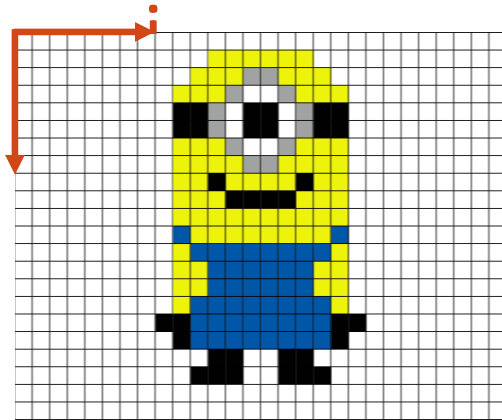


Operadores de pixel

Imagen: matriz de pixeles



- $g(x) = h(f_0(x), \dots, f_n(x))$

Es decir, una imagen de salida $g(x)$ es función $h(f(x))$ de una o más imágenes de entrada $f_i(x)$

- x : Dominio D-dimensional
- En imágenes discretas $x = (i, j)$

- $g(i, j) = h(f(i, j))$

Transformación a nivel de pixel: $g(x)$ es una transformación h de una imagen o hasta n imágenes. Convierte una o mas imágenes para dar una imagen de salida

Un procesamiento común es: $g(x) = a \cdot f(x) + b$

Donde:

- $a > 0$ es la ganancia y controla el contraste
- b es el bias y controla el brillo

También podría ser: $g(x) = a \cdot f(x) + b(x)$

Ejemplo: Si $a=1$ y $b=10 \Rightarrow$ agarramos la imagen original y le sumamos a todos los pixeles por igual un valor de 10, es decir, que a todos los pixeles les aumento el brillo en 10. El histograma no cambio de forma, pero si subió en altura.

Si $b=0$ y $a=2 \Rightarrow$ Se ensancha el histograma. Estoy cambiando el contraste, los espacios oscuros quedan oscuros y los espacios claros quedan mas claros. Tengo mas diferencias entre oscuros y claros.

- Los operadores de píxel lineales admiten superposición:

$$h(f_0 + f_1) = h(f_0) + h(f_1)$$

- Otra aplicación (transición entre dos imágenes):

$$g(x) = (1 - \alpha) \cdot f_0(x) + \alpha \cdot f_1(x) \quad 0 < \alpha < 1$$

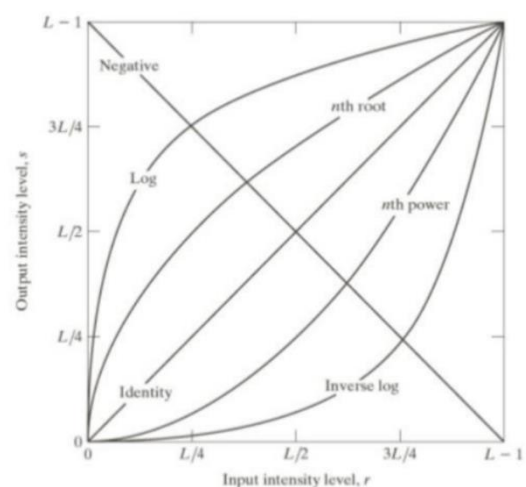
- Las transformaciones de píxel pueden ser también no lineales

$$g(x) = (f(x))^{1/\gamma}$$

Esta transformación es la conocida corrección de gamma (relación entre el valor numérico del píxel y su luminancia real, como la perciben los humanos).

[Ver este link](#)

El valor de γ típico suele ser de 2.2



Coordenadas cromáticas

Transformación de imágenes color

1. Sumar un bias a cada canal no solo modifica el brillo sino el matiz y la saturación (Solución: calcular las coordenadas cromáticas y luego manipular la luminancia Y para recalculer a una imagen con igual matiz y saturación)
2. Balance de color (por ejemplo, compensar por fuente de luz incandescente) se realiza multiplicando cada canal por un factor de escala diferente.

Para un pixel i:

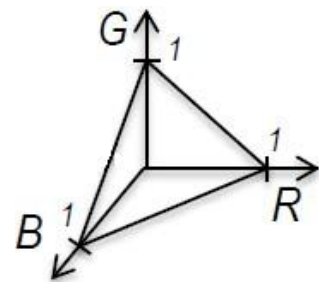
$$(R, G, B) \rightarrow \left(\frac{R}{R+G+B}; \frac{G}{R+G+B}; \frac{B}{R+G+B} \right)$$

Entonces si $s \in \mathcal{R}$ y $s \cdot (R, G, B) \rightarrow \left(\frac{sR}{sR+sG+sB}; \frac{sG}{sR+sG+sB}; \frac{sB}{sR+sG+sB} \right)$

Una imagen que haya sido convertida a sus coordenadas cromáticas es
“Descriptor invariante a los cambios de contraste”

Este nuevo descriptor es de dimensión 2, es decir,
la **proyección sobre el plano $R+G+B=1$**

▪ **Ec. Plano:** $\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1$
 $a = b = c = (X + Y + Z)$



Algoritmo White-Patch

Dependiendo el color de la fuente de iluminación tendremos variaciones en los colores reflejados.

- ¿Cómo hacemos para tratar de tener un sistema menos susceptible a esto?

Asumimos que los valores máximos de color en los tres canales es el color del blanco bajo la luz de la escena.

- $R_{max}(Img)$: Valor máximo del canal R de la imagen
- $G_{max}(Img)$: Valor máximo del canal G de la imagen
- $B_{max}(Img)$: Valor máximo del canal B de la imagen

Normalizamos entonces al color de la luz blanca puro (255, 255, 255)

- $(R, G, B) \rightarrow \left(\frac{255}{R_{max}(I)} R, \frac{255}{G_{max}(I)} G, \frac{255}{B_{max}(I)} B \right)$

Histogramas

Un histograma nos dice cómo es que los valores de intensidad están distribuidos en una imagen
Es decir, cuantos pixeles tienen un determinado valor de intensidad.

- El rango tonal se refiere a la region donde hay más valores de intensidad presents.

cv.calcHist([img],[0],mask,[256],[0,256])

Channel (canal): Donde quiero calcular el histograma. De 0 a 2 para imágenes color

Si es RGB => 0:R, 1:G, 2:B

Mask (máscara): Máscara sobre la imagen

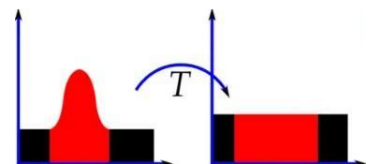


Bins (cajas): Número de subpartes en las que está dividida la escala de la imagen. En el ej: [256]

Range (Rango): Rango sobre el que se quiere calcular. Normalmente todo → [0, 256]

Ecualizacion de histogramas

La idea es “estirar” el histograma para “rellenar” el rango dinámico y a la vez mantener el histograma lo más uniforme posible.



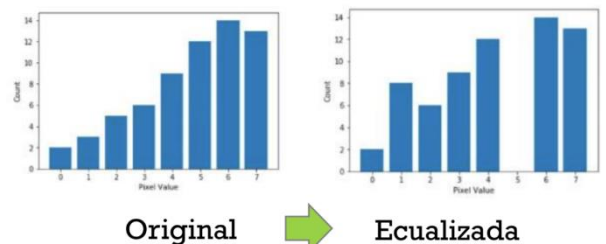
⇒ Con esto logramos una imagen con mejor contraste y gran variedad de niveles de gris

- ¿Cómo lo hacemos? Utilizando la distribución acumulativa (integral del histograma)

- $c_k = T(r_k) = (L - 1) \sum_{i=0}^k p_r(r_i) = \frac{(L-1)}{N} \sum_{i=0}^k h(i)$
- N : número de píxels en la imagen
- $I: [0,255]$ (para una imagen de 8 bits)
- $p_r(r_i) = \frac{n_i}{N}$: Probabilidad de ocurrencia del valor de intensidad r_i en la imagen
- $h(I)$: Histograma original de la imagen

función: ▪ `imagen_ecualizada = cv2.equalizeHist(imagen_original)`

r_k	n_k	$p_r(r_k)$	s_k	Redondeo
0	2	0,03	0,21	0
1	3	0,05	0,56	1
2	5	0,08	1,12	1
3	6	0,09	1,75	2
4	9	0,14	2,73	3
5	12	0,19	4,06	4
6	14	0,22	5,60	6
7	13	0,20	7,00	7



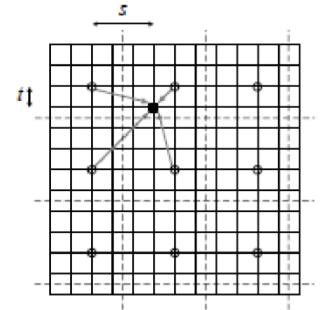
Adaptación local

A veces una ecualización completa no es conveniente → podemos usar bloques de $M \times M$

- Si los bloques son fijos se generan artefactos visuales (discontinuidades de intensidad en los bordes de los bloques)
- Si se hace un bloque deslizante de a un píxel... c /píxel serían M^2 operaciones! (se podría reducir a M reutilizando los píxeles que entran y salen de la ventana)

SOLUCIÓN: AHE (Adaptive Histogram Equalization)

- Bloque fijos + interpolación suave de las funciones transferencia al moverse entre bloques.



CLAHE (Contrast Limited Adaptive Histogram Equalization)

- Resuelve el problema de bloques más o menos uniformes, con algún ruido, que tienden a amplificar el ruido.
- La idea es limitar la pendiente de la CDF (Cumulative Distribution Function) típica de estos bloques recortando el histograma.

Transformación color a grises

1. Promediado (método más común)

$$I_{gris}(i,j) = \frac{R(i,j) + G(i,j) + B(i,j)}{3}$$

2. Brillo (Reducción de contraste)

$$I_{gris}(i,j) = \frac{\max(R(i,j); G(i,j); B(i,j)) - \min(R(i,j); G(i,j); B(i,j))}{2}$$

3. Luminosidad (Buena performance. Usado por GIMP)

$$I_{gris}(i,j) = 0,21R(i,j) + 0,72G(i,j) + 0,07B(i,j)$$

Binarización

▪ Métodos globales

1. Umbral fijo

$$g(x) = \begin{cases} 1 & \text{si } f(x) \geq T \\ 0 & \text{si } f(x) < T \end{cases}$$

2. Método Kilter e Illingworth

- Aproxima el histograma como una distribución bimodal y encuentra el punto de corte para segmentar la imagen en primer plano o fondo

$$p_{mix}(t) = \alpha \cdot p_B(t) + (1 - \alpha) \cdot p_f(t)$$

t : Threshold (segmenta la imagen en dos regiones, background y foreground)

$p_B(t)$ y $p_f(t)$ distribuciones Gaussianas

3. Método Otsu

- Calcula el valor de umbral de forma que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes

■ Métodos locales

1. Mediana

c: por defecto vale 0 $pixel = (pixel > mediana - c) ? objeto : background$

1. Niblack

■ Basado en la media y varianza local sobre una ventana de tamaño bxb alrededor del píxel

$pixel = (pixel > media + k * desvio_std - c) ? objeto : background$
k: por defecto 0.2 para objetos claros, -0.2 para objetos oscuros
c: por defecto (algoritmo original) vale 0

2. Sauvola

■ Variación del método anterior

$pixel = (pixel > media * (1 + k * (desvio_std/r - 1))) ? objeto : background$
k: por defecto 0.5
r: por defecto 128

3. Bernsen

■ El método original usa una ventana circular. Precisa un parámetro de contraste de entrada.
if(contraste local < contraste ingresado)

$pixel = (gris_medio \geq 128) ? objeto : background$
else
 $pixel = (pixel \geq gris_medio) ? objeto : background$

4. Más..

Binarización de Otsu

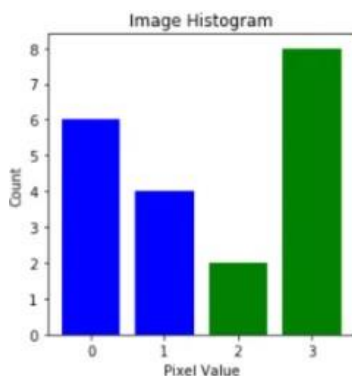
Método de Otsu (creado por Nobuyuki Otsu)

- Presupone que el histograma es bimodal y hay una relación de contraste razonable entre fondo y objetos.
- Busca un valor de umbral que minimice la varianza ponderada dentro de la clase a la vez que maximice la varianza entre clases
- Digamos que limitamos un histograma a un valor "t". Entonces tendremos dos regiones a izquierda y derecha de "t" con varianzas σ_0^2 y σ_1^2 . Entonces la varianza ponderada vendrá dada por:

$$\sigma_w^2(t) = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t)$$

$w_0(t)$ y $w_1(t)$ son los pesos dados a cada clase (proporción de píxeles totales en la clase respecto de los píxeles totales en la imagen). El problema se reduce a encontrar el mínimo de $\sigma_w^2(t)$. Esto es equivalente a encontrar el máximo de la varianza interclúster:

$$\sigma_B^2(t) = w_0(t)w_1(t)(\mu_0(t) - \mu_1(t))^2$$



$$w_0 = \frac{pix_región}{pix_totales} = \frac{6 + 4}{20} = 0.5$$

$$\mu_0 = \frac{Suma\ pesada\ int.}{pix_región} = \frac{(0 * 6) + (1 * 4)}{10} = 0.4$$

$$\sigma_0^2 = \frac{((0 - 0.4)^2 * 6) + ((1 - 0.4)^2 * 4)}{10} = 0.24$$

$$w_1 = \frac{2 + 8}{20} = 0.5$$

$$\mu_1 = \frac{(2 * 2) + (3 * 8)}{10} = 2.8$$

$$\sigma_1^2 = \frac{((2 - 2.8)^2 * 2) + ((3 - 2.8)^2 * 8)}{10} = 0.16$$

$$\sigma_w^2 = w_0\sigma_0^2 + w_1\sigma_1^2 = 0.5 * 0.24 + 0.5 * 0.16 = 0.2$$

LBP – Textura

LBP: Local Binary Patterns

1. Invariante frente a cambios monotónicos del nivel de gris (brillo)
2. Invariante a translación

$$LBP = \sum_{p=0}^{p-1} s(g_p - g_c) 2^p ; \quad s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

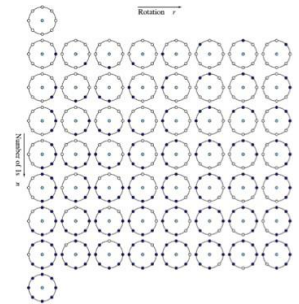
LBP – Variantes

- Definición de vecindad -> Determinada por el radio: R=1, R=2, R=3, etc.
- Comparación con umbral (t)

$$LBP = \sum_{p=0}^{p-1} s(g_p - g_c) 2^p ; \quad s(x) = \begin{cases} 1 & x \geq t \\ 0 & x < t \end{cases}$$

LBP-U (uniformes)

- Objetivo: Reducir el número de patrones
 - U: Medida de uniformidad – Nº de transiciones $0 \leftrightarrow 1$
1. A los patrones con U=0 o U=2 les asignamos un código de patrón individual
 2. Al resto de los patrones se les asigna el mismo código (no son distinguibles)
 3. LBP: 256 patrones, LBP-U: (58+1) patrones



Encontrar a Superman

Si nos quedamos con los LBP

1. tenemos NxM características
 2. Comparando punto a punto el alineamiento de la ventana sería crítico!
- Entonces? => ▪ Hacemos un histograma! ▪ Y si usamos LBP-U...pasamos de 3200 a 59 características!!

Problemas con las ventanas: Detecciones múltiples

Solución:

- Dividir en subventanas: LBP - U por bloques
- Cada histograma debe coincidir en forma y posición.
- En la práctica además de la subdivisión hay solape → caso Superman:

Nº carac.: $(16 + 9) * 59 = 1475$

Nº carac. a impl. : $(4 + 1) * 59 = 295$