

Percepción Computacional

---

# Procesamiento de imagen. Morfología matemática

## 8.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

## 8.2. Introducción a la morfología matemática

**L**a morfología matemática es una de las ramas de las matemáticas que estudia el análisis y tratamiento de estructuras geométricas basándose en topología y geometría. Nació en la Escuela de Minas de París como fruto de la tesis doctoral de Jean Serra, supervisada por Georges Matheron.

La principal utilidad de la morfología matemática es la de **simplificar el aspecto visual de una imagen**, pero conservando sus características o composición principales. Es decir, una de sus utilidades es la de eliminar detalles, ruido o incluso componentes que pudieran distraer a otros algoritmos de segmentación.

A continuación presentamos un ejemplo sencillo donde se puede apreciar cómo los operadores **dilatación (Dilation)** y **cierre (Closing)** potencian más la eliminación de los detalles de la superficie quedándose con la forma de la hoja, incluso podemos ver cómo han eliminado la rama de la hoja.

Por otro lado, los operadores **erosión (Erosion)** y **apertura (Opening)**, potencian los detalles de la hoja como son las hebras que existen en ella.

Además, se observa que los operadores de apertura y cierre conservan el tamaño y las proporciones de la imagen original. No sucede así con los operadores de erosión y dilatación. No obstante, todos estos detalles los veremos en este tema.

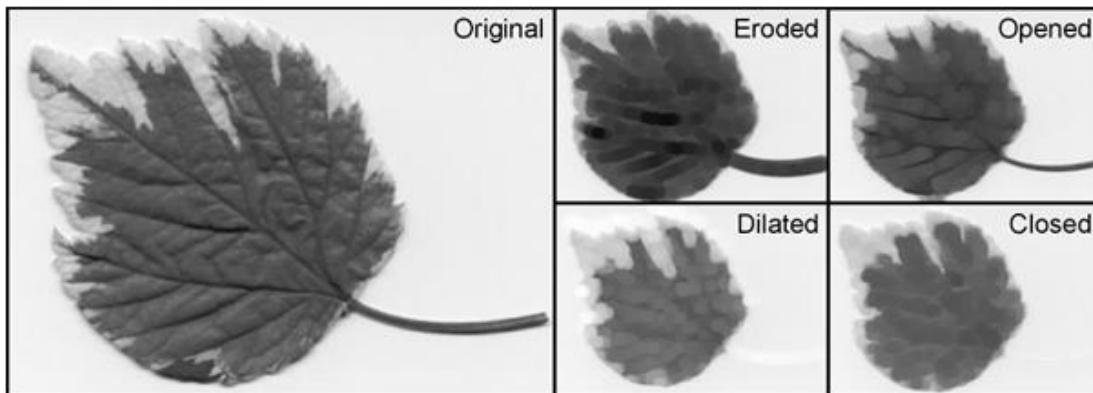


Figura 1. Resultado de aplicar los operadores más comunes de morfología matemática.

Fuente: <https://imagej.nih.gov/ij/plugins/gray-morphology.html>

Por otro lado, la morfología matemática ayuda mucho a postprocesar el resultado de una segmentación.

A continuación se presenta una ilustración con los efectos de dilatación y erosión sobre dos objetos ya segmentados: dichos objetos vienen representados por el conjunto X. Al aplicar los operadores de dilatación y erosión con el operador B, más adelante se explicará cómo funciona este operador denominado **elemento estructural**, se puede potenciar o disminuir los tamaños de los objetos en X e incluso hacer desaparecer algunos de sus elementos.

Con estos dos operadores, podríamos depurar los resultados de la segmentación eliminando aquellos residuos procedentes de una sobresegmentación o suavizar los contornos de una imagen.

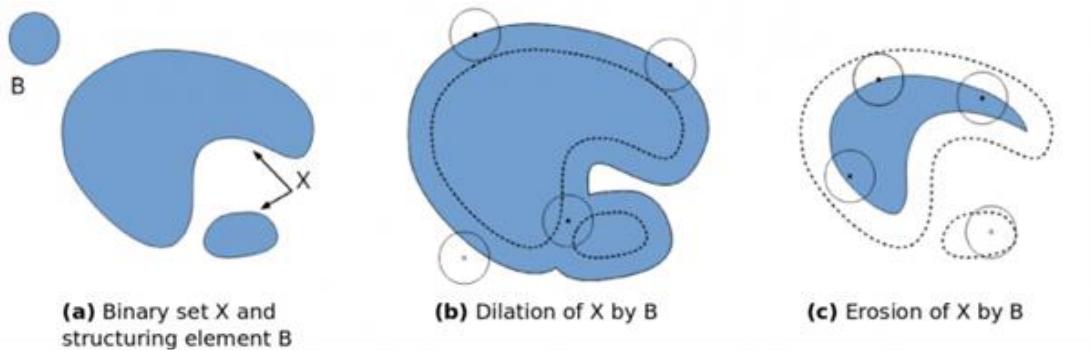


Figura 2. Efecto de dos operadores morfológicos: dilatación y erosión.

Fuente: <https://imagej.net/MorphoLib>

Así, en esta imagen vemos:

- ▶ La imagen original junto con B, el elemento estructural (a).
- ▶ Dilatación de la imagen original usando B, en este caso se aprecia cómo los dos grupos se han unido (b).
- ▶ Erosión de la imagen original usando B. El elemento más pequeño ha desaparecido y el más grande, aunque ha disminuido, ha conservado su forma (c).

Puede aplicarse también a señales unidimensionales, pero su mayor uso está concentrado en las imágenes binarias y en las imágenes de escala de grises.

Los operadores morfológicos, *per se*, no aportan información sobre la forma de los objetos ni son capaces de extraer características. Su utilidad se centra en el **limpiado de imágenes ya segmentadas** o la eliminación de detalles tanto en la imagen original como en la imagen segmentada para que los algoritmos de extracción de características puedan obtener resultados más precisos.

## 8.3. Definición de elemento estructural

**E**l elemento estructural es un ingrediente esencial de la morfología matemática. En algunos contextos se nombra también como elemento estructurante.

La función del elemento estructural es similar a la función de máscara en un filtro gaussiano de imágenes: es un **elemento matricial** (mínimo de 3x3 píxeles) que recorre la imagen de forma iterativa y sobre el cual los operadores morfológicos actúan. Es decir, un operador morfológico actúa sobre una imagen como suma de todos los efectos producidos sobre el elemento estructural cuando este recorre la imagen.

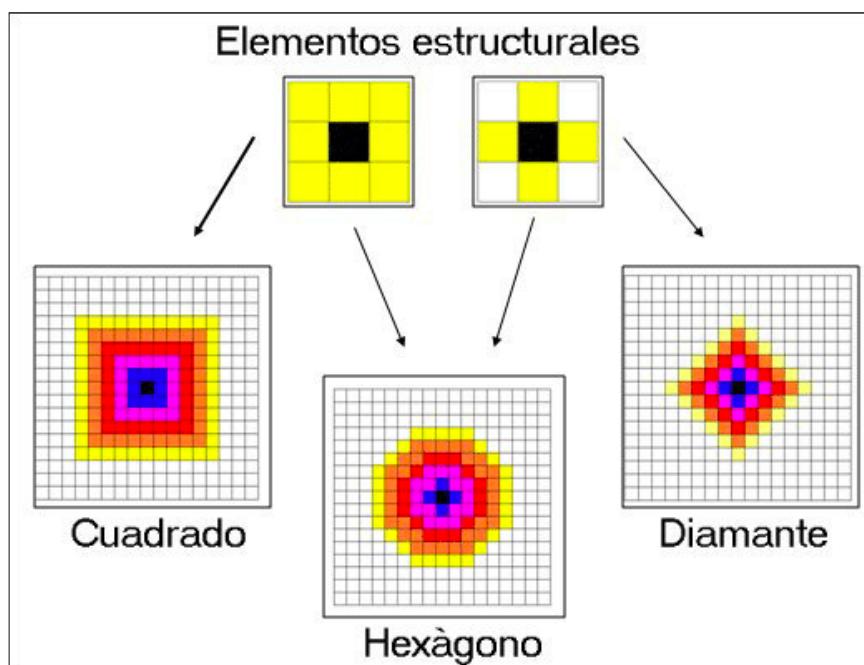


Figura 3. Ejemplos de elementos estructurales con diferentes formas.

Fuente: [https://www.researchgate.net/figure/Figura-47-Elementos-estructurales-de-la-Morfologia-matematica\\_fig9\\_313905159](https://www.researchgate.net/figure/Figura-47-Elementos-estructurales-de-la-Morfologia-matematica_fig9_313905159)

La definición de elemento estructural es esencial para el operador morfológico.

Supongamos que queremos contar el número de círculos existentes en la siguiente imagen:

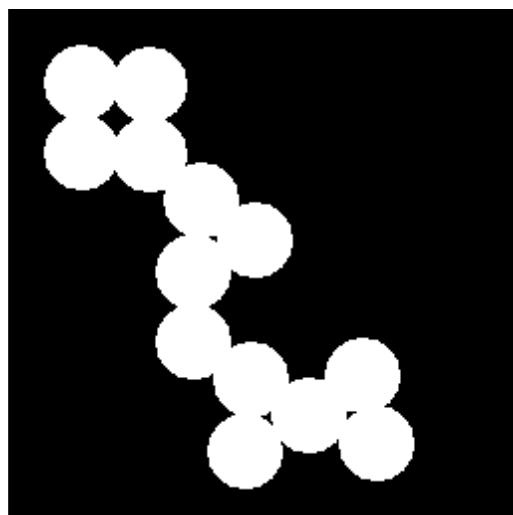


Figura 4. Ejemplo de imagen con círculos solapadas.

Fuente: <https://es.mathworks.com/help/images/ref/bwulterode.html>

La dificultad radica en contar el número de círculos. Independientemente del operador morfológico que empleemos (erosión, dilatación, etc.), el elemento estructural ha de ser lo más parecido a lo que se quiere encontrar y procesar. En este caso, el elemento estructural que mejor funciona es el circular o también llamado **de hexágono** en algunos contextos.

La razón por la cual puede llamarse hexágono viene dada por el hecho de que un círculo no puede aproximarse perfectamente en una matriz pixelada. De esta manera se puede alcanzar el resultado que buscamos.



Figura 5. Resultado de aplicar elementos estructurales circulares a la imagen.

Fuente: <http://matlab.izmiran.ru/help/toolbox/images/imerode.html>

A continuación, se presenta el resultado de haber utilizado otro elemento estructural como es el caso de una línea, de longitud 20 píxeles y orientación 90° (vertical) y 180° (horizontal) respectivamente.

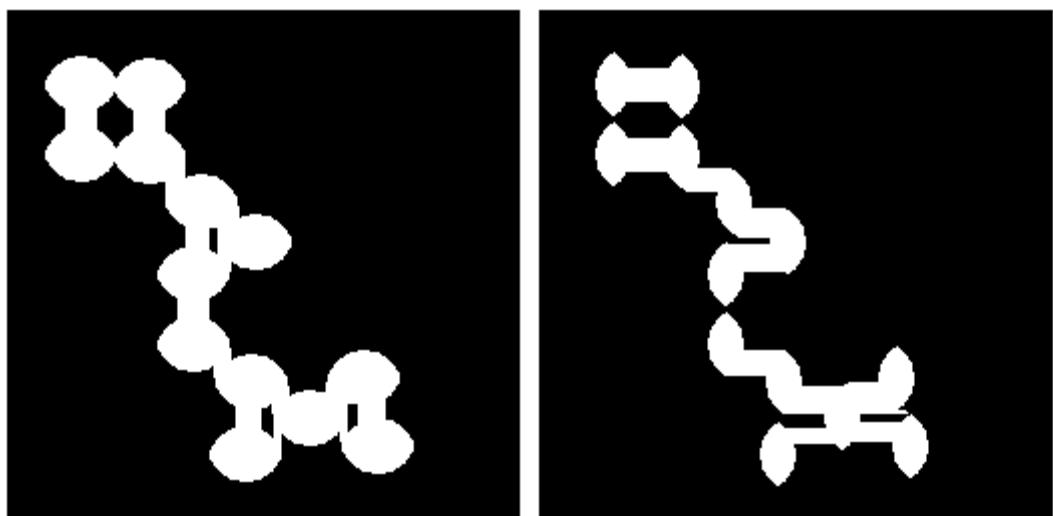


Figura 6. Ejemplos de usar un elemento estructural en forma de línea horizontal y vertical.

Fuente: <http://matlab.izmiran.ru/help/toolbox/images/imerode.html>

No obstante, el elemento estructural por sí mismo carece de sentido y ha de entenderse más en detalle con los operadores morfológicos que se verán a continuación.

## 8.4. Erosión y dilatación

**L**a erosión y la dilatación son los operadores morfológicos esenciales. A partir de ellos, se crean el resto de operadores. Aunque lo parezca (y antes de dar definición alguna sobre ellos), es importante hacer constar que la erosión no es la inversa de la dilatación ni viceversa.

La **dilatación** se define matemáticamente como:

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

Donde:

- ▶ A es la imagen original.
- ▶ B es el elemento estructural.

La función de la operación dilatación consiste en potenciar y aumentar los contornos de una imagen binaria, haciendo que los detalles se magnifiquen e incluso que algunos objetos dentro de una imagen, que originalmente estén separados, se fusionen.

Este tipo de funcionalidad tiene mucha utilidad cuando, tras una segmentación, hay objetos que se han quedado separados, pero cercanos en la imagen. La manera de implementar computacionalmente el operador dilatación (luego lo veremos también con el operador erosión) es de forma iterativa, comprobando píxel a píxel dentro de la imagen qué relación tienen el elemento estructural (B) y la imagen original (A).

En este caso, el operador mirará si el elemento estructural B coincide al menos en un píxel con la imagen original A. En ese caso, añadirá a dicha imagen el elemento estructural B, haciendo que la imagen aumente en la zona de los contornos y manteniéndose constante en el interior de la misma. Posteriormente veremos con un ejemplo cómo funciona este procedimiento.

La **erosión**, matemáticamente, se define como:

$$A \ominus B = \{z \in E | Bz \subseteq A\}$$

Donde:

- ▶ A es la imagen original.
- ▶ B es el elemento estructural.
- ▶ E es una generalización del espacio euclídeo, es decir, engloba todos los posibles puntos existentes incluyendo los de A.

La principal utilidad de la erosión como operador morfológico es el de eliminar detalles, reducir contornos y, en algunos casos, desunir objetos.

La erosión no conserva la forma original: tras usar este operador, la forma resultante puede haberse erosionado tanto que no conserve la morfología original. La manera de implementar computacionalmente el operador erosión es también de forma iterativa, comprobando píxel a píxel dentro de la imagen qué relación tienen el elemento estructural (B) y la imagen original (A).

En este caso, el operador mirará píxel a píxel si el elemento estructural B está contenido íntegramente en la imagen original A. En ese caso, se quedará únicamente con la posición/píxel donde ambos elementos (A y B) coincidan. De esta manera, el contorno de la imagen se reduce considerablemente.

Sin embargo, la mejor manera de ver ambos operadores es mediante un ejemplo visual. Primero empezaremos con el de dilatación y posteriormente aplicaremos el operador erosión sobre la imagen resultante de la dilatación.

Vamos a empezar con este ejemplo, donde a la izquierda está el elemento estructural, aunque tenga forma de cruz, puede ser un elemento estructural de disco o diamante de radio 3 píxeles. Y a la derecha, la imagen sobre la que se quiere aplicar el operador morfológico.

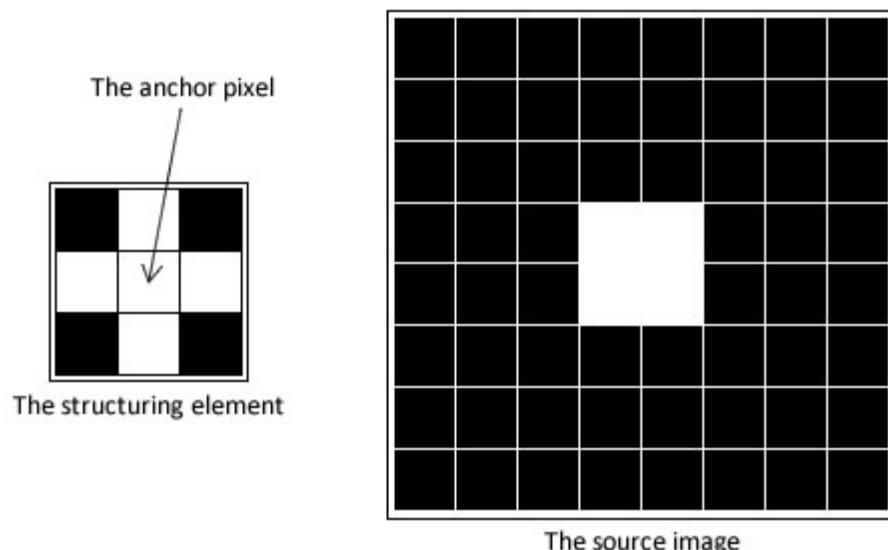


Figura 7. Elemento estructural en forma de cruz y la imagen sobre el que se va a realizar la operación morfológica.

Fuente: <http://aishack.in/tutorials/mathematical-morphology/>

Independientemente del elemento estructural empleado, la imagen resultante después de aplicar la dilatación ha de ser mayor (en el sentido de número de píxeles en blanco) que en la imagen original.

Como se ha descrito anteriormente, el operador dilatación irá moviendo el elemento estructural a lo largo de la imagen A, comprobando si A y B coinciden en al menos un píxel. Inicialmente, no existe tal solape y, por lo tanto, el operador morfológico no devuelve ningún valor en blanco.

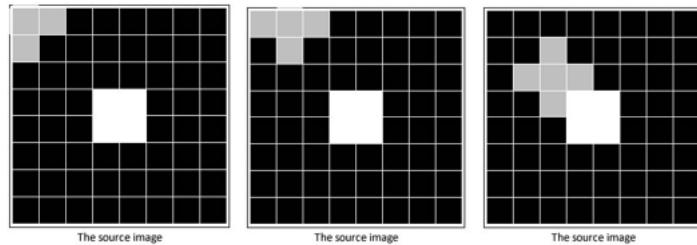


Figura 8. Elemento estructural en forma de cruz y la imagen sobre el que se va a realizar la operación morfológica.

Fuente: Adaptado de <http://aishack.in/tutorials/mathematical-morphology/>

Sin embargo, cuando el elemento estructural, toca en **al menos un píxel**, la zona de blancos dentro de la imagen original, es entonces donde el operador dilatación aplica su definición. Es decir, en aquellas posiciones donde ambos coincidan se hará la unión de ambos operadores, tal y como puede verse en la siguiente imagen.

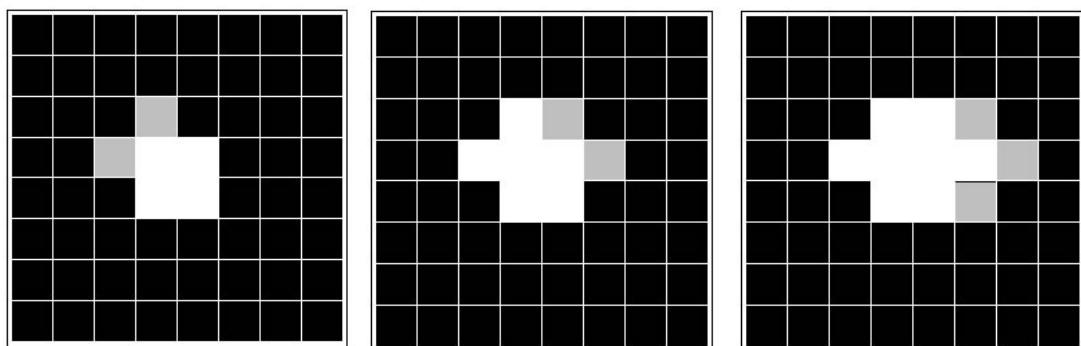


Figura 9. Paso a paso de cómo el elemento estructural entra en contacto con una zona dentro de la imagen con píxeles diferentes a cero (negro).

Fuente: Adaptado de <http://aishack.in/tutorials/mathematical-morphology/>

En este caso, y puesto que se trata de una dilatación, se conserva la unión de ambos elementos. Finalmente, tras recorrer toda la imagen, el resultado es que el cuadrado se ha dilatado, ha aumentado de tamaño con la forma del elemento estructural. En ningún momento se aprecia que se haya conservado la forma inicial (se trataba de un cuadrado) y tiene forma de cruz.

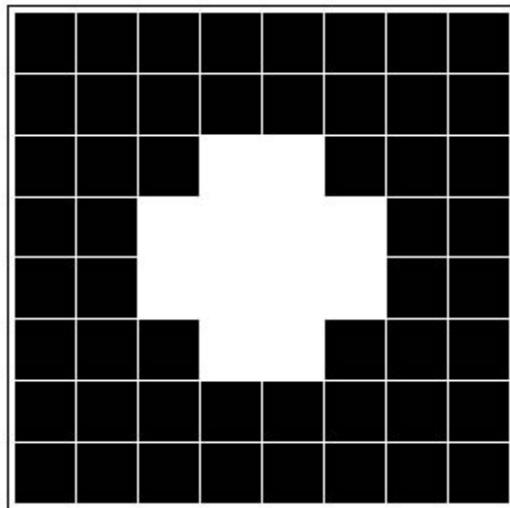


Figura 10. Resultado final del operador morfológico dilatación sobre un cuadrado de 2x2 píxeles y con un elemento estructural en forma de cruz.

Fuente: <http://aishack.in/tutorials/mathematical-morphology/>

El siguiente proceso va a consistir en aplicar la erosión a la imagen dilatada anterior. Es importante tener en mente que al aplicar el operador erosión, el número de **píxeles en blanco se verá disminuido** en el resultado.

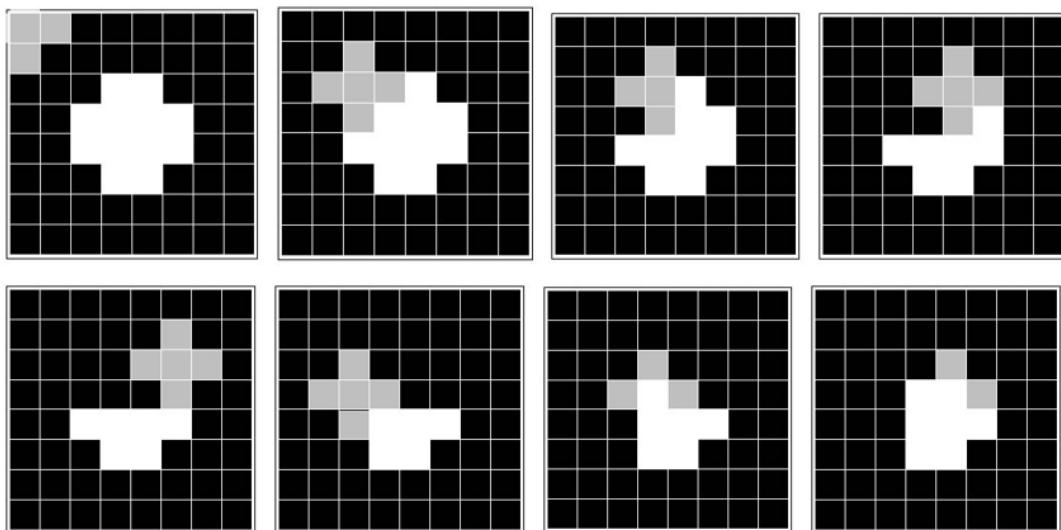


Figura 11. Proceso de erosión con un operador en forma de cruz.  
Fuente: Adaptado de <http://aishack.in/tutorials/mathematical-morphology/>

En la imagen resultante se han eliminado detalles en el contorno, quedándose con la forma inicial de cuadrado de 2x2 píxeles, aunque la imagen no conserva la forma original. Tras la erosión, la imagen tiene menos píxeles en blanco que procesar y para ciertas funciones, como el conteo de objetos, puede llegar a ser más fácil que con la imagen original. Por otro lado, si lo que se quiere es detectar el centro de un objeto, el resultado de la erosión también ayudaría, pues facilitaría el cálculo del centro del objeto al haber reducido los contornos y detalles del mismo.

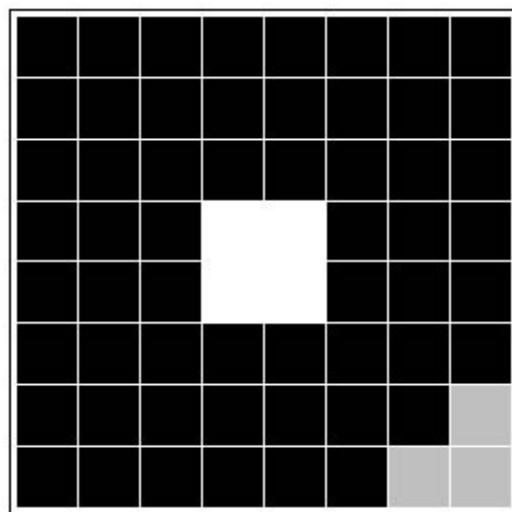


Figura 12. Resultado final del operador morfológico erosión sobre una cruz y con un elemento estructural en forma de cruz. El resultado es una imagen que no conserva la forma original.

Fuente: <http://aishack.in/tutorials/mathematical-morphology/>

Por último, y aunque en este caso haya sucedido así, repetimos que la erosión no es la inversa de la dilatación ni viceversa.

## 8.5. Apertura y clausura

**A**diferencia de los operadores anteriores, la apertura y clausura son los primeros operadores combinación de varias operaciones morfológicas. Su mayor propiedad es que son capaces de respetar, en la medida de lo posible, la morfología adicional.

Con lo cual, tienen las propiedades de la erosión y la dilatación, pero **manteniendo la forma original**. En otras palabras, son capaces de eliminar o potenciar los detalles y aun así, mantienen la proporción original.

La **apertura**, conocida en inglés como *opening*, es un operador morfológico que es la sucesión de una erosión seguida por una dilatación. Matemáticamente, la apertura se define como:

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

Donde:

- ▶ A es la imagen original.
- ▶ B es el elemento estructural.

Principalmente, este operador sirve para separar objetos, ya que elimina los posibles detalles que haya en la imagen en función del elemento estructural.

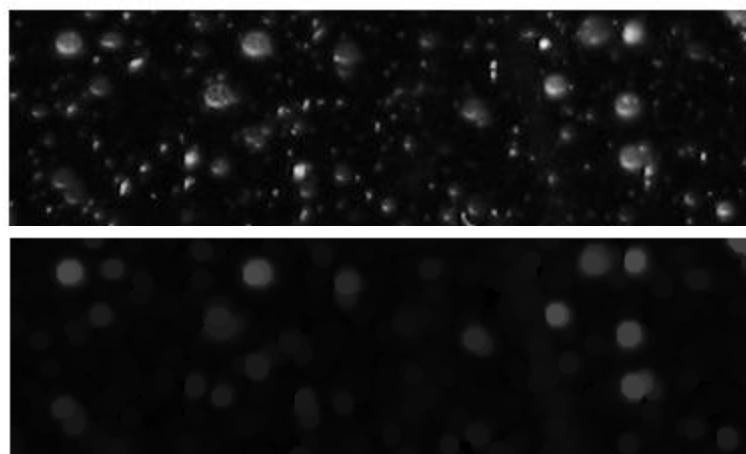


Figura 13. Resultado de aplicar el operador morfológico apertura.  
Fuente: <https://es.mathworks.com/help/images/ref/imreconstruct.html>

En este ejemplo, vemos el resultado de aplicar el operador morfológico apertura con un elemento estructural disco de 5 píxeles de radio. El efecto es el de haber eliminado los detalles (puntos) más pequeños, manteniendo los objetos más grandes.

La **clausura**, conocida en inglés como *closing*, es un operador morfológico que realiza primeramente una dilatación y posteriormente una erosión.

Matemáticamente, la clausura se define como:

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

Donde:

- ▶ A es la imagen original.
- ▶ B es el elemento estructural.

La clausura suele emplearse para reafirmar y consolidar formas dentro de una imagen.

Como podemos apreciar en la siguiente imagen, el operador ha transformado la figura inicial en un solo bloque.



Figura 14. Clausura con un disco de radio 20 píxeles.

Fuente: [https://users.cs.cf.ac.uk/Dave.Marshall/Vision\\_lecture/node34.html](https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node34.html)

## 8.6. Gradiente morfológico

Otro operador muy empleado en morfología matemática es el gradiente morfológico. Se define como la diferencia entre la dilatación y la erosión. La idea intuitiva es la de **detectar bordes en la imagen**; ya que la dilatación aumenta una imagen y la erosión la disminuye, la diferencia ha de ser similar a los bordes de una imagen.

Aquí tenemos un ejemplo de gradiente morfológico aplicado a una imagen con granos de arroz. Se ha utilizado un elemento estructural de disco con radio de 3 píxeles

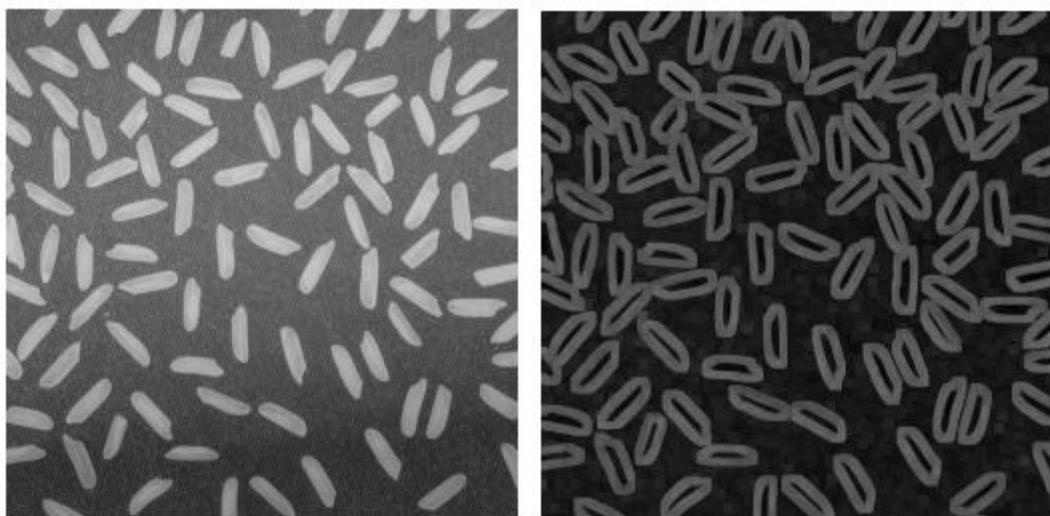


Figura 15. Ejemplo de gradiente morfológico.

Fuente: Adaptado de <https://imagej.net/MorphoLibJ>

## 8.7. Top Hat

E l operador *Top Hat* se define como la diferencia entre la imagen original y el operador morfológico apertura. Devuelve una imagen con únicamente los detalles que la apertura ha eliminado.

Esta transformada es muy útil en el caso de la detección de texto dentro de una imagen, como leer las marcas de una moneda, ya que **se centra en los detalles**. Pero si quisiésemos contar el número de monedas, esta transformada no aportaría ningún valor.

En esta imagen, vemos el resultado de aplicar la transformada *Top Hat* mediante un disco de radio de 5 píxeles.



Figura 16. Detección de los detalles de una moneda mediante la transformada *Top Hat*.

Fuente: Adaptado de <https://imagej.net/MorphoLibJ>

## 8.8. Ejercicio práctico

**C**on todo lo anterior explicado, vamos a realizar un ejercicio práctico. Partiendo de esta conocida imagen, el objetivo es quedarse únicamente con el fotógrafo y eliminar el resto de componentes, incluyendo la cámara y los edificios de atrás.



Figura 17. Detección de los detalles de una moneda mediante la transformada *Top Hat*.

Fuente: <https://blogs.mathworks.com/steve/2012/11/20/image-effects-part-2/>

Existen múltiples soluciones para hacer esto, pero los pasos que vamos a describir son una aproximación adecuada.

Lo primero de todo es eliminar las patas de la cámara. Esto podemos hacerlo con morfología matemática mediante un **operador de clausura** haciendo uso de un elemento estructural que sea perpendicular a la dirección de las patas de la cámara. La razón de esto es eliminar dichas patas lo máximo posible, puesto que al hacer la clausura difuminaría estos objetos. Así, en esta imagen vemos la aplicación de clausura con un elemento estructural de línea de 45 px y 45° de inclinación.



Figura 18. Resultado de aplicar un operador de clausura.

Fuente: Adaptado de <https://blogs.mathworks.com/steve/2012/11/20/image-effects-part-2/>

La forma del fotógrafo se ha conservado en gran medida, luego, para eliminar las posibles líneas provocadas por el anterior operador, es conveniente realizar un operador de apertura (para consolidar las formas y eliminar detalles). De esta manera, y aunque parezca muy borroso, la figura del cámara se ha conservado habiéndose eliminado la cámara y el fondo.

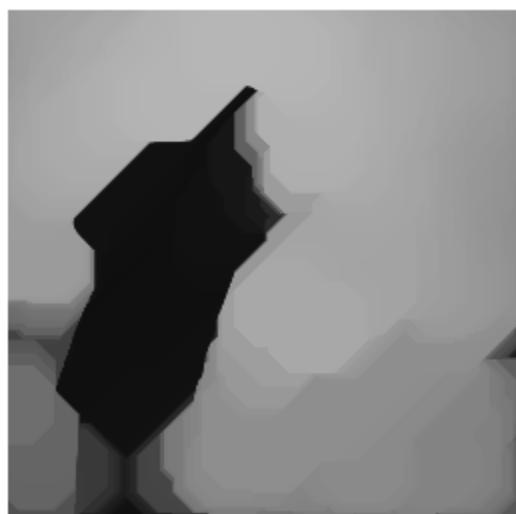


Figura 19. Resultado de aplicar un operador de apertura.

Fuente: Adaptado de <https://blogs.mathworks.com/steve/2012/11/20/image-effects-part-2/>

Únicamente queda aplicar este filtro a la imagen original. En este caso lo haremos multiplicando ambas imágenes, componente a componente, una manera muy sencilla de aplicar el filtro.

El resultado puede apreciarse en la imagen siguiente, donde hemos eliminado todo el fondo y la cámara, quedándonos únicamente con la parte correspondiente al fotógrafo. Además, como resultado del filtrado de multiplicación de las dos imágenes se ha obtenido un reescalado que permite ver más detalles del abrigo que inicialmente no eran perceptibles.



Figura 20. Resultado final del ejercicio práctico.

Fuente: Adaptado de <https://blogs.mathworks.com/steve/2012/11/20/image-effects-part-2/>

## 8.9. Referencias bibliográficas

González, R. C. y Woods, R. E. (2008). *Digital image processing*. New Jersey: Pearson Education.

Percepción Computacional

---

# Procesamiento de señales. Filtrado y análisis en frecuencia

## 9.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

## 9.2. Objetivos

**E**sta asignatura persigue presentar el concepto de análisis en frecuencia desde su origen (series de Fourier) hasta su implementación actual (FFT, *Fast Fourier Transform*). Para ello, se revisan los conceptos de análisis en frecuencia y transformada de Fourier, así como las ventajas y desventajas del uso de estas herramientas en el procesamiento computacional.

Se sentarán las bases del análisis en frecuencia, lo que te permitirá abordar otros algoritmos y herramientas como transformada del coseno (DCT), Wavelets o filtrados en frecuencia para imágenes.

## 9.3. Introducción al análisis en frecuencia

**E**l análisis en frecuencia es una de las herramientas más potentes e innovadoras que existen en el procesamiento de la información. En este apartado se va a describir, con un ejemplo muy cotidiano: la señal de fibra óptica y teléfono que llega a nuestros hogares, en qué consiste el análisis en frecuencia y por qué es tan importante.

Si el cable de red que va al router lo conectásemos a un osciloscopio o a cualquier otro dispositivo que nos permitiera medir voltajes e intensidades, veríamos algo parecido a esta imagen.

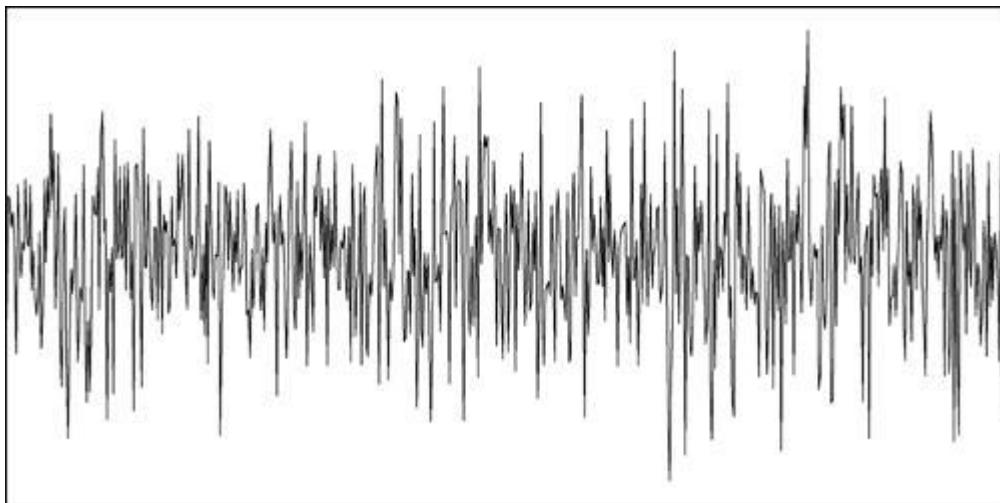


Figura 1. Ejemplo de señal compleja aperiódica.

Es el claro ejemplo de una señal compleja aperiódica o señal aleatoria, en algunos contextos se llama también caótica. En esta señal unidimensional podemos ver variaciones de intensidad y de voltaje correspondientes a la superposición de nuestra voz (llamadas telefónicas), Internet (datos que estamos recibiendo) y en algunos casos, incluso canales de televisión.

Todo ello sucede simultáneamente. Sin embargo, al esquematizar qué es lo que está pasando, nos encontramos con el siguiente ejemplo:

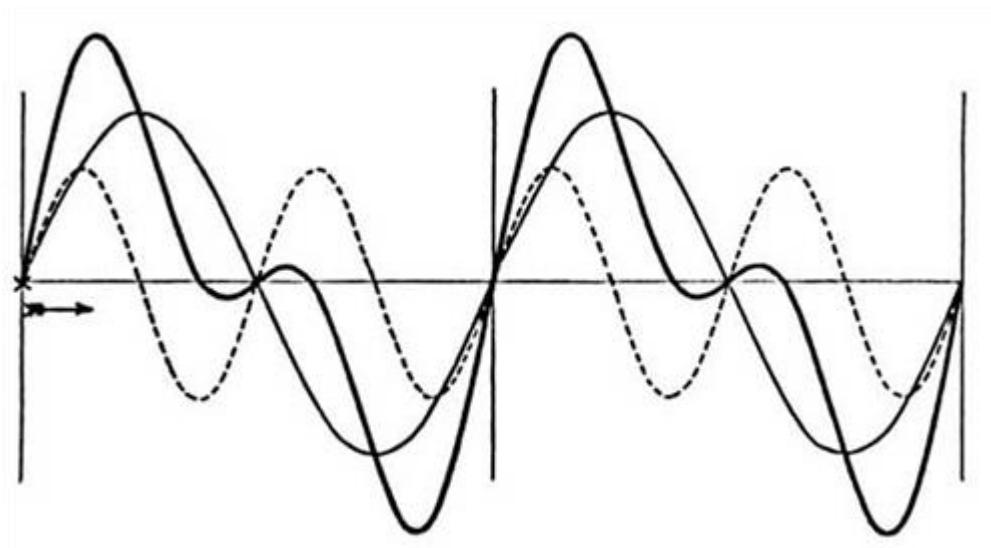


Figura 2. Ejemplo de señales superpuestas en el tiempo.

En este ejemplo, encontramos que existen tres tipos de señales que se superponen. De estas señales, podemos distinguir:

- ▶ La línea continua fina, correspondiente a un tipo de señal.
- ▶ La línea discontinua, correspondiente a un tipo de señal.
- ▶ La línea continua gruesa, suma de ambas señales.

Si nos centramos únicamente en la línea continua gruesa, no podemos saber *a priori* si es el resultado de la suma de dos señales, de un número determinado de señales o si es una única señal. De hecho, si se realizara un análisis de estas señales en el tiempo, podríamos obtener características como la media, el valor máximo, el valor mínimo, etc. Pero en cualquier caso, no podríamos saber el origen.

El **análisis en frecuencia** es la herramienta encargada de, dada la señal continua gruesa, analizar si esta proviene de la suma de dos o más señales y cómo de intensas (potencia) son dichas señales.

Volviendo al ejemplo de la fibra óptica y la voz en nuestras casas, el gráfico anterior sería una simplificación de la realidad. El análisis en frecuencia, en definitiva, es lo que permitiría separar la fibra de la voz y de la televisión. Permitiría realizar un filtrado, que a nuestros ojos, parece inviable.

¿Y qué hace exactamente el análisis en frecuencia? Principalmente, lo que hace es barrer el espectro de frecuencias y detectar, mediante un producto escalar, cuánta señal está sucediendo a esa determinada frecuencia. Posteriormente lo veremos más en detalle.

Tomemos el siguiente gráfico. Si miramos el dominio del tiempo, vemos dos señales superpuestas. Si analizamos las frecuencias, vemos que ambas señales tienen comportamientos en frecuencias diferentes, lo que nos permitiría separar las señales en frecuencia, aunque estén completamente solapadas en el tiempo.

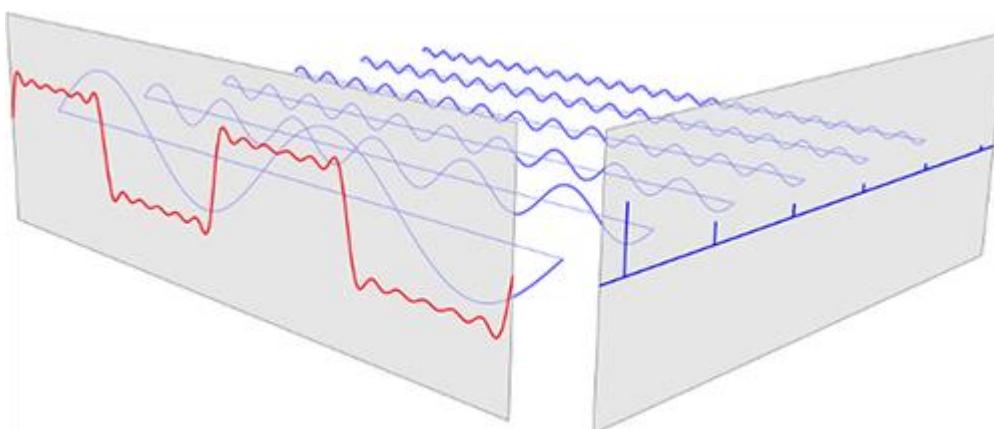


Figura 3. Ejemplo de cómo mediante el uso del dominio de la frecuencia podemos separar dos señales que en el tiempo son coincidentes.

Fuente: <https://tex.stackexchange.com/questions/127375/replicate-the-fourier-transform-time-frequency-domains-correspondence-illustrati>.

Un ejemplo más técnico es el de la siguiente figura. Podemos ver una señal en el tiempo claramente caótica. Es casi imposible entender qué está sucediendo en esa señal o a qué hace referencia. No obstante, el análisis en frecuencia ya nos indica que existen diferentes componentes, con lo cual, si se quisiera escuchar cada uno de forma clara, habría que realizar filtrados de frecuencia y no filtrados temporales.

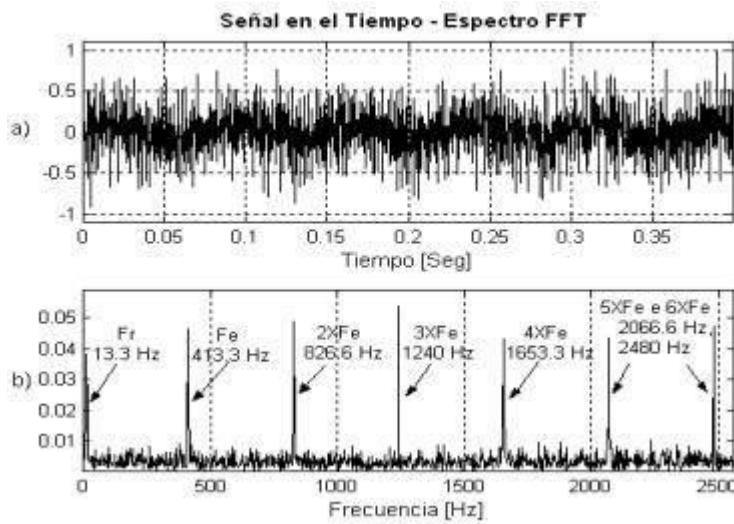


Figura 4. Comparación de representaciones Tiempo-Frecuencia.

Fuente: [https://scielo.conicyt.cl/scielo.php?script=sci\\_arttext&pid=S0718-07642004000500003](https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-07642004000500003).

La herramienta principal de análisis en el dominio de la frecuencia es la **transformada de Fourier**, que permite:

- ▶ Entender qué componentes de frecuencia posee una determinada señal temporal y cuál es la intensidad de dichas componentes.
- ▶ Separar comportamientos temporales de comportamientos de frecuencia.
- ▶ Realizar filtrados en frecuencia y separar un determinado número de señales de forma sencilla y con razonamiento matemático, cuya labor en el análisis temporal habría sido imposible.

Por último, el uso de la transformada de Fourier está muy presente en nuestros días, tan presente que sin esta herramienta matemática hoy no podríamos hablar por el móvil, comunicarnos con los satélites, enviar sondas a Marte, navegar por Internet o incluso detectar si, cuando introducimos una moneda en una máquina de *parking*, es de 1 € o de 2 €.

La transformada de Fourier, junto con otras fórmulas matemáticas, está considerada por la comunidad científica como uno de los avances más considerables de la humanidad.

## 9.4. La transformada de Fourier

**A**

ntes de comenzar a definirla, es importante entender qué camino siguió Fourier para llegar a formularla.

- Entender el origen de la transformada de Fourier: series de Fourier de funciones periódicas.
- Cómo se extienden dichas funciones al campo complejo.
- Definición de la transformada de Fourier como una extensión de las series de Fourier a señales no periódicas.

Figura 5. Pasos seguidos por Fourier.

### Series de Fourier de funciones periódicas

En primer lugar, Fourier entendía que **cualquier función del tiempo**  $f(t)$ :

- ▶ Real.
- ▶ Periódica de período  $T$ , si para todo  $t$  se cumple que  $f(t) = f(t + T)$
- ▶ Frecuencia continua:

$$\omega_1 = \frac{2\pi}{T}$$

Puede ser expresada mediante una serie infinita de funciones sinusoidales (senos y cosenos) de frecuencias  $\omega_n$ , múltiplos de  $\omega_1$ , es decir:

$$\omega_n = n \cdot \omega_1$$

Donde  $n = 1, 2, \dots, \infty$ .

Por lo que en forma de ecuación tenemos que:

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \cos(n \cdot \omega_1 t) + \sum_{n=1}^{\infty} B_n \sin(n \cdot \omega_1 t)$$

En resumen, la restricción inicial de que  $f(t)$  fuera una función periódica puede parecer realmente muy limitativa, pero lo cierto es que muchos fenómenos físicos poseen comportamientos periódicos, luego entender funciones periódicas complejas en términos de senos y cosenos era algo ciertamente avanzado, ya que las **funciones sinusoidales** tienen propiedades matemáticas muy positivas: continuidad, derivabilidad, perpendicularidad/ortogonalidad desde el punto de vista del producto escalar, etc.

Únicamente queda calcular los términos  $A_0$ ,  $A_n$  con  $n = 1, 2, \dots, \infty$  y  $B_n$  con  $n = 1, 2, \dots, \infty$ . Dichos términos tienen las siguientes relaciones (que no vamos a demostrar en este apartado):

$$\bar{f}(t) = \frac{1}{T} \int_{t_0}^{t_0+T} f(t) dt = A_0$$
$$B_n = \frac{2}{T} \int_0^T A_n = \frac{2}{T} \int_0^T f(t) \cos(n \cdot \omega_1 t) dt$$

La interpretación de estos tres términos es sencilla y, si has estudiado matemáticas, será más sencilla aún de entender:

- ▶ Fourier imaginó que toda función periódica se podría expresar como una combinación lineal (sumas, restas y coeficientes) de funciones sinusoidales.

- ▶ Dichas funciones sinusoidales son ortogonales entre sí, lo que quiere decir que las funciones están definidas de tal manera que no pueden expresarse como combinación lineales entre sí.
- ▶ Los coeficientes  $A_0$ ,  $A_n$  y  $B_n$  son el producto escalar de la función  $f(t)$  y la función sinusoidal correspondiente, en otras palabras, puede entenderse como la proyección de  $f(t)$  sobre las funciones sinusoidales.
- ▶ En otras palabras, cuanto más se parezca  $f(t)$  a una sinusoidal, mayor serán los coeficientes  $A$  y  $B$ .

Este último punto es de vital importancia. El producto escalar nos da una medida de cómo de parecidas son dos funciones, entre otras muchas propiedades. Con lo cual, aquellas componentes  $A_0$ ,  $A_n$  y  $B_n$  que sean más altos indicarán un mayor parecido con una sinusoidal.

Esto ya empieza a estar relacionado con lo que hemos hablado en la introducción, es decir, el poder encontrar una herramienta que nos diga qué componentes de frecuencia tiene una determinada señal, en este caso llamada  $f(t)$ .

Por último, y antes de continuar, únicamente decir que el gráfico de  $A_n$  y  $B_n$  en función de  $n$  o de  $\omega_n$  se conoce como el **espectro de frecuencias** de la función periódica  $f(t)$ .

## **Series de Fourier de funciones periódicas en el espacio complejo**

Cuando hacemos referencia al espacio complejo, Fourier quiere incluir funciones  $f(t)$  que tengan valores en el campo complejo, es decir, con el término  $j = \sqrt{-1}$ .

Evidentemente, no existe ninguna función que pueda medirse que dé un valor complejo, pero muchos fenómenos físicos se modelan con estos, principalmente los campos electromagnéticos, por lo que si se quiere extender las series de Fourier a esos fenómenos físicos, es necesario llevar las series de Fourier a este espacio.

Para entender dicha extensión, el alumno debe de estar familiarizado con la **fórmula de Euler**:

$$e^{j\theta} = \cos \theta + j \sin \theta$$

Para ello escribimos la serie de la siguiente manera:

$$f(t) = A_0 + \sum_{n=1}^{\infty} C_n \cos(n \cdot \omega_1 t + \phi_n)$$

Donde se cumplen las relaciones:

- $A_n = C_n \cos \varphi$ .
- $B_n = C_n \sin \varphi$ .

Finalmente,  $f(t)$  puede expresarse como suma de exponenciales complejas al obtener  $G_n$  a partir de  $f(t)$  usando la integral:

$$f(t) = A_0 + \sum_{n=1}^{\infty} G_n e^{jn \cdot \omega_1 t}$$
$$G_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-jn \cdot \omega_1 t} dt$$

En este tema no se va a justificar matemáticamente los pasos que conducen a esta demostración puesto que está fuera del enfoque de la asignatura. No obstante, recomendamos que intentes resolver esta demostración.

La interpretación de  $f(t)$  en función de las exponenciales complejas es de nuevo similar a las series de Fourier que vimos inicialmente: es la proyección de  $f(t)$  sobre dichas exponenciales complejas.

## Definición de la transformada de Fourier

Llegados a este punto, podemos preguntarnos para qué sirve la transformada de Fourier si ya tenemos los coeficientes  $A_n$  y  $B_n$ , que ya nos indican las componentes espectrales de  $f(t)$ . Muy sencillo.

**La transformada de Fourier sirve para extender la serie de Fourier a funciones no periódicas (periódicas con período infinito).**

En realidad es un proceso límite si juzgamos que la función no periódica la aproximamos a una que se repite a una distancia (tiempo) muy lejana de nosotros.

En definitiva, las expresiones son las siguientes:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$
$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega$$

Donde:

- ▶  $F(\omega)$  representa la transformada de Fourier.
- ▶  $f(t)$  se representa como la inversa de la transformada de Fourier.

La reflexión que aplica ahora es: ¿por qué  $F(\omega)$  es función de  $\omega$ ? ¿Qué es  $\omega$ ? Desde ahora en adelante (y en toda la literatura que existe sobre este tema), se utiliza  $\omega$  como **variable proporcional a la frecuencia**:

$$\omega = 2\pi f$$

Donde  $f$  es la frecuencia.

El uso de  $\omega$  queda justificado debido a la nomenclatura y no confundir la  $F$  de Fourier con la  $f$  de frecuencia. En otras palabras,  $\omega$  es la **variable de la transformada**, que ya no tiene información sobre el tiempo ( $t$ ). Nótese que la única variable  $t$  es la variable de integración que desaparece cuando la integral se resuelve.

Y además, de todo esto se deduce que no se pierde ningún tipo de información cuando se hace la transformada.

Una de las preguntas más comunes cuando se estudia la transformada de Fourier es si existe algún dispositivo que permita verla. La respuesta rápida es no. Debes recordar que la transformada de Fourier,  $F(\omega)$ , está en el espacio complejo y es solo una entelequia matemática. Lo único que puede hacerse es una aproximación del módulo de la transformada,  $|F(\omega)|$ , mediante computación (se verán algunos ejemplos).

De hecho, gracias a la computación podemos obtener representaciones de la transformada de Fourier muy interesantes, como esta que vemos a continuación. En ella puede apreciarse cómo las **componentes temporales se transforman en componentes frecuenciales**.

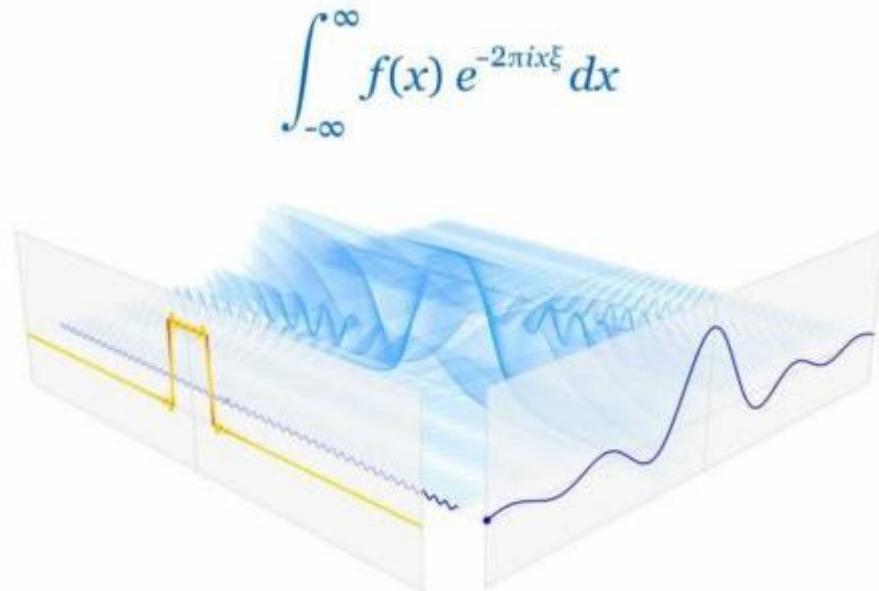


Figura 6. Vista simultánea de la función pulso (en amarillo) y su transformada de Fourier (azul).

Fuente: <http://puyaa.ir/fourier-analysis-nutshell/>

## Ejemplos de transformada de Fourier

A continuación exponemos algunos ejemplos de transformada de Fourier:

PARES DE TRANSFORMADAS DE FOURIER			
Función ( $t$ )		Función ( $w$ )	
$\delta(t)$		1	
1		$2\pi\delta(w)$	
$w(t)$		$\pi\delta(w) + \frac{1}{fw}$	
$\Pi(t/\tau)$		$\tau \frac{\sin(\omega t/2)}{\omega t/2}$	
$\cos(\omega_0 t)$		$\pi[\delta(w - \omega_0) + \delta(w + \omega_0)]$	
$\sin(\omega_0 t)$		$j\pi[\delta(w + \omega_0) - \delta(w - \omega_0)]$	

Figura 7. Tabla con los pares de transformadas más comunes.

Fuente: <http://chismesycircuitos.blogspot.com.es/2011/09/tabla-transformadas-de-fourier-ut-1-del.html>

De esta tabla, solo vamos a reflexionar sobre dos funciones: la función constante y las señales sinusoidales.

**Funciones de valor constante.** Desde el punto de vista de percepción computacional, un valor constante puede ser ruido o la componente continua de una señal unidimensional. Con respecto a estas funciones de valor constante, se aprecia como la energía se concentra en el dominio de la frecuencia entorno a la frecuencia cero.

- ▶ Hasta cierto punto parece lógico, ya que una señal continua no posee frecuencia alguna.
- ▶ Esto quiere decir que, en el dominio de la frecuencia, las componentes constantes en el tiempo están muy cercanas a cero y, por lo tanto, serán fácilmente eliminables por filtros.

**Funciones sinusoidales.** Tonos puros de sonido, como puede ser el tono producido por una nota en un instrumento musical. En estas se observa que se transforman en otras dos funciones denominadas **delta de Dirac**, las cuales indican que la función solo está en un determinado punto.

- ▶ Lógico si pensamos que la transformada de Fourier mide las componentes frecuenciales de una señal y, si esta es un tono puro, solo podrá tener un determinado valor en frecuencia.
- ▶ En la tabla se aprecian dos deltas a ambos lados de los ejes horizontales. Por ahora, no debes plantearte a qué se debe este fenómeno, pero es el reflejo de que se está trabajando en el dominio complejo de la frecuencia y, aunque físicamente no existen frecuencias negativas, matemáticamente sí son posibles.

Veamos ahora dos ejemplos con datos reales, uno para voz y otro para imagen.

**La transformada de Fourier de voz** suele ocupar unos 4 KHz (ancho de banda de la voz). Es decir, cuando se hace la transformada de Fourier de cualquier fichero de sonido donde hable una persona, nunca (o muy difícilmente) se superarán los 4 KHz.

Esto puede verse en el siguiente ejemplo donde en rojo vemos la señal de voz (un segundo de duración) y en azul, la transformada de Fourier calculada con uno de los algoritmos más importantes (*Fast Fourier Transform*, FFT). La transformada de Fourier puede variar si, en vez de tomar un segundo de duración de voz, usamos una hora, pero en ningún caso variará significativamente y casi nunca superará esos 4 KHz de ancho de banda.

De hecho (se verá mejor en la parte de filtrado basado en Fourier), si quisiésemos eliminar ruidos que no fueran voz, tendríamos que eliminar las componentes superiores a 4 KHz y oiríamos la voz nítidamente. Esto es lo que hacen nuestros dispositivos móviles cuando hablamos: filtran todas las componentes por encima de un valor y solo transmiten frecuencias menores que 4KHz.

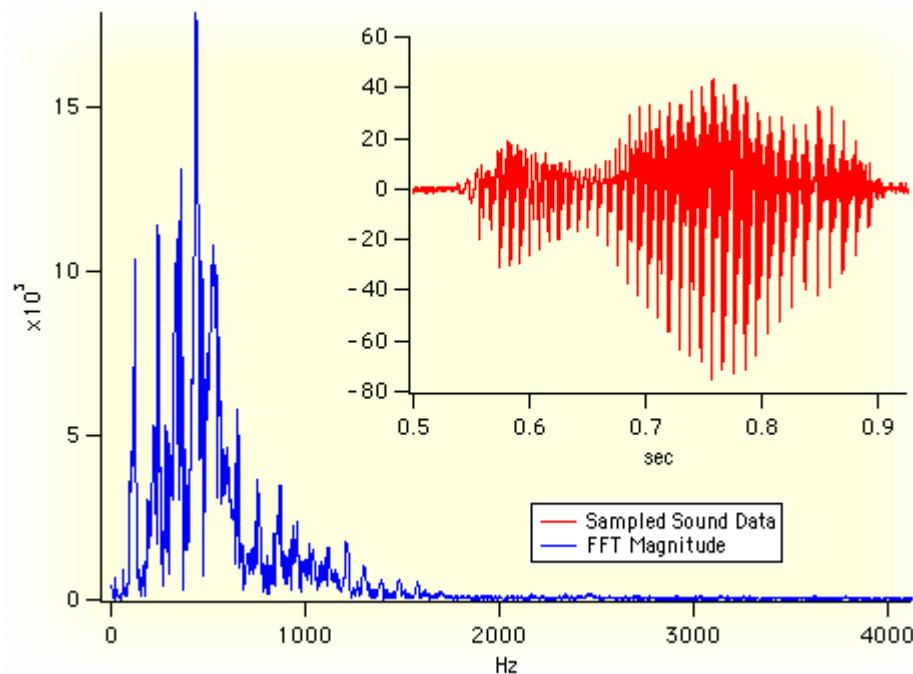


Figura 8. Ejemplo de transformación de Fourier.

Fuente: <http://www.wavemetrics.com/index.html>

**La transformada de Fourier de imagen.** Con respecto a las imágenes, la transformación es más difícil de interpretar, pero aún sigue siendo válida.

Primero, se presentan ejemplos con frecuencias puras para que se aprecie cómo se aplicaría Fourier:

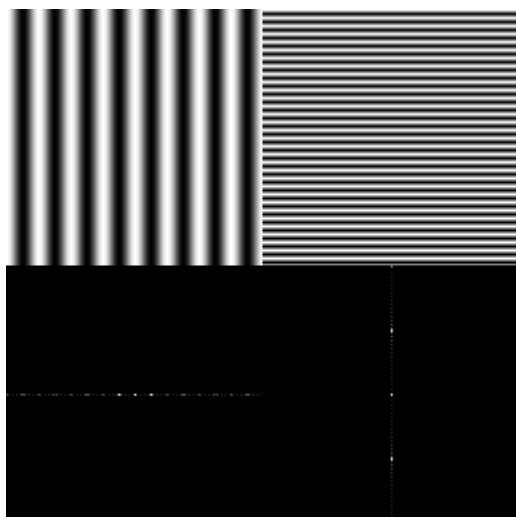


Figura 9. Transformada de Fourier de patrones de imágenes puros (rayas verticales y horizontales).

Fuente: <https://www.cs.unm.edu/~brayer/vision/fourier.html>

En sí misma, la transformada de Fourier puede ofrecer también la posibilidad de proporcionar características que identifiquen de forma unívoca qué está visualizándose en la imagen. Un ejemplo claro es el de los caracteres escritos a mano. En la siguiente figura, las componentes en frecuencia varían en función de los trazos. Las más circulares, como la «B» y la «Q», ofrecen componentes en frecuencia completamente diferentes de otras letras como la «T» o la «K».

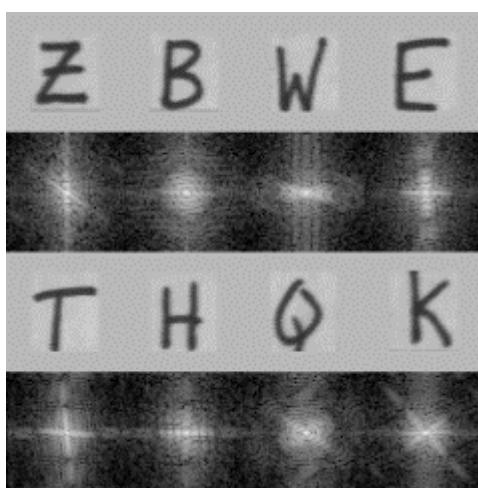


Figura 10. Transformada de Fourier de caracteres escritos a mano.

Fuente: <https://www.cs.unm.edu/~brayer/vision/fourier.html>

Como hemos explicado, se aprecia cómo las componentes en frecuencia son diferentes y permitirían una extracción de características diferente para poder distinguir las letras.

Por último, un uso de la transformada de Fourier es eliminar o detectar el ruido que existe en la imagen. La transformada de Fourier se ve afectada al desenfocarse la imagen: las componentes espectrales varían, ya que se pierde nitidez, y el ruido aparece muy marcado en una de las componentes verticales, donde aparece una raya más blanca que pasa por el origen.

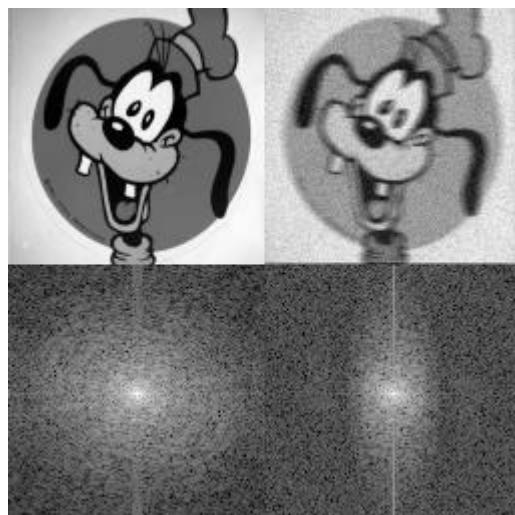


Figura 11. Efecto en la transformada de Fourier de desenfocar la imagen y aplicar ruido en ella.

Fuente: <https://www.cs.unm.edu/~brayer/vision/fourier.html>

## 9.5. Transformada discreta de Fourier (DFT) y su implementación mediante *Fast Fourier Transform* (FFT)

**T**al como hemos visto antes, la transformada de Fourier de  $f(t)$  está pensada para señales continuas, es decir, para las que no han sido discretizadas o no han sufrido un proceso de conversión analógico-discreto.

Sin embargo, si se quiere aplicar la transformada de Fourier a señales discretizadas (digitales), hay que realizar algunas modificaciones en la definición y es por eso que haremos especial hincapié.

Además, la transformada discreta de Fourier (DFT, *Discrete Fourier Transform*) ha recibido gran atención por parte de matemáticos e ingenieros para realizar su implementación; la FFT (*Fast Fourier Transform*) es la mejor aproximación que existe actualmente, desde luego la más eficiente y que mejores resultados tiene. Cabe decir que la mayoría de lenguajes de programación poseen una implementación de la FFT.

En este apartado no vamos a abordar una comprensión matemática de la DFT, ya que conlleva un conocimiento de complejos y teoría matemática más avanzada que con la transformada continua de Fourier.

En este caso, para que tengas una visión global de la DFT únicamente diremos que:

- ▶ Ya no se habla de  $f(t)$ , sino de **secuencias**.
- ▶ Se representa mediante las variables  $x, y$  o  $z$ .
- ▶ Las variables  $n$  o  $m$  (si se quiere hacer distinción entre ambas) entre **corchetes**, no paréntesis.

- ▶ Por lo que dada una secuencia discreta  $x[n]$  de longitud N, su **transformada discreta de Fourier** viene dada por la siguiente expresión:

$$X_K = \sum_{n=0}^{N-1} x[n] e^{-2\pi j k \frac{n}{N}}$$

Intuitivamente, se parece a la transformada de Fourier, pero posee algunas distinciones:

- ▶ El sumatorio está acotado entre 0 y N-1.
- ▶ La secuencia  $x[n]$  puede no ser periódica, pero será de longitud N.
- ▶ No existe el concepto de frecuencia, sino que existe la variable  $k$ . Dicha variable provoca que la DFT sea también discreta y tenga los mismos puntos que la secuencia original N.

Aquí vemos dos ejemplos que facilitan la comprensión de la teoría. En primer lugar, se ve una esquematización de la DFT y su inversa. Hay que tener en cuenta que, en algunos libros, el factor  $\frac{1}{N}$  se incluye en la propia DFT. Nosotros preferimos dejar ese factor para la **transformada inversa**, de ahí que no haya salido en la formulación anterior de la DFT.

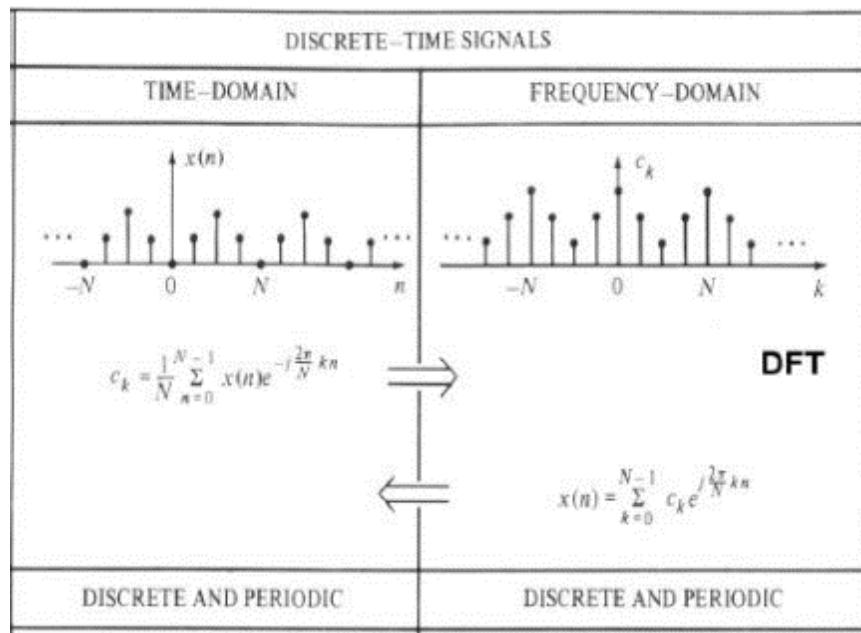


Figura 12. Comparación de la transformada discreta de Fourier entre la secuencia original y su propia transformada. En este caso se trata de una señal periódica.

Fuente: <http://kcchao.wikidot.com/dft>

En segundo lugar, un ejemplo de resolución de la DFT. Es evidente que el número de muestras de una secuencia, así como el número de muestras elegido para dar resolución a la DFT, afecta a la calidad de la transformada. En este caso, vemos un ejemplo con la función  $\cos \frac{\pi n}{2}$ , representando el módulo de la DFT. Se puede apreciar cómo la variación de los números de puntos empleados influye en la resolución de la transformada. De hecho, existe un número óptimo para evitar que aparezca ruido en la transformada.

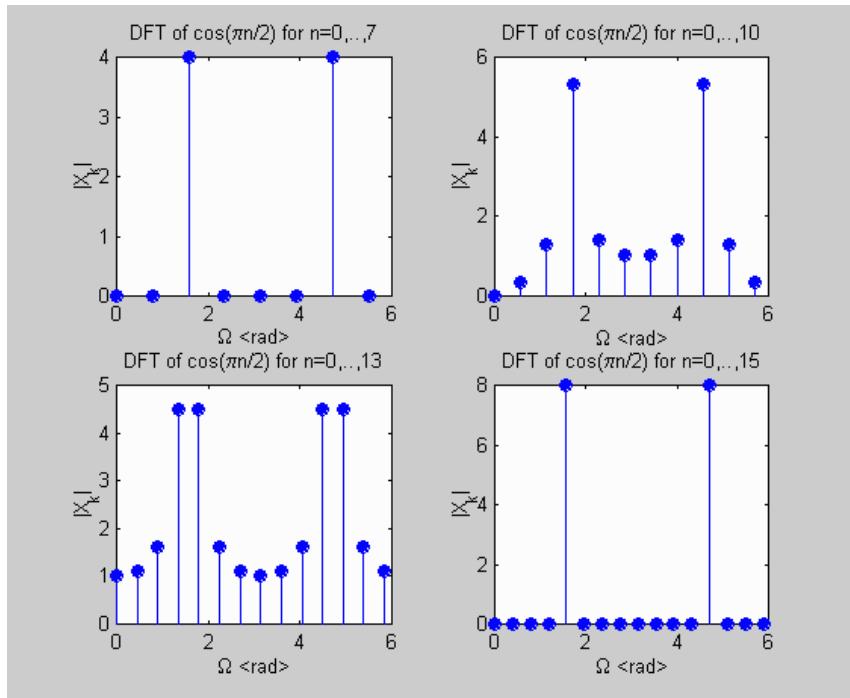


Figura 13. Comparación de la transformada discreta de Fourier entre la secuencia original y su propia transformada. En este caso se trata de una señal periódica.

Fuente: <http://www.ee.nmt.edu/~wedeward/EE342/SP99/example16.html>

Finalmente, no queríamos dejar de hablar sobre el esquema general de la transformada FFT. No deja de ser una aproximación que impone ciertas limitaciones, entre ellas que las duraciones de las secuencias deben de ser múltiplos de 2 para facilitar el cálculo. Sin embargo, la **FFT reduce la complejidad** de  $O(n^2)$  a  $O(n \log n)$ , lo cual es un avance notable.

No existe una implementación única de la FFT. La original, creada por Cooley y Tukey de IBM en 1960 y revisada por C. S. Burrus de la Universidad Rice University, permitió procesar voz en tarjetas inteligentes para dispositivos con capacidades muy limitadas, pero actualmente existe una gran cantidad de implementaciones, todas ellas en función del problema que se pretenda resolver: más rapidez, más precisión, menos consumo, etc.

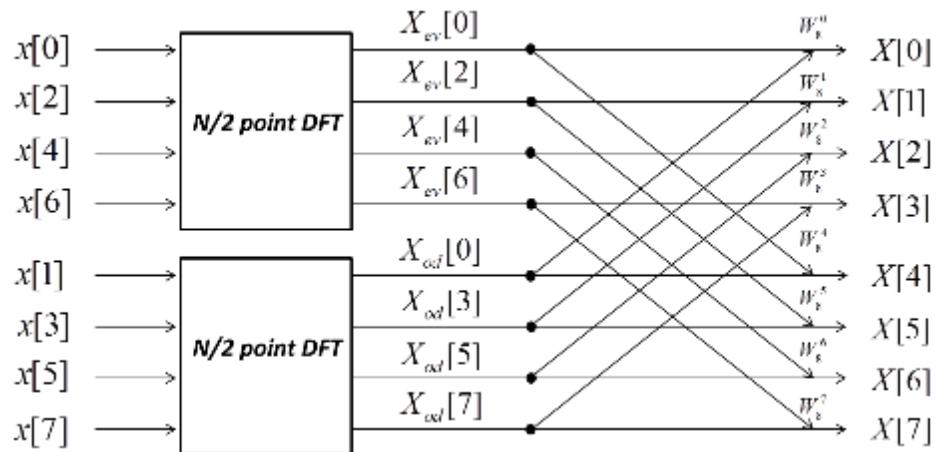


Figura 14. Funcionamiento de la FFT y ahorro de costes al calcular de forma paralela diferentes coeficientes, siempre y cuando las señales tengan una duración potencia de 2.

Fuente: <http://www.vocal.com/noise-reduction/fft-algorithms/>

Percepción Computacional

---

# Procesamiento de imagen. Crecimiento de regiones

## 10.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

## 10.2. Segmentación y crecimiento de regiones

**C**omo se ha visto en temas anteriores, la segmentación consiste en dividir una determinada imagen en regiones/segmentos de propiedades similares. Esas propiedades pueden venir definidas bien por el color, la textura o incluso por el hecho de estar acotados por contornos muy bien definidos.

El crecimiento de regiones es un algoritmo generalmente **no supervisado** (existe posibilidad de aplicar supervisión, pero no es lo habitual) que, partiendo de unas regiones concretas en la imagen, normalmente elegidas por una persona, encuentra los límites de dicha región donde el punto se encuentra.

En otras palabras, dada una región inicial, el algoritmo explora en la vecindad de dicha región y considera que aquellos píxeles más parecidos son de la misma región.

A continuación, un ejemplo de crecimiento de regiones. Inicialmente se parte de un punto y se expande alrededor de dicho punto hasta que la región encontrada empieza a no ser uniforme y el algoritmo se detiene.



Figura 1. Ejemplo de crecimiento de regiones.

Por supuesto, este enfoque posee ventajas y desventajas. Dentro de las **desventajas** encontramos el hecho de que necesita una buena inicialización, es decir, requiere de intervención humana que indique dónde comenzar a crecer.

De hecho, este tipo de algoritmos se utiliza en biomedicina: imágenes biomédicas como ecografías, radiografías, etc., donde el médico indica qué región le interesa y el algoritmo encuentra las mejores fronteras de cada región.

Adicionalmente, como desventaja, nos encontramos con que este tipo de algoritmos son computacionalmente muy costosos por dos motivos:

**Son algoritmos iterativos y no paralelizables.** Es decir, hasta que no acaba una iteración no puede comenzar la siguiente.

Las **funciones** que exploran si un píxel debe pertenecer o no a una región son **complejas de evaluar y costosas de implementar**, lo que hace que estos algoritmos sean dependientes de estas funciones de evaluación.

Como **ventaja** se encuentra el hecho de que el algoritmo, hasta cierto punto, es autónomo para encontrar las fronteras de las regiones de interés. Esta **autonomía** está basada en funciones de comparación y en funciones de maximización y minimización. Si esas funciones no están correctamente parametrizadas o definidas, el algoritmo nunca será capaz de encontrar una segmentación adecuada.

## 10.3. Técnicas empleadas en el crecimiento de regiones

E

xisten dos familias de técnicas para el crecimiento de regiones:

- ▶ Aquellas técnicas que se encargan de resolver el problema **partiendo de píxeles singulares** y encontrando regiones más globales. Llamadas *bottom-up*: parten del mayor grado de detalle (el píxel) y agrupan en función de él. Podemos decir que existe una división también dependiendo de cómo se realice la agregación, lo que da lugar a dos tipos de sub-familias:
  - Técnicas basadas en **agregación local**, es decir, para agregar únicamente nos fijamos en los vecinos más cercanos del píxel en cuestión (o de la región que se está agregando).
  - Técnicas basadas en agregación **global**, donde se tiene una visión global de toda la imagen y dos píxeles solo se unen bajo una misma región siempre y cuando a nivel global tenga sentido dicha agregación.
- ▶ Aquellas que se encargan de dividir la imagen en **secciones más pequeñas** de forma iterativa, viendo hasta qué punto dos regiones pueden unirse. Este algoritmo es conocido como *split and merge* y a estas técnicas también se denominan *top-down*.



Figura 2. Resultado de un algoritmo de crecimiento de regiones (zona roja).

Fuente: <https://es.mathworks.com/matlabcentral/fileexchange/19084-region-growing>

Como hemos comentado, el uso de este tipo de algoritmos está muy extendido en biomedicina y segmentación de imágenes médicas, donde no existe la posibilidad de obtener colores.

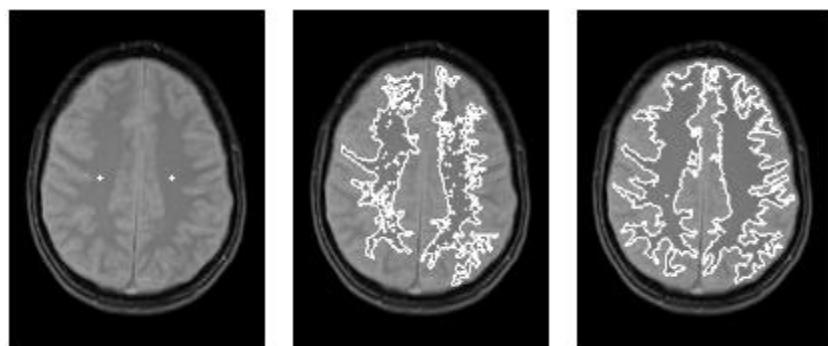


Figura 3. Evolución de un algoritmo *bottom-up* de crecimiento de regiones.

Fuente: <https://www.creatis.insa-lyon.fr/~grenier/?cat=8>



Figura 4. Ejemplo de algoritmo *split and merge*.

Fuente: <https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/hat/>

En esta última imagen vemos que, partiendo de pequeñas divisiones de la imagen (*split*) se decide unir (*merge*) aquellas regiones con mayor parecido.

## 10.4. Crecimiento de regiones basado en semillas

**E**l crecimiento basado en semillas es el más intuitivo y para ello hacen falta los siguientes ingredientes, que serán comunes al resto de métodos de crecimiento de regiones.

Inicialmente asumiremos que una semilla vendrá dada por un píxel. Otras alternativas contemplan el uso de una forma bien definida por el usuario de forma interactiva sobre la imagen, bien ya predefinida: un círculo, cuadrado, etc.

Por lo tanto, para trabajar con este método basado en semillas necesitamos:

- ▶ Un conjunto de píxeles:  $p_1, p_2, \dots, p_n$  que servirán como semillas.
- ▶ La **posición** de esos píxeles vendrá dada por la propia posición que ocupe dentro de la imagen, y las **propiedades** de cada píxel, por la función  $I(p_i)$  donde se elige el nombre  $I$ , ya que comúnmente será la intensidad de color de dicho píxel. Otros valores de la función pueden ser la saturación u otras codificaciones de color que no sean las dadas por el criterio RGB.

- ▶ Una **función de similaridad** entre dos píxeles  $p_i$  y  $p_j$ . Esta similaridad vendrá dada por  $S(p_i, p_j)$  y podrá aplicarse bien entre píxeles o bien entre regiones como posteriormente veremos.
- ▶ Una **función de pertenencia** a una región; dada  $\pi_r(p_i)$  donde  $r$  hace referencia a la región en concreto y  $p_i$  a un píxel cualquiera. Esta función es 1 si  $S(r, p_i) > T$ , es decir, si la función de similaridad supera un cierto umbral. En cualquier otro caso, será 0. Dicho umbral se fijará de forma automática por el algoritmo o bien de forma manual, si así lo decide el usuario final.

Con lo cual, un algoritmo de crecimiento de regiones basado en semillas lo que intenta es encontrar el número mínimo de regiones que maximiza todas y cada una de las funciones de similaridad y de pertenencia dentro de cada región.

Este **proceso iterativo** posee las siguientes características:

- ▶ Al ser iterativo, es un proceso que posee gran coste computacional y cuya optimización debe realizarse, en muchos casos, de forma secuencial y no paralela.
- ▶ Solo considera regiones adyacentes, por lo que si dos píxeles no adyacentes, por función de similaridad  $S$  y de pertenencia pudieran pertenecer a una misma región, no serán unidos dentro de una misma región.
  - Esto conduce en la gran mayoría de casos a una **sobresegmentación**: hay un número elevado de segmentos/grupos dentro de la imagen, y por ende, otro postprocesado para eliminar dicha sobresegmentación.
- ▶ Cada píxel tiene que tener una región, así que en muchos casos, algunos píxeles son forzados a pertenecer a una determinada región simplemente por estar en su adyacencia.
- ▶ Si la región a detectar no es homogénea y posee geometrías complicadas, este método no proporcionará un resultado preciso debido a que el algoritmo tenderá a crecer en todas direcciones, respetando la vecindad del píxel.
- ▶ Sin embargo, por su sencillez e interactividad permite encontrar una primera aproximación de la segmentación muy certera en aquellas imágenes donde el contraste sea muy alto.

- Además, suele servir como origen para otros algoritmos de crecimientos de regiones más específicos y complejos.

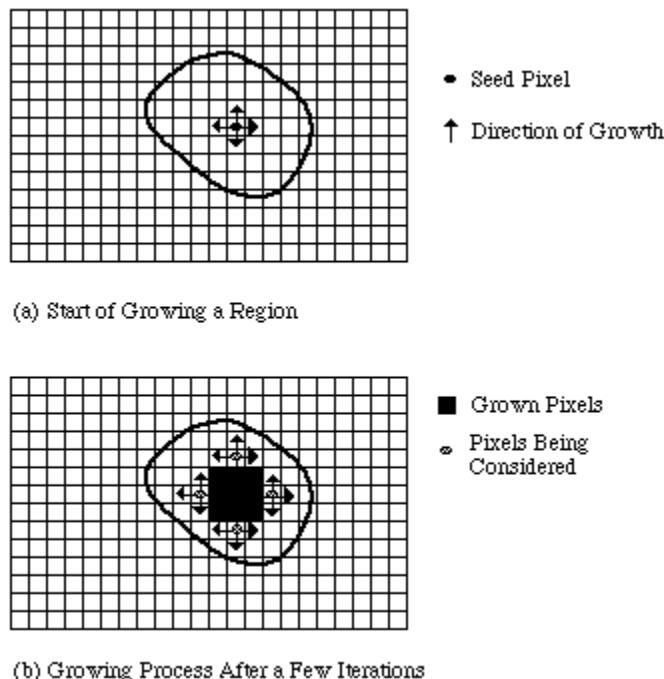


Figura 5. Comienzo del algoritmo de crecimiento de regiones basado en semillas. La dirección de crecimiento suele ser otro parámetro del algoritmo.

Fuente: [https://users.cs.cf.ac.uk/Dave.Marshall/Vision\\_lecture/node35.html](https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node35.html)

## 10.5. Crecimiento de regiones basado en *Split and Merge*

**P**ara explicar el algoritmo basado en *split and merge* voy a usar el siguiente esquema y cuatro subimágenes.

1	IMAGEN ORIGINAL $i$	Esta imagen es considerada en este método (y en este paso) como un segmento en sí mismo. La idea es ver si se pueden encontrar segmentos con funciones de similaridad y pertenencia más precisas.
2	DIVISIÓN EN CUATRO PARTES	Se divide la imagen en cuatro partes, evaluándose las funciones de similaridad y pertenencia en cada una. Se eligen cuatro en función del diseñador del algoritmo, aunque pueden escogerse el número de particiones que se desee. Para ser más óptimos computacionalmente, se suele dividir en cuatro y cada una a su vez en otras cuatro subpartes.

Figura 6. Pasos a seguir con el algoritmo *Split and Merge*.

En este ejemplo, únicamente el subconjunto  $I_4$  no verifica las condiciones de similaridad y pertenencia. Esto quiere decir que la entropía dentro de dicho segmento o la variabilidad de intensidades hace que esté por debajo de un determinado umbral y por lo tanto haya que realizar de nuevo el proceso de dividir la imagen en cuatro partes. En este caso, únicamente en el segmento  $I_4$ .

De las cuatro subimágenes que se han creado, los segmentos  $I_{43}$  e  $I_{44}$  tienen propiedades similares y por lo tanto se fusionan (*merge*). ¿Por qué se dividieron si después hay que fusionarlos? Simplemente por el funcionamiento del algoritmo. Primero realiza divisiones de forma automática, en este caso en cuatro, y posteriormente evalúa si dos segmentos deberían de seguir juntos o no.

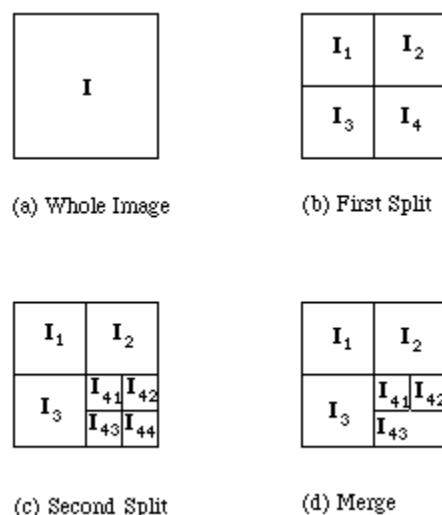


Figura 7. Esquema de pasos principales en el algoritmo *split and merge*.  
Fuente: [https://users.cs.cf.ac.uk/Dave.Marshall/Vision\\_lecture/node34.html](https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node34.html)

En la figura 7 vemos entonces que:

- ▶ La imagen original, todos los píxeles poseen la misma importancia (a).
- ▶ Se divide la imagen en partes iguales (b).
- ▶ Únicamente el segmento  $I_4$  se subdivide en más subelementos debido a que dicho segmento es el menos uniforme (c).
- ▶ Los segmentos  $I_{43}$  e  $I_{44}$  se fusionan (*merge*) en uno debido a que tienen propiedades similares (d).

Este proceso de *split and merge* es «paralelizable», ya que el análisis se puede ejecutar de forma simultánea para los segmentos  $I_1$ ,  $I_2$ ,  $I_3$  e  $I_4$ .

Además, el coste computacional de este algoritmo va decreciendo a medida que va avanzando, ya que los subconjuntos/segmentos cada vez son más pequeños en tamaño y hay menor número. Teóricamente se puede asegurar que el coste computacional de este algoritmo está acotado.

Para verlo más fácilmente, pondré un ejemplo con una imagen de 16x16 píxeles. En ella puede apreciarse que ya de por sí existen regiones con valores muy similares (valores iguales a 7).

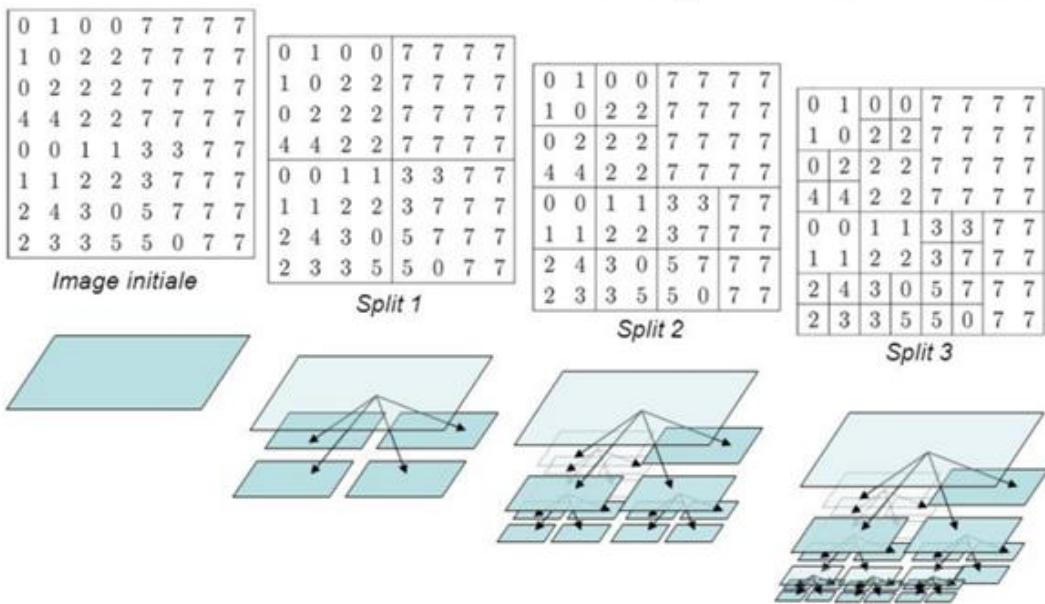


Figura 8. Máscaras correspondientes a los operadores de Prewitt para la identificación de bordes con orientación horizontal y vertical.

Fuente: <http://slideplayer.fr/slide/518235/>

- ▶ En la primera iteración (*Split 1*), se divide la imagen también en cuatro partes. De todas ellas, hay una que ya no se dividirá más puesto que todos los píxeles son iguales entre sí.
- ▶ Del resto de regiones, se dividen de nuevo por la mitad y en una tercera división se aprecia cómo algunos subconjuntos con valores similares no se han dividido de nuevo. Es aquí donde entra en juego el umbral que veíamos en la segmentación basada en crecimiento de regiones con semillas. Si ese umbral se disminuye, el algoritmo es más estricto y por lo tanto producirá sobresegmentación.
- ▶ En paralelo, se puede ver cómo se va construyendo un grafo jerárquico que explica cómo la imagen se ha ido dividiendo y cómo cada segmento contiene varios subsegmentos. Ese grafo se utiliza en otras técnicas (como graph-cuts) para poder segmentar de una manera más precisa e incluso para poder unir segmentos parecidos.

Por último, es importante indicar que aunque el algoritmo *split and merge* es uno de los más empleados, tiene una **desventaja** muy importante y es que, una vez que dos segmentos han sido separados, jerárquicamente están separados en ramas diferentes, el algoritmo no evaluará si pueden ser fusionados en un mismo segmento.

Este proceso debe hacerse de forma posterior en un postprocesado que ha de recorrer todos los segmentos y evaluar la función de similaridad y pertenencia. Sin embargo, aunque esto puede parecer costoso, como el número de segmentos es pequeño, puede realizarse sin problema.

## 10.6. Crecimiento de regiones basado en *Gradient Vector Flow* (GVF)

**L**os métodos conocidos como *Gradient Vector Flow* (GVF en adelante) o *Active Contour Models* (ACM) suelen estar clasificados como métodos de segmentación basados en modelos u orientados a modelo y contorno. Sin embargo, pueden considerarse como métodos de crecimiento de regiones en un ámbito más amplio.

Esta familia de algoritmos se conoce popularmente como el **algoritmo de la serpiente**, puesto que simula el comportamiento de este animal a la hora de ir encontrando el contorno deseado.

Este tipo de métodos se basa en ir modificando el contorno de una determinada figura base (inicialmente suele ser un círculo) e irla deformando de forma iterativa hasta que se adapta a la forma del objeto o región que se quiere segmentar.

En este sentido, son muy dependientes de la inicialización y de las funciones de coste o de optimización, es decir, el criterio que siguen estas curvas para readaptarse al contorno buscado.

A continuación se presentan tres ejemplos de cómo funciona visualmente este algoritmo.

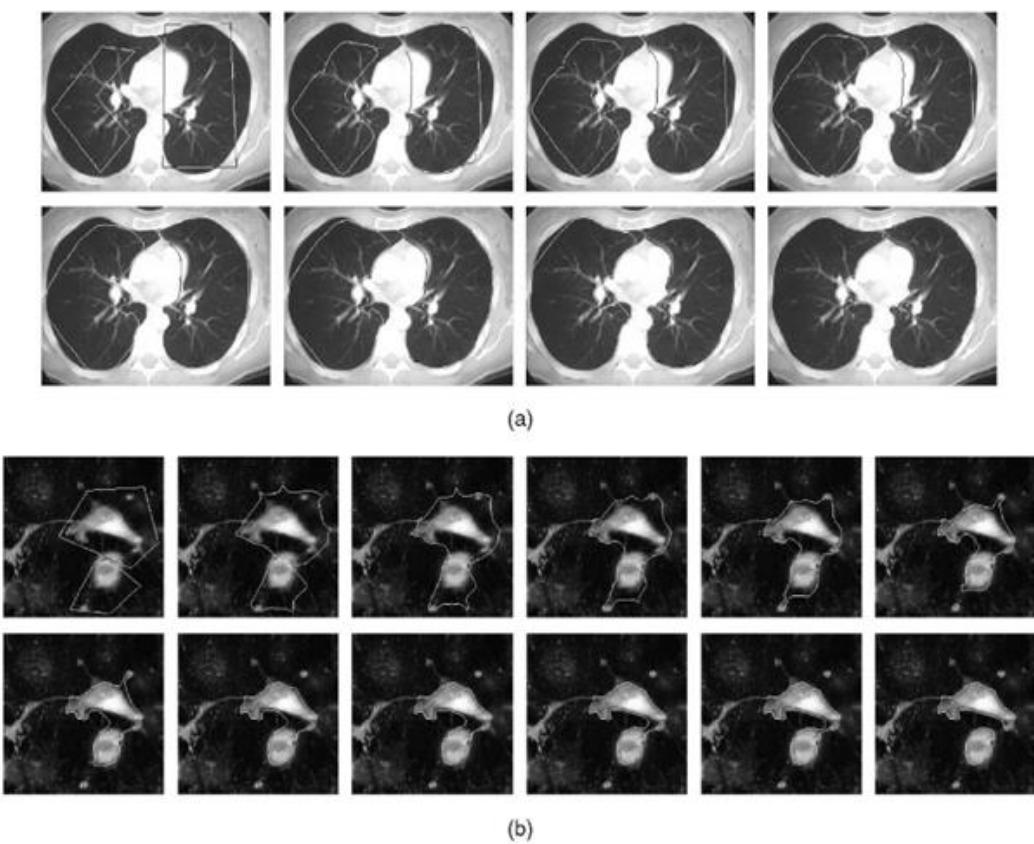


Figura 9. Funcionamiento del algoritmo GVF.

Fuente: Paragios y Ramesh, 2004.

En la figura 9, (a) y (b) son imágenes biomédicas donde se aprecia como el algoritmo comienza con un contorno inicial y dicho contorno se va adaptando a la forma a segmentar.

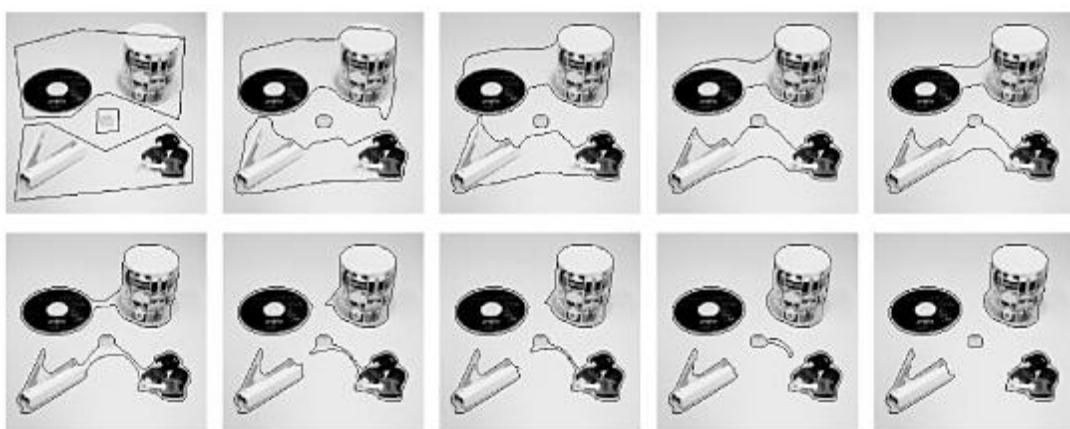


Figura 10. Funcionamiento del algoritmo GVF para la segmentación de varios objetos.

Fuente: Paragios y Ramesh, 2004.

En este último ejemplo, la segmentación es relativamente sencilla debido a que el fondo de la imagen está muy diferenciado.

No obstante, aunque GVF y ACM suelen ser sinónimos en la literatura, se puede decir que GVF es una extensión de ACM o algoritmos de serpiente. La **diferencia** más significativa radica en que GVF convergen en las concavidades del contorno y no necesitan una inicialización tan cercana al contorno como en el caso de los algoritmos ACM.

De hecho, veamos la diferencia de forma matemática. El algoritmo de serpiente original ( $v$ ) es un contorno bidimensional dinámico definido de forma paramétrica como  $v(s) = [x(s), y(s)]$  donde  $s \in [0,1]$  y que minimiza la siguiente función de energía:

$$E = \int_0^1 E_{int}(v(s)) + E_{Imagen}(v(s)) + E_{con}(v(s)) ds$$

Donde:

- ▶  $E_{int}$  denota la energía del contorno interior.
- ▶  $E_{Imagen}$  representa la energía debida a la intensidad propia de la imagen.
- ▶ Y  $E_{con}$  es una función de contorno que modela la parte exterior del contorno.

La definición de estas energías en modo de función queda fuera del ámbito de este tema, pero en literatura pueden encontrarse numerosos ejemplos de funciones para estas energías.

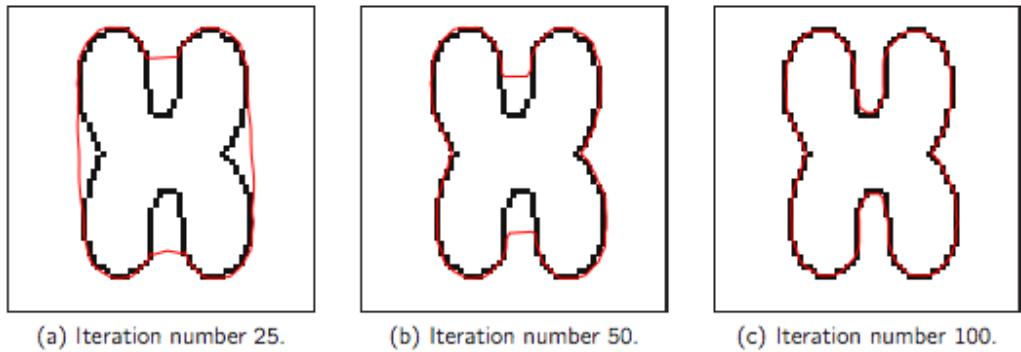


Figura 11. Esquema de funcionamiento del algoritmo GVF.

Fuente: Cartas-Ayala, s. f.

En el caso de los métodos GVF, y aquí radica la diferencia con los algoritmos de serpiente, la función de energía  $E_{con}$  utiliza una superficie paramétrica en vez de una curva, es decir, es función de  $[u(x, y), v(x, y)]$  y no solo de  $v(s)$ .

Recomendamos leer lo siguiente:

- ▶ Kass, M., Witkin, A. y Terzopoulos, D. (1987). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4), 321–331. Recuperado de <https://link.springer.com/article/10.1007/BF00133570>
- ▶ Leroy, B., Herlin, I. y Cohen, L. (1996). Multi-resolution algorithms for active contour models. En M. O. Berger et al. (Eds.). *ICAOS '96. Lecture Notes in Control and Information Sciences*, 219. Heidelberg: Springer Books. Recuperado de [https://link.springer.com/chapter/10.1007/3-540-76076-8\\_117](https://link.springer.com/chapter/10.1007/3-540-76076-8_117)

Por último, presentamos una tabla comparativa entre GVF y ACM donde se presentan las ventajas y desventajas de ambos.

GVF vs ACM	Curva inicial	Curva final
El algoritmo de serpiente tradicional debe de inicializarse con un contorno muy similar al objeto a segmentar y aun así, no converge en las concavidades.		
El algoritmo GVF puede empezar muy alejado del contorno a segmentar, pero será capaz de converger en las concavidades.		
Incluso, si la inicialización de GVF atraviesa el contorno, será capaz de converger, a diferencia de ACM.		
GVF converge a contornos definidos de forma subjetiva, es decir, se aprecian a nivel visual, pero no están claramente definidos.		

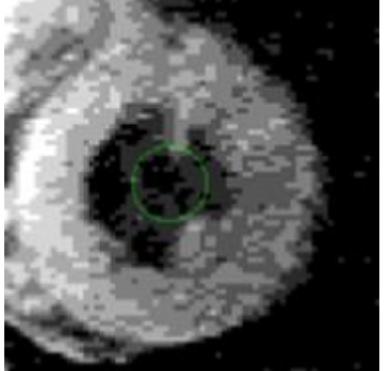
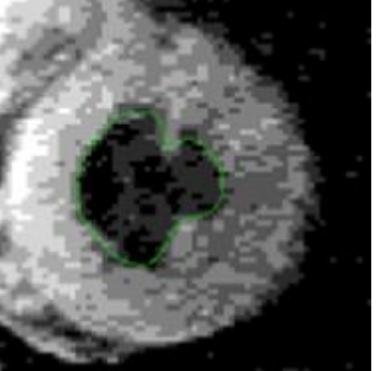
<p>GVF puede trabajar también con escalas de grises (aquí se aprecia cómo el algoritmo se adapta para encontrar las paredes del corazón en una imagen de resonancia magnética)</p>		
--	---	--

Tabla 1. Comparación entre GVF y ACM.

Fuente: Adaptado de <http://www.iidl.ece.jhu.edu/static/gvf/>

## 10.7. Crecimiento de regiones basado en Watershed

La transformada Watershed o segmentación basada en Watershed suele estar englobada dentro de la morfología matemática. Sin embargo, la motivación para ubicarla dentro de este tema se debe a que su funcionamiento está más cercano al crecimiento de regiones que a la propia morfología matemática.

La segmentación basada en Watershed se conoce popularmente como **algoritmos de inundación**, ya que utilizan el símil de la inundación de pantanos (o superficies) para explicar muy visualmente el algoritmo.

**Con respecto a otras técnicas de segmentación, el objetivo de esta técnica es dividir en regiones la imagen de nivel de grises.**

Generalmente una de las regiones se corresponde con el fondo de la imagen, que puede contener otros objetos fuera de interés, y el resto con los objetos o regiones

que se pretende extraer. El objetivo último de esta técnica es **determinar los contornos** que definen dichos objetos.

El punto de partida es considerar que los contornos de una imagen se corresponden con las líneas donde el nivel de gris varía más rápidamente que en un determinado entorno vecino. En este punto, hay que hacer uso del **gradiente de la imagen**, que proporciona la posición de los píxeles donde la variación de intensidad es más pronunciada, muy similar al concepto de derivada matemática.

El uso de dicho gradiente nos permite asumir que los contornos de la imagen original se corresponden con las líneas de cresta de la imagen gradiente.

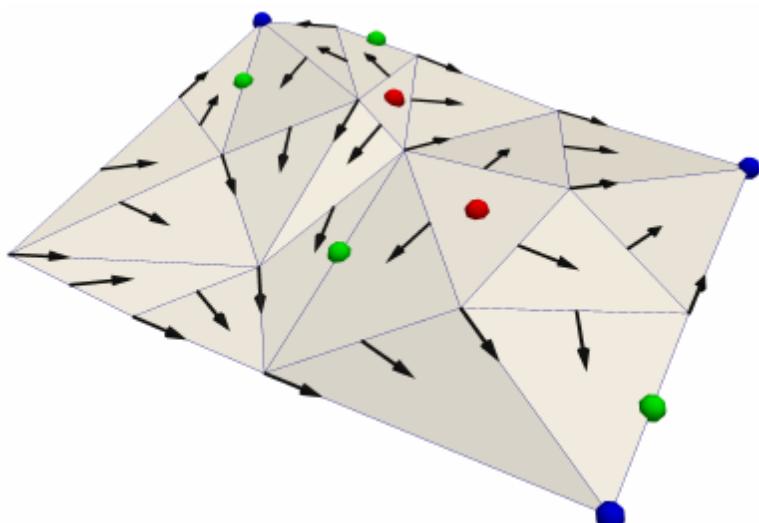


Figura 12. Ejemplo visual de gradiente de una imagen.

Fuente: [https://www.researchgate.net/figure/Example-of-a-discrete-gradient-vector-field-on-a-triangulated-terrain-Paired-simplices\\_fig1\\_261674491](https://www.researchgate.net/figure/Example-of-a-discrete-gradient-vector-field-on-a-triangulated-terrain-Paired-simplices_fig1_261674491).

El concepto de Watershed se basa en visualizar una imagen en tres dimensiones (3D): dos coordenadas espaciales frente a niveles de gris. En esta interpretación topográfica, de ahí que estos algoritmos estén dentro del campo de la morfología matemática (nacieron dentro del mundo de la topografía), se considerarán tres tipos de puntos:

1. Puntos que corresponden a mínimos locales.

2. Puntos en los que, si se coloca una gota de agua, esta cae con certeza en un único mínimo: los puntos que verifican esta condición se denominan **puntos Watershed o catchment basis**.
3. Puntos en los que el agua caería con igual probabilidad en más de uno de estos mínimos: los puntos que verifican esta condición forman las **Líneas Watershed o líneas de cresta**.

**El objetivo principal de los algoritmos de segmentación es alcanzar estas líneas divisorias o de cresta.**

La idea básica es la siguiente: supongamos que se hace un pequeño agujero en cada mínimo local y que todo el relieve topográfico es inundado desde abajo. El agua va subiendo e inundando las cuencas. Cuando el agua de dos cuencas está a punto de juntarse, se construye un dique (*dam*) para evitar la fusión. La inundación continúa y llega a un punto en que solo se ve la parte de arriba de los diques por encima de la línea de agua. Las líneas de Watershed forman un camino conexo, dando por lo tanto bordes continuos entre las regiones. Este ejemplo puede verse en la imagen a continuación.

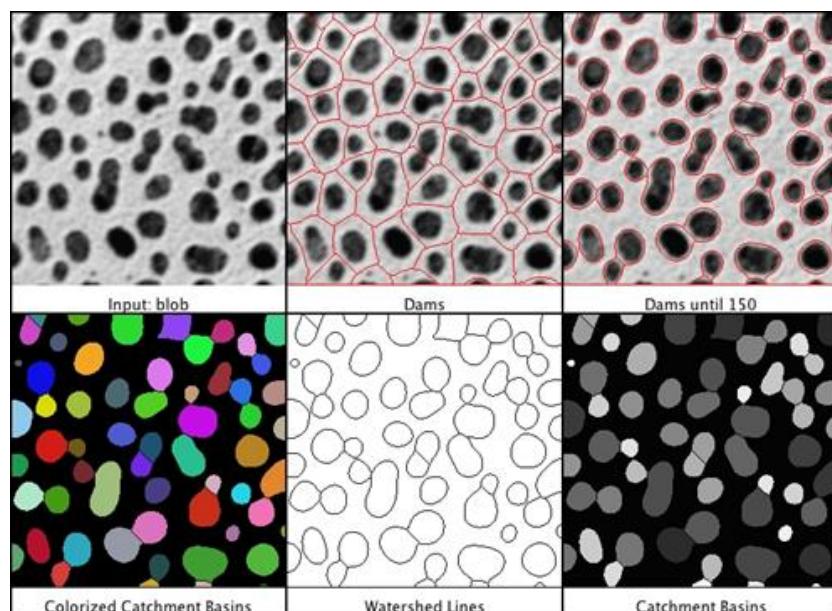


Figura 13. Pasos más comunes en la ejecución de la segmentación basada en Watershed.

Fuente: <http://bigwww.epfl.ch/sage/soft/watershed/>.

La segmentación basada en Watershed produce normalmente **sobresegmentación** y puede deberse a diferentes aspectos como la diferencia de texturas o patrones, ruido o incluso cambios en las tonalidades del color. Para ello, lo más habitual suele ser:

- ▶ Eliminar los contornos irrelevantes una vez realizado el Watershed.
- ▶ Modificar la imagen gradiente de tal forma que las regiones de depresión o valles se correspondan únicamente con los objetos deseados.
- ▶ Imponer marcadores como mínimos de la imagen gradiente.
- ▶ Suprimir todos los demás mínimos del gradiente (los irrelevantes) rellenando los correspondientes valles.
- ▶ Preservar las líneas de cresta más importantes de la imagen gradiente, localizadas entre los marcadores.

## 10.8. Crecimiento de regiones basado en grafos

**P**or último, presentamos un apartado dedicado a los algoritmos de crecimientos de regiones basadas en **grafos**. Estos algoritmos entienden la imagen como un grafo con cierta conectividad entre los píxeles y con unas **propiedades de conectividad**: funciones de similaridad, peso, distancia, etc.

En función de esas propiedades, el algoritmo itera el grafo eliminando las conexiones más débiles (menos parecidas) y potenciando las más fuertes.

Dentro de esta familia se encuentran los siguientes algoritmos principales:

**Graph-cuts** y **Normalized-cuts**: dado un grafo, eliminan aquellas conexiones, también conocidas como *edges*, entre los píxeles (llamados en la literatura como *nodes*), que son más débiles.

**Agregación multiescala:** partiendo de un grafo, crean diferentes niveles de segmentación donde el algoritmo va encontrando qué subsegmentos (subgrafos) pertenecen a un mismo segmento. Muy similar al concepto de *split and merge*, pero utilizando métodos como graph-cuts para realizar el *split*.

A continuación, se presenta un esquema resumido de ambos algoritmos.

### Graph-cuts

Para Graph-cuts se presenta el esquema típico de funcionamiento:

- ▶ En primer lugar, se obtiene la imagen original.
- ▶ En segundo lugar, se crea el grafo obteniendo para cada conexión entre píxeles (nodos dentro del grafo) una función de similaridad o parecido. La vecindad puede ser sencilla, como en este caso, o compleja si consideramos conectividades con nodos/píxeles más lejanos.
- ▶ Se buscan aquellas conectividades más débiles o que poseen una similaridad menor. El corte del grafo por esos puntos más débiles es lo que otorga el nombre de Graph-cuts al algoritmo.
- ▶ Por último, se convierte el grafo en imagen de nuevo, esta vez con una frontera que refleja la segmentación realizada.

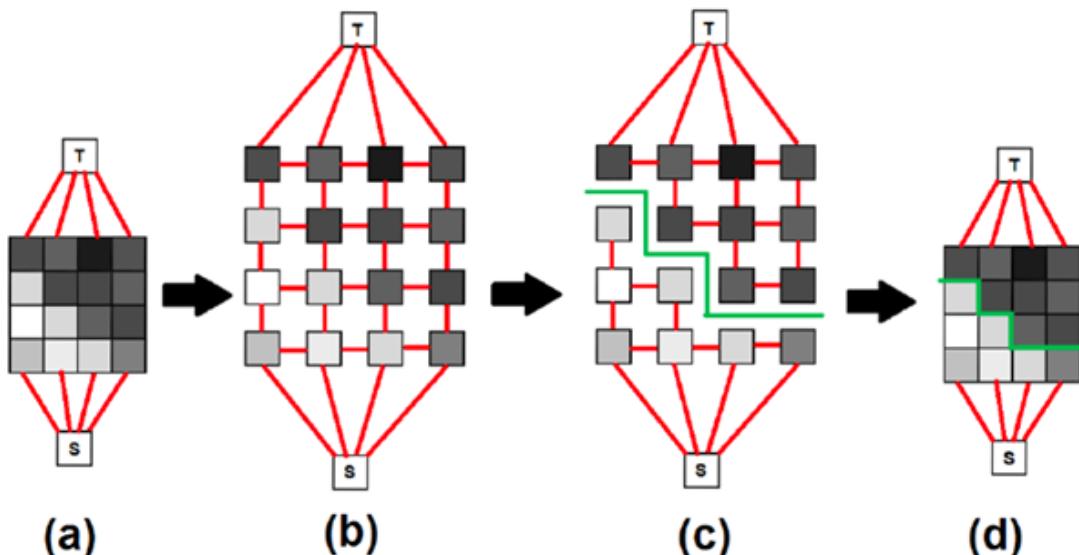


Figura 14. Funcionamiento simplificado de graph-cuts.

## Agregación multiescala

El funcionamiento es el siguiente:

- ▶ Se parte de la imagen original, donde cada píxel representará un nodo.
- ▶ Se realiza, en función de un criterio de similaridad, una agrupación basada en los nodos más próximos. Esta agregación de por sí ya es una segmentación, aunque proporciona resultados muy pocos precisos.
- ▶ Una vez agrupados los nodos en segmentos, se obtiene un **representante** (normalmente el valor medio) de dicho segmento, lo que conduce a una nueva escala (nueva capa) con menos nodos, pero cada nodo representando a un subconjunto de nodos iniciales.
- ▶ Se vuelve a repetir la agregación, esta vez para los nodos representantes y se itera este proceso hasta que, o bien se alcanza un número determinado de segmentos, o bien se verifica una condición de calidad, es decir, que los segmentos creados, independientemente del número que sean, tienen unas propiedades bien definidas.
- ▶ Finalmente, la imagen se segmenta deshaciendo las agrupaciones realizadas por cada uno de los niveles.

De esta manera, si existen regiones parecidas entre ellas, pero separadas de forma espacial en la imagen, el algoritmo encontrará la manera de asociarlas a un mismo segmento.

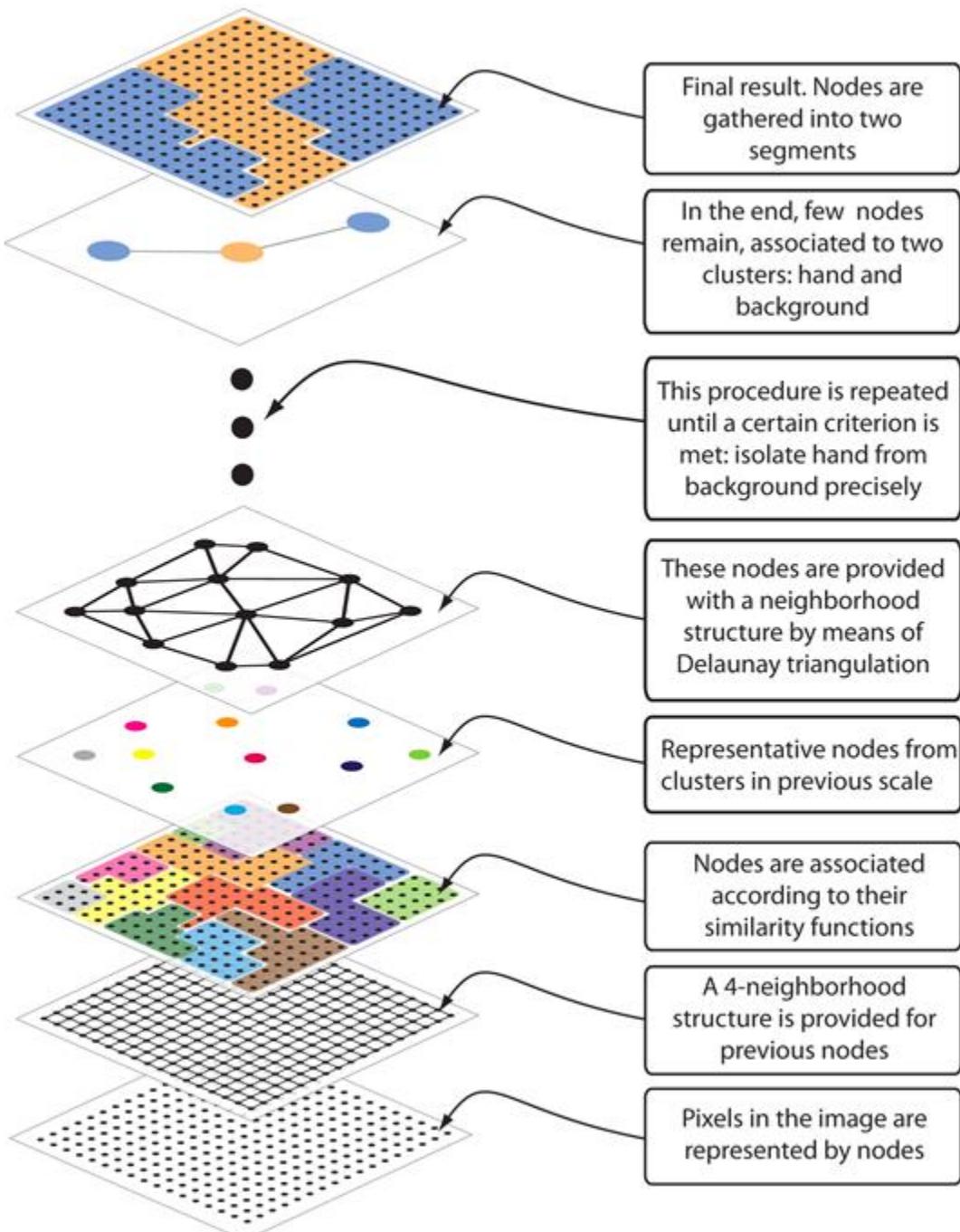


Figura 15 Procedimiento de agregación multiescala comenzando por la imagen original (parte inferior) y terminando con la imagen segmentada (parte superior).

## 10.9. Referencias bibliográficas

Cartas-Ayala, A. (Sin fecha). Gradient Vector Flow Snakes. Manuscrito presentado para su publicación. Recuperado de  
[http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/AV1011/cartas.pdf](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/cartas.pdf)

González, R. C. y Woods, R. E. (2008). *Digital image processing*. New Jersey: Pearson Education.

Paragios, N. y Ramesh, V. (2004). Gradient Vector Flow Fast Geometric Active Contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3), 402-407. Recuperado de

[https://www.researchgate.net/publication/8337757\\_Gradient\\_Vector\\_Flow\\_Fast\\_Geometric\\_Active\\_Contours](https://www.researchgate.net/publication/8337757_Gradient_Vector_Flow_Fast_Geometric_Active_Contours)

Percepción Computacional

---

# Extracción de características. Propiedades estadísticas y frecuenciales de la señal

## 11.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

**E**ste tema tiene como objetivo proporcionar herramientas con las que caracterizar el comportamiento de nuestras señales. Se van a explicar las principales variables que describen las propiedades de la señal y, como resultado, tendremos un conjunto de atributos de diversa naturaleza que cuantifican diferentes cualidades de la fuente de información que manejamos.

Estos atributos contribuyen a sintetizar la información contenida en la señal, de forma que pueda ser manejada más eficientemente en una etapa final basada en técnicas de reconocimiento de patrones.

En temas anteriores, se ha descrito cómo las señales, generalmente series temporales de naturaleza unidimensional, e imágenes, definidas en dos dimensiones, que manejamos pueden ser modeladas matemáticamente como procesos estocásticos. Asumiremos en este tema que nuestras señales o procesos estocásticos son estacionarios en sentido amplio. Por tanto, las propiedades estadísticas de la señal no varían en función de las coordenadas del proceso.

De acuerdo a este modelo, las muestras observadas en una señal  $f(\cdot)$  pueden ser consideradas como **realizaciones de una variable aleatoria**. Por ejemplo, si nuestra señal se trata de un electrocardiograma, cuya amplitud refleja el potencial del campo eléctrico generado por la actividad muscular del corazón, las muestras de la señal pueden tomarse como observaciones de una variable aleatoria que toma el valor de

dicho potencial. Por tanto, puede caracterizarse el comportamiento estadístico de esta variable.

**Las funciones de distribución y de densidad de probabilidad definen por completo el comportamiento estadístico de una variable.**

Sin embargo, en algunas aplicaciones prácticas es deseable disponer de elementos cuantitativos más fácilmente manejables que una función para describir sus propiedades. Por ejemplo: el valor medio de la variable sería uno de estos atributos. Estos elementos permiten obtener una caracterización estadística parcial de la variable frente a la descripción completa que nos ofrecen las funciones mencionadas.

Además de la caracterización puramente estadística de la variable aleatoria que representa el valor de la señal, pueden calcularse otros parámetros que contribuyen a explicar el comportamiento de la señal en el dominio en el que queda definida (por ejemplo: en el tiempo, en el caso de series temporales o en las dos coordenadas espaciales para imágenes). En este sentido, es habitual emplear **medidas cuantitativas derivadas de la teoría de la información o la teoría del caos**.

Por último, cabe destacar que hemos mencionado la extracción de atributos de la señal de partida mediante su análisis en el dominio natural. Sin embargo, el análisis en el dominio frecuencial proporciona atributos de gran valor para la definición de su comportamiento. Estos atributos se derivan de la densidad espectral de potencia de la señal de partida, que se obtiene como la transformada de Fourier de su función de autocorrelación.

La siguiente figura muestra un esquema de los principales conceptos expuestos en este tema.

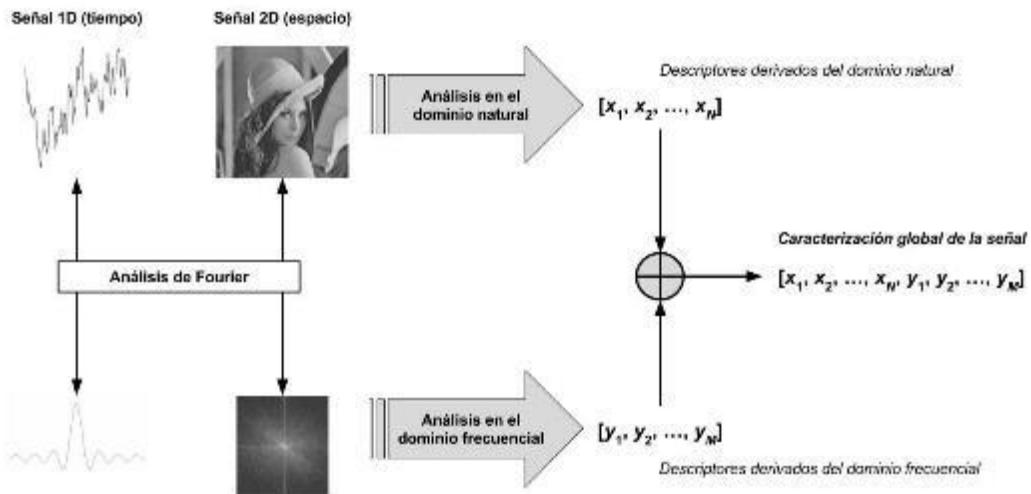


Figura 1. Proceso de caracterización de señales en su dominio natural y frecuencial para la generación de un vector global de características.

## 11.2. Caracterización de señales en el dominio natural

**E**n primer lugar, se indicarán los **parámetros más comunes** a partir del análisis de la señal en el dominio en el que se define. Entre estos, se tendrán parámetros estadísticos y otros derivados de la teoría de la información.

### Caracterización estadística parcial

#### Máximo y mínimo

La diferencia entre ambos valores define el **rango dinámico** de la señal.

#### Media

Asumamos que nuestra señal  $f(\cdot)$  está compuesta por  $M$  muestras. En el caso de una serie unidimensional, esta tendría longitud  $M$ . Para una imagen, estaría formada por

un total de  $M$  píxeles. Utilizamos  $f_j$  para denotar los diferentes valores que toma nuestra señal, asumiendo que puede ser uno de  $N$  diferentes, es decir,  $j = 1, \dots, N$ .

Finalmente, supongamos que  $k_j$  muestras de las  $M$  observadas tienen un valor  $f_j$ . Puede deducirse de forma directa que el número total de muestras es la suma de los cardinales asociados a cada uno de los posibles valores  $f_j$ :

$$\sum_j k_j = M$$

A partir de este escenario, podemos calcular el **valor medio de la muestra** de la siguiente forma:

$$\bar{f} = \sum_{j=1}^N \frac{k_j}{M} f_j$$

Es decir, el valor medio se obtiene como la media ponderada de todos los valores  $f_j$  posibles. El peso en la ponderación viene dado por la relación  $\frac{k_j}{M}$ , que aproxima la probabilidad de observar un valor  $f_j$  en la señal. Esta aproximación se obtiene, por tanto, como la fracción entre el número de observaciones del valor  $f_j$  y el número total de muestras en nuestra señal.

Cabe destacar que en un ejercicio aparentemente tan simple como el cálculo de la media se llevan cabo diferentes pasos:

- ▶ En primer lugar, se discretiza la variable continua que refleja la amplitud de la señal. Teóricamente, nuestra señal  $f(\cdot)$  toma valores en un rango continuo. Sin embargo, se ha asumido para la estimación de la media que  $f(\cdot)$  es discreta y toma únicamente uno de los  $N$  valores posibles  $f_j$  ( $j = 1, \dots, N$ ).

- ▶ Posteriormente, se ha aproximado la función de densidad de probabilidad de esta variable discreta. Para ello, se ha aproximado la probabilidad asociada a cada posible valor de la variable como la frecuencia de observación de dicho valor.

Este ejercicio de discretización y estimación de la función de densidad de probabilidad de la variable discreta se repetirá para el cálculo de otros parámetros estadísticos que permiten una caracterización parcial.

El valor medio obtenido para la señal  $f(\cdot)$  es una estimación de su media real  $\eta_F$ , que viene definida de la siguiente forma:

$$\eta_F = E\{F\} = \int_{-\infty}^{\infty} f p_F(f) df$$

Donde:

- ▶  $F$  es la variable aleatoria que representa el valor de la señal.
- ▶  $p_F(f)$  es la función de densidad de probabilidad de la variable.
- ▶  $F$  y  $E\{\cdot\}$  denota el operador esperanza matemática.

### Mediana

El valor mediano o mediana de una variable  $F$  es el valor  $f_{med}$  de esta para el que la probabilidad de observar una muestra de  $F$  en el intervalo  $(-\infty, f_{med}]$  es 0.5. Es decir, el área bajo la función de densidad de probabilidad a ambos lados de este valor es la misma y es 0.5. Por tanto, puede expresarse de la siguiente forma:

$$P(F \leq f_{med}) = P(F > f_{med}) = \int_{-\infty}^{f_{med}} p_F(f) df = \int_{f_{med}}^{\infty} p_F(f) df = 0,5$$

La definición se aplica de la misma forma a variables discretas, ya que su función de densidad de probabilidad puede escribirse en términos de funciones  $\delta(x)$ . Sin

embargo, la estimación de la mediana suele obtenerse como el valor de la observación ubicada en la posición central tras la ordenación de las muestras disponibles para la variable. Si el número de muestras disponibles es par, se toma como estimación de la mediana el valor de la **media aritmética** entre las dos observaciones centrales.

### Moda

La moda de la variable  $F$  ( $f_{mod}$ ) viene dada por el valor de la misma que se toma con mayor probabilidad. Es decir, vendría dada por el valor en el cual la función de densidad de probabilidad  $p_F()$  alcanza su máximo:

$$f_{mod} = \operatorname{argmax}\{p_F(f)\}$$

### Varianza

La varianza de una variable aleatoria es el promedio de las desviaciones, en términos cuadráticos, de los valores que toma la variable respecto a su media. A partir de un conjunto de observaciones, la varianza puede estimarse de la siguiente forma:

$$\bar{\sigma}^2 = \sum_{j=1}^N \frac{k_j}{M} (f_j - \bar{f})^2$$

Como puede apreciarse, para la **estimación de la varianza** se lleva a cabo la misma aproximación empleada para el cálculo de la media basada en la discretización de la variable y la estimación de su función de densidad de probabilidad.

De forma precisa y de acuerdo a su definición, la varianza viene dada por la esperanza matemática de las desviaciones cuadráticas de  $F$  respecto a su media:

$$\sigma_F^2 = E\{(F - \eta_F)^2\}$$

## Asimetría

La asimetría (en inglés, *skewness*) de una variable aleatoria cuantifica el grado de asimetría de su función de densidad de probabilidad respecto al eje vertical que pasa por su media.

Este **estadístico**, de forma coloquial, refleja hacia qué lado de la media es más pronunciada la cola de la distribución.

- ▶ Un valor positivo de la asimetría refleja que la cola de la función de densidad de probabilidad a la derecha de la media es más larga que a la izquierda.
- ▶ En caso contrario, el valor de la asimetría será menor que 0.
- ▶ En el caso de una variable caracterizada por una **función simétrica**, como es el caso de las variables gaussianas, su asimetría es 0.

La figura 2 muestra gráficamente la interpretación de la asimetría de una variable aleatoria.

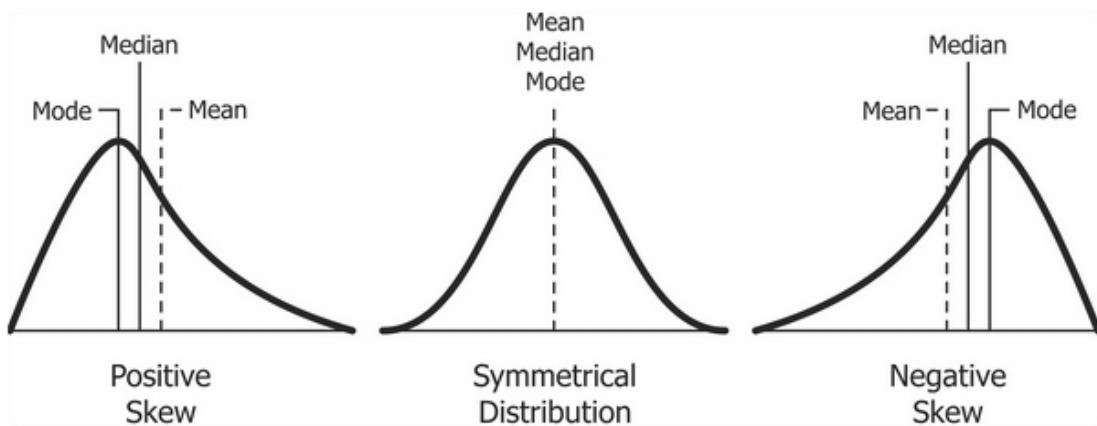


Figura 2. Interpretación de la asimetría estadística de una variable.

Fuente: <https://medium.com/@cybersiftIO/anomaly-detection-vs-ransomware-b83510a3a860>

La asimetría  $\gamma$  viene dada por la siguiente expresión:

$$\gamma = E \left[ \left( \frac{f - \eta_F}{\sigma} \right)^3 \right]$$

Donde:

- ▶  $\sigma$  es la desviación típica de la variable  $F$ , que viene dada por la raíz cuadrada positiva de la varianza.

## Curtosis

La curtosis de una variable caracteriza las **colas** de su función de densidad de probabilidad. Como referencia, se suele emplear la curtosis de la distribución normal, que es 3. Un valor inferior (curtosis negativa) refleja que los *outliers* que genera la distribución son menos extremos que los derivados de una distribución normal. La interpretación contraria es aplicable a un valor positivo de curtosis.

La siguiente imagen refleja la propiedad evaluada por este parámetro.

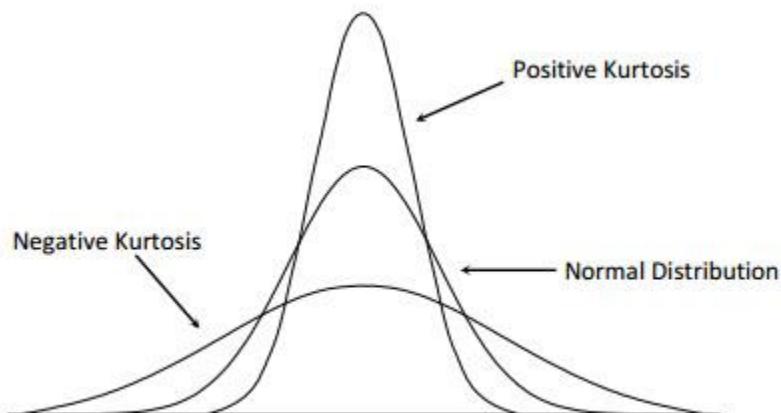


Figura 3. Interpretación de la curtosis de una variable.

Fuente: <https://medium.com/@cybersiftIO/anomaly-detection-vs-ransomware-b83510a3a860>

La expresión para el cálculo de la curtosis ( $\delta$ ) es la siguiente:

$$\delta = E \left[ \left( \frac{f - \eta_F}{\sigma} \right)^4 \right]$$

## Percentiles

Diferentes valores de percentil contribuyen a dar una idea de la distribución estadística de la variable a caracterizar. El percentil  $f_u$  de la variable  $F$  representa el umbral para el cual la probabilidad acumulada es  $u$ . Matemáticamente, viene dado por la siguiente expresión:

$$P(F \leq f_u) = u$$

Generalmente, se calcula el valor del percentil para los umbrales 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 y 0.9. A fin de reducir el número de parámetros, en los umbrales 0.25, 0.5 y 0.75.

## Rango intercuartil

Los percentiles para los umbrales de probabilidad 0.25 y 0.75 se denominan primer cuartil y tercer cuartil, respectivamente. El rango intercuartil (IQ) es la diferencia entre ambos percentiles:

$$IQ = f_{0.75} - f_{0.25}$$

## Otros parámetros

Se indican a continuación otros parámetros que sintetizan información de interés sobre el comportamiento de la señal en su dominio original.

## Entropía

Se incluye en este apartado, a pesar de que se corresponde con un parámetro estadístico más derivado de la función de densidad de probabilidad de la variable aleatoria a caracterizar. La entropía ha sido descrita en detalle en temas anteriores

de la asignatura. Como se indicó, cuantifica el promedio de la cantidad de información generada por una variable aleatoria.

De forma intuitiva, una variable tendrá mayor entropía si su función de densidad de probabilidad tiende a ser más uniforme en el rango de definición de esta. Es decir, mayor entropía está asociada a mayor incertidumbre sobre el valor que *a priori* puede tomar la variable.

Por tanto, la **distribución uniforme** será la que tenga una mayor entropía asociada, pues ninguno de los valores que la variable puede tomar posee, *a priori*, mayor probabilidad de ser observado que otros.

### Complejidad de Lempel-Ziv

La complejidad de Lempel-Ziv es una medida no paramétrica de complejidad de una fuente de información. Generalmente, su definición aplica a señales unidimensionales en forma de serie, sin embargo, es extensible a imágenes. Su cálculo se basa en la obtención del número de subsecuencias diferentes que hay contenidas en la señal original. Para cada una de ellas, se calcula la **frecuencia de repetición**.

Una mayor tasa de repetición reflejará una menor complejidad de la fuente de datos, ya que el contenido de la señal indica que su origen es menos caótico y, por tanto, su comportamiento es más predecible y hay **menor incertidumbre** asociada.

### Medida de la tendencia central

Las ecuaciones derivadas de la teoría del caos pueden emplearse para generar distintos tipos de gráficos. Entre estos, tenemos el que se conoce como el **gráfico de diferencias de primer orden**: considerando una serie temporal  $f[n]$ , la representación de  $f[n + 2] - f[n + 1]$  frente a  $f[n + 1] - f[n]$ .

Centrado en el origen, ofrece una representación gráfica del **grado de variabilidad** en la serie temporal  $f[n]$ . Con este enfoque, en lugar de definir categóricamente una serie temporal como caótica o no caótica, se cuantifica el grado de variabilidad o caos.

A partir de la representación de las diferencias de primer orden se obtiene la métrica de variabilidad conocida como medida de la tendencia central.

Un mayor valor de esta métrica refleja mayor variabilidad y, por tanto, un mayor grado de caos y de dificultad para predecir el comportamiento de la fuente de datos.

### 11.3. Características derivadas del análisis en frecuencia

**L**a representación de una señal en sus componentes frecuenciales permite apreciar atributos de esta que no son observables en su dominio natural. Dicha representación viene descrita por la teoría de Fourier, que define las operaciones de transformación directa e inversa a partir de una base ortogonal de funciones de la frecuencia.

La **densidad espectral de potencia**  $S(\omega)$ , abreviada como DEP, de una señal refleja la potencia contenida por unidad de frecuencia. Así, aquellas regiones de  $\omega$  en las que esta función tome una amplitud mayor reflejarán las componentes frecuenciales más significativas de la señal. Por ejemplo: en el caso de una señal de voz, su densidad espectral de potencia se concentra en el rango entre 300 Hz y 4 KHz, aproximadamente.

A continuación, se indican diferentes **descriptores** de la función de densidad espectral de potencia que caracterizan su perfil y comportamiento.

## Ancho de banda

El ancho de banda de una señal se define como el rango de frecuencias en el que su **densidad espectral de potencia es mayor que cero**. Se calcula como la diferencia entre las componentes frecuenciales máxima y mínima de la señal.

Generalmente no se suele emplear la definición anterior, que refleja el ancho de banda absoluto de la señal. Así, es habitual considerar el ancho de banda de acuerdo a criterios como la diferencia entre componentes frecuenciales para las que  $S(\omega)$  refleja una atenuación de 3 dB respecto a su máximo o el rango de frecuencias donde se concentra el 90 % de la potencia de la señal.

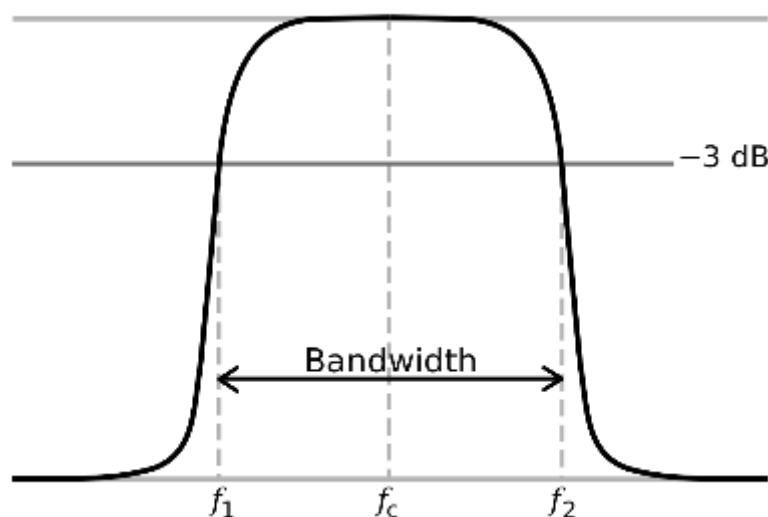


Figura 4. Definición del ancho de banda a 3 dB de una señal.

Fuente: Adaptado de [https://en.wikipedia.org/wiki/Bandwidth\\_\(signal\\_processing\)](https://en.wikipedia.org/wiki/Bandwidth_(signal_processing))

Cualitativamente, puede interpretarse que una señal caracterizada por un mayor ancho de banda presenta diferencias más significativas en la velocidad de variación de su perfil (variaciones lentas y rápidas). Por tanto, la señal contiene cambios más diversos y está asociada a una mayor cantidad de información.

## Formantes

Las frecuencias formantes se identifican como **máximos locales** en la función de densidad espectral de potencia (como se ve en la figura 5). Estas componentes se asocian con las principales frecuencias contenidas en la señal bajo estudio.

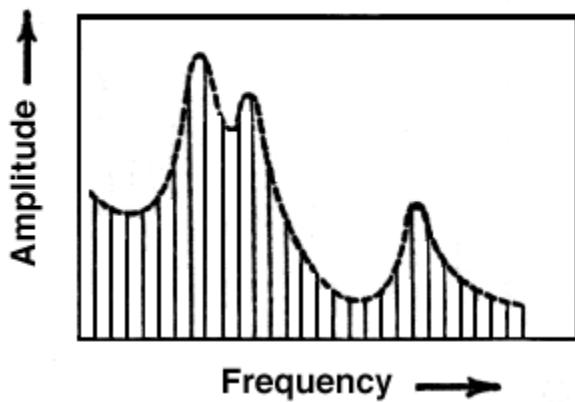


Figura 5. Ejemplo de una función de densidad espectral de potencia en la que se aprecian tres frecuencias fromantes.

Fuente: <https://www.sfu.ca/sonic-studio/handbook/Formant.html>

## Caracterización estadística

La **normalización** de la función de densidad espectral de potencia, de tal forma que el área contenida bajo la misma sea igual a la unidad, hace que pueda ser interpretada como una función de densidad de probabilidad. Esta caracterizaría el comportamiento estadístico de la variable aleatoria que viene dada por la frecuencia. Así, el área bajo dicha función de probabilidad hasta la frecuencia  $\omega_0$  representaría la probabilidad de observar componentes frecuenciales menores o iguales que  $\omega_0$  en nuestra señal.

Así, a partir de esta **función de densidad de probabilidad**, pueden obtenerse parámetros de naturaleza estadística como la media, la mediana o la varianza. Concretamente, la frecuencia mediana se ha empleado comúnmente para la caracterización del espectro de una señal.

## Entropía espectral

Por último, a partir de la función de densidad espectral normalizada se obtiene la entropía espectral. Este parámetro no es más que el cálculo de entropía para la variable aleatoria que representa la componente frecuencial.

Percepción Computacional

---

# Extracción de características. Caracterización de textura en imágenes

# Ideas clave

## 12.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

**L**a textura ha sido objeto de estudio durante las últimas décadas para la comunidad científica inmersa en el procesamiento de imágenes. Sin embargo, no hay consenso sobre su definición. A partir de las definiciones proporcionadas por diferentes investigadores podemos decir que:

La textura es una propiedad relacionada con la aspereza de una superficie y se caracteriza por la **variación de la intensidad de píxeles** en el dominio espacial.

La textura puede encontrarse en imágenes de diferente naturaleza, por ejemplo: un tejido biológico específico (tejido muscular o nervioso), un área de terreno, la superficie de un objeto o la piel de una persona o animal. Por tanto, la caracterización de la textura en una imagen es necesaria en diferentes escenarios reales. Específicamente, la textura juega un papel relevante en tareas como el diagnóstico médico, la teledetección o la identificación biométrica.

En este tema, se aborda la caracterización de la textura para su **identificación automática**. En algunos problemas, el desafío consiste en identificar automáticamente a qué tipo de textura pertenece la imagen de entrada. Por ejemplo, considérese un escenario de diagnóstico médico en el que ha de indicarse si la muestra de tejido corresponde a una zona dañada o, por el contrario, a una zona sana.

Así, tomemos la figura 1, donde se recogen diferentes ejemplos de textura. En la parte superior se aprecian tres texturas diferentes y en la parte inferior reúne también diferentes tipos de textura. Esta imagen refleja aquellos problemas de **segmentación** en los que se trata de delimitar diferentes zonas de la imagen de forma automática. En este caso, puede pensarse en una instantánea de un paisaje en la que se pretende distinguir áreas como el cielo, el terreno y la pared de una casa.

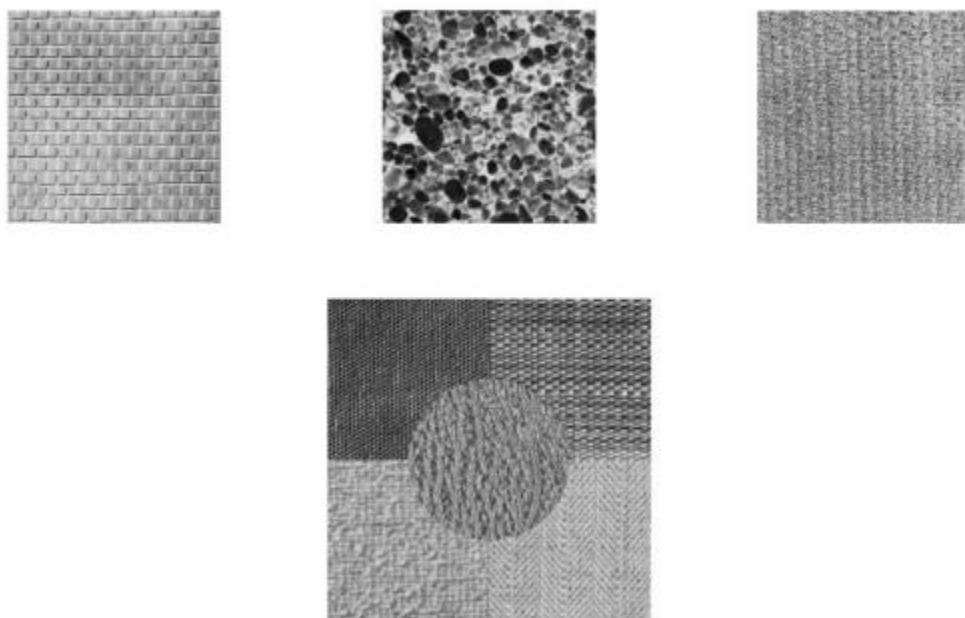


Figura 1. Ejemplo de tres texturas diferentes correspondientes a un muro de ladrillo, un suelo empedrado y un tejido textil.

Fuente: Adaptado de <http://www.ux.uis.no/~tranden/brodatz.html>

A fin de identificar una textura, el primer paso es la extracción de descriptores que permitan diferenciarla de otras. Existen diferentes métodos que pueden ser empleados para la caracterización. En este tema se describirán tres de ellos:

- ▶ La matriz de coocurrencia de Haralick.
- ▶ Los patrones binarios locales.
- ▶ La caracterización basada en el análisis de la imagen en un dominio transformado.

El siguiente esquema recoge, de forma breve, los conceptos desarrollados a lo largo de este tema.

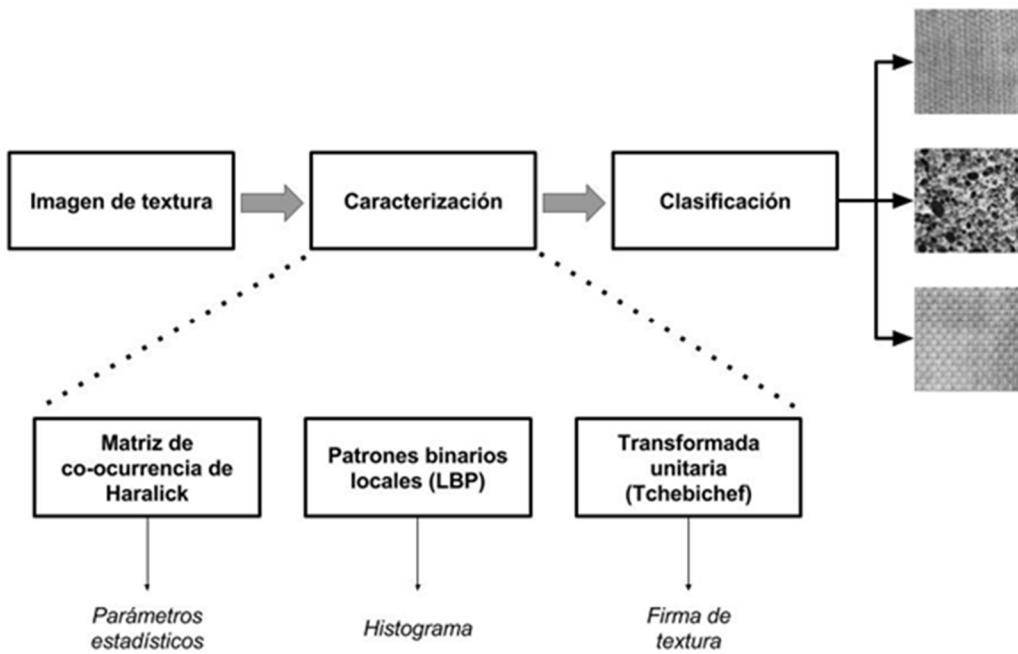


Figura 2. Principales herramientas para la caracterización de la textura en una imagen.

## 12.2. Matriz de coocurrencia de Haralick

**L**a textura está caracterizada por la relación espacial entre las intensidades correspondientes a sus píxeles. La matriz de coocurrencia refleja estas relaciones, obteniéndose para una dirección determinada entre píxeles vecinos:  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  y  $135^\circ$ . A partir de la matriz coocurrencia es posible obtener un gran número de descriptores estadísticos que facilitan una descripción precisa de la textura.

Esta matriz es cuadrada y de dimensión  $N_g$ , donde  $N_g$  es el número de niveles utilizados para la cuantificación de los valores de intensidad (gris) en la imagen. El elemento  $[i,j]$  de la matriz se genera contando el número de veces que un píxel con valor de intensidad  $i$  es adyacente a un píxel con valor  $j$ . Finalmente, se divide la matriz completa por el número total de las comparaciones realizadas. En otras palabras, cada elemento de la matriz puede interpretarse como la probabilidad de que un píxel con valor  $i$  se encuentre adyacente a un píxel de valor  $j$ .

La identificación de píxeles adyacentes depende de la dirección considerada. Así, dado que los píxeles de una imagen son cuadrados, pueden considerarse los cuatro ángulos indicados anteriormente para el cálculo de la matriz.

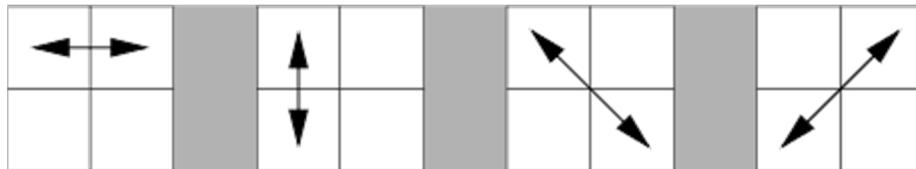


Figura 3. Cuatro direcciones posibles para la identificación de píxeles adyacentes en una imagen y el cómputo de la matriz de coocurrencia de Haralick.

Fuente: [http://murphylab.web.cmu.edu/publications/boland/boland\\_node26.html](http://murphylab.web.cmu.edu/publications/boland/boland_node26.html)

Comúnmente, se emplea  $N_g = 8$  y se calculan las cuatro matrices posibles (una por cada ángulo diferente). La figura 4 muestra la estructura de la matriz resultante.

$$\mathbf{G} = \begin{bmatrix} p(1,1) & p(1,2) & \cdots & p(1, N_g) \\ p(2,1) & p(2,2) & \cdots & p(2, N_g) \\ \vdots & \vdots & \ddots & \vdots \\ p(N_g,1) & p(N_g,2) & \cdots & p(N_g, N_g) \end{bmatrix}$$

Figura 4. Matriz de coocurrencia de Haralick con  $N_g$  niveles y para una dirección determinada.

Fuente: [http://murphylab.web.cmu.edu/publications/boland/boland\\_node26.html](http://murphylab.web.cmu.edu/publications/boland/boland_node26.html)

A partir de estas matrices, se calculan características como las que se describen a continuación:

### Momento angular de segundo orden

$$\sum_i \sum_j p^2(i,j)$$

### Energía

$$\sqrt{\sum_i \sum_j p^2(i,j)}$$

### Contraste

$$\sum_i \sum_j (i-j)^2 p(i,j)$$

### Entropía

$$-\sum_i \sum_j p(i,j) \log(p(i,j))$$

### Media

$$\mu_x = \sum_i \sum_j i p(i,j), \mu_y = \sum_i \sum_j j p(i,j)$$

### Varianza

$$\sigma_x = \sqrt{\sum_i \sum_j (i - \mu_x)^2 p(i,j)}, \sigma_y = \sqrt{\sum_i \sum_j (j - \mu_y)^2 p(i,j)}$$

### Correlación

$$\sum_i \sum_j p(i,j) (i - \mu_x)(j - \mu_y) / (\sigma_x \sigma_y)$$

## 12.3. Patrones binarios locales

**E**l operador de patrones binarios locales o LBP, del inglés *local binary patterns*, permite obtener una transformación píxel a píxel de la imagen original. El valor transformado de un píxel se obtiene a partir de la

comparación de su valor de intensidad con los píxeles vecinos. Dada la capacidad de este operador para caracterizar la textura y su simplicidad computacional, los LBP se han convertido en una técnica muy común en diversas aplicaciones dentro del procesado de imagen.

Quizás la propiedad más importante del operador de LBP en aplicaciones del mundo real es su **robustez frente a los cambios monótonos** de la escala de grises causados, por ejemplo, por las variaciones de iluminación (ruido). Como se mencionaba previamente, otra propiedad interesante de los LBP es su facilidad de cálculo, lo que hace de este operador una herramienta muy útil para el análisis de imágenes en entornos donde el tiempo real es crítico.

El operador LBP se basa en la idea de que las texturas superficiales bidimensionales pueden describirse mediante dos **medidas complementarias**:

- ▶ Patrones espaciales locales.
- ▶ Contraste de escala de grises.

El operador original de LBP, propuesto por Ojala (1996), se obtiene de la siguiente forma para cada píxel de la imagen:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p, s = 1 \text{ si } x > 0 \text{ y } s = 0 \text{ si } x < 0$$

Donde:

- ▶  $g_c$  hace referencia a la intensidad del píxel para el que se está calculando la transformación.
- ▶ Los píxeles  $g_p$  ( $p = 0, \dots, P - 1$ ) son sus  $P$  píxeles vecinos para el radio  $R$  considerado.

Para aquellos puntos de la vecindad que no coincidan exactamente con la posición de un píxel de la imagen, se obtiene una estimación de la intensidad en ese punto mediante la interpolación a partir de los píxeles más próximos. La notación  $(P, R)$  se usa para identificar las características de la vecindad considerada, lo que significa  $P$  puntos de muestreo en un círculo de radio de  $R$ .

Por tanto, el procedimiento para el **cálculo del LBP** conlleva:

1. Tomar la muestra de píxeles vecinos.
2. Calcular la diferencia del píxel central con estos.
3. Aplicar el umbral 0 sobre esta diferencia.
4. Posteriormente, se transforma la cadena binaria obtenida en el número decimal correspondiente.

En la siguiente imagen se muestra un ejemplo del cálculo del operador LBP.

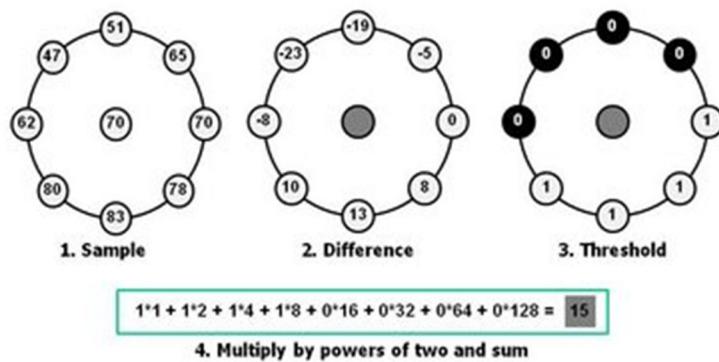


Figura 5. Etapas involucradas en el cálculo del operador LBP.  
Fuente: [http://www.scholarpedia.org/article/Local\\_Binary\\_Patterns](http://www.scholarpedia.org/article/Local_Binary_Patterns)

Como resultado, cada píxel de la imagen es asignado con una de las  $2^P$  posibles etiquetas. Comúnmente, el radio  $R$  es tal que  $P = 8$ . El histograma de estas  $2^8 = 256$  etiquetas diferentes se puede usar como un **descriptor de textura**.

Dado el elevado número de contenedores del histograma resultante, el resultado de la transformación tenderá a generar histogramas con un número significativo de 0,

pues muchas de las  $2^P$  etiquetas posibles no habrán sido asignadas a ningún píxel. Además, el resultado de la transformación variará si la imagen original es rotada un determinado ángulo. A fin de superar estas limitaciones, se propuso una **extensión del operador LBP** original basada en la definición de los llamados patrones uniformes. Estos permiten reducir la longitud del vector de características e implementar un descriptor invariante de rotación simple.

Esta extensión se inspiró en el hecho de que algunos patrones binarios aparecen con mayor frecuencia en imágenes de textura que otros.

Un patrón binario local se denomina **uniforme** si su cadena de bits contiene como máximo dos transiciones de 0 a 1 o viceversa. Por ejemplo:

- ▶ Los patrones 00000000 (0 transiciones), 00001110 (2 transiciones) y 11110011 (2 transiciones) son uniformes.
- ▶ Mientras que los patrones 00110110 (4 transiciones) y 10101101 (6 transiciones) no lo son.

En la extensión propuesta del operador se utiliza una única **etiqueta** para todos aquellos píxeles que den lugar a un patrón no uniforme. En el caso de los patrones uniformes, se emplea una etiqueta diferente para cada patrón resultante. Hay  $P + 1$  patrones uniformes diferentes. Por tanto, el número total de etiquetas para el cómputo del histograma empleado en la caracterización de la textura será de  $P + 2$ . Por ejemplo, cuando se usa el vecindario (8, 1), hay un total de 256 patrones, 58 de los cuales son uniformes, lo que da como resultado 59 etiquetas diferentes.

Ojala detectó en sus experimentos con imágenes de textura que los patrones uniformes representan casi el 90 % de todos los patrones cuando se usa el vecindario (8,1), y alrededor del 70 % en el vecindario (16, 2). Cada etiqueta o código LBP se puede considerar como una microestructura de textura (*microtexton*) en una imagen en la que esta estructura se repite varias veces. Las primitivas locales codificadas por

estas etiquetas se corresponden con elementos específicos de la imagen como diferentes tipos de bordes curvos, manchas, zonas planas, etc.

El operador de LBP invariante de rotación original, designado como LBPrui2, se obtiene rotando circularmente cada patrón de bits al valor mínimo. Por ejemplo, las secuencias de bits 1000011, 1110000 y 0011100 surgen de diferentes rotaciones del mismo patrón local y todas corresponden a la secuencia 0000111 una vez normalizadas.

La siguiente imagen (figura 6) muestra los diferentes patrones uniformes que podemos encontrar cuando el operador LBP se calcula con R=1 y P=8.

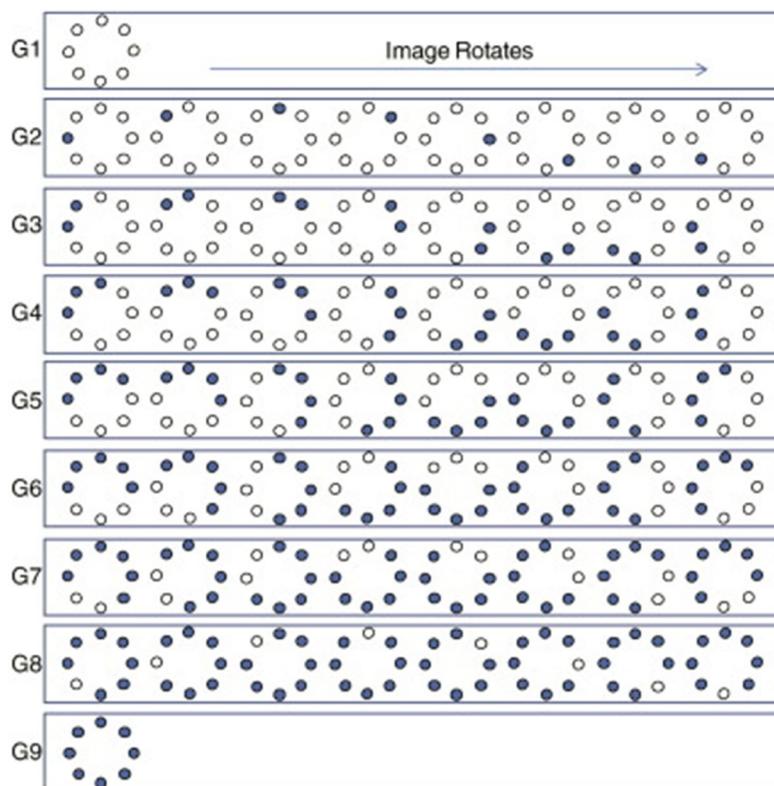


Figura 6. Patrones uniformes observables para R=1 y P=8.

Fuente: Qi et al., 2015.

## 12.4. Caracterización basada en transformadas unitarias

Las transformadas unitarias permiten la representación de la imagen, considerada una señal  $f(x, y)$  definida en el espacio  $R^2$ , en un espacio transformado. Para ello, se computa la proyección de la señal sobre funciones unitarias (ortonormales), que suponen una base para el espacio de funciones del dominio transformado. Comúnmente, estas funciones base son de tipo sinusoidal, característica que las hace una herramienta útil para capturar las propiedades de textura en una imagen.

El motivo de este hecho es que la textura, como se ha dejado intuir previamente, se caracteriza por la **repetición de ciertas estructuras en la imagen**. Por tanto, una función compuesta por un tren de pulsos o sinusoides, al calcularse la proyección de la imagen sobre esta, lleva a cabo el muestreo de la superficie definida por la función  $f(x, y)$ , que es la imagen. Este muestreo captura las repeticiones inherentes a la presencia de una textura.

Existen diferentes tipos de transformadas, es decir, de funciones unitarias que son base de un espacio de funciones (espacio transformado). La transformada de Fourier o la transformada del coseno son algunos ejemplos. En este apartado dedicado a la caracterización de la textura, se expone el uso de la transformada de Chebyshev para este propósito.

Los coeficientes de la transformada de Chebyshev son conocidos como **momentos de Chebyshev**. Fueron definidos por Mukundan (2001) con el objetivo de suprimir las limitaciones encontradas en los momentos de Zernike y Legendre.

El cálculo de la **transformada o momentos de Chebyshev** ( $T_{pq}$ ) se lleva a cabo mediante la proyección de la imagen o señal original  $f(x, y)$  sobre el polinomio de Chebyshev  $T_{pq}(x, y)$ :

$$T_{pq} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} t_{pq}(x, y) f(x, y)$$

El **polinomio de Chebyshev**  $t_{pq}(x, y)$  puede expresarse como el producto de los polinomios unidimensionales  $t_p(x)$  y  $t_q(y)$ . Por tanto, la expresión de  $T_{pq}$  descrita previamente es equivalente a la siguiente:

$$T_{pq} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} t_p(x) t_q(y) f(x, y)$$

Donde:  $t_n(x)$  es el polinomio de Chebyshev de grado  $n$  en una dimensión.

La siguiente figura muestra la forma de estas funciones.

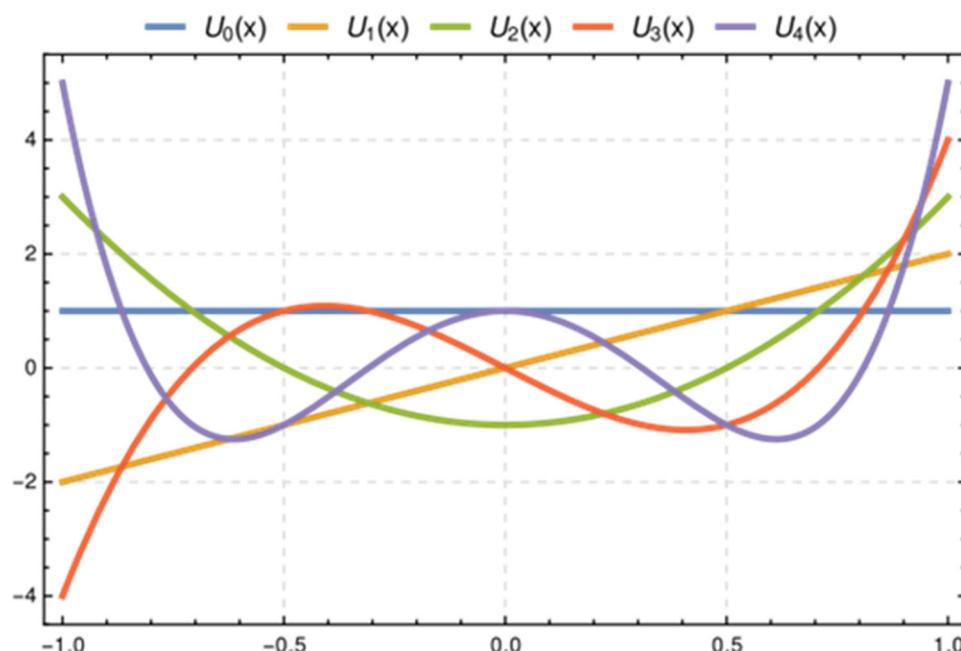


Figura 7. Perfil de los cinco primeros polinomios de Chebyshev.

Fuente: [https://en.wikipedia.org/wiki/Chebyshev\\_polynomials](https://en.wikipedia.org/wiki/Chebyshev_polynomials)

Como puede observarse, las funciones base tienen un perfil sinusoidal en una dimensión. En dos dimensiones, este perfil se prolonga a lo largo de ambos ejes, tal y como se representa en la siguiente figura. El efecto resultante es el de un tren de pulsos en  $x$  e  $y$  que muestrea la superficie de la función  $f(x, y)$ .

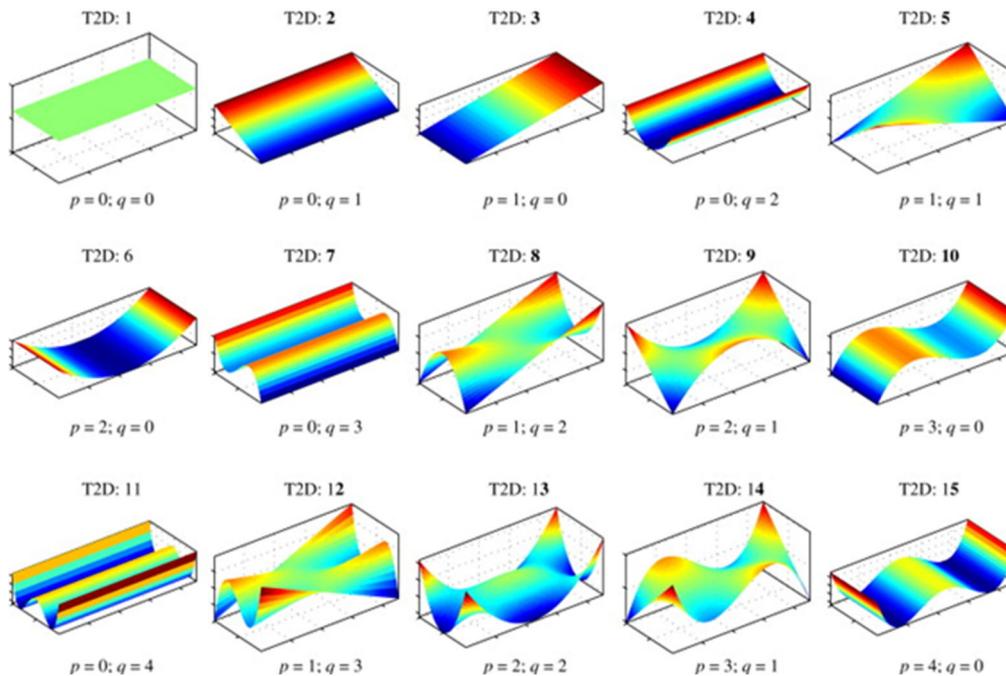


Figura 8. Polinomios de Chebyshev en dos dimensiones.

Fuente: Wang et al., 2011.

Como se demuestra en el trabajo de Marcos y Cristóbal (2013), existe una relación entre el orden del polinomio  $s = p + q$  y sus componentes frecuenciales. Así, órdenes mayores están asociados con mayores frecuencias de muestreo: un momento  $Tn$  de mayor magnitud que otro momento  $Tm$ , con  $n > m$ , refleja que la imagen original contiene estructuras que se repiten con mayor frecuencia. Si una textura está compuesta, por tanto, por elementos representativos de menor tamaño que se repiten más frecuentemente a lo largo de la imagen, dará lugar a momentos de orden alto de mayor magnitud.

A partir de esta observación, se propuso caracterizar la textura mediante la suma de las magnitudes de los momentos de un mismo orden.

Se obtiene como resultado una curva  $M(s)$ , función del orden  $s$ , tal y como se representa en la siguiente figura.

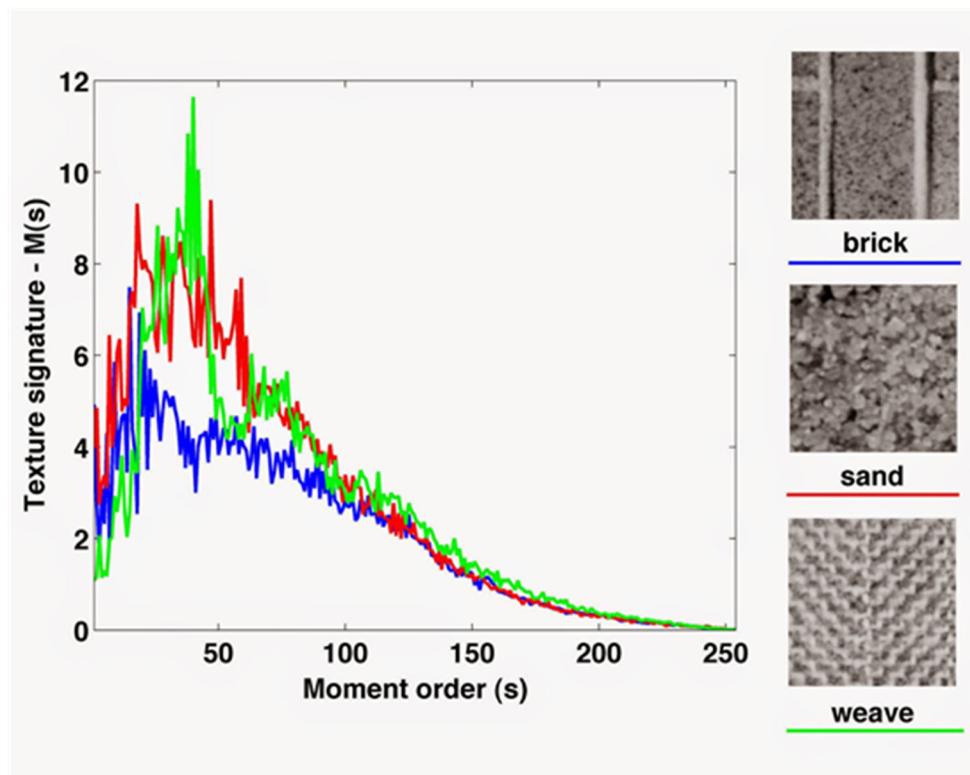


Figura 9. Firmas de textura para tres imágenes diferentes derivadas de los momentos Chebyshev de la imagen original.

Fuente: <http://eximarinis.blogspot.com.es/2014/05/a-novel-method-for-image-texture.html>.

Esta curva o firma de textura refleja la repetición de estructuras similares en la imagen con la frecuencia representada por el orden  $s$ . Así, la textura que muestra un muro de ladrillo se caracteriza por una curva con un pico en valores menores de  $s$  que las texturas de arena y tejido. En estas últimas, las estructuras que caracterizan a la textura son de menor tamaño y, por tanto, se encuentran en mayor número para una misma superficie de imagen considerada. Como resultado, las firmas de textura a las que dan lugar se concentran en valores mayores de  $s$ .

Puede observarse que el efecto de este operador es similar al de un banco de filtros paso banda centrados en diferentes frecuencias.

## 12.5. Referencias bibliográficas

- Marcos, J. V. y Cristóbal, G. (2013). Texture classification using discrete Tchebichef moments. *Journal of the Optical Society of America A*, 30(8), 1580-1591.
- Mukundan, R., Ong, S. H. y Lee, P. A. (2001). Image analysis by Chebyshev moments. *IEEE Trans. Image Process.* 10, 1357–1364.
- Qi, X., Shen, L., Zhao, G., Li, Q. y Pietikäinen, M. (2015). Globally rotation invariant multi-scale co-occurrence local binary pattern. *Image and Vision Computing*, 43, 16-26.
- Soh, L. K. y Tsatsoulis, C. (1999). Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. *IEEE Transactions on Geoscience and Remote Sensing*, 37, 780–795.
- Wang, W. et al. (2011). Finite element model updating from full-field vibration measurement using digital image correlation. *Journal of Sound and Vibration*, 330(8), 1599-1620.
- Zhang, J. y Tieniu T. (2002). Brief review of invariant texture analysis methods. *Pattern recognition*, 35(3), 735-747.

Percepción Computacional

---

# Extracción de características. Procesamientos multiescala y métodos avanzados

# Ideas clave

## 13.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

Sin llegar a entender la matemática detrás de estos complejos algoritmos de extracción de características, se dotará de los conceptos necesarios para entender la funcionalidad de las Wavelets, filtros de Gabor y algoritmo SIFT.

Las nociones aquí presentadas te permitirán entender en qué caso y qué algoritmo deberás de emplear y qué parámetros necesitas para su correcto funcionamiento.

## 13.2. Introducción a la extracción de características basada en procesamientos multiescala

**D**e todos los métodos presentados para la extracción de características, la mayoría de ellos carece de la posibilidad de reconocer patrones más complejos y elaborados como puede ser la identificación de un objeto. El análisis de imágenes basado en histogramas de color, variaciones de las intensidades o incluso reconocimiento de patrones para identificar una textura sufren, en muchos casos, de limitaciones para identificar la rotación de un objeto en una imagen o el hecho de que parte de un objeto esté oculto.

Un problema tan sencillo de definir como puede ser la identificación de un objeto independientemente de su rotación o traslación (como se puede ver en la imagen a continuación), es altamente complejo usando las extracciones de características presentadas hasta ahora, por no decir imposible.



Figura 1. Ejemplo de funcionamiento del algoritmo SIFT para la translación de una imagen.

Fuente: <http://timothykurek.com/10-sift-image-processing/introduction-to-sift-scale-invariant-feature-transform-opencv-on-sift-image-processing/>

Es por eso que surgen otras herramientas como son la transformada Wavelet, los filtros de Gabor o el algoritmo SIFT (*Scale Invariant Feature Transformation*) que son capaces de proporcionar características más robustas para:

- ▶ Mejorar la información extraída de una imagen y proporcionar una visión más detallada de texturas, variabilidad de la imagen u objetos presentes.
- ▶ Proporcionar características invariantes al cambio de escala, rotación y ruido que pueda estar presente en una captura de imagen.
- ▶ Resolver, en definitiva, problemas más avanzados de procesamiento de imagen.

No obstante, las matemáticas necesarias para comprender el funcionamiento de estos algoritmos son muy avanzadas y no se presentarán de forma detallada en este tema. Se planteará la **descripción de estos algoritmos de forma funcional** entendiendo qué hacen, cómo y por qué esto supone una ventaja con respecto a otros algoritmos de extracción de características.

Entre las **aplicaciones más comunes** de estos algoritmos están:

- ▶ Identificar a las personas en función de su huella u iris, también denominados **algoritmos de biometría**. La identificación basada en características biométricas precisa de matemática y algoritmos avanzados para procesar la imagen más allá de una variación de intensidades. Gracias a las Wavelets y los filtros de Gabor, podemos extraer características invariantes al reescalado del iris (hay que considerar que el iris y la pupila varían su tamaño casi un número infinito de veces al día, lo que dificulta el reconocimiento), intensidad de brillo así como el color de la imagen.
- ▶ El capturar con nuestro móvil una secuencia de fotos y hacer una panorámica. Los algoritmos como SIFT son capaces de encontrar esos puntos en común entre dos fotos y aplicar las transformadas necesarias de rotación y reescalado para poder fusionar las imágenes correctamente.
- ▶ La identificación de monedas en una máquina de expendedoras (*vending*). En algunas se capture una foto de la moneda (la cual no presenta siempre el mismo grado de rotación) y con ello se realiza la identificación del valor de la misma. Esta técnica requiere de Wavelets para extraer características invariantes a la rotación.

### 13.3. Definición de transformada Wavelet

**E**xisten numerosas aproximaciones para describir las Wavelets, en algunos libros conocidas como «ondículas» incluso «onditas». En este tema, y puesto que previamente hemos presentado en detalle la transformada de Fourier, vamos a partir de esta misma transformada para entender qué aspecto del que carecen las transformadas de Fourier pretenden cubrir las Wavelets.

Recordemos brevemente que la **transformada de Fourier** permite descomponer una señal en sus componentes sinusoidales de diferentes frecuencias; en otras palabras: puede pensarse que es una técnica matemática para transformar el punto de vista de una señal desde la base de tiempo a la base de la frecuencia. La transformada de Fourier cubre un amplio espectro de problemas, sin embargo, su **limitación** comienza cuando, dada una señal con gran variabilidad de frecuencia, no se puede determinar en qué momento del tiempo sucedió dicha variación de frecuencia.

Hay que recordar que la transformada de Fourier está pensada únicamente para el análisis de frecuencias de una señal temporal y, para ello, se olvida de toda componente temporal analizando las componentes espectrales (de frecuencia).

Cuando se observa una señal, producto de la transformación de Fourier, resulta imposible determinar cuándo ocurre un determinado evento o cuándo está presente una determinada frecuencia si las propiedades de la señal que se está analizando no cambian demasiado en el tiempo, es decir, si se está trabajando con una **señal estacionaria**.

Esta desventaja no resulta muy relevante, como en el caso de señales periódicas, por ejemplo. Sin embargo, un importante número de señales de interés presentan características no estacionarias o transitorias, tales como: una tendencia, cambios abruptos, comienzos o finales de eventos, etc.

A menudo, estas características no estacionarias resultan ser las secciones más interesantes de las señales, y la transformada de Fourier no está preparada para detectarlas o analizarlas.

## Evolucionar la transformada de Fourier: STFT

En un esfuerzo por corregir la deficiencia presentada en el punto previo, en 1946 Denis Gabor adaptó la transformada de Fourier para poder analizar una pequeña sección de la señal en un determinado tiempo mediante una especie de ventana. Esta

adaptación es la que se conoce como STFT (*Short Time Fourier Transform*), la cual lleva una señal del plano del tiempo al plano bidimensional de tiempo y frecuencia.

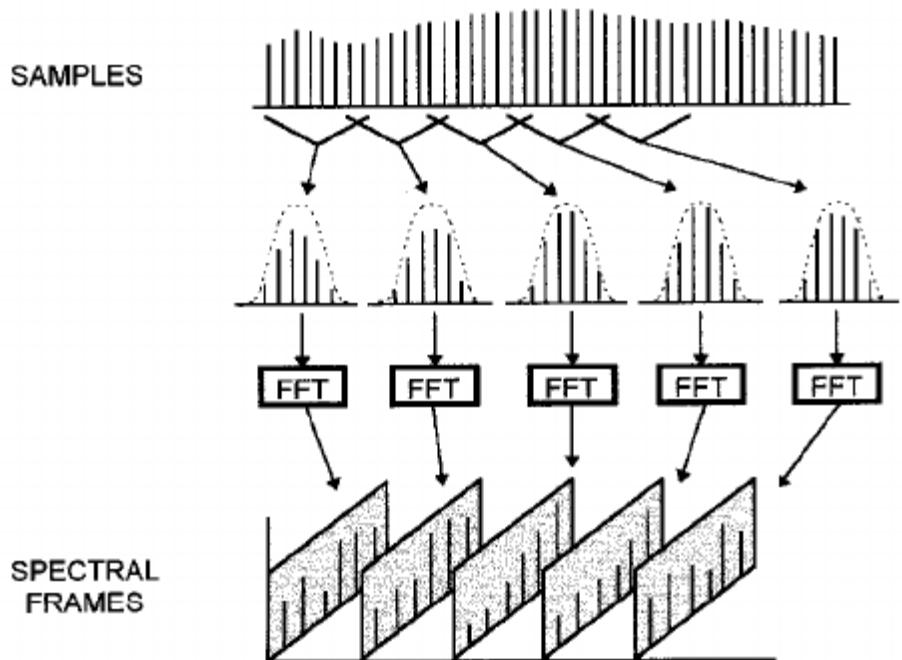


Figura 2. Funcionamiento visual de la Short Time Fourier Transform.

Fuente: [https://www.researchgate.net/figure/Short-time-Fourier-transform-STFT-with-envelope-and-two-sample-overlap\\_fig7\\_231828310](https://www.researchgate.net/figure/Short-time-Fourier-transform-STFT-with-envelope-and-two-sample-overlap_fig7_231828310)

En la anterior imagen se puede ver que, donde para cada ventana de tiempo, se calcula la transformada de Fourier (FFT), analizando posteriormente las componentes espectrales (de potencia por ventana)

Es importante mencionar que la STFT representa un compromiso entre el dominio del tiempo y el de la frecuencia de una señal, proporcionando información acerca de cuándo y a qué frecuencia de una señal ocurre un determinado evento. Sin embargo, solamente se puede obtener dicha información con una **precisión limitada**, la cual está acotada por el tamaño de la ventana.

A continuación se presentan dos ejemplos de STFT sobre bioseñales, concretamente electromiogramas. En el caso de la izquierda, se aprecia cómo la señal, a medida que pasa el tiempo, posee diferentes componentes en frecuencia. Dichas componentes no se habrían apreciado con un análisis en frecuencia basado en Fourier, ya que no

habría tenido en cuenta esa variación temporal de frecuencias. En resumen, en esta comparativa podemos ver cómo, en función del tiempo, se ven las diferentes componentes espectrales.

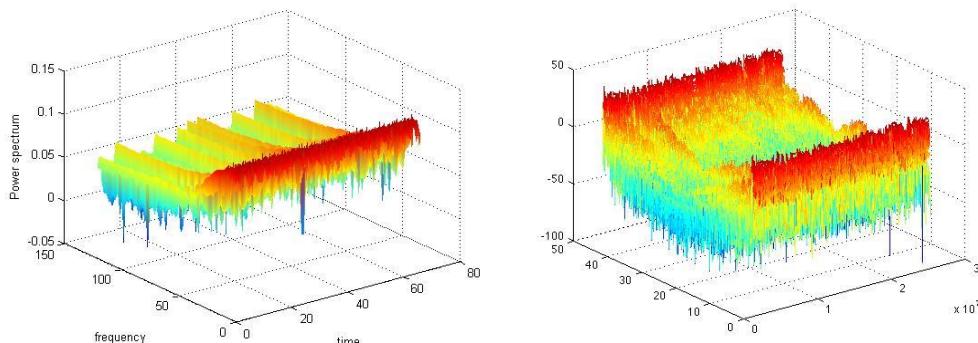


Figura 3. Dos visualizaciones de STFT de señales unidimensionales provenientes de electromiogramas.  
Fuente: Adaptado de <https://es.mathworks.com/matlabcentral/fileexchange/45405-short-time-fourier-transform-stft?focused=3809422&tab=function> y <https://stackoverflow.com/questions/14586685/stft-computation-without-using-spectrogram-function>

Mientras que el compromiso entre la información del tiempo y la frecuencia puede resultar útil, el inconveniente surge cuando, una vez que se escoge un determinado tamaño para la ventana de tiempo, dicha ventana es la misma para todas las frecuencias. Muchas señales requieren un acercamiento más flexible, de tal modo que sea posible **variar el tamaño de la ventana** para determinar con mayor precisión el tiempo o la frecuencia.

## Wavelets como evolución de STFT

El análisis Wavelet representa el paso lógico siguiente a la STFT; es una técnica que, mediante ventanas con **regiones de tamaño variable**, permite:

- ▶ El uso de intervalos grandes de tiempo en aquellos segmentos en los que se requiere mayor precisión en baja frecuencia.
- ▶ Regiones más pequeñas donde se requiere información en alta frecuencia.

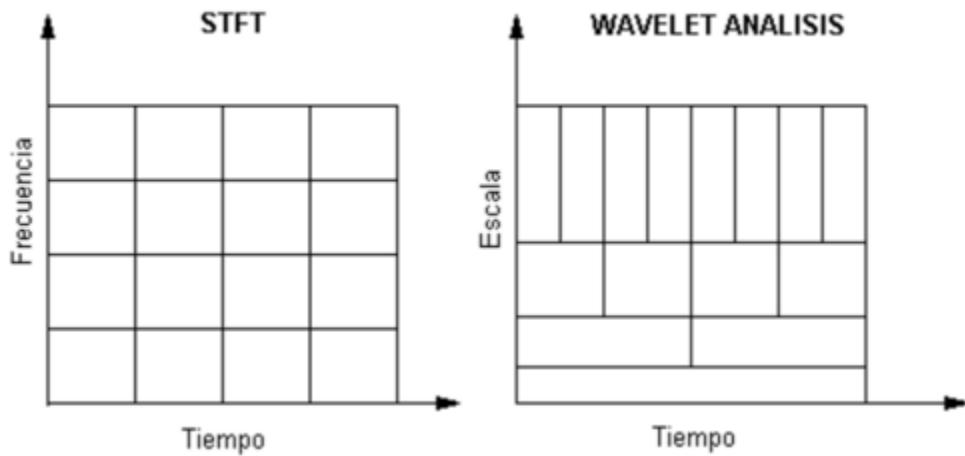


Figura 4. Comparativa de esquema entre STFT y Análisis con Wavelets.

Fuente: Adaptado de [https://www.researchgate.net/figure/a-STFT-with-fixed-aspect-ratio-b-wavelet-transform-with-variable-aspect-ratio\\_fig2\\_228413897](https://www.researchgate.net/figure/a-STFT-with-fixed-aspect-ratio-b-wavelet-transform-with-variable-aspect-ratio_fig2_228413897)

Esta transformada es eficiente para el análisis local de señales no estacionarias y de rápida transitoriedad. Al igual que la transformada de Fourier con ventana, mapea la señal en una representación de tiempo-escala; la diferencia está en que la transformada Wavelet provee **análisis de multirresolución con ventanas dilatadas**:

- ▶ El análisis de las frecuencias de mayor rango se realiza usando ventanas angostas.
- ▶ El análisis de las frecuencias de menor rango se hace utilizando ventanas anchas.

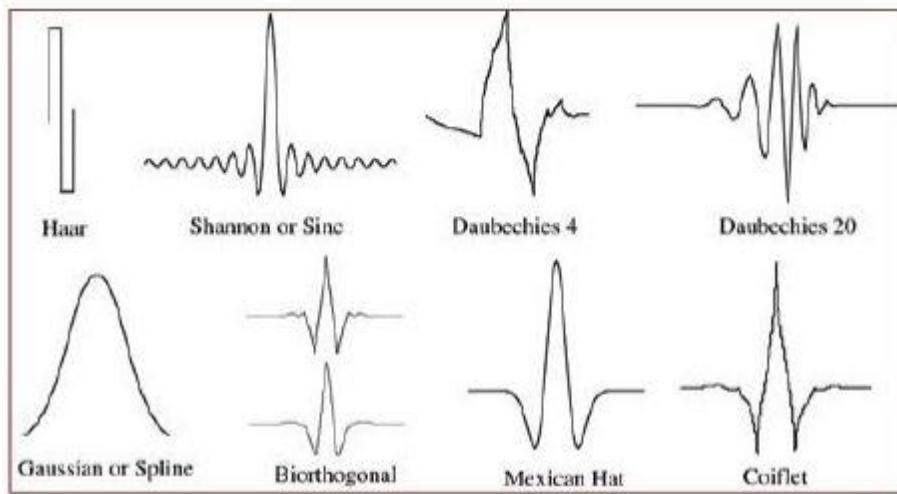


Figura 5. Resumen con las Wavelets más comunes.

Fuente: <http://www.continuummechanics.org/wavelets.html>

Una forma sencilla de comprender el modo de operación de esta transformada es pensar que la señal en base de tiempo es pasada por varios filtros paso bajo y paso

alto, los cuales permiten separar las porciones de la señal de alta frecuencia de aquellas de baja frecuencia.

Este procedimiento se repite cada vez sobre algunas porciones de la señal correspondientes a aquellas frecuencias que han sido eliminadas de la señal original.

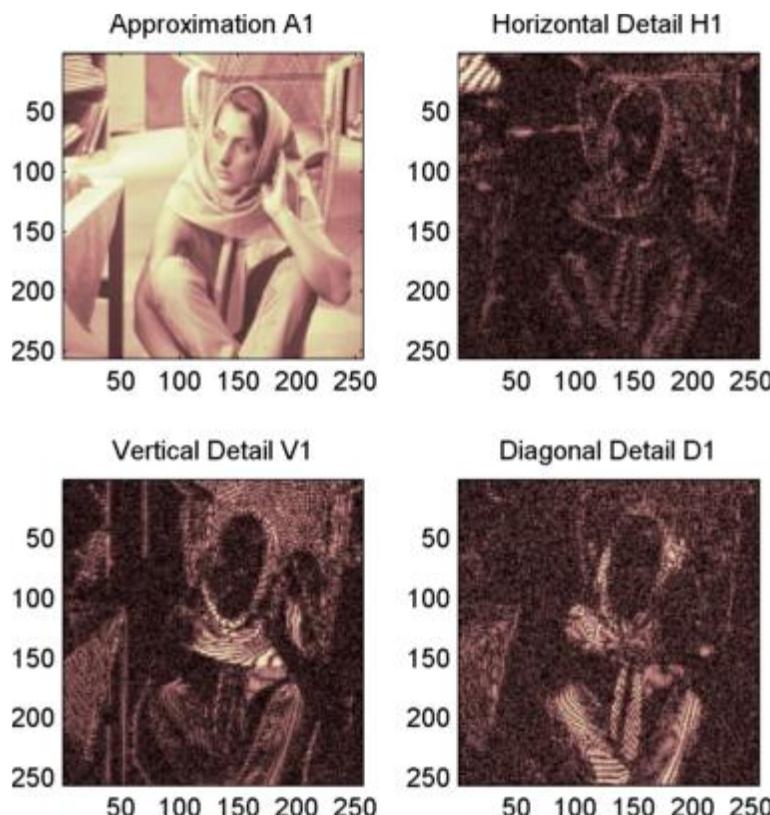


Figura 6. Ejemplo de descomposición multiescala en sus componentes horizontales, verticales y diagonales.

Fuente: <https://es.mathworks.com/help/wavelet/ug/two-dimensional-discrete-wavelet-analysis.html>

## Utilidades de la transformada Wavelet

Como ya se ha mencionado anteriormente, el procesamiento de señales a través de Wavelets tiene innumerables aplicaciones en diversos ámbitos de la ciencia e ingeniería. A continuación se describe una lista de ejemplos:

- ▶ Detección de **discontinuidades o de puntos de quiebre** en señales, en una o varias dimensiones: resulta de gran utilidad, en especial en el tratamiento de imágenes, en donde interesa detectar la frontera entre colores y formas, o también en

sistemas altamente dinámicos, en donde concierne determinar cuándo o dónde se producen los cambios.

- ▶ Estudio de fractales: mediante Wavelets se puede reconocer un **patrón repetitivo** en una señal o imagen, lo que la convierte en una herramienta poderosa en el estudio de fractales.
- ▶ Identificación de frecuencias puras: como se trata de una transformada compuesta por una base ortogonal de señales (análoga con la base sinusoidal de Fourier), también pueden ser utilizadas para **estudiar el contenido espectral** de señales.
- ▶ Eliminación de ruido: el análisis de señales mediante Wavelet también permite la **eliminación o filtrado de ruido** tanto en señales unidimensionales como en imágenes (bidimensionales).
- ▶ **Compresión** de imágenes: se trata de una de las aplicaciones más importantes de Wavelets, se realiza mediante el análisis en dos dimensiones.
- ▶ Aplicaciones en medicina: se ha incorporado el análisis con Wavelets a señales biológicas, permitiendo interpretar los resultados de exámenes médicos, facilitando el **diagnóstico de las enfermedades**. Por ejemplo, se ha aplicado con éxito en el análisis de electroencefalogramas, debido a que la naturaleza de este tipo de señales son altamente no estacionarias (impidiendo el uso de Fourier):
  - Las Wavelets permiten transformar la señal al dominio tiempo-frecuencia, relacionando el contenido espectral al momento de su ocurrencia. Ha sido aplicado en el diagnóstico de pacientes con alzheimer, enfermedad que hasta ahora era difícil de diagnosticar. El procedimiento que se realiza es entrenar redes neuronales con los datos obtenidos (a través de Wavelets de multiresolución) de pacientes de los que se sabe padecen la enfermedad, para luego introducir las mediciones de aquellos que se realizan exámenes, y de este modo poder establecer si se siguen los mismos patrones.

## 13.4. Filtros de Gabor

**L**os filtros de Gabor son una particularidad de la STFT donde la ventana tiene una distribución gaussiana. Detrás de los filtros de Gabor existe una explicación basada en la naturaleza. De hecho, tanto las Wavelets como los filtros de Gabor son **los que mejor imitan el funcionamiento del ojo natural**.

Desde la década de 1980 es bien conocido que las células simples de la corteza visual de nuestro cerebro presentan una sensibilidad especial hacia la orientación y la frecuencia espacial en una escena visual.

Por **orientación** nos referimos al ángulo formado por un objeto y un eje de referencia elegido arbitrariamente, mientras que la **frecuencia** espacial da cuenta de la repetición de un patrón en el espacio, medido en unidades inversas de longitud.

Dichas células aparecían en agrupaciones binarias, donde cada par era capaz de detectar una cierta orientación y frecuencia. Sin embargo, había una diferencia de fase de  $\pi/2$  radianes entre ambas células, es decir: mientras la primera podía detectar objetos simétricos o repetidos (líneas paralelas), la otra se encargaba de los objetos asimétricos.

Al contrario que en las Wavelets, que son unidimensionales, los filtros de Gabor son bidimensionales, convirtiéndolo en una excelente herramienta para la descomposición multiescala de imágenes.



Figura 7. Representación de la parte real de un filtro de Gabor.  
Fuente: <http://www.di.uniba.it/~ig/GaborFilter/html/filtro.html>.

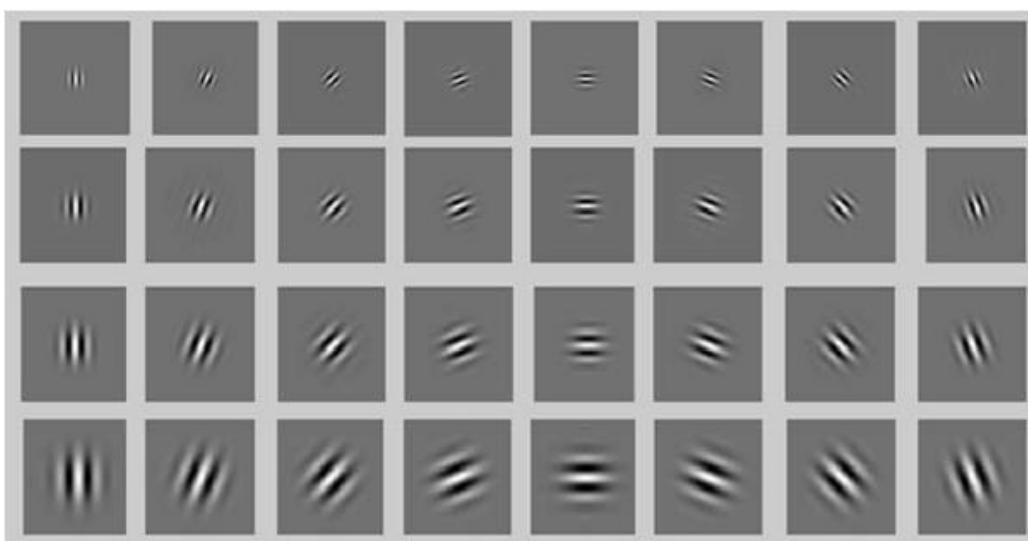


Figura 8. Parte real de los filtros Gabor con cuatro diferentes frecuencias y ocho orientaciones.  
Fuente: [https://www.researchgate.net/figure/Gabor-wavelets-filter-bank-which-has-been-used-for-the-active-basis-model\\_fig2\\_262781727](https://www.researchgate.net/figure/Gabor-wavelets-filter-bank-which-has-been-used-for-the-active-basis-model_fig2_262781727).

El concepto esencial detrás de los filtros de Gabor es similar al de las Wavelets y al de Fourier: descomponer una imagen o una señal en determinadas componentes y, con ello, poder definir unas características que proporcionen información adicional de la imagen para su mejor entendimiento. Si una imagen se convoluciona con todas y cada una de las funciones arriba representadas, se detectarán no solo bordes y contornos, sino también diferentes texturas o patrones.

Las aplicaciones más comunes de los filtros de Gabor, más allá de la detección de texturas, están relacionadas con la **biometría** o, lo que es lo mismo, la verificación e identificación de usuarios con base en patrones biométricos como la huella o el iris. Concretamente, son huella e iris los que más emplean filtros de Gabor para el correcto funcionamiento.

A continuación se presenta un ejemplo de cómo se aplican los filtros de Gabor (y el resultado obtenido) en un caso de uso real de biometría de iris. El objetivo es reducir la complejidad de textura del iris a un patrón que pueda ser comparado con otros patrones mediante una distancia sencilla (como puede ser la distancia de Hamming). De ahí que el resultado, después de obtener el filtro de Gabor, parezca un código QR.

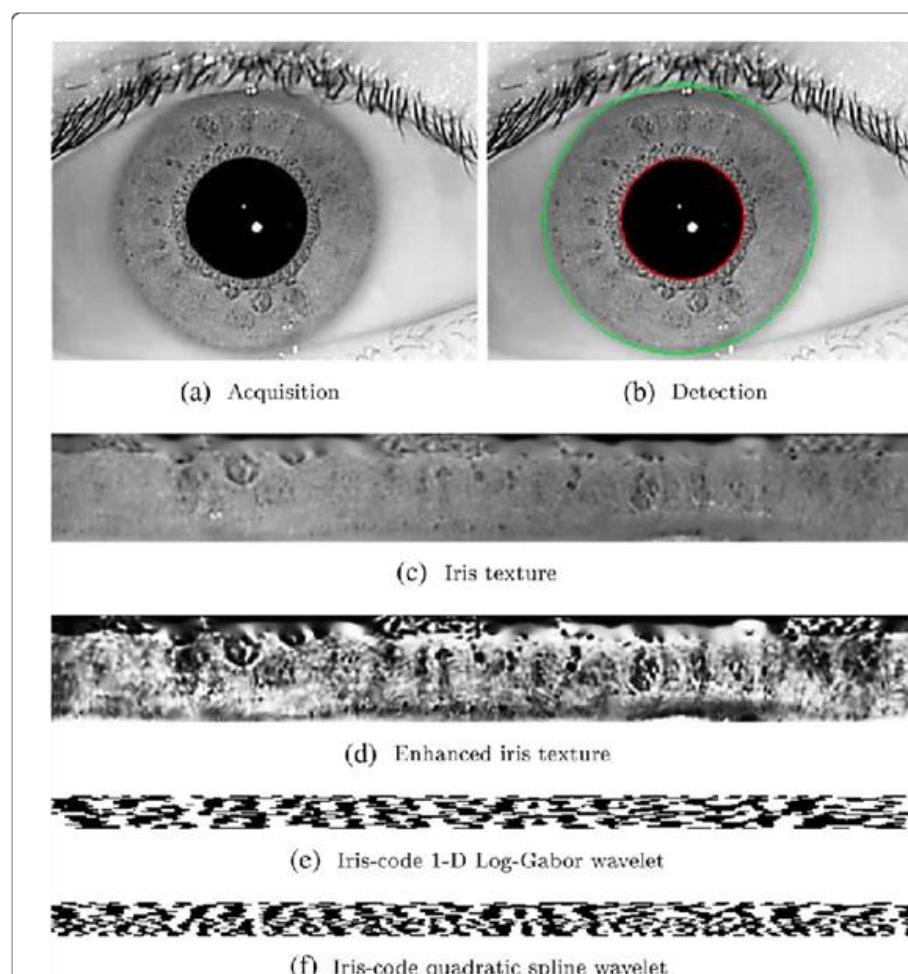


Figura 9. Funcionamiento de un sistema biométrico basado en iris con filtros de Gabor.

Fuente: [https://www.researchgate.net/publication/309659737\\_Unlinkable\\_improved\\_multibiometric\\_iris\\_fuzzy\\_vault](https://www.researchgate.net/publication/309659737_Unlinkable_improved_multibiometric_iris_fuzzy_vault)

En esta imagen vemos:

- ▶ Imagen original (**a**).
- ▶ Imagen segmentada, detectando la región de iris a procesar (**b**).
- ▶ La textura del iris convertida a coordenadas rectangulares (**c**).
- ▶ Textura con alto contraste (**d**).
- ▶ Código de iris después de aplicar filtro de Gabor (**e**).
- ▶ Código de iris después de aplicar una wavelet cuadrática (**f**).

## 13.5. Transformada SIFT

**E**sta transformada es de las que tiene más utilidad hoy en día y que resuelve problemas muy complejos en procesamiento de imágenes. Uno de ellos es el de encontrar objetos parcialmente ocultos en imágenes. Vamos a verlo con el siguiente ejemplo:



Figura 10. Localización de objetos parcialmente ocultos en una imagen.

Fuente: <http://eric-yuan.me/sift/>.

A la izquierda de esta imagen se pueden ver dos objetos: una rana y una máquina de tren. En la imagen de la derecha, hay una rana (oculta detrás de un objeto negro), así como las dos máquinas de tren (una de ellas incluso en vertical).

Si intentamos responder a la pregunta: «¿Están los objetos tren y rana dentro de la imagen?», no podríamos dar respuesta con ninguno de los métodos presentados hasta este tema. Con las Wavelets y filtros de Gabor podríamos habernos acercado a detectar alguna característica común, pero en ningún momento con la precisión que el algoritmo SIFT lo realiza. Las iniciales del algoritmo significan *Scale Invariant Feature Transform* y proporciona el siguiente resultado. En ella se aprecian varios recuadros y colores:

- ▶ Cada **color** corresponde con un objeto encontrado: amarillo y verde para las máquinas de tren, rojo para la rana.
- ▶ El **recuadro más grande** indica dónde está el objeto en su totalidad.
- ▶ Los recuadros más pequeños indican las **características** que se han detectado tanto en las imágenes de los objetos originales como en la imagen donde se encuentran los objetos ocultos.
- ▶ Cada uno de estos recuadros posee un tamaño y una orientación, ambos parámetros son devueltos por el algoritmo SIFT.



Figura 11. Resultado del algoritmo SIFT.

Fuente: <http://eric-yuan.me/sift/>.

En esta imagen se ve en diferentes colores y con un gran recuadro la ubicación de los objetos buscados. Además, se presentan también, con recuadros más pequeños, las características detectadas.

Veamos ahora el **funcionamiento de este algoritmo**.

Para obtener un conjunto de descriptores SIFT de una imagen es necesario, por un lado, obtener los puntos característicos y después, para cada punto de interés, calcular su vector descriptor a partir de la información de los píxeles que lo rodean. SIFT fue propuesto para imágenes en escala de grises, por lo que el vector de características de 128 elementos que define cada píxel contiene información sobre cómo se distribuyen los niveles de intensidad alrededor de cada punto de interés previamente obtenido.

Por lo tanto, el algoritmo consta de dos partes claramente diferenciadas:

- ▶ Obtención de los puntos característicos.
- ▶ Descripción de la región alrededor de cada punto de interés.

La **obtención de los puntos característicos** o puntos de interés, a los que habitualmente se llama en inglés *keypoints*, consiste en detectar aquellas regiones de la imagen en las que se producen diferencias de gradiente significativas a ambos lados de dicho punto. Si el método solamente hiciera eso, se podría pensar que esta etapa se realizaría utilizando un detector de esquinas. SIFT propone no detectar únicamente esquinas, sino *blobs*, y hacerlo de manera que esa detección sea consistente cuando el punto característico aparezca a diferentes escalas.

La diferencia entre una esquina (*corner*) y un *blob* se explica a continuación: si nos fijamos en su definición, una **esquina** es un punto o un área pequeña en una imagen donde confluyen, al menos, dos bordes. Se define un ***blob*** como una región de una imagen que se caracteriza porque algunas de sus propiedades se mantienen aproximadamente constantes. Dentro del contexto de la detección de puntos

característicos (*keypoints*), la propiedad que se suele considerar constante en un *blob* es la **similitud en su nivel de gris**, típicamente medida a partir de la variación del gradiente en esa región a lo largo de diferentes direcciones.

En resumen, buscamos puntos que pueden ser pequeñas regiones con un nivel de intensidad estable y alrededor de las cuales se produce una variación significativa del gradiente en más de una dirección.

Para obtener el **descriptor de cada punto característico**, se calcula un histograma de direcciones del gradiente local alrededor del punto de interés. El descriptor que se obtiene se convierte en invariante a la escala al normalizar el tamaño del vecindario local al punto de interés en función de ella. Además, es invariante a la rotación porque se determina la orientación dominante de los vectores del gradiente en el vecindario del punto característico, y se aplica dicha información para orientar la rejilla utilizada para calcular el histograma.

En resumen, los pasos a seguir son:

1. Detección de extremos en el espacio-escala: se buscan puntos de interés en toda la imagen y en todas las escalas consideradas utilizando una diferencia de gaussianas.
2. Localización precisa de puntos característicos: para cada uno de los puntos de interés anteriores se ajusta un modelo que permite determinar su localización y escala. Además, se seleccionan los puntos característicos, *keypoints*, eliminando los que están próximos a los bordes o tienen bajo contraste.
3. Asignación de la orientación: a cada punto característico se le asigna una o varias orientaciones en función de las direcciones del gradiente local. Esta orientación, conjuntamente con la ubicación y la escala calculadas anteriormente, permite que el descriptor sea invariante a estas tres situaciones.
4. Descripción del punto característico: alrededor de cada punto característico se miden los gradientes locales de la imagen y se utiliza su histograma para obtener

una representación de esa región que es robusta a cambios significativos en la iluminación y a pequeñas distorsiones en la forma.



Figura 12. Detección con SIFT de objetos complejos (y sin formas definidas) en una imagen.

Fuente: <http://slideplayer.com/slide/5282534/>.

Percepción Computacional

---

# Decisión. Principios e implementación de algoritmos de ayuda en la toma de decisiones

## 14.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

**E**n la mayoría de las aplicaciones, el procesado completo de una señal requiere la interpretación automática de esta como etapa final. Por ejemplo, en el caso de una señal biomédica, el objetivo final irá dirigido a determinar si dicha señal denota la existencia o no de una patología. Podemos, asimismo, pensar en una aplicación basada en el manejo de señales de voz que nos permita identificar si la palabra o frase pronunciada forma parte de nuestro diccionario particular, que habrá sido definido previamente. En imágenes es habitual tener que identificar determinados objetos, por ejemplo, conocer si corresponde o no a una persona; por lo que tras reducir el ruido de la imagen, incrementar el contraste de esta o llevar a cabo una segmentación, será necesario clasificar los elementos resultantes.

Todas estas aplicaciones precisan como elemento culminante una etapa de **reconocimiento de patrones**. Esta será específica de la aplicación en cuestión y, por tanto, su construcción se adaptará a la tarea concreta que deberá llevar a cabo. Así, el sistema de reconocimiento de patrones para la identificación de palabras es diferente al empleado para hallar automáticamente una persona en una imagen. Sin embargo, la metodología para su construcción es común en ambos casos.

Dicha **metodología** se basa en la disponibilidad de un conjunto amplio de ejemplos del problema a resolver. Por ejemplo, a fin de implementar un sistema de reconocimiento de voz, este conjunto deberá incluir tanto palabras que no están en nuestro diccionario como las que sí lo están. O en el caso de las imágenes,

instantáneas de figuras que no se corresponden con una persona frente a las que sí lo son. A partir de estos ejemplos, es posible inferir qué propiedades diferencian unos casos de otros, resultando en un **modelo**.

En este punto, cabe destacar que cada una de estas tareas es realizada por nuestro cerebro, que es capaz de identificar una palabra conocida o si el elemento de una imagen es una persona. En él se llevan a cabo las etapas previas de tratamiento de señal que se han visto en la asignatura: captura, eliminación de ruido, segmentación y extracción de características a partir de la fuente de información original.

La etapa inmediatamente anterior al sistema de reconocimiento de patrones es la **extracción de características** y sobre la que hemos tratado en temas previos. En esta etapa, una vez que tenemos identificado el objeto a interpretar (una señal o segmento de la misma, una imagen, una textura, una silueta, etc.), se busca sintetizar la información de dicho objeto en un conjunto de variables. Este ejercicio de síntesis conlleva la reducción de la dimensionalidad de los datos de entrada, de una imagen o señal formados por un número determinado de píxeles o puntos, pasamos a un vector de características de longitud notablemente menor.

La motivación principal de la etapa de extracción de características es optimizar, mediante esta síntesis, la implementación del sistema de reconocimiento de patrones posterior. Para ello, además de permitir trabajar en un espacio de dimensión menor, es de vital importancia que las características extraídas definan propiedades significativas de los objetos en el contexto del problema a resolver. Por ejemplo, el valor máximo de una señal de voz puede no tener relevancia si lo que se pretende identificar es si ha sido emitida por un hombre o una mujer, pero sí si se pretende inferir si el emisor de la señal muestra un estado de tranquilidad o no. Es decir, la etapa de extracción de características ha de minimizar la pérdida de información resultante de la reducción de dimensionalidad llevada a cabo, reteniendo aquellas propiedades más relevantes de la fuente de información.

La siguiente imagen recoge, en forma de esquema, los principales conceptos de este tema.

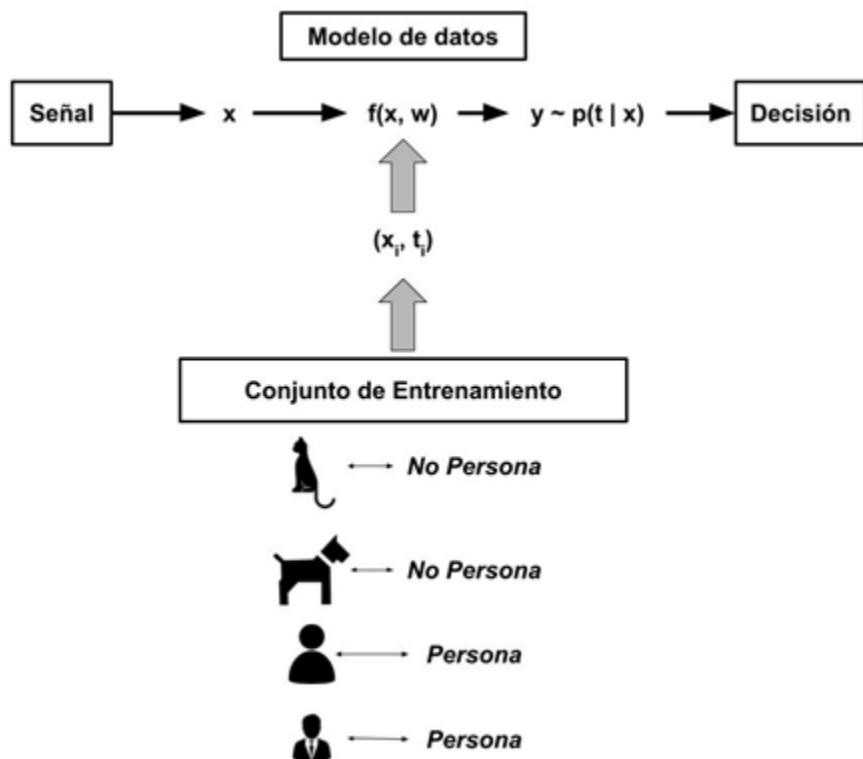


Figura 1. Metodología para la implementación de un sistema de decisión.

## 14.2. Clasificación y reconocimiento de patrones

**E**l sistema de reconocimiento de patrones es el **modelo** resultante de inferir las diferencias existentes entre varias categorías de objetos. Dicho modelo, como se expresaba al comienzo de este tema, se obtiene a partir de un conjunto de ejemplos de todas las categorías involucradas en nuestra aplicación. Finalmente, nuestro modelo se deriva de un proceso de optimización guiado por un algoritmo y una métrica de evaluación a maximizar/minimizar. Este algoritmo suele ser de carácter iterativo, de manera que se va dando forma al modelo mediante la observación repetida de los diferentes ejemplos.

Como se ha comentado, la implementación del sistema de decisión se realiza a partir de:

- ▶ Un conjunto de ejemplos  $D$ , referido comúnmente como **conjunto de entrenamiento**.
- ▶ Cada uno de los ejemplos en este conjunto viene dado por el par formado por:
  - Un objeto  $x$ .
  - Y la categoría a la que pertenece  $t$ .
- ▶ Este escenario se conoce en el ámbito del aprendizaje automático (*machine learning*) como aprendizaje supervisado, ya que se sabe *a priori* el valor de la variable objetivo para los datos de entrenamiento disponibles.

El conjunto  $D$  puede representarse, por tanto, de la siguiente forma:

$$D = \{(x_i, t_i)\}_{i=1,\dots,M}$$

Donde  $M$  es el número total de ejemplos en el conjunto de entrenamiento.

La pregunta es cómo tomar decisiones, minimizando el error en la decisión, ante objetos  $x$  que no están en  $D$  y para los que desconocemos la categoría a la que pertenecen.

Estadísticamente, la herramienta que proporciona una descripción completa de las muestras en  $D$  es la función de densidad de probabilidad conjunta  $p(x, t)$ , que puede expresarse como:

$$p(x, t) = p(t|x)p(x)$$

Donde:

- ▶  $p(t|x)$  es la función de densidad de probabilidad de la variable  $t$  dado  $x$ .
- ▶  $p(x)$  es la función de densidad de probabilidad de  $x$ .

En este punto, puede observarse que  $p(t|x)$  es la función a modelar a fin de hacer predicciones sobre  $t$  a partir de un valor observado de  $x$ . Es decir, esta será la **función a modelar por nuestro sistema de decisión**.

## Clasificación

Cuando la variable  $t$  es categórica, el problema se corresponde con una tarea de clasificación.

Matemáticamente consiste en determinar, para una muestra  $x$ , a cuál de las  $C$  categorías posibles ( $C_1, C_2, \dots, C_k$ ) pertenece. El objeto de entrada queda descrito por un conjunto de características o descriptores cuantitativos, por lo que  $x$  será un vector de la forma  $x = [x_1, x_2, \dots, x_n]$ , que representa un punto en el espacio de entrada  $n$ -dimensional.

El **clasificador** viene dado por la función  $f$  que mapea el espacio de entrada  $\Re^n$  en la variable categórica  $t$ :

$$f: \Re^n \rightarrow t \in \{C_1, C_2, \dots, C_k\}$$

La salida del clasificador puede expresarse como  $y = f(x, w)$ , donde se hace explícita la dependencia de la función de mapeo  $f$  con el vector de entrada  $x$  y un conjunto de parámetros adaptativos  $w$ .

La aproximación estadística al problema de clasificación, es decir, para la estimación de la función  $f$ , asume que las muestras de  $x$  que corresponden a la categoría  $C_i$  han sido generadas de acuerdo a la función de densidad de probabilidad  $p(x|C_i)$ .

En este escenario, la **regla de decisión de Bayes** determina cómo ha de llevarse a cabo la decisión a fin de minimizar el riesgo en la predicción. De acuerdo a esta regla, un objeto  $x$  debe asignarse a la categoría  $C_i$  para la que el riesgo condicionado  $R(C_i|x)$  es mínimo.

Este riesgo se define como sigue:

$$R(C_i|x) = \sum_{j=1}^k L(C_i, C_j) p(C_j|x)$$

Donde:

- ▶ Donde la función  $L(C_i, C_j)$  cuantifica el **coste del error** cuando se decide  $C_i$  siendo  $C_j$  la verdadera categoría a la que pertenece la muestra  $x$ .
- ▶  $p(C_j|x)$  es la probabilidad *a posteriori* de la categoría  $C_j$ . Se conoce también como **función de pérdida**. Normalmente, se emplea una función de pérdida 0/1 para la implementación del clasificador, que toma un valor:
  - 0 si  $i = j$ .
  - 1 si  $i \neq j$ .

El resultado de utilizar esta función de pérdidas es la denominada como **regla del máximo a posteriori** (MAP, del inglés *maximum a posteriori*). Supone la forma más conocida de la regla de decisión de Bayes. Una muestra  $x$  se asigna a la categoría  $C_i$  si se cumple la siguiente condición:

$$p(C_i|x) > p(C_j|x), i \neq j$$

Es decir, la decisión será tomar aquella categoría más probable una vez que el valor de  $x$  es conocido.

La estrategia descrita al problema de reconocimiento de patrones y, concretamente, a la clasificación representa una aproximación estadística al mismo. En ella, se pretende obtener una estimación de la función  $p(C_i|t)$  a fin de poder tomar decisiones de acuerdo a la regla de Bayes. Muchos de los **algoritmos para la implementación de modelos de datos** se basan en esta aproximación, es el caso de la regresión logística o las redes neuronales MLP y RBF. Asimismo, en otros modelos

estadísticos se estima la función de densidad condicionada  $p(x|C_i)$  para obtener  $p(C_i|x)$  mediante el teorema de Bayes:

$$p(C_i|x) = p(x|C_i)p(C_i)/p(x)$$

Algunos algoritmos que siguen esta estrategia son K-vecinos próximos, las redes neuronales probabilísticas, los clasificadores gaussianos (discriminante lineal o cuadrático) o el Naïve Bayes.

Fuera de este marco puramente estadístico, podemos encontrar métodos de clasificación como los árboles de decisión o las máquinas de vectores soporte. En ambos casos, no se persigue una estimación directa de las funciones de densidad de probabilidad mencionadas previamente.

## Entrenamiento

En principio, la forma de la función  $f(x, w)$  es desconocida y se requiere de un conjunto de ejemplos  $D$  para su estimación. Por tanto, la función  $f$  vendrá dada por una expresión matemática que contiene un número determinado de parámetros ajustables  $w$ . Esta expresión está determinada por el conjunto de ejemplos entrada-salida en  $D$  mediante el proceso de aprendizaje o entrenamiento. Por este motivo, el conjunto de ejemplos de muestras de entrada y su correspondiente categoría se denomina conjunto de entrenamiento.

El proceso de entrenamiento conlleva la minimización de una función de error que cuantifica la distancia o grado de disparidad entre el valor de salida de nuestro clasificador ( $y$ ) y su valor objetivo correspondiente ( $t$ ) para un objeto dado ( $x$ ). En este caso, se habla de entrenamiento supervisado, ya que cada muestra  $x$  está asociada a un valor objetivo  $t$ .

La finalidad del proceso de entrenamiento es construir un modelo estadístico del proceso que da lugar a las muestras  $x$ . Es decir, no se busca una función  $f(x, w)$  cuyo

valor coincida exactamente con la variable  $t$  para cada par  $(x, t)$ , sino que dicha función se aproxime a  $p(t|x)$ . Una estimación plausible de esta función de densidad de probabilidad resultará en un modelo con una gran **capacidad de generalización**; este concepto se refiere a la capacidad del modelo de producir respuestas precisas ante muestras de entrada que no se han procesado previamente. Fundamentalmente, esta capacidad de generalización depende de tres factores:

**El tamaño del conjunto de entrenamiento y su representatividad del problema.** Será necesario contar con un número elevado de ejemplos diferentes entre sí para lograr una buena descripción del problema. Cada ejemplo en  $D$  es, al fin y al cabo, una muestra de la función  $p(x, t)$ , por lo que su estimación será más precisa si se dispone de un mayor número de muestras.

**La capacidad de adaptación (flexibilidad) del modelo.** Cuanto mayor sea el número de funciones  $f(x, w)$  que pueden definirse mediante el ajuste de  $w$ , mayor es la posibilidad de encontrar una solución que minimice la distancia respecto de la función objetivo  $p(t|x)$ , resultando en una mejor capacidad de generalización. Este número de funciones a las que el modelo puede dar lugar viene dado por la cantidad de parámetros  $w$  ajustables en el mismo. Así, cuantos más parámetros se deban optimizar durante el entrenamiento, el modelo tendrá una capacidad de adaptación mayor a las muestras de  $p(t|x)$  recogidas en  $D$ .

**La complejidad del problema de clasificación.** Un problema de clasificación de mayor dificultad para modelar vendrá dado por una función  $p(t|x)$  más compleja. La siguiente imagen (figura 2) muestra un ejemplo en el que se aprecian dos conjuntos de muestras  $(x, t)$ :

- ▶ D1 (izquierda) y D2 (derecha) describen dos problemas de clasificación diferentes.
- ▶ Como se aprecia, el problema definido por D2 es de mayor dificultad, pues requiere una frontera de decisión en el espacio bidimensional de entrada de la forma:

$$(x_1 - k_1)^2 + (x_2 - k_2)^2 = k_3$$

Donde  $k_1$ ,  $k_2$  y  $k_3$  son constantes y los objetos  $x$  de entrada quedan definidos por los valores de las variables  $x_1$  y  $x_2$ .

- ▶ Mientras tanto, el problema descrito por el conjunto de muestras D1 requiere una frontera de decisión de la siguiente forma:

$$x_1 = k_1x_2 + k_2$$

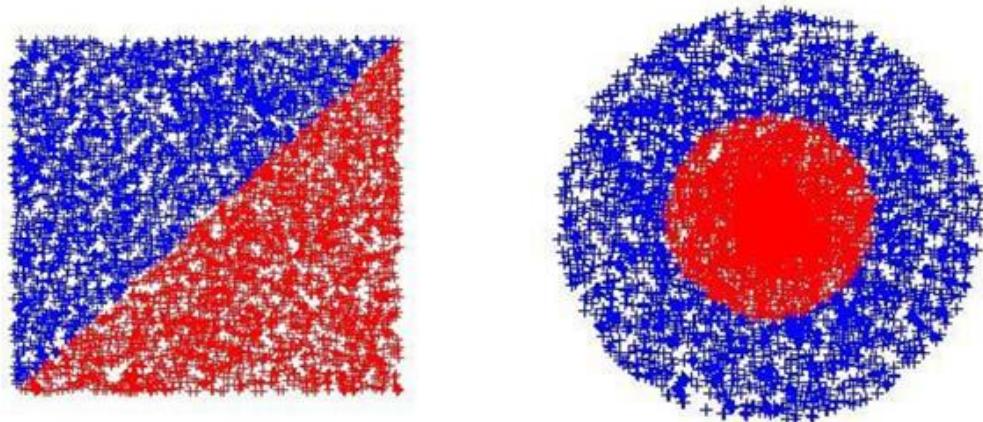


Figura 2. Ejemplo de dos problemas de clasificación diferentes.

Como comentábamos, los datos de la imagen de la derecha reflejan una mayor complejidad de la función  $p(t|x)$  que los que generó respecto a los observados en la imagen de la izquierda.

Mientras la dificultad del problema de clasificación resulta un factor fuera de control para el usuario, los dos primeros factores están estrechamente relacionados. Un **modelo rígido o poco flexible**, con un número reducido de parámetros ajustables durante el entrenamiento, puede no ajustarse a las muestras de la función a aproximar. Lo contrario sucede con un modelo excesivamente **flexible y maleable**, pues será capaz de ajustar perfectamente los pares en  $D$ .

En ambas situaciones, la capacidad de generalización del modelo resultante será pobre.

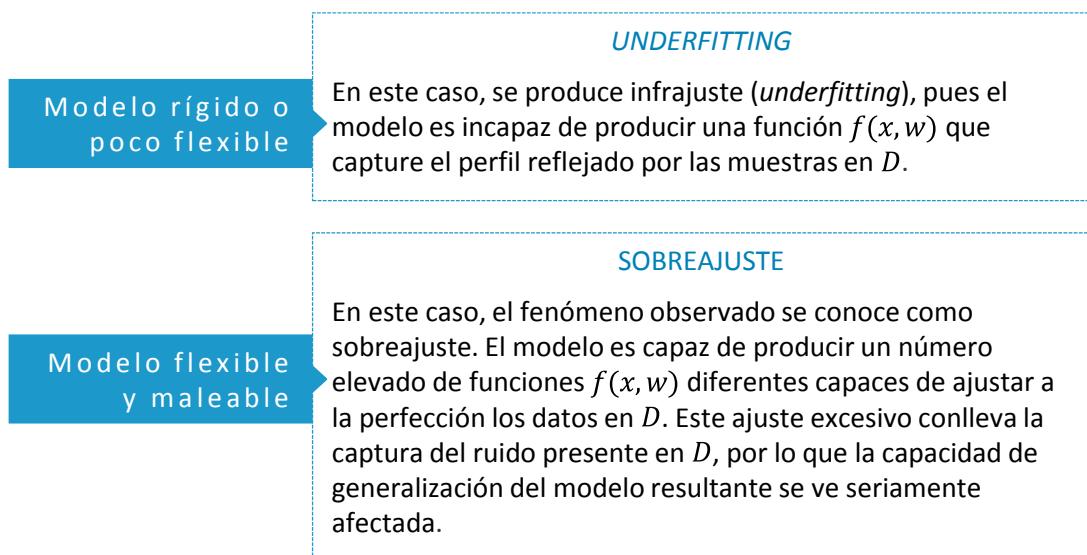


Figura 3. Problemas de infrajuste o sobreajuste en la capacidad de generalización del modelo.

Existen diferentes estrategias para paliar ambos efectos:

- ▶ En el caso del *underfitting*, la solución más efectiva es emplear modelos con un mayor grado de flexibilidad, de forma que sean capaces de generar funciones  $f(x, w)$  de un perfil más complejo.
- ▶ En el caso del *overfitting* (sobreajuste), puede prevenirse mediante la utilización de un conjunto de entrenamiento más amplio. Sin embargo, esto no es posible en la mayoría de los escenarios reales. Por tanto, se suele optar por otras soluciones más fácilmente alcanzables como la reducción de la dimensión del espacio de entrada o la combinación de modelos diferentes entre sí, técnicas conocidas como *ensembles* de las que *bagging* y *gradient boosting* son las más comúnmente empleadas.

## Sesgo y varianza

Los conceptos de infrajuste y sobreajuste mencionados se encuentran estrechamente relacionados con el sesgo y la varianza de un modelo de datos. Estos representan dos componentes distintas del error en el que incurre un modelo de datos. Antes de definirlas, la siguiente imagen ilustra la idea detrás de ambas componentes.

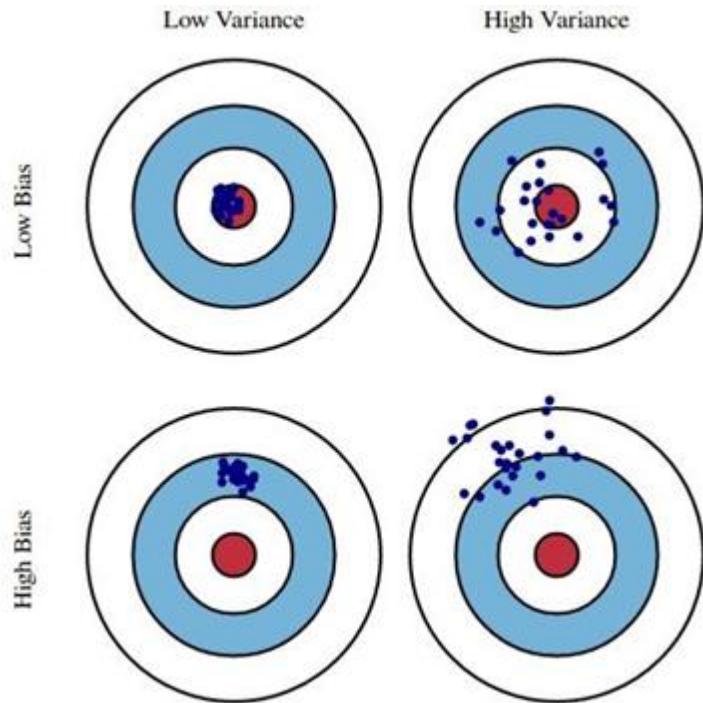


Figura 4. Ilustración de los conceptos de sesgo y varianza como fuentes de error en un modelo de datos.

Fuente: <https://www.kdnuggets.com/2016/08/bias-variance-tradeoff-overview.html>

- ▶ Si atendemos en primer lugar al sesgo, vemos que, cuando este es elevado en un modelo, el error cometido por este se debe a una tendencia clara en la predicción. En la imagen anterior, el sesgo del modelo se refleja en una tendencia a estimar por encima del objetivo. Esta tendencia desaparece cuando el modelo tiene un sesgo bajo.
- ▶ Por otro lado, la varianza se manifiesta en una distribución más amplia de las predicciones generadas por el modelo a pesar de que el objetivo, como en el ejemplo, no cambia. Así, cuando la varianza del modelo es elevada, hay mayor grado de incertidumbre sobre la salida de este.

Por tanto, tras este primer análisis, podemos definir sesgo y varianza:

El **sesgo** obedece a la componente sistemática del error. Dada una entrada  $x$ , se computa como la diferencia entre el valor medio de las predicciones de los modelos derivados de diferentes conjuntos de entrenamiento  $D$  y el valor objetivo  $t$  asociado a  $x$ .

$$Bias^2 = [E_D\{y(x)\} - t|x]^2$$

El error asociado a un sesgo elevado es propio de modelos simples, con poca flexibilidad o capacidad de ajuste. Estos modelos son incapaces de aproximar funciones complejas y tienden a producir infraestimación o sobreestimación. Esta situación refleja infrajuste de los datos en  $D$ .

La **varianza** es la componente del error derivada de la dependencia del modelo resultante  $f(x, w)$  con el conjunto de entrenamiento  $D$ . Es decir, este término refleja que las funciones  $f(x, w)$  derivadas del proceso de entrenamiento difieren entre sí cuando los conjuntos de entrenamiento empleados son distintos.

$$Varianza = E_D[\{y(x) - E_D\{y(x)\}\}]^2$$

El error derivado de la varianza es característico de modelos con una gran capacidad de adaptación a los datos. Potencialmente, estos modelos son capaces de definir diferentes funciones que ajustan los datos en  $D$ , capturando incluso el ruido presente en este conjunto en forma de anomalías. Se produce, por tanto, el sobreajuste de los datos de entrenamiento. Un mayor número de ejemplos en  $D$  contribuiría a limitar el espacio de funciones que el modelo sería capaz de definir para el ajuste de  $D$  y, por tanto, se reduciría su varianza.

### Estimación basada en máxima verosimilitud e inferencia bayesiana

Tras el análisis de las fuentes de error en un modelo de datos, retomamos el proceso de entrenamiento. Como se ha indicado, en el mismo se tiene como objetivo modelar el generador de los datos en  $D$ , que queda completamente descrito por la función  $p(x, t)$ .

A fin de estimar el valor de  $t$  para un  $x$  dado, la función e interés será  $p(t|x)$ . Para incluir explícitamente la dependencia del modelo en la estimación, puede expresarse

$p(t|x, w)$ , donde  $w$  refleja el conjunto de parámetros del modelo ajustables durante el proceso de entrenamiento. Estadísticamente, esto puede obedecer a dos principios: regla de máxima verosimilitud o inferencia bayesiana.

El principio de máxima verosimilitud ajusta los pesos  $w$  de tal forma que, dado estos, se maximice la probabilidad de observar los datos en  $D$ .

Si asumimos independencia entre las muestras en  $D$ , la probabilidad de observar este conjunto de datos según el modelo definido por  $w$  puede expresarse de la siguiente forma:

$$p(D|w) = \prod_{i=1}^M p(x_i, t_i|w) = \prod_{i=1}^M p(t_i|x_i, w) \equiv L_D(w)$$

Donde el término  $L(w)$  se identifica como la probabilidad asociada al conjunto  $D$  y depende del modelo  $w$ .

Los pesos de  $w$  se escogen de forma que se maximice esta función, es decir, se elige el modelo que hace más probable dar lugar al conjunto de datos. En la práctica, se lleva a cabo la operación equivalente que consiste en minimizar el logaritmo negativo de  $L_D$ . La **función de error** resultante se obtiene como:

$$E_D = -\log L_D = -\sum_{i=1}^M \log[p(t_i|x_i, w)] - \sum_{i=1}^M \log[p(x_i)]$$

Dado que el segundo término de  $E_D$  no depende de  $w$ , la función de error resultante viene dada por el primer sumando de la ecuación. En el caso de un problema de clasificación con dos categorías diferentes  $\{C1, C2\}$ , de forma que  $t = 1$  para C1 y  $t = 0$  para C2, la **función aproximada por la salida del modelo** puede escribirse de la siguiente forma:

$$p(t|x, w) = y^t(x, w)[1 - y(x, w)]^{1-t}$$

Sustituyendo esta expresión en la ecuación del error de acuerdo al principio de máxima verosimilitud, obtenemos la siguiente ecuación que se conoce como **función de entropía cruzada**:

$$E_D = \sum_{i=1}^M \{t_i \log(y_i) + (1 - t_i) \log(1 - y_i)\}$$

Sin embargo, el modelo que asegura una mayor probabilidad de observar  $D$  mediante la minimización del error anterior, no asegura la mejor capacidad de generalización. Este hecho se ha visto previamente con el problema de sobreajuste, común en modelos con un alto grado de varianza. La aproximación bayesiana para el ajuste de  $w$  considera este conjunto de parámetros como una variable aleatoria, de forma que esté caracterizada por una función de densidad de probabilidad  $p(w)$ . Una vez que el conjunto  $D$  es observado, la probabilidad asociada a  $w$  cambia y puede ser obtenida a través del **teorema de Bayes**:

$$p(w|D) = \frac{p(D|w)p(w)}{p(x)}$$

La estimación de  $p(w|D)$  permite realizar predicciones teniendo en cuenta todos los posibles valores de la variable  $w$ , es decir, todos los modelos potencialmente obtenibles a partir de  $D$  tras la observación de este conjunto. Así, el **valor de salida del modelo** vendría dado por la siguiente expresión:

$$p(t|x, D) = \int p(t|x, w) p(w|D) dw$$

Donde  $p(t|x, w)$  sería el valor de salida de un modelo  $w$  concreto para la entrada  $x$ . Por tanto, el valor final de salida es un promediado de las salidas de todos los modelos  $w$  observables, ponderados por la probabilidad de cada uno de ellos tras la observación de  $D$ , que viene dada por el término  $p(w|D)$ .

## 14.3. Aplicación de técnicas *machine learning* al procesado de señales

**E**l punto anterior recoge los principios elementales en la implementación de un sistema de decisión o reconocimiento de patrones. Estos principios se han presentado en un marco estadístico que permite la utilización de herramientas matemáticas para la obtención de una solución al problema. A continuación, se indica cómo las técnicas derivadas de este marco pueden aplicarse a señales para la implementación de señales de decisión.

Comenzaremos con los problemas más habituales en el ámbito del tratamiento de señales que precisan de la implementación de un sistema de decisión.

### Clasificación de señales

Una de las aplicaciones más comunes de los sistemas de decisión se encuentra en la identificación automática de la categoría de una señal, tanto si pensamos en una función unidimensional de la variable tiempo como en una imagen. En este escenario, puede que nos encontremos en una de las siguientes situaciones:

- ▶ El problema de clasificación no es sencillo de resolver por el ser humano mediante la observación de las señales. En este caso, precisamos de herramientas avanzadas que capturen información de la señal que nosotros no somos capaces de extraer.
- ▶ El ser humano es capaz de llevar a cabo la clasificación que se persigue, pero se trata de una tarea que precisa de elevados recursos (materiales, tiempo, profesionales involucrados...). Por tanto, en esta situación el objetivo es **automatizar el procedimiento**.

El problema de clasificación de señales debe acotarse a un escenario concreto, previamente definido, a fin de obtener un sistema de decisión efectivo. Por ejemplo, en el caso de una imagen, el propósito de la tarea de clasificación se ceñiría a identificar si la figura que aparece en ella se corresponde o no con una persona. La interpretación automática de qué ser vivo o cosa es dicha figura, de entre todas las posibilidades existentes, es un problema excesivamente abierto y complejo de abordar. Otro caso típico en sistemas de ayuda a la decisión basado en imágenes se tiene en un contexto médico. Así, podemos pensar en la implementación de un clasificador que permita identificar automáticamente tejido pulmonar dañado frente a tejido sano sin lesiones relevantes.

De forma similar, en el caso de las **señales en el dominio del tiempo**, es habitual enfrentar problemas en los que el propósito es separar registros patológicos de aquellos correspondientes a un paciente sano. Suele ser común el procesado de señales como el electrocardiograma (ECG), el electroencefalograma (EEG) o la saturación de oxígeno (SaO).

Por último, las **herramientas de reconocimiento de voz** implementan también clasificadores de señales. Una herramienta relativamente sencilla sería un sistema capaz de distinguir si la voz percibida se corresponde con un hombre o una mujer. Las máquinas para la interpretación automática del habla poseen mayor complejidad. El sistema dispone de un léxico o conjunto de palabras con las que ha sido entrenado y, a partir de la señal de audio capturada, se busca cada una de las palabras en este léxico para poder interpretar el mensaje completo.

## Segmentación

En el contexto del procesado de imágenes, los sistemas de decisión pueden emplearse para la identificación de diferentes regiones y el cálculo de las fronteras entre estas. Para ellos, es común emplear filtros espaciales que, centrados en un píxel y con un tamaño determinado, clasifican el píxel central asignándole una de las posibles categorías. Si estas han sido previamente definidas, estamos ante un

**problema supervisado**, por lo que el clasificador será entrenado a partir de un conjunto de ejemplos previamente etiquetados, tal y como hemos supuesto desde el inicio en este tema.

El **problema no supervisado** en el que no disponemos de un conjunto de ejemplos y hemos de inferir las diferentes categorías existentes, requiere de la utilización de técnicas de *clustering*. La figura 5 muestra un ejemplo de un problema de clasificación de regiones en una imagen.



Figura 5. Segmentación de regiones en una imagen.

Fuente: <http://web.eecs.umich.edu/~silvio/teaching/lectures/Vision%20Research%20Lab%20-%20A%20Multiresolution%20Approach%20to%20Image%20Segmentation%20Based%20on%20EdgeFlow.htm>

## Predictión

El trabajo con señales unidimensionales que son función del tiempo presenta la necesidad de estimar valores futuros de la señal a partir de muestras históricas. Como un ejemplo claro puede pensarse en cotizaciones bursátiles. El sistema de decisión consistiría en un clasificador capaz de capturar si la cotización del día siguiente será mayor o menor respecto a la del día anterior, basándose en muestras del histórico de la serie.

En los diferentes escenarios descritos, la **metodología para la implementación** del clasificador es común, tal y como se describía en el comienzo de este tema:

1. Identificación del objeto a clasificar: píxel de una imagen, región de una imagen, segmento de señal, etc.
2. Preprocesado de la fuente de información: eliminación de *outliers* y ruido, detección de bordes, selección de altas frecuencias, etc.
3. Caracterización del objeto: definición del conjunto de atributos que describen el objeto sobre el que se va a decidir; este proceso se trata de la construcción del vector  $x$  de características a partir de las técnicas descritas en temas anteriores.
4. Definición del problema a modelar: definición de la variable objetivo  $t$ : clasificación de una señal de ECG como patológica o no; identificación de una palabra en un diccionario; figura de una persona o no, etc.
5. Construcción de un conjunto de ejemplos: conjunto de entrenamiento  $D$  formado por pares  $(x_i, t_i)$ .
6. Entrenamiento y test: implementación del sistema de decisión y evaluación de su rendimiento sobre un conjunto de muestras no empleadas para el ajuste del modelo.

Percepción Computacional

---

# Aplicaciones actuales del tratamiento de la señal

# Ideas clave

## 15.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer con atención las ideas clave que se desarrollan a continuación.

Como último tema de la asignatura, la idea es aplicar los conocimientos presentados a dos tecnologías actuales y entender qué algoritmos y soluciones son los más adecuados para cada problema. Además, se presenta un conjunto de retos y limitaciones que son el trabajo actual y futuro de las investigaciones asociadas a percepción computacional.

## 15.2. *Biometrics*

**L**a biometría es el conjunto de técnicas de reconocimiento de patrones capaces de verificar e identificar a una persona de forma unívoca. Actualmente, la seguridad y protección de la información reside en tres pilares:

- ▶ «Lo que sabes», también conocido como **contraseñas** en todas sus modalidades: usuario y contraseña corriente, sistemas de clave pública y privada, etc.
- ▶ «Lo que tienes», también conocido como **token**, consistente en identificar o más bien validar a un usuario por tener un determinado objeto/texto con información que únicamente el usuario posee. Este *token* es comúnmente usado en los bancos de dos maneras:
  - Mediante el envío de un mensaje al teléfono móvil, siendo en este caso el móvil el propio *token*.

- Tarjeta de coordenadas, donde el banco te indica un número de dos cifras y el usuario debe completar con 3-4 cifras las correspondientes a dichas dos cifras.
- ▶ «Lo que eres», la **biometría**: rasgos únicos extraídos de propiedades físicas del individuo como puede ser el iris, la retina o la propia huella digital.

La seguridad de la biometría reside en «lo que eres»: en características físicas y de comportamiento únicas para cada usuario o que, sin ser tan únicas, permiten al menos poder identificar a los usuarios lo suficiente como para distinguirlos del resto de usuarios. Es importante distinguir dos usos dentro de estos sistemas biométricos:

**Verificación.** En la verificación, el **usuario proporciona algún tipo de identificación** y el sistema biométrico comprueba que realmente es quien dice ser. Para ello, el sistema biométrico compara los rasgos extraídos con el patrón (también llamado *template*) biométrico de dicho usuario. Si dicha comparación no supera un umbral, entonces el usuario es rechazado. Por el contrario, si dicha comparación es satisfactoria, el sistema verificaría al individuo positivamente.

**Identificación.** Al contrario que en la verificación, el usuario no proporciona **ningún tipo de identificación previa**, luego el sistema tiene que reconocer/identificar de qué usuario se trata. Esto lo realiza comparando los rasgos biométricos adquiridos en el sistema de captura con todos y cada uno de los patrones (*templates*) almacenados en la base de datos.

En cualquier caso, parece obvio que el usuario ha de estar primeramente registrado en el sistema. Este proceso se conoce como *enrollment*. Y para ello se realizan las siguientes operaciones:

- ▶ Captura de la información biométrica mediante sensores especializados: cámaras de infrarrojo, sensores de huella, etc.
- ▶ Extracción de características para la creación de un patrón.
- ▶ Almacenamiento en la base de datos.

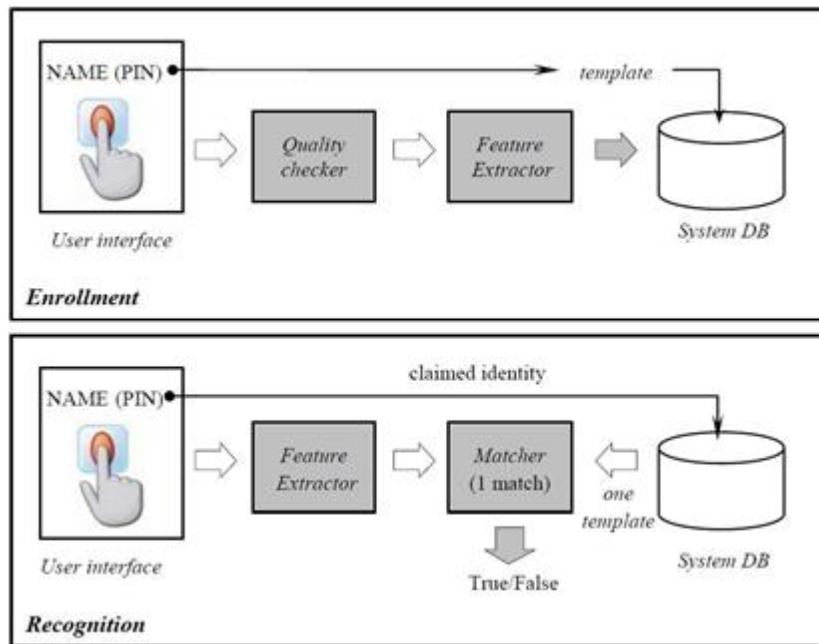


Figura 1. Funcionamiento de un sistema biométrico en sus dos fases más típicas.

Fuente: [http://biometrics.cse.msu.edu/pub/secure\\_biometrics.html](http://biometrics.cse.msu.edu/pub/secure_biometrics.html)

En la imagen anterior vemos esquemáticamente cómo son las fases de un sistema biométrico: dar de alta un usuario (*enrollment*) y comprobación del usuario (*recognition*); en este caso concreto, la comprobación se trata de una verificación.

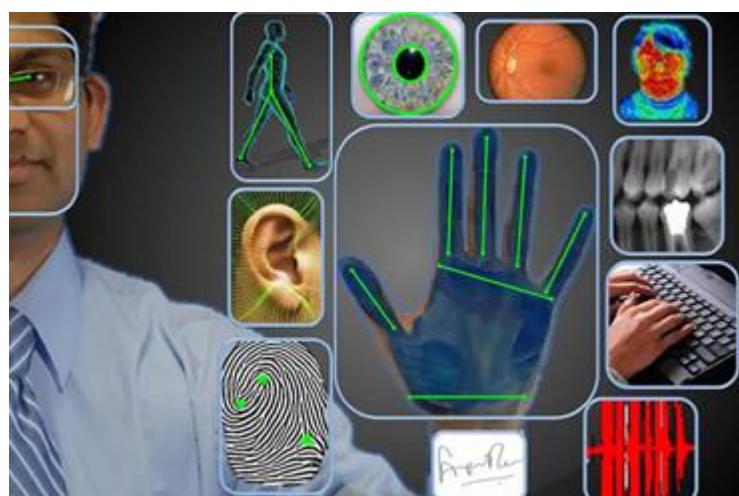


Figura 2. Rasgos físicos más comunes empleados en el reconocimiento biométrico.

Fuente: <https://www.vrstech.com/biometric-systems-security-services.html>

A continuación, presentamos un resumen de las técnicas más comúnmente empleadas en biometría: reconocimiento facial y biometría de iris, junto con los algoritmos tanto de segmentación como de extracción de características más extendidos en la literatura para cada técnica.

## Reconocimiento facial

La cara, desde un punto de vista geométrico, ofrece características muy identificativas: la distancia entre ojos, proporción de la nariz, etc.

Para la **segmentación**, un detector de color de piel (segmentación basada en color) suele ser suficiente, aunque existen enfoques donde se persigue la segmentación basada con infrarrojos o bien con sensores de temperatura, mucho más avanzados y costos económicamente.

Para la **extracción de características**, el concepto de *eigenfaces* es el más extendido. Este método emplea el uso de *Principal Component Analysis* para encontrar las componentes principales en una imagen; realiza, además, una descomposición en vectores propios (de ahí el nombre de *eigenfaces*) que proporciona un resultado muy preciso con una computación razonable.

Las ventajas de la biometría basada en reconocimiento facial se basan en su poca invasividad: el usuario no necesita colaborar sustancialmente para que el proceso de verificación/identificación se lleve a cabo. Las dificultades que presenta el reconocimiento facial son su dependencia de agentes externos como la iluminación o el uso de gafas, barba o incluso heridas que puedan modificar los rasgos biométricos.

En la siguiente imagen vemos que, a partir de diferentes caras del usuario, se obtienen sus rasgos más significativos.



Figura 3. Ejemplo visual de *eigenfaces*.

Fuente: <https://www.emaze.com/@AQCLWRRF/Facial-Recognition>

## Reconocimiento de iris

Uno de los más potentes a nivel de precisión por la gran entropía y unicidad que ofrece el iris del ojo humano. Cada ojo, incluso entre izquierdo y derecho, es único a nivel de textura.

La **segmentación** está basada en tres principios:

- ▶ Segmentación basada en **color**: aunque de las más sencillas, es la menos precisa de todas debido a su gran variabilidad de iluminación.
- ▶ Segmentación basada en **forma**: asumiendo que el iris es circular, intentar detectar zonas circulares dentro de la imagen del ojo. Tiene mucho sentido emplear algoritmos de crecimiento de regiones en este tipo de segmentaciones.
- ▶ Segmentación basada en **textura**: empleada en la mayoría de casos debido a que las capturas se suelen realizar con sensores infrarrojos.

La **extracción de características** viene dada por filtros de Gabor, aunque existen infinitas posibilidades, todos ellos basados en texturas: *Markov Random Fields*, Wavelets, DCT, etc.

La comparación de unas características biométricas basadas en iris con su correspondiente patrón biométrico suele ser muy sencillo computacionalmente (distancia euclídea, distancia de Hamming, etc.). De esta manera, la comparación es muy rápida y permite poder tomar decisiones rápidamente.

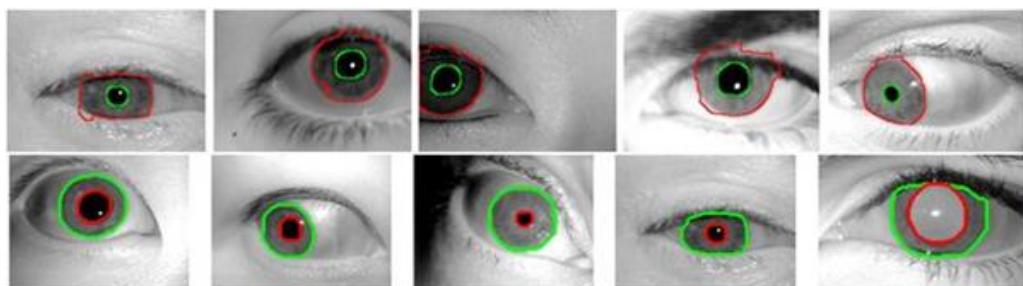


Figura 4. Ejemplos de segmentación de iris y pupila.

Fuente: <https://www.andrew.cmu.edu/user/thihoanl/Research.html>

Cuando se emplean cámaras infrarrojas, la segmentación basada en color no produce buenos resultados y hay que acudir a segmentaciones más avanzadas basadas en formas o en texturas.

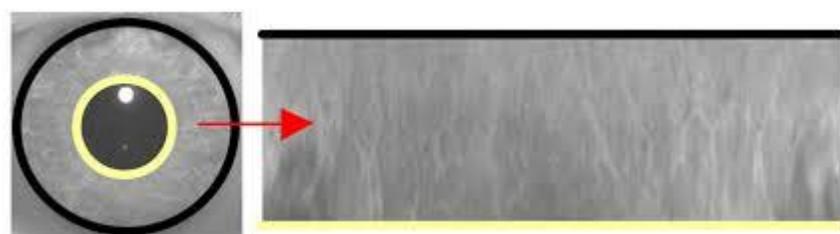


Figura 5. Transformación de coordenadas polares a cartesianas del tejido correspondiente al iris.

Fuente: <http://www.cs.princeton.edu/~andyz/irisrecognition>

Esta transformación facilita la posterior aplicación de filtros y comparación de patrones.

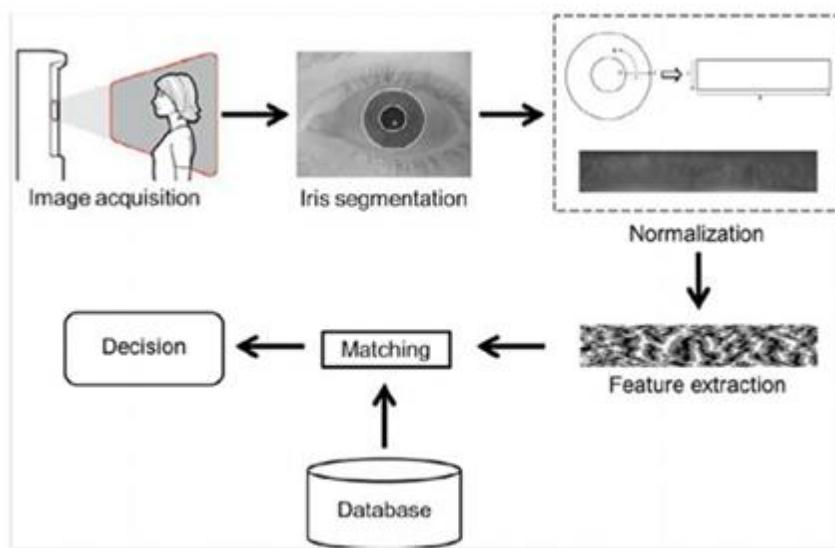


Figura 6. Esquema general de un sistema de reconocimiento biométrico basado en iris.

Fuente: <https://www.bayometric.com/biometric-iris-recognition-application/>

Y por último, se presenta un resumen breve de otras técnicas biométricas con los algoritmos de segmentación y extracción de características más comunes en la literatura.

### **Huella digital**

Sin duda, una de las técnicas biométricas más empleadas, y no por ella la más sencilla.

- ▶ Sistemas de captura: suelen ser sistemas de infrarrojos o escáneres.
- ▶ Segmentación: muy sencilla, ya que se delega en el sensor toda la dificultad de la captura de la huella. Suele tratarse de una segmentación basada en clustering de dos clases junto con morfología matemática para eliminar errores.
- ▶ Extracción de características: propiedades geométricas de los patrones de las líneas de la huella. Dichas propiedades son muy similares a los algoritmos SIFT.

### ***Palmpint***

Esta técnica consiste en leer las líneas de la palma de la mano, no solo las más visibles, también la textura; similar a la huella del dedo, pero en la palma de la mano.

- ▶ Sistemas de captura: muy variados, aunque los más usados son los que emplean fotografías infrarrojas.
- ▶ Segmentación: más compleja que en la huella; en este caso hay que identificar correctamente la palma de la mano y hacer una segmentación basada en intensidad.
- ▶ Extracción de características muy similares a la de la huella digital.

### Geometría de mano

Esta técnica permite identificar a las personas en función de la forma de su mano.

- ▶ Sistemas de captura: muy sencillos, bastaría con una cámara del móvil.
- ▶ Segmentación: cuanto más sencillo sea el dispositivo de captura, más compleja será la segmentación. En este caso, la segmentación suele conllevar agregación multiescala, segmentación basada en color o crecimiento de regiones.
- ▶ La extracción de características está orientada a medidas relativas de los dedos, mano y muñeca.

### Venas de la mano y retina

- ▶ Sistema de captura: infrarrojo obligatoriamente para poder ver a través de la piel.
- ▶ Segmentación: basada en color y en intensidades.
- ▶ Extracción de características: suele usarse filtros de Gabor y algoritmos que simulan redes comparando las venas a redes de nodos.

### Firma

- ▶ Sistema de captura muy especializado: o bien un sensor con un bolígrafo específico (*online*), o bien una fotografía de una firma (*offline*).
- ▶ En el caso *online* no hay segmentación. En el *offline*, la segmentación es muy sencilla: binarización de la imagen.

- Extracción de características: se suelen utilizar métodos basados en parametrización de curvas y propiedades parecidas a SIFT. No obstante, la comparación en esta técnica requiere especial atención. Algoritmos basados en programación dinámica como Dynamic Time Warping suelen ofrecer resultados excelentes ya que trabajan muy bien con señales que pueden estirarse en el tiempo y en el espacio.

En definitiva, la biometría es uno de los campos, junto con las imágenes biomédicas, donde más se ha avanzado en la percepción computacional y donde más se está mejorando las capacidades humanas de reconocimiento.

### 15.3. *Self-driving car*

**R**ecientemente están comenzando a surgir diferentes empresas capaces de crear coches que conducen por sí mismos. Muchas de estas marcas afirman que los coches son capaces de detectar más información (y procesarla) de la que un humano es capaz de detectar y procesar.

En este capítulo vamos a tomar el esquema inicial presentado en el tema de *Elementos de un sistema de percepción* para modelar la percepción computacional: captura de información, procesamiento y toma de decisiones.

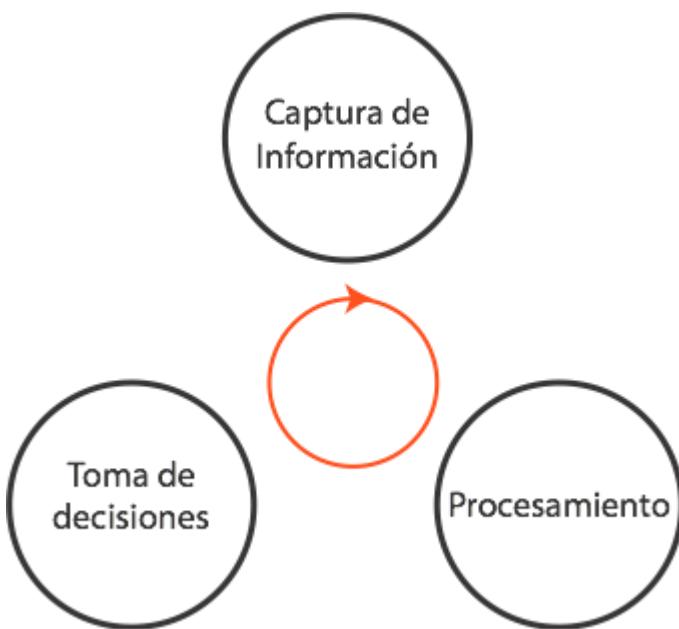


Figura 7. Esquema básico de percepción computacional.

Con respecto a la captura de información, existe una gran diversidad de sensores que en efecto superan a los humanos: infrarrojos, radares, sensores de aceleración, de movimiento, de humedad, posicionamiento, etc. Hoy en día no se contempla, pero posiblemente, en un futuro, la existencia de drones aéreos vigilando y proporcionando una vista aérea permita tener un control del tráfico enorme y poder proporcionar a los *self-driving car* la posibilidad de entender todo el contexto de la situación.

A continuación, se presenta un breve esquema de todas las posibilidades sensoriales que ofrece un coche autodirigido.

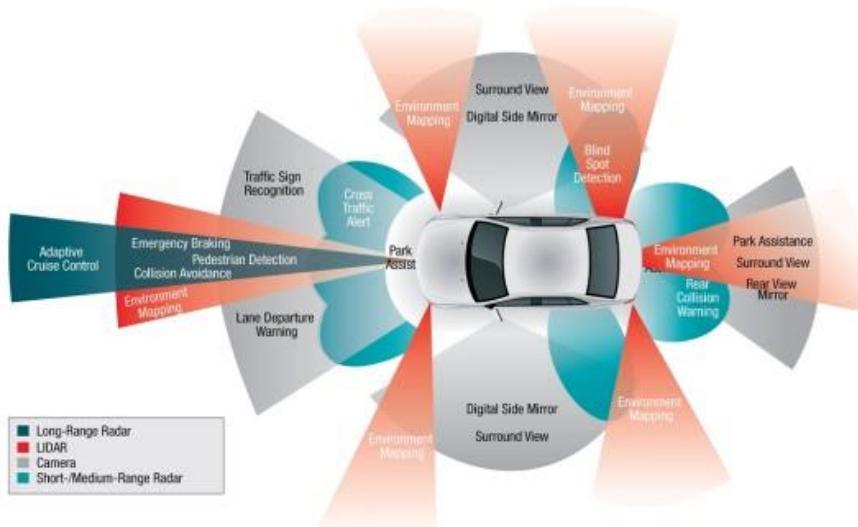


Figura 8. Descripción de sensores incluido en un *self-driving car*.

Fuente: <https://www.shofior.com/wordpress/self-driving-car-how-it-works/?lang=en>

Es obvio que para capturar toda esta información, sincronizarla, preprocesarla y limpiarla, entre otras tareas, hace falta un computador a bordo. De hecho, dicho computador está especialmente dedicado a estas tareas, considerando que el procesamiento de vídeo es una de las tareas computacionales más costosas. Además, en muchos casos los sensores son capaces de construir representaciones en 3D de la realidad, haciendo que los algoritmos de segmentación, como puede ser el algoritmo de morfología matemática del que hemos hablado en esta asignatura, aumenten sustancialmente en coste computacional, ya que el procesamiento ha de hacerse en el momento, en tiempo real.

De hecho, la computadora encargada del procesamiento puede ocupar el maletero del propio coche. Sin embargo, el reto está precisamente en el procesamiento de la información capturada por múltiples motivos.

- ▶ En primer lugar, se está capturando información de diversa naturaleza:
  - Información **unidimensional** como puede ser la información proveniente de los sensores.
  - Información **bidimensional** como las propias imágenes o vídeos que se capturan para analizar el movimiento.

- Información **tridimensional** como son los radares que dotan al coche de la posibilidad de entender la profundidad en el espacio.
- ▶ En segundo lugar, la información ha de procesarse a la misma velocidad: no puede procesarse primero la sensorización de humedad (por poner un ejemplo) y posteriormente el vídeo procedente de la cámara. Un desalineamiento temporal en este caso produciría que la decisión final (de la cual hablaremos más adelante) sea errónea y se esté tomando con base en datos completamente incorrectos.
- Además, dicho procesamiento ha de contemplar la posibilidad de que no pueda devolver ningún resultado satisfactorio, con lo que precisa de una lógica necesaria para actuar en estos casos: una mala segmentación, cambio drástico de luz, etc.
- Por último, debe almacenar todos los datos procesados, ya que moralmente y en caso de accidente se deberá analizar qué sucedió, qué vio el coche y por qué actuó de esa manera.



Figura 9. Hardware necesario para procesar toda la información capturada por un *self-driving car*.

Fuente: <https://www.nytimes.com/es/2016/02/10/los-autos-que-se-manejan-solos-todavia-necesitan-a-los-humanos/>

## ¿Qué algoritmos podría aplicar un *self-driving car*?

Algoritmos de **segmentación** con la finalidad de:

- ▶ Separar el fondo del vídeo que se captura eliminando, por ejemplo, las zonas de césped, o detectando qué zona se corresponde con un cambio de rasante.
- ▶ Detección de diferentes autos dentro de la vía.

Algoritmos de **detección** de bordes con la finalidad de:

- ▶ Detectar correctamente las fronteras de la vía.
- ▶ Detectar la existencia de líneas en la carretera, como puede ser la línea continua.

Algoritmos de **extracción de características** con la finalidad de:

- ▶ Detectar objetos en movimiento; mediante Wavelets y filtros de Gabor se pueden detectar texturas diferentes a las del asfalto o cambios de texturas en el paisaje.
- ▶ Detección de señales; mediante el algoritmo SIFT se puede identificar señales de tráfico fácilmente.

Algoritmos de **corrección de ruido**: un ejemplo claro son las señales de tráfico. En caso de lluvia o condiciones adversas, puede que el sensor de señales confunda la señal de prohibido; ir a más de 60 km/h con la de ir a 80 km/h, con las correspondientes consecuencias.

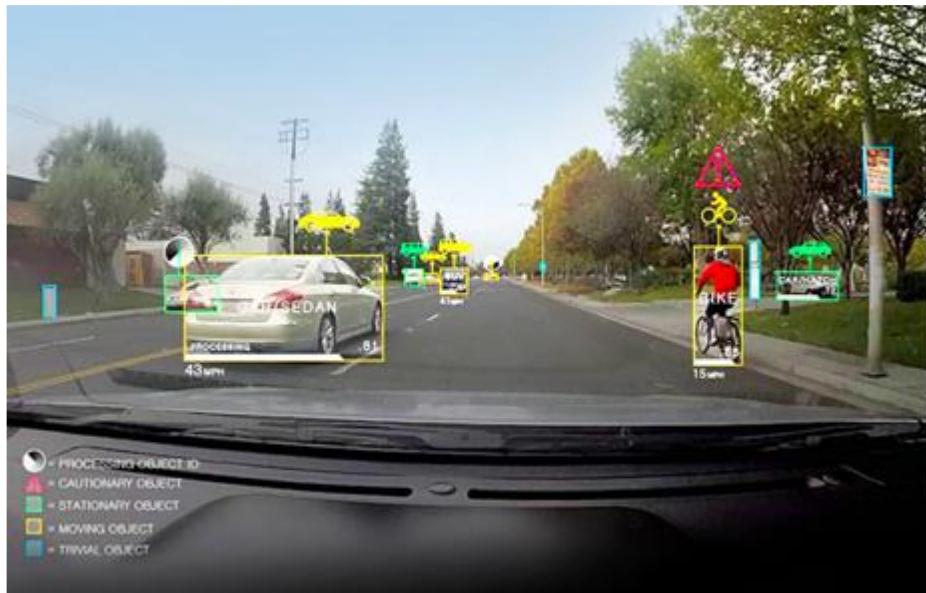


Figura 10. Detección de objetos en un *self-driving car*.

Fuente: <http://www.eenewsautomotive.com/news/nvidia-computer-processing-hub-self-driving-cars>

La identificación de diferentes objetos, así como el reconocimiento y clasificación de los mismos favorece la toma de decisiones.



Figura 11. Segmentación de imágenes aplicada a un *self-driving car*.

Fuente: <https://blogs.nvidia.com/blog/2016/01/05/eyes-on-the-road-how-autonomous-cars-understand-what-theyre-seeing/>

En diferentes colores se aprecian los diferentes tipos de objetos en la imagen: asfalto (morado), personas (verde), señales de tráfico (amarillo) y vehículos (rojo).

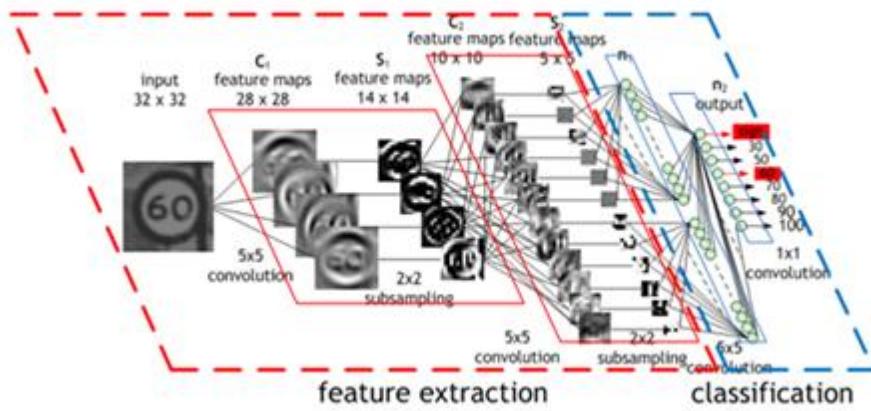


Figura 12. Ejemplo de red neuronal para la extracción de características y clasificación de una señal de tráfico.

Fuente: <https://towardsdatascience.com/beginning-my-journey-in-self-driving-car-udacity-nano-degree-a39d898658a2>

En esta imagen se aprecia cómo, tras la clasificación, la imagen se detecta correctamente como una señal de limitación de 60 km/h.

Y por último, el *self-driving car* consta del **módulo de toma de decisiones**. En este caso, la ética aún no se ha pronunciado sobre qué decisiones debe tomar un *self-driving car*. Por ahora, lo único que hace en una situación anómala es avisar al conductor para que tome el control del vehículo.

Pero en el caso real de un conductor distraído o de un coche completamente autónomo, ¿qué decisiones podrá tomar un vehículo? ¿Quién sería el responsable si el vehículo toma una decisión incorrecta? ¿Y si, en el caso extremo, tiene que elegir entre salvar la vida de sus ocupantes o la de los viandantes? Y en un caso más liviano, una multa de tráfico, ¿de quién sería la responsabilidad?

Como todo módulo de percepción computacional proporciona un soporte a la decisión del humano que, en función de los datos proporcionados por el sistema, decidirá qué opción es la más adecuada.

## 15.4. Retos y limitaciones de la percepción computacional

**E**n la percepción computacional existe aún mucho espacio de mejora. Los algoritmos que hemos visto, por su gran generalidad, requieren de una **especificación muy alta** que hace que pierdan esa capacidad de generalidad.

De hecho, algoritmos que detecten una mano en una imagen, por ejemplo, no podrán usarse si se cambia el sistema o las condiciones de captura. En la mayoría de los casos, esto impide que los sistemas de percepción computacional sean generales.

La incorporación de nuevas técnicas de aprendizaje como *Deep Learning* está haciendo que esa meta esté más cerca, pero aún queda mucho camino por recorrer.

Nos gustaría poner un ejemplo llamativo de hasta qué punto se está exigiendo a la inteligencia artificial el ser capaz de tomar decisiones o de decidir una clasificación para una imagen.

Existen competiciones donde se ponen a prueba los algoritmos de clasificación de imágenes. Uno de ellos es la competición *Muffin or Chihuahua*. El parecido entre imágenes de ambas clases es comprensible y confunde enormemente al clasificador y al sistema de percepción computacional que hay detrás.



Figura 13. Competición *Muffin or Chihuahua*.

Fuente: <https://medium.freecodecamp.org/chihuahua-or-muffin-my-search-for-the-best-computer-vision-api-cbda4d6b425d>

No obstante, ¿qué se puede hacer para mejorarlo?

- ▶ Incluir información relevante a la textura con la intención de distinguir el *muffin* del pelo del perro.
- ▶ No usar algoritmos como SIFT ya que podrían confundir la mayoría de estas imágenes.
- ▶ Aplicar algoritmos basados en Wavelets para poder potenciar los detalles.
- ▶ Aplicar características que modele la simetría de un perro en comparación a la asimetría (o simetría casual) de un *muffin*.

En cualquier caso, el sistema de percepción resultante sería muy específico y no podría usarse para identificar ningún otro tipo de objeto o hacer ningún otro tipo de comparación.

Por último, indicamos una lista de los retos a los que se enfrentan actualmente estos sistemas de percepción computacional. Esta lista es muy dinámica y seguramente, en la fecha en la que escribimos estas limitaciones, ya han surgido enfoques o soluciones que afrontan dichas limitaciones porque un aspecto positivo de la percepción computacional es su amplia comunidad y alto interés.

**Sesgos.** Uno de los aspectos más inquietantes es el propio sesgo humano: ¿hasta qué punto puede ser que un algoritmo aprenda o imite exactamente cómo decide un humano y no desarrolle toda la potencialidad?

**Ética.** ¿Se puede confiar en lo que una máquina decide por sí misma? ¿Se puede confiar la vida de personas a un sistema de conducción automática? ¿Y a una vigilancia automática? ¿Podremos vivir con sistemas de percepción integrados en nuestros sistemas de visión humanos?

**Falta de generalidad.** ¿Se podrá crear un algoritmo capaz de aprender y entender qué hay en la imagen sin necesidad de que dentro lleve múltiples clasificadores? ¿Podrá la percepción computacional deducir nuevos conceptos de una imagen así como lo hace un humano?

**Arte.** Puesto que estos sistemas pueden percibir imágenes, ¿podrán crear las suyas propias? Y en ese caso, ¿serían completamente originales?