__Laplacian Filter__

$$ \begin{bmatrix} 0 & 1 & 0\\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} $$

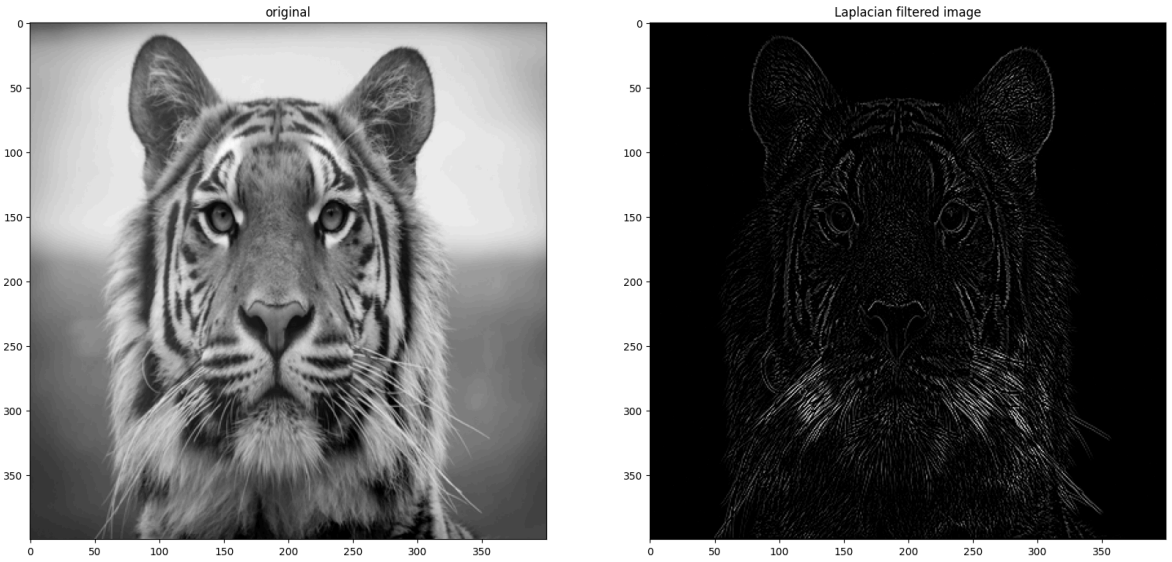
Applying Laplacian filter to an image get a new image that highlights edges and other discontinuties

In [1]:
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

In [2]:
```
def Laplacian_filter(input_image):
  img = input_image.resize((400,400), Image.Resampling.LANCZOS)

  fig = plt.figure()
  fig.set_figheight(20)
  fig.set_figwidth(20)

  #plotting original image
  fig.add_subplot(1,2,1)
  plt.imshow(img, cmap='gray')
  plt.title('original')


  # convert to numpy array
  numpy_image = np.array(img)
  # array for padding
  array_b = np.zeros((402,402))

  # to pad initial array with zeros in all side
  array_b[1:401,1:401] = numpy_image

  #defining filter
  filter_array = np.array([[0,1,0],
                           [1,-4,1],
                           [0,1,0]])

  #creating empty list
  lst = []

  for i in range(400):
    for j in range(400):
      #extracting part of array equal to filter size
      array_c = array_b[i:(3+i),j:(3+j)]

     #applying filter
      array_mul = np.multiply(filter_array,array_c)
      array_sum = np.sum(array_mul)

      # putting calculated value in list
      lst.append(array_sum)

  # resizing lst to shape of original array
  final_array = np.resize(lst,(400,400))

  final_image = Image.fromarray(final_array)
```

```
  final_image= final_image.convert("L")

    #plotting filtered image
  fig.add_subplot(1,2,2)
  plt.imshow(final_image, cmap='gray')
  plt.title('Laplacian filtered image')
```

In [3]:
```
# reading image and converting to gray scale
img = Image.open('../images/tiger.jpg').convert('L')
# Calling function
Laplacian_filter(img)
```



__Variants of Laplace__

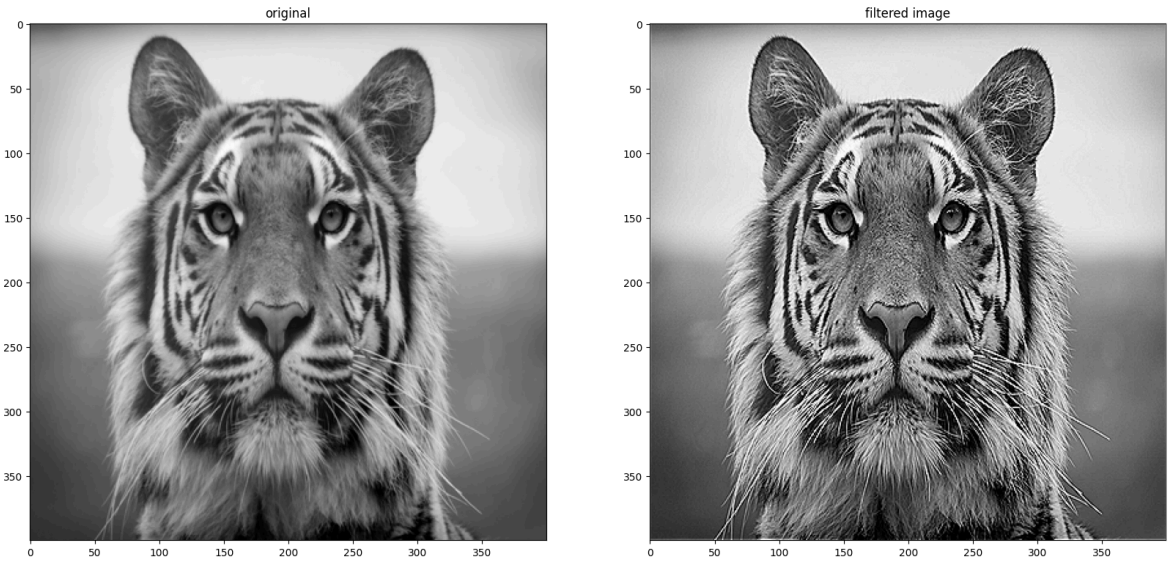
subtract laplace from original image to generate our final sharpened enhanced image

$$ \begin{bmatrix} 0 & -1 & 0\\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} $$

In [4]:
```
def Sharpened_Laplacian_filter(input_image):
  img = input_image.resize((400,400), Image.Resampling.LANCZOS)

  fig = plt.figure()
  fig.set_figheight(20)
  fig.set_figwidth(20)

  #plotting original image
  fig.add_subplot(1,2,1)
  plt.imshow(img, cmap='gray')
  plt.title('original')

  # convert to numpy array
  numpy_image = np.array(img)
  # array for padding
  array_b = np.zeros((402,402))

  # to pad initial array with zeros in all side
  array_b[1:401,1:401] = numpy_image

  #defining filter
  filter_array = np.array([[0,-1,0],
                           [-1,5,-1],
                           [0,-1,0]])
```

```
  #creating empty list
  lst = []

  for i in range(400):
    for j in range(400):
      #extracting part of array equal to filter size
      array_c = array_b[i:(3+i),j:(3+j)]

     #applying filter
      array_mul = np.multiply(filter_array,array_c)
      array_sum = np.sum(array_mul)

      # putting calculated value in list
      lst.append(array_sum)

  # resizing lst to shape of original array
  final_array = np.resize(lst,(400,400))

  final_image = Image.fromarray(final_array)
  final_image= final_image.convert("L")

    #plotting filtered image
  fig.add_subplot(1,2,2)
  plt.imshow(final_image, cmap='gray')
  plt.title('filtered image')
```

In [5]:
```
# reading image and converting to gray scale
img = Image.open('../images/tiger.jpg').convert('L')
# Calling function
Sharpened_Laplacian_filter(img)
```



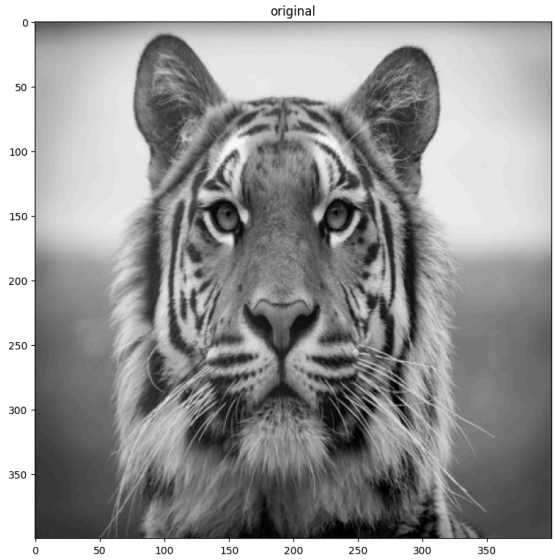
Variant 1

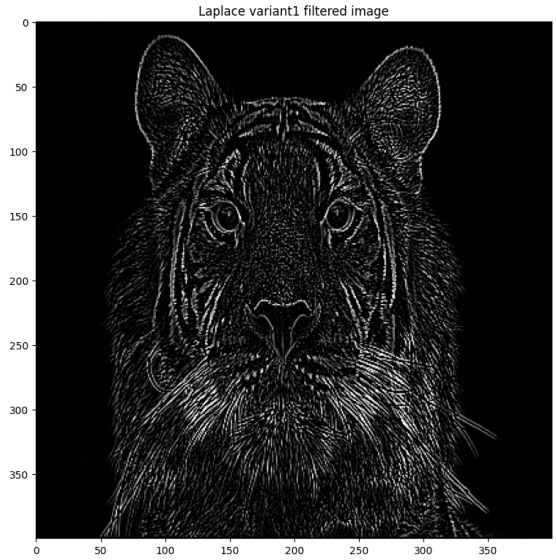
$$ \begin{bmatrix} 1 & 1 & 1\\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} $$

In [6]:
```
def Laplacian_Variant1(input_image):
  img = input_image.resize((400,400), Image.Resampling.LANCZOS)

  fig = plt.figure()
  fig.set_figheight(20)
  fig.set_figwidth(20)
```

```
#plotting original image
fig.add_subplot(1,2,1)
plt.imshow(img, cmap='gray')
plt.title('original')

# convert to numpy array
numpy_image = np.array(img)
# array for padding
array_b = np.zeros((402,402))

# to pad initial array with zeros in all side
array_b[1:401,1:401] = numpy_image

#defining filter
filter_array = np.array([[1,1,1],
                         [1,-8,1],
                         [1,1,1]])

#creating empty list
lst = []

for i in range(400):
  for j in range(400):
    #extracting part of array equal to filter size
    array_c = array_b[i:(3+i),j:(3+j)]

   #applying filter
    array_mul = np.multiply(filter_array,array_c)
    array_sum = np.sum(array_mul)

    # putting calculated value in list
    lst.append(array_sum)

# resizing lst to shape of original array
final_array = np.resize(lst,(400,400))

final_image = Image.fromarray(final_array)
final_image= final_image.convert("L")

  #plotting filtered image
fig.add_subplot(1,2,2)
plt.imshow(final_image, cmap='gray')
plt.title('Laplace variant1 filtered image')
```

In [7]:
```
Laplacian_Variant1(img)
```





Variant 2

$$ \begin{bmatrix} -1 & -1 & -1\\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} $$

In [8]:

```
def Laplacian_Variant2(input_image):
  img = input_image.resize((400,400), Image.Resampling.LANCZOS)

  fig = plt.figure()
  fig.set_figheight(20)
  fig.set_figwidth(20)

  #plotting original image
  fig.add_subplot(1,2,1)
  plt.imshow(img, cmap='gray')
  plt.title('original')

  # convert to numpy array
  numpy_image = np.array(img)
  # array for padding
  array_b = np.zeros((402,402))

  # to pad initial array with zeros in all side
  array_b[1:401,1:401] = numpy_image

  #defining filter
  filter_array = np.array([[-1,-1,-1],
                           [-1,9,-1],
                           [-1,-1,-1]])

  #creating empty list
  lst = []

  for i in range(400):
    for j in range(400):
      #extracting part of array equal to filter size
      array_c = array_b[i:(3+i),j:(3+j)]

     #applying filter
      array_mul = np.multiply(filter_array,array_c)
      array_sum = np.sum(array_mul)

      # putting calculated value in list
      lst.append(array_sum)
```

```
# resizing lst to shape of original array
final_array = np.resize(lst,(400,400))

final_image = Image.fromarray(final_array)
final_image= final_image.convert("L")

  #plotting filtered image
fig.add_subplot(1,2,2)
plt.imshow(final_image, cmap='gray')
plt.title('Laplace variant2 filtered image')
```

In [9]:
```
Laplacian_Variant2(img)
```