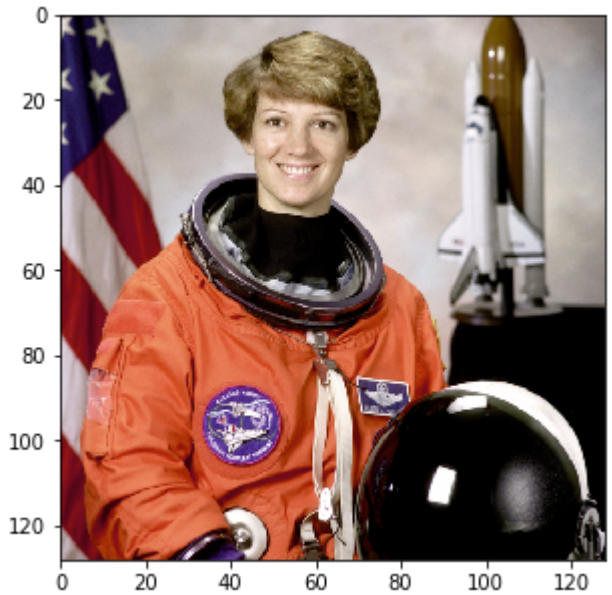
Importacion de librerias ¶

In [207]:

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt #Necesaira para mostar las imagenes
import numpy as np #Necesaria para las operacionaes matemáticas
from skimage import data, color, feature, io, filters, morphology #Librería que perm
from skimage.morphology import square, disk
from scipy import ndimage as ndi
import warnings # Esta es la clase base de todas las clases de categorías de adverte
warnings.filterwarnings('ignore')
img1=data.astronaut() # cargamos la imagen astrnaut
fig, (ax1) = plt.subplots(ncols=1, figsize=(15,5), sharex=True, sharey = True)
ax1.imshow(img1, extent=(0, 128, 128, 0))
```

Out[207]:

```
<matplotlib.image.AxesImage at 0x1a4104e610>
```



Generar Ruido

In [208]:

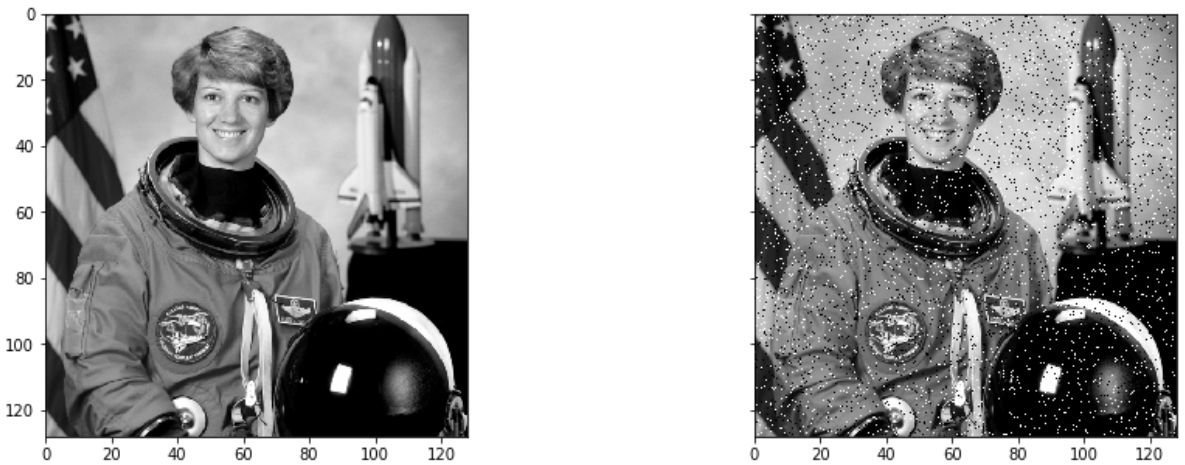
```
def Ruido(img, amount=0.5, s_vs_p = 0.5):
    row,col = img.shape # devuelve una tupla con el tamaño del array
    out = np.copy(img) #realizamos una copia de la imagen para poder trabajar
    #Sal
    num_sal = np.ceil(amount * img.size * s_vs_p) #Devuelve el techo de la entra
    coords = [np.random.randint(0, i - 1, int(num_sal))
        for i in img.shape]
    out[coords] = 1
     #Pimienta
    num_pim = np.ceil(amount* img.size * (1. - s_vs_p))
    coords = [np.random.randint(0, i - 1, int(num_pim))
        for i in img.shape]
    out[coords] = 0
    return out
print("¿ingresar % de ruido de 1% al 100% ?")
ruido = float(int(input())/100) # asignamos el porcentaje de ruido que deseamos
img = color.rgb2gray(data.astronaut()) # trasformamos la imagen que estamos trab
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(15,5), sharex=True, sharey = Tr
ax1.imshow(img, extent=(0, 128, 128, 0), interpolation='nearest', cmap="gray")
ax2.imshow(Ruido(img, ruido), extent=(0, 128, 128, 0), interpolation='nearest',

```

```
¿ingresar % de ruido de 1% al 100% ?
7
```

Out[208]:

```
<matplotlib.image.AxesImage at 0x1a3b74be90>
```



Filtrar Ruido

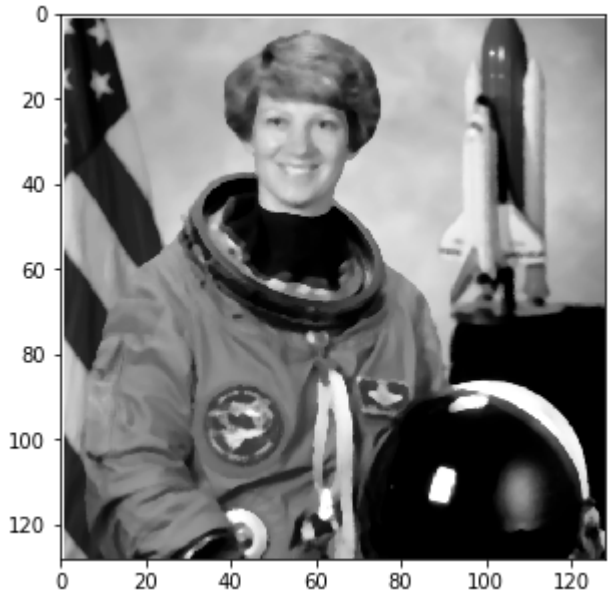
In [209]:

```
ng,n=fil):
n_columns = img.shape #los c=valores de filas y columans es igual al valor de la tupl
= np.copy(img) # copia de la imagen
range(n_rows):
j in range(n_columns):
img_copy[i, j] = np.median(img[i-int((n/2)):i+int((n/2)), j-int((n/2)):j+int((n/2))])
ng_copy #retornamos la imagen resultante
esar % de filtrado de 1% al 100% ?")
int(input()) # utilizar el 7% para una correcto filtrado
uido(img, ruido) # la imagen que se va a elimianar el ruido
plt.subplots(ncols=1, figsize=(15,5), sharex=True, sharey = True)
llro(img_modf, fil), extent=(0, 128, 128, 0), interpolation='nearest', cmap="gray")
```

```
¿ingresar % de filtrado de 1% al 100% ?
7
```

Out[209]:

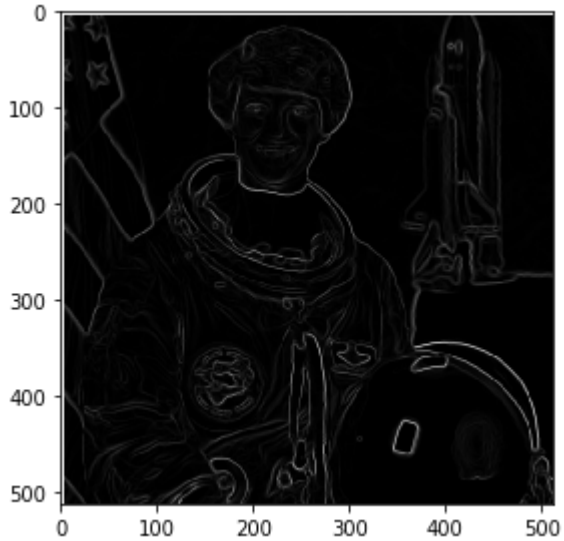
```
<matplotlib.image.AxesImage at 0x1a3b9ec510>
```



Obtencion de Bordes

In [210]:

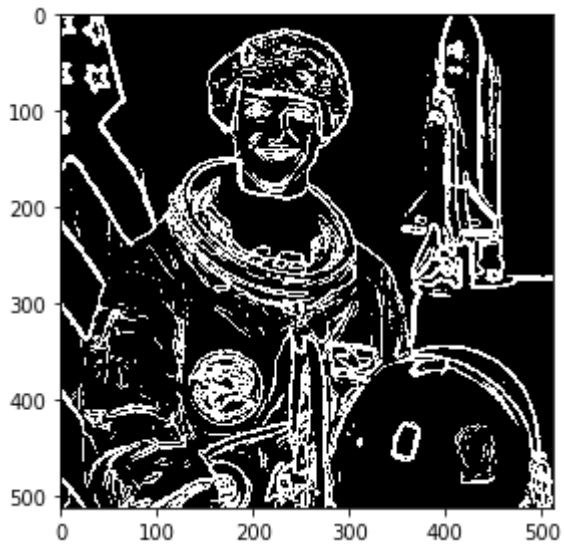
```
image= filtro(img_modf)# asignamos la imagen a obtener el borde
imsobel = filters.sobel(image) # filtro de sobel
io.imshow(imsobel)
io.show()
```




```
Con la obtencionde bordes mediante el metodo de sobel se obtuvo el resultado espero por lo cual no se procedio
a realizar mas filtrado de datos, porque al hacerlo no se obtine el resultado esperado, a continuacion se puede
observar el resultado trasnformado a binario
```

In [193]:

```
threshold = 0.05
binary_custom = imsobel > threshold
io.imshow(binary_custom)
io.show()
```



In [205]:

```
im0= morphology.remove_small_objects(binary_otsu,9)
im1=morphology.dilation(image=im0, selem=disk(2))
im2=morphology.erosion(image=im1, selem=disk(1))
im3= ndi.binary_fill_holes(im2)
im4=morphology.erosion(image=im3, selem=disk(5))
im5=morphology.dilation(image=im4, selem=disk(4))

fig, axes = plt.subplots(2, 4, figsize=(16, 16))
axes[0,0].imshow(binary_otsu, cmap='gray', aspect='equal')
axes[0,0].set_title('imagen binaria')
axes[0,1].imshow(im0, cmap='gray', aspect='equal')
axes[0,1].set_title('imagen dilatacion')
axes[0,2].imshow(im1, cmap='gray', aspect='equal')
axes[0,2].set_title('imagen erosion')
axes[0,3].imshow(im2, cmap='gray', aspect='equal')
axes[0,3].set_title('imagen binaria')
axes[1,0].imshow(im3, cmap='gray', aspect='equal')
axes[1,0].set_title('imagen binaria')
axes[1,1].imshow(im4, cmap='gray', aspect='equal')
axes[1,1].set_title('imagen erosion')
axes[1,2].imshow(im5, cmap='gray', aspect='equal')
axes[1,2].set_title('imagen dilatacion')
axes[1,3].imshow(image, cmap='gray', aspect='equal')
axes[1,3].set_title('imagen original')
```

Out[205]:

```
Text(0.5, 1.0, 'imagen original')
```