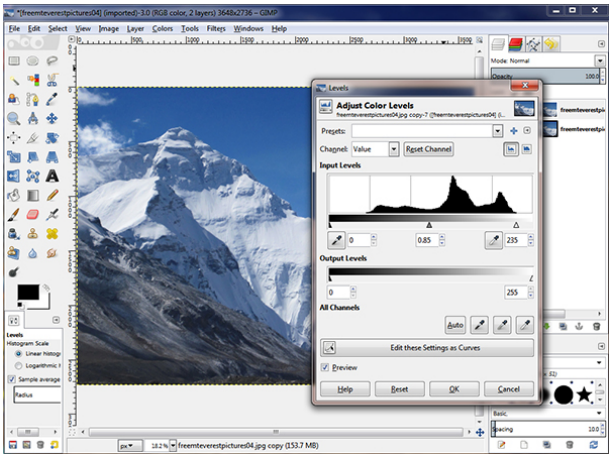
Practica_2

Informática 2014-2015

F. Matemática, Enero, 2015

Práctica 2, Ecualización de imágenes

Hay muchas aplicaciones informáticas que permiten *modificar/mejorar/transformar* una fotografía digital.



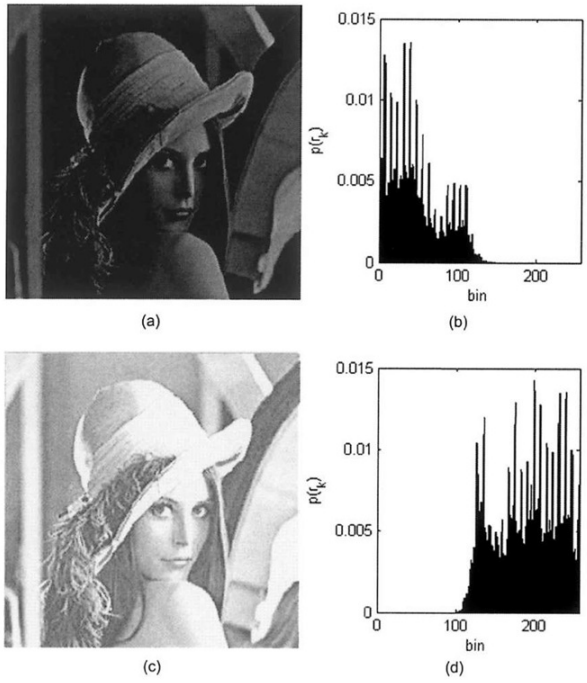
Una técnica que se utiliza frecuentemente para mejorar el contraste en las fotografías es la ecualización del histograma, en el siguiente enlace de la wikipedia pueden verse ejemplos y una definición precisa: http://en.wikipedia.org/wiki/Histogram_equalization

Es más fácil de lo que puede parecer. El problema puede resolverse a partir de funciones muy sencillas. Sigue los pasos que te indicamos a continuación y lo comprobarás tú mismo. Por sencillez, vamos a considerar imágenes en blanco y negro, por tanto, cada píxel de la imagen tiene un valor entre 0 y 255.

Al final del enunciado diremos cómo se hace para trabajar con imágenes en color

1. Histograma de una imagen

El histograma de una imagen es una función que para cada posible valor entre 0 y 255 nos dice el número de pixeles de la imagen que tienen dicho valor. Es decir, es la función de distribución de los valores. Estudia las imágenes siguiente:



A la izquierda aparecen dos fotografías y a la derecha sus correspondientes histogramas. La fotografía superior es muy oscura y por tanto el histograma acumula muchos valores en la zona baja, en los números próximos a 0. Por el contrario, la fotografía inferior es muy clara y la distribución de valores es alta en los números próximos al máximo, es decir a 255.

Escribe una función `histogram()` que recibe como parámetro una imagen, en blanco y negro, y que devuelve una lista (de 256 elementos) con la frecuencia de aparición de cada uno de los posibles valores de los pixeles de la imagen (es decir, las ves que aparece el 0, las veces que aparece el 1, . . . , las veces que aparece el 255).

Ejemplo

Ejemplo de uso de la función `histogram()` con una imagen. La imagen original tiene poco contraste, en este caso concreto todos los valores de los píxeles están entre 3 y 104. Si queréis hacer pruebas, la imagen puede conseguirse en http://wild.mat.ucm.es/img/tesla_bw.png

In [2]:
```
i = Image.open("img/tesla_bw.png",'r')
i.show()
h = histogram(i)
type(h), len(h), h[0:10]
```

Out [2]:
```
(list, 256, [0, 0, 0, 2, 14, 56, 85, 101, 60, 84])
```

2. La distribución acumulada

A partir del histograma de una imagen, es sencillo construir la función de distribución acumulada que nos dice el número de pixeles que una imagen que son menores o iguales a un cierto valor. Formalmente, $\text{cdf}[v] = \sum_{i \in \{0..v\}} \text{hist}[i]$ para cada $v \in \{0..255\}$.

Escribe una fución que se llama `cumulative_distribution()` que a partir de un histograma calcula la función de distribución acumulada de dicho histograma.

Ejemplo

Continuando con el ejemplo anterior, vamos a calcular la distribución acumulada.

In [4]:
```
i = Image.open("img/tesla_bw.png",'r')
i.show()
h = histogram(i)
print type(h), len(h), h[0:10]
cdf = cumulative_distribution(h)
print type(cdf), len(cdf), cdf[0:10], cdf[250:]
```

```
<type 'list'> 256 [0, 0, 0, 2, 14, 56, 85, 101, 60, 84]
<type 'list'> 256 [0, 0, 0, 2, 16, 72, 157, 258, 318, 402] [125000,
125000, 125000, 125000, 125000, 125000]
```

3. Dibujando histogramas y su distribución acumulada.

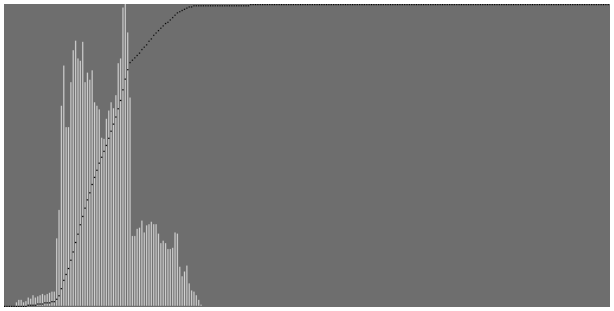
Ya que mirar listas tan largas no es práctico ni sencillo, vamos a pintar el histograma y su distribución acumulada para poder analizarlas de un 'vistazo'. Aquí tienes una función sencilla que permite esta visualización.

In [5]:
```
def draw_histogram_cumulative_distribution(hist,cdf):
    """
        Draw a graphic representing the given histogram
        and cumulative distribution funcion
        @type hist: list
        @type  cdf: list
    """
    k = 2
    width = 256*k
    lenght = 128*k
    bg_color = 110
    h_color = 210
    cdf_color = 0
    result=Image.new("L",(width,lenght), bg_color)
    aspect_factor_h = float(lenght-1)/max(hist)
    aspect_factor_cdf = float(lenght-1)/cdf[255]
    for x in xrange(256):
        for i in xrange(lenght-1-int(hist[x]*aspect_factor_h),lenght-1):
            result.putpixel((k*x,i),h_color)
        result.putpixel((k*x,lenght-1-int(cdf[x]*aspect_factor_cdf)), cdf_color)
    return result
```

Ejemplo

Ahora podemos visualizar el histograma y la distribución acumulada. Con los colores por defecto en la función anterior, el histograma sale en un color más claro que el fondo y la distribución acumulada en negro.

In [6]:
```
i = Image.open("img/tesla_bw.png",'r')
i.show()
h = histogram(i)
cdf = cumulative_distribution(h)
view = draw_histogram_cumulative_distribution(h,cdf)
view.show()
```



4. Ecualización del histograma

La ecualización consiste en transformar el valor de cada pixel con la fórmula siguiente:

$$h(v) = \text{round}\left(\frac{cdf[v] - cdf_{min}}{(width \times height) - cdf_{min}} \times (256 - 1)\right)$$

Dada una cdf sólo necesitamos saber el menor valor no nulo de cdf (cdf_min). El valor de cdf[255] es siempre el número de píxeles de la imagen considerada, que es exáctamente el valor de $(width \times height)$.

h es una función:

$$h : \{0, 1, \ldots, 255\} \to \{0, 1, \ldots, 255\}$$

Codificaremos la función h mediante una lista `l` de forma que `l[v]` = $h(v), \forall v \in \{0, 1, \ldots, 255\}$

Escribe una función `transformation_table()` que toma como parámetro una lista que representa una distribución acumulada y devuelve la lista `l` que codifica la función h.

Ejemplo

La forma de utilizar la función `transformation_table()` es la siguiente:

In [9]:
```
i = Image.open("img/tesla_bw.png",'r')
h = histogram(i)
cdf = cumulative_distribution(h)
equ = transformation_table(cdf)
print type(equ), len(equ), equ[0:10], equ[250:]
```

```
<type 'list'> 256 [0, 0, 0, 0, 0, 0, 0, 1, 1, 1] [255, 255, 255, 255,
255, 255]
```

5. Imagen ecualizada

La imagen ecualizada se obtiene de la original sustituyendo el valor `v` por $h(v)$ en cada pixel. Las funciones de los apartados anteriores nos permiten llegar a nuestro objetivo.

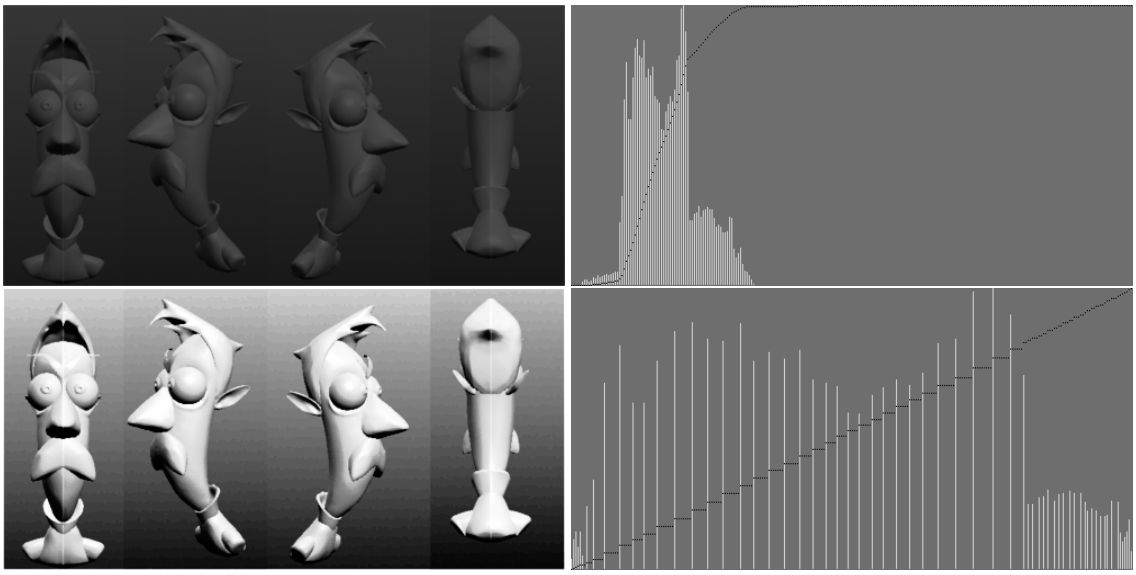
Escribe una función `equalization()` que recibe como parámero una imagen y devuelve la imagen ecualizada.

Ejemplo

Un ejemplo de la ejecución completa sería el siguiente:

In [11]:
```
#Imagen original
i = Image.open("img/tesla_bw.png",'r')
i.show()
h = histogram(i)
cdf = cumulative_distribution(h)
view = draw_histogram_cumulative_distribution(h,cdf)
view.show()

#Transformamos la imagen
equ = transformation_table(cdf)
j = equalization(i)
j.show()
h_eq = histogram(j)
cdf_eq = cumulative_distribution(h_eq)
view_eq = draw_histogram_cumulative_distribution(h_eq,cdf_eq)
view_eq.show()
```



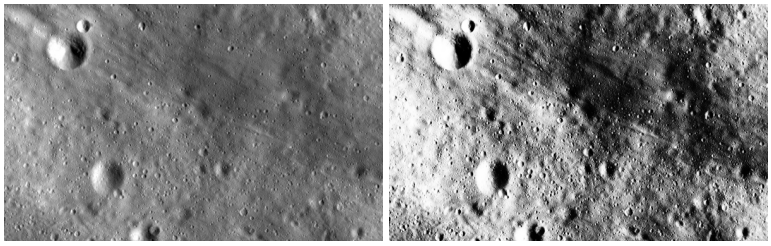
Epílogo

Observa como la distribución acumulada de la imagen ecualizada es lineal. Ese es precisamente el objetivo, aunque hay otras ecualizaciones posibles...

Podemos probar nuestra función de ecualización con otras imágenes, pero conviene recordar lo que dice la wikipedia: *Histogram equalization often produces unrealistic effects in photographs; however it is very useful for scientific images like thermal, satellite or x-ray images, often the same class of images that user would apply false-color to.*

Prueba con imágenes en la que el efecto pueda ser 'interesante': fotos con poco contraste, radiofrafías, imágenes espaciales...

In [12]:
```
i = Image.open("PIA.png",'r')
i.show()
j = equalization(i)
j.show()
```



Una imagen en color está formada en realidad por tres imágenes en 'blanco y negro', una para la intensidad de rojo, otra para la de verde y otra para la de azul. Para trabajar con imágenes en color, hay que hacer la equalización para cada capa por separado.