Computer Vision

Teacher Activity: Edge Detection

Importing Packages

Packages are imported in following manner.

```
import package_name
```

In the next cell we have imported the following packages.

1. `cv2` . It is a part of openCV library of Python which helps in solving computer vision problems.
2. `numpy` . It adds support for arrays, along with a collection of mathematical functions to operate on these arrays.
3. `matplotlib` . It is a plotting library for python. `.pyplot` is a sub-package or set of functions available in matplotlib which we'll be using.

`np` , `plt` are all `aliases` for their corresponding packages. `Alias` are second name assigned to values or variables.

`%matplotlib inline` is a "magic function" renders plots

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Creating `edge_detection` function

The function takes `image` as the argument `.Canny()` function has been used on `image` . It accepts a gray scale image as input and it uses a min and max threshold values as additional arguments. Canny Edge Detection is used to detect the edges in an image.

`images` is a `list` of variables `image` , `edges_detected` .

`location` is the list created for subplot, it takes a 3-digit input to create subplots. Digits are mapped as nrows, ncols, index.

Applying the `for` loop for the elements in the `zip`

`.subplots` method provides a way to plot multiple plots on a single figure.

`.imshow` displays image specifying the display range as a two-element vector. The other parameter is a colormap we have assigned it a value "gray scale"

plt. show() starts an event loop, looks for all currently active figure objects, and opens one or more interactive windows that display your figure or figures.

In [2]:
```
def edge_detection(image):
    edges_detected = cv2.Canny(image , 100, 200)
    images = [image , edges_detected]
```

```
    location = [121, 122]
    for loc, edge_image in zip(location, images):
        plt.subplot(loc)
        plt.imshow(edge_image, cmap='gray')
        plt.show()
```

Creating a grayscale image

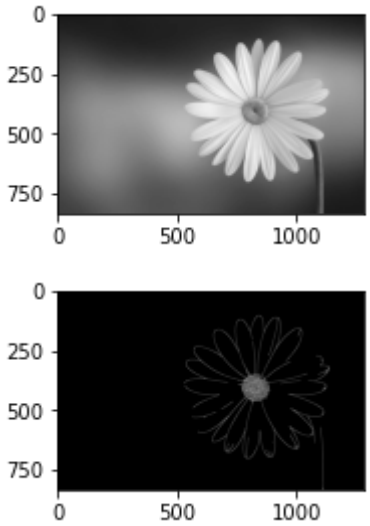
`cv2.imread()` method loads an image from the specified file and converts it based on second parameter i.e., `0` .

In [13]:
```
img = cv2.imread('1.jpeg', 0)
```

Calling function edge_detection

Function call using `img` as parameter

In [14]:
```
edge_detection(img)
```



In [ ]: