

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 4602: Signals & Systems Lab
Digital Image Processing

Digital Image

- Digital images are stored in different file formats such as jpg, png, tif, svg, etc.
- The image can be a binary, grayscale, or color image. The figure below shows three different variations of an image.



- Binary image has two colors - black and white – represented using 0 and 1, respectively.
- Grayscale image consists of 256 shades of gray, from 0 to 255. It's a monochrome image.
- Color image (also called RGB image) consists of 3 channels – red, green, and blue. Each channel is a grayscale image.

Write the following codes to import and view an image in MATLAB.


<pre>img1 = imread('cameraman.tif') img2 = imread('yellowlily.jpg')</pre>	<p>This imports the image into MATLAB. The output is a numeric array, a 2D or 3D matrix depending on the type.</p>
---	--

<code>imshow(img)</code>	This will display the image.
--------------------------	------------------------------

- Pixels: Pixels are the smallest component of a digital image. The pixel values represent the intensity of light.
- Resolution: the resolution of the image refers to the number of pixels used to form the image. The `img1` above is a `256*256 uint8` matrix. So, there are a total of 65536 pixels. `img2` has a size of `1632*1224*3`, since it is an RGB image.

As you can see, `img2` has a higher resolution. With more resolution, more visual information can be captured. But resolution only does not guarantee the quality of the image. There are other factors that determine the quality of the image such as exposure, focus, sensor efficiency, etc.

Images can also be captured using other wavelengths that are invisible to the human eye. X-ray, Ultrasound, Infrared, etc. are examples of such imaging techniques. Different scientific images are usually captured this way. They have a wide range of applications in different domains.

 Write the following code to view an image in MATLAB's image viewer app.

<code>imtool(img2)</code>	<p>Hover the mouse over the image and you can see the associated pixel values on the bottom-left corner of the image.</p> <ul style="list-style-type: none"> ▪ Check out other options on the image viewer app.
---------------------------	--

<code>imfinfo('yellowlily.jpg')</code>	This will provide you with the meta-data of the image. The output is a structure datatype that contains different information (file size, for instance) regarding the image.
--	--

Image Conversion

You can easily convert an image from one type to another. You can use the following conversion functions –

- *im2bw* – converts an image to binary.
- *im2gray* – converts an RGB image to grayscale.

Any operation on images is usually conducted on the grayscale image. They require less memory (3x less compared to RGB) and are faster to operate on. The structural information of an object is intact in the grayscale image. So, using a color image is redundant as color information is often not needed.

Basic operations on images

You can easily perform some basic operations like rotating, resizing, cropping, transforming, etc. using the following functions. Check their operations with the help of documentation.

- *imrotate (img, angle)*
- *imcrop ()*
- *imresize (img, scaling factor)*
- *imtransform ()*

<i>imresize(img, 0.2)</i>	It will reduce the image resolution as well as the memory space.
<i>imrotate (img, 30)</i>	It will rotate the image by 30 degrees.
<i>imwrite(img, 'img_new.png')</i>	You can save the modified image using this function. Check out the ‘quality’ parameter from the documentation.

Gamma Correction

It's a simple image enhancement technique. It adjusts the brightness of the image through numeric operation.

$$Int_{out} = Int_{in}^{\gamma}$$

Here, Int refers to the intensity value of the pixels. The γ value controls the brightness. Try out different values of γ [0-1] and see the change in the input image.

One thing to consider while doing such arithmetic operations on images is that the pixel values are in the uint8 format. During the operations, values can easily get maxed out to 255. Therefore, you need to convert it to 'double' to avoid such scenarios.

$$img2 = im2double(img)$$

$$img3 = img2.^{0.2}$$

Thresholding

Look at the concrete image. You can see there is a crack. How to find what percentage of the concrete has a crack?

Thresholding is a segmentation technique that helps you differentiate between different patterns in the image. The basic idea is to divide the image into foreground and background segments. In the above scenario, the crack has a much darker shade than the concrete and we can utilize this to segment our ROI (region of interest).

There are different thresholding methods. It depends on the task which segmentation technique would work well.



For this case, a simple approach would be to divide the image pixel values into two segments – on one side you have the dark crack portion (low-intensity value), and on the other side, you have the

comparatively lighter portion (higher-intensity value). The challenge is in finding the optimal value that will properly differentiate the two segments.

MATLAB offers some built-in functions that can perform this task automatically.

```
a = imread('crack.jpg')  
a2 = im2gray(a)  
a3 = imbinarize(a2)  
montage ({a, a2, a3, ~a3})
```

The ‘imbinarize’ function converts the image into a binary image using a global threshold value, which is calculated using Otsu’s method (check documentation). To find the threshold that is used by the algorithm, use the below function –

```
th = graythresh(a2)  
pixel_value = th*255
```



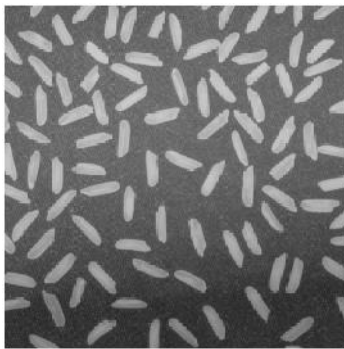
You can also manually try out different values for the threshold using the 2nd optional parameter inside the function. Try out the ‘numeric slider’ for flexibility in your live editor. Use the ‘montage’ function to compare different changes in the image.

🔧 Now, how to find out the percentage of cracks in the image?

- You can use the ‘nnz’ function to calculate the total number of white pixels (represents the crack).
- You can also automate the process for hundreds of concrete images using MATLAB’s image batch processor app.

- Thresholding approaches:
 - i. Global threshold
 - ii. Adaptive threshold
 - iii. Multilevel threshold

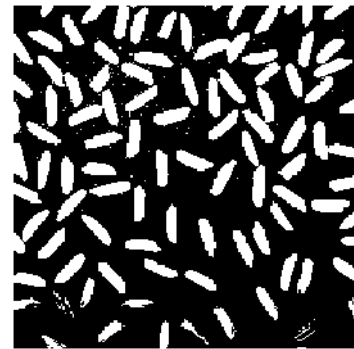
Global threshold is calculated using the entire image. However, if different portions of the image have different illumination/intensity, this approach might not work well. In such cases, adaptive threshold would work well. You can apply this using the same ‘imbinarize’ function with the 2nd parameter fixed as ‘adaptive’.



original rice image



global thresholding

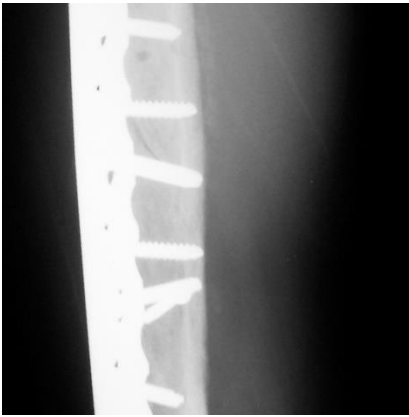


threshold value = 0.5

While using a threshold value of 0.5, you can see some noise in the image. There are other ways to mitigate this problem. Try adaptive thresholding and see how it works.

Some images, however, might require different label-based segmentation. Look at the following X-ray images. You can see the pattern is not really visible. The image cannot be divided into only foreground and background. It requires different labels to segment the tissue, bone, and metal from the background.

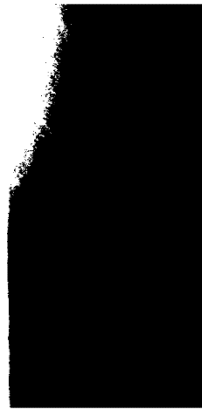
Use the following code to perform multi-level segmentation. You have to specify how many segments you want, in our case - 3.



original



binary (global)



binary (adaptive)

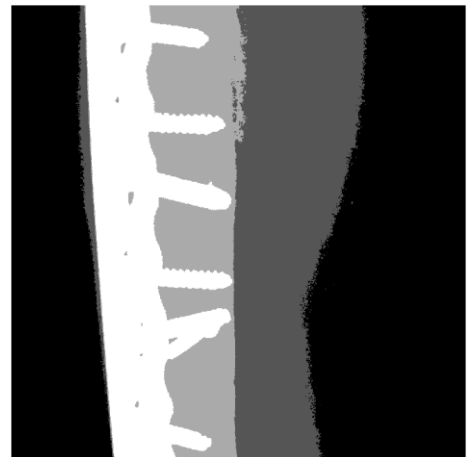
Now you can separate different segments in the x-ray image.

```
x= imread('armxray.png')
```

```
val= multithresh(x,3)
```

```
label= imquantize(x,val)
```

```
imshow (label,[ 1])
```



Task

How to make the writing clear?

What is Image Filtering in the Spatial Domain?

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement.

Filtering is a neighborhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. (See Neighborhood or Block Processing: An Overview for a general discussion of neighborhood operations.) *Linear filtering* is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

Convolution

Linear filtering of an image is accomplished through an operator called *convolution*. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*. A convolution kernel is a correlation kernel that has been rotated 180 degrees.

For example, suppose the image is

```

x = [ 12  24  3  8  55
      23  5  7  34  16
       4  4  13  20  22
      19  12  10  25  7 ]

```

Image Histogram

Image Histogram represents the distribution of the pixel intensity values. You can manipulate the histogram to change/enhance/adjust the image contrast.

```
ben=imread('ben.png')  
ben_gray= im2gray(ben)  
imhist(ben_gray)  
imadjust(ben_gray)  
histeq(ben_gray)  
adapthisteq(ben_gray)
```

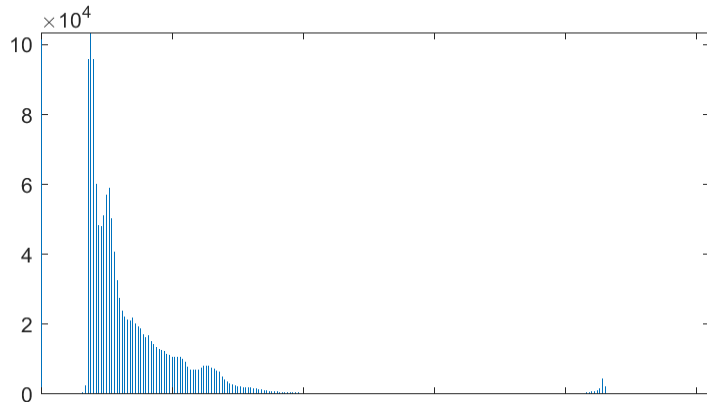


Image processing apps

MATLAB provides different image processing apps for easier manipulation of images. Check out the following apps –

- Image Viewer app
- Color thresholder app
- Image Segmenter app
- Image region analyzer app

Task:

- i. Import an image from the workspace in the image viewer app. Then use the ‘tools’ bar to adjust image contrast and see the outcome simultaneously.
- ii. Import a color image in the Color thresholder app and check other color spaces such as HSV, YCbCr, and L*a*b*. Modify the values and see the changes in the image.