Procesamiento de imágenes #07

Segmentación de la imagen

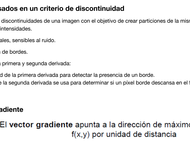
Dra. C. Mirela Escobedo Nicot

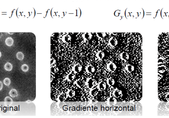
Resumen

- Segmentación.
- Métodos basados en un criterio de discontinuidad.
  - Operador gradiente.
  - Derivadas de segundo orden.
- Métodos basados en el criterio de homogeneidad.
  - Umbralización.
- Métodos de segmentación que son basados en la evolución de contornos.
  - Contornos activos paramétricos.
  - Contornos activos geométricos.

Segmentación

- Proceso que subdivide una imagen en sus partes constituyentes u objetos.





Imágenes Microscópicas de Leucocitos. *Tomada de Biomedical Image Analysis: Segmentation*

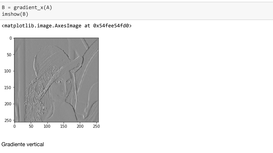
Métodos basados en un criterio de discontinuidad

- Enfatizan las discontinuidades de una imagen con el objetivo de crear particiones de la misma basada en los cambios abruptos de intensidades.
- Técnicas locales, sensibles al ruido.
- La detección de bordes.
- Cálculo de la primera y segunda derivada:
  - Magnitud de la primera derivada para detectar la presencia de un borde.
  - Signo de la segunda derivada se usa para determinar si un píxel borde descansa en el fondo o en un borde.

Operador gradiente

El **vector gradiente** apunta a la dirección de máximo crecimiento de f(x,y) por unidad de distancia

$$G_x(x,y) = f(x,y) - f(x,y-1) \qquad G_y(x,y) = f(x,y) - f(x+1,y)$$

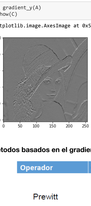


In [1]:
```
from pylab import *
rcParams['image.cmap'] = 'gray'
```

In [2]:
```
A = imread("dataset/lena_gray_256.tif").astype('float')
```

In [3]:
```
imshow(A)
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x54fe7cb940>
```



Gradiente horizontal

In [4]:
```
def gradient_x(image):
    R = zeros_like(image)

    for x in range(0, image.shape[0]):
        for y in range(0, image.shape[1]):
            R[x,y] = image[x,y] - image[x, y-1]

    return R
```

In [5]:
```
B = gradient_x(A)
imshow(B)
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x54fee54f00>
```

Gradiente vertical

In [6]:
```
def gradient_y(image):
    R = zeros_like(image)

    for x in range(0, image.shape[0]):
        for y in range(0, image.shape[1]):
            try:
                R[x,y] = image[x,y] - image[x+1, y]
            except:
                R[x,y] = image[x,y] - image[x, y]
    return R
```

In [7]:
```
C = gradient_y(A)
imshow(C)
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x54feec2cf8>
```

Métodos basados en el gradiente primer orden

Operador	Gradiente Fila (En y)	Gradiente Columna (En x)
Prewitt	$\frac{1}{3}\begin{pmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{pmatrix}$	$\frac{1}{3}\begin{pmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{pmatrix}$
Sobel	$\frac{1}{4}\begin{pmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{pmatrix}$	$\frac{1}{4}\begin{pmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{pmatrix}$
Frei-Chen	$\frac{1}{2+\sqrt{2}}\begin{pmatrix}1&0&-1\\\sqrt{2}&0&-\sqrt{2}\\1&0&-1\end{pmatrix}$	$\frac{1}{2+\sqrt{2}}\begin{pmatrix}-1&-\sqrt{2}&-1\\0&0&0\\1&\sqrt{2}&1\end{pmatrix}$
Roberts	$\begin{pmatrix}-1&0\\0&1\end{pmatrix}$	$\begin{pmatrix}0&-1\\1&0\end{pmatrix}$

In [8]:
```
from skimage import img_as_float

def imshow_all(*images, titles=None):
    images = [img_as_float(img) for img in images]

    if titles is None:
        titles = ['] ' * len(images)
    vmin = min(map(np.min, images))
    vmax = max(map(np.max, images))
    ncols = len(images)
    height = 5
    width = height * len(images)
    fig, axes = plt.subplots(nrows=1, ncols=ncols,
                             figsize=(width, height))
    for ax, img, label in zip(axes.ravel(), images, titles):
        ax.imshow(img, vmin=vmin, vmax=vmax)
        ax.set_title(label)
```

Prewitt

In [9]:
```
from skimage.filters import prewitt_h, prewitt_v

PH = prewitt_h(A); PV = prewitt_v(A)

imshow_all(PH, PV, titles=["prewitt horizontal", "prewitt vertical"])
```

Sobel

In [10]:
```
from skimage.filters import sobel_h, sobel_v

SH = sobel_h(A); SV = sobel_v(A)

imshow_all(SH, SV, titles=["sobel horizontal", "sobel vertical"])
```


Muchas de estas operaciones ya vienen programadas en `skimage`, pero recordemos que aplicar un filtro no es más que aplicar una operación de correlación o convolución entre la imagen y la matriz correspondiente al filtro.

Roberts

In [11]:
```
from scipy.ndimage import correlate
```

In [12]:
```
roberts_y = array([
    [ 1, 0],
    [ 0, -1]
])
```

In [13]:
```
roberts_x = array([
    [0, -1],
    [1, 0]
])
```

In [14]:
```
RH = correlate(A, roberts_x)
RV = correlate(A, roberts_y)

imshow_all(RH, RV, titles=["roberts horizontal", "roberts vertical"])
```


Frei-Chen

In [15]:
```
fchen_y = array([
    [ 1, 0, -1],
    [sqrt(2), 0, -sqrt(2)],
    [ 1, 0, -1]
])*(1/(2+sqrt(2)))
```

In [16]:
```
fchen_x = array([
    [ 1, sqrt(2), -1],
    [ 0, 0, 0],
    [-1, sqrt(2), 1]
])*(1/(2+sqrt(2)))
```

In [17]:
```
FH = correlate(A, fchen_x)
FV = correlate(A, fchen_y)

imshow_all(FH, FV, titles=["frei-chen horizontal", "frei-chen vertical"])
```