```
!pip install nilearn
```

```
Requirement already satisfied: nilearn in /usr/local/lib/python3.6/dist-packages (0.6.2)
Requirement already satisfied: sklearn in /usr/local/lib/python3.6/dist-packages (from nilearn) (0.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from nilearn) (0.14.1)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.6/dist-packages (from nilearn) (1.4.1)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.6/dist-packages (from nilearn) (1.17.5)
Requirement already satisfied: nibabel>=2.0.2 in /usr/local/lib/python3.6/dist-packages (from nilearn) (2.3.3)
Requirement already satisfied: scikit-learn>=0.19 in /usr/local/lib/python3.6/dist-packages (from nilearn) (0.
Requirement already satisfied: six>=1.3 in /usr/local/lib/python3.6/dist-packages (from nibabel>=2.0.2->nilear
Requirement already satisfied: bz2file in /usr/local/lib/python3.6/dist-packages (from nibabel>=2.0.2->nilearn
```

```
import numpy as np
from scipy import linalg
from matplotlib import pyplot as plt

from nilearn.input_data import NiftiMasker
```

```
%%time

from nilearn import datasets

n_subjects = 416

oasis_dataset = datasets.fetch_oasis_vbm(n_subjects=n_subjects)
```

```
CPU times: user 23 ms, sys: 4.2 ms, total: 27.2 ms
Wall time: 29.9 ms
/usr/local/lib/python3.6/dist-packages/nilearn/datasets/struct.py:316: UserWarning: Only 403 subjects are avai
  % n_subjects)
/usr/local/lib/python3.6/dist-packages/numpy/lib/npyio.py:2358: VisibleDeprecationWarning: Reading unicode str
  output = genfromtxt(fname, **kwargs)
```

```
cdr = oasis_dataset.ext_vars['cdr'].astype(float)
```

```
cdr
```

```
array([0. , 0. , 0.5, nan, nan, nan, nan, nan, 0. , 0. , nan, 0. , nan,
       0.5, 0.5, nan, 0. , 0. , 0. , 0.5, 0.5, 0.5, nan, 0. , 1. , nan,
       0. , 1. , 0. , 0. , 0. , 1. , nan, nan, 0.5, nan, 0.5, 0.5, nan,
       0. , nan, 0.5, nan, nan, nan, nan, 1. , 1. , nan, nan, 1. , 0. ,
       nan, 0.5, nan, 0. , nan, 0. , 0. , 1. , 0. , 0. , 0. , 0. , 0. ,
       1. , 0. , 0. , nan, nan, 0. , nan, nan, nan, 0.5, 0.5, 0. , 0. ,
       nan, nan, nan, nan, nan, 0.5, nan, 0. , nan, 0.5, nan, nan, nan,
       nan, nan, nan, 0. , nan, nan, 0. , 0. , nan, 0. , 0. , 0. , 0.5,
       0. , nan, nan, 0.5, nan, 0.5, 0.5, nan, nan, nan, nan, 0. , nan,
       nan, 0. , 1. , 0. , nan, 1. , 0. , 0. , nan, nan, 0.5, 0.5, nan,
       nan, 0. , nan, nan, nan, nan, nan, nan, 0.5, nan, 0.5, nan, nan,
       0.5, nan, nan, 0.5, 0. , 0.5, nan, nan, 0. , 0. , nan, nan, 0. ,
       0. , nan, 0.5, 0. , 0. , nan, nan, 1. , 1. , 0. , 0. , nan, nan,
       nan, nan, nan, 0. , 0. , nan, 0. , 0. , 0. , nan, 0. , 0. , 0.5,
       0. , 0. , 0. , nan, 0.5, nan, 0. , nan, nan, 0. , 0.5, nan, 0. ,
       0. , 1. , nan, 0.5, nan, 0. , 0. , nan, nan, nan, 0.5, 0. , nan,
       nan, 0. , 0.5, nan, 0.5, 0. , 0.5, 0. , nan, 0.5, nan, nan, nan,
       0. , 0. , 0. , nan, 0. , 0. , nan, 0. , 0.5, nan, nan, 0. , 0.5,
       1. , 0. , 0. , 0.5, 0.5, 0. , nan, nan, 1. , 0. , 0. , nan, nan,
       0. , nan, 0.5, 0.5, 0. , 0.5, 1. , 0. , 0. , nan, nan, nan, 0.5,
       0. , 0.5, 0. , nan, 0. , 0.5, nan, 0.5, 2. , nan, nan, 0.5, nan,
       nan, 0.5, 1. , 0. , nan, nan, nan, 0. , 0. , nan, 0. , nan, nan,
       0.5, 0. , nan, 0. , nan, 0.5, nan, 0. , 0. , 0.5, nan, 0. , 0. ,
       0. , nan, 0. , nan, nan, nan, nan, 2. , 0.5, nan, 0. , 0. , 0. ,
       0. , 0. , nan, nan, 0.5, 0. , 0. , 0. , 0. , nan, 0. , nan, 0. ,
       0. , 1. , 0.5, nan, nan, nan, 0. , nan, 0.5, 0. , 1. , nan, nan,
       nan, nan, nan, 1. , nan, 0.5, nan, nan, nan, nan, nan, 0. , 1. ,
       0.5, 0. , 0.5, nan, 0. , 1. , nan, nan, nan, nan, nan, 0.5, nan,
       nan, nan, nan, 0.5, nan, nan, nan, 0. , 0. , 1. , 1. , 0. , nan,
       1. , nan, 0.5, 0. , nan, nan, nan, 0. , nan, 0.5, 0.5, nan, 0. ,
       nan, 0. , 0. , 0.5, nan, 0. , nan, 0.5, 1. , 0.5, 0.5, 0. , 0. ])
```

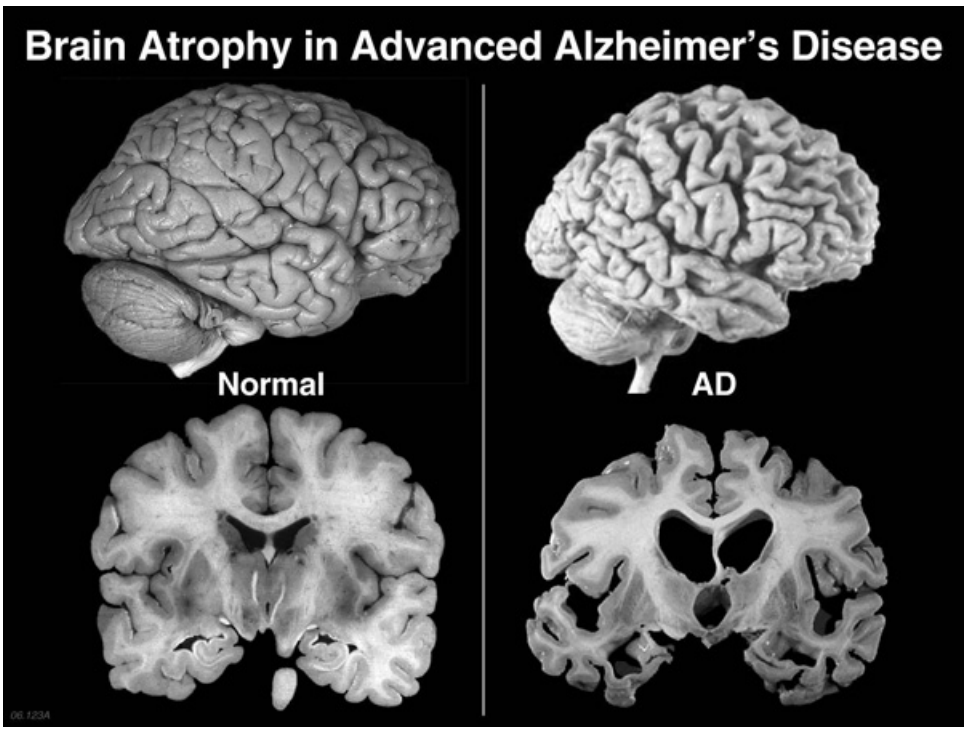
```
cdr_numpy_arr = np.array(cdr)
```

```
for i in range(len(cdr_numpy_arr)):
  if(np.isnan(cdr_numpy_arr[i])):
    cdr_numpy_arr[i] = 1

  elif(cdr_numpy_arr[i] > 0.0):
    cdr_numpy_arr[i] = 1
```

```
cdr_numpy_arr
```

```
array([0., 0., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 1., 0.,
       0., 0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
       1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0.,
       0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0.,
       1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0.,
       1., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1.,
       1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 1.,
       1., 0., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 0., 1., 1.,
       1., 1., 0., 1., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1.,
       0., 0., 0., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1.,
       1., 0., 1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 0., 1., 1., 0.,
       0., 1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1.,
       0., 1., 1., 1., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1., 1., 1., 1.,
       1., 1., 1., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0., 0., 1., 0.,
       1., 0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1.,
       1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1.,
       0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0.])
```

Normal vs Alzheimer Brain



Loading a normal and alzheimer affected brain MRI and storing it for further implementing image processing operations.

```
gray_matter_map_filenames = oasis_dataset.gray_matter_maps
```

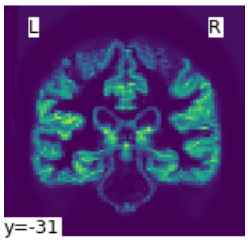
```
#without smoothing images
alz_img = gray_matter_map_filenames[4]
nor_img = gray_matter_map_filenames[8]

from nilearn import plotting

normal_plot = plotting.plot_img(nor_img, display_mode='y', cut_coords=[-31], output_file='normal_brain.png')
alzheimer_plot = plotting.plot_img(alz_img, display_mode='y', cut_coords=[-24], output_file='alzheimer_brain.png')
```

```
/usr/local/lib/python3.6/dist-packages/nilearn/plotting/displays.py:1596: MatplotlibDeprecationWarning: Adding
  fraction * (x1 - x0), y1 - y0])
```

Displaying the Normal and Alzheimer affected Brain MRI

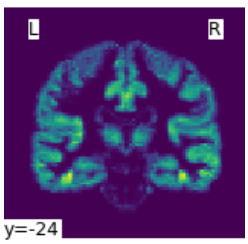
```
from IPython.display import HTML, Image, display
print("----------------")
print("NORMAL BRAIN MRI")
print("----------------")
Image('normal_brain.png', width=150, height=150)
```

```
----------------
NORMAL BRAIN MRI
----------------
```




```
print("-------------------")
print("ALZHEIMER BRAIN MRI")
print("-------------------")
Image('alzheimer_brain.png', width=150, height=150)
```

```
-------------------
ALZHEIMER BRAIN MRI
-------------------
```



Implementing Image Smoothing and Edge Detection from Scratch

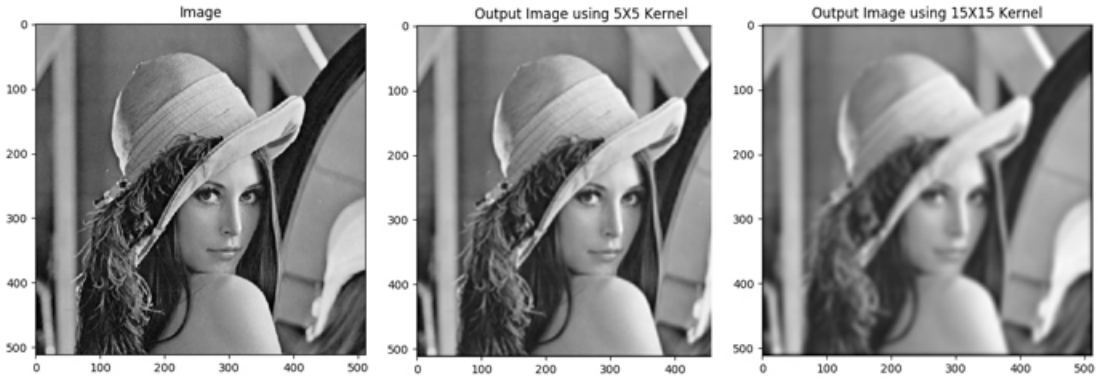
- Image Smoothing - Gaussian Filter
- Image Edge Detection - Sobel Edge Detection

Code for Convolution Operation (common to both smoothing and edge detection)

1 cell hidden

Gaussian Smoothing Function

$$f\left(x\right)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\left(x-\mu\right)^2/\left(2\sigma^2\right)}$$

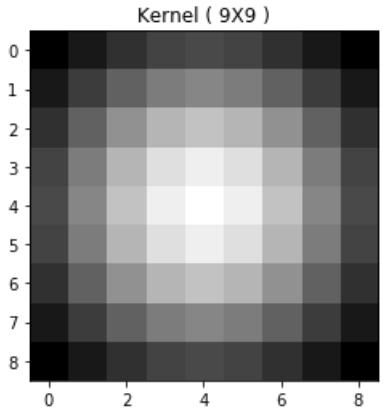



```
import math

def dnorm(x, mu, sd):
    # Gaussian filter formula below
    return 1 / (np.sqrt(2 * np.pi) * sd) * np.e ** (-np.power((x - mu) / sd, 2) / 2)

def gaussian_kernel(size, sigma=1, verbose=False):
    kernel_1D = np.linspace(-(size // 2), size // 2, size)
    for i in range(size):
        kernel_1D[i] = dnorm(kernel_1D[i], 0, sigma)
    kernel_2D = np.outer(kernel_1D.T, kernel_1D.T)

    kernel_2D *= 1.0 / kernel_2D.max()

    if verbose:
        plt.imshow(kernel_2D, interpolation='none', cmap='gray')
        plt.title("Kernel ( {}X{} )".format(size, size))
        plt.show()

    return kernel_2D


def gaussian_blur(image, kernel_size, verbose=False):
    kernel = gaussian_kernel(kernel_size, sigma=math.sqrt(kernel_size), verbose=verbose)
    return convolution(image, kernel, average=True, verbose=verbose)
```

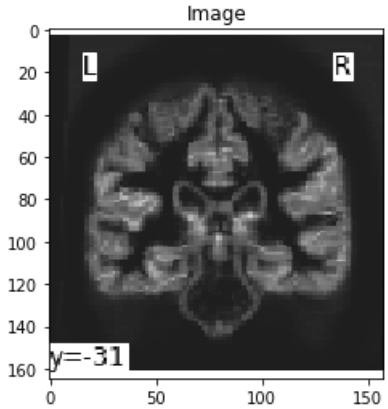
Implementing Gaussian Smoothing Operation On Our Brain MRI Images

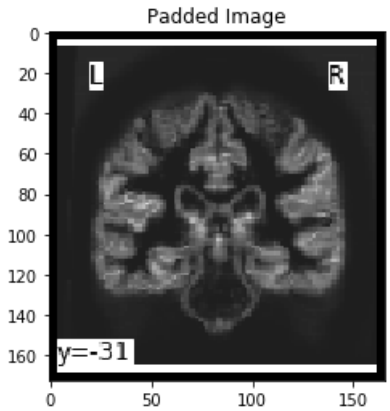
```
print("----------------")
print("NORMAL BRAIN MRI")
print("----------------")
print()

image = cv2.imread('normal_brain.png')
image = gaussian_blur(image, 9, verbose=True)
```

```
----------------
NORMAL BRAIN MRI
----------------
```

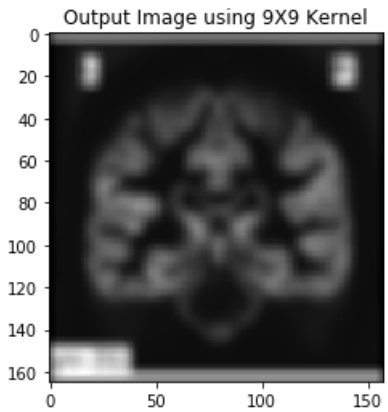



```
Found 3 Channels : (165, 158, 3)
Converted to Gray Channel. Size : (165, 158)
Kernel Shape : (9, 9)
```





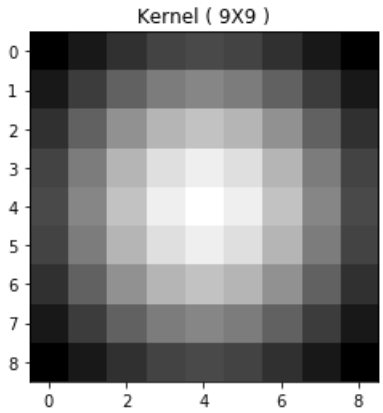

```
Output Image size : (165, 158)
```



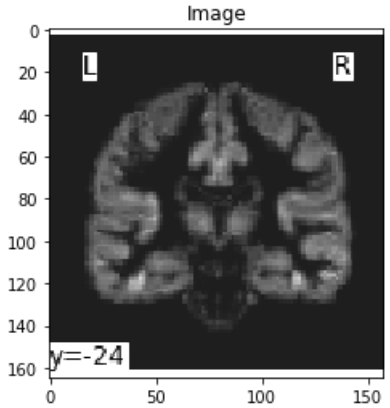

```
print("--------------------")
print("ALZHEIMER BRAIN MRI")
print("--------------------")
print()

image = cv2.imread('alzheimer_brain.png')
```

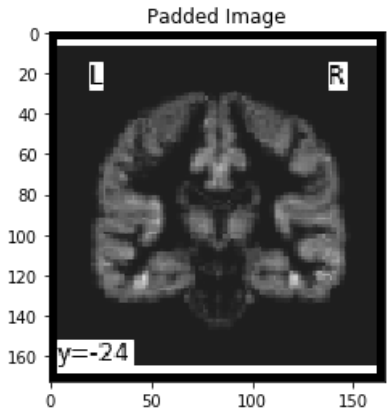
```
image = gaussian_blur(image, 9, verbose=True)
```

```
--------------------
ALZHEIMER BRAIN MRI
--------------------
```

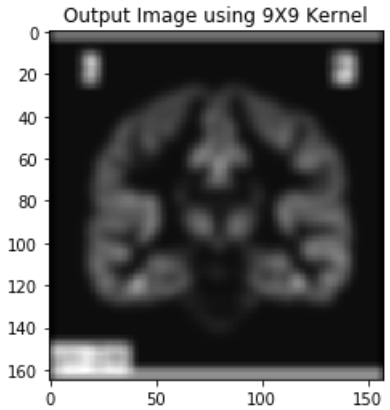



```
Found 3 Channels : (165, 158, 3)
Converted to Gray Channel. Size : (165, 158)
Kernel Shape : (9, 9)
```






```
Output Image size : (165, 158)
```



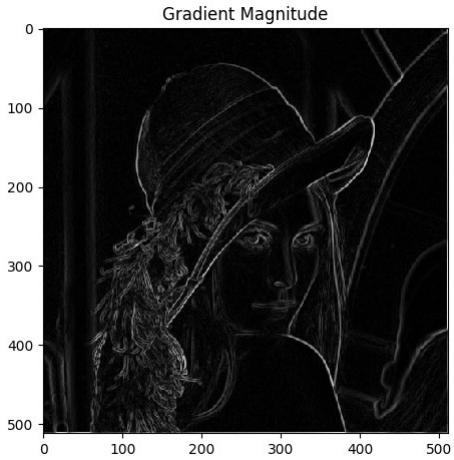
Sobel Edge Detection Function

-1	0	1
-2	0	2
-1	0	1

Vertical

1	2	1
0	0	0
-1	-2	-1

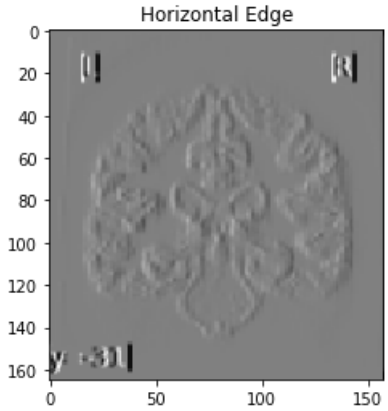
Horizontal



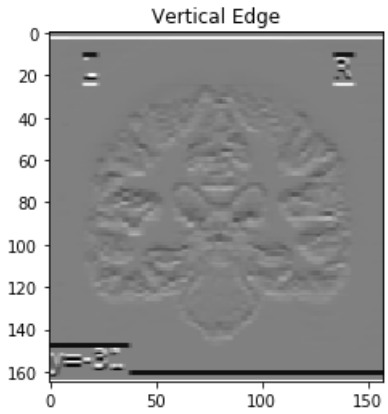

```
def sobel_edge_detection(image, filter, verbose=False):
    new_image_x = convolution(image, filter, verbose)

    if verbose:
        plt.imshow(new_image_x, cmap='gray')
        plt.title("Horizontal Edge")
        plt.show()

    new_image_y = convolution(image, np.flip(filter.T, axis=0), verbose)

    if verbose:
        plt.imshow(new_image_y, cmap='gray')
        plt.title("Vertical Edge")
        plt.show()

    return None
```

```
print("--------------------")
print("NORMAL BRAIN MRI")
print("--------------------")
print()

# SOBEL FILTER
filter = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

image = cv2.imread('normal_brain.png')
sobel_edge_detection(image, filter, verbose=True)
```

```
------------------
NORMAL BRAIN MRI
------------------

Found 3 Channels : (165, 158, 3)
Converted to Gray Channel. Size : (165, 158)
Kernel Shape : (3, 3)
Output Image size : (165, 158)
```

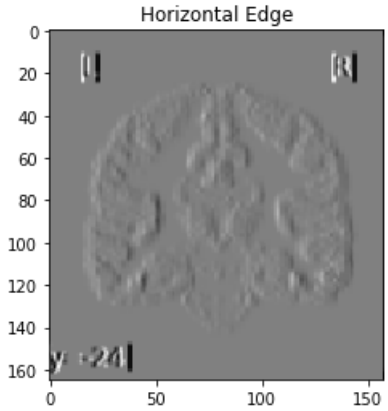



```
Found 3 Channels : (165, 158, 3)
Converted to Gray Channel. Size : (165, 158)
Kernel Shape : (3, 3)
Output Image size : (165, 158)
```

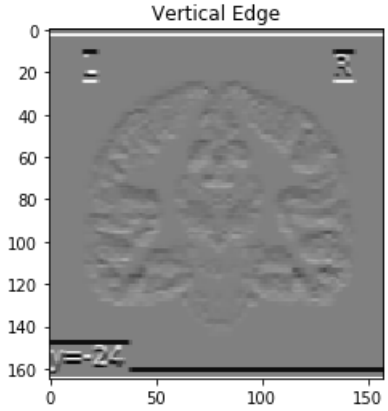



```
print("--------------------")
print("ALZHEIMER BRAIN MRI")
print("--------------------")
print()

image = cv2.imread('alzheimer_brain.png')
# image = gaussian_blur(image, 9, verbose=True)
sobel_edge_detection(image, filter, verbose=True)
```

```
------------------
ALZHEIMER BRAIN MRI
------------------

Found 3 Channels : (165, 158, 3)
Converted to Gray Channel. Size : (165, 158)
Kernel Shape : (3, 3)
Output Image size : (165, 158)
```




```
Found 3 Channels : (165, 158, 3)
Converted to Gray Channel. Size : (165, 158)
Kernel Shape : (3, 3)
Output Image size : (165, 158)
```



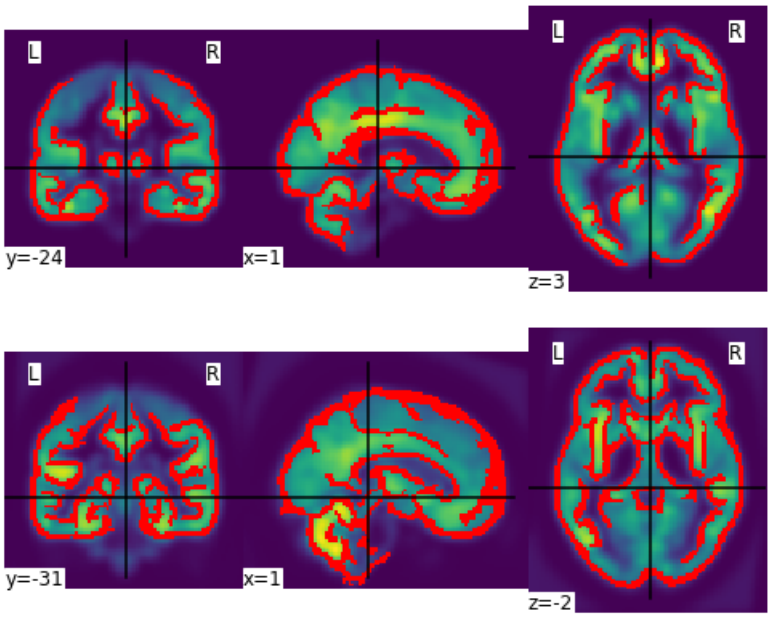
Using Inbuilt Library Functions For Various Plots From nilearn

```
# Default title text
from nilearn import plotting
from nilearn.image import smooth_img

# the two sample images
alzheimer_img = smooth_img(gray_matter_map_filenames[4], fwhm=8)
normal_img = smooth_img(gray_matter_map_filenames[8], fwhm=8)
```

```
# display.add_edges(img)

print("Plotting Function - Normal Image Plot")
print("-------------------------------------")

print("First image - Alzheimer Brain MRI")
plot1 = plotting.plot_img(alzheimer_img)
plot1.add_edges(alzheimer_img)

print("Second image - Normal Brain MRI")
plot2 = plotting.plot_img(normal_img)
plot2.add_edges(normal_img)
```

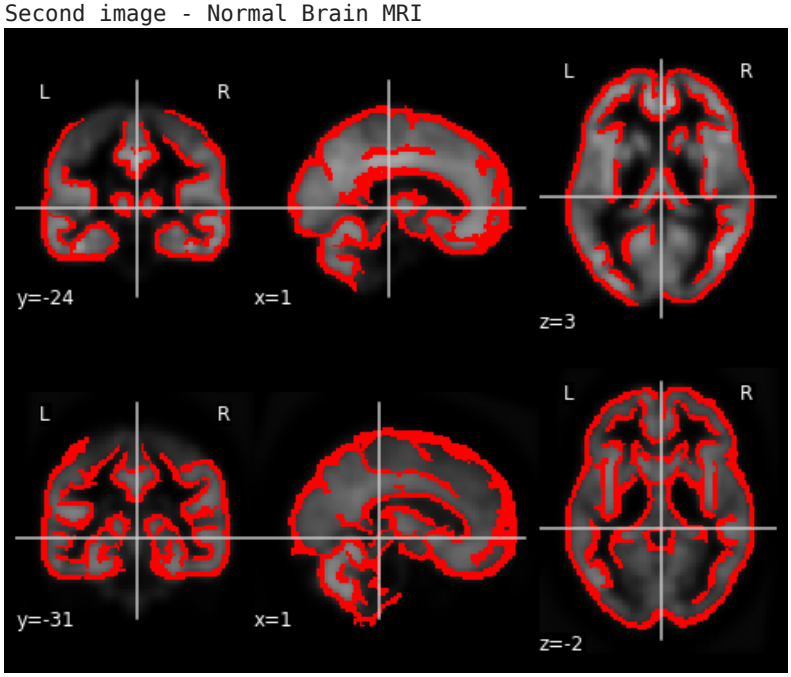
```
print()
print("Observe the greater hollowness in Alzheimer Brain as compared with the Normal Brain.")
print()
print("The brain boundaries are clearly defined in the second image (normal) but not in the first one.")
print("Alzheimer's disease destroys neurons and their connections in parts of the brain involved in memory thus cre
print()
```

```
Plotting Function - Normal Image Plot
-------------------------------------
First image - Alzheimer Brain MRI
Second image - Normal Brain MRI

Observe the greater hollowness in Alzheimer Brain as compared with the Normal Brain.

The brain boundaries are clearly defined in the second image (normal) but not in the first one.
Alzheimer's disease destroys neurons and their connections in parts of the brain involved in memory thus creat
```



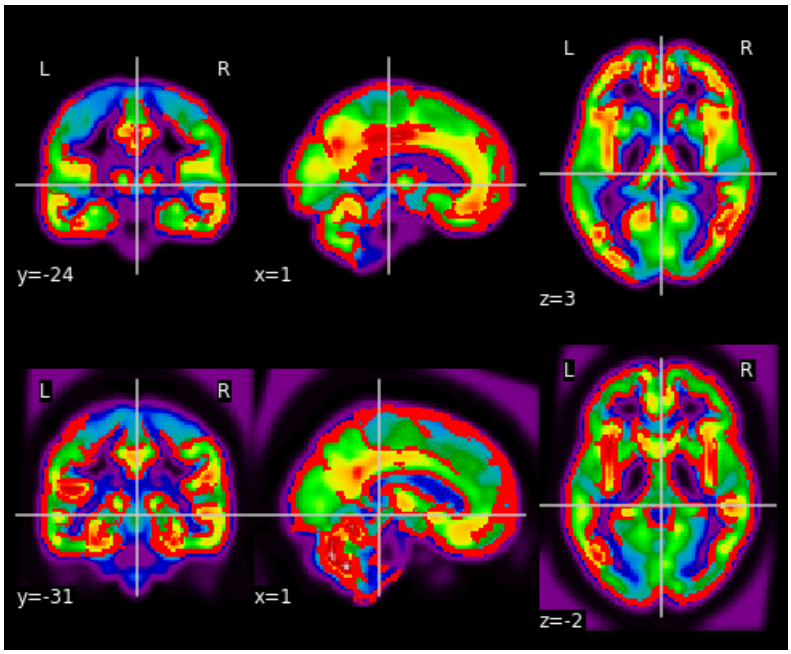

```
print("Plotting Function - Anatomy (Grayscale) Image Plot")
print("-------------------------------------")

print("First image - Alzheimer Brain MRI")
plot1 = plotting.plot_anat(alzheimer_img)
plot1.add_edges(alzheimer_img)

print("Second image - Normal Brain MRI")
plot2 = plotting.plot_anat(normal_img)
plot2.add_edges(normal_img)
```

```
Plotting Function - Anatomy (Grayscale) Image Plot
-------------------------------------
First image - Alzheimer Brain MRI
Second image - Normal Brain MRI
```




```
print("Plotting Function - EPI Plot")
print("-------------------------------------")

print("First image - Alzheimer Brain MRI")
plot1 = plotting.plot_epi(alzheimer_img)
plot1.add_edges(alzheimer_img)

print("Second image - Normal Brain MRI")
plot2 = plotting.plot_epi(normal_img)
plot2.add_edges(normal_img)

print()
print("Carefully observe the width of the purple colored section of the brain.")
print("It is wider in Alzheimer affected Brain MRI")
print()
```

```
Plotting Function - EPI Plot
------------------------------------
First image - Alzheimer Brain MRI
Second image - Normal Brain MRI

Carefully observe the width of the purple colored section of the brain.
It is wider in Alzheimer affected Brain MRI
```



/