Introducción al Procesamiento de Imágenes

Ivan Cruz Aceves
ivan.cruz@cimat.mx

Centro de Investigación en Matemáticas, A.C. (CIMAT)

Cubo- I304, Ext. 4506

Contenido

Lectura de imagen Python-Numpy/Scipy

```
from scipy import misc
import matplotlib.pyplot as plt
imagen = misc.imread('img.jpeg')
print imagen.shape
plt.imshow(imagen)
```

Estadísticas de imagen Python-Numpy/Scipy

```
from scipy import misc
import matplotlib.pyplot as plt
imagen = misc.imread('img.jpeg')
print imagen.shape
print type(imagen)
print imagen.mean()
print imagen.max()
print imagen.min()
plt.imshow(imagen)
```

Operador Negativo de imagen Python-Numpy/Scipy

```
from scipy import misc
import matplotlib.pyplot as plt
imagen = misc.imread('img.jpeg')
fig, (uno, dos) = plt.subplots(1, 2)
uno.imshow(imagen)
imagen = 255 - imagen
dos.imshow(imagen);
```

Lectura de imagen Python-PIL

```
from PIL import Image
im = Image.open('img.jpeg')
im.show()
```

Obteniendo datos de imagen Python-PIL

```
from PIL import Image
im = Image.open('img.jpeg')
im.show()
W = im.size[0]
H = im.size[1]
print W,H,im.size, im.mode, im.format
```

284 177 (284,177) RGB JPEG

Guardando imagen en formato de grises Python-PIL

```
from PIL import Image
im = Image.open('img.jpeg')
im.show()
imgris=im.convert('L') # luminance
imgris.save('img_gris.png')
```

Lectura en C++

- OpenCV
- Imágenes PGM
- GDCM
- Cairo
- WxWidgets
- Qt
- Gtk

Trabajando con GDCM en C++

```
#include <gdcmImageReader.h>
#include <gdcmImage.h>
#include <gdcmReader.h>
#include <gdcmTag.h>
----------------------
gdcm::ImageReader reader;//imagen gdcm
reader.SetFileName(pathdcm1.mb_str());
-------------------------------------
gdcm::File &file = reader.GetFile();
gdcm::DataSet &ds = file.GetDataSet();
const gdcm::Image &gimage = reader.GetImage();
const unsigned int* dimension=gimage.GetDimensions();
const double* pixelratio = gimage.GetSpacing();
```

```
gdcm::Tag tsis9(0x0028,0x0030); // PixelSpacing
if ( ds.FindDataElement( tsis9 ) )
{
const gdcm::DataElement &sis=ds.GetDataElement(tsis9);
const gdcm::ByteValue *bv = sis.GetByteValue();
std::istringstream is;
std::string dup( bv->GetPointer(), bv->GetPointer()
+ bv->GetLength() );
is.str( dup );
Grid1->SetCellValue(9,0,wxString(dup.c_str(),
wxConvUTF8));
}
else
Grid1->SetCellValue(9,0,wxT("Not present"));
```

```
gdcm::Tag tsis12(0x0010,0x0010); // PatientName
if ( ds.FindDataElement( tsis12 ) )
{
const gdcm::DataElement &sis=ds.GetDataElement(tsis12);
const gdcm::ByteValue *bv = sis.GetByteValue();
std::istringstream is;
std::string dup( bv->GetPointer(), bv->GetPointer()
+ bv->GetLength() );
is.str( dup );
Grid1->SetCellValue(12,0,wxString(dup.c_str(),
wxConvUTF8));
}
else
Grid1->SetCellValue(12,0,wxT("Not present"));
```

Algunas operaciones puntuales:

- Operación de escala de grises

Algunas operaciones puntuales:

- Operación de escala de grises

$$Y = 0.3R + 0.59G + 0.11B$$
$$Y = (R + G + B)/3.0$$

Algunas operaciones puntuales:

- Operación de escala de grises

$$Y = 0.3R + 0.59G + 0.11B$$
$$Y = (R + G + B)/3.0$$

- Operador Inverso o Negativo

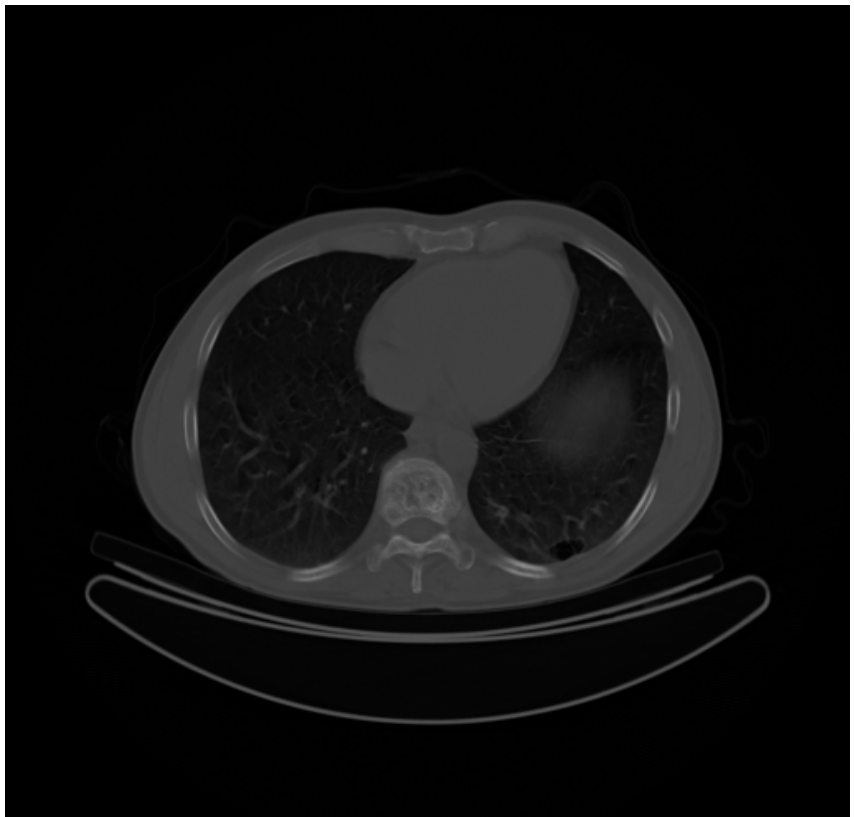
Algunas operaciones puntuales:

- Operación de escala de grises

$$Y = 0.3R + 0.59G + 0.11B$$
$$Y = (R + G + B)/3.0$$

- Operador Inverso o Negativo

$$g(x, y) = 255 - f(x, y)$$

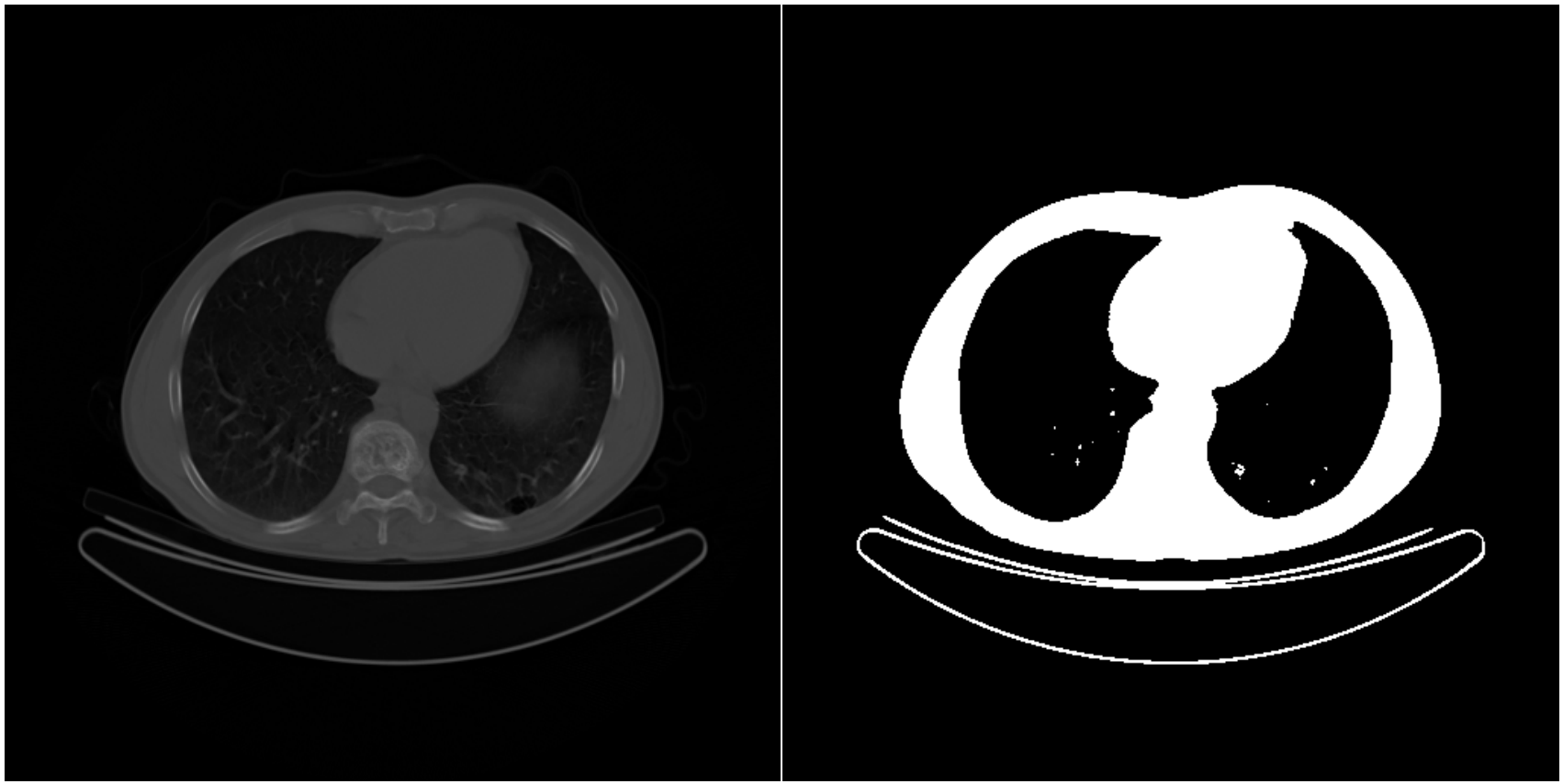


Operador de Negativo en C++

```
void Negative(unsigned char *data, int w, int h)
{
//regresamos el negativo de una imagen
for(int y=0; y < h; y++)
for(int x=0; x < w; x++)
{
long pos = (y * w + x) * 3;
data[pos]   = 255- data[pos];
data[pos+1] = 255- data[pos+1];
data[pos+2] = 255- data[pos+2];
}
}// fin de funcion
```

Operación de Umbral

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq Th = (50/255) \\ 0 & \text{if } f(x,y) < Th = (50/255) \end{cases}$$



Otras operaciones de umbral

- Operador de umbral binario

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \leq Th_1 \text{ } or \text{ } f(x,y) \geq Th_2 \\ 0 & \text{if } Th_1 < f(x,y) < Th_2 \end{cases}$$

Otras operaciones de umbral

- Operador de umbral binario

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \leq Th_1 \text{ or } f(x,y) \geq Th_2 \\ 0 & \text{if } Th_1 < f(x,y) < Th_2 \end{cases}$$

- Reducción del nivel de gris a n niveles (compresión)

$$g(x,y) = \begin{cases} 0 & \text{if } f(x,y) \leq p_1 \\ q_1 & \text{if } p_1 < f(x,y) \leq p_2 \\ q_2 & \text{if } p_2 < f(x,y) \leq p_3 \\ q_n & \text{if } p_{n-1} < f(x,y) \leq 255 \end{cases}$$

Otras operaciones de umbral

- Operador de umbral binario

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \leq Th_1 \text{ or } f(x,y) \geq Th_2 \\ 0 & \text{if } Th_1 < f(x,y) < Th_2 \end{cases}$$

- Reducción del nivel de gris a n niveles (compresión)

$$g(x,y) = \begin{cases} 0 & \text{if } f(x,y) \leq p_1 \\ q_1 & \text{if } p_1 < f(x,y) \leq p_2 \\ q_2 & \text{if } p_2 < f(x,y) \leq p_3 \\ q_n & \text{if } p_{n-1} < f(x,y) \leq 255 \end{cases}$$

- Algunas variantes:
  - Umbral binario invertido
  - Umbral de escala de grises
  - Umbral de escala de gris invertido

Transformaciones geométricas

- Traslación o Desplazamiento:

$$g(x, y) = f(x + \Delta, y + \Delta)$$

Transformaciones geométricas

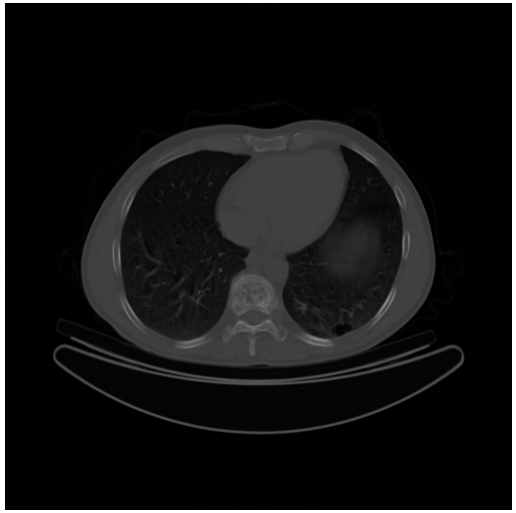
- Traslación o Desplazamiento:

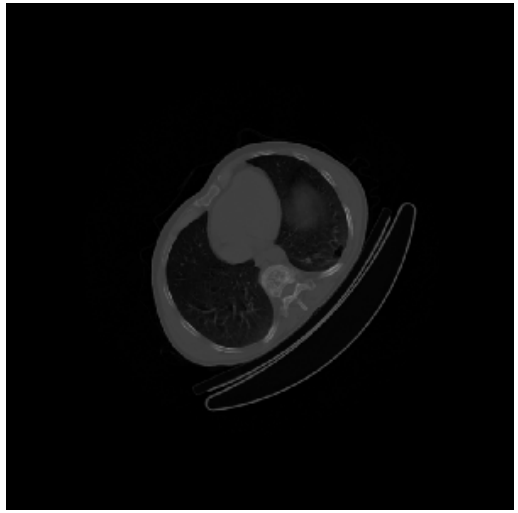
$$g(x, y) = f(x + \Delta, y + \Delta)$$

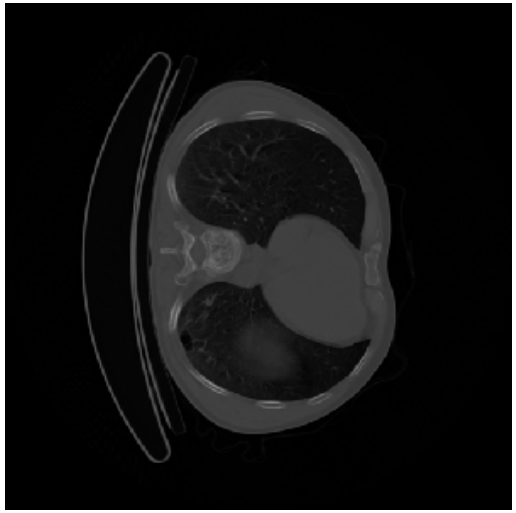
- Rotación :

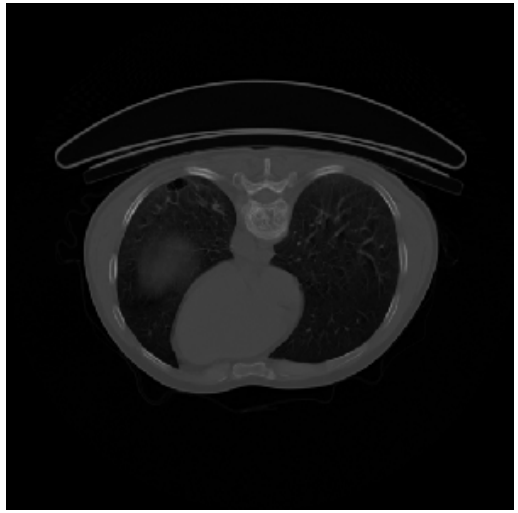
$$R_{\theta_i} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) \\ sin(\theta_i) & cos(\theta_i) \end{bmatrix}$$

Operación de Rotación









Transformaciones geométricas

- Escalamiento:

$$x_2 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1) + y_1$$

Transformaciones geométricas

- Escalamiento:

$$x_2 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1) + y_1$$

- Zoom (expansión lineal de valor medio):

$$IMG = \begin{bmatrix} 20 & 12 & 20 \\ 12 & 20 & 12 \\ 20 & 8 & 20 \end{bmatrix} \qquad ZOOM = \begin{bmatrix} 20 & 16 & 12 & 16 & 20 \\ 16 & 16 & 16 & 16 & 16 \\ 12 & 16 & 20 & 16 & 12 \\ 16 & 15 & 14 & 15 & 16 \\ 20 & 14 & 8 & 14 & 20 \end{bmatrix}$$

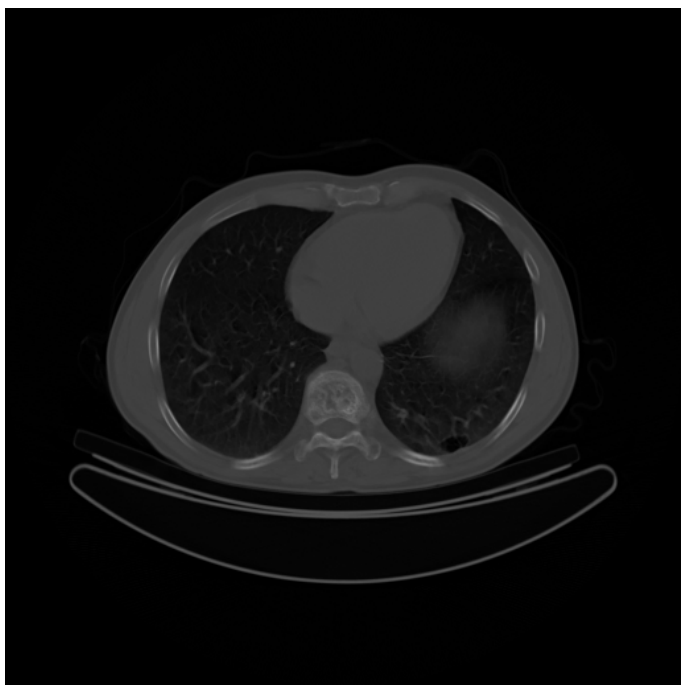
Ajuste lineal en C++

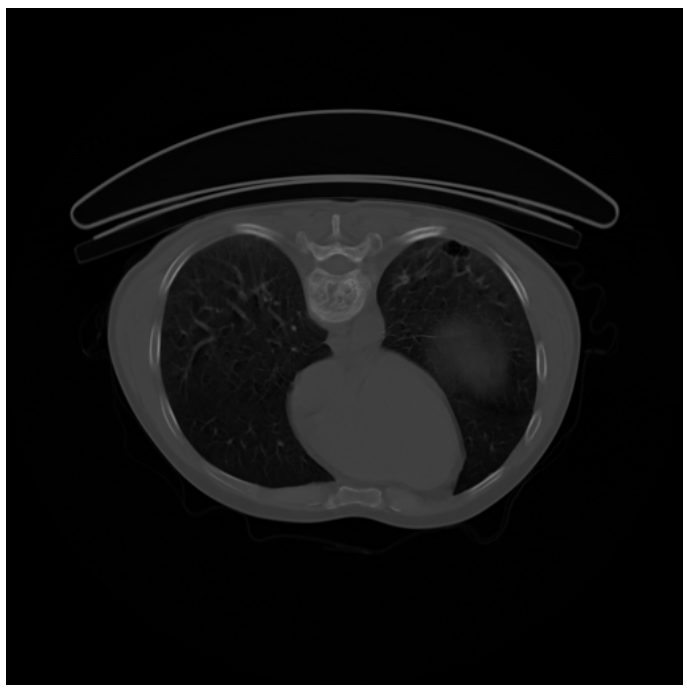
```
void Ajuste(unsigned char *data, int w, int h)
{
int max = data[0],min = data[0];
for(int x=0; x < (h*w*3); x++)
{
if(data[x] < min)
min = data[x];
if(data[x] > max)
max = data[x];
}
for( int i = 0 ; i < (h*w*3) ; i++ )
data[i] = (255.0/(max-min))*(data[i]-min);
}
```

Transformaciones geométricas

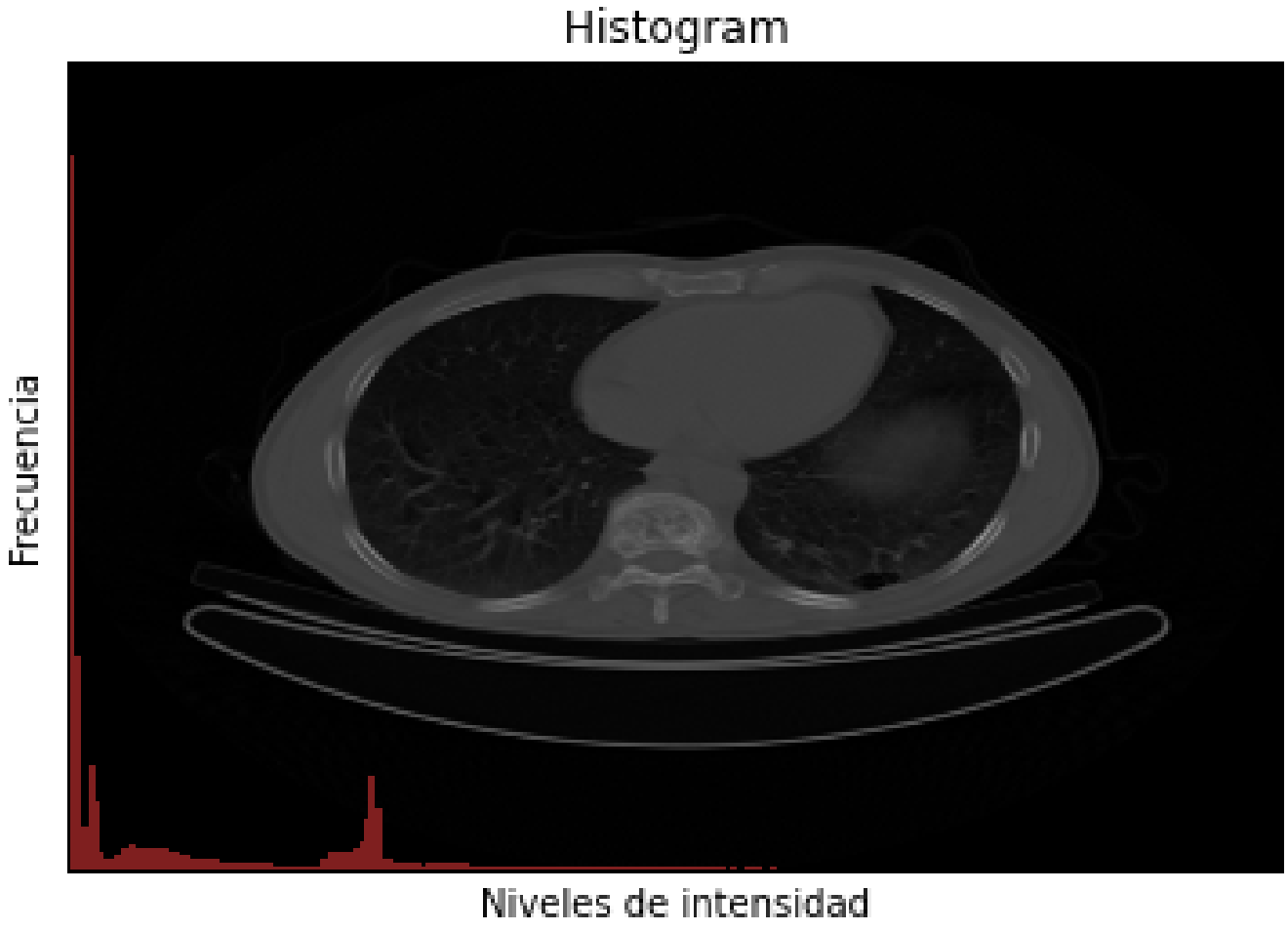
Funciones Espejo

- Vertical: $g(x, y) = f(N - x, y)$
- Horizontal: $g(x, y) = f(x, M - y)$
- Diagonal: $g(x, y) = f(N - x, M - y)$





Histograma de una imagen



Histograma de imagen en Python

```
from PIL import Image
imagen = '2.png'
image = Image.open(imagen).convert('L')#luminance
histo = image.histogram()
histo_string = ''
for i in histo:
histo_string += str(i) + "\n"
```

Operaciones sobre el Histograma

- Contracción y Expansión:

$$x_2 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1) + y_1$$

Operaciones sobre el Histograma

- Contracción y Expansión:

$$x_2 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1) + y_1$$

- Desplazamiento:

$$g(x, y) = f(x, y) + \Delta$$

Operaciones sobre el Histograma

- Contracción y Expansión:

$$x_2 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1) + y_1$$

- Desplazamiento:

$$g(x, y) = f(x, y) + \Delta$$

- Ecualización:

función de transferencia

Operaciones de dos imagenes punto a punto

- Operaciones aritméticas
  - Adicion: $g(x, y) = f_1(x, y) + f_2(x, y)$
  - Substracción: $g(x, y) = f_1(x, y) - f_2(x, y)$
  - Producto: $g(x, y) = f(x, y) * C$

Operaciones de dos imagenes punto a punto

- Operaciones aritméticas
  - Adicion: $g(x,y) = f_1(x,y) + f_2(x,y)$
  - Substracción: $g(x,y) = f_1(x,y) - f_2(x,y)$
  - Producto: $g(x,y) = f(x,y) * C$
- Operaciones lógicas
  - Negativo: $\sim f(x,y)$
  - AND: $g(x,y) = f_1(x,y) \& f_2(x,y)$
  - OR: $g(x,y) = f_1(x,y) | f_2(x,y)$
  - XOR: $g(x,y) = f_1(x,y) \wedge f_2(x,y)$

Tarea no entregable

- Lectura de Imágenes PGM (Python/Cpp)
- Lectura de imágenes DICOM (Python/Cpp)(Conversión)