

Análisis de Imágenes y Vídeos

AIV



Tema 1. Introducción

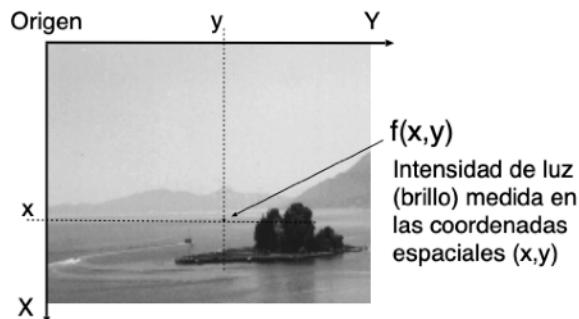
Definición de imagen

Las imágenes constituyen el soporte de uno de los medios más importantes del sistema perceptivo humano. Innumerables aplicaciones tales como el reconocimiento óptimo de caracteres (OCR), reconocimiento de huellas dactilares, identificación de rostros, reconocimiento de elementos en imágenes biomédicas, teledetección, análisis de escenas, robótica, ...

Dentro de cada una de estas aplicaciones pueden surgir problemas específicos asociados a la gran cantidad de información que en ocasiones hay que manejar. El procesado de imágenes tiene interés por se, con independencia de su uso para el reconocimiento.

A continuación, se definirán ciertos términos importantes:

- **Imagen:** Función $f(x, y)$ que representa el *brillo* en cada punto de coordenadas (x, y) .
- **Imagen Digital:** $f(x, y)$ discretizada en su dominio (coordenadas) y rango (brillo).
- **Muestreo:** Discretización del dominio.
- **Cuantificación:** Discretización del rango.
- **Resolución espacial:** píxeles/pulgada.
- **Niveles o colores:** bits/píxel.

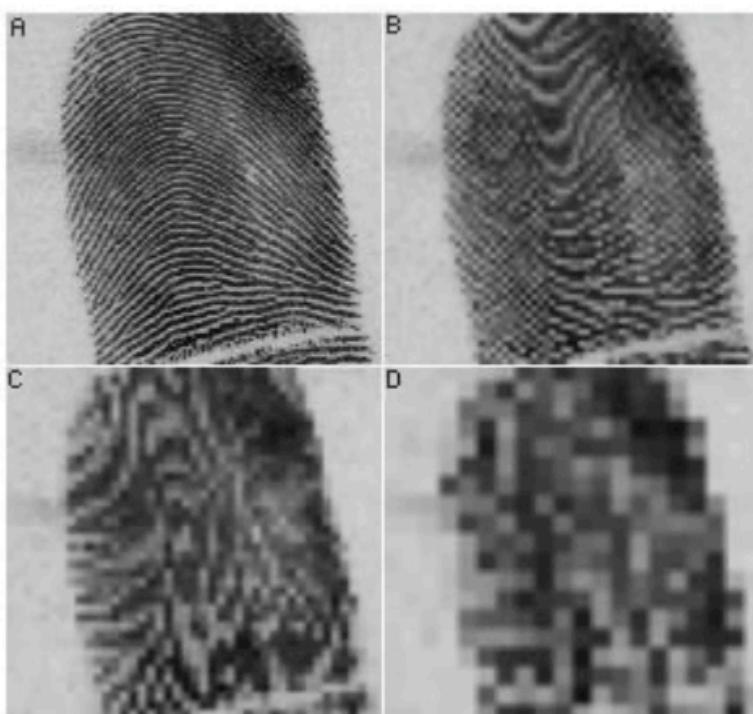


La resolución espacial tiene ciertos límites. Conforme se decremente la relación píxeles/pulgada (frecuencias espaciales) se pierde calidad e información en la imagen.

Imagen original e imágenes muestreadas con *frecuencias espaciales o resoluciones de 128, 64 y 32 puntos por pulgada (ppp)*:



En principio, a medida que se decremente su frecuencia espacial, la imagen ocupará menos espacio en disco. Es decir, es una forma de compresión (muy ineficiente, pero la es). Con la pérdida de calidad de las imágenes pueden darse factores extraños, como el **aliasing**.



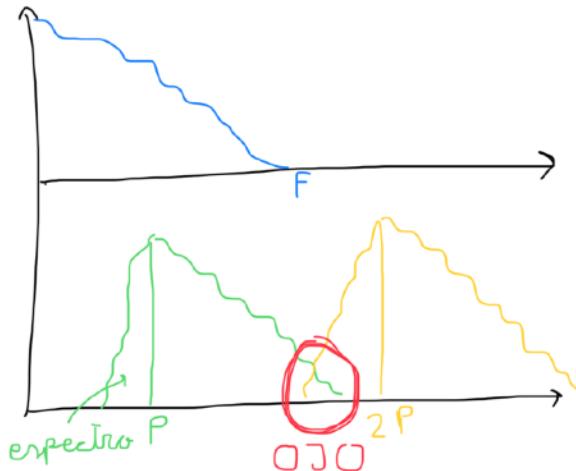
Tal y como ocurre en esta imagen con resoluciones de 140, 70, 44 y 22ppp; conforme decrece esta relación, menor es la calidad de la imagen y será más difícil identificar los patrones o detalles de la imagen original. Pero ocurre algo extraño, ¿qué ocurre en la imagen B con respecto a A? En principio, esta segunda imagen podría ser considerada una huella dactilar

distinta, pero en realidad no lo es. Es un problema que ocurre a la hora de reducir los ppp, que crea patrones no-reales o ficticios de la imagen original.

Ahora es donde entra la **Frecuencia de Nyquist** (Teorema del Muestreo), que trata de determinar una serie de condiciones para que no suceda este hecho. Dice así:

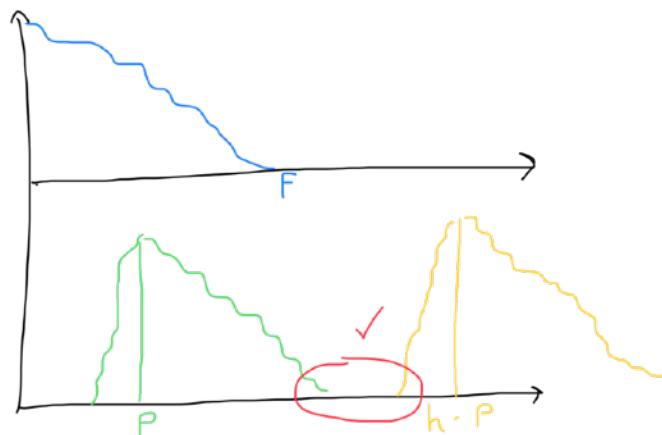
“Si P es la frecuencia espacial de la periodicidad más fina en una imagen y F es la resolución de muestreo, la imagen sólo podrá ser fielmente reproducida si: $F > 2P$ ”

Por ejemplo:



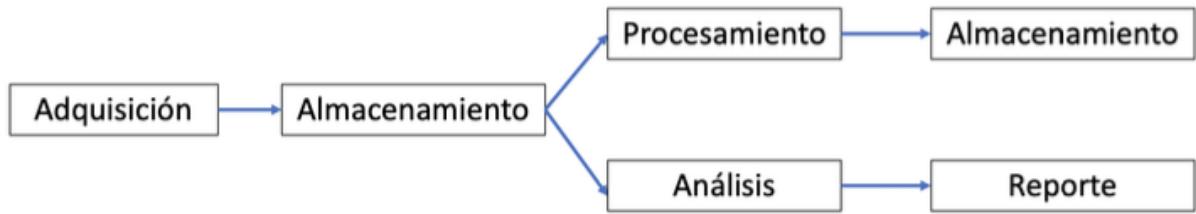
En este caso observamos que, al representar una nueva P en dentro de $F > 2P$, se produce un intercambio o mezcla de frecuencias entre P y el espectro de $2P$. Este intercambio o mezcla es el que se ve representada en la figura B anterior, en la información mostrada puede llevar a confusión.

Si se aplicase este teorema correctamente (siendo $h > 2$), tendríamos:



A grandes rasgos, la imagen es un array (2D o 3D, dependiendo de si es escala de grises o en color).

Las imágenes tienen ciclo de vida. Normalmente una imagen se adquiere, se almacena, se procesa o analiza y opcionalmente se vuelve a almacenar. Dicho procesamiento puede consistir en una mejora de la imagen o en el análisis y obtención de ciertos hallazgos en la misma. **La adquisición y el almacenamiento son potenciales procesos** donde puede haber pérdida de información.



Procesamiento y análisis de imágenes

Diferenciamos entre procesamiento y análisis:

- **Procesamiento:**

- Mejora de aspectos visuales (**contraste, ecualización**)
- **Cambios de resolución.**
- **Cambios de cuantización de colores** o niveles de gris.
- **Compresión.**

- **Análisis:**

- Clasificación.
- **Detección de objetos.**
- **Segmentación de objetos.**

Técnicas clásicas

- Umbralización.
- Modificación de brillo o contraste.
- Aplicación de filtros (convoluciones):
 - Filtros de detección de contornos (paso alto).
 - Filtros de eliminación de ruido (paso bajo).
- Técnicas morfológicas:
 - Dilatado morfológico.
 - Erosionado morfológico.
 - Operador de apertura y cierre.
- Componentes conexas.

*En técnicas de *Deep Learning*, se suele emplear el *Data Augmentation*. Sirve para aportar generalización a las redes neuronales.

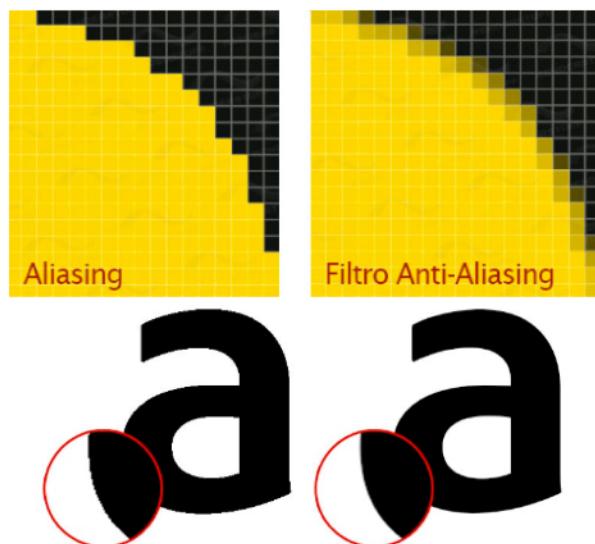
Tema 2.

Representación De Imágenes

Resolución espacial, canales y precisión

Resolución espacial

Ampliando la sección ya comentada en el tema anterior, existe un filtro para evitar el *aliasing*. Este consiste en “difuminar” los bordes de los objetos en la matriz de píxeles. De esta manera, y al muestrearlo en la frecuencia adecuada, no se forman patrones o errores en las imágenes.



Canales

Normalmente, las imágenes tienen 1 o 3 canales (escala de grises y color, respectivamente). Aún así podemos encontrar aplicaciones concretas (multiespectral) donde las imágenes tengan un número diferente de canales.

Esto requiere tener formatos de almacenamiento que soporten diferentes números de canales. Al mismo tiempo requiere visores que soporten diferentes números de canales.

Cuantificación

Se suele emplear 8 bits por canal (256 valores diferentes). En el caso de imágenes RGB serían 24 bits por píxel (8 bits por canal). Aún así podemos encontrar aplicaciones concretas donde se empleen 16 bits por canal o incluso más.

Almacenamiento, formatos y compresión

Como bien es conocido, existen numerosos formatos de almacenamiento de imágenes. Cada uno tiene sus peculiaridades y algoritmos de compresión. A continuación, se presentará los más comunes:

BMP: Bitmap Image File (.bmp)

- Desarrollado por Microsoft para Windows.
- No hay pérdida de información (no quiere decir que no haya compresión).
- Es un formato propietario.
- Ficheros muy grandes en tamaño.
- Puede manejar hasta 16 millones de colores (RGB con 8 bits por canal).

PNG: Portable Network Graphics (.png)

- Sí que tiene compresión, pero sin pérdida de calidad ("lossless compression").
- Ficheros más pequeños que BMP en general.
- Puede manejar hasta 16 millones de colores (RGB con 8 bits por canal).

JPG: Joint Photographic Experts Groups (.jpg o .jpeg)

- Formato más popular.
- Hay pérdida de información ("loosey format").
- El algoritmo de compresión es K-Means.
- La calidad es proporcional al tamaño del fichero, inversamente proporcional a la cantidad de compresión.
- Se suelen conseguir altas calidades con tamaños pequeños.
- Puede manejar hasta 16 millones de colores (RGB con 8 bits por canal).

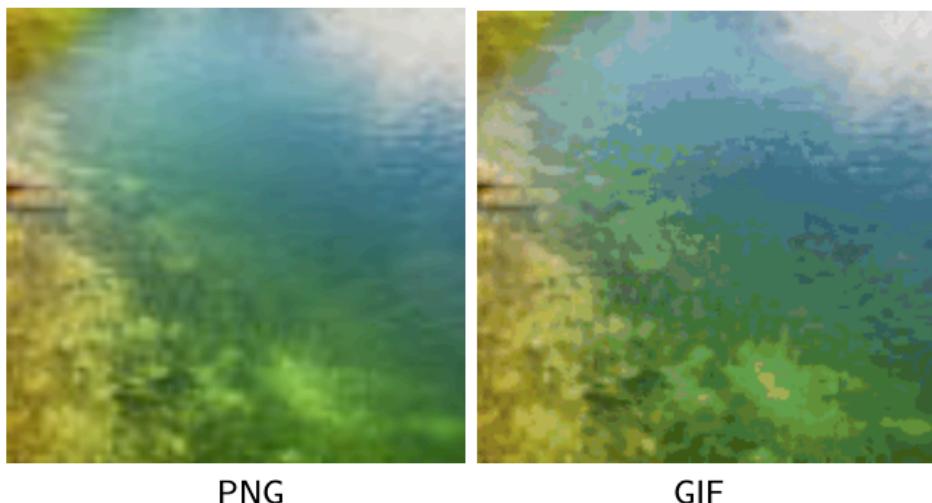
TIFF: Tagged Image File Format (.tif o .tiff)

- Sin compresión, pero tiene opciones de hacerlos.

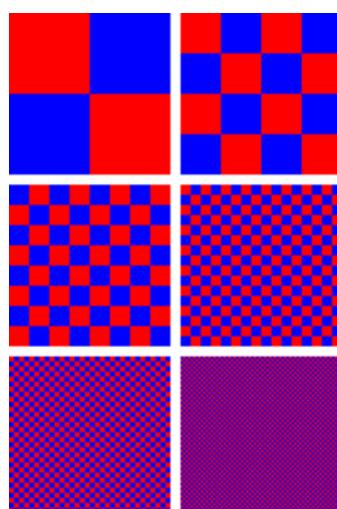
- Ficheros grandes por lo general.
- Puede manejar hasta 16 millones de colores (RGB con 8 bits por canal).

GIF: Graphics Interchange Format (.gif)

- Formato inicialmente pensado para la web.
- **Sin compresión.**
- Solo soporta 256 colores diferentes en una misma imagen.
- Esta paleta de colores se selecciona según la imagen.
- Soporta animaciones.
- Suelen ser pequeños (8 bits por píxel).

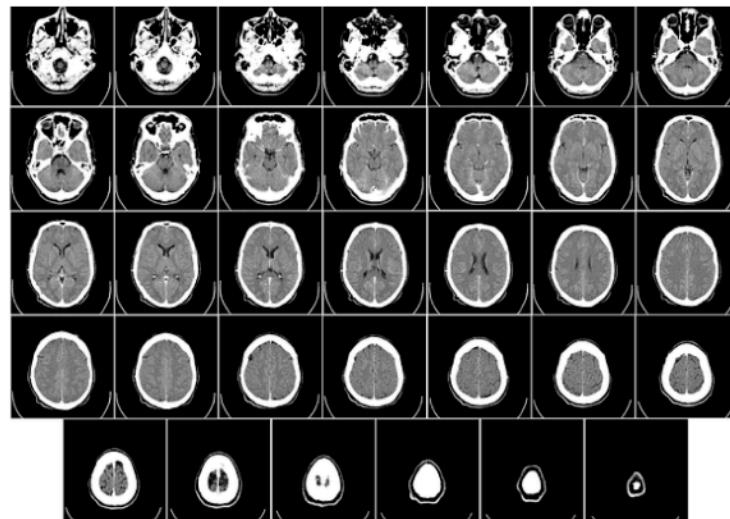


- Existe un “truco” para poder tratar mejor los pocos colores que se tienen en una paleta, se trata del **Dithering**. Consiste en la representación conjunta de varios colores para poder representar mediante un efecto óptico otro color. Un ejemplo claro es tener una paleta de rojos y azules, pero se desea representar el morado; pues bien, mediante la representación seguida de rojo y azul se puede crear un efecto óptico que cree el morado, que no está en la paleta de colores.



Otros

- **DICOM.**
- Imagen médica.
- **Digital Imaging and Communications in Medicine.**
- Es un protocolo estándar de comunicación entre sistemas de información.
- DICOM además **es un formato de almacenamiento de imágenes médicas.**
- Soluciona los **problemas de interoperabilidad** entre tipos de dispositivos.
- **Cada fichero contiene**, además de la imagen, **información sobre el paciente.**
- También almacena **parámetros de adquisición.**
- Imagen aérea, satélite. Hiperespectral.
- Geo Tiff.
- CCRS.
- ...



Tema 3. Técnicas Clásicas

3.1. Mejora de imagen, modificaciones, ecualizado

Técnicas de mejora de imagen

El objetivo de estas técnicas es mejorar la calidad visual de la imagen. Los principales defectos que aparecen en la imagen provienen de la fase de adquisición. Es posible que existan problemas de:

- Balances de blancos y calidad del sensor.
- Bajo rango dinámico.
- Zonas de la imagen con muy bajo contraste.

En general, estas técnicas se basan en una función que transforma el nivel de intensidad (gris o RGB) de cada píxel.

Esta modificación solo depende del nivel de intensidad presente en el píxel en cuestión y no en la vecindad.

$$g(x, y) = T(f(x, y))$$

Donde T es una función de transformación que va del dominio de la intensidad de entrada al dominio de la intensidad de salida. Normalmente $T : [0,255] \rightarrow [0,255]$

Modificación de brillo y contraste

Este tipo de mejoras se basan en la siguiente función de transformación:

$$T(f(x, y)) = \alpha \cdot f(x, y) + \beta$$

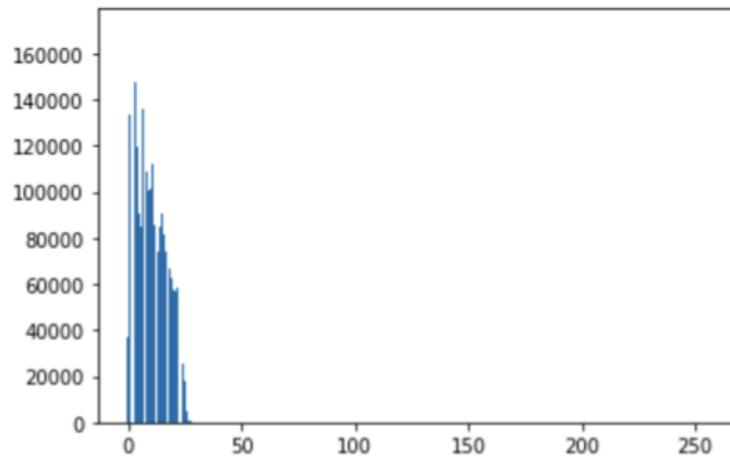
donde α actúa sobre el contraste y β sobre el brillo.

Es muy importante respetar el dominio de valores de salida. Por ello, se realiza la siguiente operación:

$$T(f(x, y)) = \text{clip}(\alpha \cdot f(x, y) + \beta, 0, 255)$$

Además de cuidado con el tipo de dato obtenido (uint8).

Para modificar el brillo y el contraste de una imagen de baja calidad, en primer lugar debe examinar su histograma. El histograma de una imagen representa la frecuencia de aparición de cada uno de los valores del dominio de representación. Es el número de veces que cada uno de los valores de intensidad aparecen en la imagen o cuántos píxeles tienen cada uno de los niveles de intensidad. Por ejemplo, para el siguiente histograma de una imagen:

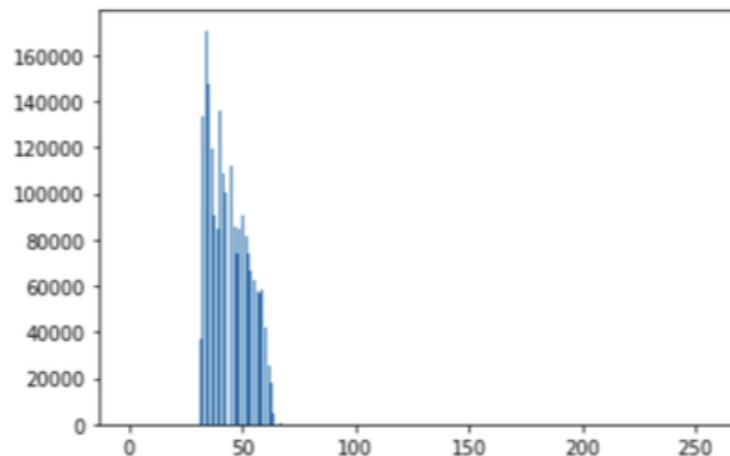


Se observa que la imagen está muy oscura, la totalidad de su representación está en tonalidades oscuras. Para modificarlo, se emplean las técnicas de brillo y contraste.

$$T(f(x, y)) = \text{clip}(\alpha \cdot f(x, y) + \beta, 0, 255)$$

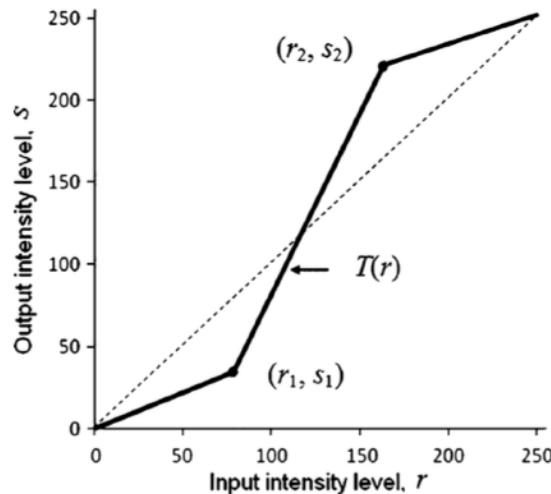
- Brillo: β desplaza hacia la derecha (si es positivo) el histograma.
- Contraste: α ensancha (si es mayor que 1) el histograma.

Aplicando una transformación de $\alpha = 1.25$ y $\beta = 32$, se obtiene el siguiente histograma:



Se han movido notoriamente sus valores, y la imagen se ha hecho más clara (pero no mucho).

Ahora bien, no solo se puede trabajar con un α y un β , también se puede aplicar modificaciones lineales a trozos, o *contrast stretching*.



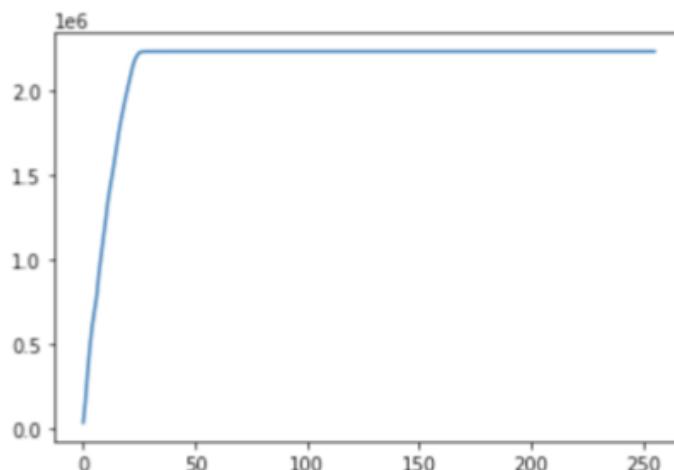
En general podríamos definir cualquier función de transformación T dentro del dominio. E incluso esta función puede estar ya precalculada (recomendable) y almacenar su resultado en un array del tipo $T(g) = H[g]$.

Ecualizado del histograma

La ecualización de histograma permite aprovechar al máximo el rango dinámico. Para ello, se persigue que todos los valores del dominio de representación tengan la misma contribución en la imagen. Esto quiere decir que se persigue obtener una imagen con un histograma plano.

Para ello, se emplea la función de densidad acumulada como función de transformación T debidamente normalizada.

La función de densidad acumulada $F(x)$ representa cuántos píxeles de la imagen tienen un nivel igual o inferior a x .



La función de densidad acumulada es siempre creciente. Alcanza el máximo en el número total de píxeles de la imagen, o sea ancho x alto.

La función de densidad acumulada adecuadamente normalizada es la propia función de transformación:

$$T(I(x, y)) = \frac{255}{(ancho \cdot alto)} \cdot F(I(x, y))$$

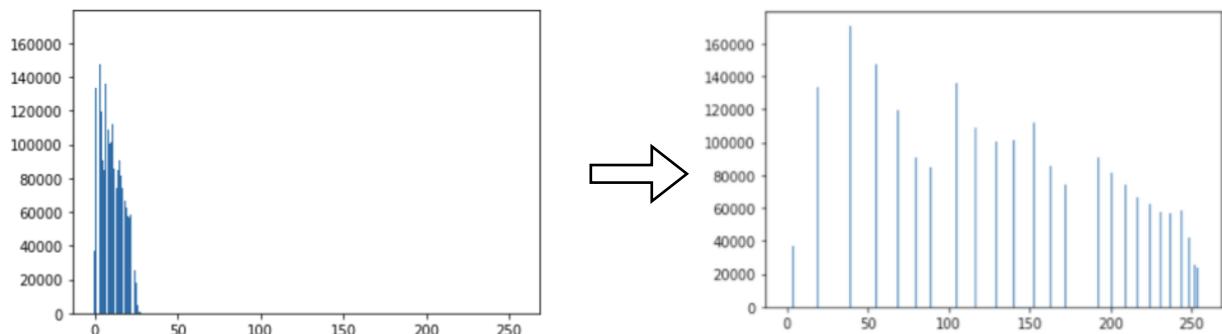
Lo que se suele hacer es precalcular los valores en H:

$$H[g] = \frac{255}{(ancho \cdot alto)} \cdot F(g), g \in [0, 255]$$

$$T(I(x, y)) = H[I(x, y)]$$

Para esta parte, no es necesario emplear ningún elemento α o β de apartados anteriores.

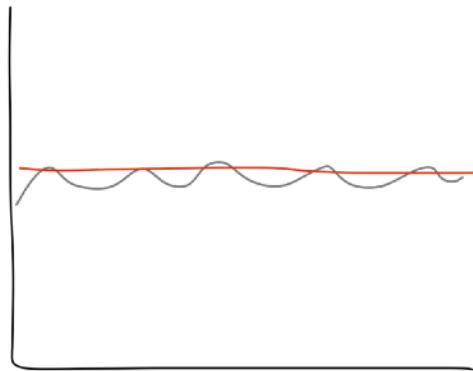
Con el ecualizado, se obtiene el siguiente resultado:



Claro, todo lo explicado hasta ahora sobre ecualización está aplicado a imágenes en escala de grises, ¿qué ocurre cuando tenemos una imagen en color? Un error muy común es ecualizar los canales RGB por separado. Sin embargo, lo que se debe hacer es convertir la imagen RGB a otro espacio de color donde tengamos en un plano la “luminancia”, como LAB.

Una vez en este nuevo espacio de color, se debe ecualizar sólo la componente de luminaria, en este caso L. Después de vuelve al espacio RGB.

¿Cuál sería el ecualizado “perfecto”? Pues sería algo parecido al histograma de abajo. La línea roja sería el histograma ecualizado, y la línea gris la imagen inicial.

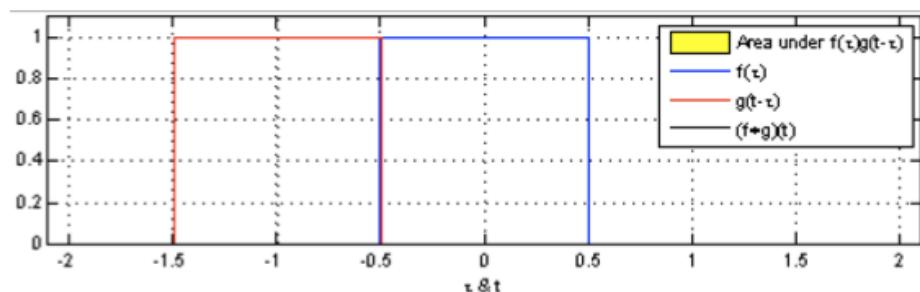
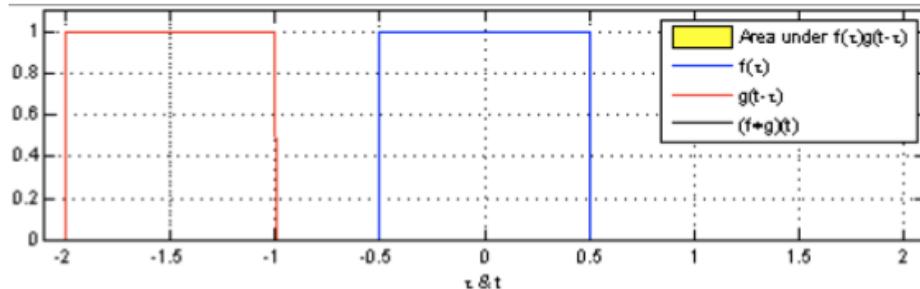


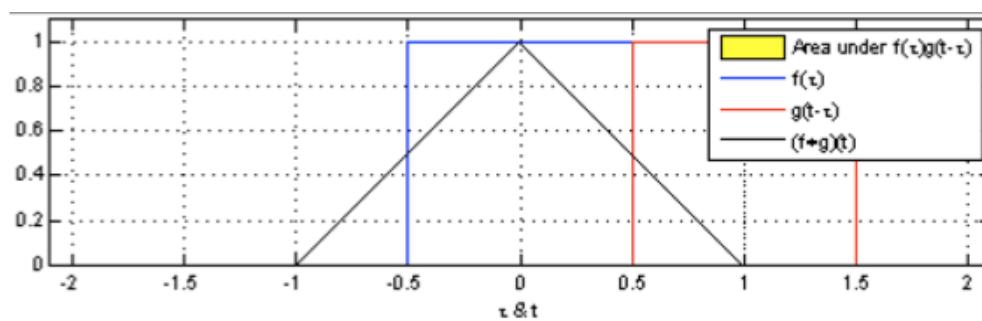
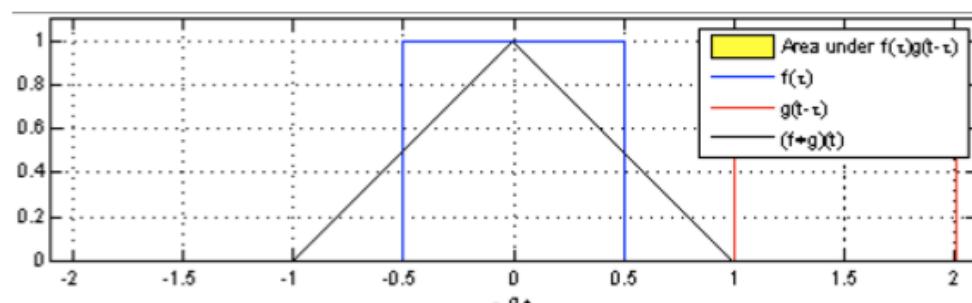
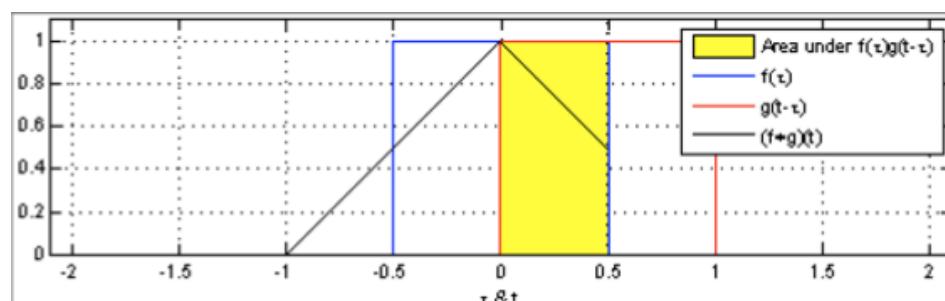
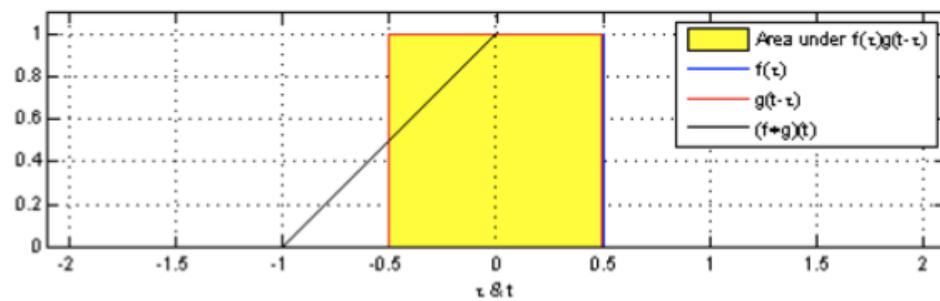
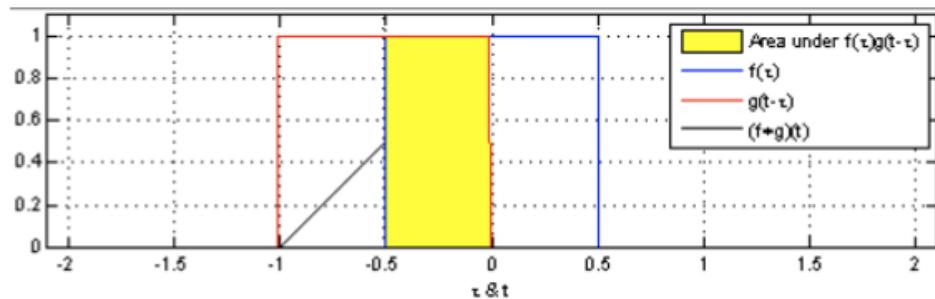
3.2. Convolución, filtros de ruido y contornos

Convolución

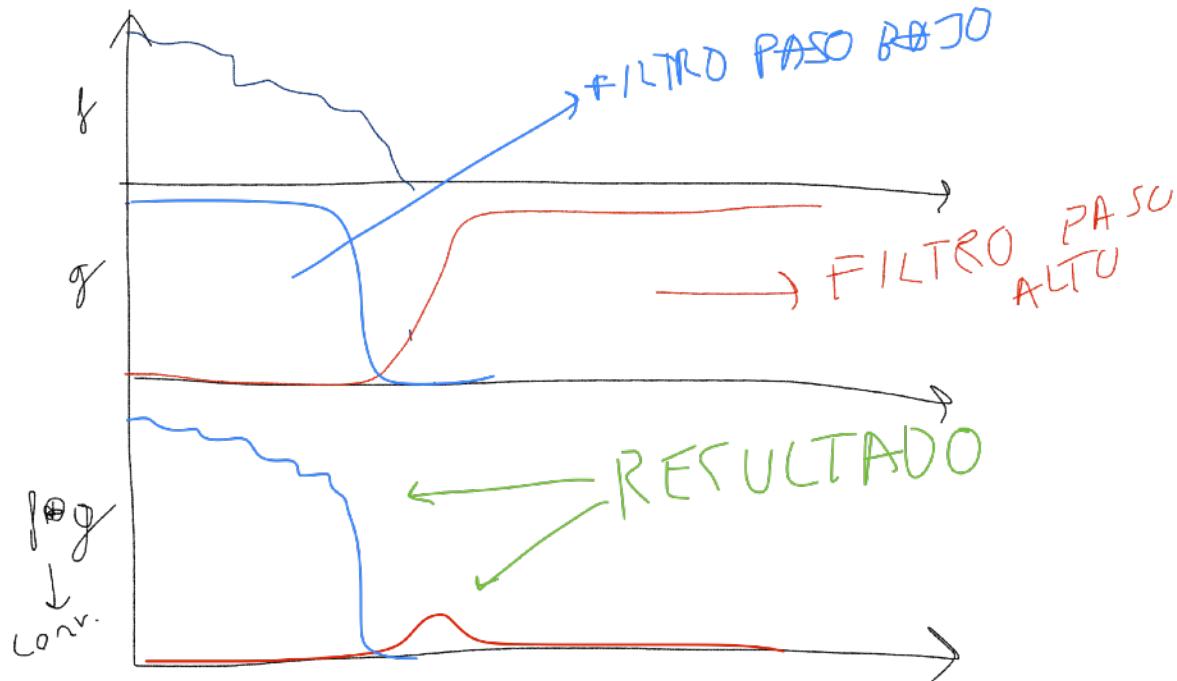
Dadas dos funciones (dos señales) f y g , definimos la **convolución** como:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t - \nu)g(\nu) d\nu$$



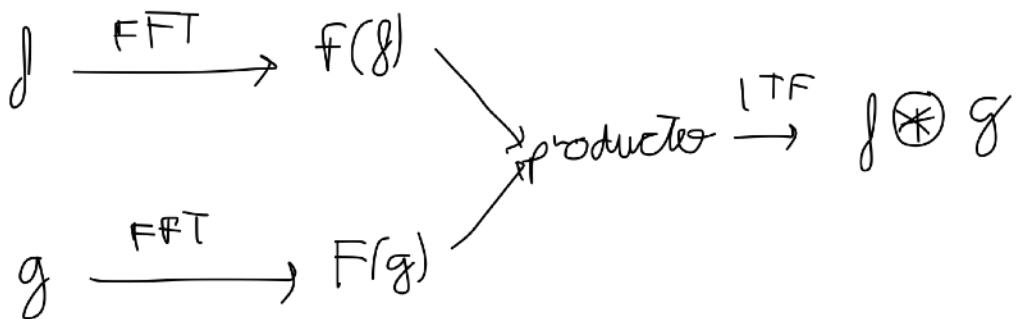


Una vez visto de qué trata de obtener la convolución, se comentará los diferentes tipos de filtros. Previamente cabe destacar que f es las frecuencias de la imagen original, g es el **filtro** o **kernel**, y la convolución es el resultado de aplicar el filtro a f .



¿Cuál es la relación entre la convolución y la transformada de Fourier? **La transformada de Fourier de una convolución es el producto de las transformadas de Fourier:**

$$F(f \circledast g) = F(f) \cdot F(g)$$



Se pueden **eliminar ciertas frecuencias** que aparecen en f convolucionando con una función g que no las tenga, así el resultado no las tendrá.

Existen mayoritariamente dos tipos de filtros:

- Filtro **paso alto**. Se corresponde al filtro **rojo** de la imagen anterior. **Su función es filtrar las frecuencias bajas y dejar únicamente las altas.** En el resultado final, se aprecia que el filtro ha erradicado las frecuencias bajas y medias, dejando únicamente las altas.
- Filtro **paso bajo**. Se corresponde al filtro **azul** de la imagen anterior. **Su función es filtrar las frecuencias altas y dejar únicamente las bajas.** En el resultado final, se aprecia que el filtro ha erradicado las frecuencias altas, dejando únicamente las bajas y medias.

Convolución discreta

La convolución de señales discretas es:

$$(f \circledast g)[n] = \sum_{k=-\infty}^{+\infty} f[n-k] \cdot g[k]$$

Donde además tenemos la misma relación con la transformada *discreta* de Fourier:

$$DF(f \circledast g) = DF(f) \cdot DF(g)$$

De esta manera, se puede observar de nuevo que g es un filtro que se puede aplicar a una señal discreta.

Convolución discreta en 2D

Es posible extender la convolución a N dimensiones. En concreto para señales discretas de dos dimensiones tendríamos:

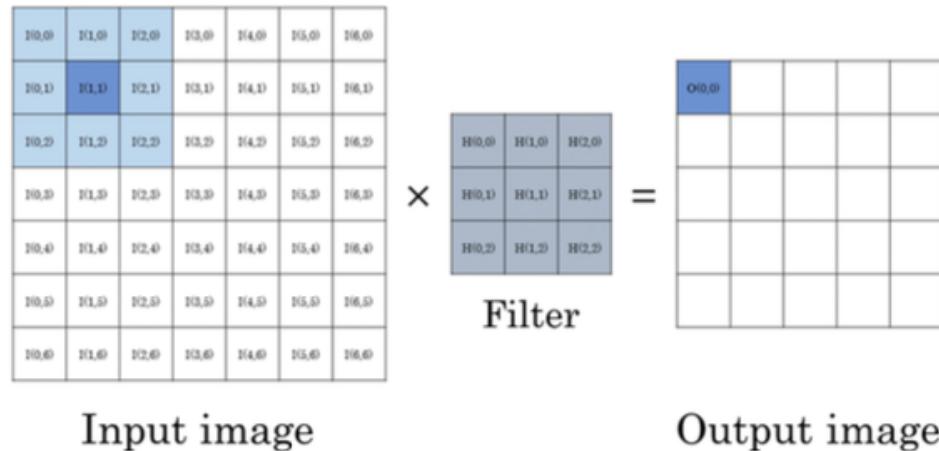
$$(f \circledast g)[n, m] = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} f[n-j][m-k] \cdot g[j][k]$$

En este caso, f y g son imágenes discretas (arrays). Usualmente, g es una imagen pequeña que actúa como filtro.

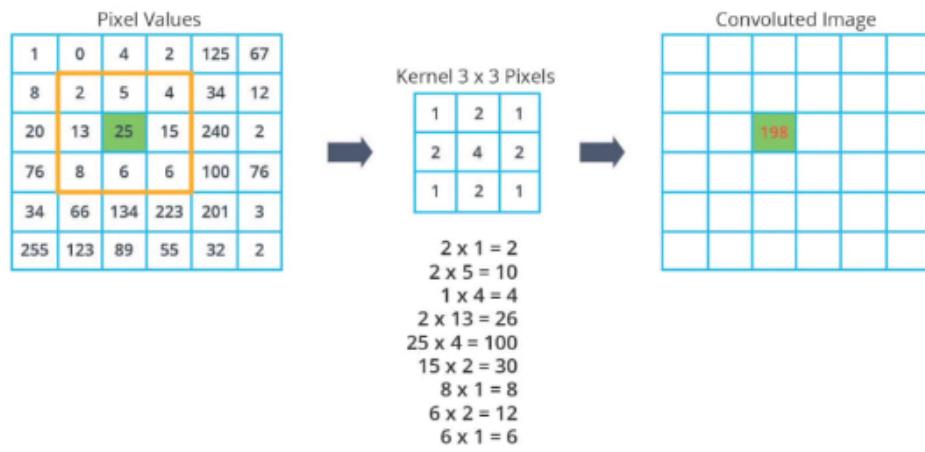
Si, por ejemplo, g fuese de tamaño 3x3, la función anterior quedaría tal que así:

$$(f \circledast g)[n, m] = \sum_{j=-1}^1 \sum_{k=-1}^1 f[n-j][m-k] \cdot g[j][k]$$

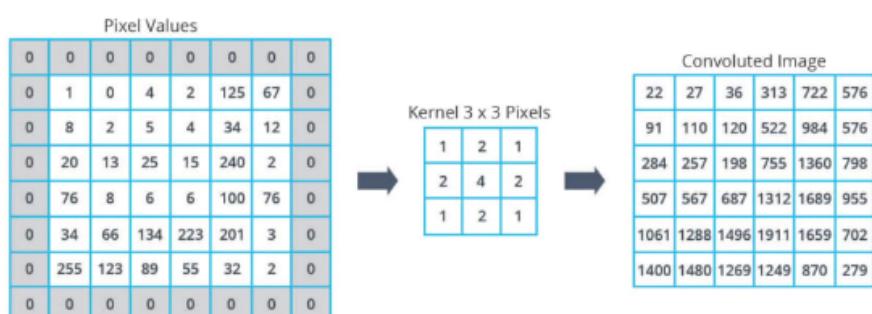
El sumatorio va de -1 hasta 1 porque el centro de la convolución se considera el centro del filtro 3x3.



Un ejemplo de una convolución normal quedaría tal que así:



En todo caso, se obtiene una imagen de salida menor que la de entrada. Para que esto no ocurra, si se desea, se puede **ampliar la imagen de entrada con ceros en todos sus bordes**. De esta manera, al aplicarle la convolución, el tamaño de la de salida será igual que el tamaño de la de entrada antes de añadirle los ceros. A este proceso se le conoce como **padding**.



Sin la aplicación de *padding*, ¿cuál será el tamaño de la imagen de salida? Tomando una imagen de tamaño $n \times m$, y un filtro $k \times j$; el tamaño sera:

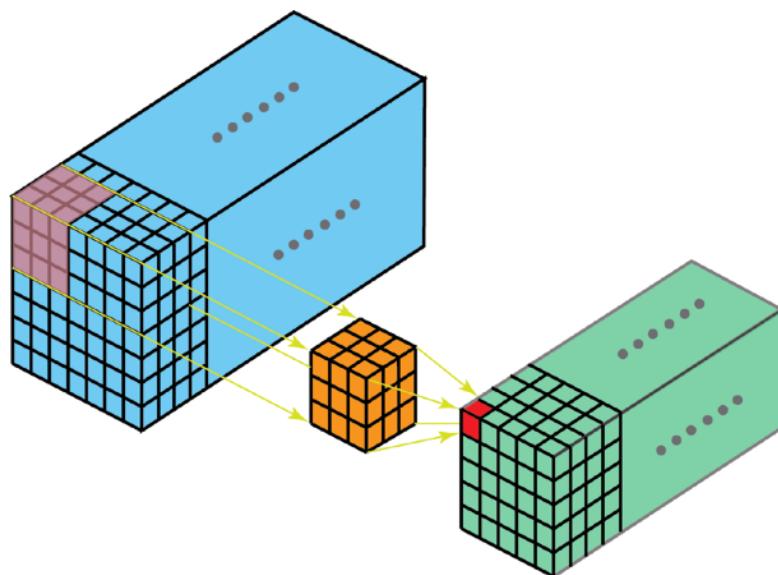
$$n \times m \otimes k \times j = (n - k + 1) \times (m - j + 1)$$

Y otra cosa, ¿cuántas multiplicaciones serán necesarias para realizar la convolución?

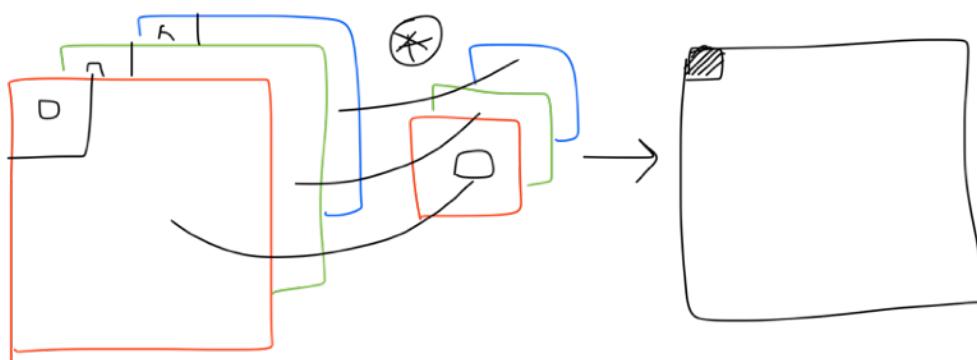
$$O(n \cdot m \cdot k \cdot j)$$

Convolución en 3D

De manera similar a la anterior, la convolución en 3D es:



Para un ejemplo sencillo en una imagen en 3D, la convolución queda de la siguiente forma:



$$R \cdot filtroR + G \cdot filtroG + B \cdot filtroB = posicionImagenSalida$$

Filtros de Reducción de Ruido

También son conocidos como filtros de desenfoque (*blurring*).

El más sencillo es el filtro de media:

$$g = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

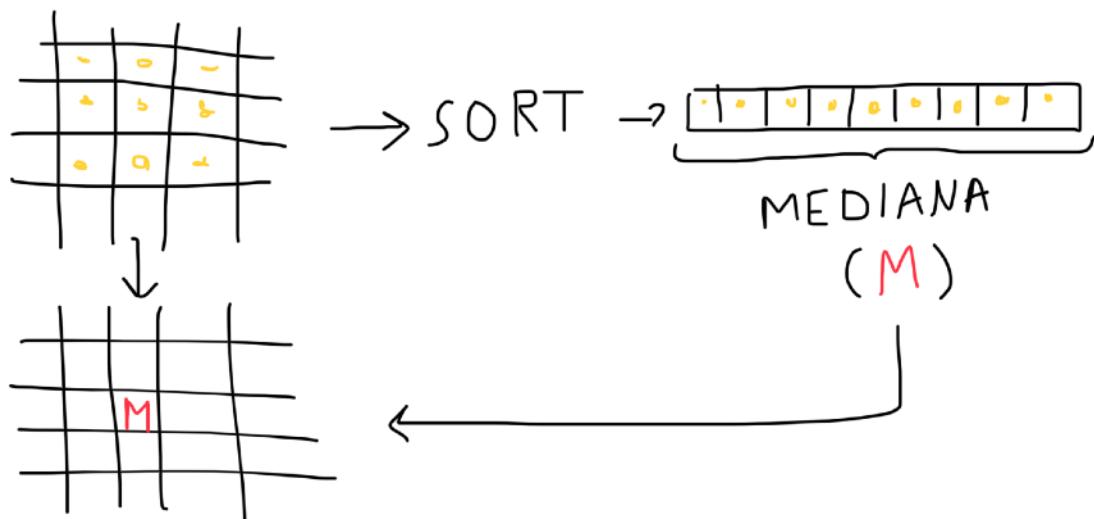
Como se ha visto, para conservar el nivel de la señal habría que dividir por la suma total del kernel. Este filtro es más adecuado para ruido uniforme, aunque quita contornos y emborrona, también quita ruido.

Pero, por ejemplo en imágenes con ruido binario o *sal y pimienta* este filtro no es del todo efectivo. En ese caso, el filtro efectivo es el de *mediana 3x3*:

$$\text{salida}[n, m] = \text{Median}(\text{crop}(f, n - 1, m - 1, n + 1, m + 1))$$

La gran ventaja de estos filtros es que se pueden aplicar una y otra vez hasta obtener el resultado deseado.

Una explicación gráfica mejor del filtro de mediana es:



Filtros de Detección de Contornos

Los contornos pertenecen a las frecuencias altas, ya que los cambios repentinos de formas son muestreos por este tipo de frecuencia. Por tanto, estos filtros serán paso alto. Además, se pueden obtener a partir de los paso bajo, ya que los contornos es lo que queda cuando se eliminan las frecuencias bajas. Entonces, se puede implementar un filtro paso alto restando a la imagen original el resultado de un filtro paso bajo:

$$G = f - PB(f)$$

Pero también se pueden implementar con sus propios kernels.

Existen varios tipos de filtros de detección de contornos, como:

Filtro de Sobel

$$g_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \quad g_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$G = \sqrt{g_x^2 + g_y^2}$$

Filtro de Laplace

$$g_x = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Filtro de Prewit

$$g_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}; \quad g_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$G = \sqrt{g_x^2 + g_y^2}$$

Filtro de Roberts

$$g_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}; \quad g_y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$G = \sqrt{g_x^2 + g_y^2}$$

3.3. Segmentación, Umbralización, Componentes Conexas

Segmentación de Imágenes

La segmentación de imágenes consiste en el particionado de la imagen en diferentes regiones que corresponden a distintos objetos o clase de objetos.

En su versión más clásica, la segmentación se fundamenta en la idea que dos píxeles con valor de intensidad similares y conectados pertenecen al mismo objeto.

Por lo tanto, debemos ser capaces de aislar píxeles según su intensidad (umbralización), así como ser capaces de conectarlos para formar objetos (componentes conexas).

Un ejemplo de segmentación es el siguiente:



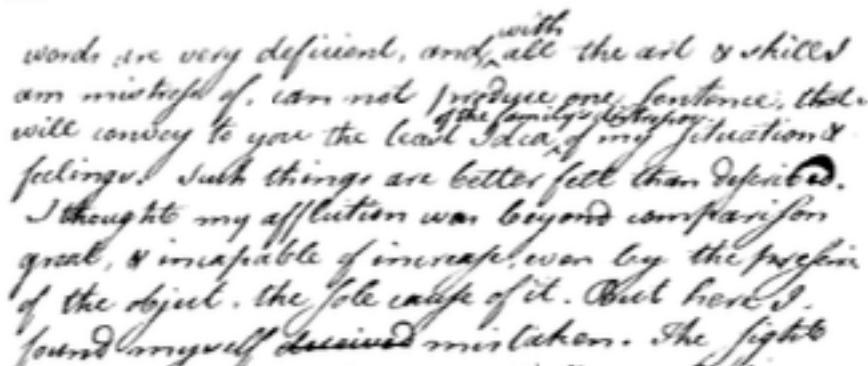
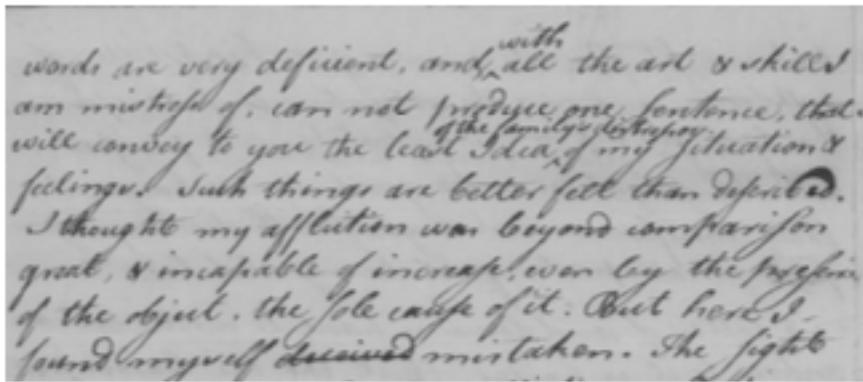
Umbralización

Su objetivo es determinar en la imagen aquellos píxeles que cumplen cierto criterio relacionado con la intensidad. Normalmente, se obtiene una imagen binaria (**máscara**), donde tenemos un 1 en los píxeles que cumplen el criterio establecido y un 0 en los píxeles que no.

El resultado puede ser directamente la máscara binaria (que se suele utilizar para procesos posteriores). También puede ser una combinación de la imagen original y la máscara.

La umbralización puede aplicarse, a parte de a la imagen original, a aquellas imágenes resultantes de algún proceso de mejora de calidad o brillo: ecualización, filtros...

Un ejemplo es el siguiente:



Ahora bien, ¿qué hay que utilizar para poder llevar a cabo la umbralización? Se utiliza el **algoritmo de Otsu**. Este método trata de encontrar un umbral que minimice la varianza intra-clase que recae a cada uno de los lados de dicho umbral. Se aplica en el histograma de una imagen y recorre todos los posibles valores ($t \in [0, 255]$).

$$d = P_0(t) \cdot \sigma_0^2(t) + P_1(t) \cdot \sigma_1^2(t)$$

- P_0 es la cantidad relativa de píxeles por debajo del umbral y P_1 es la cantidad relativa de píxeles por encima de él. $P_0 + P_1 = 1$. Se utilizan los píxeles a cada lado para ponderar por posición el cálculo de las varianzas.
- σ_o^2 y σ_1^2 son la varianza de los píxeles por debajo y por encima del umbral respectivamente.¹

$$\sigma^2 = \frac{1}{N} \cdot \sum_x (X - X_m)^2$$

¹ Cuidado con divisiones entre 0 y 1 muestras únicamente. Evitar.

Otsu devuelve un valor, que será el umbral “perfecto”.

Una aplicación del algoritmo queda así:

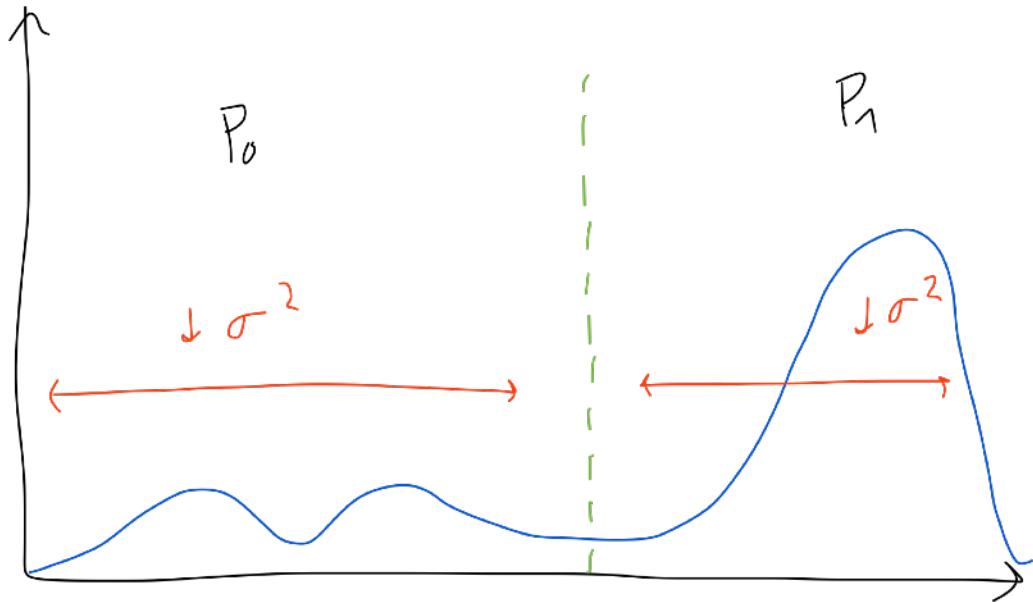
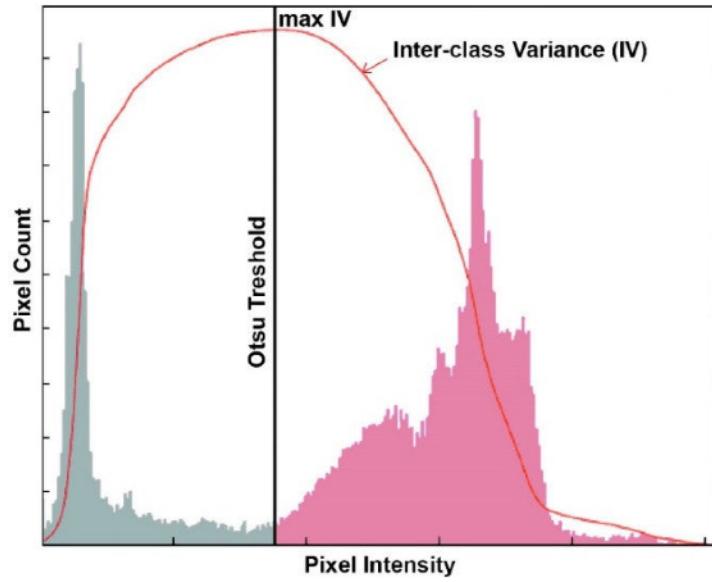
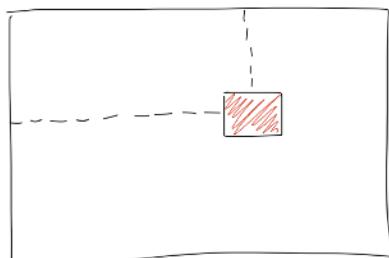


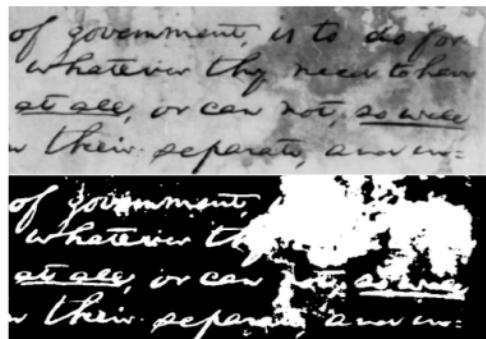
Imagen Integral

Se utilizan para sacar la media y la varianza de forma optimizada. Para el píxel seleccionado, este tendrá la suma de todos los niveles de gris anteriores a él.



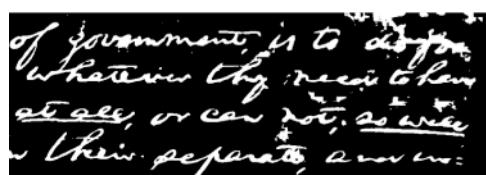
Umbralización adaptativa

Es muy posible que aplicando *Otsu* a nivel global existan manchas o imperfecciones que perjudiquen al resultado del valor del umbral. Un ejemplo es este:



Para evitar este problema, se aplica *Otsu* a nivel local. Esto conlleva que se defina un tamaño de ventana de imagen y se recorra toda la imagen en función de esos tamaños. El solapado de las ventanas conduce al uso de estrategias de votación apropiadas.

Como resultado:



Detección

La detección se puede interpretar como la siguiente operación:

Umbralización + CompConexas + CajaMinimaInclusion

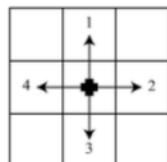


Componentes Conexas

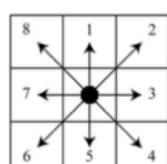
Tiene por objetivo **detectar y numerar objetos aislados** (no conectados) **en la imagen binaria**.

Un objeto puede definirse de varias formas. Se estudiará el **4-conectadas** y el **8-conectadas**:

- **4-conectadas.** Se trata de identificar a un objeto/componente a partir de los vecinos arriba, abajo, izquierda y derecha para un píxel dado. Es decir, que si debajo de un 1 (en la imagen binaria) existe otro 1, ellos dos forman un objeto. **En el algoritmo de dos pasadas, basta con mirar arriba y a la izquierda del píxel.**



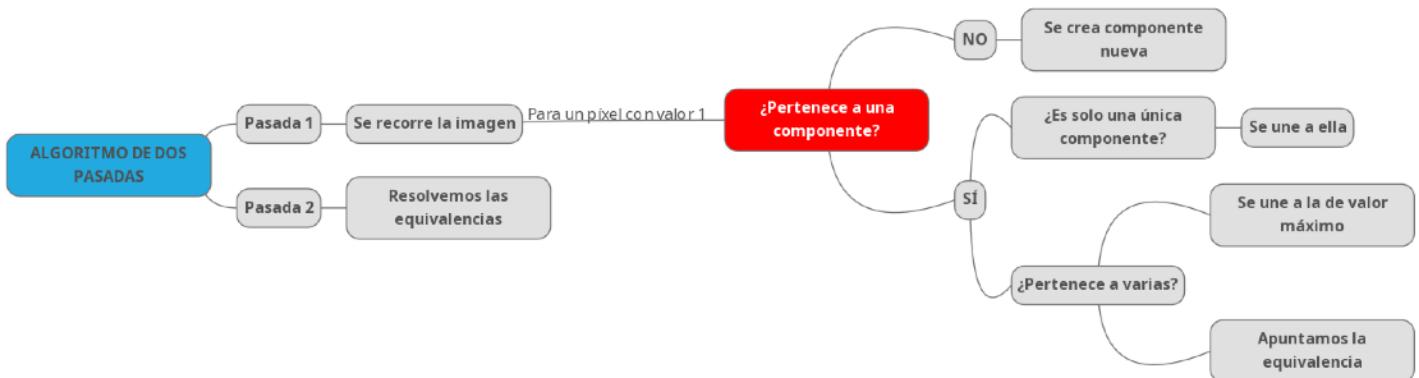
- **8-conectadas.** Se trata de identificar a un objeto/componente a partir de los vecinos arriba, abajo, izquierda, derecha y diagonal para un píxel dado. Es decir, que si en la diagonal derecha superior de un 1 (en la imagen binaria) existe otro 1, ellos dos forman un objeto. **En el algoritmo de dos pasadas, basta con mirar arriba, a la izquierda, a la diagonal superior derecha y a la diagonal superior izquierda del píxel.**



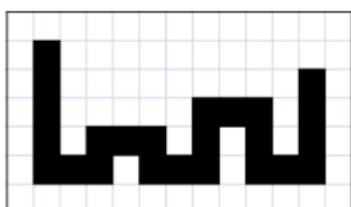
Por tanto, según el criterio que se utilice, en una misma imagen es posible detectar distintos objetos:

1	1	0	0	1
0	1	0	1	0
0	1	0	0	0
0	0	1	1	1
0	0	0	0	1
1	1	0	0	2
0	1	0	3	0
0	1	0	0	0
0	0	4	4	4
0	0	0	0	4
1	1	0	0	2
0	1	0	2	0
0	1	0	0	0
0	0	1	1	1
0	0	0	0	1

Para hallar estas componentes, se utilizará el **algoritmo de dos pasadas**. Un mapa conceptual de sus pasos es el siguiente:



Resolver las equivalencias significa que dos componentes que están juntas se unen formando una componente única. Un ejemplo es el de la imagen a continuación. En ella, se observa la primera pasada de un 4-conectadas. La segunda pasada juntaría las clases 1, 4, 3 y 2 en una única componente.



Caja de Mínima Inclusión

Para cada componente conexa, se detectan:

- El píxel más a la izquierda (x_1).
- El píxel más a la derecha (x_2).
- El píxel más arriba (y_1).
- El píxel más bajo (y_2).

Con esos 4 puntos, se dibuja un rectángulo.



3.4. Filtros Morfológicos

Las operaciones morfológicas son el **dilatado** y el **erosionado**. Aunque estas operaciones también se pueden definir sobre imágenes en escala de grises, **normalmente se aplican a imágenes binarias**.

Estas operaciones **pueden ser combinadas para crear operadores compuestos, como la apertura y el cierre.**

Erosión

La erosión binaria se define matemáticamente como:

$$A \ominus B = \{z \in E \mid B_z \subseteq A\}$$

donde A es la imagen binaria, B el elemento estructurante (*kernel*) y E es el espacio de todos los píxeles.

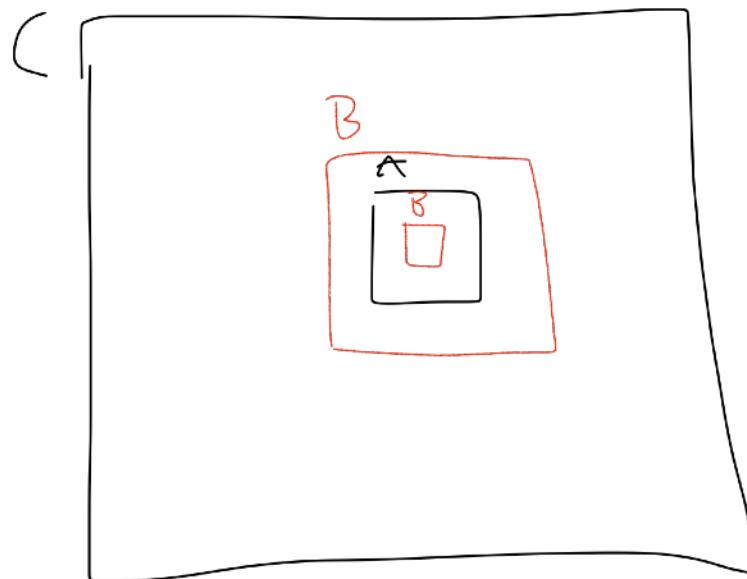
El resultado de un erosionado es la retención (poner a 1) de aquellos píxeles de A tal que si se sitúa sobre ellos el elemento estructurante B, **todos los píxeles que se solapan con B son 1**. Su operación lógica es *AND*.

0000000000	
0000001100	
0111011110	1 1 1
0111111110	1 1 1
0111011100	1 1 1
0010000000	
0000000000	
	↓
0000000000	
0000000000	
0000000000	
0010001000	
0000000000	
0000000000	
0000000000	

El resultado del erosionado debe almacenarse en una imagen nueva. El elemento estructurante no tiene por qué tener la forma anterior. En el caso de que tenga otra forma (estrella, triángulo...), únicamente se tendrán en cuenta los 1.

Propiedades

- Invariante a la translación. Si trasladas A, también se traslada el resultado.
- Creciente: si $A \subseteq C$, entonces $A \ominus B \subseteq C \ominus B$.



- Es anti-extensiva: $A \ominus B \subseteq A$.
- Se cumple que: $(A \ominus B) \ominus C = A \ominus (B \oplus C)$, donde \oplus es el operador de dilatado.

“Erosionar dos veces seguidas es lo mismo que erosionar con el dilatado de los elementos estructurantes”

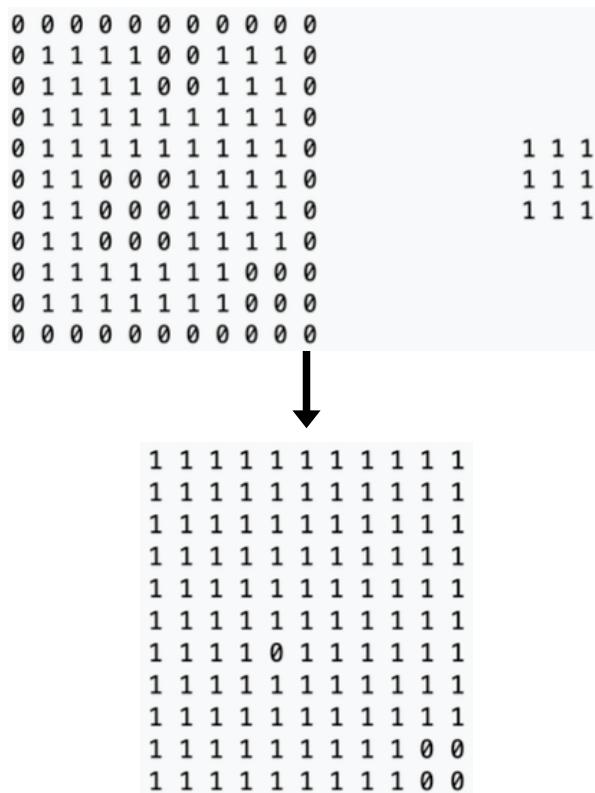
Dilatado

El dilatado binario se define matemáticamente como:

$$A \oplus B = \cup_{b \in B} A_b$$

donde A es la imagen binaria y B el elemento estructurante.

Un píxel de A se pone a 1 si hay al menos un 1 en la zona de solape con B. Es como una operación *OR*.



Propiedades

- Invariante a la traslación. Si trasladas A, también se traslada el resultado.
- Creciente: si $A \subseteq C$, entonces $A \oplus B \subseteq C \oplus B$.
- Es extensiva: $A \subseteq A \oplus B$.
- Es comutativa: $A \oplus B = B \oplus A$.
- Es asociativa: $(A \oplus B) \oplus C = A \oplus (B \oplus C)$.

- Se cumple que: $(A \ominus B) \ominus C = A \ominus (B \oplus C)$, donde \oplus es el operador de dilatado.

“Erosionar dos veces seguidas es lo mismo que erosionar con el dilatado de los elementos estructurantes”

Curiosidades entre erosión y dilatado

“Dilatar el objeto es erosionar el fondo”

“Erosionar el objeto es dilatar el fondo”.

Sea $\mathbf{!A}$ la imagen binaria invertida de A , sea B el elemento estructurante,

- ¿Qué se obtendría de $\mathbf{!A} \oplus B$?

$$\mathbf{!A} \oplus B = \mathbf{!(A \ominus B)}$$

- ¿Qué se obtendría de $\mathbf{!A} \ominus B$?

$$\mathbf{!A} \ominus B = \mathbf{!(A \oplus B)}$$

Peculiaridad del erosionado

Si se realiza un erosionado y, después, se le resta la imagen original a dicho resultado, ¿qué estamos obteniendo?

$$A - (A \ominus B)$$

0 0 0 0 0 0 0 0 0	
0 1 1 1 1 1 1 0 0	
0 1 1 1 1 1 1 0 0	
0 1 1 1 1 1 1 0 0	1 1 1
0 1 1 1 1 0 0 0 0	1 1 1
0 1 1 1 1 0 0 0 0	1 1 1
0 1 1 1 1 0 0 0 0	
0 0 1 1 1 0 0 0 0	
0 0 0 0 0 0 0 0 0	

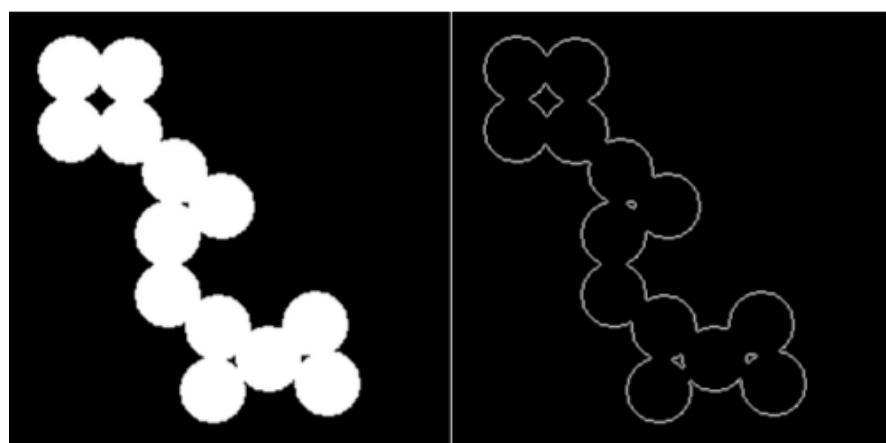
Veamos la erosión,

```
000000000  
000000000  
001111000  
001100000  
001100000  
001100000  
001100000  
000000000  
000000000
```

Ahora, la diferencia entre la imagen original y el erosionado,

```
000000000  
011111100  
010000100  
010001100  
01001|0000  
010010000  
010010000  
001110000  
000000000
```

¡Se obtienen las fronteras de la imagen!



Apertura

Una apertura es la combinación de un erosionado y un dilatado:

$$A \circ B = (A \ominus B) \oplus B$$

En concreto, primero se realiza un erosionado y después un dilatado. Se obtiene una apertura, es decir, una desconexión entre zonas de la imagen. Normalmente, se emplea para desconectar componentes conexas colindantes pero restaurando su tamaño original. Si únicamente se erosionara, no se obtendría el tamaño original de las componentes. También se utiliza para eliminar estructuras pequeñas.

Un ejemplo:



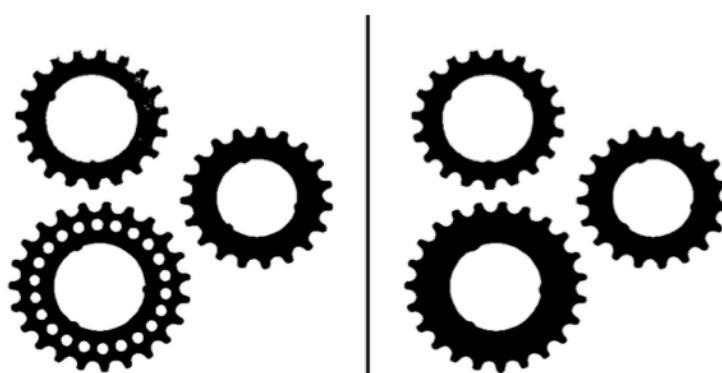
Cierre

Un cierre es la combinación de un dilatado y un erosionado.

$$A \bullet B = (A \oplus B) \ominus B$$

En concreto, primero se realiza un dilatado y después un erosionado. Un cierre es una conexión entre zonas de la imagen. Normalmente se emplea para llenar huecos que suelen estar provocados por defectos de la umbralización.

Un ejemplo:



Hit or Miss

En este caso, se consideran tanto los 1 como los 0 del elemento estructurante, que puede tener cualquier forma. En la imagen se pone un 1 si y solo si coincide la zona de solape con el elemento estructurante.

Es posible definir zonas de no-relevancia en el elemento estructurante, que podrá contener los valores: 1, 0 y X (da igual lo que haya en esta posición).

Es como un **pattern matching selectivo**.

Un ejemplo:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & \textcolor{red}{1} & 1 & 1 \\ X & 1 & 1 & X \end{bmatrix}$$

*El centro del elemento estructurante es el número señalado.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Morfología en Escala de Grises

Erosionado sobre grises

Se define el erosionado sobre una imagen en escala de grises como:

$$(A \ominus B)(x) = \min_{y \in E}(A(x - y) - B(y))$$

Obviamente, los valores del elemento estructurante no tienen por qué ser binarios. En cada píxel donde se sitúe B, se obtiene el mínimo de la diferencia de los píxeles que solapan con B.

Básicamente, se pone B en A. Para los píxeles que coincidan, se resta A - B y se obtiene el mínimo.

Si, por algún casual, el mínimo fuese negativo, se podría un 0.

Dilatado sobre grises

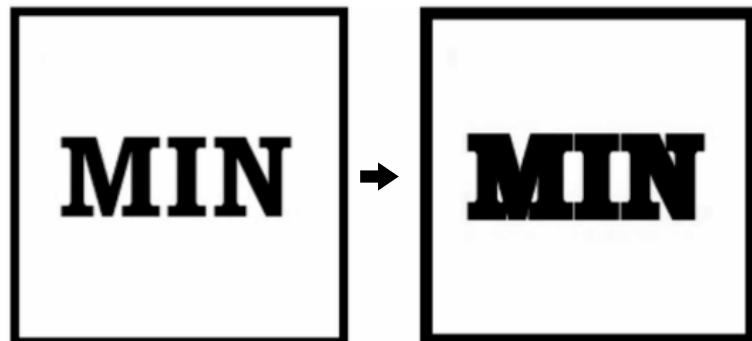
Se define el dilatado sobre una imagen en escala de grises como:

$$(A \oplus B)(x) = \max_{y \in E}(A(x - y) + B(y))$$

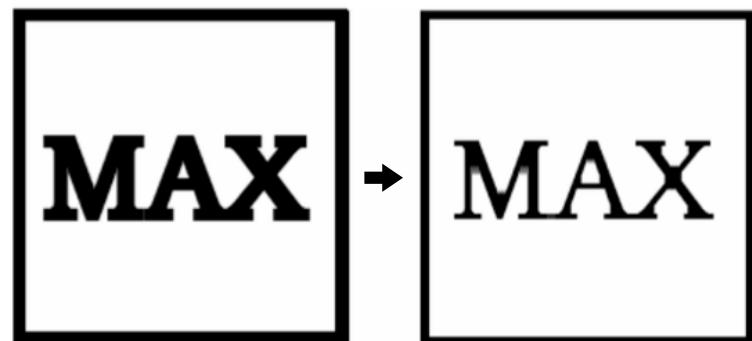
En cada píxel donde se sitúe B, se obtiene el máximo de la suma de los píxeles que solapan con B. Es decir, para los píxeles que coincidan, se suma A + B y se obtiene el máximo.

Filtros Min y Max

- **Min:** hace un erosionado de lo brillante (lo blanco).



- **Max:** hace un dilatado de lo brillante (lo blanco).



Estos filtros tienen poca utilidad.