**Introduction**

Before understanding Image Processing let us look in to what images are. Images for a computer is a two dimensional matrix of numbers. Each cell in the matrix represents the corresponding pixel in the image. Following illustration will make it more clear.

<Insert Image>

Image processing is the field of studying and analyzing an image. Using various properties of an image like the colors, structure and other fine details we try to extract meaningful information like edges, objects, contours and more. This information can then be used in various applications like medicine, security applications, social media services, self-driving cars. Some these are elaborated below. During the course of this book we will look into details of how to implement algorithms that can help us extract meaningful information from images.

Applications of Image Processing:

* Medicine

Medicine has seen a lot of advancement in the recent years due to rapid advancements in related fields. Image processing is such one field which has helped in the growth of Medicine. Tumors can be detected using image processing algorithms (figure 1.1) which helps the doctors to cure the patients. Another application in the medical field is the ultra Sonography.

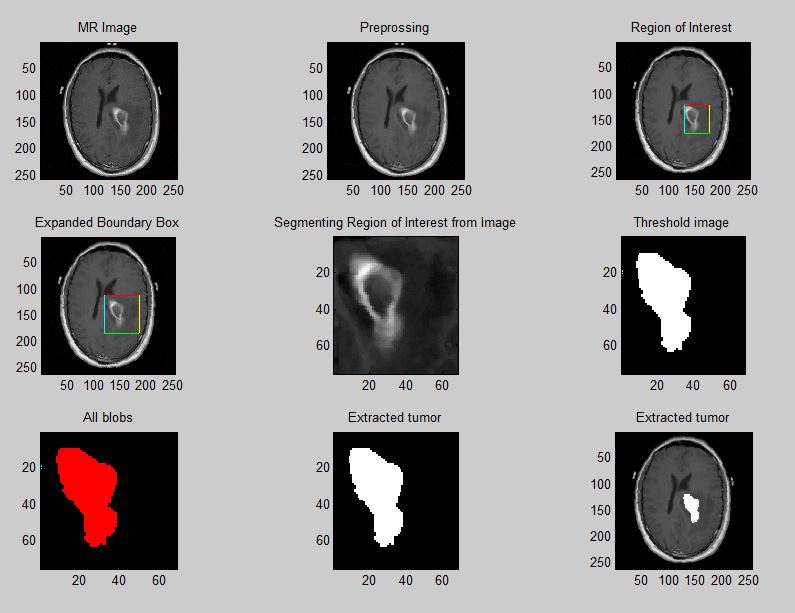


figure 1.1 Detection of tumor using image processing techniques.

* Security

Image processing has helped in development of efficient security methods. There has been a lot of new technologies used in mobile phones, laptops etc. like finger print unlock system, face recognition unlock system, eye iris recognition system. With the use of these techniques unlocking devices has become very simple and easy compared to typing and remembering passwords. These concepts of facial recognition, eye iris recognition has been extended to home security systems.

* Various social media websites also use some of the image processing techniques like facial recognition for example Facebook has the auto tag feature which suggests you to tag the person in the photo. Another application is Google image search. Searching for the visually similar images uses image processing concepts.

**Image processing libraries**

In this section we will introduce two image processing libraries scikit-image and pillow for python3 and show you how to use these libraries to carry out basic operations over images.

**Pillow**

We are going to cover the topics like how to install Pillow on different operating systems and what are the basic functions in the library.

**Installation**

* **Windows Installation**

Pillow can be installed using the pip command

pip install Pillow

* **macOS installation**

First download and install Homebrew. After installing Homebrew install the dependencies:

brew install libtiff libjpeg webp little-cms2

After installing dependencies use pip command to install Pillow

pip install Pillow

If you have both python2 and python3 installed, then to install Pillow for python3 use this command

python3 –m pip install Pillow

* **Linux installation**

Use pip command to install

pip install Pillow

**Tutorial**

In this tutorial we will learn about basic functions offered by the Pillow library. We will first learn about the functions in the image module.

**Reading an Image**

To read an image there is a function called open() in the image module. This function returns an image object. The image object contains information like image mode, image size, image format.

>>> from PIL import Image

>>> img = Image.open("image.png")

To display the image use the show() function

>>> img.show()

**Writing/Saving an Image**

To write or save an image use the save() function in the image module.

>>> img.save("temp.png")

**Crop an Image**

Use crop() function in the image module to crop the image. It takes two coordinates – the upper left corner and the bottom right corner of the rectangle. The function returns the cropped image.(figure 1.2)

>>> dim = (100,100,400,400)

>>> crop\_img = img.crop(dim)

>>> crop\_img.show()



figure 1.2

**Converting Image to grayscale**

Images can be converted from one mode to another using the convert function of the image module in Pillow library. To convert an from RGB mode to grayscale mode use the “L” mode(figure 1.3). There are various other modes available like “1” which is 1 bit pixel mode, “P” 8 bit pixel mode, “RGB” 3X8 bit pixel, “RGBA” 4X8 bit pixel.

>>> grayscale = img.convert("L")

>>> grayscale.show()



figure 1.3

**Geometrical Transformation**

You can resize the image, rotate it using the Pillow library in python.

To resize an image, use the resize() function which takes a tuple of new size as argument

>>> resize\_img = img.resize((200,200))

>>> resize\_img.show()

To rotate an image use the rotate() function. It takes degrees to be rotated(Counter clockwise) as argument.(figure 1.4)

>>> rotate\_img = img.rotate(90)

>>> rotate\_img.show()



figure 1.4

**Image Enhancement**

ImageEnhance is a separate module in the pillow library which offers functions like changing contrast, brightness, color balance, sharpness of an image.

First you need to import the ImageEnhance module

>>> from PIL import ImageEnhance

* Change brightness of an image:

>>> enhancer = ImageEnhance.Brightness(img)

>>> enhancer.enhance(2).show()

The enhance function takes a float as an argument which describes the factor of brightness. Less the factor less will be the brightness. Factor value equal 1 will give the original image as output. The return type of the enhance function is an image with the changed brightness.



figure 1.5 This figure shows the increase in the brightness of the image. Right image is the original picture and left picture is the enhanced image.

* Change Contrast of the image:

>>> enhancer = ImageEnhance.Contrast(img)

>>> enhancer.enhance(2).show()

Again the enhance function takes an float argument. Factor equal to 1 will give you the original image.



figure 1.6 This figure shows the change in the contrast of the image. Right image is the original picture and left picture is the enhanced image.

**Pixel Access Class**

Pillow library also has a PixelAccess module which has functions to manipulate the pixel values of an image. getpixel(), putpixel() are some of the functions in the PixelAccess library.

* getpixel(self, xy)

getpixel() function returns the value of the pixel at the (x,y) coordinate. It takes a tuple as argument.

>>> img.getpixel((100,100))

output

(132, 56, 40)

* putpixel(self, xy, color)

It changes the value of the pixel at the (x, y) coordinate given as an argument to the function. The value is changed to the color value in the argument. If the image has more than one band of colors then a tuple is passed as an argument to the function.

>>> img.putpixel((100,100),(0,0,0))

>>> img.getpixel((100,100))

output

(0, 0, 0)

**Color Spaces and Channels**

An image can be represented in many modes like RGB, CMYK, Y’UV etc. these different modes are called color spaces. The colors in the image are derived by the mixture of the colors in the color space. An image is 2D matrix of the pixel intensity value of the colors in the color space and generally intensity value of a color is represented by 8 bits. For example, in RGB color mode each pixel value of each color is represented by 8 bits and there are 3 colors therefore 24 bits will be used to determined the color of a pixel. The values of R, G and B will determine the actual color of the image. If the red value is 255(the value will be between 0 and 255 since 8 bits are used to represent the color) and both G and B are 0 then the color of that pixel will be red, if all are 255 then the color will be white and if all are 0 then the color will be black.

In case of color modes which uses mixture of more than one color to determine the color of the image multiple 2D matrices will be used to determine the pixel value of the image where each matrix represents one of the color in the color space, these are called color channels for example in RGB Red, Blue and Green are the color channels and each color is represented by different 2D matrix.

If you try to print the pixel value of an grayscale image you will only get one value but if you try to print the pixel value of a RGB image then you will get three values, this shows that RGB has three channels red, green, blue and grayscale images have only one value.

>>> from PIL import Image

>>> img = Image.open("image.png")

>>> img.getpixel((100,100))

output

(132, 56, 40)

>>> img.convert("L").getpixel((100,100))

output

76

The figure below shows the different color channels in a RGB image.



figure 1.7 Red, green, blue respectively.

**Image depth**

Image depth or the color depth is the number of bits used to represent a color of a pixel. The image depth determines the range of colors an image can have for example if the only 4 bits are used to represent the pixel value then the value will range from 0 to 15 whereas if 8 bits are used then the value will range from 0 to 255 giving a more specific color. The number of bits also determine the number of colors which can be used in an image for example 1 bit = 2 colors, 2 bit = 4colors, 8 bits = 256 colors etc.

**Scikit-image library**

In this section we are going to learn about another python library scikit-image for image processing.

**Installation**

* **MacOs**

Use pip command :

python3 -m pip install -U scikit-image

python3 -m pip install scipy

python3 -m pip install matplotlib

* **Linux**

sudo apt-get install python3-skimage

* **Windows**

pip3 install scikit-image

**Tutorial**

In this tutorial we learn about some inbuilt functions in the scikit-image library.

**Read an Image**

Image can be read using the imread() function in the io module of the library. It returns an ndarray. An ndarray in python is N dimensional array.

>>> from skimage import io

>>> img = io.imread("image.png")

>>> io.imshow("image.png")

>>> io.show()

**Write Image**

**Data module**

This module provides some standard test images which one can work on like a grayscale camera image, grayscale “text” image, coffee cup etc. these images can be used as great examples to demonstrate some of the algorithms in image processing.

**skimage.data.camera()**

returns an image array

>>> from skimage import data

>>> io.imshow(data.camera())

>>> io.show()

****

figure 1.8 image returned by the camera() function

**skimage.data.text()**

returns an image array

>>> from skimage import data

>>> io.imshow(data.text())

>>> io.show()

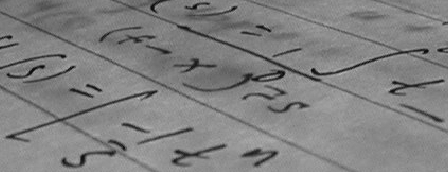


figure 1.9 image returned by text() function and it can used as an example for corner detection

**Color module**

This module of the library contains functions like converting image of one color space to another. One of the functions is:

**Convert RGB to Gray**

rgb2gray() function in the module can be used to convert a RGB image to a grayscale image. It takes the RGB image array as input and returns the grayscale image array.

>>> from skimage import io, color

>>> img = io.imread("image.png")

>>> gray = color.rgb2gray(img)

>>> io.imshow(gray)

>>> io.show()



figure 1.10

**Draw module**

Draw module has various functions to draw different shapes like circle, line , polygon etc.

**Circle**

circle() function of the draw module takes the centre coordinates and the radius as input and returns all the pixel coordinates which lie within the circle of the given coordinates and radius. After getting the pixels within the circle assign them the value 1 in the 2D matrix and all the other points make it 0. This would give you circle.

>>> import numpy as np

>>> from skimage import io, draw

>>> img = np.zeros((100, 100), dtype=np.uint8)

>>> x , y = draw.circle(50, 50, 10)

>>> img[x, y] = 1

>>> io.imshow(img)

>>> io.show()

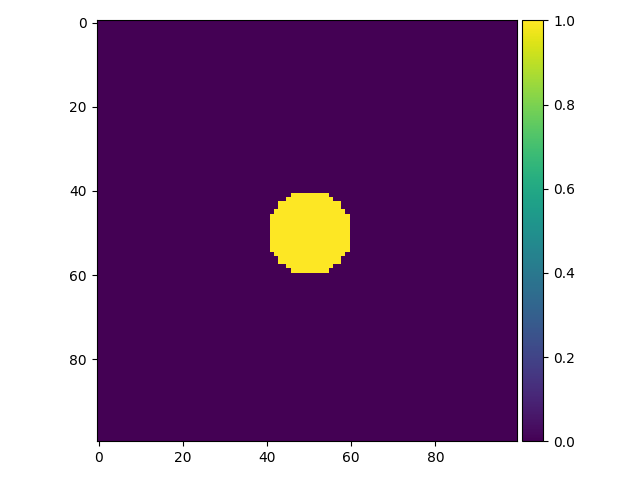


figure 1.11 circle of radius 10 and Centre (50, 50)

**Ellipse**

ellipse() function of the draw module can be used to get the coordinates of the pixels within the ellipse of given parameters. Then these pixels can be distinguished from others by increasing the pixel value.

>>> import numpy as np

>>> from skimage import io, draw

>>> img = np.zeros((100, 100), dtype=np.uint8)

>>> x , y = draw.ellipse(50, 50, 10, 20)

>>> img[x, y] = 1

>>> io.imshow(img)

>>> io.show()

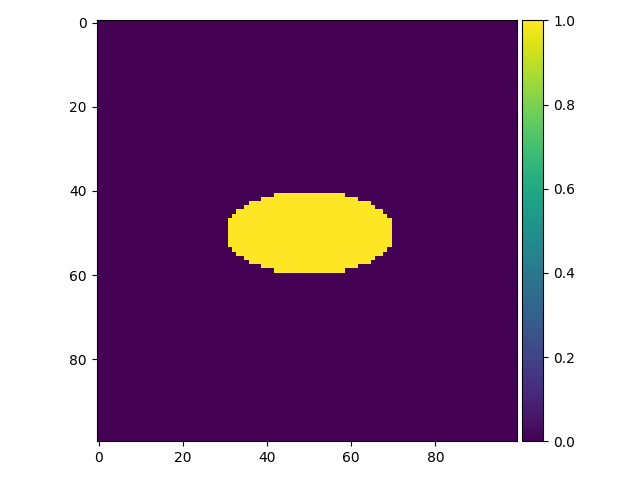


figure 1.12

**Polygon**

polygon() function takes the array of x and y coordinates of the vertices and returns the pixel coordinates which lie within the polygon.

>>> import numpy as np

>>> from skimage import io, draw

>>> img = np.zeros((100, 100), dtype=np.uint8)

>>> r = np.array([10, 25, 80, 50])

>>> c = np.array([10, 60, 40, 10])

>>> x, y = polygon(r, c)

>>> img[x, y] = 1

>>> io.imshow(img)

>>> io.show()

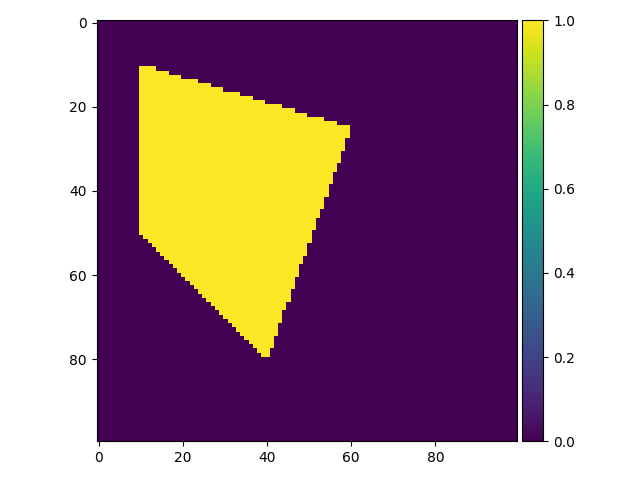


figure 1.13