

Eliminación de ruido en imágenes con wavelets.

Análisis de señales

Grupo E: Alejandra Venegas, Rebeca Company, Marta Medina, Alejandro Cornelio y Ilia Zhigarev.

2025-01-07

Índice

1	Introducción	1
2	Fundamento teórico: eliminación de ruido con wavelets	2
3	Funciones de programacion empleadas	2
4	Desarrollo y resultados	4
4.1	Inclusión de ruido sintético en las imágenes	5
4.2	Función imwd	5
4.2.1	Redimensionando las imágenes	5
4.2.2	Ampliando las imágenes	9
4.3	Función denoise.dwt.2d	11
4.4	Transformada de Fourier	16
5	Conclusiones	17

1 Introducción

En el campo del análisis de señales, un problema importante es la eliminación de ruido en imágenes digitales. Las señales a menudo están contaminadas por ruido debido a interferencias en los procesos de captura o transmisión. Este ruido, que puede existir de formas muy diversas como ruido gaussiano, artefactos asociados a frecuencias altas o bajas, o efectos multiplicativos, complica la extracción de características significativas.

El uso de wavelets es una herramienta clave para abordar este problema. Este método ofrece la posibilidad de adaptarse a las características particulares de cada tipo de ruido para su eliminación.

En este trabajo se exploraron la capacidad de los wavelets para la eliminación de ruido en imágenes. Para ello, se generarán diferentes tipos de ruido sintético y determinando los parámetros más eficaces de los wavelets para su eliminación a través de 3 métodos distintos.

2 Fundamento teórico: eliminación de ruido con wavelets

Los wavelets son funciones matemáticas que suelen emplearse para descomponer una señal en componentes de diferentes escalas, lo que resulta útil para identificar y procesar características específicas. Este enfoque es especialmente relevante en la eliminación de ruido, donde las frecuencias indeseadas pueden ser separadas y reducidas sin afectar significativamente las características principales de la señal original. A diferencia de la transformada de Fourier, que opera globalmente y no ofrece información sobre el espacio, los wavelets permiten un análisis localizado, facilitando la identificación de patrones y anomalías en los datos.

El proceso de reducción de ruido con wavelets generalmente incluye tres etapas principales: la descomposición de la señal utilizando una wavelet madre, la modificación de los coeficientes wavelet mediante técnicas de umbral, y la reconstrucción de la señal. La selección de la wavelet madre adecuada y los parámetros de umbral son aspectos críticos que deben adaptarse a las características específicas del ruido y la señal.

Los wavelets resultan especialmente útiles en aplicaciones donde la reconstrucción fiel de la imagen si ruido es esencial, como en imágenes médicas, procesamiento de audio o análisis de datos científicos.

3 Funciones de programación empleadas

Se ha escogido emplear tres metodologías distintas para realizar la eliminación de ruido. Dos de ellas se basan en emplear la transformada wavelet. Sin embargo, también se ha optado por emplear en un tercer método el análisis mediante transformadas de Fourier con el objetivo de encontrar similitudes y diferencias con respecto a los métodos que emplean wavelets.

Con respecto a los métodos con wavelets utilizaremos un método muy directo, empleando la función `denoise.dwt.2d` del paquete `waveslim` de R. Por otro lado, emplearemos un método más manual que consiste en realizar la transformada wavelet discreta con la función `imwd`, aplicar una función para realizar el thresholding y a continuación realizar la transformada inversa.

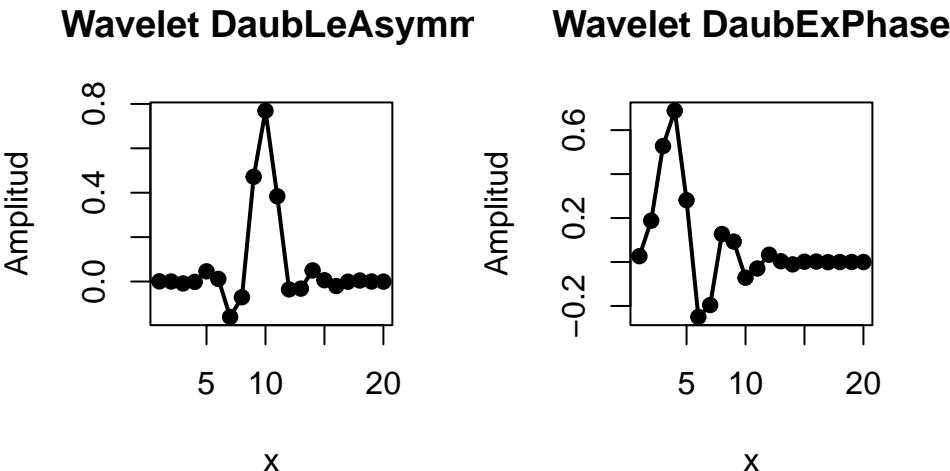
Para finalizar, se realizará la eliminación de ruido a través de transformadas de Fourier con la función `fft` y compararemos los resultados obtenidos con cada método.

Método 1. Eliminación de ruido con el algoritmo de Mallat

Empleamos la función `imwd()` del paquete `wavethresh`. Esta función realiza una transformada discreta wavelet de acuerdo con el algoritmo de Mallat.

- `imwd(image, filter.number=10, family="DaubLeAsymm", type="wavelet",...).`

El argumento `image` de la función debe ser una matriz cuadrada cuya dimensión sea potencia de dos. `filter.number` elige la suavidad de la wavelet a emplear, siendo por defecto 10. `family` indica la familia de wavelets a emplear ("DaubExPhase" ó "DaubLeAsymm"). Para tratar las fronteras, mantendremos el parámetro `bc = "periodic"` por defecto.



Para la eliminación de ruido aplicaremos la función `threshold()` al objeto que devuelve la función `imwd()`.

- `threshold(imwd, levels = 3:(nlevelsWT(imwd) - 1), type = "hard", policy = "universal", by.level = FALSE, value = 0, return.threshold = FALSE, compression = TRUE, Q = 0.05, ...)`

`levels` es el número de niveles a los cuáles deseamos aplicar un umbral, mientras que `type` indica si queremos un umbral más suave (“soft”) ó más fuerte (“hard”). El parámetro `policy` selecciona la técnica para elegir umbral. `by.level = FALSE` significa que se aplica un umbral global a todos los niveles indicados, mientras que `by.level = TRUE` calcular threshold para cada nivel por separado. El parámetro `value`, el valor del umbral, se usará si elegimos técnicas manuales en `policy`. Si queremos obtener el valor umbral aplicado, pondremos `return.threshold = TRUE`. Finalmente, dejaremos el parámetro `compression = TRUE` por defecto, para obtener un objeto más pequeño.

Durante el desarrollo del trabajo variaremos estos parámetros para observar su efecto en la eliminación de diversos tipos de ruido y encontrar aquellos que generen un mejor resultado en cada caso. Una vez realizado el thresholding, emplearemos la función `imwr` para realizar la transformada wavelet inversa y poder visualizar la imagen tras la eliminación de ruido.

Método 2. Función denoise.dwt.2d

Empleamos la función `denoise.dwt.2d()` del paquete `waveslim`, que emplea la transformada wavelet discreta en las dos direcciones de la imagen (DWT-2D).

- `denoise.dwt.2d(x, wf = "la8", J = 4, method = "universal", H = 0.5, noise.dir = 3, rule = "hard")`

`wf` es el filtro wavelet a emplear, pudiendo tomar diversos valores predeterminados que indican la forma de la wavelet y el número de coeficientes. `J` es el número de niveles de la descomposición, la profundidad de la misma. `method` es la técnica empleada para elegir umbral. `H` es el parámetro de Hurst, una medida de autocorrelación a largo plazo de los datos. `noise.dir` define el número de direcciones espaciales en las que se calcula la estimación del ruido. Finalmente `rule` indica la regla a emplear para la aplicación del umbral.

Esta función descompone la imagen en coeficientes wavelet, aplicando un filtro wavelet seleccionado (`wf`). Como resultado, la imagen queda dividida en 4 subimágenes o subbandas:

- Aproximación (LL): Componentes de baja frecuencia tanto en las filas como en las columnas. Es la versión suavizada o “borrosa” de la imagen, que captura las características generales.

- Horizontal (LH): Componentes de baja frecuencia en las filas y alta frecuencia en las columnas. Contiene detalles horizontales de la imagen.
- Vertical (HL): Componentes de alta frecuencia en las filas y baja frecuencia en las columnas. Representa los detalles verticales.
- Diagonal (HH): Contiene las componentes de alta frecuencia tanto en las filas como en las columnas.

Esta descomposición en 4 cuadrantes se repite tantas veces como niveles (J) se especifiquen en la función.

El siguiente paso es calcular y aplicar el umbral para ajustar o eliminar los coeficientes de más alta frecuencia en las subbandas de detalle HL, LH y HH en cada nivel de descomposición. Cabe destacar que no se aplica el umbral a la subimagen LL, ya que contiene las componentes de baja frecuencia, que -generalmente- no están afectadas por el ruido.

Después de aplicar el umbral para suprimir el ruido, la imagen se reconstruye empleando la transformada wavelet inversa (IDWT). Como se observa, esta función es una aplicación más directa del método anterior, ya implementada en una única función.

Al igual que con el método anterior, emplearemos esta función variando diferentes parámetros e interpretando los resultados.

Método 3. Transformada de Fourier (función fftshift)

El método usado usa la transformada rápida de fourier para poder eliminar altas frecuencias correspondientes a ruido en la imagen. La función `fftshift` desplaza las bajas frecuencias al centro del espectro y las altas a los extremos para realizar aplicar `fft`. Una vez se visualiza el espectro, se procede a disminuir la magnitud de las altas frecuencias cercanas a los extremos asignándoles una magnitud menor. A continuación, se reconstruye la imagen deshaciendo el desplazamiento y realizando la transformada inversa.

Además de las funciones ya implementadas en R a lo largo del trabajo se programaron diversas funciones que ayudarán a automatizar el proceso.

4 Desarrollo y resultados

Comenzamos cargando y visualizando las fotografías a emplear.

Vamos a visualizar las imágenes empleadas. Se han tomado 5 imágenes distintas pero todas con la misma temática. La elección se ha realizado pensando en poder observar como afecta la eliminación de ruido en detalles como la textura o defectos en las frutas. Además, las imágenes 1 a 4 se han tomado con una cámara de gran calidad, mientras que la imagen 5 cuenta con una resolución mucho menor. Se busca también estudiar que efecto tiene la resolución de la imagen en la eliminación de ruido.



4.1 Inclusión de ruido sintético en las imágenes

Se crean la función `add_noise_to_image` y `NOISE_TYPES` para añadir el ruido a las imágenes. Los ruidos que se han generado, con parámetros ajustables a variar, son los siguientes:

- Ruido gaussiano de media 0 y desviación típica ajustable.
- Ruidos sinusoidales de alta y baja frecuencia.
- Ruido sal y pimienta: Este tipo de ruido sustituye píxeles aleatorios en la imagen por valores mínimos (negros) o máximos (blancos), simulando un patrón de puntos oscuros y brillantes
- Ruido gamma multiplicativo: multiplica el valor de los píxeles por una distribución gamma.
- Ruido uniforme multiplicativo: multiplica el valor de los píxeles por una distribución uniforme.

Estos ruidos sintéticos pueden imitar ruidos que se encuentren en imágenes reales. Por ejemplo, el ruido sinusoidal introduce un patrón periódico de oscilaciones que pueden simular interferencias periódicas. Otro ejemplo es el ruido sal y pimienta, que puede simular fallos en la captura de las imágenes (la aparición de puntos blancos y negros).

4.2 Función imwd

Uno de los parametros de la función `threshold` explicada anteriormente es `policy`, el cual determina el valor del umbral utilizado en el proceso de eliminación de ruido. Se exploran tres enfoques diferentes para la selección del umbral: `universal`, `fdr` y `manual`.

Umbral Universal

El umbral “universal”, fue propuesto por Donoho y Johnstone(1995). Esta estrategia calcula el umbral aplicado a los coeficientes de wavelet en función del tamaño de la señal y una estimación del nivel de ruido. Este enfoque tiene como objetivo establecer un umbral de manera que se eliminen los coeficientes de wavelet que corresponden al ruido, mientras se conservan aquellos que contienen la señal significativa. La fórmula del umbral “universal” es

$$\sigma \sqrt{2 \log nd}$$

donde σ es una estimación del ruido y nd es el número de coeficientes en la subbanda de detalles correspondiente a un nivel de la transformada wavelet. Este valor se obtiene accediendo a los coeficientes de la subbanda D de cada nivel.

Umbral FDR

La tasa de falsos positivos (FDR) es una técnica estadística utilizada para controlar la tasa de falsos positivos en el proceso de selección de coeficientes relevantes, tal como se describe en el trabajo de Abramovich y Benjamini (1996). En el contexto de la reducción de ruido mediante la Transformada Wavelet, el objetivo principal de FDR es identificar y eliminar los coeficientes asociados al ruido, mientras se preservan aquellos que contienen información significativa, como bordes, texturas o detalles importantes de la imagen. Esto se logra calculando, para cada coeficiente de la transformada, la probabilidad de que dicho coeficiente sea un falso positivo, es decir, que corresponda a ruido pero sea erróneamente considerado relevante.

Umbral Manual

Se fija un umbral manualmente.

4.2.1 Redimensionando las imágenes

Para aplicar el algoritmo de Mallat (IMWD), es necesario que la imagen tenga una forma cuadrada cuyas dimensiones sean potencia de dos. Dado que muchas imágenes no son cuadradas, es necesario convertirlas antes de aplicar el algoritmo, por lo que debemos realizar un pre-procesamiento de las imágenes. Se han

escogido 2 maneras distintas para obtener imágenes con el tamaño adecuado. Por un lado, redimensionaremos las imágenes con la función `resize`, lo que podría conllevar problemas de distorsión si las imágenes estaban lejos de tener dimensiones cuadradas. Por ello, también vamos a emplear otra técnica y rellenaremos las matrices de las imágenes con valores de 0 ó 1 hasta alcanzar las dimensiones adecuadas.

Como ya se ha visto para poder aplicar la función `imwd()` es necesario partir de una matriz cuadrada cuyas dimensiones sean potencia de dos. Por ello, en primer lugar creamos una función `resize_imwd()` tal que dada una foto busca la submatriz cuadrada y potencia de dos más grande posible y a continuación redimensiona la imagen a dicha submatriz cuadrada.

Por otro lado, también creamos una función para el post-procesamiento de las imágenes tras la eliminación de ruido. Queremos devolverlas a su tamaño original con el objetivo de comparar con las imágenes iniciales.

Comenzamos generando una función `procesar_imagen_wavelet` con parámetros `foto`, `tipo` y `policy`. Esta función realiza en primer lugar la transformada wavelet a cada uno de los tres canales de una imagen. A continuación, se realiza el thresholding con la función `threshold`, pudiendo variar de el tipo de “hard” a “soft” y el parámetro `policy` (modificando adecuadamente los parámetros necesarios en la función `threshold` en este último caso). Una vez realizada la eliminación de ruido, se aplica la transformada inversa `imwr` para por último reconstruir la imagen a partir de los tres canales.

```
procesar_imagen_wavelet <- function(foto, tipo = "hard", policy = "universal") {
  # 1. Realizamos la transformada wavelet a cada canal
  lwd <- lapply(1:3, function(canal) {
    imwd(foto[, , canal])
  })

  # 2. Aplicamos el umbral a los coeficientes de la transformada wavelet
  lwd_threshold <- lapply(lwd, function(canal_wd) {
    niveles <- canal_wd$nlevels
    wavethresh::threshold(canal_wd, levels = 3:(niveles-1), type = tipo,
                           policy = policy, by_level=TRUE, compression=FALSE)
  })
  # 3. Aplicamos la transformada wavelet inversa a cada canal umbralizado
  ilwd <- lapply(lwd_threshold, function(canal_umbralizado) {
    wavethresh::imwr(canal_umbralizado) # Transformada wavelet inversa
  })

  # 4. Reconstruir la imagen combinando los tres canales procesados
  imagen_reconstruida <- abind::abind(ilwd[[1]], ilwd[[2]], ilwd[[3]], along = 3)
  imagen <- Image(imagen_reconstruida, colormode = 'Color')

  return(imagen)
}
```

Para comenzar el análisis, vamos a emplear dos imágenes muy parecidas (imágenes 4 y 5). Una de ellas tiene muy alta resolución mientras que la segunda cuenta con una calidad mucho menor. El objetivo es determinar si la resolución de la imagen afecta a la hora de eliminar ruido de esta. Vamos a probar con el primer tipo de ruido, ruido gaussiano.

Aplicamos la función `resize_imwd` creada anteriormente para obtener una matriz con las dimensiones necesarias para aplicar la transformada wavelet.

Una vez tenemos las imágenes con ruido generadas y redimensionadas adecuadamente, podemos aplicar la función `imwd` a cada uno de los tres canales (R, G y B). Empleamos la función `procesar_imagen_wavelet` que devuelve las imágenes reconstruidas después del thresholding. Para ruido gaussiano y para comenzar, vamos a elegir los parámetros por defecto de la función para el thresholding.

Finalmente, visualizamos los resultados. Primeramente, observamos las imágenes redimensionadas con ruido y la imagen obtenida tras el uso del método de thresholding para la eliminación de este. Observamos una principal diferencia entre ambas: la foto que contaba con menor resolución presenta también el peor resultado. Aunque el ruido haya sido eliminado, sus bordes están más difuminados y tiene muy baja calidad.



En segundo lugar, vamos a visualizar las imágenes originales y las imágenes sin ruido redimensionadas a su tamaño original, usando la función `resize_imwd_to_original`.



Vemos que efectivamente, la imagen que originalmente contaba con un número de píxeles mucho menor, la imagen 5, presenta una gran distorsión tras la eliminación de ruido.

A continuación vamos a comprobar que es lo que ocurre cuando añadimos ruido sintético sinusoidal y si la frecuencia de este afecta al resultado de la eliminación de ruido. Trabajaremos con una única fotografía: la imagen 1.

Está claro que los parámetros por defecto de la función `threshold` no son capaces de eliminar el ruido de tipo sinusoidal de la manera en que si lo era con el ruido de tipo gaussiano, un tipo de ruido aleatorio, al contrario que el sinusoidal, que es una señal periódica.



Vamos a variar parámetros de la función threshold para intentar quitar este ruido de manera más manual. En primer lugar, cambiamos el número de niveles al que aplicamos el umbral, para incluirlos a todos. Cambiamos policy a “manual” y variamos el valor del umbral de forma manual hasta encontrar uno que sea satisfactorio.

Con un valor de umbral de 4 y variando el tipo a “soft”, observamos que hemos conseguido eliminar el ruido sinusoidal de alta frecuencia, pero pagando un precio muy alto: los bordes de la imagen se disorsionan completamente y tenemos una muy baja resolución. Por otro lado, el ruido de frecuencia baja, aunque ha disminuido, claramente sigue presente en la imagen.



Probamos otros tipos de ruido: gamma, salt and pepper y ruido uniforme en las imágenes 2 y 3.

Siguiendo el mismo procedimiento, redimensionamos las imágenes y y realizamos las transformadas wavelet y el thresholding con la función `procesar_imagen_wavelet`.

Visualizamos los resultados. Parece que el algoritmo funciona correctamente para los tres tipos de ruido.

Redimensionamos las imágenes obtenidas a su tamaño original para poder compararlas con estas. Vemos que en este caso la eliminación de ruido ha sido bastante buena y no hay apenas distorsión ni suavizado de los bordes.



4.2.2 Ampliando las imágenes

Vamos a emplear ahora el otro método: aumentando la matriz con 0 hasta el tamaño adecuado para poder aplicar la función `imwd`. Comenzamos generando las imágenes con ruido. Se elige la foto con menor resolución (imagen 5) porque, al aplicar la función a fotos con mayor cantidad de píxeles, se genera un problema con el uso de la memoria en R para cargarlas debido a que la matriz se hace demasiado grande.

Se genera una función que ajusta cualquier imagen rectangular a un tamaño cuadrado. La diferencia con el método empleando `resize` es que mantiene sus proporciones originales al agregar relleno si es necesario. Esta transformación asegura que la imagen sea compatible con `imwd`.

Aplicamos esta función a la imagen 5 con los distintos tipos de ruido.

La imagen pasa de tener tamaño 1.600x1.066 en cada dimensión de color a 2.048x2.048, habiendo llenado este espacio con espacio negro.

Una vez tenemos las imágenes con ruido generadas y ampliadas en potencia de 2, aplicamos la función `imwd` a cada uno de los tres canales (R, G y B). Usando la función `procesar_imagen_wavelet` que devuelve las imágenes reconstruidas después del thresholding.

Aplicamos en cada ruido dos técnicas para seleccionar el umbral: Universal y fdr, además especificamos que el umbral sea 'hard'. Las imágenes se presentan sin el relleno para que sea más fácil de visualizar.

Recordemos que la imagen 5 es la de menor resolución, por lo que, al eliminar el ruido, los cambios no son tan notorios como en otras imágenes.

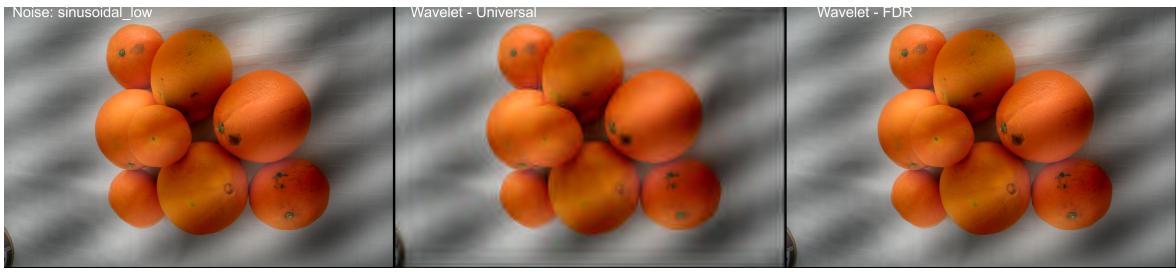
Eliminar Ruido Gaussiano: Se observa que al eliminar el ruido gaussiano el umbral FDR podría ofrecer una ligera mejora en la eliminación de ruido en comparación con el umbral Universal.



Eliminar Ruido Sinusoidal de alta frecuencia: Al igual que en el ruido gaussiano, se observa que al eliminar el ruido sinusoidal alto el umbral FDR podría ofrecer una ligera mejora en la eliminación de ruido en comparación con el umbral Universal.



Eliminar Ruido Sinusoidal de baja frecuencia: En este caso es muy difícil observar diferencias



En los casos de ruido tipo 'salt and pepper' y uniforme, se observa que el ruido se elimina, obteniendo una imagen mejorada con FDR.

Eliminar Ruido Salt and pepper:



Eliminar Ruido uniforme:



En el caso de ruido gamma, no se logra una eliminación adecuada con FDR, y el método universal parece ofrecer mejores resultados.

Eliminar Ruido gamma:



El umbral Universal emplea un valor fijo basado en el tamaño de la señal, lo que limita su capacidad para capturar completamente la variabilidad del ruido. En contraste, FDR estima la probabilidad de que un coeficiente wavelet provenga del ruido, lo que le permite adaptarse mejor a las características específicas de la señal. Como resultado, FDR puede ofrecer una eliminación de ruido más adaptativa, preservando mejor los detalles de la imagen. Por lo que, parece ser una opción más adecuada en la mayoría de los casos de ruido analizados anteriormente.

4.3 Función denoise.dwt.2d

En segundo lugar, vamos a emplear la función `denoise.dwt.2d`, un método más directo que realiza la transformada wavelet, el thresholding y la transformada inversa mediante esta función ya implementada en R. Modificaremos distintos argumentos de esta función para comparar los resultados:

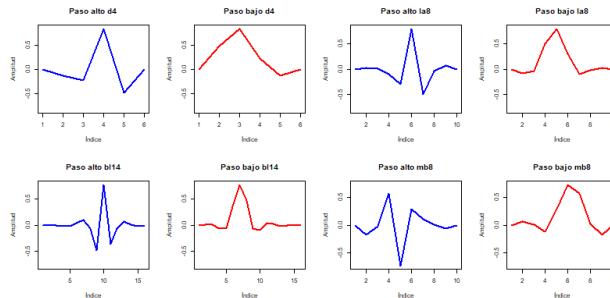
- Reglas de umbral (`rule`):
 - hard: Anula los coeficientes por debajo del umbral, asignándoles valor 0. Es una opción más agresiva.
 - soft: En lugar de anular directamente los coeficientes, les otorga un valor gradual en función de su cercanía al umbral. Permite una reducción del ruido menos abrupta.
- Niveles de descomposición (`J`):

Probaremos con 2, 3 y 4 niveles de descomposición. El ruido suele encontrarse en las altas frecuencias de los niveles más bajos, mientras que en los niveles altos se encuentran los detalles más finos. Esperamos que un mayor nivel de descomposición genere imágenes más suavizadas, pero con menos detalles.

- Filtros wavelet (`wf`):

Finalmente, realizaremos la descomposición de la imagen con 4 filtros wavelet distintos, 4 wavelets con diferentes formas (pueden ser simétricas o no), y diferente numero de coeficientes (suavidad).

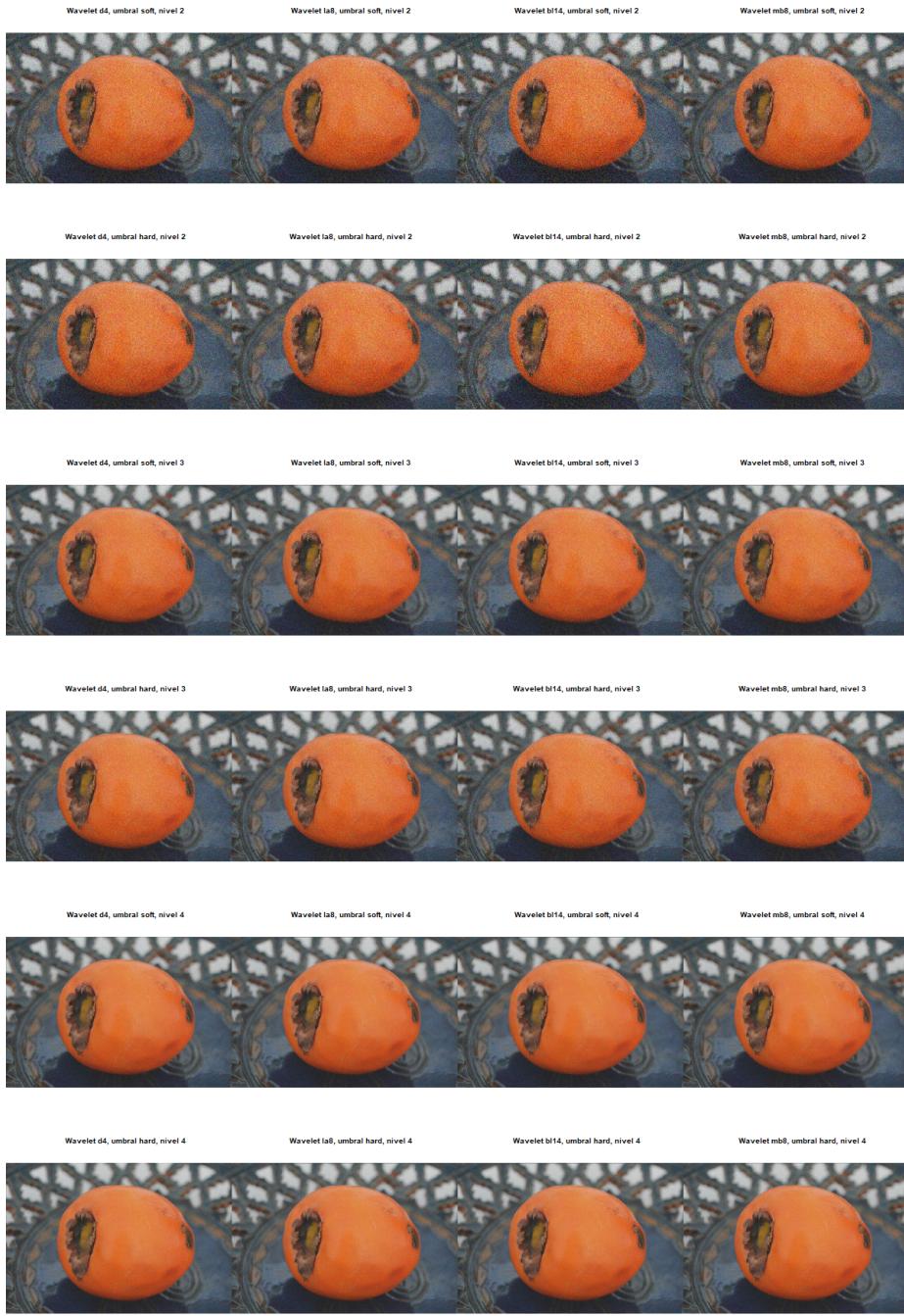
- d4 (Daubechies 4), la8 (Least Asymmetric 8), bl14 (Best Localized 14), mb8 (Maximum Flat 8)



Dado que se han realizado múltiples pruebas y los resultados son muy similares, se muestran solo aquellos que se han considerado más relevantes.

Primero, mostramos el resultado del proceso de eliminación del ruido de la imagen 1, a la que se le añadió ruido gaussiano. Este caso se utiliza como ejemplo porque ilustra las conclusiones que generalmente son válidas para el resto de imágenes y tipos de ruido.





Se observa que el uso de los diferentes filtros wavelet y reglas de aplicación del umbral no genera resultados con diferencias significativas. Sin embargo, el nivel de descomposición sí tiene un impacto notable.

Con un menor número de niveles (2-3), se logran conservar los detalles de la imagen, pero el ruido no se elimina completamente, con solo 2 niveles, el ruido sigue siendo evidente. Al aumentar a 4 niveles, se obtiene una imagen en la que el ruido es prácticamente inapreciable, aunque aparece algo más suavizada. Si se aumentaran aún más los niveles de descomposición, se empezarían a perder detalles importantes de la imagen. Con 4 niveles se consigue un buen equilibrio entre la eliminación del ruido y la conservación de los detalles. Este comportamiento se repite en los distintos tipos de ruido analizados.

A continuación se muestran algunas particularidades relevantes de los resultados obtenidos:

Por ejemplo, en el caso del ruido sinusoidal de alta frecuencia, el resultado del proceso de eliminación de ruido es distinto.



En este caso, vemos que en ninguno de los casos logramos eliminar completamente el ruido sinusoidal. Esto puede deberse a que este tipo de ruido presenta componentes muy específicas de alta frecuencia, que pueden coincidir con las frecuencias de los detalles importantes de la imagen, lo que dificulta eliminar el ruido sin comprometer los detalles de la imagen. El ruido sinusoidal era también el más complicado de eliminar cuando aplicabamos la función `imwd` y `threshold`.

Otro ruido que tiene resultados peculiares es el ruido de tipo salt-and-pepper.



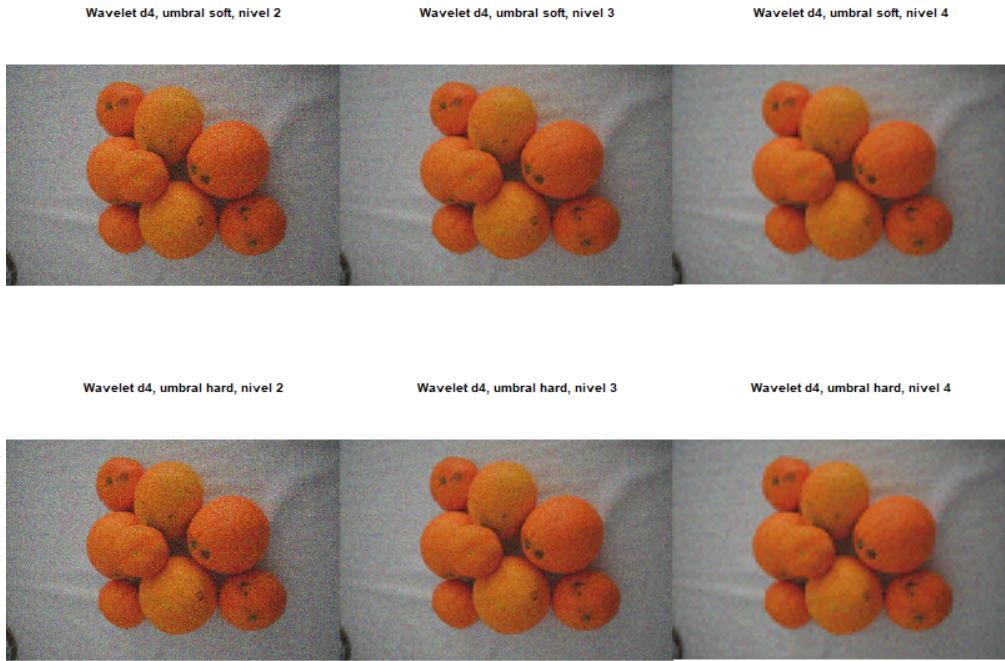


En este caso, observamos diferencias sutiles en los resultados en cuanto a:

- La regla de aplicación del umbral: Al utilizar la regla soft, se consigue una mejor eliminación del ruido. Esto podría deberse a que el ruido salt and pepper asigna valores extremos (cercaos a 0 y 1) a algunos píxeles. Al aplicar el método hard, no se atenúan adecuadamente los picos de ruido debido a su naturaleza más agresiva.
- El filtro wavelet empleado: El rendimiento con el filtro d4 es inferior al de otros filtros. Esto se debe a que el filtro d4 es un filtro corto (solo 4 coeficientes) y tiene una resolución de frecuencia limitada. Como resultado, no puede capturar eficazmente los picos abruptos de este tipo de ruido

Por último, presentamos los resultados del proceso de denoising de la imagen 5, a la que se le ha aplicado ruido gaussiano. A diferencia de la imagen 1, en este caso la resolución de la imagen es considerablemente inferior, lo que parece influir en los resultados obtenidos.





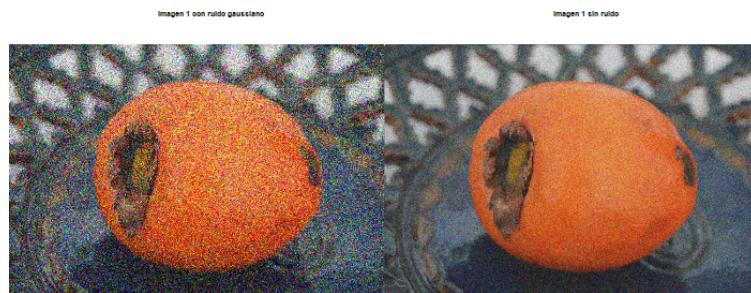
La diferencia más notable en comparación con el resto de resultados, es que en este caso, al aumentar el numero de niveles de descomposición a un número que nos permita eliminar la mayor parte del ruido, la calidad de la imagen se ve significativamente afectada, y obtenemos una imagen excesivamente suavizada. Esto puede deberse a que a medida que aumentan los niveles de descomposición, la imagen se descompone en frecuencias cada vez más altas, provocando la pérdida de detalles finos.

La baja resolución de la imagen hace que sea más difícil mantener un equilibrio entre la eliminación del ruido y la preservación de los detalles. Al aumentar los niveles de descomposición, el ruido se elimina en mayor medida, pero también se pierde mucha información útil.

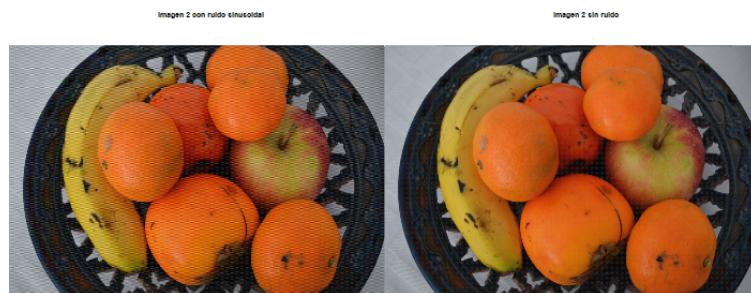
4.4 Transformada de Fourier

Por último, vamos a ver que es lo que ocurre cuando en lugar de transformadas wavelet empleamos transformadas de fourier.

Comenzamos probando con la imagen 1 y el ruido gaussiano. Se realiza la transformada de fourier y en el espacio de frecuencias se representan las frecuencias bajas en el centro y las altas en los extremos. El ruido, generalmente asociado a frecuencias altas, se intentará eliminar reduciendo las frecuencias altas.



La siguiente prueba se ha realizado con la imagen 2 y ruido sinusoidal alto. Este tipo de ruido es el que más problemas generó con el primer método.



Como se puede observar, la eliminación de este ruido con transformadas de Fourier es mejor que con el método de transformadas wavelet y thresholding manual, obteniendo resultados parecidos al de la función `denoise.dwt.2d`.

5 Conclusiones

A lo largo del trabajo se han desarrollado 3 métodos distintos para eliminación de ruido sintético en imágenes, dos basados en transformadas wavelet y uno en transformadas de Fourier. Se han generado diversos tipos de ruido (gaussiano, sinusoidal, gamma...) y se han analizado los resultados variando distintos parámetros.

Comenzamos por el método de transformada wavelet más manual. El primer problema con el que nos encontramos es el tamaño de las imágenes, lo cual solucionamos de dos maneras distintas: redimensionando las imágenes (lo que puede inducir distorsión) o agregando espacio en negro (lo que provocaba problemas de memoria en fotos más grandes o de mayor calidad). Obtenemos que la eliminación de ruido es bastante buena aunque se observa el suavizado de las imágenes, salvo en el caso sinusoidal, donde no se ha conseguido emplear parámetros adecuados para su eliminación.

`denoise.dwt.2d`, función implementada en R, emplea también transformadas wavelet y con este método encontramos resultados mejores. Por un lado, no se tienen problemas de tamaño, sin embargo, vemos que al emplear imágenes de baja resolución, se pierden muchos detalles al eliminar el ruido. Además, uno de los parámetros más influyentes en el resultado es el nivel de descomposición. En general, obtenemos buenos resultados tras la eliminación de ruido, habiendo un buen equilibrio entre el suavizado y el ruido. Finalmente, esta función sí es capaz de eliminar el ruido sinusoidal.

Finalmente, empleamos transformadas de Fourier. Destacamos en este caso que realizando manualmente la disminución de altas frecuencias si se ha conseguido eliminar el ruido sinusoidal sin una gran distorsión a la imagen.

Los tres métodos son capaces de realizar la eliminación de ruido sintético, destacando `denoise.dwt.2d` por su simpleza, facilidad de uso y resultados.