

Published with MATLAB® R2017a

References

https://www.researchgate.net/publication/283585724_An_Image_Processing_Approach_to_Detect_Lanes_Pot_Holes_and_Recognize_Road_Signs_in_Indian_Roads

http://www.ncsa.illinois.edu/People/kindr/phd/PART1.PDF

http://www.ece.iit.edu/~ksuzuki/wp-content/uploads/2017/04/HeChaoSuzuki_CombinationalAlgoCCL_JReal-TimeImagProc2014.pdf

https://www.researchgate.net/publication/303868869_Hole-filling_Based_on_Disparity_Map_and_Inpainting_for_Depth-Image-Based_Rendering

https://uk.mathworks.com/help/images/boundary-tracing-in-images.html

https://uk.mathworks.com/help/images/ref/bwconncomp.html

https://uk.mathworks.com/matlabcentral/answers/175548-bwboundaries-explanation-of-a-example