T3-Manuel_Pasieka

January 6, 2019

In [1]:
```
import skimage
from skimage import data, io
from matplotlib import pyplot

fig_size = (10,10)
```

In [2]:
```
# Load image, convert it to grayscale, normalize it
image = skimage.color.rgb2gray(data.astronaut())
image = image/image.max()
fig = plt.figure(figsize= fig_size) #create an empty figure to plot into with 20x20 si
io.imshow(image)
plt.show()
```

1 Ruido

Crear una función que modele la adición, de forma aleatoria, de artefactos impulsivos. El resultado tendrá que apreciarse en la visualización de la imagen, que deberá estar afectada por ruido de tipo ńsal y pimientaż. Como se ha visto en la asignatura, estos artefactos toman valores de intensidad máximos o mínimos y afectan, aleatoriamente, a los píxeles de la imagen.

- La función a implementar debe aceptar la imagen original y devolver la imagen afectada por el ruido. Además, sería deseable que aceptara un argumento adicional para indicar el porcentaje de píxeles que se verán afectados por estos artefactos.

```
In [3]: def sp_noise(image, p, seed=42):
            np.random.seed(seed)
```

```
    N = int(image.size * max(min(p, 1.0), 0.0) * 0.5)

    noisy_image = np.copy(image)
    # Add salt and pepper
    noisy_image.flat[np.random.randint(noisy_image.size, size=N)] = 1.0
    noisy_image.flat[np.random.randint(noisy_image.size, size=N)] = 0.0

    return noisy_image
```

In [4]:
```
# Add 1% of salt and pepper noise
sp_image = sp_noise(image, 0.01)
fig = plt.figure(figsize=fig_size)
io.imshow(sp_image)
plt.show()
```

2 Filtrar ruido

A partir de la imagen ruidosa, buscaremos eliminar los artefactos impulsivos para el posterior tratamiento de la imagen. Para ello, se deberá aplicar un filtro adecuado para este tipo de ruido.

In [5]:
```
# Apply a median filter to get rid of s&p
median_image = skimage.filters.median(sp_image)
```

```
/Users/manuel.pasieka/anaconda3/envs/py3/lib/python3.6/site-packages/skimage/util/dtype.py:130
  .format(dtypeobj_in, dtypeobj_out))
```

In [6]:
```
fig = plt.figure(figsize=fig_size)
io.imshow(median_image)
plt.show()
```

3 Caculate outlines

Por último, a partir de la imagen obtenida en la etapa anterior, se busca identificar las siluetas de las estructuras en ella presentes. A este fin, se debe identificar y razonar qué tipo de operador corresponde aplicar.

```
In [7]: # Apply a sobel filter to detect edge intensity
        sobel_image = skimage.filters.sobel(median_image)
        fig = plt.figure(figsize=fig_size)
        io.imshow(sobel_image)
        plt.show()
```