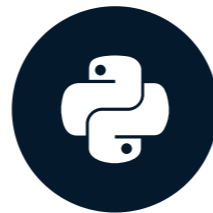


Intensity Values

BIOMEDICAL IMAGE ANALYSIS IN PYTHON



Stephen Bailey
Instructor

Pixels and voxels

- **Pixels** are 2D picture elements
- **Voxels** are 3D volume elements
- Two properties: intensity and location



Data types and image size

Array's data type controls
range of possible intensities

Type	Range	No. Val.
uint8	0, 255	256
int8	- 128, 127	256
uint16	0, 2^{16}	2^{16}
int16	-2^{15} , 2^{15}	2^{16}
float16	$\sim -2^{16}$, $\sim 2^{16}$	$\gg 2^{16}$

```
import imageio

im=imageio.imread('foot-xray.jpg')

im.dtype
dtype('uint8')
im.size
```

153600

```
im_int64 = im.astype(np.uint64)
im_int64.size
```

1228800

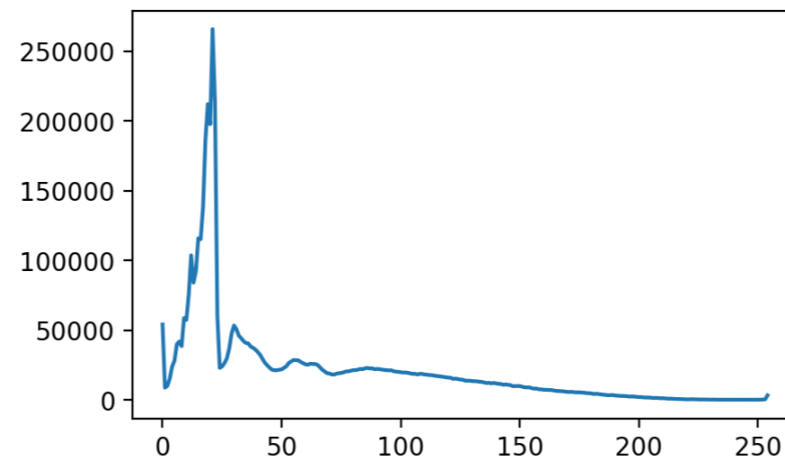
Histograms

- **Histograms:** count number of pixels at each intensity value.
- Implemented in `scipy.ndimage`
 - higher-dimensional arrays
 - masked data
- Advanced techniques and functionality in `scikit-image`.

```
plt.plot(hist)  
plt.show()
```

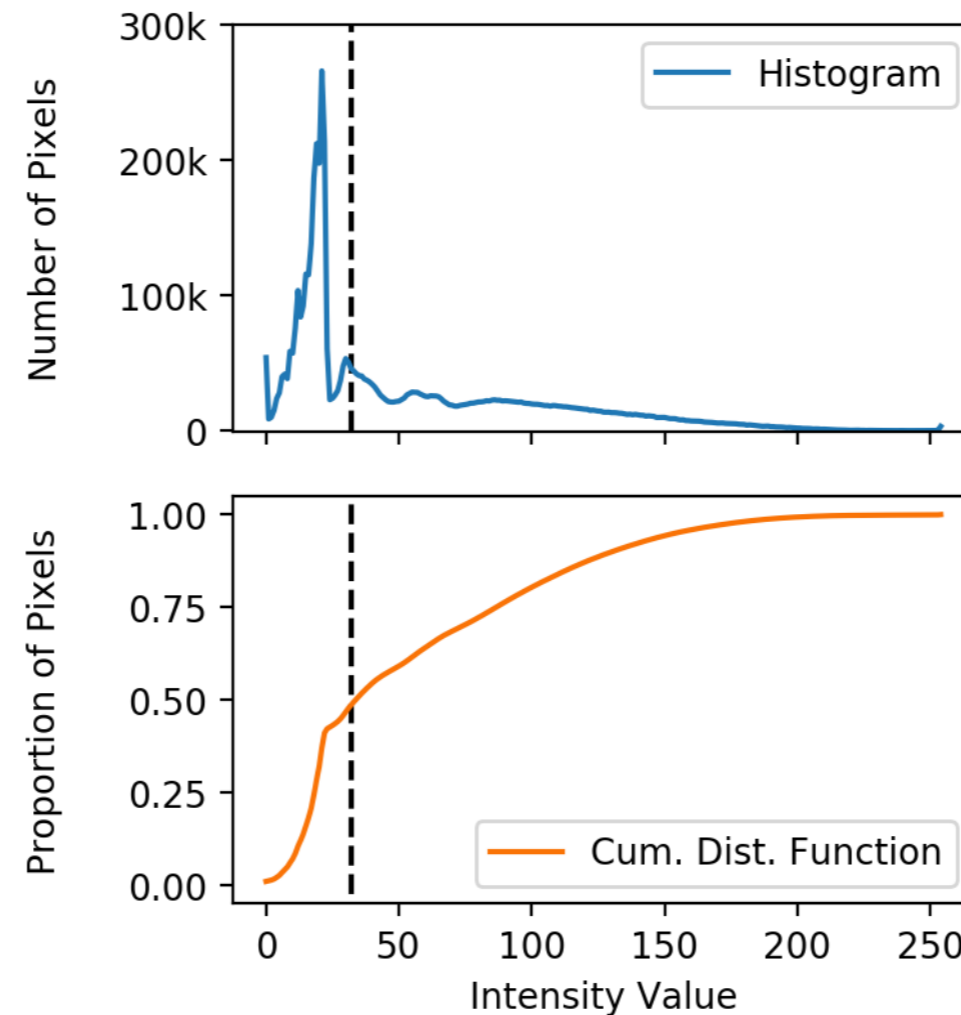
```
import scipy.ndimage as ndi  
hist=ndi.histogram(im, min=0,  
                   max=255,  
                   bins=256)  
  
hist.shape
```

```
(256,)
```



Equalization

- Distributions often skewed toward low intensities (background values).
- **Equalization:** redistribute values to optimize full intensity range.
- **Cumulative distribution function:** (CDF) shows proportion of pixels in range.



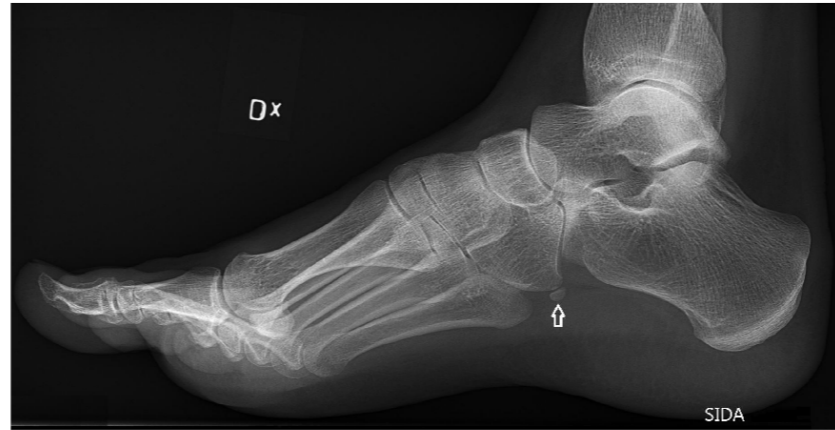
Equalization

```
import scipy.ndimage as ndi
hist = ndi.histogram(im, min=0,
                    max=255,
                    bins=256)

cdf = hist.cumsum() / hist.sum()
cdf.shape
```

(256,)

```
im_equalized = cdf[im] * 255
fig, axes = plt.subplots(2, 1)
axes[0].imshow(im)
axes[1].imshow(im_equalized)
plt.show()
```

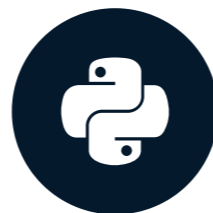


Let's practice!

BIOMEDICAL IMAGE ANALYSIS IN PYTHON

Masks

BIOMEDICAL IMAGE ANALYSIS IN PYTHON



Stephen Bailey
Instructor

Masks

Raw image

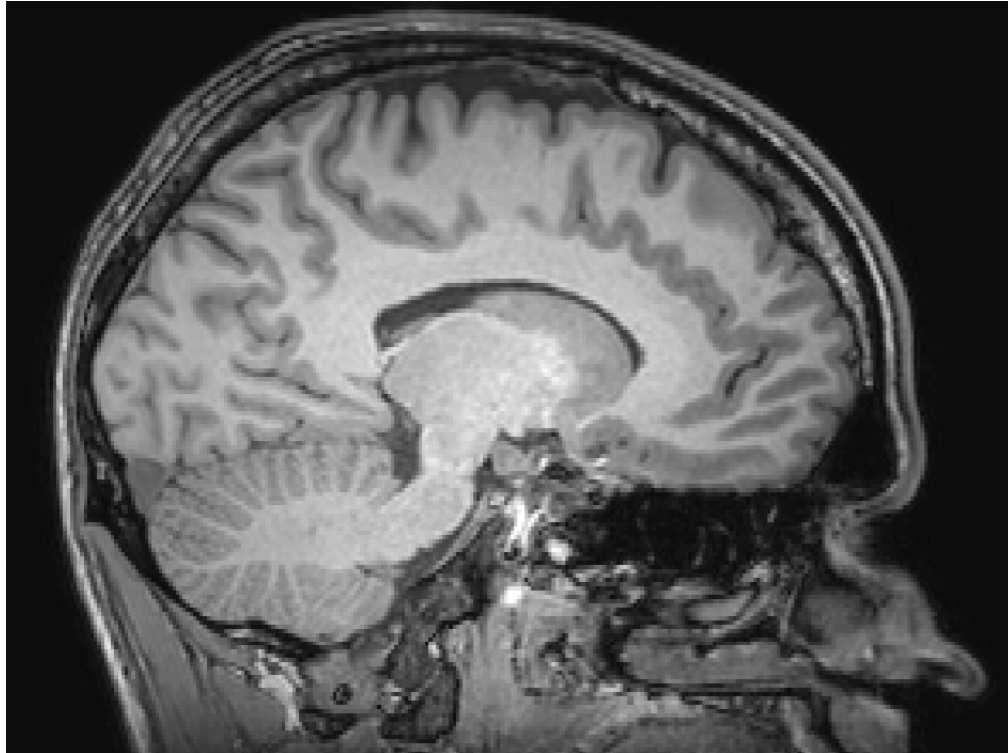
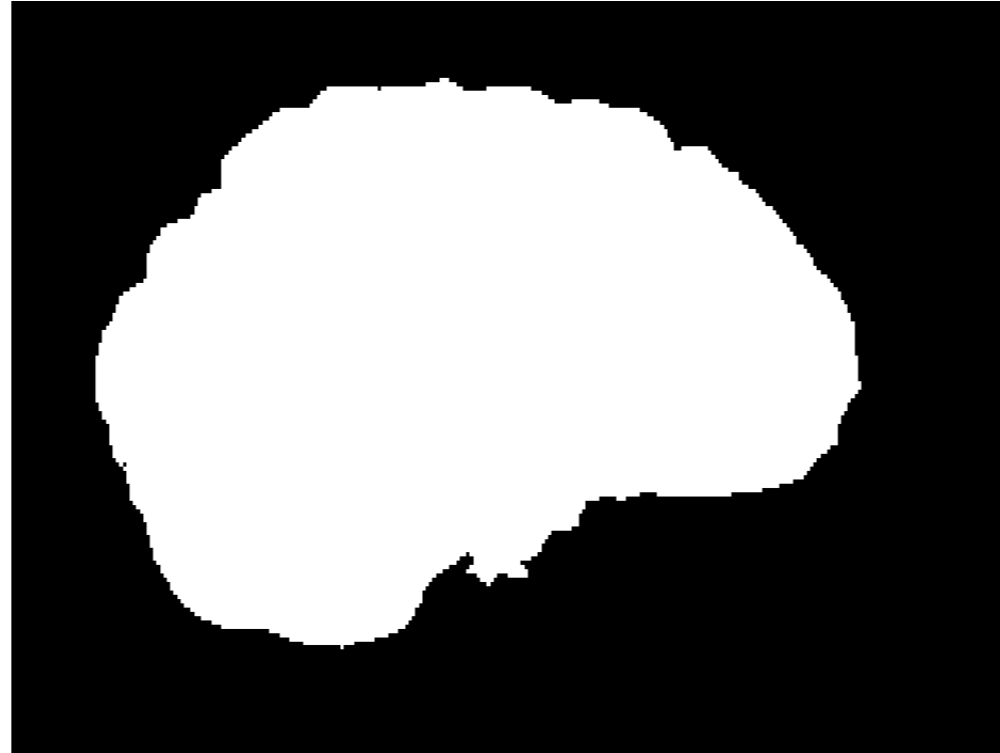


Image mask



Creating masks

Logical operations result in
`True` / `False` at each pixel

```
mat = np.array([[1, 2, 3],  
                [4, 5, 6],  
                [7, 8, 9]])  
  
mat > 5
```

```
np.array([[False, False, False],  
        [False, False, True ],  
        [True,  True,  True ]])
```

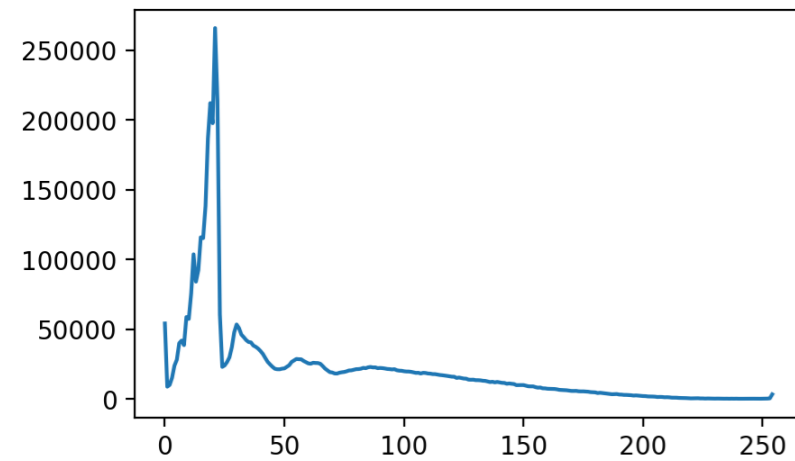
Sample Operations

Operation	Example
Greater	<code>im > 0</code>
Equal to	<code>im == 1</code>
X and Y	<code>(im > 0) & (im < 5)</code>
X or Y	<code>(im > 10) (im < 5)</code>

Creating masks

```
hist=ndi.histogram(im, 0, 255, 256)
```

```
mask1 = im > 32
```



Creating masks

```
mask2 = im > 64
```

```
mask3 = mask1 & ~mask2
```



Applying masks

```
np.where(condition, x, y)
```

controls what data passes through the mask.

```
import numpy as np  
im_bone = np.where(im > 64, im, 0)
```

```
plt.imshow(im_bone, cmap='gray')  
plt.axis('off')  
plt.show()
```



Tuning masks

```
m = np.where(im > 64, 1, 0)
```



```
ndi.binary_dilation(m, iterations=5)
```



Tuning masks

```
ndi.binary_erosion(m, iterations=5)
```

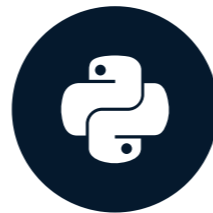


Let's practice!

BIOMEDICAL IMAGE ANALYSIS IN PYTHON

Filters

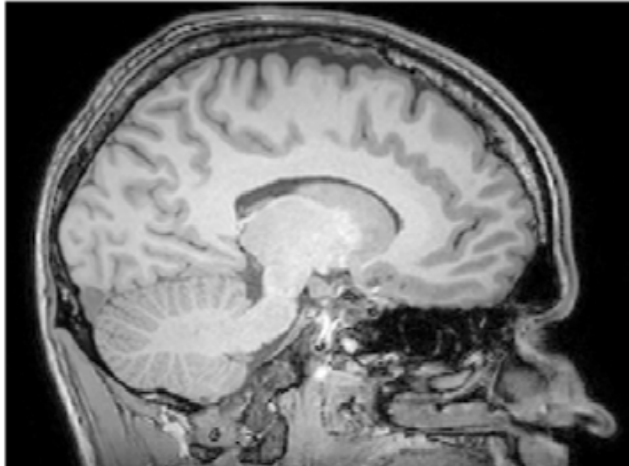
BIOMEDICAL IMAGE ANALYSIS IN PYTHON



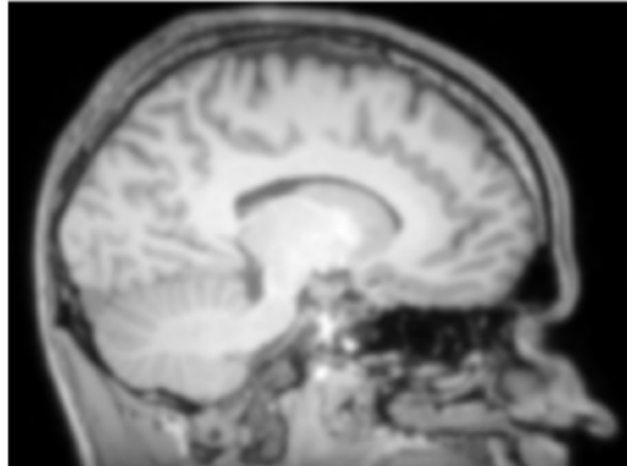
Stephen Bailey
Instructor

Filters

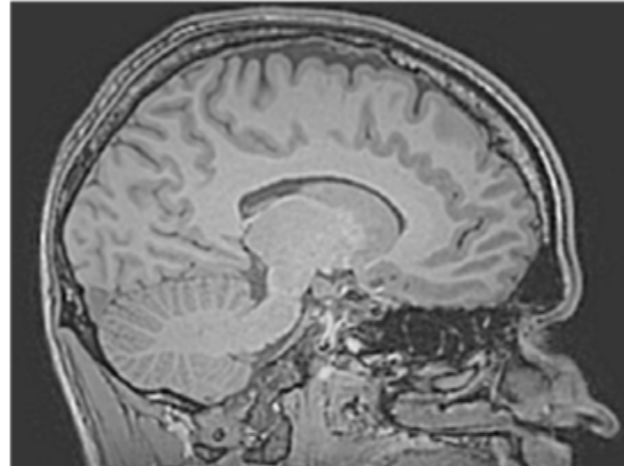
Original



Smoothed



Sharpened



Convolution with a sharpening filter

Input Array

1	1	1	1	1
1	1	1	1	1
1	1	2	1	1
1	1	1	1	1
1	1	1	1	1

*

Filter Weights /
Kernel

0	-1	0
-1	5	-1
0	-1	0

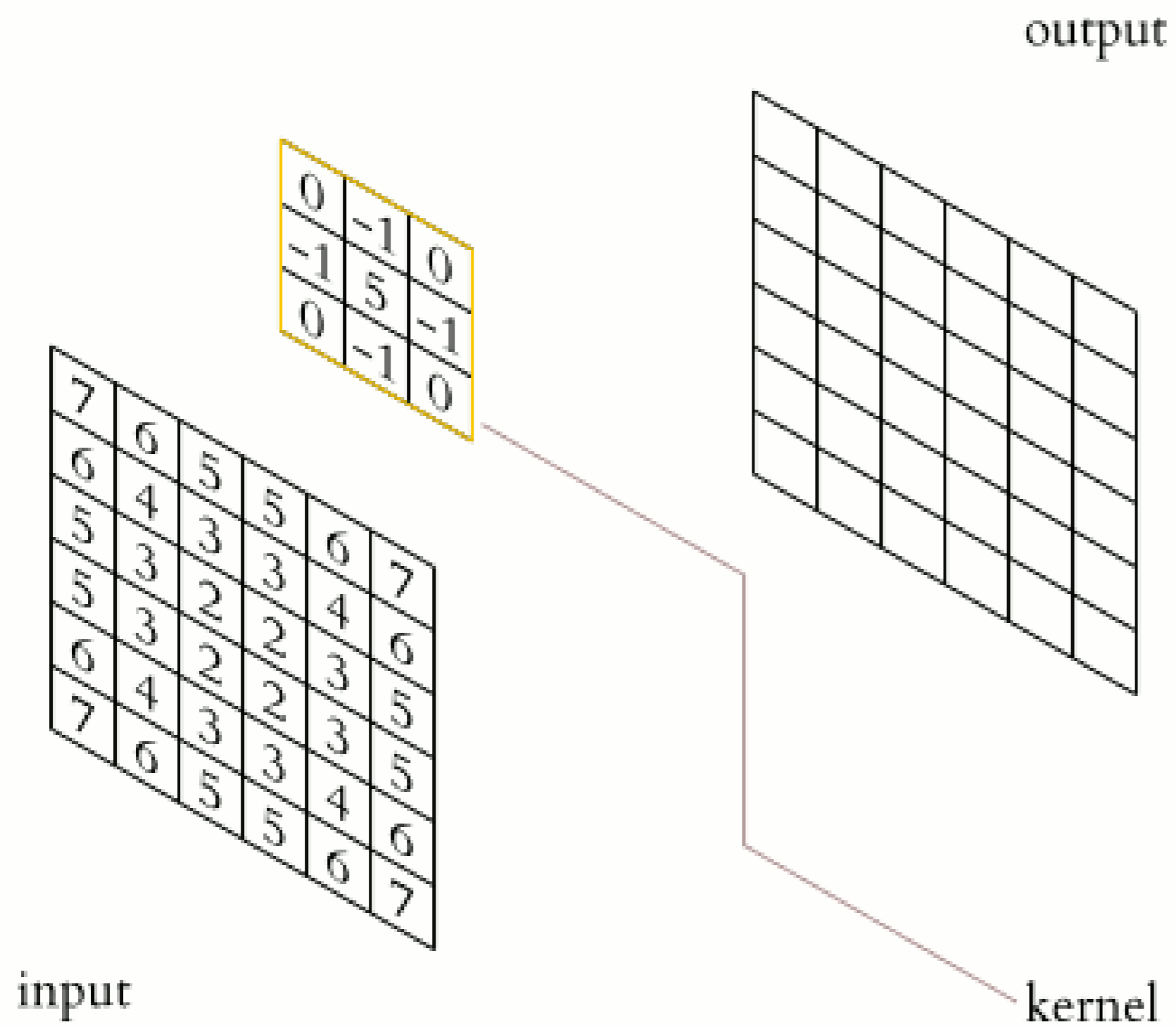
Convolution with a sharpening filter

Sum

$1 * 0$	$1 * -1$	$1 * 0$
$1 * -1$	$2 * 5$	$1 * -1$
$1 * 0$	$1 * -1$	$1 * 0$

=

6



input

7	6	5	5	6	7
6	4	3	3	4	6
5	3	2	2	3	5
5	3	2	2	3	5
6	4	3	3	4	6
7	6	5	5	6	7

0	-1	0
-1	5	-1
0	-1	0

output

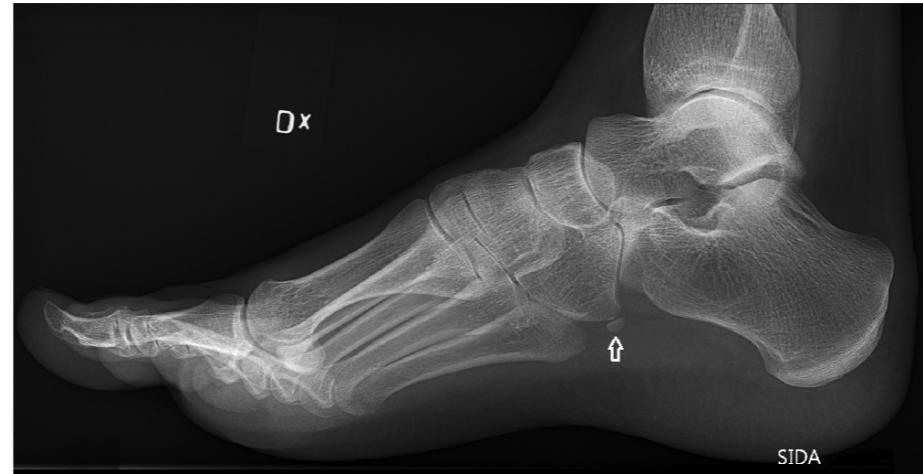
Image convolution

```
import imageio
import scipy.ndimage as ndi

im=imageio.imread('foot-xray.jpg')
weights = [[.11, .11, .11],
           [.11, .12, .11],
           [.11, .11, .11]]

im_filt = ndi.convolve(im, weights)
```

```
fig, axes = plt.subplots(2, 1)
axes[0].imshow(im, cmap='gray')
axes[1].imshow(im_filt, cmap='gray')
plt.imshow()
```



Filtering functions

`scipy.ndimage.filters`

includes:

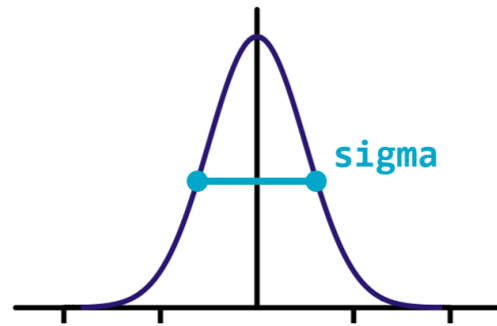
- `median_filter()`
- `uniform_filter()`
- `maximum_filter()`
- `percentile_filter()`

```
ndi.median_filter(im, size=10)
```

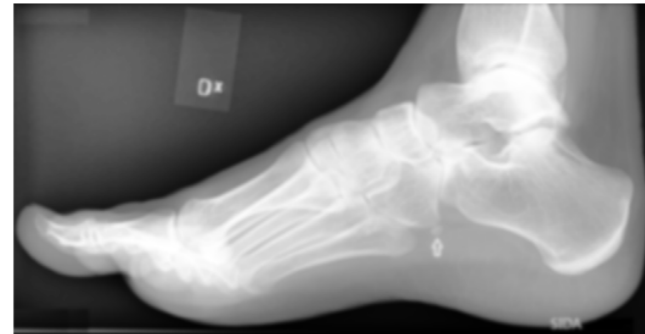


Gaussian filtering

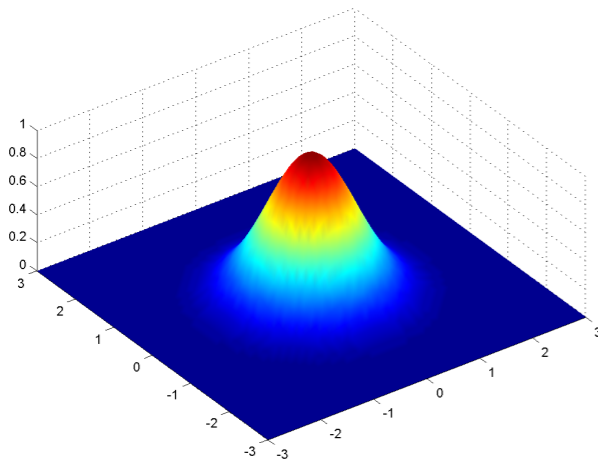
Gaussian distribution in 1 dimension



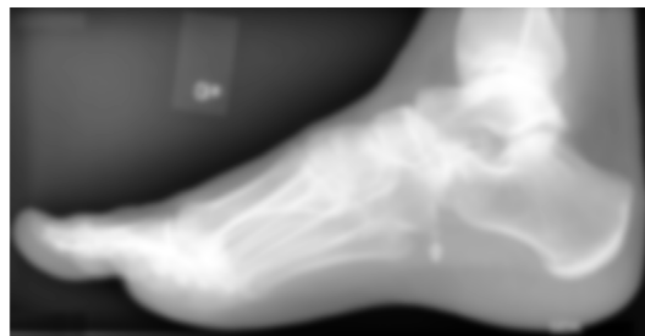
```
ndi.gaussian_filter(im, sigma=5)
```



Gaussian distribution in 2 dimensions



```
ndi.gaussian_filter(im, sigma=10)
```



Let's practice!

BIOMEDICAL IMAGE ANALYSIS IN PYTHON

Feature detection

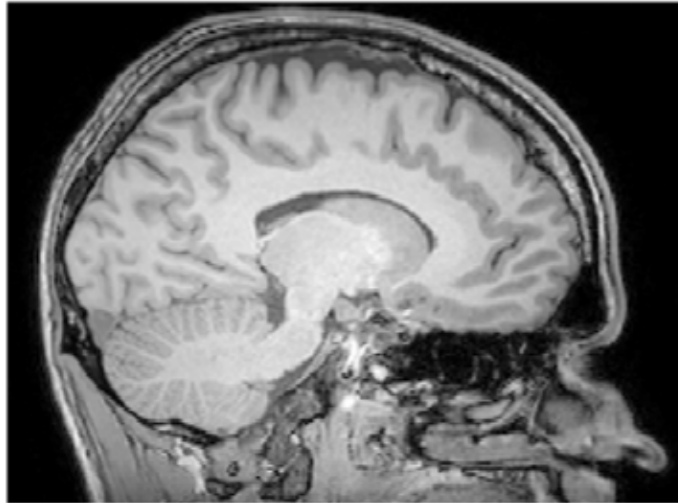
BIOMEDICAL IMAGE ANALYSIS IN PYTHON



Stephen Bailey
Instructor

Edges: sharp changes in intensity

Original



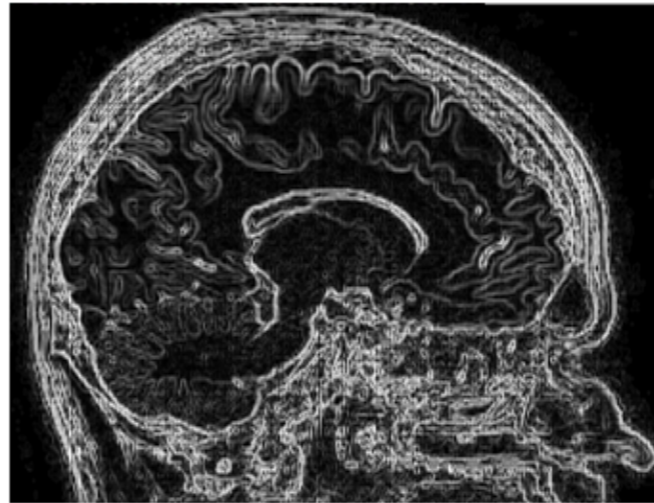
Kernel

?	?	?
?	?	?
?	?	?

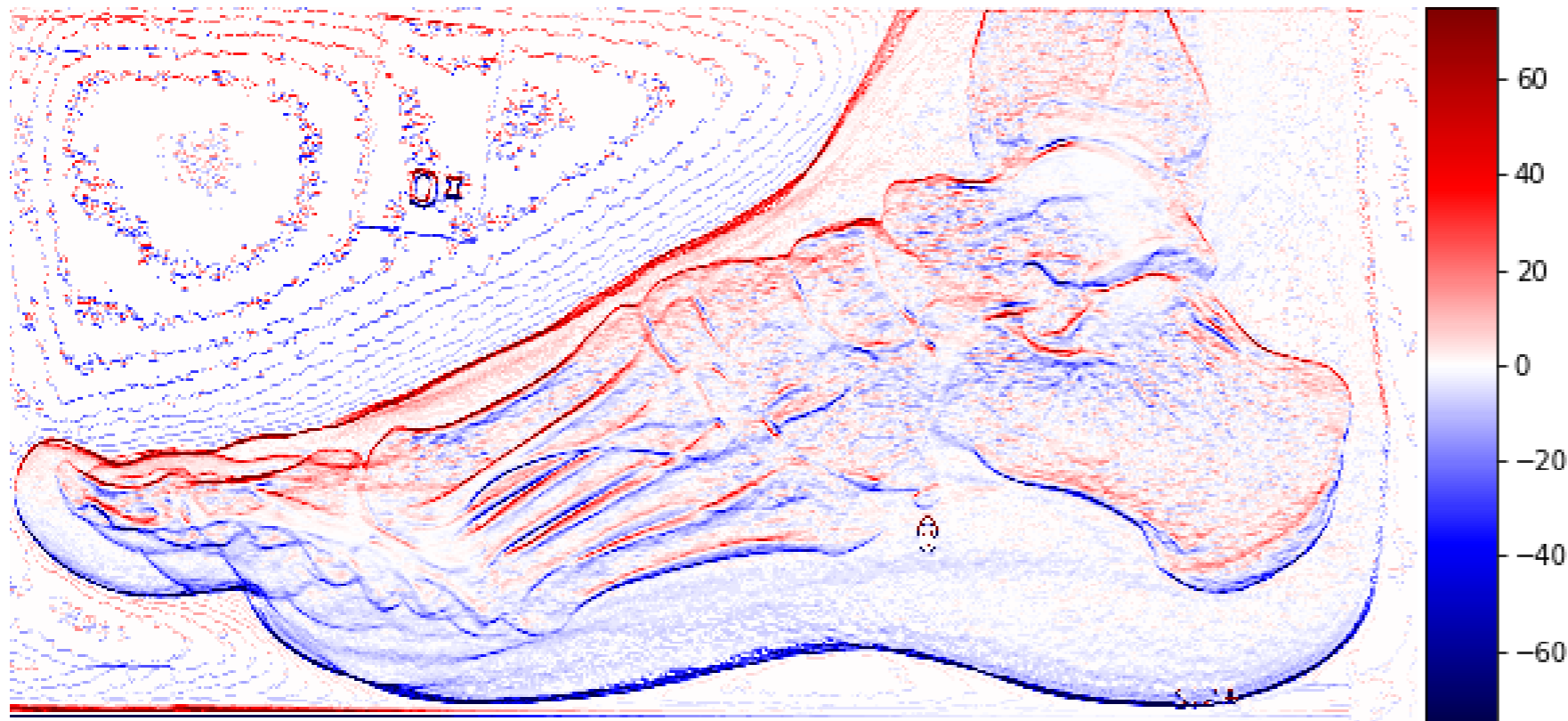
*

=

Edges



```
im=imageio.imread('foot-xray.jpg')
weights = [[+1, +1, +1],
           [ 0,  0,  0],
           [-1, -1, -1]]
edges = ndi.convolve(im, weights)
plt.imshow(edges, cmap='seismic')
```



Sobel filters

Sobel (H)

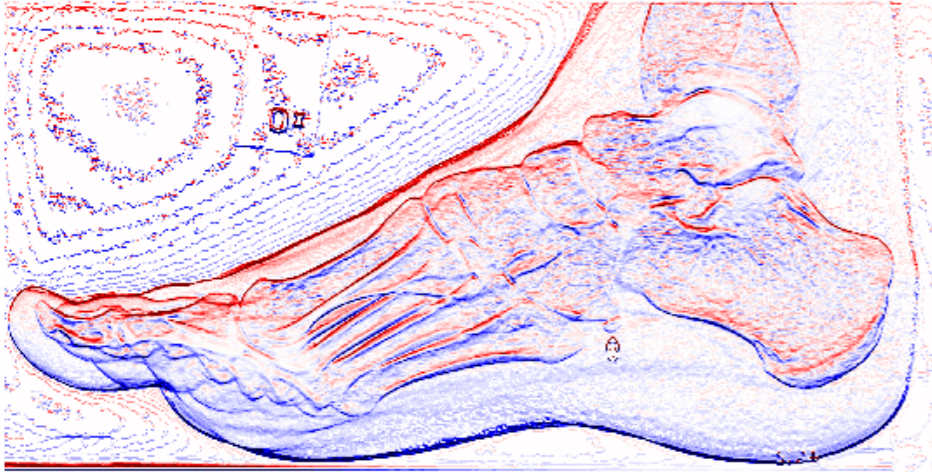
1	2	1
0	0	0
-1	-2	-1

Sobel (V)

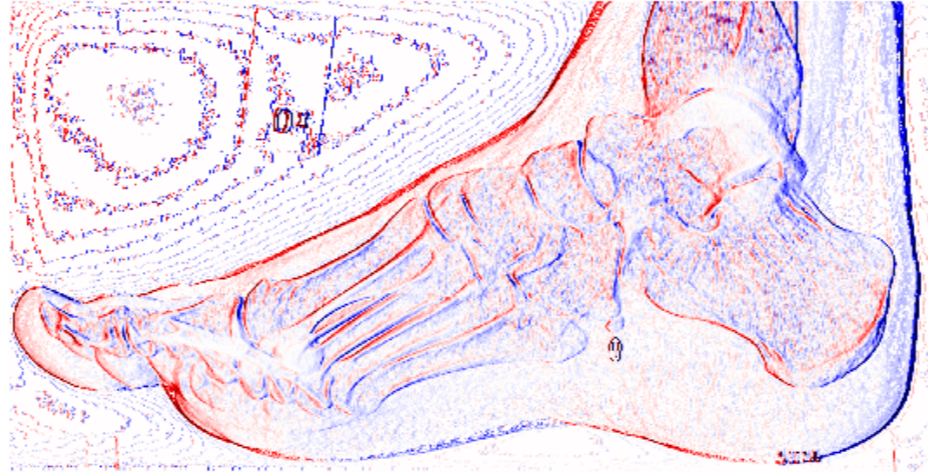
1	0	-1
2	0	-2
1	0	-1

Sobel filters

```
ndi.sobel(im, axis=0)
```



```
ndi.sobel(im, axis=1)
```



Sobel filter magnitude

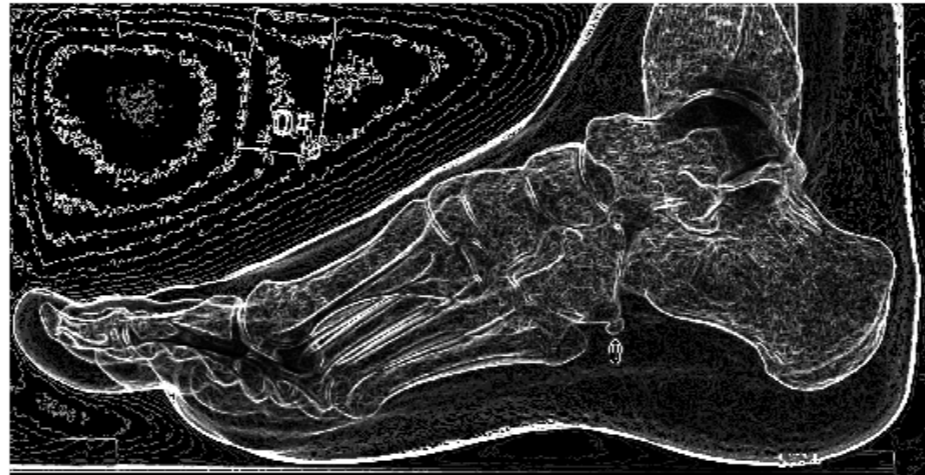
Combine horizontal and vertical edge data by calculating distance:

$$z = \sqrt{x^2 + y^2}$$

```
edges0=ndi.sobel(im, axis=0)
edges1=ndi.sobel(im, axis=1)

edges=np.sqrt(np.square(edges0) +
              np.square(edges1))
```

```
plt.imshow(edges, cmap='gray')
```



Let's practice!

BIOMEDICAL IMAGE ANALYSIS IN PYTHON