Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Planteamiento del problema

En el campo de la medicina cada vez es más importante la incorporación de las nuevas tecnologías. Así las imágenes médicas son de gran utilidad para la realización de un diagnóstico, encontrándonos actualmente diversas técnicas de captación, como son la tomografía computarizada, la resonancia magnética, los rayos X o los ultrasonidos. Sin embargo a la hora de extraer datos para la realización del diagnóstico, la calidad de estas imágenes no siempre es tan buena como se desearía, de ahí que existan numerosas líneas de investigación para la mejora de las mismas.

Cada tipo de imagen es degradada por diferentes factores, por lo tanto cada una de ellas requerirá un procesado diferente. En el caso de las imágenes médicas procedentes de ultrasonidos, éstas se ven afectadas por la presencia de ruido *speckle*, ruido de naturaleza no aditiva sino multiplicativa. Para la mayoría de la aplicaciones este ruido dificulta el análisis de la información, por lo tanto las técnicas de realce en las ecografías estarán encaminadas a la eliminación de este tipo de ruido. Además para el posterior tratamiento de la imagen, es necesario un preprocesado con el que se consiga mejorar el contraste, difuminar ciertas regiones de interés y hacer más agudos los bordes y las estructuras útiles, facilitando así operaciones posteriores, como una segmentación de estructuras.

Se puede observar que la mayoría de las técnicas ordinarias no permiten preservar bien los bordes cuando se suavizan las imágenes que contienen ruido [Aja01]. Es decir, el principal problema que se encuentra en la eliminación de ruido, es la existencia de un compromiso entre la cantidad de ruido eliminado y la conservación de los bordes y estructuras finas. Sin embargo debido a la naturaleza no-lineal que poseen los filtros borrosos y a la flexibilidad de diseño que permiten, se encuentra la posibilidad de realizar un procesado en función de la información más bien vaga que aporta la imagen ruidosa. Por este motivo el uso de técnicas borrosas [Mendel95] en este campo, ha cobrado gran importancia en los últimos años.

Echando un vistazo rápido al estado del arte de los filtros borrosos existentes se encuentran varios tipos genéricos:

- *Filtrado directo*: se basan en la utilización de reglas borrosas del tipo *si ... entonces*. Mediante la combinación adecuada de estas reglas se puede conseguir el efecto de filtrado deseado. Una clase importante de este tipo de filtrado son los filtros de inferencia borrosa gobernados por una acción adicional ó FIRE (*Fuzzy Inference Ruled by Else-action*), desarrollados por F.Russo y Ramponi [Russo98] .

- *Extensión borrosa de filtros existentes*: esta extensión consiste en la sustitución de un parámetro de umbral por una simple función de pertenencia, o en la aplicación de un conjunto de reglas borrosas para adaptar algún parámetro del filtro correspondiente.

- *Fusión suave de filtros existentes*: mediante reglas borrosas se combinan distintos filtros no borrosos para poder aprovechar así las ventanjas de cada uno de ellos y evitar sus inconvenientes.

- *Filtros borrosos de mediana ponderada*: el objetivo es el cálculo de los pesos del filtro de mediana ponderada mediante un proceso de inferencia borroso.

- *Filtros de agrupamiento borroso*: son un tipo de filtros en los que la salida se obtiene generalmente por medio de un proceso iterativo.

La elaboración de este Proyecto Fin de Carrera se centrará en el tratamiento de imágenes médicas captadas mediante ultrasonidos. Se intentará adecuar la imagen para una posterior segmentación, por lo tanto el preprocesamiento de la imagen consistirá en la eliminación del ruido *speckle* y el realce de bordes. Para ello se hará uso de la lógica borrosa aprovechando su potencia en el tratamiento de datos imprecisos. Por otra parte se ha de tener en cuenta que la mayoría de los filtros que se pueden encontrar en la literatura están diseñados para eliminar ruido gaussiano, impulsivo o para realzar bordes, pero muy pocos se preocupan específicamente del ruido *speckle*. Por lo tanto el trabajo que se va a realizar, estará encaminado a la mejora o diseño de un filtro específico válido para la eliminación de este tipo de ruido.

En la bibliografía, son muchos los filtros existentes para el tratamiento de imágenes, tanto en el dominio espacial como frecuencial [Pratt91]. Sin embargo este proyecto se basará en la manipulación directa de píxeles, es decir, el filtrado estará localizado en el dominio espacial. Se realizará un procesado por ventanas, pues la información en la que se basará la inferencia provendrá tanto de cada uno de los píxeles como de sus vecinos. Esto se debe a que se quiere tratar de forma diferente las zonas de la imagen relativamente constantes y aquellas que pertenecen a un borde o discontinuidad, intentando solventar así el compromiso bordes vs. ruido que viene apareciendo tradicionalmente en el filtrado de imágenes.

Por lo tanto en este proyecto, haciendo uso de la lógica borrosa y siguiendo las características de los filtros de control borroso (IFCF) desarrollados por Farbiz, Mehaj, Motamedi y Hagan [Kerre00], se implementarán diferentes filtros paso alto para la detección de bordes (Prewitt, Sobel ...) así como filtros anisótropos [Aja01] que presenten un buen comportamiento ante la presencia del ruido *speckle*.

1.2. Objetivos

El **objetivo fundamental** perseguido durante la realización de este proyecto, es el tratamiento de las imágenes captadas por ultrasonidos adecuándolas para una posterior segmentación. De esta meta principal se derivan los siguientes objetivos:

- Diseño e implementación de filtros paso alto detectores de bordes que se comporten bien ante la presencia de ruido *speckle*. Se desarrollarán y analizarán diferentes operadores para máscaras de diversos tamaños.
- Desarrollo de nuevos filtros anisótropos actuando sobre imágenes con ruido *speckle*.
- Implementación de un *toolbox* de filtros de difusión anisótropa. Así se desarrollarán diferentes versiones del filtro de Perona-Malik integrándolo con los filtros borrosos paso alto implementados anteriormente, y se insertará un controlador borroso que determine el valor de ciertos parámetros así como el número de iteraciones [Chan97].
- Comparación de los nuevos filtros implementados con los ya existentes para poder dilucidar cuál o cuales son los más adecuados en el problema que nos ocupa. Será de ayuda las medidas dadas por diferentes índices de calidad, como la similitud estructural (SSIM) [Wang04].

1.3. Fases

Para conseguir los objetivos propuesto, el desarrollo del proyecto se organizará de acuerdo con las siguientes **fases**:

- La primera fase se corresponderá con el proceso de documentación. Este se puede dividir en tres bloques fundamentales:
  - Lectura de bibliografía sobre lógica borrosa.
  - Estudio del estado del arte en temas de ruido *speckle*.
  - Análisis pormenorizado de filtros iterativos de control borroso [Kerre00] y filtrado anisótropo [Aja01], [Weickert98].

- Diseño de filtros paso alto siguiendo las directrices de los filtros iterativos de control borroso. Realización de un *toolbox* que contenga dichos filtros e implementación en C de los mismos, debido a la gran carga computacional y el elevado número de bucles que se requieren en el procesado de imágenes. Más concretamente se utilizará la utilidad *mex* de la que dispone *Matlab* y mediante la que dentro de un programa escrito en C se puede tener acceso a las funciones que *Matlab* tiene en sus *toolboxes* [Apiref] [Apiguide].
- Diseño de filtros anisótropos siguiendo también el esquema de los filtros IFCF. Estos filtros también se implementarán en C.
- Diseño de nuevas versiones del filtro de *Perona-Malik*. Para ello modificaremos el algoritmo introduciendo los filtros paso-alto creados en la primera fase y añadiremos un controlador borroso que determine ciertos parámetros así como el número de iteraciones.
- Análisis de los resultados obtenidos y comparación de los posibles nuevos filtros con los ya existentes. Se pretende llegar a una serie de conclusiones en términos de las ventajas e inconvenientes de cada filtro, y bajo qué condiciones. Se hará uso de ciertos índices de calidad, como la similitud estructural (SSIM) [Wang04].

1.4. Estructura del proyecto

La memoria de este proyecto se estructura en 8 capítulos y 3 apéndices, el contenido de los cuales se detalla a continuación:

- El capítulo 1 corresponde a esta introducción.
- En el capítulo 2 se describen los principales métodos empleados para captación de datos médicos. En concreto se describe la tomografía computerizada, la resonancia magnética, los ultrasonidos y los rayos X.
- El capítulo 3 constituye una visión panorámica de la lógica borrosa así como de la teoría de los conjuntos borrosos. Se introducen la notación y los conceptos que serán utilizados en los desarrollos posteriores. Se presenta este tipo de lógica en comparación con la lógica clásica ya que posiblemente la comprensión de aquella sea más fácil.
- En el capítulo 4 se da un repaso a los métodos clásicos de realce. Principalmente se describen los métodos basados en el dominio espacial y los métodos frecuenciales.
- Posteriormente, en el capítulo 5, se explican los distintos tipos de realce de naturaleza borrosa existentes. Se distingue entre métodos de filtrado directo, extensión borrosa de filtros ya existentes, fusión suave de filtros existentes y filtros de agrupamiento borroso.

- En el capítulo 6 se presentan las principales aportaciones del proyecto. Se realiza una explicación detallada de los diferentes filtros implementados en cada uno de los tres *Toolbox* creados: *HPFborroso*, *Toolbox Anisótropo* y *Toolbox Difusión*. De cada uno de ellos se explica su algoritmo, los diferentes opciones que se han estudiado para su implementación y las posibles mejoras que se podrían introducir.

- En el capítulo 7 se muestran los resultados de las experimentaciones realizadas. Se estudia el comportamiento de cada uno de los filtros ante tres situaciones diferentes: cuando se enfrentan a ecografías, a imágenes sintéticas afectadas por ruido *speckle* y a imágenes sintéticas con ruido *gaussiano*.

- En el capítulo 8, se presentan las conclusiones y aportaciones del presente trabajo. Se hace una reflexión final sobre todos los aspectos tratados en esta memoria y se proponen algunas líneas futuras de trabajo.

- En el apéndice A se lista el manual de usuario creado para que la utilización de los programas realizados sea fácil. Para cada uno de los filtros realizados se da una pequeña descripción general del algoritmo, la sintáxis correcta que hay que emplear, el tipo de datos que se soporta y un ejemplo de utilización.

- En el apéndice B se describen algunos de los detalles principales de implementación de los diferentes *toolboxes*, además de una visión general de cómo funcionan los archivos *mex* de *matlab*.

- Finalmente el apéndice C, en el cual se muestra el desarrollo matemático del gradiente realizado para poder obtener la expresión que permite aplicar el algoritmo de difusión anisótropa al realce de imágenes.

Capítulo 2

Imágenes médicas

2.1. Introducción

Las aplicaciones médicas de visualización 3D constan de los siguientes pasos: adquisición de datos, procesado de imágenes, creación de modelos y operaciones de visualización.

En este capítulo se analiza el primero de estos pasos, la captación de datos volumétricos. A lo largo del mismo se describirán las distintas técnicas empleadas en la actualidad para obtener dichos datos, haciendo especial hincapié en los ultrasonidos, dado que con las imágenes obtenidas con este método se ha trabajado en este proyecto[1].

Últimamente se viene denominando *modalidades de imagen* a las diferentes técnicas de obtención de imagen médica. El elemento básico que define las diferentes modalidades es el tipo de energía utilizada. Como en casi todo proceso de medida, la obtención de imágenes médicas implica irradiar la muestra (el paciente, en este caso) con algún tipo de energía. El carácter de la misma dará nombre a la modalidad correspondiente. Las modalidades fundamentales de imagen médica son: Radiología (radiación electromagnética: rayos X), Ecografía (energía ultrasónica), Medicina Nuclear (radiación electromagnética: radiación gamma) y Resonancia Magnética (radiación electromagnética: ondas de radio). Es interesante resaltar que el tipo de energía utilizada determina el tipo de interacción bioquímica que se produce en los tejidos biológicos y, por tanto, en qué medida puede ser nociva para el organismo. Se denominan radiaciones ionizantes aquellas que por su alta energía son capaces de inducir directamente reacciones químicas, a través fundamentalmente de la ionización de diferentes moléculas. Las radiaciones no ionizantes se limitan a producir calentamiento que, en principio, no presenta efectos biológicos relevantes si es ligero.

La introducción de estas *técnicas de imagen avanzadas* ha mejorado significativamente la calidad de la atención médica que se da a los pacientes. Los métodos de imagen no

[1] La mayor parte de la información necesaria para la realización de este capítulo ha sido obtenida de [Bronzino95]

invasivos permiten que los especialistas realicen diagnósticos cada vez más precisos y que se puedan utilizar tratamientos más adecuados. Las técnicas de *imagen médica* se aplican en medicina de laboratorio, en intervenciones quirúrgicas, en radioterapia, medicina nuclear y radiología del diagnóstico, entre otras.

Es muy frecuente clasificar las modalidades en *morfológicas* (o estructurales) y *funcionales*. Las primeras se caracterizan por producir imágenes de muy buena resolución, que permiten una representación muy detallada de la anatomía del paciente. Las segundas, en cambio, se caracterizan por aportar información sobre el funcionamiento de los diferentes órganos o sistemas: algún rasgo de su metabolismo, su perfusión sanguínea, su capacidad para acumular ciertas sustancias, etc.

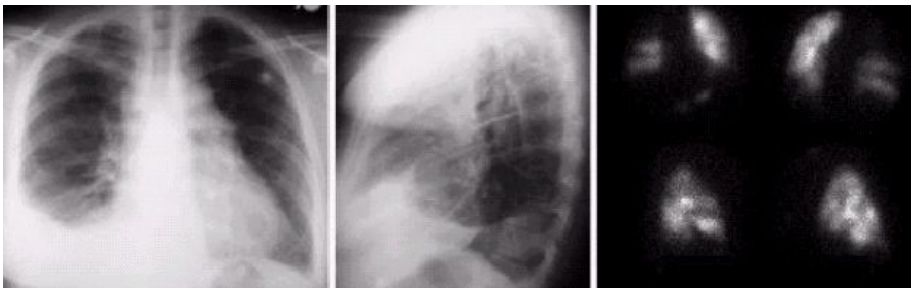


Figura 2.1: Imagen morfológica y funcional. Izquierda y centro: Radiografías de tórax. Derecha: Gammagrafía pulmonar, en la que se aprecia una clara embolia pulmonar, invisible en las radiografías de tórax.

La figura 2.1 ilustra esta diferencia: a la izquierda vemos un estudio radiográfico de tórax, modalidad morfológica que representa los pulmones con una resolución inferior al milímetro. A la derecha, una gammagrafía pulmonar de perfusión del mismo paciente: es una imagen de Medicina Nuclear que representa el flujo sanguíneo pulmonar, con una resolución que puede ser de uno o dos centímetros (obsérvese el aspecto borroso de la imagen). Sin embargo, en esta última podemos observar la existencia de defectos de perfusión que corresponden a una embolia pulmonar, totalmente invisibles en la radiografía torácica. Esta es la ventaja de las imágenes funcionales: asumimos su falta de resolución a cambio de la interesante información que proporcionan sobre diferentes aspectos del comportamiento de sistemas biológicos.

Otra característica importante de las imágenes médicas deriva de su capacidad para separar objetos que se hallan a diferentes profundidades. Se llaman *imágenes proyectivas* aquéllas que representan la suma de todas las estructuras del objeto, proyectadas sobre una superficie bidimensional. Por el contrario, cuando el método de imagen es capaz de separar diferentes planos (cortar la muestra en rodajas), cada uno de los cuales se representa en una imagen bidimensional, se denomina *tomográfico*. La imagen tomográfica crea menos problemas de superposición de objetos, facilitando notablemente su interpretación.

2.2. Tomografía Computerizada

El campo médico [Cho93] utiliza la tomografía computerizada para obtener imágenes de cortes transversales del cuerpo humano. Estas imágenes pueden servir en sí para ayudar a los médicos en sus diagnósticos. Pero además, estas imágenes se pueden combinar y obtener una visualización tridimensional de las mismas. Si a esta visualización tridimensional se le añade la posibilidad de ocultar tejidos u órganos haciéndolos transparentes, visualizar parte del volumen, la posibilidad de navegar por el interior de los mismos, etc., es en estos casos cuando estas herramientas se pueden considerar como de gran ayuda para el diagnóstico, planificación o simulación quirúrgica. Herramientas que cada vez adquieren mayor relevancia en el ámbito hospitalario y que gracias a los esfuerzos realizados en los últimos tiempos están adquiriendo cada vez mayores prestaciones.

Estrictamente hablando, la imagen médica apareció en 1895 con Wilhelm Konrand Röntgen que descubrió los rayos X, que posibilitaban visualizar estructuras internas del cuerpo humano tales como huesos. La imagen de rayos X es una imagen agrupada en la que todos los objetos entre la fuente de radiación X y la radiografía aparecen superpuestos uno encima de otro. Cuando los rayos X pasan a través del cuerpo se absorben en distinta cantidad dependiendo de la densidad de los objetos con los que se encuentran. Los tejidos blandos absorben pequeñas cantidades de radiación, mientras que los huesos absorben más. Como resultado, en la imagen final, los tejidos blandos aparecen oscuros y los huesos aparecen claros. Desde los tiempos de Röntgen, quizá, el descubrimiento más revolucionario fue la tomografía computerizada en 1972. La tomografía crea una imagen de cortes transversales de un objeto sólido.

La Tomografía Computerizada (TC) se utiliza principalmente de 3 modos [Baxes94]. La técnica original es el *modo transmisivo* y utiliza una fuente de radiación X. Los rayos X se transmiten a través del objeto y se reciben en los dispositivos de detección de rayos X. La señal recibida en el detector es proporcional a la densidad de los elementos del objeto. También, se pueden utilizan fuentes de presión ultrasónica junto con los detectores asociados para crear imágenes tomográficas computerizadas transmisivas.

El *modo emisivo* de Tomografía Computerizada se basa en la emisión de una señal detectable del objeto. El objeto se puede excitar directamente o se puede introducir una sustancia que sea excitante. En cualquier caso, los detectores reciben la señal emitida. Existen varios sistemas de este tipo, por ejemplo: Imágenes de Resonancia Magnética (RM) y Tomografía por Emisión de Positrones. La tomografía RM excita determinadas moléculas del objeto colocándolo en un gran campo magnético que cambia. Las reacciones moleculares se miden detectando las emisiones de frecuencia de radio de las moléculas en respuesta al campo magnético. En las imágenes de Tomografía por Emisión de Positrones se introduce una sustancia en el objeto. La sustancia emite un flujo constante de positrones. Cuando los positrones emitidos se encuentran en reposo, interactúan con un electrón y crean dos fotones de rayos gamma; alejándose entre ellos. Dos detectores siguen la pista a los fotones, y determinan la posición del positrón emisor. Se pueden utilizan fuentes de presión ultrasónica junto con los detectores asociados para crear imágenes

tomográficas computerizadas transmisivas.

El tercer tipo de Tomografía Computerizada es el *modo reflectivo*. Al igual que en el modo transmisivo, una fuente transmite una señal al objeto. En vez de pasar por el objeto, la señal entra en el objeto y es reflejada por los elementos internos del objeto volviendo al dispositivo detector. La señal recibida en el detector es proporcional a la densidad de los elementos del objeto. La Tomografía Computerizada reflectiva tiene la ventaja de no necesitar que el objeto se rodee con fuentes y detectores. Para implementar la Tomografía Computerizada reflectiva se utilizan presión ultrasónica, fuentes de radar y detectores.

Se puede crear imagen tomográfica computerizada sintética retroproyectando proyecciones de imágenes individuales. Dado que la geometría de la imagen resultante se crea a partir de la información de las proyecciones, la Tomografía Computerizada puede actualmente considerarse como una forma de restauración de imágenes, en su definición más amplia.

Las técnicas de Tomografía Computerizada crean imágenes de cortes transversales de un objeto juntando numerosas imágenes de proyección sobre el objeto en el plano de interés. Una imagen de proyección es una imagen unidimensional en la que el brillo de cada píxel es igual a la absorción de rayos X en la sección del objeto. Combinando las múltiples vistas de proyección, se sintetiza la imagen del corte transversal. Según el plano de orientación existen tres tipos de cortes: axial, coronal y sagital. Estos planos se pueden visualizar en Fig.2.2.

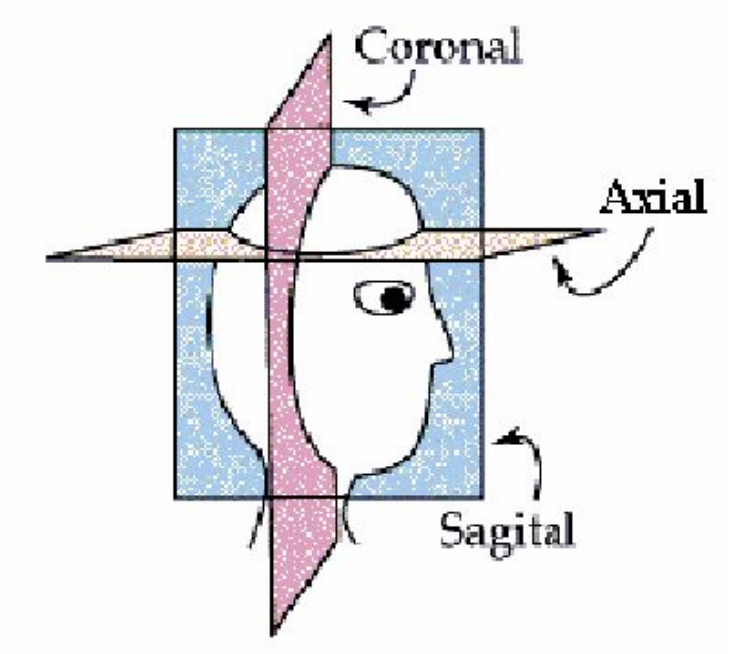


Figura 2.2: Orientación de los planos axial, coronal y sagital.

En TC si el corte es axial dicha imagen se conoce como TAC. Una vez tomadas las proyecciones de un corte concreto, puede empezar el proceso de reconstrucción. La imagen

del corte se crea retroproyectando las imágenes de proyección unidimensional individuales. En Fig.2.3 se muestra un ejemplo de un TC axial, sagital y coronal.

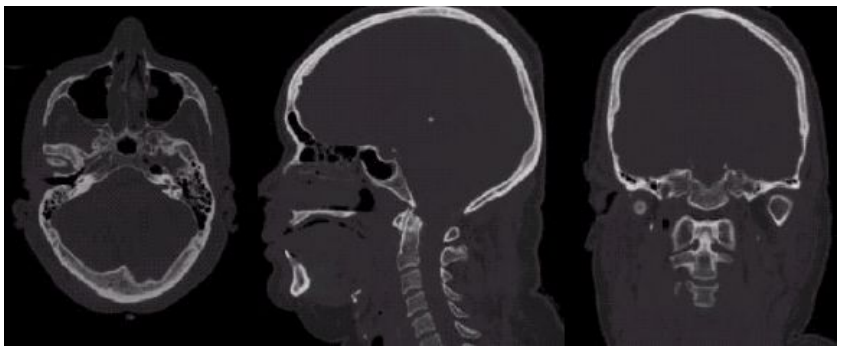


Figura 2.3: Figura 18. TC axial, sagital y coronal de la cabeza.

2.2.1. Tipos de escáneres TC

En esta sección se comentarán las distintas generaciones de escáneres que han ido apareciendo [Seeram94]. En la primera generación de escáneres, cada proyección se crea cuando una fuente de rayos X dirige un rayo X al objeto y se recibe su energía en el detector del otro lado del objeto, como se muestra en Fig. 2.4. El detector crea un brillo que es proporcional a la absorción del material en el objeto a lo largo de la línea entre el detector y su fuente.

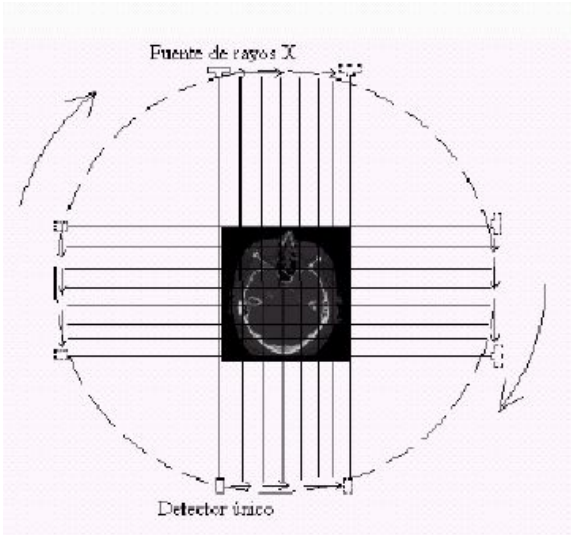


Figura 2.4: 1ª generación escáner TC (Rayos paralelos, movimiento-giro).

La fuente y el detector de rayos X son parte de una estructura circular que rodea al objeto en el plano de la sección transversal deseada. Después de la adquisición de la primera

imagen de proyección, se gira ligeramente (un grado) la estructura alrededor del objeto y se adquiere otra imagen de proyección. Este proceso se realiza 180^o alrededor del objeto. Así, se han adquirido 180 imágenes. Mientras que cada imagen de proyección tiene poco valor por sí misma, cuando se combinan se obtiene la imagen del corte transversal.

El proceso rota sólo 180^o alrededor del objeto, y no 360^o. Dadas las propiedades sintéticas del sistema, la proyección tomada en 0^o es idéntica a la tomada en 180^o, justo invertida. Las adquisiciones entre 180^o y 360^o únicamente imitarían a las adquiridas entre 0^o y 180^o, no aportando nueva información.

La primera generación de escáneres utiliza fuentes y detectores paralelos para crear imágenes de proyección, en las que el tiempo necesario para tomar una imagen varía entre 4.5 minutos y 5.5 minutos.

La segunda generación de escáneres, rayos en abanico, se basa en el principio de la primera, mover-girar, con algunas diferencias. La proyección de rayos en abanico utiliza una fuente de rayos X única para iluminar una línea de detectores (mayor número que en el caso anterior), como se muestra en Fig.2.5. Los rayos son divergentes en lugar de ser paralelos. El funcionamiento es el siguiente, en primer lugar, se lanzan los rayos, se gira el tubo de rayos X y el conjunto de detectores con un incremento grande (con respecto a los escáneres de la primera generación). El proceso se repite para cubrir 180^o. Los incrementos en los giros y el mayor número de detectores hacen que el tiempo de escaneo varíe entre 20 segundos y 3.5 minutos. En general, el tiempo es inversamente proporcional al número de detectores.

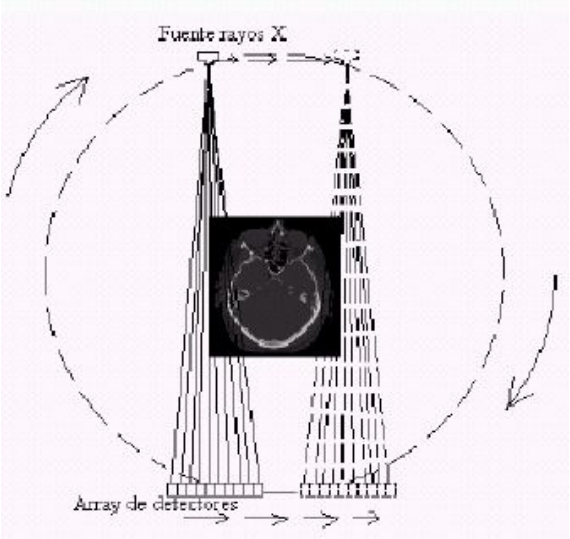


Figura 2.5: 2ª generación escáner TC (Rayos en abanico, movimiento-giro).

La tercera generación de escáneres se basa en el uso de una geometría de rayos en abanico, que gira continuamente alrededor del paciente 360^o. Los detectores ahora no forman una línea recta, sino que siguen una trayectoria curva, y forman un arco de 30^o a

40^o con el tubo de rayos X. Cuando el tubo de rayos X y los detectores giran, se toman las imágenes de proyecciones. Se toma una vista por cada punto fijo del tubo y cada detector, como se ve en Fig.2.6. En este caso se toman imágenes de los 360^o, no de 180^o como en los casos anteriores. La toma de datos en esta generación es más rápida que la de las generaciones anteriores, generalmente unos pocos segundos.

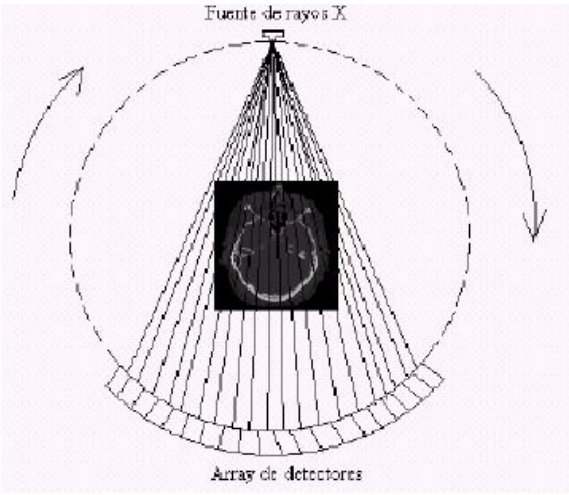


Figura 2.6: 3ª generación escáner TC (Rayos en abanico, sólo giro)

La cuarta generación de escáneres, tiene dos tipos de geometría, un tubo o fuente de rayos en abanico con un número de detectores fijo dispuestos en forma de anillo, y un tubo de rayos fuera del anillo de detectores.

En la primera geometría, el tubo de rayos X está fijo en una posición. Los rayos describen un gran abanico. El tubo se mueve de punto a punto en el círculo, los rayos chocan con un detector de punto a punto. Los rayos no se producen al mismo tiempo (como en la tercera generación), sino secuencialmente, cuando el tubo se mueve de punto a punto durante su recorrido circular. Los tiempos de escaneado son muy cortos y varían entre escáneres, dependiendo del fabricante. El camino seguido por el tubo de rayos X es circular. Esto se aprecia en Fig.2.7.

En la segunda geometría, el tubo de rayos X gira fuera del anillo. Cuando gira, el anillo se inclina de forma que el rayo choque con un array de detectores situado lo más alejado posible del tubo de rayos X, mientras que los detectores más cercanos al tubo están fuera del alcance de los rayos. Véase Fig.2.8.

Los escáneres de primera y segunda generación están obsoletos y actualmente no están disponibles comercialmente. Cada vez los clínicos precisan mayor resolución en las imágenes y menor tiempo en su obtención. Esto hace que día a día los productores de escáneres mejoren sus productos y ofrezcan mayores prestaciones.

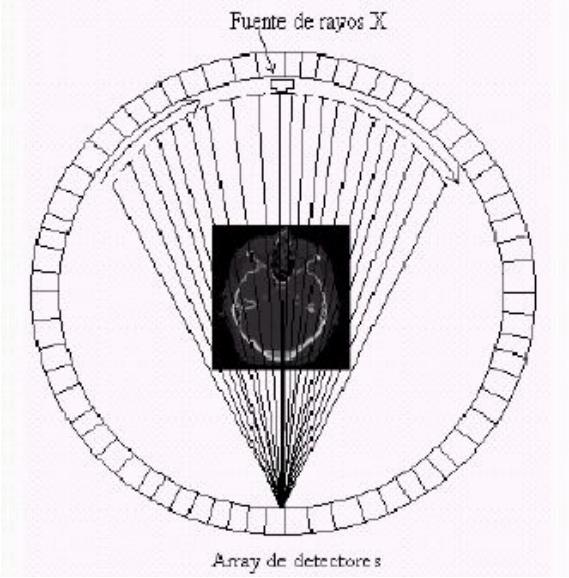


Figura 2.7: 4ª generación escáner TC (Rayos en abanico, detector circular)

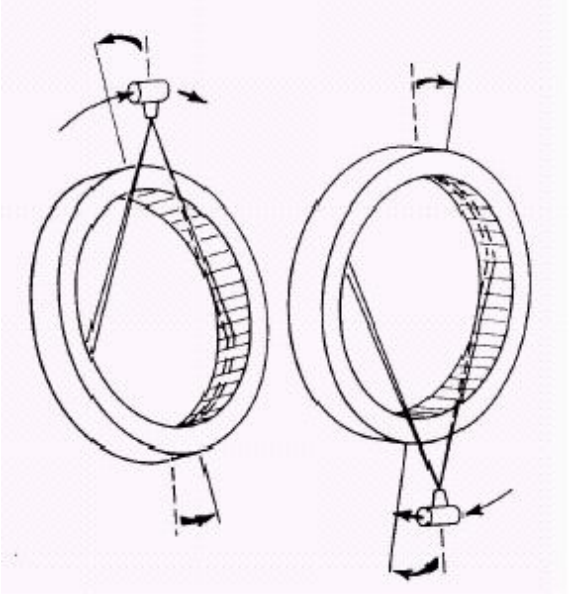


Figura 2.8: Fuente de rayos X fuera del anillo.

2.3. Resonancias Magnéticas

La resonancia magnética (RM) es similar a la TC en el sentido que también permite la obtención de conjuntos de datos tridimensionales correspondientes a la anatomía del paciente. Sin embargo, RM difiere sustancialmente con la anterior modalidad en la metodología empleada para la adquisición de las imágenes. Los escáneres de TC utilizan radiación de rayos-X para generar los datos necesarios para reconstruir las estructuras internas. Por otra parte, RM utiliza ondas de radio frecuencia (RF) y campos magnéticos para obtener las imágenes tridimensionales.

Las resonancias magnéticas (RM) representan una de la mayores innovaciones en la tecnología de imágenes médicas. RM proporciona, mediante técnicas no invasivas, imágenes del cuerpo humano que presentan un gran contraste entre tejidos blandos. El fenómeno de la resonancia magnética nuclear es relativamente nuevo dentro del campo del diagnóstico. Sin embargo, la técnica en sí misma es conocida desde hace unos 60 años en el campo de la química física. Fue descubierta en 1946 por Bloch y Purcel, dos investigadores que trabajaban de forma independiente en Stanford y Harvard respectivamente.

Los escáneres de RM usan la técnica de resonancia magnética nuclear para inducir y detectar una señal de radiofrecuencia, que es la manifestación del magnetismo nuclear. El término *magnetismo nuclear* se refiere a propiedades magnéticas débiles que exhiben algunos materiales, como consecuencia del spin nuclear asociado con el núcleo de sus átomos. En particular, el protón, que es el núcleo del átomo de hidrógeno, posee un spin distinto de 0 y es una excelente fuente de señales de RM. El cuerpo humano contiene un número enorme de átomos de hidrógeno –especialmente en el agua (H_2O) y moléculas de lípidos, hecho del que se aprovechan la gran mayoría de los estudios clínicos basados en RM.

2.3.1. Fundamentos de Resonancia Magnética

La resonancia, en un sentido físico, se define como la absorción de energía de una fuente a una frecuencia específica, generalmente denominada frecuencia natural o frecuencia de resonancia. En el caso de la resonancia magnética, la fuente es energía de radiofrecuencia, y el objeto resonante es el núcleo de los átomos bajo la acción de un campo magnético externo. En esta situación los núcleos promocionan a estados de mayor energía mediante la absorción de dicha energía. Este estado de mayor energía o excitación no es estable y no puede ser mantenido de forma indefinida. De esta manera, el núcleo regresa a su estado natural, radiando la energía RF previamente absorbida. Esta energía emitida puede ser considerada como la señal RM y depende del entorno molecular del núcleo emisor. A partir de esta radiación se puede obtener diversa información referente al entorno molecular. Un ejemplo simplificado del proceso se puede observar en Fig. (2.9).

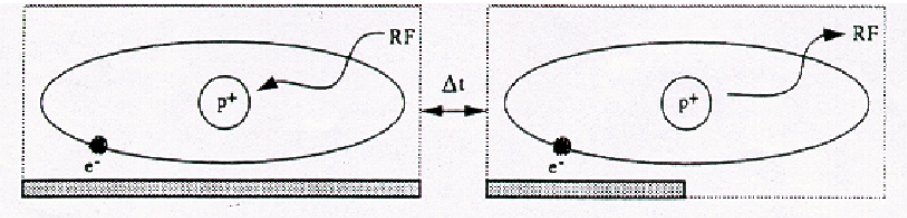


Figura 2.9: Representación simplificada del fenómeno de resonancia magnética nuclear. La barra inferior en cada figura representa la energía en el sistema.

2.3.2. Propiedades del núcleo bajo campo magnético externo

El fenómeno de resonancia magnética no aparece en todos los núcleos. Para que éste tenga lugar es necesario que el núcleo pueda rotar o que tenga un número impar de protones o neutrones. Los materiales en los que se verifica esto tienen un momento magnético nuclear que, aunque es pequeño, es observable.

Este es el caso de los protones (^{1}H), que son los más utilizados, aunque, como ya se ha comentados, también tienen interés el carbono (^{13}C), fósforo (^{31}P), sodio (^{23}Na) y flúor (^{19}F).

Los momentos nucleares están normalmente distribuidos de forma aleatoria, pero cuando se sitúan bajo un campo magnético intenso se alinean. A la colección de momentos nucleares producida se le suele llamar magnetización o spins. En cualquier caso, esta magnetización nuclear es muy débil comparada con el campo magnético aplicado y no puede ser medida cuando está alineada con el campo magnético externo.

Para que este débil momento pueda ser medido se utilizan técnicas de resonancia. La idea es medir el momento mientras oscila en un plano perpendicular al campo externo.Cuando el momento es perpendicular a este campo, sufre un par de torsión proporcional a la intensidad del campo externo. El par es siempre perpendicular a la magnetización, y provoca que los spins oscilen en un plano perpendicular al campo externo. La frecuencia de la rotación ω_0, llamada frecuencia de Larmour, es proporcional a la intensidad del campo:

$$\omega_0 = -\gamma \vec{B}_0$$

donde $\vec{B}_0$ denota el campo magnético externo medido en Teslas (T) y γ es la relación giromagnética, una constante específica del núcleo.

Concretamente, la relación giromagnética γ expresa la sensibilidad de un núcleo respecto a su señal RM. Esta relación representa la frecuencia de resonancia del núcleo bajo un campo externo de 1-Tesla. γ es una constante para cada isótopo. La sensibilidad se incrementa proporcionalmente con la frecuencia de la señal. El Hidrógeno presenta la relación giromagnética más alta y por lo tanto es el que manifiesta mayor sensibilidad a la acción de un campo externo.

Para poder apreciar la oscilación citada, es decir, para que la magnetización creada sea observable, hay que desviarla de la dirección de $\vec{B}_0$. Esto se consigue con un campo

de radiofrecuencia RF rotatorio débil. Se puede demostrar que un campo de este tipo introduce un campo ficticio en la dirección z de intensidad ω/γ. Sintonizando este campo de RF a ω_0 borramos, a efectos prácticos, el campo $\vec{B}_0$. Se puede decir que el campo de RF va desviando lentamente la magnetización del eje z.

Como los momentos de oscilación constituyen un flujo variable con el tiempo, producen un voltaje que puede ser medido en una antena acoplada para obtener las componentes x e y de la inducción. Como ya se ha comentado, la señal que nos da la RM del cuerpo humano se debe, fundamentalmente, a los protones del agua. Como estos protones están en entornos magnéticos idénticos (moléculas de H_2O), todos resuenan a la misma frecuencia, con lo que la señal es simplemente proporcional al volumen de agua. La innovación clave para RM es imponer variaciones espaciales al campo magnético para distinguir los spins por su localización. Aplicando un gradiente de campo magnético se produce una oscilación en cada región del volumen a distinta frecuencia. Mediante un análisis de Fourier de la señal se obtiene un mapa de la distribución espacial de los spins.

2.3.3. Mecanismos de Contraste

La tremenda utilidad de la RM es debida a la gran variedad de mecanismos que se pueden usar para crear una imagen de contraste. Si las imágenes sólo se pudieran obtener a partir de la densidad de agua, las resonancias magnéticas serían mucho menos útiles, ya que muchos tejidos aparecerían idénticos. Afortunadamente se pueden usar muchos mecanismos de contraste para distinguir los diferentes tejidos.

Los principales mecanismos usan la relajación de la magnetización. A continuación se describe los dos tipos de relajación:

- **Relajación de spin-red, T_1:** velocidad de recuperación de la componente z de magnetización hacia el equilibrio, después de ser polarizado por los pulsos de RF. La recuperación está dada por

$$M_z(t) = M_o(1 - e^{-t/T_1}) + M_z(0)e^{-t/T_1} \tag{2.1}$$

  donde M_0 es la magnetización de equilibrio. Diferencias en la constante de tiempo T_1 se pueden usar para obtener el contraste de la imagen. En la parte izquierda de la Fig. (2.10) se puede ver la recuperación de dos componentes T_1 diferentes. La componente con T_1 menor se recupera más rápido y produce más señal.

- **Relajación spin-spin, T_2:** velocidad de decaimiento de las componentes transversales de la magnetización (M_x y M_y), después de ser creadas. La señal es proporcional a la magnetización transversal y viene dada por

$$M_{xy}(t) = M_{xy}(0)e^{-t/T_2} \tag{2.2}$$

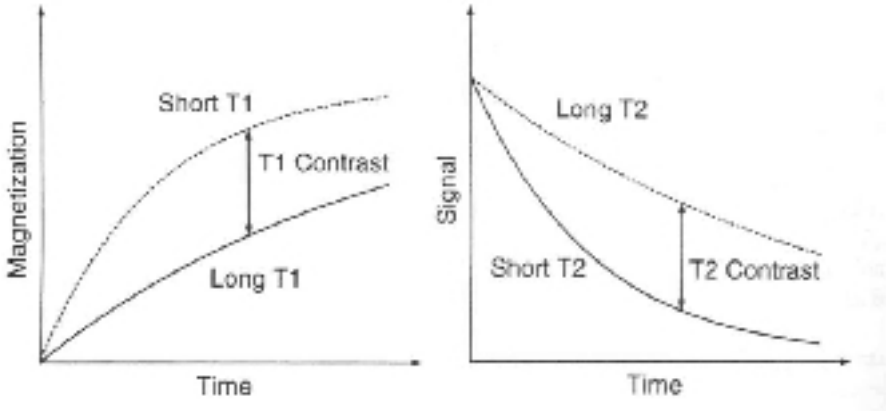


Figura 2.10: Los dos mecanismos principales de contraste en resonancias magnéticas, T_1 y T_2.

La imagen de contraste se obtiene retrasando la adquisición de datos. En la parte derecha de la Fig. (2.10) se muestra el decaimiento de dos componentes T_2 diferentes. La señal de la componente con T_2 menor decae más rápidamente. En el momento de recoger los datos, la componente con T_2 mayor produce más señal.

En la Fig. (2.11) aparecen ejemplos de estos dos tipos básicos de contraste. Estas imágenes son de la misma sección del cerebro. La imagen de la izquierda está obtenida a partir de T_1. El anillo exterior brillante es de grasa (materia blanca), que tiene un menor T_1 que la materia gris. La imagen de la derecha está obtenida a partir de T_2. El fluido cerebro espinal de los ventrículos es más brillante, debido a su mayor T_2. La materia blanca tiene un menor T_2 que la materia gris, por lo que aparece más oscura en la imagen.

Además de estos métodos básicos para obtener contraste, se pueden introducir agentes artificiales. Normalmente se administran de forma intravenosa u oral. Hay muchos mecanismos de este tipo, pero los agentes más usuales disminuyen T_1 y T_2. Disminuyendo T_1 se consigue una recuperación más rápida de la señal y una señal más alta en una imagen que se base en T_1. De esta forma, las regiones con el contraste realzado se muestran más brillantes con respecto al resto de la imagen.

2.3.4. Ventajas

Las imágenes de RM se caracterizan por el excelente contraste entre varios tipos de tejidos blandos del cuerpo. Además, para pacientes sin cuerpos ferromagnéticos extraños en el interior de su cuerpo, la RM es perfectamente segura y puede ser repetida, sin peligro, con tanta frecuencia como sea necesaria. Esto proporciona una de las principales ventajas de la RM con respecto a los rayos-X convencionales y los escáneres de tomografía computerizada. La señal usada en RM no es bloqueada en absoluto por regiones

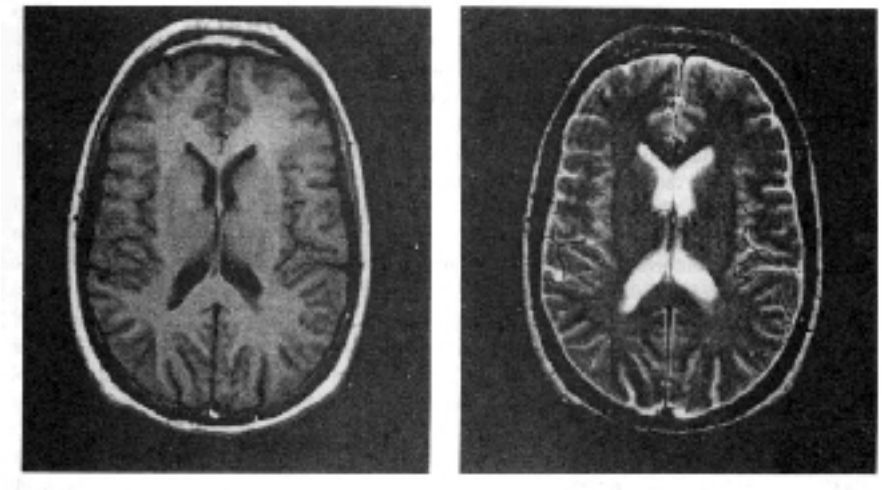


Figura 2.11: Ejemplos de imágenes de un voluntario normal mostrando el contraste T_1 a la izquierda y el T_2 a la derecha.

de aire o hueso dentro del cuerpo, lo cual supone una importante ventaja sobre los ultrasonidos. También, al contrario que en el escaneado mediante medicina nuclear, no es necesario suministrar materiales radiactivos al paciente.

2.4. Radiografías

Los rayos X y γ son formas de radiación electromagnética lo suficientemente energéticas para que al interactuar con átomos, puedan liberar electrones de átomos vecinos. Cuando esto se produce, se forman un par de iones, el electrón (e^-) y una molécula con carga positiva. Por lo tanto, los rayos X y γ son formas de *radiación ionizante*, y esta característica diferencia a estos rayos del resto del espectro electromagnético[2].

Una onda electromagnética de frecuencia ν tiene una energía proporcional a dicha frecuencia, con constante de proporcionalidad dada por la constante de Plank, h:

$$E = h\nu$$

donde $h = 4{,}135 \times 10^{-15}$ eV-s. Para el diagnóstico de imágenes médicas de rayos X, el rango de energías incidentes sobre los pacientes va desde los 10.000eV (10 keV) a unos 150 keV. En términos de longitudes de onda:

$$\lambda = \frac{c}{\nu}$$

[2]La mayor parte de la información necesaria para la realización de esta sección ha sido obtenida de [Beutel00]

donde *c* es la velocidad de la luz. El rango de longitudes de onda correspondiente a imágenes de diagnóstico va desde unos 0.1 nm a 0.01 nm.

Los rayos X y γ tienen diferentes características espectrales, pero fundamentalmente un rayo X de energía E es exactamente lo mismo que un rayo γ de energía E. Por definición, los rayos γ se originan en los núcleos de los átomos, mientras que los rayos X se originan en el nivel atómico del átomo.

Los rayos X pueden producirse por medio de varios métodos diferentes, sin embargo, la tecnología más comúnmente utilizada es el tubo de rayos X estándar que emite radiación de freno (*bremsstrahlung*, en alemán) con las características de los rayos X.

2.4.1. Radiación de freno

De acuerdo con la teoría clásica, si una partícula cargada se acelera, radiará energía electromagnética. Cuando electrones con energía inciden sobre un objetivo de metal (como en un tubo de rayos X), los electrones interactúan con el campo de los núcleos de los átomos del objetivo y experimentan un cambio en sus velocidades, y después se deceleran. La radiación de freno se produce por medio de este proceso. La intensidad total de la radiación de freno (integrada sobre todos los ángulos y energías) resultante de una partícula cargada de masa *m* y carga *ze* que incide sobre un objetivo con carga *Ze* es proporcional a:

$$I_{radiacion\_de\_freno} \alpha \frac{Z^2 z^4 e^6}{m^2} \tag{2.3}$$

La eficiencia de la radiación de freno se reduce si una partícula como un protón o una partícula α es la partícula cargada. Comparando con los electrones, los protones y las partículas α son aproximadamente tres millones de veces menos eficientes a la hora de producir radiación de freno. Por lo tanto, en la práctica han de escogerse los electrones para producir radiación de freno. El término Z^2 de la ecuación (2.3) indica que la eficiencia de la radiación de freno aumenta si el número atómico del objetivo es mayor, por lo que parece conveniente usar como objetivos átomos con un Z elevado.

La producción de la radiación de freno se ilustra en la Fig. (2.12). Los electrones incidentes se muestran pasando cerca del núcleo atómico del objetivo, y se emite radiación de freno a diferentes energías (E_1, E_2 y E_3). Aquellos electrones que tienen una forma de incidencia como e_1^- sólo dejan una pequeña fracción de su energía cinética. El rayo x resultante tiene una energía relativamente pequeña, E_1, y el electrón todavía tendrá una energía cinética considerable con lo que continuará interactuando con otros átomos del objetivo. Aquellos electrones que incidan como lo hace e_3^- liberarán toda su energía cinética, radiando como un rayo X con energía igual a la del electrón incidente E_3.

Cuando el objetivo al que se enfrenta la radiación es grueso parece claro que los electrones incidentes van a ir chocando con átomos del objetivo hasta que pierdan toda su energía, por lo que éstos no llegarán a zonas profundas del objetivo. Este proceso se

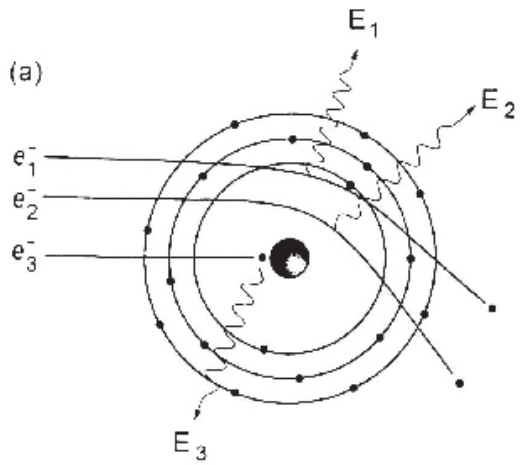


Figura 2.12: La radiación de freno se produce cuando electrones con energía son decelerados por medio del campo eléctrico del objetivo.

describe matemáticamente como:

$$\psi(E) = kZ(E_{max} - E) \tag{2.4}$$

donde $\psi(E)$ es un histograma de la intensidad (número$\times$ energía) de los rayos X de energía E por intervalo de energía, k es una constante, Z es el número atómico del objetivo, E_{max} es la energía cinética del haz de electrones incidente, y $E \leq E_{max}$. Nótese que aquí la dependencia de Z es lineal, mientras que en la ecuación (2.3), porque aquí el espectro total de la radiación de freno no se considera.

2.4.2. Detección de rayos X

Los detectores de rayos X pueden clasificarse en directos e indirectos. Un detector directo graba la carga eléctrica (normalmente como voltaje o corriente) que se obtiene directamente de la ionización de los átomos en el detector. Detectores de gas, como el *xenon*, usados en tomografía computerizada son detectores directos. Las películas son también detectores directos, y graban una especie de ”foto química”. Los detectores directos de estado sólido para imágenes de rayos X están hechos de material sólido (fotoconductor) situado entre dos electrodos (un electrodo está pixelado). En ausencia de rayos X, el fotoconductor actúa como un aislante, y fluye carga entre ambos electrodos. Cuando los rayos X alcanzan el fotoconductor, los electrones de valencia pasan a la banda de conducción y pasan a ser electrones de conducción, mientras que en la capa de valencia del fotoconductor quedan huecos móviles. Entonces, estas cargas migran a uno u otro electrodo con lo que la carga que se acumula se puede medir electrónicamente (ver Fig. (2.13) (a)).

Un sistema de detección indirecto, para los propósitos prácticos, es un detector de rayos X basado en chispas. Los rayos X interactúan con un fósforo, causando que éstos

emitan luz dentro o cerca del rango visible. Los fotones de luz visible entonces se propagan por difusión óptica a un fotodetector, como un fotodiodo de silicio (ver Fig. (2.4 (b))). Entonces, el fotodetector graba el patrón de luz visible dado por el fósforo como una imagen.

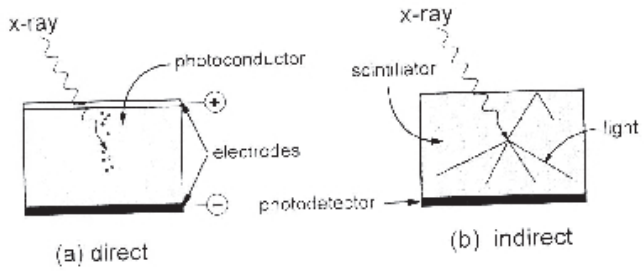


Figura 2.13: (a) Estrategia de detección para detectores directos. (b) Estrategia de detección usada por los detectores indirectos.

2.4.3. Eficiencia de absorción

Los detectores de rayos X necesitan interactuar con el rayo de fotones incidente para grabar su presencia; los rayos X que pasan a través del detector sin atenuarse son esencialmente malos. La clave de diseño de todos los detectores de rayos X para imágenes médicas es maximizar la eficiencia de absorción del detector, dados otros parámetros de eficiencia (como la resolución espacial). La eficiencia de detección cuántica (QDE) de un detector está dada por:

$$QDE = \frac{\int_{E=0}^{E_{max}} \Phi(E)(1-e^{-\mu(E)x})dE}{\int_{E=0}^{E_{max}} \Phi(E)dE} \tag{2.5}$$

donde x es el grosor del detector y $\mu(E)$ es su coeficiente de atenuación lineal, y $\Phi(E)$ es el espectro de rayos X. El QDE es simplemente la fracción de fotones incidentes del rayo X que son atenuados por el detector. Tanto en los detectores de rayos X directos como en los indirectos, la señal está relacionada con el total de energía absorbida en el detector, no con el número de fotones de rayos X. Esto es, los detectores de rayos X no son ”contadores de fotones” (como en aplicaciones de medicina nuclear), sino que son ”integradores de energía”. La eficiencia de absorción de energía (EAE) de un detector de

rayos X está dada por:

$$EAE = \frac{\int_{E=0}^{E_{max}} \Phi(E) E \left(\frac{\mu_{en}(E)}{\mu(E)} \right) (1 - e^{-\mu(E)x})}{\int_{E=0}^{E_{max}} \Phi(E) E dE} \tag{2.6}$$

El denominador de la ecuación (2.6) es la cantidad de energía incidente sobre el detector de rayos X (por unidad de área). La fracción de fotones atenuada en el detector está dada por el término $(1 - e^{\mu(E)x})$, y la cantidad de energía absorbida en el detector por fotón de rayo X atenuado está dada por $E(\mu_{en}(E)/\mu(E))$.

2.5. Ultrasonidos

El uso de los ultrasonidos para la aplicación a la imagen médica se remonta a 1950. Durante esa década, los avances de la tecnología y de la práctica clínica, hicieron de los ultrasonidos una modalidad de la imagen médica para el diagnóstico. En la actualidad, las técnicas de obtención de imágenes basadas en ultrasonidos, permiten la consecución de gran cantidad de información de diagnóstico útil tanto a nivel cuantitativo como cualitativo. Se basan en la emisión de pulsos de ultrasonido y la posterior recepción de los ecos procedentes de las estructuras internas. Las técnicas de ultrasonidos son muy atractivas debido a que consiguen obtener imágenes en tiempo real utilizando equipos portátiles con un coste relativamente bajo frente a las técnicas anteriores. Esta característica de imágenes en tiempo real, permite a los médicos observar el movimiento de las estructuras en el interior de un paciente, lo cual es especialmente relevante en los campos de ginecología, cardiología y pediatría, entre otros.

Una característica importante de los ultrasonidos es su probada inocuidad bajo dosis controladas. Este hecho, junto a la portabilidad de los equipos, su bajo coste y los modos de adquisición cuantitativos hacen de los ultrasonidos la técnica de obtención de imágenes médica más utilizada en la actualidad, a pesar de que las imágenes obtenidas son muy ruidosas.

Las principales características de las imágenes ultrasónicas son las siguientes [Sakas95]:

- Tienen una gran cantidad de ruido *speckle*.
- Su rango dinámico es mucho más pequeño que el de las imágenes tomográficas.
- Las regiones que marcan los bordes no son nítidas y tienen una anchura de varios pixels.
- Si existen dos objetos en la dirección del haz ultrasónico, el más lejano no se podrá ver. Por ejemplo, si hablamos de ecografías fetales, la mano del feto puede ocultar su cara.

- En los dispositivos que realizan un escaneado paralelo, se producen variaciones en la media del nivel de gris en imágenes consecutivas. Además, el alineamiento entre imágenes consecutivas no es todo lo bueno que sería deseable.

2.5.1. Aspectos físicos

Dado que los ultrasonidos se pueden considerar una técnica no invasiva, hoy en día las imágenes de ultrasonidos se utilizan en gran variedad de aplicaciones clínicas, como son la cardiología, obstetricia, visualización del abdomen en general y visualización vascular. Además están teniendo gran auge en aplicaciones quirúrgicas e intravasculares. Vemos como se van incrementando las aplicaciones de los ultrasonidos en el campo de la medicina, siendo necesarias técnicas de procesado de señal cada vez más efectivas.

Básicamente todos los escáneres de ultrasonidos de uso clínico proporcionan imágenes de los ecos recibidos. Se emite una onda acústica hacia el cuerpo del paciente utilizando un transductor manual móvil. La onda de ultrasonido interactúa con los tejidos internos que reflejan o dispersan parte de la energía transmitida, que va a ser la señal detectada por el transductor. La distancia d desde el transductor al objeto que causa el eco, está relacionada con el tiempo t total transcurrido entre el instante en el que se envía el pulso y el momento en el que se recibe, y la velocidad del sonido en ese medio c según la ecuación (2.7). Si conocemos la velocidad de transmisión del ultrasonido en el tejido, vamos a poder determinar la distancia desde el transductor al lugar donde se produjo la interacción.

$$d = \frac{1}{2}tc \tag{2.7}$$

Por otra parte, la ondas de presión se propagan a través del tejido a una velocidad característica. La velocidad del sonido en los tejidos varía en función del tipo de tejido, la temperatura y la presión. Normalmente se considera la temperatura y presión normal del cuerpo, dependiendo entonces la velocidad del sonido solo del tipo de tejido. Así, las características de la señal recibida (amplitud, fase, etc.) van a darnos información de la naturaleza de la interacción y, por tanto, información del tipo de tejido en el que ocurrió dicha interacción. La velocidad del sonido en los tejidos corporales blandos varía poco, tomándose en general como velocidad media en el tejido humano 1450 m/s.

La intensidad de la señal recibida $S(t)$ está relacionada con la señal transmitida, $T(t)$, las propiedades del transductor, $B(t)$, la atenuación del camino de ida y vuelta desde el dispersor, $A(t)$, y la intensidad del dispersor, $\eta(t)$.

Debido a que parte de la energía de la onda transmitida se absorbe, dispersa y refleja de forma continua según pasa a través del tejido, la onda se atenúa cada vez más según penetra más fuertemente dentro del tejido. La atenuación es debida principalmente a la absorción y a la dispersión. La atenuación es una función exponencial de la distancia,

normalmente modelada según la expresión

$$A(x) = A_o e^{-\alpha x} \tag{2.8}$$

donde $A(x)$ es la amplitud de la onda, x es la distancia recorrida, A_o es la amplitud inicial y α es el coeficiente de atenuación en Nepers, que depende de la frecuencia. En general los tejidos presentan una atenuación de 0.75dB/cm/MHz, por lo que la intensidad de la onda disminuye a la mitad cada 0.8 cm.

La energía acústica regresa al transductor debido a dos efectos: *reflexión especular* y *difracción*. La reflexión especular es debida a cambios en la impedancia acústica en las zonas de cambio de medio que sean significativamente mayores en extensión que la longitud de onda acústica. Hay que tener en cuenta que la magnitud de la onda recibida por el transductor de un reflector especular va a depender de la orientación angular. Una estructura interna va a devolver una señal fuerte, si la superficie es normal a la dirección de propagación de la onda y una señal muy débil, si la superficie es paralela a la dirección de propagación.

La difracción ocurre cuando las ondas acústicas interactúan con estructuras comparables o menores que la longitud acústica de la onda. Estas estructuras reflejan ondas débiles en todas las direcciones (dispersión tipo Rayleigh). La magnitud de la señal de retorno procedente de un volumen difusor puede que dependa, o puede que no, de la orientación angular a diferencia de la orientación especular.

2.5.2. Transductores

Un transductor de ultrasonido genera ondas acústicas, convirtiendo energía magnética, térmica, o eléctrica a energía mecánica. La técnica más efectiva para ultrasonidos médicos usa el efecto piezoeléctrico, que fue demostrado por primera vez por Jacques y Pierre Curie en 1880. Aplicaron una presión a un cristal de cuarzo y detectaron una diferencia de potencial entre las caras opuestas del material. También descubrieron el efecto piezoeléctrico inverso y vieron que aplicando un campo eléctrico a través del cristal, se induce una deformación mecánica en el mismo. De esta forma, tenemos un transductor piezoeléctrico, que convierte una señal eléctrica oscilante en una onda acústica y viceversa. Para usos médicos, el material ferroeléctrico más usado es el titanato-circonato de plomo (PZT).

Transconductores de pistón

El transductor más simple tiene un único elemento con forma de pistón. Suele ser circular y tiene cierta curvatura para enfocar la onda acústica. Este elemento se puede desplazar mecánicamente para recoger la información necesaria para formar una imagen, o se puede mantener fijo en el caso de una señal unidimensional. La ventaja del método mecánico es su simplicidad. Las desventajas son el enfoque fijo y la necesidad del equipo

mecánico. Los transductores de pistón único representan una tecnología ya antigua que no se utiliza en la actualidad. Solamente se pueden encontrar en sondas estáticas doppler que se utilizan en cierta pruebas de cardiología.

Transductor en arrays circular

Está formado por varios anillos piezoeléctricos concéntricos. Estos transductores se pueden enfocar de forma eléctrica, tanto en trasmisión como en recepción, desfasando los pulsos transmitidos por los anillos para la transmisión y mediante técnicas de conformación de haz en recepción. El array se escanea de forma mecánica para formar la imagen. Su desventaja sigue siendo la parte mecánica.

Transconductores en arrays de estado sólido

Es la tecnología que más se utiliza en la actualidad. El número de elementos activos que se utiliza para transmitir y recibir señales enfocadas electrónicamente es bastante elevado (entre 48 y 200). Un *array de fase* tiene una apertura muy pequeña (del orden de 15 mm), por lo que se utiliza para iluminar un único punto. Son adecuados para escanear en ventanas acústicas limitadas, como por ejemplo para exploraciones cardiacas. En ellas el transductor debe escanear a través de una ventana pequeña para evitar las obstrucciones de costillas y pulmones. También se utilizan para examinar zonas profundas del abdomen. Los ultrasonidos no pueden explorar a través de hueso y aire. Un *array lineal* tiene mayores aperturas (típicamente 40 mm o mayores) y mayor cantidad de elementos. Estos transductores se emplean en gran cantidad de visualizaciones abdominales, periféricas y de zonas pequeñas. Finalmente, el *array curvo lineal* tiene dispuestos los elementos activos en una superficie convexa, dando lugar a un campo de vista más ancho.

2.5.3. Consideraciones Económicas

La obtención de imágenes mediante ultrasonidos tiene una serie de ventajas económicas sobre otros métodos de obtención de imágenes médicas como la tomografía computerizada o la resonancia magnética. El sistema es, normalmente, mucho más barato y no es necesaria la preparación específica de las instalaciones, como el aislamiento en los rayos-X y tomografía, o el campo magnético uniforme necesario para realizar una resonancia magnética. La mayoría de los sistemas de ultrasonidos pueden transportarse fácilmente de un sitio a otro, de forma que un mismo sistema puede ser compartido por varias salas de exploración o incluso llevarse a la habitación de algunos enfermos especialmente graves.

Los gastos en cada exploración son mínimos. Básicamente es necesario utilizar un gel para acoplar el transductor a la piel del paciente y una cinta de vídeo o una película para realizar la grabación. Esto hace que sea el sistema preferido para obtener imágenes médicas, siempre que permita obtener unos resultados adecuados. El bajo coste también

permite que estos sistemas puedan encontrarse en clínicas privadas y ser usados sólo ocasionalmente.

2.5.4. Obtención de imágenes por ultrasonidos: el ruido *speckle*

Los tejidos del cuerpo no son homogéneos y las señales que se envían hacia ellos, como por ejemplo pulsos de sonido de alta frecuencia, son reflejadas y dispersadas por esos tejidos. La dispersión de parte de la energía de la señal incidente hacia otras direcciones por pequeñas partículas es la causa de que se vea una especie de niebla en las imágenes obtenidas, conocida como *speckle*. Este *speckle* se manifiesta en la aparición de un moteado aleatorio en la imagen, que oculta los detalles pequeños y dificulta la detección de lesiones bajo contraste. No hay consenso sobre si el propio *speckle* proporciona o no información clínicamente relevante. Muchos médicos son contrarios a las técnicas de reducción de *speckle*. Sin embargo, se están desarrollando muchas técnicas de procesado de imagen para filtrar el *speckle* y mejorar la calidad de las imágenes de ultrasonidos para la detección de lesiones de bajo contraste.

Capítulo 3

Introducción a la lógica borrosa

3.1. Introducción

3.1.1. Origen y Propósito

Si la lógica es la ciencia de los principios formales y normativos del razonamiento, la lógica borrosa se refiere a los principios formales del razonamiento aproximado, con el razonamiento preciso considerado como caso límite. Su origen fue, a partir de 1965, el concepto de subconjunto borroso como intento de superar la rigidez de la teoría clásica de conjuntos para agrupar proposiciones que, por la naturaleza de lo que representan, no presentan un cambio brusco al anteponerles la negación no; es decir, para poder trabajar matemáticamente con clases la pertenencia a las cuales es menos una cuestión de ”sí” o ”no” que una cuestión de grado. Estos *conjuntos* surgen en el conocimiento común al clasificar objetos de un universo conocido que responden a determinada propiedad de manera que no sólo la verifican o no la verifican, sino que la verifican parcialmente en muchos casos; propiedades que se predican, de los objetos en cuestión, en cierto grado.

De hecho, el razonamiento ordinario (y por ende la vida ordinaria) se realiza básicamente sobre tales predicados; es en base a razonamientos *débiles*, por oposición a los *fuertes* de las matemáticas, como se califica a los estudiantes de una clase; como se controla un motor de vapor; como frecuentemente se juzga en los tribunales de justicia; como se determinan las inversiones bursátiles; como se efectúan los diagnósticos médicos; como se avanza en la investigación científica; etc. Y, con todo ello, se suele condicionar el futuro a partir de decisiones de presente, que se toman a partir de conclusiones obtenidas con premisas que reflejan un conocimiento que es inexacto, incompleto y no totalmente fiable.

Lo que es el aspecto central de los sistemas de la lógica borrosa es que, a diferencia de los de la lógica clásica, tienen la capacidad de modelar modos de razonamiento no preciso, que juegan un papel esencial en la notable habilidad humana para tomar decisiones racionales en entornos de incerteza e imprecisión. Tal habilidad depende, a su vez, de

la capacidad humana para inferir respuestas aproximadas a preguntas inexactas, de manera que al modificarse aumentativamente las premisas, se modifiquen, a veces completamente, las conclusiones, lo que marca una diferencia fundamental con el razonamiento deductivo, que conserva las conclusiones al aumentar las premisas. La lógica borrosa analiza los métodos y principios de razonamiento a partir de proposiciones imprecisas que relacionan magnitudes y valores lingüísticos y cualitativos modelados por conjuntos borrosos. El elemento primario de la lógica borrosa es el lenguaje natural, y sus esquemas de razonamiento son esquemas de *razonamiento aproximado* con proposiciones imprecisas, típicamente de carácter lingüístico, como podrían ser las reglas que se obtienen a partir de la expresión lingüística del conocimiento de un operador humano versado en el control de un determinado proceso. En suma, la lógica borrosa, o más adecuadamente la teoría de conjuntos borrosos, encuentra sus aplicaciones más importantes en sistemas complejos que contienen no linealidades en su forma de operar, y que por extensión ha encontrado una de sus aplicaciones más fructíferas en el control de:

- Sistemas demasiado complejos como para ser modelizados con precisión.
- Sistemas con moderadas o significativas no linealidades operacionales.
- Sistemas que tienen incertidumbre tanto en su definición como en sus entradas.

La lógica borrosa permite representar el conocimiento común en un lenguaje matemático especial (el de la teoría de los subconjuntos borrosos y las distribuciones de posibilidad a ellos asociadas) y, a través de un cálculo lógico que flexibiliza la noción operativa de verdad, vista como un predicado singular, permite efectuar inferencias aproximadas a partir de patrones de razonamiento que reproducen aceptablemente los modos usuales de razonamiento que son, mayormente, de tipo lingüístico cualitativo y no necesariamente cuantitativos. Por descontado, con la posibilidad de reinterpretación de los resultados finales obtenidos a través de procesos de naturaleza matemática. Llegar a conclusiones razonables a través de fórmulas es una de las ayudas usuales que nos prestan las matemáticas.

La lógica borrosa está prestando una valiosa ayuda al avance del conocimiento a través, precisamente, de facilitar la escritura de situaciones relacionales que presentan una vaguedad intrínseca. Al respecto de esto, la introducción de [Trillas92] afirma que a lo largo del presente siglo se ha ido haciendo cada vez más claro que no hay una única lógica independiente de todo contenido sino que, en cada dominio, hay una lógica más adecuada, que existe una interdependencia de lo lógico y de lo físico, de lo formal y de lo real. Y, tal vez, una de las más interesantes aportaciones lógicas de la lógica borrosa sea una nueva visión de la noción de proposición no ligada a la idea de Tarski-Wittgenstein de descomposición en su forma canónica, sino ligada a una base relacional de conocimientos; una nueva visión más contingente que necesaria, pero que parece la adecuada a las maneras flexibles del razonamiento común. Y todo ello, en el fondo, como un intento de llegar a analizar la vaguedad, un nuevo mundo a matematizar como lo fuera en su momento el análisis de las figuras planas antes de Euclides, la resistencia de materiales antes

de Galileo, la caída de graves antes de Newton y el azar antes de Laplace y Gauss. Es la línea de siempre; la de que la matemática surge del estudio *de las reglas para escudriñar la naturaleza y llegar a conocer todo lo que existe, todo misterio y todo secreto*, al decir del antiguo Papiro Rhind.

En todo caso (y aunque rastreando en los libros pueden encontrarse antecedentes de estas ideas) la lógica borrosa es muy joven. Si la introducción de la idea de subconjunto borroso por Lotfi A. Zadeh es de 1965, las primeras ideas propiamente de cálculo lógico son de alrededor de 1975 con las primeras aplicaciones al control de procesos. Y no es sorprendente, después de lo dicho, que ya en los años 80 hayan aparecido las primeras realidades del cálculo electrónico borroso. Lo que ya es más sorprendente es que, en tan poco tiempo, tales ideas teóricas hayan dado lugar a una tecnología emergente con interesantes productos en el mercado de gran consumo.Así hoy en día son muchas las aplicaciones tanto industriales como domésticas que hacen uso de este paradigma. Raras veces las mismas personas que originan las nuevas teorías ven los productos tecnológicos generados en base a ellas; desde este punto de vista la lógica borrosa ya es un fenómeno digno de estudio.

3.1.2. Lógica Borrosa y Lenguaje Natural

Una de las formas de representación del conocimiento es utilizar reglas de decisión del tipo; “Si ... (condición) ... entonces ... (decisión)”. Pero generalmente, el conocimiento de un experto (o la información que se obtiene de un proceso) se explica mediante conceptos imprecisos e implica operaciones y decisiones no bien definidas o concretas.

La teoría de conjuntos difusos fue introducida como un mecanismo de representación de la vaguedad e imprecisión de los conceptos empleados en el lenguaje natural. Expresiones del tipo *ese hombre es alto*, o *voy a tardar un rato* son habituales en nuestro lenguaje. Sin embargo no es fácil precisar qué entendemos por *alto* o *un rato*. Los conjuntos borrosos representan en cierta manera, *valores lingüísticos* de una determinada magnitud. Surge así el concepto de *variable lingüística*, como aquella variable que toma valores *lingüísticos*, por ejemplo, altura = “muy alto”, “bajo”, etc, frente a la variable numérica que sólo admite valores concretos, como altura = 1.85, 1.73, etc.

Un ejemplo de cómo nuestro pensamiento es borroso por naturaleza estaría en la premisa *Si hace mucho calor baja un poco la calefacción*. ¿Cuánto es *mucho calor*? ¿20^{o}C, 30^{o}C, 40^o? Y por otra parte ¿cuánto es *un poco*? Sin embargo los seres humanos no encontramos dificultades en razonar con estos conceptos imprecisos, cualquiera de nosotros sabría “procesar” esa instrucción y llevarla a cabo.

Un interesante estudio sobre la relación entre Lógica y Lingüística aparece en [Trillas92] en el capítulo sobre *Lógica y Lingüística* del profesor A. Sobrino.

3.1.3. Aplicaciones de la lógica Borrosa

Algunos de los campos en los que se ha aplicado la lógica borrosa con éxito son:

- Aplicaciones de control: control de aeronaves (Rockwell Corp.), operaciones en el metro de Sendai (Hitachi), control de cruceros (Nissan), transmisión automática (Nissan, Subaru), modelos de coches autoaparcables (Universidad tecnológica de Tokio) y aterrizajes de cápsulas espaciales (NASA).
- Aplicaciones de planificación y optimización: programación de ascensores (Hitachi, Fujitech, Mitsubisi) y análisis del mercado de stocks (Yamaichi Securities).
- Análisis de la señal para sintonizado e interpretación: ajuste de la imagen de la TV (Sony), reconocimiento de escritura manuscrita (Sony Palm Top), autofocus para videocámaras (Sanyo/Fisher, Canon) y estabilizadores de imagen de vídeo (Matshushita/Panasonic).

En [Reyero95] se nos cita también distintos campos de aplicación, como podrían ser el Control de sistemas, el Modelado, Sistemas de Información Inteligente, Reconocimiento de Formas, Tratamiento de Señal, Análisis de decisiones, Sistemas de Vehículos Inteligentes, etc.

3.2. Conceptos Básicos y Terminología

En esta sección se realiza una introducción en la que se glosen los conceptos principales necesarios a la hora de trabajar con sistemas borrosos.

Lógica borrosa Se denomina *lógica borrosa* a los mecanismos de inferencia basados en reglas que emplean términos lingüísticos representados por conjuntos borrosos. Los conjuntos borrosos son capaces de captar por sí mismos la vaguedad lingüística de palabras y frases comúnmente aceptadas. Dado un conjunto de reglas que relacionan una serie de variables y dado un conjunto de valores iniciales (en general expresados como conjuntos borrosos) de algunas de estas variables, el objetivo de estos mecanismos de inferencia es deducir el valor(expresado en forma de conjunto borroso) del resto de las variables.

Crisp Set (*Conjunto Clásico*) Se dice que un conjunto A es un *crisp set* si los elementos de un conjunto universal S o bien pertenecen a A o no pertenecen. Por lo tanto, el término *crisp* (en inglés *crujiente*) se usa en la teoría de conjuntos borrosos para referirnos a los conjuntos clásicos. (Generalmente, se utiliza el término *crisp* como opuesto a *fuzzy* cuando se habla de conjuntos y otros conceptos en los que resulte apropiado).

Fuzzy Set (*Conjunto Borroso*) Se dice que un conjunto A es borroso si sus elementos pertenecen a él con un cierto *grado de pertenencia* dado. El grado de pertenencia es una función cuyos valores están típicamente entre 0 y 1. Cuanto mayor es el valor, mayor es el grado de pertenencia de un elemento a ese conjunto.

Un conjunto *crisp* es, por lo tanto, un conjunto borroso cuya función de pertenencia sólo puede tomar los valores $\{0,1\}$. (Es decir, o pertenece, o no pertenece).

Un ejemplo clásico es el ejemplo de las edades. Queremos dividir a la población entre gente joven y gente que ya no es joven. Podemos decidir que hasta los 20 años una persona es joven. Pero alguien que tiene 20 años y 1 día no se aleja mucho de la que tiene 20 años menos 1 día. Por lo tanto, se puede hacer una división que no sea tan drástica. Usando conjuntos borrosos se podrá decir que hasta 20 años es joven y a partir de 40 ya no es joven.

El grado de pertenencia de un elemento al conjunto *ser joven* es 1 de 0 a 20 años, es 0 para los mayores de 40 años y toma valores intermedios para las edades entre 20 y 40 años.

Por lo tanto, un conjunto borroso definido en un universo U se caracteriza por una *función de pertenencia*, $\mu_F(x)$ que toma valores en el intervalo $[0,1]$. Esta función de pertenencia es la que da una medida del grado de pertenencia anteriormente citado, de un elemento dentro de un conjunto.

$$\mu_F(x) : X \rightarrow [0,1] \tag{3.1}$$

En la lógica borrosa, un elemento puede pertenecer a más de un conjunto. En el ejemplo citado antes, se podría añadir un conjunto que fuera *personas adultas*. Este conjunto toma valor 0 hasta 20 años, valor 1 a partir de 40 y valores intermedios entre 20 y 40. Un elemento h de 28 años pertenecería a los dos conjuntos, con un nivel de pertenencia distinto de 0 y de unos en ambos casos, y a la vez distintos entre sí.

Número Fuzzy Los números borrosos surgen como un intento de introducir vaguedad en los números reales. Un número borroso es un conjunto borroso A definido en una recta real $\Re$. De otra forma se puede decir que es un conjunto borroso cuyos elementos son números reales.

Fuzzyfication y Defuzzification (*borrosificación y desborrosificación*) Procesos que convierten una medida crisp en un conjunto borroso, y proceso que convierten conjuntos borrosos en variables crisp, respectivamente. Se han desarrollado muchas técnicas con este propósito.

Variable Lingüística es una variable cuyos valores son números *borrosos*, el significado de los cuales son conceptos como *pequeño, mediano, grande*. Se denomina *variable lingüística* a una variable cuyos valores pueden expresarse en términos del lenguaje natural. Cada una de estas palabras o términos se conocen como *etiqueta lingüística*

y se representa por medio de un conjunto borroso definido sobre el universo de discurso de la variable. Por ejemplo, la temperatura del cuerpo humano puede ser catalogada como *'baja'*, *'normal'*, *'alta' o 'muy alta'*. Cada uno de estos términos es una etiqueta lingüística que puede definirse como un conjunto borroso

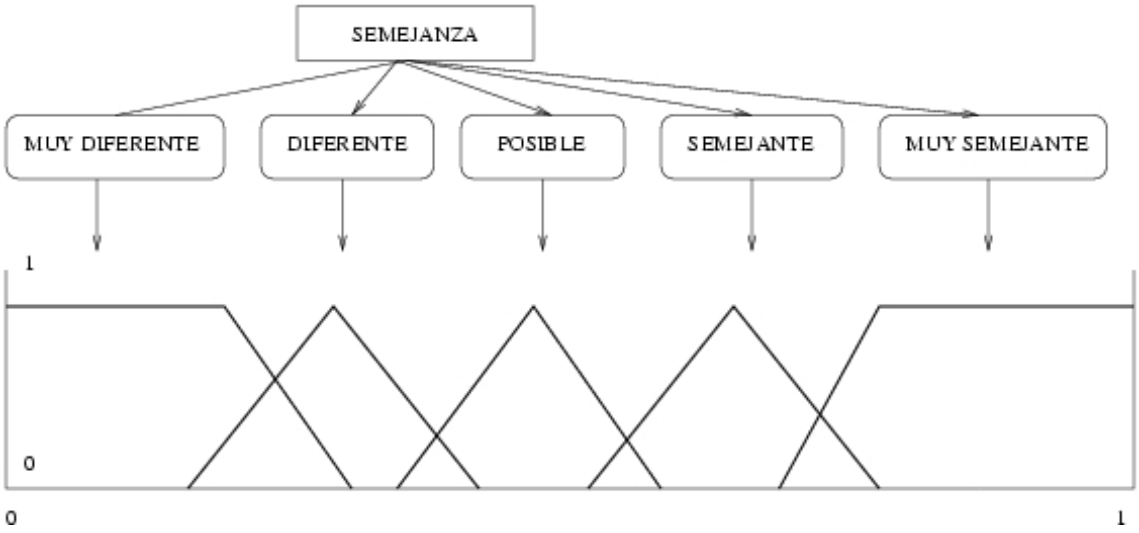


Figura 3.1: El concepto de una variable lingüística (en este caso *semejanza*)

Modificadores Lingüísticos (o *Linguistic Hedges*), son modificadores de los valores de una variable lingüística. Por lo tanto, un *hedge* da lugar a un nuevo conjunto borroso con una función de pertenencia distinta. Ejemplos de estos modificadores son *muy, ligeramente*, etc.

3.3. Diferencia entre Lógica Borrosa y Clásica

Antes de entrar a fondo en la lógica borrosa, se mostrará las principales diferencias que existen entre la lógica borrosa y la lógica clásica.

La lógica borrosa, como su nombre indica, es la lógica que soporta modos de razonamiento que tienen más de aproximados que de exactos. La importancia de la lógica borrosa se deriva del hecho de que la mayoría de los modos del razonamiento humano (y especialmente el razonamiento de sentido común) son aproximados por naturaleza. Es importante notar que, a pesar de su omnipresencia, el razonamiento aproximado sale fuera extensamente del alcance de la lógica clásica porque es una tradición profundamente enraizada en la lógica ocuparse de los modos de razonamiento, y sólo de los modos de razonamiento, a los que se les puede aplicar análisis y formulación precisas.

Algunas de las características esenciales de la lógica borrosa están relacionadas con lo siguiente [Trillas92]:

- En lógica borrosa, el razonamiento exacto se visualiza como un caso límite del razonamiento aproximado.
- En lógica borrosa, todo es cuestión de grado.
- Cualquier sistema lógico puede hacerse borroso.
- En lógica borrosa, el conocimiento se interpreta como un conjunto de acotaciones borrosas elásticas o similar, sobre un conjunto de variables.
- La inferencia se visualiza como la propagación de acotaciones elásticas.

Las diferencias principales entre la lógica borrosa y la lógica clásica son las siguientes:

Verdad: En sistemas lógicos bivalentes, la aserción de verdad puede tomar solamente dos valores: verdadero o falso. En sistemas multivaluados, la verdad de una proposición puede ser un elemento de:

1. Un conjunto finito,
2. un intervalo tal como [0, 1], o
3. un álgebra de Boole.

En lógica borrosa, el valor de verdad de una proposición puede ser un subconjunto borroso de un conjunto parcialmente ordenado, pero generalmente se asume que es un subconjunto del intervalo [0,1] o, más sencillamente un punto en ese intervalo. Los así llamados valores lingüísticos de verdad expresados como verdadero, muy verdadero, no muy verdadero, etc., se interpretan como etiquetas de subconjuntos borrosos del intervalo unidad.

Predicados: En sistemas bivalentes, los predicados son nítidos, por ejemplo: *mortal, par, mayor que*. En lógica borrosa los predicados son borrosos, por ejemplo: *alto, enfermo, pronto, veloz, mucho mayor que*. Nótese que la mayoría de los predicados en lenguaje natural son más frecuentemente borrosos que nítidos.

Modificadores de predicados: En sistemas clásicos, el único modificador de predicados ampliamente usado es la negación, *not*. En lógica fuzzy, existe una variedad de modificadores de predicados que actúa como separadores, por ejemplo: *muy, más o menos, bastante, más bien, extremadamente*. Tales modificaciones de predicados juegan un papel esencial en la generación de valores de una variable lingüística, por ejemplo: *muy joven, no muy joven, más o menos joven*, etc.

Cuantificadores: En los sistemas lógicos clásicos sólo existen dos cuantificadores: *universal y existencias*. La lógica borrosa admite, además, una amplia variedad de cuantificadores borrosos, como, por ejemplo, *pocos, muchos, normalmente, la mayoría de, casi siempre, frecuentemente, alrededor de cinco*, etc. En lógica borrosa un cuantificador borroso se interpreta como un número borroso o una proporción borrosa.

Probabilidades: En los sistemas lógicos clásicos, la probabilidad se evalúa numéricamente o por un intervalo. En lógica *fuzzy*, existe la opción adicional de emplear probabilidades borrosas más generales o lingüísticas, como, por ejemplo, *apropiado, inapropiado, muy apropiado, unos 0,8, alto*, etc. Tales probabilidades se pueden interpretar como números borrosos que pueden ser manipulados mediante aritmética borrosa.

Además de las probabilidades fuzzy, la lógica borrosa hace posible operar con eventos borrosos. Un ejemplo de un evento borroso es: *mañana será un día caluroso*, donde *caluroso* es un predicado borroso. La probabilidad de un evento fuzzy puede ser un número concreto o borroso.

Posibilidades: En contraste con la lógica modal clásica, el concepto de posibilidad en lógica borrosa es gradual en vez de bivalente. Además, como en el caso de las probabilidades, las posibilidades pueden ser tratadas como variables lingüísticas con valores tales como *posible, bastante posible, casi imposible*, etc. Tales valores se pueden interpretar como etiquetas de subconjuntos fuzzy.

Es importante observar que en todos los ejemplos de lógica borrosa se añaden opciones a las que tienen los sistemas lógicos clásicos. En este sentido, la lógica borrosa se puede considerar como una extensión de estos sistemas, en vez de como un sistema de razonamiento que entra en conflicto con los sistemas clásicos.

3.4. Conjuntos Borrosos (Fuzzy Sets)

3.4.1. Introducción. Sistemas Lógicos

El Cálculo Proposicional [Reyero95] trata de las combinaciones de variables lógicas en proposiciones arbitrarias que representan a su vez variables lógicas. Básicamente estas proposiciones representarán unas reglas heurísticas de decisión. Los elementos básicos para la construcción del cálculo proposicional son las *sentencias* y *conectivos lógicos*, mediante los cuales se pueden interrelacionar varias sentencias para constituir otras más complejas.

Una vez construido el Cálculo Proposicional, la verdad o falsedad de una sentencia compuesta puede determinarse a partir de la verdad o falsedad de sus sentencias componentes, siempre que éstas se hayan construido de acuerdo con el cálculo operacional establecido.

Las diferentes lógicas como bases para el razonamiento pueden distinguirse esencialmente por tres características concretas:

- Valores de Verdad.
- Vocabulario (Operadores lógicos, p.e. AND, OR, etc.).

- Procedimiento de Inferencia o Razonamiento.

En Lógica Booleana (Lógica "Clásica" o Lógica Bivaluada), los valores de verdad pueden ser 0 (FALSO) ó 1 (VERDADERO) y los operadores se definen por medio de "tablas de verdad"perfectamente determinadas.

La introducción de nuevos valores de verdad (p.e., tri-valuada, tetra-valuada, n-valuada o incluso valuada continuamente) dio lugar a la aparición en los años 1920 de diversos sistemas de Lógica Multivaluada, pero cuyo desarrollo no pasó de lo puramente académico.

La Lógica Borrosa, que hoy en día se encuentra en constante evolución, nació en los años 1960 como la lógica del razonamiento aproximado, y en ese sentido podía considerarse una extensión de la Lógica Multivaluada. La situación actual es que la Lógica Borrosa está relacionada y fundamentada en una teoría de los Conjuntos Borrosos (teoría de clases cuyos límites no son estrictos), en ese sentido, el grado de pertenencia de un elemento respecto a un conjunto va a venir determinado por una función de pertenencia, que puede tomar todos los valores reales comprendidos en el intervalo [0,1]; no obstante, la mayor parte de las aplicaciones de la Lógica Borrosa contiene conceptos que no son parte de la lógica multivaluada, como: variable lingüística, forma canónica, razonamiento interpolativo, modificadores predicativos, cuantificación borrosa, probabilidades borrosas, etc. Ahora bien, no es nuestro interés describir las múltiples posibilidades que nos ofrece la Teoría de los Conjuntos Borrosos como: los grafos borrosos, la topología borrosa, la programación matemática borrosa, y un largo etcétera que actualmente es motivo de investigación; nosotros nos ceñiremos a los conceptos básicos de la Teoría de Conjuntos Borrosos, para lo que utilizaremos en sus conceptos más elementales ciertas analogías con la Teoría Clásica de Conjuntos.

Los operadores que van a aparecer en Lógica Borrosa (AND, OR, etc.) se definen también usando tablas de verdad, pero mediante un "principio de extensión"por el cuál gran parte del aparato matemático clásico existente puede ser adaptado a la manipulación de los conjuntos borrosos.

Quizás la operación más importante para el desarrollo y creación de Reglas Lógicas es la "Implicación", simbolizada por que representa el "Entonces" de las reglas heurísticas ("Si (...) Entonces($\rightarrow$)(...)"). En la Lógica Borrosa hay muchas maneras por las que puede definirse la implicación. La elección de una "función implicación" que represente analíticamente dicha operación va a reflejar de alguna forma cierto criterio intuitivo.

La última característica de los sistemas lógicos es el procedimiento de razonamiento, que permite inferir resultados lógicos a partir de una serie de antecedentes. Generalmente, el razonamiento lógico se basa en silogismos, en los que los antecedentes son por un lado las proposiciones condicionales (nuestras reglas), y las observaciones presentes por otro (serán las premisas de cada regla).

3.4.2. Revisión de la Teoría Clásica de Conjuntos: Lógica Clásica

Antes de abordar el estudio de la Teoría de Conjuntos Borrosos enunciada por Zadeh, se recordarán algunos de los conceptos básicos de la Teoría Clásica, con el objeto de alcanzar una mayor comprensión de ambas.

Para mayor simplicidad en la exposición, consideraremos que ya son conocidas las nociones de conjunto y elemento.

Conjuntos y Predicados Clásicos

Sea X un universo de discurso del cual cualquier conjunto A es subconjunto, esto es:

$$A \subseteq X, \forall A \tag{3.2}$$

En teoría clásica de conjuntos cualquier elemento x perteneciente a X pertenece o no, pertenece al subconjunto A de manera clara e inequívoca, sin que exista ninguna otra posibilidad al margen de estas dos.

La pertenencia o no de un elemento arbitrario x a un subconjunto A viene dada en la mayoría de los casos por la verificación o no de un predicado que caracteriza a A y da lugar a una bipartición del universo de discurso X.

Por ejemplo, sea X el universo de discurso formado por todos los ríos del mundo. Se define el conjunto A como aquel que está formado por todos aquellos elementos de X que verifiquen el predicado "x fluye por Europa". Por citar algunos ejemplos ilustrativos:

$$\begin{array}{l} "Sena" \in A \\ "Nilo" \notin A \\ "Ebro" \in A \end{array} \tag{3.3}$$

Debemos hacer notar que ha sido posible dar una definición clásica del conjunto A porque su correspondiente predicado permite la bipartición del universo X.

Función de pertenencia

El concepto de pertenencia o no de un elemento a un conjunto A puede expresarse numéricamente mediante la función de pertenencia, también llamada a veces función de verdad o función característica. Esta función asigna a cada elemento x del universo de discurso un dígito binario (1 ó 0) según x pertenezca o no al conjunto A.

$$\mu_A : X \longrightarrow \{0,1\} \;\; \mu_A(X) = \begin{cases} 1 \;\; si \;\; x \in A \\ 0 \;\; si \;\; x \notin A \end{cases} \tag{3.4}$$

Cualquier conjunto A c X se puede definir por los pares que forman cada elemento x del universo y su función de pertenencia, expresándose de la siguiente forma:

$$A = \{(x, \mu_a(x)) / x \in X\} \tag{3.5}$$

Por ejemplo, el conjunto A = {3,4,5,6,7,8,9,10} se puede representar por su función característica

$$\mu_A(X) = \begin{cases} 1 \ \ para \ \ x \in \{3,4,5,6,7,8,9,10\} \\ 0 \ \ de \ \ otro \ \ modo \end{cases} \tag{3.6}$$

La importancia de esta función va a ser especialmente relevante a la hora de estudiar la teoría de conjuntos borrosos.

Operaciones entre Conjuntos

Dados dos conjuntos cualesquiera A y B incluidos en X es posible definir nuevos conjuntos a partir de ellos o, lo que es lo mismo, es posible operar con ellos. A continuación se describen las operaciones básicas entre conjuntos:

1. Intersección: se denota por $A \cap B$ y se define como el conjunto formado por aquellos elementos de X que pertenecen a A y a B simultáneamente:

$$x \in A \cap B \ \ si \ \ x \in A \ \ y \ \ x \in B \tag{3.7}$$

   La función de pertenencia correspondiente es:

$$\mu_{A \cap B}(x) = min(\mu_A(x), \mu_B(x)) \tag{3.8}$$

2. Unión: Es el conjunto formado por aquellos elementos que pertenecen a A, o pertenecen a B, o bien a ambos simultáneamente. Se denota por $A \bigcup B$ y su función de pertenencia es, evidentemente:

$$\mu_{A \bigcup B}(x) = max(\mu_A(x), \mu_B(x)) \tag{3.9}$$

3. Complemento: El complementario de A se denota por $\overline{A}$, y está formado por todos los elementos de X que no pertenecen a A (operador unario).

$$\begin{array}{l} x \in \overline{A} \ \ si \ \ x \notin A \\ \mu_{\overline{A}}(x) = 1 - \mu_A(x) \end{array} \tag{3.10}$$

Propiedades de los conjuntos clásicos

Las operaciones entre conjuntos clásicos presentan ciertas leyes y propiedades:

1. Propiedad conmutativa

$$\begin{array}{l} A \bigcup B = B \bigcup A \\ A \bigcap B = B \bigcap A \end{array} \tag{3.11}$$

2. Propiedad asociativa

$$\begin{array}{l}(A\bigcup B)\bigcup C = A\bigcup(B\bigcup C)\\ (A\bigcap B)\bigcap C = A\bigcap(B\bigcap C)\end{array} \tag{3.12}$$

3. Leyes de idempotencia

$$\begin{array}{l}A\bigcup A = A\\ A\bigcap A = A\end{array} \tag{3.13}$$

4. Leyes de absorción

$$\begin{array}{l}(A\bigcup B)\bigcap A = A\\ (A\bigcap B)\bigcup A = A\end{array} \tag{3.14}$$

5. Propiedad distributiva

$$\begin{array}{l}A\bigcup(B\bigcap C) = (A\bigcup B)\bigcap(A\bigcup C)\\ A\bigcap(B\bigcup C) = (A\bigcap B)\bigcup(A\bigcap C)\end{array} \tag{3.15}$$

6. Propiedades de absorción por S y 0

$$\begin{array}{l}A\bigcup X = X\\ A\bigcap\emptyset = \emptyset\end{array} \tag{3.16}$$

7. Propiedades de identidad

$$\begin{array}{l}A\bigcup\emptyset = A\\ A\bigcap X = A\end{array} \tag{3.17}$$

8. Involución del complemento

$$\overline{\overline{A}} = A \tag{3.18}$$

9. Leyes de Morgan

$$\begin{array}{l}\overline{A\bigcup B} = \overline{A}\bigcap\overline{B}\\ \overline{A\bigcap B} = \overline{A}\bigcup\overline{B}\end{array} \tag{3.19}$$

10. Leyes complementarias

$$\begin{array}{ll}A\bigcup\overline{A} = X & \text{(principio del tercero excluido)}\\ A\bigcap\overline{A} = \emptyset & \text{(principio de no contradicción)}\end{array} \tag{3.20}$$

Lógica Clásica: Operadores y Procedimientos de Inferencia

El razonamiento en Lógica Clásica se realiza según dos esquemas de inferencia conocidos como “modus ponens”(MP) y “modus toliens”(MT). El primero de ellos es el fundamento del procedimiento de encadenamiento hacia adelante, mientras que el segundo se puede identificar con el encadenamiento hacia atrás.

Antecedente:	$\mathcal{X}$ es A	
Regla:	si $\mathcal{X}$ es A,	entonces $\mathcal{Y}$ es B
Consecuente:	$\mathcal{Y}$ es B	

Antecedente:	$\mathcal{Y}$ no es ´B	
Regla:	si $\mathcal{X}$ es A,	entonces $\mathcal{Y}$ es B
Consecuente:	$\mathcal{X}$ no es A	

En "razonamiento clásico"se compara el antecedente, fruto de la observación, con el condicional de la regla. En caso de verificarse éste último se infiere de manera inmediata el correspondiente consecuente. La verificación ha de ser exacta, pues de lo contrario la regla en cuestión no será utilizable y no podremos efectuar razonamiento alguno.

Los operadores y conectivos lógicos quedan definidos mediante tablas de verdad, p.ej.

A	B	AND	OR	$A \rightarrow B$
1	1	1	1	1
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Una realización funcional clásica de estos operadores, a la vista de las tablas, sugiere representaciones del tipo,

$$\begin{array}{l} AND \;\rightarrow\; min(A,B),\;\; A*B,\;\; etc \\ OR \;\rightarrow\; max(A,B),\;\; A+B-A*B\;\; etc, \\ etc... \end{array} \tag{3.21}$$

3.4.3. Extensión a Conjuntos Borrosos

En el apartado (3.4.1) se ha visto cómo la mayoría de las veces los conjuntos clásicos se definen mediante un predicado que da lugar a una clara bipartición del universo de discurso X. Sin embargo, el razonamiento humano utiliza frecuentemente predicados de los cuales no resulta esa bipartición: Son los denominados *predicados vagos*.

Siguiendo con el universo de discurso anterior, el formado por todos los ríos del mundo, se puede definir en él el conjunto B como aquél formado por los ríos "largos".

Por supuesto, es imposible dar a B una definición clásica, ya que su correspondiente predicado no divide el universo X en dos partes claramente diferenciadas. No resulta nada fácil afirmar con rotundidad que un río es "largo" o no lo es. El problema podría resolverse en parte considerando que un río es "largo" cuando su longitud supera cierto umbral fijado de antemano. Decimos que el problema tan sólo se resuelve en parte, y de manera no muy convincente, por dos motivos: de una parte el umbral mencionado se establece de una

manera arbitraria, y por otro lado podría darse el caso de que dos ríos de longitudes muy diferentes fuesen considerados ambos como "largos". Evidentemente, el concepto "largo" así definido nos daría una información muy pobre sobre la longitud del río en cuestión.

La manera más apropiada de dar solución a este problema es considerar que la pertenencia o no pertenencia de un elemento x al conjunto B no es absoluta sino gradual. En definitiva, definiremos B como un conjunto borroso (*fuzzy set*). Su función de pertenencia ya no adoptará valores en el conjunto discreto $\{0,1\}$, sino en el intervalo cerrado $[0,1]$. Desde este nuevo punto de vista resulta obvio que los conjuntos borrosos son una generalización de los conjuntos clásicos.

Mediante notación matemática se define un conjunto borroso A como:

$$A = \{(x, \mu_A(X))/x \in X\} \tag{3.22}$$

Función de Pertenencia

La función característica es remplazada por una *función de pertenencia* que se define

$$\mu_A : X \longrightarrow [0,1] \tag{3.23}$$

de tal modo que $\mu_A(x) \in$ [0,1] es el grado con el que un elemento $x \in$ (siento S el universo del discurso) pertenece al conjunto borroso A. Cuando $mu_A(x) = 0$ el elemento no pertenece al conjunto y cuando $mu_A(x) = 1$ pertenece totalmente. La forma de la función de pertenencia tiene una cierta componente *subjetiva*, frente a la forma rígida (objetiva) de las funciones características de la lógica clásica. En función de la aplicación de los conjuntos o de los conceptos representados por ellos, estas funciones pueden adquirir muy

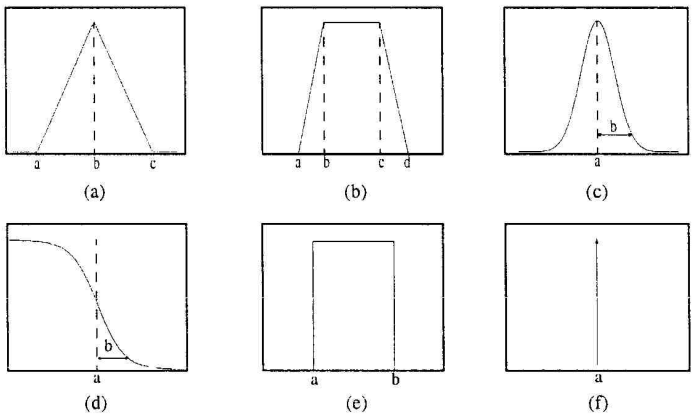


Figura 3.2: Funciones utilizadas habitualmente como funciones de pertenencia y sus parámetros: (a) triangular; (b) trapezoidal; (c) campaniforme; (d) sigmoidal; (e) rectangular; (f) delta.

diversas formas, y muchas veces pueden ser elegidas con un amplio grado de libertad por parte del "diseñador", por lo que en la práctica puede traducirse como la posibilidad de incluir cierto conocimiento experto. Aunque las funciones de pertenencia pueden tener cualquier forma, generalmente y por razones prácticas se suelen definir analíticamente, de manera que sólo sea necesario especificar el valor de alguno de los parámetros para determinar la función. Algunas de las formas más utilizadas son funciones triangulares, trapezoidales, campaniformes o sigmoidales como muestra la Fig. (3.2).

La Fig. (3.3) muestra algunos conjuntos borrosos definidos en el universo de discurso Edad. El conjunto borroso "Joven" representa el grado de pertenencia respecto al parámetro juventud que tendrían los individuos de cada edad. Dicho de otra manera, el conjunto expresa la posibilidad de que un individuo sea caracterizado como "Joven". Un conjunto borroso podría ser considerado una distribución de posibilidad, que es difete a una distribución de probabilidad.

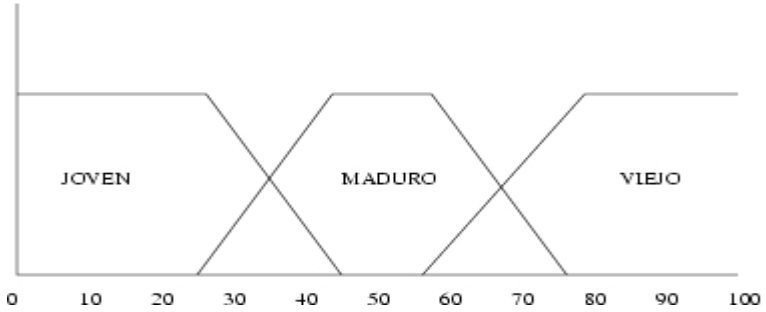


Figura 3.3: Ejemplo de conjuntos borrosos

Se puede ver que los conjuntos borrosos de la Fig. (3.3) se superponen, entonces un individuo x, podría tener un grado de pertenencia en dos conjuntos: "Joven" y "Maduro", indicando que posee cualidades asociadas con ambos conjuntos, el grado de pertenencia de x en A, como ya hemos señalado anteriormente se representa por $\mu_{PA}(x)$. El conjunto borroso A es la unión de los grados de pertenencia para todos los puntos en el universo de discurso X, que también puede expresarse como:

$$A = \int_X \frac{\mu_A(x)}{x} \tag{3.24}$$

Bajo la notación de los conjuntos borrosos $\mu_A(x)/x$ es un elemento del conjunto A. La operación $\int_x$ representa la unión de los elementos borrosos $\mu_A(x)/x$. Los universos de discurso con elementos discretos utilizan los símbolos "$+$" y "$\sum$" para representar la operación unión.

Suele ser conveniente definir un conjunto borroso con la ayuda de alguna fórmula, de tal manera que, por ejemplo, el conjunto "Joven" podría ser:

$$Joven = \int_0^{25} 1 + \int_{25}^{45} (2{,}25 - \frac{x}{20}) \tag{3.25}$$

Construcción de Un Conjunto Borroso

En las asignaciones de *grados de pertenencias*, es decir en la construcción de un conjunto borroso que modelice un predicado vago, se pueden distinguir, claramente, algunos procedimientos útiles:

- Proceso individual o colectivo de asignación directa. Es el caso de evaluaciones curriculares, valoraciones de precios, etc., que pueden basarse en procesos más o menos complicados, pero que no se hacen explícitos en el momento de la asignación. Este tipo de asignaciones han sido estudiadas por Zysno, Norwich y Turksen.
- Procesos estadísticos o probabilísticos. Corresponden a los procedimientos clásicos frecuenciales de distribuciones (*variable aleatoria, función de densidad, función de distribución*, etc.). A partir de las frecuencias de las respuestas dadas por una muestra de una población que opina sobre la idoneidad de los elementos del universo considerado para ser representativos del predicado vago, se va construyendo el conjunto borroso. En algunos casos las técnicas de simulación, la teoría de juegos, las utilidades, etc., pueden ayudar a realizar estas construcciones frecuenciales.
- Procesos de análisis de alternativas. Un caso particularmente interesante es el de la teoría de los procesos jerárquicos analíticos de T.L. Saaty, que constituye un bello ejemplo de cómo establecer un modelo razonable y útil para tratar problemas de toma de decisiones, establecimiento de medidas, control, jerarquización y protocolos. Esta teoría usa como técnicas matemáticas esenciales las de álgebra lineal, valores propios de matrices, grafos, utilidades, etc. El propio Saaty ha propuesto construcciones de conjuntos borrosos contrastando *la representatividad* de los elementos del universo respecto de otros elementos alternativos.
- Procesos de medición directa o indirecta. Basados en magnitudes elementales o derivadas y en general objetivables. Tal es el caso de muchos ejemplos aritméticos o geométricos, donde combinando divisores, lados, diámetros, etc., pueden construirse conjuntos borrosos que representen a predicados vagos tales como *número grande, rectángulo alargado, curva chata*, etc.
- Proceso de Zhang.

Operaciones entre Conjuntos Borrosos

Las tres operaciones básicas definidas sobre los conjuntos clásicos (complemento, introducción y unión) pueden ser generalizadas a los conjuntos borrosos de diversas formas. Dentro de la teoría de los conjuntos borrosos tiene especial relevancia la que hace uso de operaciones conocidas como *operaciones estándar* (Fig. (3.4), definidas como:

$$\begin{array}{lcl} \mu_{A\cap B}(x) & = & min(\mu_A(x), \mu_B(x)) \\ \mu_{A\cup B}(x) & = & max(\mu_A(x), \mu_B(x)) \\ \mu_{\overline{A}}(x) & = & 1-\mu_A(x) \end{array} \tag{3.26}$$

No obstante, al contrario que pasa con los conjuntos clásicos, ésta no es la única forma posible de definir estas operaciones; diferentes funciones pueden ser apropiadas para representarlas en diferentes contextos. Por lo tanto, no sólo las funciones de pertenencia de los conjuntos borrosos van a ser dependientes del contexto sino también de las operaciones sobre dichos conjuntos [Klir95].

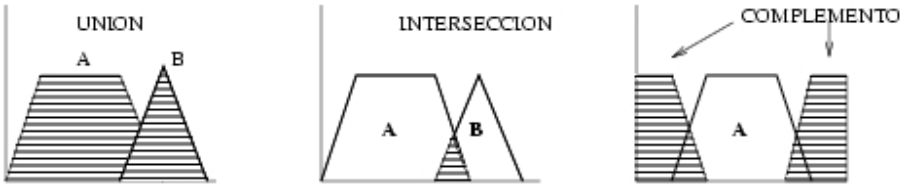


Figura 3.4: Operaciones Básicas entre conjuntos borrosos. Definición estándar

Intersección borrosa: t-norma

Una norma triangular o t-norma es una aplicación $t : [0,1] \times [0,1] \longrightarrow [0,1]$ que verifica las siguientes propiedades:

1. Es no decreciente en cada argumento:

$$\text{si } x \leq y \text{ y } w \leq z \text{ entonces } t(x,w) \leq t(y,z)$$

2. Conmutatividad:

$$t(x,y) = t(y,x), \ \ \forall x,y \in [0,1] \tag{3.27}$$

3. Asociatividad

$$t(t(x,y),z) = t(x,t(y,z)), \ \ \forall x,y,z \in [0,1] \tag{3.28}$$

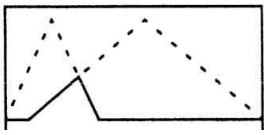
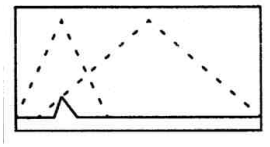
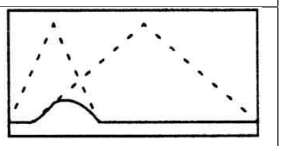
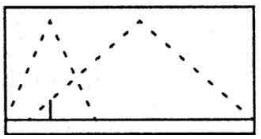
4. Se satisfacen las condiciones de contorno:

$$t(x,0) = 0, \ \ t(x,1) = x, \ \ \forall x,y \in [0,1] \tag{3.29}$$

Las t-normas se utilizan para expresar la intersección entre conjuntos borrosos:

$$\mu_{A \cap B}(x) = t(\mu_A(x), \mu_B(x)) \tag{3.30}$$

Algunos ejemplos de t-normas se recogen en la tabla (3.1)

Nombre	Definición	Representación
Mínimo	$t(x,y) = min(x,y)$	
Producto acotado	$t(x,y) = max(0, x+y-1)$	
Producto	$t(x,y) = x\Delta y$	
Producto drástico	$t_w(x,y) = \begin{cases} min(x,y) & si\ \ max(x,y) = 1 \\ 0 & en\ \ cualquier\ \ otro\ \ caso \end{cases}$	

Cuadro 3.1: T-normas

Unión borrosa: t-conorma

Una conorma triangular, también llamada t-conorma o s-norma, es una aplicación $s : [0,1] \times [0,1] \longrightarrow [0,1]$ que satisface los siguientes requisitos:

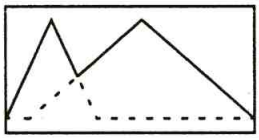
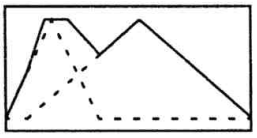
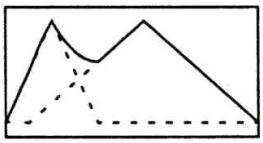
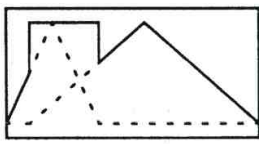
1. s es no decreciente en cada argumento.
2. Conmutatividad.
3. Asociatividad.
4. Condiciones de contorno:

$$s(x,o) = x \;\; s(x,1) = 1 \;\; \forall x \in [0,1] \tag{3.31}$$

Las s-normas expresan normalmente el operador unión:

$$\mu_{A \cup B}(x) = t(\mu_A(x), \mu_B(x)) \tag{3.32}$$

Algunos ejemplos ser recogen en la tabla (3.2)

Nombre	Definición	Representación
Máximo	$s(x,y) = max(x,y)$	
Suma acotada	$s(x,y) = min(1, x+y)$	
Suma algebraica	$s(x,y) = x+y-x\Delta y$	
Suma drástica	$s_w(x,y) = \begin{cases} max(x,y) & si\ \ min(x,y) = 0 \\ 1 & en\ \ cualquier\ \ otro\ \ caso \end{cases}$	

Cuadro 3.2: S-normas

Complemento borroso

Dado un conjunto borroso A, se define el complemento como el conjunto borroso $\overline{A}$ cuya función de pertenencia viene dada por la expresión

$$\mu_{\overline{A}}(x) = C(\mu_{\overline{A}}), \ \ \forall x \in U \tag{3.33}$$

donde $C(x)$ es *C-norma*, y como tal, es un función que cumple las siguientes propiedades:

1. Condiciones de contorno
$$C(0) = 1, C(1) = 0. \tag{3.34}$$
2. Condición de no incremento: para todo $a, b \in [0,1]$, si $a \leq b$, entonces $C(a) \geq C(b)$

Como en los casos anteriores, existen muchas funciones que cumplen estas propiedades y que, por tanto, pueden ser utilizadas para representar el complemento de un conjunto borroso.

Relación entre operaciones borrosas

Dadas una t-norma y una s-norma cualesquiera resulta fácil comprobar que estas funciones están acotadas por las funciones máximo y mínimo:

$$t_w(x,y) \leq t(x,y) \leq min(x,y) \leq max(x,y) \leq s(x,y) \leq s_w(x,y), \ \ \forall x,y \in [0,1] \tag{3.35}$$

Una determinada elección de los operadores unión, intersección y complemento pueden verificar las leyes de Morgan generalizadas:

$$\begin{array}{rcl} C(t(x,y)) & = & s(C(x),C(y)) \\ C(s(x,y)) & = & t(C(x),C(y)) \end{array} \tag{3.36}$$

En este caso se dice que las t-norma y la t-conorma son duales respecto al complemento borroso. En general, dada una función de complemento se puede asociar una t-norma a cada s-norma (y viceversa).

$$s(x,y) = 1 - t(1-x, 1-y) \tag{3.37}$$

$$t(x,y) = 1 - s(1-x, 1-y) \tag{3.38}$$

No todas las t-normas y t-conormas van a ser duales ni van a ser distributivas.

3.4.4. Propiedades de los Conjuntos Borrosos

Las leyes y propiedades que según hemos visto cumplen los conjuntos clásicos no siempre se cumplen en el caso de los conjuntos borrosos. Veamos, analizándolas una por una, qué leyes verifican los conjuntos borrosos y cuáles no:

1. Propiedad conmutativa: siempre se verifica, debido a que las t-normas y las s-normas son conmutativas por definición.

2. Propiedad asociativa: también se verifica puesto que las t-normas y las s-normas también son asociativas.

3. Leyes de idempotencia: se cumplen si elegimos el mínimo y el máximo como operadores para la intersección y la unión respectivamente. Pero si escogemos por ejemplo el producto algebraico y la suma algebraica, ya no se cumplen.

4. Leyes de absorción: también se cumplen si elegimos el par mínimo-máximo. Con otras normas no ocurre lo mismo.

5. Propiedad distributivo: también se cumple para el mínimo y el máximo, pero no así para otras normas.

6. Propiedades del menor y mayor.- siempre se cumplen debido a la última propiedad de las t-normas y s-normas.

7. Involución del complemento: es cierta si definimos $\mu_A(x)$=1 - $\mu_A(x)$, ya que entonces:

$$\mu_{\overline{A}} = 1 - \mu_A(x) = 1 - (1 - \mu_{\overline{A}}(x)) = \mu_{\overline{A}}(x) \tag{3.39}$$

8. Leyes de De Morgan: se garantiza su cumplimiento si la t-norma y s-norma elegidas se derivan la una de la otra. Es decir: t(x,y) = 1 - s(1 - x, 1 - y).

9. Leyes complementarías: en general no se cumplen. Es quizás la consecuencia más clara de introducir el concepto de borrosidad en los conjuntos.

Se puede comprobar fácilmente que en el caso de que los conjuntos sean clásicos (función de pertenencia restringida a 0 ó 1) las diferencias entre las diversas normas desaparecen, convirtiéndose en los operadores intersección y unión clásicos.

Algunos autores críticos de la Teoría de Conjuntos Borrosos achacan a ésta el hecho de que exista arbitrariedad a la hora de elegir los operadores de unión e intersección. No obstante, esto que parece un inconveniente puede ser por otro lado una ventaja, ya que nos permite una gran flexibilidad a la hora de abordar distintos problemas que involucren conceptos vagos. Si queremos que se nos mantengan a toda costa ciertas propiedades de los conjuntos clásicas, podemos elegir a nuestro gusto una t-norma y una s-norma que nos lo permita. Esta elección dará lugar a un tipo u otro de lógica borrosa.

3.4.5. Principio de extensión

Por medio de este principio se propone un camino para calcular los conjuntos borrosos obtenidos por medio de una transformación concreta *crisp* de cierto número (digamos N) de conjuntos borrosos. Específicamente, si X_1,X_2,...,X_N son conjuntos borrosos con funciones de pertenencia $\mu_1(x)$,$\mu_2(x)$,...,$\mu_N(x)$, el nuevo conjunto borroso $Y = f(X_1, X_2, \ldots, X_N)$ tendrá como función de pertenencia:

$$\mu(y) = \sup_{x=f^{(-1)}(y)} [\text{mín}[\mu_1(x_1), \mu_2(x_2), \ldots, \mu_N(x_N)]] \tag{3.40}$$

3.4.6. α-cortes

Existe una manera muy elemental de pasar de conjuntos borrosos a conjuntos clásicos: mediante los llamados α-cortes [Reyero95]. Dado un número $\alpha \in [0,1]$ y un conjunto borroso A, se define el α-corte de A como el conjunto clásico A_α, que tiene la siguiente función de pertenencia:

$$\mu_{A_\alpha}(x) = \begin{cases} 1 & si\ \ \mu_A(x) \leq \alpha \\ 0 & en\ \ cualquier\ \ otro\ \ caso \end{cases} \tag{3.41}$$

En definitiva, el α-corte se compone de aquellos elementos cuyo grado de pertenencia supera o iguala el umbral α.

Se habla de α-cortes estrictos si:

$$\mu_{A_\alpha}(x) = \{ \begin{array}{ll} 1 & si\ \ \mu_A(x) > \alpha \\ 0 & en\ \ cualquier\ \ otro\ \ caso \end{array} \tag{3.42}$$

Cualquier conjunto borroso A se puede representar mediante la unión de sus α-cortes de la siguiente manera:

$$\mu_A(x) = sup_{\alpha\in[0,1]}[\alpha\mu_{A_\alpha}(x)] \tag{3.43}$$

Los α-cortes son de especial utilidad en el estudio de propiedades tales como la reflexividad, simetría y transitividad en conjuntos borrosos. No obstante, este tema se sale del objeto de nuestro trabajo y no nos referiremos a él.

3.4.7. Números Borrosos

Un caso particular y de especial interés de los conjuntos borrosos son los llamados *números borrosos*. Surgen éstos como un intento de introducir vaguedad en los números reales.

Un número borroso es un conjunto borroso A definido en la recta real $\Re$ y que cumple además las siguientes propiedades:

1. Es normal, o lo que es lo mismo, existe al menos un elemento x de $\Re$ tal que $\mu_A(x) = 1$.

2. Es convexo, lo cual quiere decir que

   $$\forall\delta \in [0,1]\ \ \forall a,x \in \Re\mu_A(\delta x + (1-\delta)y) \leq min(\mu_A(x),\mu_A(y)) \tag{3.44}$$

   Geométricamente esta propiedad quiere decir que todos los α-cortes de A son intervalos cerrados en $\Re$. La función de pertenencia es creciente hasta llegar al punto en que $\mu_A(x) = 1$ y decreciente a partir de él.

3. La función de pertenencia es continua a trozos.

4. El soporte de A es acotado.

La enorme importancia de los números borrosos estriba en que son muchos los ejemplos prácticos en los que el grado de pertenencia de un determinado elemento del universo X se puede expresar como función de una característica mensurable del mismo.

Por ejemplo, recordemos que los ríos eran más o menos "largos" según su longitud, característica que se puede expresar perfectamente de una manera cuantitativa.

Como ejemplo visual que ayude a comprender este concepto, presentamos unas posibles definiciones gráficas de los números borrosos que expresan los predicados "aproximadamente 4" y "aproximadamente 6". De un modo arbitrario se han escogido funciones de tipo gaussiano y triangular, respectivamente, que representamos en la Fig. (3.5).

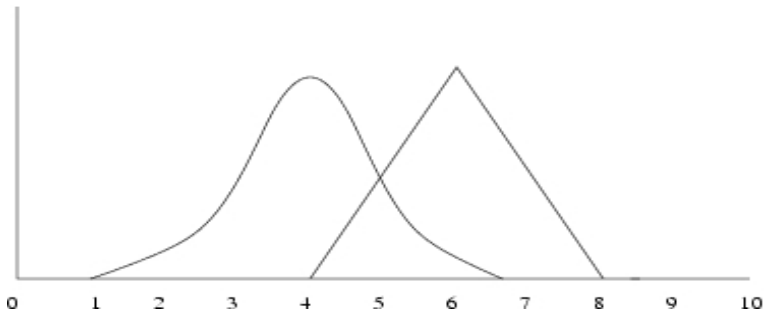


Figura 3.5: Números Borrosos “aproximadamente 4” y “aproximadamente 6”

3.4.8. Modificadores lingüísticos

Un modificador lingüístico o *hedge*, es una operación que modifica el significado de un término, o de manera más general de un conjunto borroso [Klir95]. Por ejemplo, si tenemos un conjunto borroso que es *presión débil*, entonces *presión muy débil*, *presión más o menos débil*, *presión extremadamente débil* son ejemplos de modificadores aplicados a este conjunto. Los modificadores se pueden ver como operadores que actúan sobre la función de pertenencia de un conjunto borroso para modificarla. Algunos de estos operadores pueden ser:

1. Concentración
$$\mu_{CON(U)}(x) = [\mu_U(x)]^{1/2} \tag{3.45}$$
2. Dilatación
$$\mu_{dil(U)}(x) = [\mu_U(x)]^2 \tag{3.46}$$
3. Modificadores artificiales
$$\begin{aligned} \mu_{plus(U)}(x) &= [\mu_U(x)]^{1,25} \\ \mu_{minus(U)}(x) &= [\mu_U(x)]^{0,75} \end{aligned} \tag{3.47}$$

En la Fig. (3.6) se muestra gráficamente el efecto de los dos primeros modificadores sobre el conjunto borroso con función de pertenencia trapezoidal. Además de cambiar la forma de las funciones de pertenencia, algunos modificadores también cambian la posición de los conjuntos sobre el eje de ordenadas; por ejemplo el modificador *muy* aplicado sobre el conjunto GRANDE desplaza a esta hacia la derecha (Fig. (3.7)).

3.5. Sistemas de Lógica Borrosa (Fuzzy Logic Systems)

Un Sistema de Lógica Borrosa es la aplicación de la inferencia borrosa a la automatización de procesos. Un controlador borroso típicamente infiere los consecuentes de

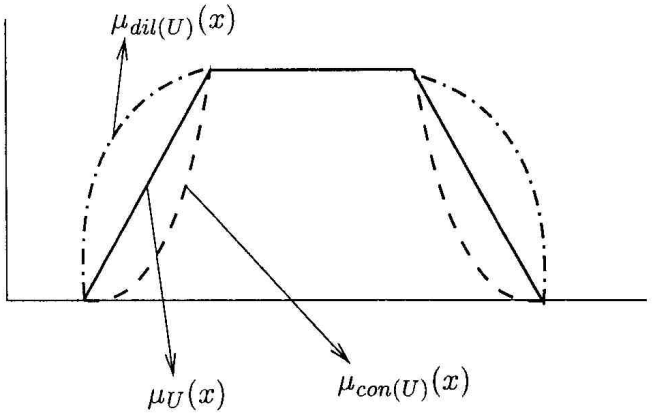


Figura 3.6: Ejemplo de modificadores aplicados sobre un conjunto borroso

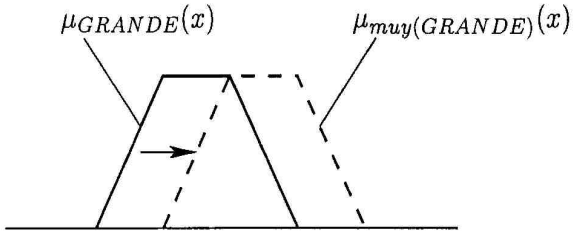


Figura 3.7: Ejemplo de modificadores aplicados sobre un conjunto borroso

un conjunto más o menos grande de reglas simples (base de conocimiento); tal proceso de razonamiento se puede realizar en paralelo, obteniéndose el resultado (consecuente) mediante una sencilla suma lógica. Esta capacidad de procesamiento en paralelo permite que incluso controladores relativamente complejos puedan realizar la inferencia borrosa en un tiempo de cálculo mínimo.

En este capítulo se recogen las principales ideas que subyacen en la aplicación de las técnicas de Lógica Borrosa a sistemas reales.

3.5.1. Elementos

En la Fig. (3.8) aparecen los elementos principales de un Sistema de Lógica Borrosa (FLS). Este esquema es el usado tanto en controladores como en aplicaciones de procesado. Contiene cuatro componentes fundamentales[1]:

Interfase de Borrosificación que realiza un escalado de los valores de las entradas para adecuarlos a los valores típicos para los que se define el sistema, y una "borrosifi-

[1]Estos cuatro componentes son los que aparecen en [Reyero95]. En el texto [Mendel95] las cuatro partes son *reglas*, *fuzzificador, motor de inferencias* y *defuzzyficador*, que en el fondo es lo mismo.

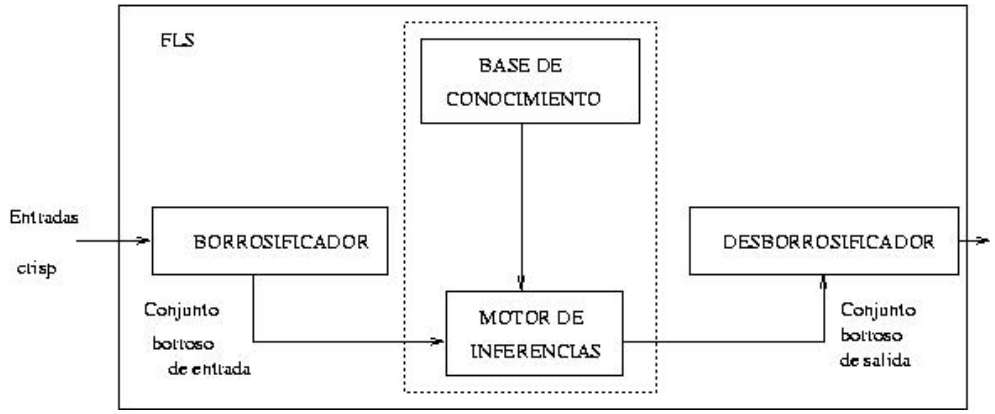


Figura 3.8: Sistema de Lógica Borrosa (FLS)

cación" que convierte los datos de entrada en valores lingüísticos adecuados para la manipulación de éstos como entidades borrosas.

Base de Conocimiento formada por una "base de datos", que recoge la definición de las funciones de pertenencia de las entradas y el sistema, y una "base de reglas", que caracteriza y resume la política y objetivos del control de un experto por medio de un conjunto de reglas lingüísticas de control.

Motor de Inferencias que inferirá las acciones del sistema empleando alguna representación de la implicación borrosa, así como de los procedimientos de inferencia en Lógica Borrosa.

Interfase de Desborrosificación que convertirá la acción "borrosa" actualmente inferida en una acción concreta susceptible de aplicación sobre el proceso, y realizará un escalado para adecuar los rangos de salida para los que se ha definido el sistema con las entradas del proceso.

A continuación analizaremos una por una estas cuatro partes. Para un estudio matemático más detallado, se recomienda la lectura de [Mendel95].

3.5.2. Base de Conocimiento

Base de Reglas

Considérese como ejemplo ilustrativo un problema genérico que puede ser resuelto en términos de razonamiento aproximado por un ser humano experto en la materia, generalmente siguiendo el esquema de *modus ponens generalizado*. Esto quiere decir que el experto es capaz, mediante un conjunto de reglas o pautas de actuación que constituyen su experiencia sobre el tema, elaborar unas conclusiones o consecuentes a partir de unos

hechos observados o antecedentes. El experto podrá transmitir sus conocimientos, al menos parcialmente, mediante un conjunto de N reglas del tipo "if-then", con antecedentes relacionados por el conectivo "AND" en la mayoría de los casos.

La experiencia del ser humano, recogida y almacenada de esta manera, es lo que podemos denominar *base de reglas* (o *base de conocimiento*) por similitud con los métodos de trabajo y la terminología propios de los sistemas expertos.

Existen varios modos de derivación de las reglas de control, siendo este punto fuerte de numerosas investigaciones actualmente. Entre éstos se podrían destacar:

- Basados en la experiencia de un experto, generalmente a través de una verbalización introspectivo de ésta, o a partir de cuestionarios cuidadosamente organizados.
- Basados en las acciones de control de un operador, en función de los datos de entrada-salida observados.
- Basados en un modelo borroso del proceso.
- Basados en aprendizaje (controladores auto-organizados).

Funciones de Pertenencia. Base de Datos

Los conceptos asociados con la Base de Datos se usan para caracterizar las reglas "borrosas" y la manipulación de los datos en un sistema borroso. Estos conceptos son definidos subjetivamente y están basados en la experiencia y juicio de un experto sobre el proceso. Así podríamos citar como algunos de estos aspectos, la cuantificación y normalización de los universos de discurso, número de conjuntos borrosos o categorías lingüísticas de entradas y control, y la elección de las funciones de pertenencia asociadas a estas últimas.

La elección de niveles de cuantificación tiene una influencia esencial en la precisión y "finura" del sistema obtenido. El número de categorías lingüísticas determina la granulidad del sistema final.

Hay dos métodos para definir conjuntos borrosos (mediante sus funciones de pertenencia), dependiendo de si los universos de discurso son discretos o continuos:

- Definición numérica: En el caso de universos discretos, la función de pertenencia de un conjunto borroso se representa como un vector de números cuya dimensión depende del grado de discretización.
- Definición funcional: En el caso de universos continuos, las funciones de pertenencia se expresan por medio de funciones regulares, típicamente triangulares o trapezoidales.

La elección de la forma de las funciones de pertenencia se basa en criterios subjetivos o característicos de cada proceso en cuestión. Por ejemplo, si los datos medibles van a estar afectados por ruidos, las funciones de pertenencia deberían ser suficientemente anchas para reducir la sensibilidad del sistema frente a éstos. Vemos, pues, que la especificación de las funciones de pertenencia afectará a la robustez del sistema.

3.5.3. Motor de Inferencias. Inferencia Borrosa

El motor de inferencias representa el núcleo del FLS, y agrupa toda la lógica de inferencia borrosa del sistema, de barrido de las reglas durante ésta, elección refinada de reglas a utilizar, etc.

La inferencia borrosa es el proceso mediante el cual se obtiene como consecuente un conjunto borroso U a partir de unos antecedentes también borrosos E y DE.

Supongamos ahora que el experto observa los antecedentes "E and DE". El consecuente que se infiere como resultado de esa observación se define como el conjunto borroso que resulta de componer este antecedente con la base de conocimiento o, en otras palabras, se deduce a partir de la regla de inferencia composicional empleando las definiciones de la implicación borrosa y de los conectivos "AND" y "OR".

3.5.4. Borrosificación y Desborrosificación

En el caso de que el conjunto de reglas constituya el núcleo de un controlador para sistemas dinámicos, se nos plantea un problema adicional: las entradas del mismo no son conjuntos borrosos sino valores numéricos concretos provenientes de sensores, mientras que el control inferido representará una magnitud borrosa, frente a los valores concretos de control que necesita el proceso. Por eso es necesario establecer algún tipo de interfaz entre ambos conceptos. Por un lado debemos saber cómo elaborar conjuntos borrosos a partir de las entradas concretas del controlador (borrosificación) y por otro debemos calcular un valor concreto de la señal de control a partir del conjunto borroso obtenido en el proceso de inferencia (desborrosificación [2]).

Borrosificación (Fuzzify)

La primera parte del problema tiene una solución muy sencilla. Si e_o y de_o son los valores concretos de las entradas del controlador en el período de muestreo actual, parece lógico definir los correspondientes conjuntos borrosos de entrada mediante las siguientes funciones de pertenencia:

$$\mu_E(e) = \{ \begin{array}{ll} 1 & si\ \ e = e_0 \\ 0 & en\ \ otro\ \ caso \end{array} \tag{3.48}$$

[2] Estos términos se corresponden a un intento de traducción de los términos ingleses *fuzzify* y *defuzzify*.

$$\mu_{DE}(de) = \{ \begin{array}{ll} 1 & si\ \ de = de_0 \\ 0 & en\ \ otro\ \ caso \end{array} \tag{3.49}$$

Se observa que en el fondo estos conjuntos tienen una definición clásica.

De esta forma, los resultados derivados de la inferencia borrosa del controlador podrán simplificarse y resumiese como:

$$\mu_U(u) = \max_{1 \leq i \leq N} \{min(\alpha_i, \mu_{Ui}(u))\} \tag{3.50}$$

en donde

$$\alpha_i = min\{\mu_{Ei}(e_0), \mu_{DEi}(de_0)\}. \tag{3.51}$$

Desborrosificación (Defuzzify)

Para calcular la señal de control que se debe aplicar al proceso se puede recurrir a distintos métodos. Desafortunadamente, no existe ningún procedimiento sistemático para la elección de la estrategia de desborrosificación, siendo las más comúnmente usadas en la actualidad las siguientes:

1. Método de máxima pertenencia: el control u_0 aplicado es aquél cuyo grado de pertenencia al conjunto U inferido es máximo. Es decir:

   $$\max_{u \in U} \mu_U(u) = \mu_U(u_0) \tag{3.52}$$

   Si existe un número r de controles u^i cuyo grado de pertenencia es máximo, se suele tomar como señal de control el promedio de todos ellos:

   $$u_0 = \sum_{i=1}^{r} \frac{u^i}{r} \tag{3.53}$$

   El método de máxima pertenencia tiene el inconveniente de que da lugar a un controlador multirelay. Es decir, el conjunto posible de controles que puede calcular el controlador no es continuo, sino discreto. Estos posibles controles son aquellos cuyos grados de pertenencia a cada uno de los conjuntos u_i son máximos. Por eso no es un método muy utilizado.

2. Método del centro de gravedad. Es sin duda el más recurrido y eficiente de todos los métodos, ya que proporciona variaciones suaves y continuas de la señal de control. Ésta se calcula como el centro de gravedad de la función de pertenencia del conjunto U. Esto es:

   $$u_0 = \frac{\int u \mu_u(u) du}{\int \mu_u(u) du} \tag{3.54}$$

3. Método del máximo indexado: Este método calcula el centro de gravedad del subconjunto borroso del consecuente inferido, correspondiente a los puntos cuyo valor de pertenencia al consecuente es superior a un umbral dado.

En [Mendel95] y se detalla la descripción de muchos otros métodos de desborrosificación.

3.5.5. Sistemas Borrosos Aditivos (Additive Fuzzy Systems)

Kosco propone lo que él llama *Sistemas Borrosos Aditivos*. A pesar de que propone estos sistemas en general, se centra en un caso particular que llama el *modelo aditivo estándar* (en inglés *standard additive model*, SAM). La Fig. (3.9) muestra el diagrama de bloques de un SAM.

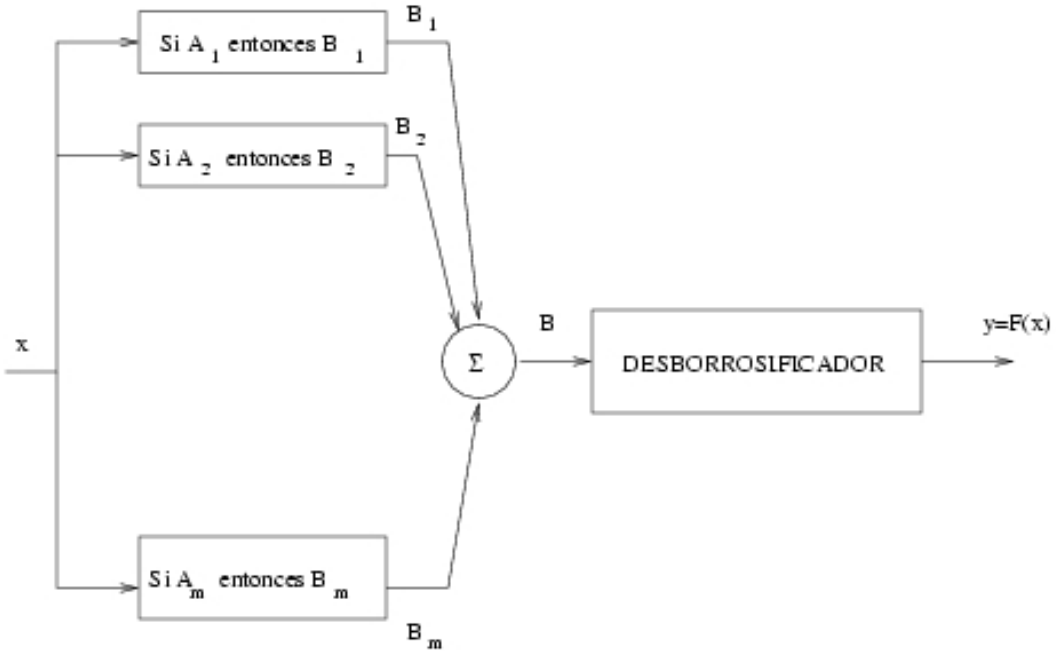


Figura 3.9: Diagrama de bloques de un *modelo aditivo estándar*

El sistema funciona de la manera siguiente: la entrada *crisp* (concreta) x dispara la parte *si* de las reglas A_j hasta un cierto nivel(llamaremos a ese nivel $a_j(x)$). La parte *entonces* de las reglas está escalada de acuerdo a ese nivel y obtenemos un conjunto borroso B'_j. Estos conjuntos borrosos se escalan por un coeficiente ω_j y se suman para obtener el conjunto resultado B. Finalmente, el conjunto borroso se desborrosifica para obtener un valor *crisp* y.

Kosko defiende esta arquitectura alegando distintas razones. Las más importantes son:

- El cálculo computacional que requiere esta arquitectura es pequeña. Se debe al hecho de que algunas de las variables intermedias que se necesitan para calcular la solución final pueden ser precalculadas y almacenadas. Si, por ejemplo, desborrosificamos con el centroide del conjunto borroso resultante, podemos escribir[3]:

$$\begin{aligned} F(x) &= centroide(B) = \frac{\int yb(y)dy}{\int b(y)dy} \\ &= \frac{\int y\sum_{j=1}^{m}\omega_j b'_j(y)dy}{\int \sum_{j=1}^{m}\omega_j b'_j(y)dy} \\ &= \frac{\int y\sum_{j=1}^{m}\omega_j a_j(x)b_j(y)dy}{\int \sum_{j=1}^{m}\omega_j a_j(x)b_j(y)dy} \\ &= \frac{\sum_{j=1}^{m}\omega_j a_j(x)\int yb_j(y)dy}{\sum_{j=1}^{m}\omega_j a_j(x)\int b_j(y)dy} \\ &= \frac{\sum_{j=1}^{m}\omega_j a_j(x)V_j\frac{\int yb_j(y)dy}{V_j}}{\sum_{j=1}^{m}\omega_j a_j(x)V_j} \\ &= \frac{\sum_{j=1}^{m}\omega_j a_j(x)V_j c_j}{\sum_{j=1}^{m}\omega_j a_j(x)V_j} \end{aligned} \tag{3.55}$$

  donde los parámetros V_j y c_j son, respectivamente, el *hipervolumen* encerrado por la función de pertenencia b_j y su centroide.

  Como se puede deducir de la expresión, tanto los volúmenes V_j como los centroides c_j pueden ser precalculados. El cálculo real sólo será de los *niveles de disparo* (firing levels) $a_j(x)$ y el uso de la expresión (3.55).

- Kosko propone diferentes operaciones borrosas a las propuestas por otros autores (por ejemplo, Kir Yuan) sobre conjuntos borrosos:

  1. Los niveles se obtienen directamente de la función de pertenencia de la parte *si* de las reglas en el punto x.

     En el caso de que la entrada sea un vector (es decir, en el caso de que existan varias condiciones *si* en las reglas), Kosko sugiere la multiplicación de los niveles de disparo individuales:

$$a_j(x) = \prod_{i=1}^{n} a_j^i(x_i) \tag{3.56}$$

     como oposición al clásico operador *min*. La razón para esto es que el multiplicador reduce el nivel de proposiciones en los que los *hechos* no coinciden con la parte *si* de la reglas. Sin embargo, el operador *min* no puede realizar esto.

[3] En este caso omitimos la dependencia de $b(y)$ con x, aunque la relación $F(x)$ es obviamente función de x

Si las entradas son borrosas, los niveles se calculan como

$$a_j(A) = \int_{R^n} a(x)a_j(x)dx \tag{3.57}$$

2. Respecto a la combinación aditiva de la parte *entonces* de las reglas, Kosko dice que el operador *max* tiende a crear funciones de pertenencia que se aproximan más a pulsos rectangulares cuanto más reglas se combinan.

   Por otro lado, el operador *suma* tiene la tendencia a crear curvas con forma de campana, lo que sugiere la posibilidad de la existencia de un teorema del límite similar al teorema del límite central.

- Los SAMs son sistemas probabilísticos, ya que calculan la media de la variable perteneciente a la parte *entonces* de las reglas dado un valor x de la variable de entrada.

  Esta afirmación se sostiene con el siguiente razonamiento:

$$\begin{aligned} F(x) &= centroide(B) = \frac{\int yb(x,y)dy}{\int b(x,y)dy} \\ &= \int y f_Y(y|x)dy \\ &= E\{Y|X=x\} \end{aligned} \tag{3.58}$$

  donde vemos que el conjunto de reglas dan lugar a una función densidad de probabilidad condicional porque las cosas funcionan así.

  Si esto es cierto, entonces los SAMs son obviamente estimadores óptimos en el sentido de la mínima media cuadrada. También, el funcionamiento de estos sistemas estará atado claramente a su base de reglas.

Capítulo 4

Realce de imágenes

4.1. Introducción

El realce de imágenes consiste en la aplicación a la imagen bajo estudio de una técnica o conjunto de ellas, cuyo objetivo sea resaltar alguna característica de la imagen que resulte de interés. Es un compendio de diferentes clases de operaciones espaciales destinadas a mejorar el contraste de una imagen, difuminar ciertas regiones de interés y hacer más agudos los bordes y las estructuras útiles [Alberola01]. El principal objetivo de las técnicas de realce de imagen es el de procesarla de forma que el resultado obtenido sea más útil que la imagen original para una aplicación *específica*. Es importante comprender la naturaleza *específica* del realce de imágenes, ya que un método que realza de manera adecuada las imágenes de rayos X, por ejemplo, no es, necesariamente, una opción aceptable para tratar cualquier otro tipo de imágenes.

Las técnicas de realce pueden tener dos posibles destinatarios:

1. Observador humano: el objetivo del realce será hacer la representación de la imagen más apta para la interpretación humana.

2. Máquina (un ordenador o procesador dedicado): el objetivo será preparar a la imagen para su posterior procesado. En este caso, el resultado del realce no tiene por qué ser interpretable por un ser humano.

Tradicionalmente se han englobado las técnicas de realce acorde a las siguiente categorías:

1. Operaciones punto a punto: en las técnicas de este tipo, el valor de la imagen realzada en un píxel es función del valor de su píxel correspondiente (aquel cuyas coordenadas espaciales coinciden con el del original) en la imagen original.

2. Operaciones espaciales: ahora el valor de un píxel en la imagen realzada es función del valor de su pixel correspondiente, y también, de los píxeles vecinos a éste en la imagen original.

3. Operaciones en el dominio transformado: son un caso particular de las técnicas anteriores, pero se suelen englobar en un apartado diferente por claridad expositiva. Estas operaciones consisten en operaciones de realce, no en el dominio espacial original, sino en otros dominios que se consideran más cómodos o útiles para llevar a cabo las operaciones de realce. Los dominios típicamente empleados serán el dominio de la frecuencia (transformada de fourier y otras) y el dominio logarítmico (filtrado homomórfico).

4. Operaciones para imágenes en color: en este caso se incluye la extensión multiespectral de las técnicas anteriores, así como el filtrado pseudocolor.

No hay ninguna teoría general de realce de imágenes ya que no hay ningún estándar general que evalúe la calidad de una imagen y que pueda servir como criterio para el diseño de un algoritmo de realce de imágenes. La validez o no de un método de realce está dada, en la mayoría de los casos, en observaciones humanas y, por lo tanto, es de naturaleza subjetiva [Pratt91].

Además hay que señalar que si el procesado de imagen se caracteriza por un alto contenido heurístico, el realce de imagen en particular es tal vez el máximo exponente de tal heurística.

4.2. Operaciones punto a punto

Las operaciones punto a punto implican la generación de una nueva imagen modificando el valor del píxel en una simple localización de la imagen original. El proceso consiste en obtener el valor del píxel de una localización dada en la imagen, modificándolo por una operación lineal o no lineal y colocando el valor del nuevo píxel en la correspondiente localización de la nueva imagen.

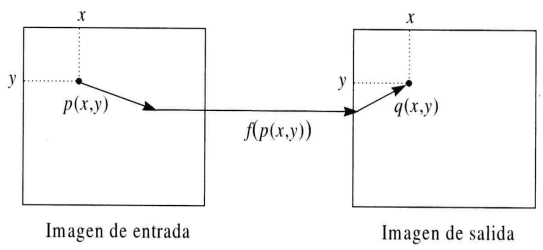


Figura 4.1: Operaciones punto a punto.

Como se muestra en Fig.(4.1) el proceso se repite para todas y cada una de las localizaciones de los píxeles en la imagen original. El operador individual es una transformación uno a uno. El operador T se aplica a cada píxel en la imagen o sección de la imagen y la salida depende únicamente de la magnitud del correspondiente píxel de entrada, siendo la salida independiente de los píxeles adyacentes. La función transforma el valor del nivel de gris de cada píxel y el nuevo valor se obtiene a través de la ecuación:

$$g(x,y) = T[f(x,y)] \tag{4.1}$$

4.2.1. Operaciones de ajuste de intensidad

Negativos de imágenes

El negativo de una imagen digital se obtiene mediante la función de transformación

$$s = T(r) = -(r + L - 1); 0 \leq r \leq L-1; 0 \leq s \leq L-1$$

donde L es el número de niveles de gris, $r = f(x,y)$ y $s = g(x,y)$. La idea que se encuentra en este método es la de cambiar el orden del blanco al negro de forma que la intensidad de los píxeles de la imagen de salida decrezca cuando la intensidad de la entrada aumenta.

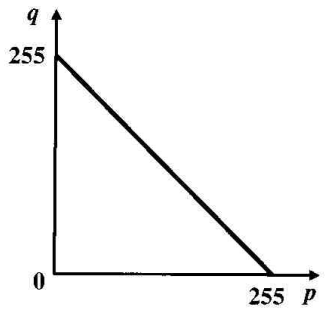


Figura 4.2: Función de trasferencia del operador inverso o negativo de uno imagen.

Aumento de contraste

El objetivo de este método es incrementar el rango dinámico de los niveles de gris de la imagen. La forma de T suele ser de naturaleza no lineal. La forma más usual de esta transformación es la que se muestra en Fig. (4.3). Los puntos (r_1, s_1) y (r_2, s_2) controlan la forma de la transformación. Así, si $r_1 = r_2$, $s_1 = 0$ y $s_2 = L-1$ el resultado de la transformación es una imagen binaria.

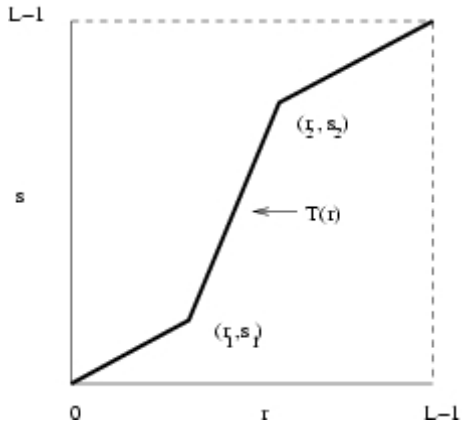


Figura 4.3: Operador T para el aumento de contraste.

Compresión del rango dinámico

Hay ocasiones en las que el rango dinámico de una imagen supera la capacidad de un dispositivo para mostrarla correctamente. De esta forma el dispositivo sólo mostrará correctamente ciertas partes de la imagen. Una manera efectiva para comprimir el rango dinámico de las intensidades de los píxeles de una imagen es realizar la transformación

$$x = c \cdot log(1 + |r|) \tag{4.2}$$

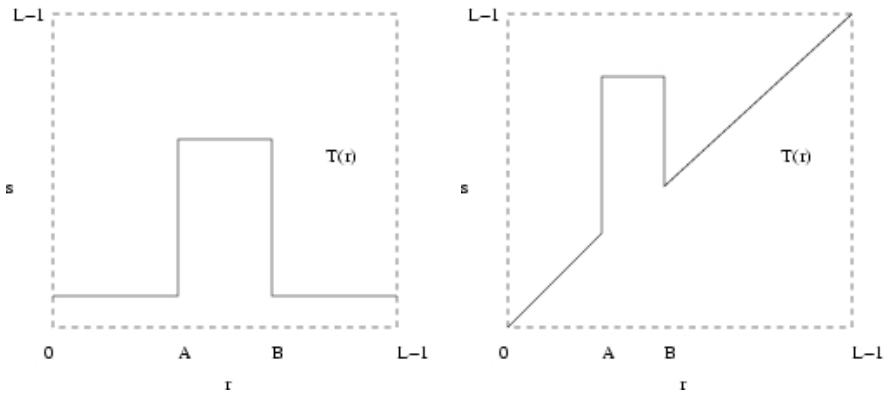
donde c es una constante, y la función logaritmo realiza la compresión deseada.

Corte de niveles de gris (*Gray-level slicing*)

A menudo resulta conveniente resaltar un rango específico de niveles de gris. Existen muchas formas de realizar este realce, pero la mayoría de ellas son variaciones de dos básicas. La primera de ellas asigna un valor elevado a todos los niveles de gris comprendidos en el rango de interés y un valor bajo a todos los niveles de gris restantes. La segunda aumenta considerablemente el valor de los niveles de gris que están dentro de rango deseado, pero deja como están al resto de niveles. La forma de las transformaciones realizadas en ambos casos se puede observar en Fig. (4.4).

Corte en el plano de bits (*Bit-plane slicing*)

Este método, en vez de resaltar rangos de intensidades, resalta la contribución de un bit específico a la imagen total.



(a) (b)

Figura 4.4: Operador T para el corte de niveles de gris: (a) Sin mantener los niveles de gris. (b) Manteniendo los niveles de gris.

4.2.2. Procesado del histograma

El histograma de una imagen es el cálculo muestral de la función de densidad de probabilidad de los valores de intensidad de los píxeles de la imagen. Concretamente, en abcisas se coloca el rango de intensidad de la imagen, digamos, todos los valores enteros desde 0 hasta un valor máximo $L-1$. En ordenadas se calculan cuántos píxeles han tomado cada uno de los valores de abcisas. Por ello, el valor de cada una de las ordenadas será proporcional a la probabilidad muestral de que la imagen tome un determinado valor de la intensidad.

Dada una imagen digital que posee niveles de gris dentro del rango $[0, L-1]$, el histograma es una función discreta $p(r_k) = n_k/n$, donde r_k es el k-ésimo nivel de gris, n_k es el número de píxeles de la imagen que tienen ese nivel de gris, n es el número total de píxeles de la imagen, y $k = 0, 1, 2, \ldots, L-1$. Se puede decir que $p(r_k)$ da una estimación de la probabilidad de que el nivel r_k aparezca en imagen. Un gráfico de esta función para todos los valores de k ofrece una descripción global de la apariencia de la imagen y, además, la forma de este gráfico proporciona información importante sobre las posibilidades de realce que tiene la imagen.

Ecualización del histograma

La igualación o ecualización del histograma consiste en un procedimiento cerrado que trata de convertir el histograma de la imagen original en un histograma plano. La expresión analítica que proporciona la ecualización del histograma es la siguiente:

$$s_k = T(r_k) = \sum_{p=0}^{k} \frac{n_p}{n} = \sum_{p=0}^{k} p_p(r_p) \qquad 0 \leq r_k \leq 1 \text{ y } k = 0, 1, \ldots, L-1 \qquad (4.3)$$

La transformación inversa está dada por:

$$r_k = T^{-1}(s_k) \qquad 0 \leq s_k \leq 1$$

$T(r_k)$ ha de ser monovaluada y monótonamente creciente en el intervalo $0 \leq r_k \leq 1$, y además, debe cumplirse que $0 \leq T(r_k) \leq 1$ para el mismo intervalo de r_k. De igual forma, $T^{-1}(s_k)$ debe satisfacer ambas condiciones. Tras la ecualización del histograma de una imagen se consigue un incremento en el rango dinámico de los niveles de gris de los píxeles y, consecuentemente, un incremento en el contraste de la imagen. La ventaja que ofrece este método con respecto al *aumento de contraste* es que es un proceso totalmente automático.

Especificación del histograma

En algunas ocasiones puede resultar útil poder especificar una forma particular de histograma con la que se resalten ciertos rangos de niveles de gris de la imagen.

Sin embargo, puede que en ocasiones se necesiten realzar diversos detalles en zonas de la imagen pequeñas, los cuales no se verían realzados mediante un procesamiento global de la imagen. Para conseguirlo se definen ventanas cuadradas o rectangulares en torno a un píxel y se va moviendo el centro de la ventana hasta recorrer toda la imagen. Para cada ventana se calcula su histograma y, sobre este histograma, se pueden aplicar las técnicas de *ecualización* o *especificación* para modificar el valor del píxel central de la ventana. Este *realce local* realizado, puede estar basado en otras propiedades que no sean el valor de los niveles de gris de los píxeles que forman la ventana.

4.2.3. Resta de imágenes

La diferencia entre dos imágenes, $f(x,y)$ y $h(x,y)$, expresada como

$$g(x,y) = f(x,y) - h(x,y) \tag{4.4}$$

se obtiene calculando la diferencia entre todos los pares correspondientes de f y h. Cuando las imágenes proceden de una misma escena, tomadas desde el mismo punto, pero en instantes de tiempo distintos, entonces la imagen diferencia realzará las diferencias que hayan podido surgir entre las dos imágenes. Naturalmente, estas diferencias procederán de la entrada en escena de nuevos agentes, o del movimiento de estructuras que ya se encontraban en la escena. El empleo de la diferenciación en el caso de imagen médica suele ser para esto último, es decir, para realzar el movimiento de entidades previamente iluminadas. Un ejemplo muy típico es el seguimiento de un contraste en un vaso sanguíneo.

4.2.4. Promediado de imágenes

Supóngase que hay una imagen $g(x,y)$ formada por la adición de ruido $\eta(x,y)$ a la imagen original $f(x,y)$ de forma que:

$$g(x,y) = f(x,y) + \eta(x,y) \tag{4.5}$$

El objetivo del procedimiento es reducir los efectos del ruido añadiendo un conjunto de imágenes ruidosas, $\{g_i(x,y)\}$. Se supone que el ruido es de media cero e incorrelado. Considérese la función $\overline{g}(x,y)$, formada por la suma de M imágenes ruidosas:

$$\overline{g}(x,y) = \frac{1}{M}\sum_{i=1}^{M} g_i(x,y) \tag{4.6}$$

Parece claro a partir de la ecuación (4.6) que se cumple que

$$E\{\overline{g}(x,y)\} = f(x,y) \tag{4.7}$$

y

$$\sigma^2_{\overline{g}(x,y)} = \frac{1}{M}\sigma^2_{\eta(x,y)} \tag{4.8}$$

donde $E\{\overline{g}(x,y)\}$ es el valor esperado de $\overline{g}$, y $\sigma^2_{\overline{g}(x,y)}$ y $\sigma^2_{\eta(x,y)}$ son las varianzas de $\overline{g}$ y η, todo en las coordenadas (x,y). La desviación estándar en cualquier punto de la imagen promediada es

$$\sigma_{\overline{g}(x,y)} = \frac{1}{\sqrt{M}}\sigma_{\eta(x,y)} \tag{4.9}$$

Las ecuaciones (4.8) y (4.9) indican que a medida que M crece, la variabilidad del valor de los píxeles en cada punto (x,y) decrece. Por otra parte, la ecuación (4.7) quiere decir que $\overline{g}(x,y)$ se aproxima a $f(x,y)$ a medida que el número de imágenes ruidosas usadas en el promediado aumenta.

Debe tenerse en cuenta, para llevar a cabo esta operación, que las diferentes imágenes a promediar deben estar perfectamente alineadas. De no ser así, la operación de promediado seguiría reduciendo el efecto de ruido pero a costa de difuminar las aristas presentes en la imagen. El efecto neto sería una cierta sensación de desenfoque, tanto mayor cuando mayor fuese el número de imágenes promediadas, y cuanto más severo fuese el desenfoque.

4.3. Operaciones espaciales

En esta categoría se engloban las técnicas de tipo *convolutivo*, léase, las técnicas en las que el valor de un píxel en la imagen realzada es función del valor de su píxel correspondiente en la imagen original, y también, de los píxeles vecinos a éste en la imagen

original. Los operadores espaciales se suelen llevar a la práctica a través de máscaras de convolución.

Las funciones empleadas se pueden expresar como

$$v = T(u, u_1, u_2, u_3, ..., u_{p^2-1)} \tag{4.10}$$

donde u y v son el valor original y transformado respectivamente del nivel de intensidad de un cierto pixel, y $(u_1, \ldots, u_{p^2-1})$ son los valores de intensidad de los píxeles *vecinos* del píxel bajo estudio. Estos píxeles definen subimágenes centradas en (x,y). Así, el centro de la subimagen va variando hasta recorrer toda la imagen. Es práctica habitual resumir estos operadores mediante una máscara de valores, la cual consiste en una rejilla de *PxP* píxeles, típicamente con *P* impar y centrada en torno al píxel bajo estudio, en la que cada casilla tiene el coeficiente por el que se multiplicará a cada píxel que intervenga en tal operación. De esta forma, con sólo una inspección de la máscara se puede interpretar rápidamente en qué consiste el operador en cuestión.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figura 4.5: Máscara de tamaño 3×3.

De esta forma si el centro de la máscara está situado en el punto (x,y) de la imagen, el nivel de gris del píxel localizado en (x,y) será reemplazado por v como muestra la ecuación (4.11). El proceso se repetirá hasta que se haya cubierto por completo toda la imagen.

$$v = w_1u_1 + w_2u_2 + \ldots + u_9z_9 \tag{4.11}$$

Normalmente la forma de estas subimágenes es cuadrada o rectangular. Sin embargo ésta no es la única elección posible, sino que nos podemos encontrar con subimágenes en forma de cruz, aproximadamente circulares, ... Los *filtros espaciales* no lineales también operan sobre ventanas, pero no utilizan máscaras sino que sus operaciones se basan directamente en los valores de los píxeles de la ventana.

Dentro de la categoría de operaciones espaciales o de vecindad, se incluyen las operaciones de filtrado. Las operaciones de filtrado tienen la peculiaridad de eliminar un determinado rango de frecuencias de las imágenes. Además de éstas, se incluyen dentro de las operaciones de vecindad, cualquier otra en la que participen los píxeles vecinos del que se está transformando realmente. Si analizamos estas operaciones espaciales desde el punto de vista de su funcionalidad, distinguiremos entre filtros de suavizado (*smoothing*

filters) y filtros que resaltan bordes (*sharpening filters*), estando los primeros encaminados a mejorar la calidad de la imagen, mientras que los segundos permiten extraer los bordes subyacentes.

4.3.1. Filtros de suavizado (Smoothing filters)

A continuación se presentan varios filtros de suavizado, el primero de ellos es lineal y los otros dos no lineales.

Filtro espacial paso bajo

La forma de la respuesta al impulso necesaria para implementar un filtro espacial paso bajo es de sobra conocida e indica que el filtro ha de tener todos los coeficientes de la máscara positivos. Aunque esta forma puede ser modelada, por ejemplo, mediante una gaussiana, la forma más simple de realizar un filtro de este tipo es utilizar una máscara de tamaño 3×3 en la que todos sus coeficientes tienen valor uno. Sin embargo, esto llevaría a que el resultado *v* pudiese estar fuera del rango válido de niveles de gris. La solución es dividir este resultado por nueve (Fig. (4.6)).

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figura 4.6: Máscara de tamaño 3×3 que produce una respuesta paso bajo.

Las diferentes implementaciones darán más o menos importancia al píxel central respecto de los vecinos, pero el comportamiento global es similar. Es práctica habitual que los coeficientes se normalicen, de forma que su suma sea igual a uno. De este modo, la ganancia del filtro para frecuencias espaciales nulas es unitaria, de forma que la componente continua de la imagen no se ve alterada. Si se utilizasen máscaras de mayor tamaño la filosofía de diseño sería la misma.

El filtro paso bajo es un filtro de promediado. Este tipo de filtrado tiene como objeto reducir el ruido de adquisición. El inconveniente de esta técnica es el suavizado de las transiciones (debido al recorte de componentes espectrales de alta frecuencia) de forma que el aspecto final de la imagen será de un cierto desenfoque, tanto mayor cuanto mayor sea el filtro.

Filtro de mediana

Se trata de un operador espacial no lineal. Es un filtro de *estadísticos ordenados*, en el que el papel que juega la máscara simplemente es el de extraer los valores de intensidad de los píxeles que forman parte de la ordenación. Este filtro remplaza el nivel de gris de un píxel por el valor mediano de los niveles de gris de los píxeles de la ventana, en vez de la media como ocurre en el caso anterior. El valor mediano de un conjunto de valores es aquel que cumple que la mitad de los elementos del conjunto son mayores que él y la otra mitad menores.

Concretamente, asumiendo que las máscara es de *PxP* elementos, el funcionamiento del floro sería:

1. Se extraen los P^2-1 valores de intensidad alrededor del píxel bajo estudio así como el valor de intensidad del píxel bajo análisis.

2. Se ordenan los valores de intensidad.

3. El valor se salida que toma el filtro es el valor de intensidad situado en la posición $\frac{p^2+1}{2}$.

Este filtro no oscurece los bordes ni los detalles finos de la imagen tan fuerte como el filtro de suavizado. Los valores medianos se emplean en estadística para evitar la influencia de valores muy atípicos (*outliers*). Por tanto, este tipo de filtrado es muy apropiado para evitar ruido impulsivo.

Filtro de pseudomediana

La carga computacional del filtro de mediana crece exponencialmente con el tamaño de la ventana. *Pratt et al* propusieron un operador computacionalmente más simple (*filtro de pseudomediana*) que posee muchas de las propiedades del filtro de mediana [Pratt91]. El valor mediano de una secuencia de cinco elementos a,b,c,d,e se puede expresar como:

$$\begin{aligned} med(a,b,c,d,e) = max[&min(a,b,c), min(a,b,d), min(a,b,e),\\ &min(a,c,d), min(a,c,e), min(a,d,e), min(b,c,d),\\ &min(b,c,e), min(b,d,e), min(c,d,e)]\\ med(a,b,c,d,e) = min[&max(a,b,c), max(a,b,d), max(a,b,e),\\ &max(a,c,d), max(a,c,e), max(a,d,e), max(b,c,d),\\ &max(b,c,e), max(b,d,e), max(c,d,e)] \end{aligned} \tag{4.12}$$

En general la mediana de una secuencia de L elementos es el máximo (o mínimo) de los mínimos (o máximos) de todas las $L!/[M!][(L-M)!]$ subsecuencias donde $M=(L+1)/2$

y $L! = L(L-1)(L-2)\ldots$. La pseudomediana de cinco elementos se define como:

$$pmed(a,b,c,d,e) = (1/2)max[min(a,b,c),min(b,c,d),min(c,d,e)] + (1/2)min[max(a,b,c),max(b,c,d),max(c,d,e)] \quad (4.13)$$

Se puede observar que, en vez de utilizar diez subsecuencias de números como se hace en las ecuaciones (4.12), sólo se usan tres de ellas. La primera parte de la ecuación (4.13) da como resultado el valor mediano o un valor inferior a éste, mientras que la segunda parte de la ecuación produce el valor mediano a un valor superior. Al realizar la media de estos dos números se consigue el valor mediano o bien un valor que se le aproxima más que cualquiera de las dos resultados parciales. Un análisis de las ciento veinte disposiciones posibles de cinco elementos indica que sólo en ocho casos la pseudomediana es igual a la mediana. Sin embargo, en ninguna disposición en valor de la pseudomediana es extremo; en la mayoría de los casos la pseudomediana es la media de dos valores cercanos a la mediana.

Se puede generalizar la definición (4.13) para una secuencia $\{S_L\}$ de elementos $s_1, s_2, \ldots, s_L$. De esta forma la pseudomediana de la secuencia $\{S_L\}$ viene dada por:

$$pmed\{S_L\} = (1/2)maximin\{S_L\} + (1/2)minimax\{S_L\} \quad (4.14)$$

Donde para $M = (L+1)/2$

$$\begin{aligned} maximin\{S_L\} &= max\{[min(s_1,\ldots,s_M)],[min(s_2,\ldots,s_{M+1}], \\ &\quad \ldots,[min(s_{L-M+1},\ldots,s_L]\} \\ minimax\{S_L\} &= min\{[max(s_1,\ldots,s_M)],[max(s_2,\ldots,s_{M+1}], \\ &\quad \ldots,[max(s_{L-M+1},\ldots,s_L]\} \end{aligned} \quad (4.15)$$

Este concepto unidimensional de pseudomediana puede extenderse de varias formas. Una de ellas tiene que ver con el uso de ventanas en el procesado de una imagen. De igual forma que el filtro de medianas tiende a suavizar al imagen. Considérese una ventana de la siguiente forma:

$$\begin{array}{ccc} & y_1 & \\ & \vdots & \\ x_1 \cdots & x_M & \cdots x_c \\ & \vdots & \\ & y_R & \end{array}$$

Las secuencias $\{X_c\}$ y $\{Y_R\}$ denotan los elementos de los ejes horizontal y vertical de la ventana respectivamente. Nótese que el punto x_M pertenece a ambas secuencias. Entonces, la pseudomediana se define:

$$\begin{aligned} pmed = &(1/2)max[maximin\{X_c\}, maximin\{Y_R\}] + \\ &+(1/2)min[minimax\{X_c\}, minimax\{Y_R\}] \end{aligned} \quad (4.16)$$

4.3.2. Filtros de afilado (Sharpening filters)

El principal objetivo del afilado es destacar detalles finos de una imagen o realzar detalles que han sido desenfocados por culpa de un error o debido a un efecto natural del método de adquisición de la imagen. A continuación se muestra una serie de filtros espaciales de este tipo.

Filtro espacial paso alto básico

La forma típica de la respuesta al impulso necesaria para implementar un filtro paso alto indica que la máscara debe tener coeficientes positivos cerca del centro, y coeficientes negativos a medida que se aleja de él. Por ejemplo, para una máscara de tamaño 3×3, eligiendo un coeficiente positivo para el punto central y tomando el resto de coeficientes valores negativos, se cumpliría la condición. Supóngase que los coeficientes de una máscara de 3×3 valen 8/9 en el punto central y -1/9 en el resto de los puntos (Fig. (4.7)). Se puede observar que la suma de todos los coeficientes de la máscara es cero. De esta forma, si la máscara está recorriendo un área constante o con muy poca variación de niveles de gris, la salida que producirá será cero o un valor muy pequeño. Es decir, este filtro elimina los términos de baja frecuencia. Como consecuencia se reduce el valor medio de los niveles de gris de la imagen a cero y reduce significativamente el contraste global de la imagen. El hecho de que la media de los valores de la imagen sea cero indica que la imagen ha de contener niveles de gris negativos. Como normalmente los niveles de gris son valores positivos, se deben modificar los niveles resultantes del filtrado paso alto para que al final del proceso estén dentro del intervalo $[0, L-1]$.

$\frac{1}{9} \times$

−1	−1	−1
−1	8	−1
−1	−1	−1

Figura 4.7: Máscara de tamaño 3×3 que produce una respuesta paso alto.

Filtro de empujón alto (*Unsharp masking*)

Una imagen resultante de un filtrado paso alto puede ser calculada como la diferencia entre la imagen original y una versión de esta imagen que ha pasado por un filtro paso bajo; es decir;

$$\text{Paso alto} = \text{Original} - \text{Paso bajo} \tag{4.17}$$

Multiplicando la imagen original por un factor de amplificación (A), se obtiene la definición del filtro *unsharp masking*:

$$\begin{aligned} \text{High boost} &= (A)(\text{Original}) - \text{Paso bajo} \\ &= (A-1)(\text{Original}) + \text{Original} - \text{Paso bajo} \\ &= (A-1)(\text{Original}) + \text{Paso alto} \end{aligned} \tag{4.18}$$

Si A toma el valor de uno, como resultado se obtiene un filtro paso alto normal. En el caso de $A > 1$, parte del propio original se añade al resultado del filtro paso alto, lo que devuelve parcialmente las componentes de frecuencias bajas perdidas en el proceso del filtrado paso alto. El resultado es que la imagen resultante del filtrado se parece más a la imagen original, con un grado relativo de mejora de los bordes que depende del valor de A. Por tanto, si la constante A es mayor que 2, el filtro *unsharp masking* equivale a añadir a la versión original amplificada de la imagen una parte de la componente paso alto de la misma. En este caso se conseguirá enfatizar las transiciones sin perder información de la estructura original de la imagen.

Filtros diferenciales

El promediado de los píxeles de una región tiende a difuminar la imagen. Como la integración es análoga al promediado, parece lógico esperar que la diferenciación produzca un aumento de nitidez de la imagen.

El método más común de diferenciación en las aplicaciones de procesado de imagen es el *gradiente*. Para una función $f(x,y)$, el gradiente de f en las coordenadas (x,y) se define como el vector

$$\Delta \mathbf{f} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{4.19}$$

El módulo de este vector,

$$\Delta f = mag(\Delta \mathbf{f}) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right] \tag{4.20}$$

es la base de las aproximaciones a la diferenciación de la imagen. Considérese la región de una imagen mostrada en Fig. (4.8(a)), donde los diferentes subíndices de la variable z indican los diferentes valores de los niveles de gris. La ecuación (4.20) puede aproximarse en torno al punto z_5 de distintas formas. La más simple de ellas es emplear la diferencia $(z_5 - z_8)$ en la dirección x y la $(z_5 - z_6)$ en la dirección y, combinándolas de la forma

$$\Delta f \approx [(z_5 - z_8)^2 + (z_5 - z_6)^2]^{1/2} \tag{4.21}$$

En vez de utilizar cuadrados y raíces cuadradas, es posible obtener resultados similares empleando valores absolutos:

$$\Delta f \approx |(z_5 - z_8)| + |(z_5 - z_6)| \tag{4.22}$$

Otra forma de aproximar la ecuación (4.20) es utilizar diferencias cruzadas:

$$\Delta f \approx [(z_5 - z_9)^2 + (z_6 - z_8)^2]^{1/2} \tag{4.23}$$

o bien, de valores absolutos

$$\Delta f \approx |(z_5 - z_9)| + |(z_6 - z_8)| \tag{4.24}$$

Tanto las ecuaciones (4.21), (4.22), (4.23), (4.24) pueden implementarse utilizando máscaras de tamaño 2×2, como las que se muestran en la Fig. (4.8(b)). Estas máscaras reciben el nombre de operadores de *Roberts*.

Otra aproximación, usando máscaras de tamaño 3×3, es

$$\Delta f \approx |(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)| + |(z_3 + z_6 + z_9) - (z_1 + z_3 + z_7)| \tag{4.25}$$

Para implementar esta ecuación se pueden usar las máscaras definidas en Fig. (4.8 (c)), conocidas con el nombre de *operadores de Prewitt*. Finalmente, la Fig. (4.8 (d)) muestra todavía otro par de máscaras (denominadas *operadores de Sobel*) que pueden emplearse para aproximar el módulo del gradiente.

4.4. Operaciones en el dominio transformado

Estas operaciones no son nada nuevo con respecto a lo visto en las secciones anteriores, sino simplemente una forma de operar en dominios alternativos al espacial original. La idea en la que se basan las técnicas de realce de imágenes en el dominio transformado es el teorema de convolución. Sea $g(x,y)$ una imagen formada por la convolución de una imagen $f(x,y)$ y un operador lineal invariante de posición $h(x,y)$[1]:

$$g(x,y) = h(x,y) * f(x,y) \tag{4.26}$$

Entonces se cumple la siguiente relación en el dominio de la frecuencia:

$$G(u,v) = H(u,v)F(u,v) \tag{4.27}$$

donde G, H y F son respectivamente las transformadas de Fourier de g, h y f. En la terminología que se usa en la teoría de sistemas lineales, la transformación $H(u,v)$ se denomina

[1]Un operador invariante de posición es aquel cuyo resultado depende sólo del valor $f(x,y)$ en un punto de la imagen y no de la posición del punto.

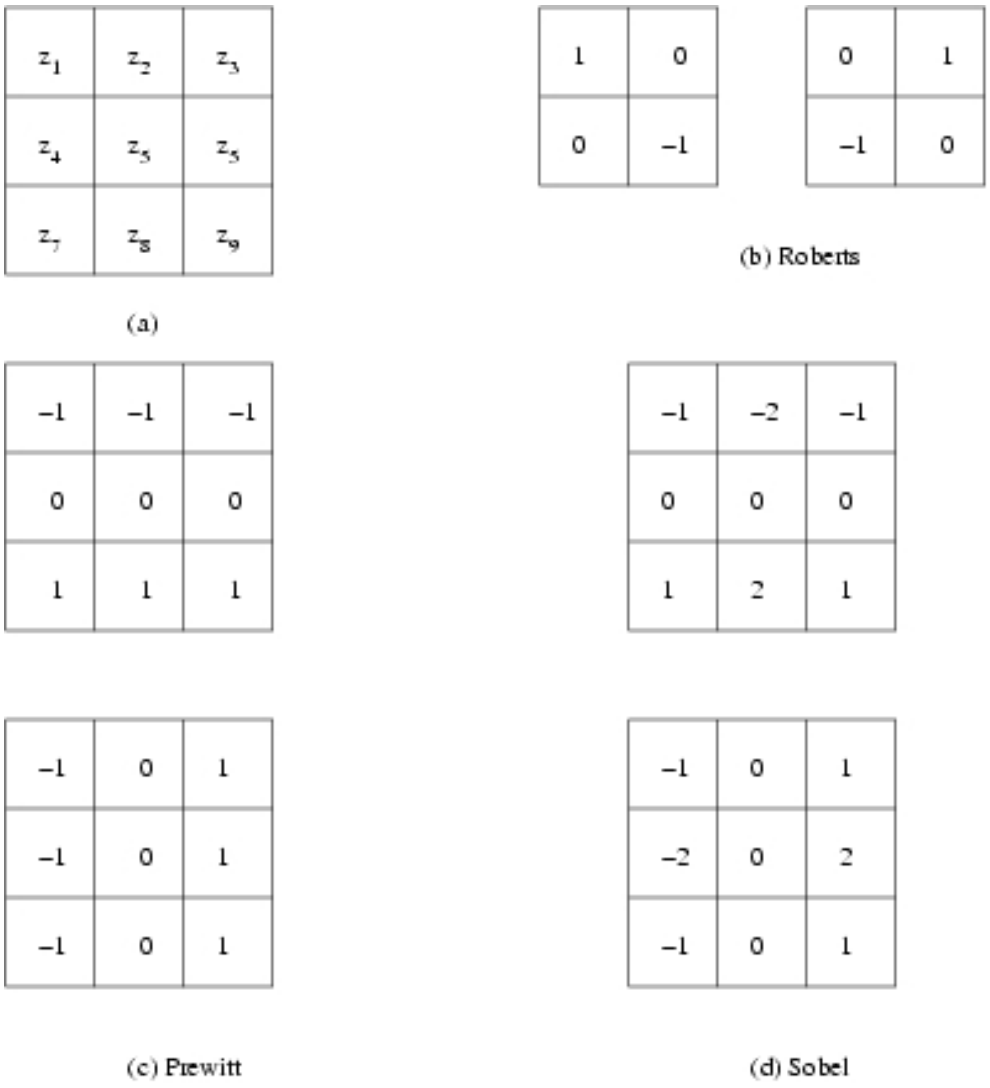


Figura 4.8: Ventana de tamaño 3×3 y varias máscaras usadas para calcular la derivada en el punto z_5. Notar que todos los coeficientes de las máscaras suman cero, que indica una respuesta nula en áreas constantes, como se debe esperar de un operador diferencial. (a) Máscara de Roberts; (b) Máscara de Prewitt; (c) Máscara de Sobel.

función de transferencia del proceso. Numerosos problemas de realce de imagen se pueden expresar en la forma de la ecuación (4.27). Típicamente $f(x,y)$ es conocida y el objetivo, es seleccionar la función de transferencia $H(u,v)$ para que la imagen deseada,

$$g(x,y) = \mathfrak{F}^{-1}[H(u,v)F(u,v)] \tag{4.28}$$

presente resaltada alguna característica de $f(x,y)$.

Un resultado conocido de la teoría de sistemas lineales es que un sistema lineal e invariante de posición queda completamente especificado por su respuesta al impulso, $h(x,y)$. Si se dispone de un sistema lineal e invariante y se desea conocer la función de transferencia, basta con introducir a la entrada un impulso unitario y a la salida se obtendrá la respuesta al impulso. Al realizar la transformada de Fourier de esta respuesta al impulso se consigue la función de transferencia del sistema.

La ecuación (4.26) describe un proceso espacial que es análogo al empleo de máscaras visto en el punto (4.3). Por esta razón, $h(x,y)$ se denomina frecuentemente *máscara de convolución espacial*.

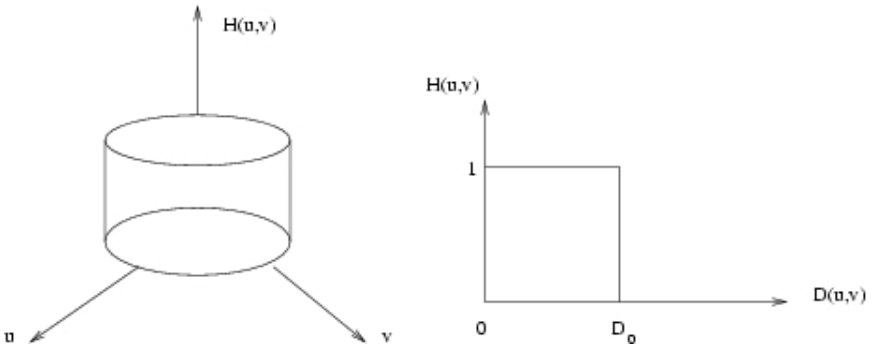
Se puede decir que los principios de realce de imagen en el dominio de la frecuencia son claros. Se trata de calcular la transformada de Fourier de la imagen a realzar, multiplicar el resultado por la función de transferencia de un filtro y, finalmente, tomar la transformada de Fourier inversa para llegar a la imagen realzada.

La idea de pérdida de nitidez por reducción del contenido de altas frecuencias, o de mejor definición incrementando la magnitud de las componentes de alta frecuencia en relación con las de baja frecuencia, proceden de conceptos directamente relacionados con la transformada de Fourier. De hecho, la idea general del filtrado lineal es bastante más atractiva e intuitiva en el dominio de la frecuencia. En la práctica las pequeñas máscaras espaciales son mucho más empleadas que las transformadas de Fourier debido a su facilidad de implementación y a su velocidad de operación. Sin embargo, es esencial la comprensión de los conceptos en el dominio de la frecuencia para solucionar los muchos problemas que no pueden ser resueltos por medio de técnicas espaciales.

4.4.1. Filtrado paso bajo

Los bordes y otras transiciones bruscas en los niveles de gris de una imagen contribuyen significativamente al contenido en altas frecuencias de su transformada de Fourier. Por lo tanto el suavizado (*smoothing*) se consigue, en el dominio de la frecuencia, a base de atenuar un rango específico de componentes de alta frecuencia en la transformada de una imagen dada.

Lo difícil de realizar, a partir de la ecuación (4.27), es seleccionar la función de transferencia $H(u,v)$ que proporcione el tipo de realce deseado sobre $F(u,v)$. A continuación se presenta una serie de filtros paso bajo que actúan de igual forma sobre las partes real e imaginaria de la transformada de Fourier de la imagen. Estos filtros se denominan *filtros de cambio de fase nulo*, pues no alteran la fase de la transformada.



(a) (b)

Figura 4.9: (a) Perspectiva en tres dimensiones de la función de transferencia de un filtro paso bajo ideal; (b) Sección transversal de la función de transferencia.

Filtro ideal

Un filtro paso bajo bidimensional es aquel cuya función de transferencia verifica la relación (Fig. (4.9)):

$$H(u,v) = \begin{cases} 1 & si\ D(u,v) \leq D_0 \\ 0 & si\ D(u,v) > D_0 \end{cases} \tag{4.29}$$

donde D_0 es un valor no negativo, y $D(u,v)$ es la distancia desde el punto (u,v) al origen de coordenadas del plano de frecuencias; es decir:

$$D(u,v) = (u^2 + v^2)^{1/2} \tag{4.30}$$

El nombre de *filtro ideal* indica que todas las frecuencias dentro de un círculo de radio D_0 pasan sin atenuación, mientras que todas las frecuencias fuera de este círculo quedan atenuadas completamente. La sección transversal de un filtro paso bajo viene caracterizada por el punto de transición entre $H(u,v) = 1$ y $H(u,v) = 0$, que se suele denominar *frecuencia de corte*. En el caso de este filtro, el valor de la frecuencia de corte es D_0. Este concepto de *frecuencia de corte* es muy útil para especificar las características del filtro y también sirve como una base común para comparar el comportamiento de diferentes tipos de filtros.

Filtro de Butterworth

La función de transferencia espacial del filtro paso bajo de Butterworth de orden n, y con emplazamiento de la frecuencia de corte a una distancia D_0 del origen, está definido por la relación

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}} \tag{4.31}$$

donde $D(u,v)$ está dado por la ecuación (4.30).Al contrario de lo que sucede con el filtro paso bajo ideal, la función de transferencia del filtro paso bajo de Butterworth carece de una discontinuidad brusca que establezca un corte claro entre las frecuencias transmitidas y las filtradas. Para los filtros cuya función de transferencia cambie sin brusquedad, es habitual definir la frecuencia de corte a partir del lugar de los puntos donde la función $H(u,v)$ se corresponde a una determinada fracción de su valor máximo.

4.4.2. Filtrado paso alto

Como los bordes y demás cambios abruptos en los niveles de gris se asocian a las componentes de alta frecuencia de las imágenes, el afilado de las imágenes (*sharpenning*) se puede conseguir mediante un procesado paso alto en el dominio de la frecuencia, ya que éste atenúa los componentes de baja frecuencia sin producir ninguna distorsión en la información de las componentes de alta frecuencia. A continuación se presentan una serie de filtros paso alto en el dominio de la frecuencia que, además, son de *cambio de fase nulo*.

Filtro ideal

Un filtro ideal bidimensional paso alto es aquel cuya función de transferencia satisface la relación

$$H(u,v) = \begin{cases} 0 & \text{si } D(u,v) \leq D_0 \\ 1 & \text{si } D(u,v) > D_0 \end{cases} \tag{4.32}$$

donde D_0 es la frecuencia de corte medida desde el origen del plano de frecuencias, y $D(u,v)$ está dado por la ecuación (4.30). Este filtro es totalmente contrario al filtro presentado en la sección (4.4.1) porque atenúa todas las frecuencias contenidas en un círculo de radio D_0 mientras que deja pasar, sin atenuación, todas las frecuencias que caen fuera del círculo. Como ocurre con el filtro presentado en la sección (4.4.1), no es físicamente realizable ya que no existe ningún elemento físico que sea capaz de producir transiciones bruscas de uno a cero o viceversa.

Filtro de Butterworth

La función de transferencia del filtro paso alto de Butterworth de orden n y con frecuencia de corte localizada a la distancia D_0 del origen, se define mediante la relación

$$H(u,v) = \frac{1}{1 + [D_0/D(u,v)]^{2n}} \tag{4.33}$$

donde $D(u,v)$ está dado por la ecuación (4.30). Es importante darse cuenta que cuando $D(u,v) = D_0$, $H(u,v)$ cae a la mitad de su valor máximo.

4.4.3. Filtrado homomórfico

El procesado homomórfico parte de la premisa de que una imagen $f(x,y)$ se puede expresar, en términos de sus componentes de iluminación y reflectancia, de la siguiente manera

$$f(x,y) = i(x,y)r(x,y) \tag{4.34}$$

donde $i(x,y)$ es la cantidad de luz incidente sobre la escena que se está visualizando y $r(x,y)$ es la cantidad de luz que se refleja en los objetos que hay en dicha región. Típicamente la iluminación será una señal centrada en banda baja mientras que la reflectancia será un proceso de mayor variación espacial, y por ello, de mayor contenido de alta frecuencia.

Sin embargo, no se puede operar de forma separada con las componentes frecuenciales de iluminación y reflectancia porque la transformada de Fourier del producto de dos funciones no es separable, en otras palabras:

$$\mathfrak{F}\{f(x,y)\} \neq \mathfrak{F}\{i(x,y)\}\mathfrak{F}\{r(x,y)\} \tag{4.35}$$

Los comportamientos de ambos procesos son fácilmente desacoplables mediante una operación logarítmica, ya que ésta convierte logaritmos en sumas:

$$z(x,y) = ln(f(x,y)) = ln(i(x,y)) + ln(r(x,y)) \tag{4.36}$$

Entonces,

$$\mathfrak{F}\{z(x,y)\} = \mathfrak{F}\{ln(f(x,y))\} = \mathfrak{F}\{ln(i(x,y))\} + \mathfrak{F}\{ln(r(x,y))\} \tag{4.37}$$

o

$$Z(u,v) = I(u,v) + R(u,v) \tag{4.38}$$

donde $I(u,v)$ y $R(u,v)$, son las transformadas de Fourier de $ln(i(x,y))$ y $ln(r(x,y))$, respectivamente.

Ahora un simple filtrado lineal permite separar de forma sencilla estas componentes, y a partir de este momento, llevar a cabo un procesado paralelo por cada uno de los dos canales resultantes. Si se procesa la función $Z(u,v)$ por medio de una función de filtrado $H(u,v)$ el resultado que se obtiene es:

$$S(u,v) = H(u,v)Z(u,v) = H(u,v)I(u,v) + H(u,v)R(u,v) \tag{4.39}$$

donde $S(u,v)$ es la transformada de Fourier del resultado. En el dominio espacial,

$$s(x,y) = \mathfrak{F}^{-1}\{S(u,v)\} = \mathfrak{F}^{-1}\{H(u,v)I(u,v)\} + \mathfrak{F}^{-1}\{H(u,v)R(u,v)\} \tag{4.40}$$

Si se denotan $i'(x,y) = \mathfrak{F}^{-1}\{H(u,v)I(u,v)\}$ y $r'(x,y) = \mathfrak{F}^{-1}\{H(u,v)R(u,v)\}$ entonces la ecuación (4.40) se puede expresar de la siguiente forma:

$$s(x,y) = i'(x,y) + r'(x,y) \tag{4.41}$$

Finalmente, como $z(x,y)$ se obtuvo tomando el logaritmo neperiano de la imagen original $f(x,y)$, la operación inversa será la que produzca la imagen realzada $g(x,y)$:

$$g(x,y) = exp[s(x,y)] = exp[i'(x,y)] \cdot exp[r'(x,y)] = i_0(x,y)r_0(x,y) \tag{4.42}$$

donde $i_0 = exp[i'(x,y)]$ y $r_0(x,y) = exp[r'(x,y)]$, son las componentes de iluminación y reflectancia de la imagen de salida.

El método presentado está basado en una clase especial de sistemas conocidos como *sistemas homomórficos*. La clave del método es la separación de la imagen original en las componentes de iluminación y reflectancia. Así, la *función de filtrado homomórfico* $H(u,v)$ puede operar sobre esas dos componentes de manera separada.

4.5. Operaciones para imágenes en color

El uso del color en el procesamiento de imágenes está motivado por dos factores principales. En primer lugar, en análisis de imágenes, el color es un potente descriptor que a menudo simplifica la identificación y extracción de objetos de una escena. En segundo lugar, el ojo humano puede distinguir una amplia gama de colores comparado con los niveles de gris [Pajares03].

En el tratamiento de imágenes, su procesamiento en color se divide en dos áreas fundamentales: *color* y *pseudocolor*. En la primera categoría se procesan las imágenes obtenidas con un sensor de color. En la segunda las imágenes monocromas, esto es, imágenes de grises, son coloreadas por asignación de un color a un determinado nivel de intensidad de gris. El resultado en este caso es una imagen con tres bandas de color.

Las operaciones de realce para el caso de imágenes en color se pueden realizar de la manera expuesta en las secciones anteriores sin más que trabajar en la banda de intensidades como si de una imagen en tonos de gris se tratase. Por ello, es conveniente trabajar en sistemas de color en lo que la banda de intensidades sea una de las bandas del sistema (por ejemplo, el sistema HSI) y no sistemas en los que la intensidad sea una *función*, como es el caso del sistema RGB. En este último, concretamente, el empleo de técnicas de realce de forma separada en cada una de las bandas suele traer consigo falsas coloraciones en la imagen realzada, y por ello no es extraño obtener efectos que causen cierta sorpresa.

4.6. Filtrado de Difusión Anisótropa

Se habla de técnicas de filtrado basadas en la difusión cuando las operaciones de limpieza de ruido, suavizado y realce de bordes se modelan como un proceso de difusión entre celdas adyacentes. A medida que el proceso evoluciona en el tiempo, las zonas similares se vuelven más homogéneas, y los bordes se realzan, con lo que se dispone de un filtro que suaviza las zonas homogéneas (elimina ruido) a la vez que realza los bordes

de los objetos [Aja00]. Si la difusión de la que se habla es *isótropa* entonces la difusión se produce por igual en todas las direcciones. Si por el contrario, se habla de *anisótropa* entonces la difusión variará según la dirección. Al depender el valor que toma cada píxel de su píxel correspondiente en la imagen original así como de sus píxeles vecinos, este método de realce debería estar englobado dentro del apartado de operaciones espaciales, pero será analizado independientemente dada su importancia en el trabajo realizado en este proyecto.

4.6.1. Base matemática

Este proceso tiene como base matemática la ecuación de difusión del calor o difusión de gases. Esta ecuación expresada para el procesado de imágenes se representaría del siguiente modo:

$$\frac{\partial I(x,y,t)}{\partial t} = \operatorname{div}\left(c(x,y,t)\nabla I(x,y,t)\right) \tag{4.43}$$

donde I(x,y) representaría la función de la intensidad de la imagen, I(x,y,t) es la evolución de la imagen en el tiempo y div es el *operador divergencia*, definido:

$$\operatorname{div} f(\vec{x}) = \sum_{i=0}^{n-1} \frac{\partial f}{\partial x_i} \tag{4.44}$$

y el operador ∇ denota el *gradiente*:

$$\nabla f(\vec{x}) = \left(\frac{\partial f}{\partial x_0}, \ldots, \frac{\partial f}{\partial x_{n-1}}\right) \tag{4.45}$$

La función *c(x,y,t)* se conoce con el nombre de *coeficiente de difusión* y junto con el *gradiente* describe el flujo ente celdas cercanas:

$$\Phi = c(x,y,t)\nabla I(x,y,t) \tag{4.46}$$

Los procesos en los que $c(x,y,t) = c$, es decir, el coeficiente de difusión es constante, se conocen como procesos *isótropos*, en caso contrario son *anisótropos*.

Este *coeficiente de difusión* se define como una función del módulo del gradiente:

$$c(\vec{x},t) = f(\nabla I(\vec{x},t)) \tag{4.47}$$

Perona y Malik [Perona90] propusieron dos funciones para el coeficiente de difusión:

$$c_1(\vec{x},t) = \exp(-(\frac{|\nabla I(\vec{x},t)|}{K})^2) \tag{4.48}$$

$$c_2(\vec{x},t) = \frac{1}{1+(\frac{|\nabla I(\vec{x},t)|}{K})^2} \tag{4.49}$$

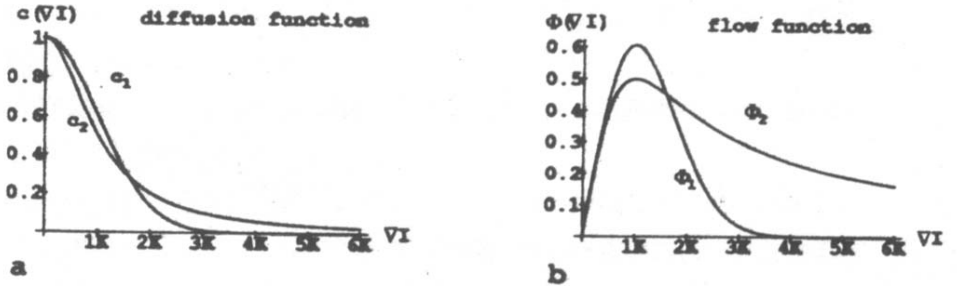


Figura 4.10: (a)Representación de las dos funciones propuestas por Perona y Malik para c. (b) Representación de los flujos frente a ∇I

Se observa en Fig. (4.10) que el *coeficiente de difusión* decrece monótonamente a medida que el *gradiente* aumenta. Por tanto, cuando el *gradiente* es grande, la difusión que se va a producir va a ser pequeña, y viceversa.

El comportamiento de estas dos funciones es diferente, la primera (4.48) va dar preferencia a los bordes con contraste alto frente a los de contraste bajo y la segunda (4.49) va dar preferencia a las regiones anchas frente a las estrechas. En cuanto al parámetro K, éste va influir de manera importante en el grado de difusión. A medida que éste es mayor la difusión es mayor para el mismo *gradiente*. Su valor va a depender de cual sea la aplicación concreta, y éste será elegido según el nivel de ruido y la intensidad de los bordes existentes. Este valor, para la realización del proceso de difusión, puede fijarse de antemano, o puede calcularse de alguna manera en cada iteración y por tanto ir variando.

En Fig. (4.10) además de mostrarse las dos funciones propuestas para el *coeficiente de difusión*, en la izquierda se representan los dos flujos correspondientes en función de ∇I. Viendo esa gráfica se puede comprender la relación existente entre el parámetro K y ∇I. Se observa que el máximo flujo se genera cuando ∇I es igual a K. Cuando decrece por debajo de K el flujo tiende a cero, ya que en las regiones homogéneas existe un flujo mínimo o nulo. Por encima de K el flujo otra vez decrece a cero, haciendo así que la difusión se pare en aquellas zonas con grandes gradientes.

4.6.2. Difusión anisótropa. Práctica

Formulación discreta 1D

Para el tratamiento de imágenes es necesario realizar una reformulación en forma discreta de la ecuación de difusión (4.43) definida anteriormente para los casos continuos. En este caso, en vez de calcular los gradientes locales, sus valores se pueden aproximar por las diferencias existentes entre los elementos de datos cercanos. Para proporcionar una mejor apreciación de lo que significa el proceso de difusión no lineal, primero se va a realizar la reformulación para el caso de una dimensión (1D). Para ello se va a tomar

como eje de coordenadas el eje x. Teniendo en cuenta esto, la ecuación de difusión para 1D se escribiría del siguiente modo:

$$\frac{\partial I(x,t)}{\partial t} = \text{div}\,(c(x,t)\nabla I(x,t)) \tag{4.50}$$

Considerando que para 1D: $\text{div} = \nabla = \frac{\partial}{\partial x}$, y sustituyendo en (4.50):

$$\frac{\partial I(x,t)}{\partial t} = \frac{\partial}{\partial x}\left(c(x,t)\frac{\partial I(x,t)}{\partial x}\right) \tag{4.51}$$

Si ahora se tiene en cuenta la aproximación:

$$\frac{\partial f(x,t)}{\partial x} \approx \frac{1}{\Delta x}(f(x+\frac{\Delta x}{2},t) - f(x-\frac{\Delta x}{2},t))$$

y se aplica en (4.51):

$$\frac{\partial I(x,t)}{\partial t} \approx \frac{\partial}{\partial x}[c(x,t) * (I(x+\frac{\Delta x}{2},t) - I(x-\frac{\Delta x}{2},t))] \tag{4.52}$$

Si de nuevo se aplica la aproximación de la diferencial en la anterior ecuación se obtiene:

$$\frac{\partial I(x,t)}{\partial t} \approx \frac{1}{\Delta x^2}[c(x+\frac{\Delta x}{2},t)(I(x+\Delta x,t) - I(x,t)) - c(x-\frac{\Delta x}{2},t)(I(x,t) - I(x-\Delta x,t))] \tag{4.53}$$

Esta última ecuación se puede expresar:

$$\frac{\partial I(x,t)}{\partial t} \approx \Phi_d - \Phi_i \tag{4.54}$$

donde Φ_d representa el flujo local hacia la derecha y Φ_i representa el flujo local hacia la izquierda (ver Fig. (4.11)).

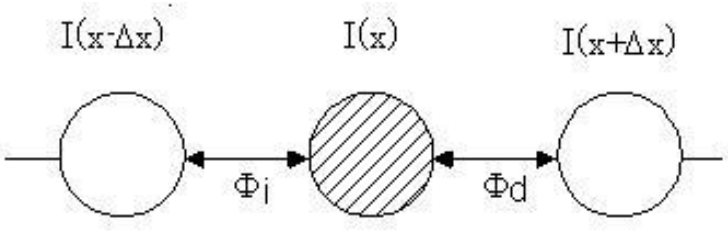


Figura 4.11: Representación de las contribuciones de los flujos existentes para el caso 1D.

Si ahora, de forma similar, en (4.54) se aplica:

$$\frac{\partial f(x,t)}{\partial t} \approx \frac{1}{\Delta t}(f(x,t+\Delta t) - f(x,t))$$

entonces se obtiene:

$$I(x,t+\Delta t) \approx I(x,t) + \Delta t(\Phi_d - \Phi_i) \tag{4.55}$$

Después de haber realizado todo este desarrollo matemático, se puede observar que al discretizar la ecuación de calor se llega a un proceso iterativo, en el que simplemente se indica que el valor de I, para una cierta coordenada x en el siguiente instante de tiempo (en la siguiente iteración) va a venir dado por el valor de I en este instante de tiempo más un cierto valor que va a depender de la diferencia entre los flujos hacia la izquierda y hacia la derecha existentes en torno al punto considerado. La estabilidad de este proceso iterativo, lógicamente, se va a obtener mediante la elección de un valor adecuado para Δt. Como se explicará más adelante, Δt no va poder superar cierto valor.

Formulación discreta 2D

Una vez visto el desarrollo para el caso de una dimensión, se puede proceder a ver cual sería el resultado que se obtendría en el caso 2D. Para ello vamos a tomar como ejes de coordenadas los ejes x e y. A priori, visto el resultado anterior, parece que uno puede predecir fácilmente cual va a ser la expresión final de la ecuación. Pero para evitar que se hagan predicciones erróneas, veamos como sería el desarrollo. La ecuación de la que se parte, para 2D, tendría la siguiente expresión:

$$\frac{\partial I(x,y,t)}{\partial t} = \text{div}[c(x,y,t) * \nabla I(x,y,t)] \tag{4.56}$$

Teniendo en cuenta las aproximaciones de las que se ha hablado para 1D y haciendo una extensión de ellas para 2D, la anterior ecuación se podría expresar:

$$\begin{aligned}
\frac{\partial I(x,y,t)}{\partial t} \approx\ & \frac{1}{\Delta x^2}[c(x+\frac{\Delta x}{2},y,t) * (I(x+\Delta x,y,t) - I(x,y,t)) \\
-\ & c(x-\frac{\Delta x}{2},y,t) * (I(x,y,t) - I(x-\Delta x,y,t))] \\
+\ & \frac{1}{\Delta y^2}[c(x,y+\frac{\Delta y}{2},t) * (I(x,y+\Delta y,t) - I(x,y,t)) \\
-\ & c(x,y-\frac{\Delta y}{2},t) * (I(x,y,t) - I(x,y-\Delta y,t))]
\end{aligned}$$

Se aprecia que el primer sumando en el término de la derecha de la ecuación (4.57) es similar al término que se tenía a la derecha de la igualdad en (4.53), realmente sólo se diferencia en la presencia de la coordenada y, por tanto este término será equivalente a la diferencia entre dos flujos locales. Uno de ellos será un flujo hacia la derecha y el otro hacia la izquierda en el eje x, al igual que en el caso de una dimensión. El segundo sumando en (4.57) es igual al primero, salvo que las variaciones se producen en y en vez de en x. Por tanto, también equivaldrá a la diferencia entre dos flujo locales, uno hacia la derecha y el otro hacia la izquierda, pero ahora en el eje y. Viendo esto, entonces, se puede ya dar la expresión final de la discretización para el caso 2D:

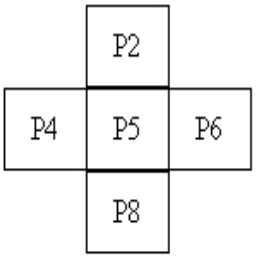


Figura 4.12: Representación de los píxeles situados al norte, sur, este y oeste del pixel de estudio.

$$I(x,y,t+\Delta t) \approx I(x,y,t) + \Delta t * (\Phi_e - \Phi_w + \Phi_n - \Phi_s) \tag{4.57}$$

donde Φ_e y Φ_w son los flujos hacia el este y hacia el oeste, que se corresponderían con los flujos hacia la derecha e izquierda en el eje x y Φ_n y Φ_s son los flujos norte y sur que se corresponden con los flujos hacia la derecha e izquierda en el eje y. Como en el caso 1D se obtiene un proceso iterativo en el que en cada nuevo paso temporal $t+\Delta t$ una nueva imagen es generada a partir de la imagen del paso temporal t más un cierto valor que va a depender de las diferencias entre los flujos locales. Lógicamente la estabilidad del resultado va a venir determinada por el parámetro Δt.

Aplicación a imágenes

Puesto que lo que interesa es poder aplicar la ecuación (4.57) a imágenes, en concreto a imágenes médicas, se debe expresar dicha ecuación en función de los píxeles de la imagen. I(x,y,t), para un cierto valor de x y otro de y, hará referencia a la intensidad de un cierto píxel de la imagen en un instante de tiempo. El valor de éste en el siguiente paso temporal como se ha dicho ya varías veces, estará influenciado por los cuatro vecinos situados al norte, sur, este y oeste Fig. (4.12). Pero para obtener mejores resultados, a la hora de calcular los flujos, no sólo se van a tener en cuenta estos cuatro píxeles, sino que también se consideraran los cuatros vecinos diagonales. Por tanto, se va a trabajar con una matriz de 3×3 centrada en el píxel de estudio Fig. (4.13).

Las expresiones de los cuatro flujos, que se deben de poner en función de los píxeles, son las siguientes:

$$\Phi_e = \frac{1}{\Delta x^2}\left[c(x+\frac{\Delta x}{2},y,t) * (I(x+\Delta x,y,t) - I(x,y,t))\right]$$
$$\Phi_w = \frac{1}{\Delta x^2}\left[c(x-\frac{\Delta x}{2},y,t) * (I(x,y,t) - I(x-\Delta x,y,t))\right]$$

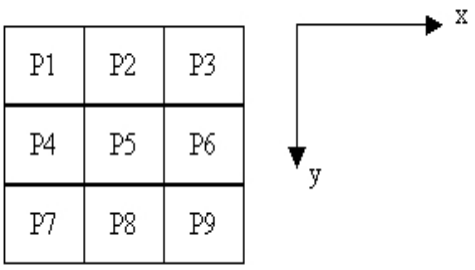


Figura 4.13: Vecindario 3x3.

$$\Phi_n = \frac{1}{\Delta y^2}\left[c(x,y+\frac{\Delta y}{2},t)*(I(x,y+\Delta y,t)-I(x,y,t))\right]$$
$$\Phi_s = \frac{1}{\Delta y^2}\left[c(x,y-\frac{\Delta y}{2},t)*(I(x,y,t)-I(x,y-\Delta y,t))\right]$$

Además el *coeficiente de difusión* siguiendo la recomendación de Perona y Malik se expresa como una función del *gradiente*:

$$c(x,y,t) = f(\|\nabla I(x,y,t)\|) \tag{4.58}$$

Y si se hace la aproximación del *gradiente* vista anteriormente, se tiene para un cierto t:

$$\|\nabla I(x,y)\| \approx \sqrt{\frac{1}{\Delta x^2}\left[I(x+\frac{\Delta x}{2},y)-I(x-\frac{\Delta x}{2},y)\right]^2+\frac{1}{\Delta y^2}\left[I(x,y+\frac{\Delta y}{2})-I(x,y-\frac{\Delta y}{2})\right]^2} \tag{4.59}$$

Tras las operaciones pertinentes (ver apéndice A), se pueden obtener los siguientes *gradientes* en función de los píxeles de la matriz centrada en P_5:

$$\begin{aligned}
\|\nabla I(x+\frac{\Delta x}{2},y)\| &\approx \sqrt{\frac{1}{\Delta x^2}(P_6-P_5)^2+\frac{1}{\Delta y^2}(\frac{P_9-P_3}{2})^2}\\
\|\nabla I(x-\frac{\Delta x}{2},y)\| &\approx \sqrt{\frac{1}{\Delta x^2}(P_5-P_4)^2+\frac{1}{\Delta y^2}(\frac{P_7-P_1}{2})^2}\\
\|\nabla I(x,y+\frac{\Delta y}{2})\| &\approx \sqrt{\frac{1}{\Delta x^2}(\frac{P_9-P_7}{2})^2+\frac{1}{\Delta y^2}(P_8-P_5)^2}\\
\|\nabla I(x,y-\frac{\Delta y}{2})\| &\approx \sqrt{\frac{1}{\Delta x^2}(\frac{(}{P_3-P_1}2)^2+\frac{1}{\Delta y^2}(P_5-P_2)^2}
\end{aligned} \tag{4.60}$$

Estas expresiones permiten ya expresar los flujos en función de los píxeles:

$$\begin{aligned} \Phi_e &= \Phi_e(P_6, P_5, P_9, P_3) \\ \Phi_w &= \Phi_w(P_5, P_4, P_7, P_1) \\ \Phi_n &= \Phi_n(P_8, P_5, P_9, P_7) \\ \Phi_s &= \Phi_s(P_5, P_2, P_3, P_1) \end{aligned} \tag{4.61}$$

Se representan de forma esquemática cuales son los píxeles que intervienen en cada flujo en la Fig. (4.14). Se aprecia, como ya se dijo antes y se indica en (6.19), que para calcular el flujo local en una determinada dirección se tienen en cuenta además los dos píxeles vecinos en las diagonales, así, por ejemplo, para calcular el flujo local hacia el norte no sólo se tienen en cuenta P_8 y P_5, sino que también P_9 y P_7.

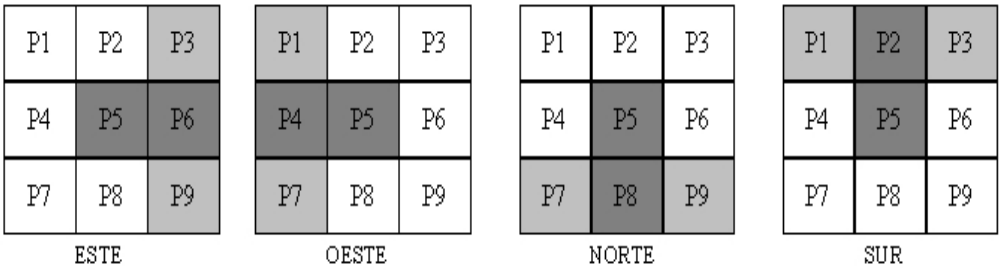


Figura 4.14: Píxeles que intervienen en los distintos flujos locales.

Capítulo 5

Realce borroso: filtrado borroso de imágenes

5.1. Introducción

Como se ha visto en el capítulo 4, el realce es un paso importante en muchas aplicaciones de procesado de imágenes. El tipo de algoritmo de realce a usar depende del objetivo que se pretende conseguir por medio del proceso de realce, así como de la aplicación particular en sí misma. Eliminar el ruido y realzar los bordes son procesos inherentemente conflictivos, ya que al suavizar una región se destruyen los bordes y al tratar de realzar éstos puede que se introduzca ruido innecesario.

El filtrado de ruido puede entenderse como el remplazo del valor de nivel de gris de cada uno de los píxeles de la imagen por un nuevo valor dependiente del contexto local. Idealmente, el algoritmo de filtrado debería variar de un píxel a otro dependiendo del contexto local. Por ejemplo, si la región local no tiene apenas variaciones en los niveles de gris, entonces el nuevo valor del pixel debe ser un tipo de promedio de los valores locales. Por otra parte, si la región local contiene bordes o ruido impulsivo, se debería usar otro tipo de filtrado. Sin embargo, es extremadamente difícil, si no imposible, establecer las condiciones bajo las cuales elegir el tipo de filtro seleccionado en cada momento ya que las condiciones locales puede que sean evaluadas de forma vaga en algunas partes de la imagen. Por lo tanto, un sistema de filtrado necesita ser capaz de realizar razonamientos con información vaga e incierta.

Precisamente una de las características clave de la lógica borrosa es su habilidad a la hora de tratar con las típicas incertezas que caracterizan los sistemas físicos. A este respecto, las técnicas borrosas tienen una oportunidad mejor de llegar a ser un componente fundamental de cualquier sistema de instrumentación artificial que las técnicas tradicionales.

Desde 1992, el número de aproximaciones al filtrado borroso se ha ido incrementando progresivamente [Russo98]. En la literatura de procesado de imagen se pueden encontrar

técnicas como la modificación de niveles de gris, operaciones sobre el histograma, filtros basados en propiedades estadísticas, filtrado adaptativo, filtros que hacen uso de la transformada de Fourier o de la transformada de wavelet, difusión anisótropa, algoritmos genéticos, filtros morfológicos, ... Una posible clasificación de las técnicas existentes se muestra en la Fig. (5.1).

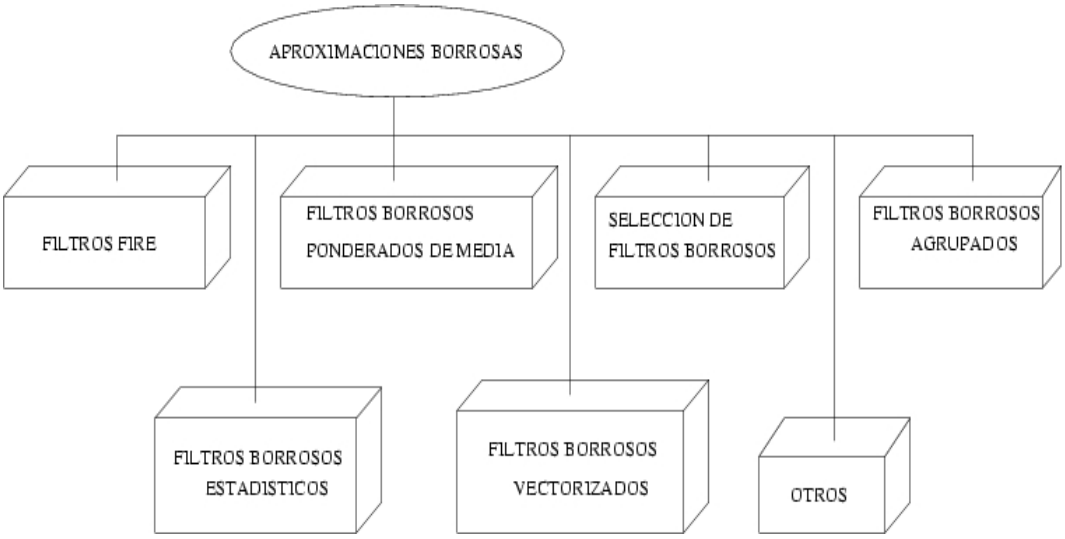


Figura 5.1: Clasificación de métodos de filtrado borroso.

5.2. Definiciones previas

A continuación se muestran algunas definiciones que pueden resultar de utilidad cuando se trabaja en el tratamiento borroso de imágenes. Estas definiciones fueron propuestas por Tizhoosh y gozan de bastante aceptación en la literatura. Para más información consultar [Tizhoosh97b] [Tizhoosh99c] [Kerre00].

Una imagen X de tamaño $M \times N$ con L niveles de gris $g = 0, 1, 2, \cdots, L-1$ puede verse como un conjunto de *fuzzy singletons* (conjuntos borrosos cuyo soporte es un sólo punto), de forma que el valor de cada pixel indica la pertenencia de dicho pixel a un conjunto de propiedades dado: brillo, difusión, etc.

$$X = \bigcup_{m=1}^{M} \bigcup_{n=1}^{N} \frac{\mu_{mn}}{g_{mn}} \qquad \text{con } \mu_{mn} \in [0,1] \tag{5.1}$$

donde μ_{mn}/g_{mn} denota, en la notación de conjuntos borrosos, la función de pertenencia del mn-ésimo pixel. La función de pertenencia μ_{mn} caracteriza una propiedad de la imagen y se puede definir de forma global o local. La definición de los valores de la función de pertenencia depende de los requerimientos específicos de la aplicación.

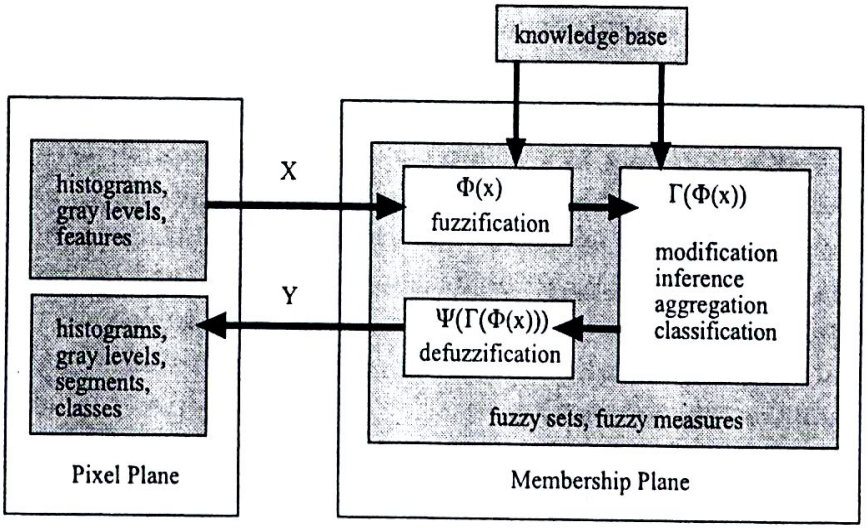


Figura 5.2: Estructura general de un procesado borroso de imágenes.

El procesado borroso de imágenes consiste en tres pasos: borrosificación Φ, operaciones sobre valores de pertenencia Γ y, si es necesario, desborrosificación Ψ (Fig. (5.2)).

La salida Y del sistema para una entrada X se produce mediante

$$Y = \Psi(\Gamma(\Phi(X))) \tag{5.2}$$

La diferencia principal con otras metodologías de procesado de imagen es que la entrada X será procesada en el llamado *plano de pertenencia* donde se pueden usar las posibilidades que ofrece la lógica borrosa. En este plano se modifican los valores obtenidos tras el proceso de borrosificación y posteriormente, estos valores modificados se transforman otra vez en niveles de gris.

5.3. Filtrado borroso

En los últimos años, se han propuesto muchas aproximaciones al filtrado borroso de imágenes. A groso modo se pueden distinguir entre tres tipos de aproximaciones diferentes:

- Filtrado borroso puro: usa solamente reglas del tipo *if-then*.
- Extensión borrosa de filtros existentes: usa funciones de pertenencia borrosas o reglas borrosas para extender la idea que se propone en filtros clásicos.
- Técnicas de fusión borrosa: mezcla los resultados de diferentes filtros para combinar sus ventajas.

5.3.1. Filtrado borroso puro

Este tipo de filtros están basados en reglas borrosas de tipo *if-then*, donde el efecto de filtrado deseado se consigue usando un conjunto adecuado de reglas lingüísticas. Russo y Ramponi han propuesto una clase de filtros a los que han llamado filtros de inferencia borrosa gobernados por una acción adicional ó FIRE (*Fuzzy Inference Ruled by Else-action*) fácilmente aplicables al filtrado de imágenes. Ejemplos concretos de filtros FIRE son los que se presentan en [Russo95b], [Russo95a], [Russo96c] y [Russo96a].

Los filtros FIRE, que originalmente fueron propuestos para una sóla dimensión (1-D) y dos dimensiones (2-D), han ido mejorando su estructura progresivamente. Típicamente, un filtro FIRE evalúa la información dentro de una ventana mediante reglas borrosas las cuáles tratan con diferencias de luminancia entre el pixel central de la ventana y el resto de píxeles de la ventana. La base de reglas borrosas se encarga de proporcionar el término de corrección que trata de eliminar el ruido (acción *THEN*). Si ninguna regla de la base se satisface, el pixel central de la ventana no se modifica (acción *ELSE*).

Para describir estos métodos, supóngase que se trate con imágenes que tienen L niveles de gris. Sea $x(\mathbf{n})$ la luminancia del pixel localizado en las coordenadas $\mathbf{n} = [n_1, n_2]$ de la imagen de entrada e $y(\mathbf{n})$ la correspondiente luminancia del pixel de la imagen de salida ($0 \leq x(\mathbf{n}) \leq L-1$, $0 \leq y(\mathbf{n}) \leq L-1$). Sea $H(\mathbf{n}) = \{x_i(\mathbf{n}); i = 0, \cdots, N\}$ el conjunto de píxeles que pertenecen a la ventana cuyo centro es $x(\mathbf{n})$, donde $x_0 = x(\mathbf{n})$.

Como ejemplo, considérese un filtro FIRE para la eliminación de ruido impulsivo. En este filtro H incluye cuatro píxeles vecinos de x_0: $H = \{x_0, x_1, x_2, x_3, x_4\}$. Entonces, las entradas del filtro son cuatro diferencias de luminancia: $\Delta x_1 = x_1 - x_0$, $\Delta x_2 = x_2 - x_0$, $\Delta x_3 = x_3 - x_0$, $\Delta x_4 = x_4 - x_0$. La variable de salida $\Delta y(\mathbf{n})$ es el término de corrección que, sumado a $x(\mathbf{n})$, produce el valor de luminancia resultante $y(\mathbf{n}) = \Delta x(\mathbf{n}) + \Delta y(\mathbf{n})$.

La base de reglas incluye dos subbases de reglas simétricas y una regla tipo *ELSE* como sigue:

IF(Δx_1,LP) AND (Δx_2,LP) AND (Δx_3,LP) THEN ($\Delta y, PO$)

IF(Δx_2,LP) AND (Δx_3,LP) AND (Δx_4,LP) THEN ($\Delta y, PO$)

IF(Δx_1,LP) AND (Δx_3,LP) AND (Δx_3,LP) THEN ($\Delta y, PO$)

IF(Δx_1,LP) AND (Δx_2,LP) AND (Δx_4,LP) THEN ($\Delta y, PO$)

IF(Δx_1,LN) AND (Δx_2,LN) AND (Δx_3,LN) THEN ($\Delta y, NE$)

IF(Δx_2,LN) AND (Δx_3,LN) AND (Δx_4,LN) THEN ($\Delta y, NE$)

IF(Δx_1,LN) AND (Δx_3,LN) AND (Δx_3,LN) THEN ($\Delta y, NE$)

IF(Δx_1,LN) AND (Δx_2,LN) AND (Δx_4,LN) THEN ($\Delta y, NE$)

ELSE (Δy,ZE)

donde LP (*Large Positive*), LN (*Large Negative*), PO (*Positive*), ZE (*Zero*) y NE (*Negative*) denotan conjuntos borrosos. Las dos subbases de reglas simétricas están diseñadas

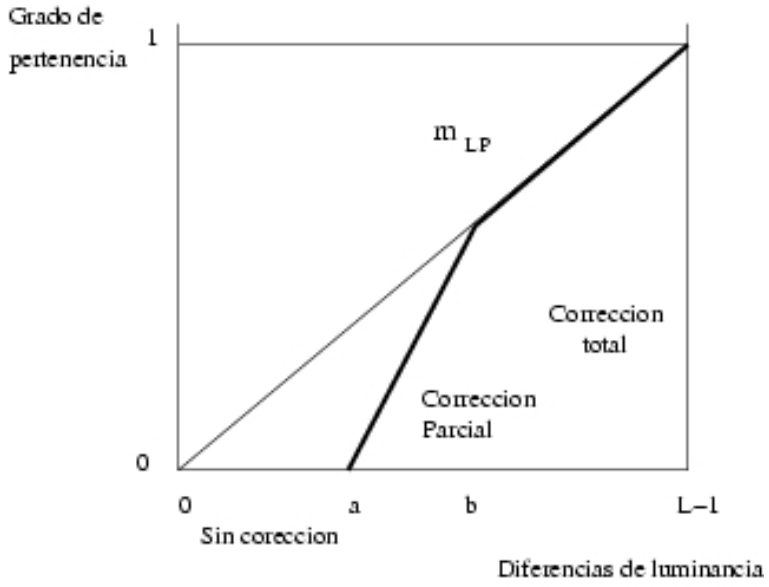


Figura 5.3: Forma del conjunto borroso *LP* diseñado para cancelar pulsos de ruido sin perder nitidez en los detalles.

para encontrar pulsos de ruido negativos y positivos, respectivamente. Para este propósito, cada subbase de reglas incluye cuatro reglas direccionales y cada una de las reglas trata con un patrón específico.

La salida se calcula mediante la relación

$$\Delta y(\mathbf{n}) = (L-1)(\lambda_1(\mathbf{n}) - \lambda_2(\mathbf{n}))$$

donde

$$\lambda_1(\mathbf{n}) = max_{j=1,\ldots,M}\{min_{i\in I_j}\{m_{LP}(\Delta x_i(\mathbf{n}))\}\}$$

$$\lambda_2(\mathbf{n}) = max_{j=1,\ldots,M}\{min_{i\in I_j}\{m_{LN}(\Delta x_i(\mathbf{n}))\}\}$$

y I_j es el conjunto de índices que define el j-ésimo patrón de píxeles. Los conjuntos borrosos *LP* y *LN* deben ser simétricos: $m_{LN}(u) = m_{LP}(-u)$. Las formas de los conjuntos borrosos deben ser diseñadas cuidadosamente para combinar el suavizado y el mantenimiento de detalles (Fig. (5.3)). El filtro se aplica de manera recursiva a la imagen, de forma que el nuevo valor $y(\mathbf{n})$ se asigna a la luminancia $x(\mathbf{n})$ al final del procesado. Se pueden tratar diferentes estadísticas de ruido mediante la elección adecuada de los conjuntos borrosos, reglas y mecanismo de agregación. Además, se pueden combinar reglas que traten diferentes estadísticas de ruido en la misma estructura de filtrado.

Siguiendo una filosofía similar a los métodos FIRE, se encuentran también los trabajos de Farbiz *et al.* [Farbiz00].

5.3.2. Extensión borrosa de filtros existentes

Otra posibilidad para aplicar la lógica borrosa al filtrado de imágenes es la extensión de filtros ya conocidos. Unas veces la extensión usa una simple función de pertenencia que sustituye a un parámetro umbral mientras que en otras ocasiones se aplica un conjunto de reglas borrosas para adaptar algún parámetro del filtro correspondiente. Parece evidente que la forma en la que se extienden estos filtros ya existentes no es única sino que se pueden usar todas las herramientas que proporciona la lógica borrosa para la extensión del filtro.

Dentro de esta clasificación se encuentran el filtro borroso gaussiano propuesto por Law *et al.* en [Law96], el filtro borroso de mediana propuesto por Taguchi *et al.* en [Taguchi96c], los filtros borrosos de pila desarrollados por Yang y Toh, el filtro borroso de media, los filtros borrosos de mediana, el filtro de difusión anisótropa propuesto por Aja *et al.* en [Aja01],...

Los filtros de pila (*stack filters*) aparecieron como una generalización de los filtros de orden elevado en un esfuerzo por incrementar las operaciones no lineales disponibles. Un *stack filter* es un filtro no lineal en el que la salida (para cada ventana que recorre la imagen) se obtiene al superponer los resultados de la pila de funciones Booleanas que operan sobre versiones umbralizadas de los datos originales en la ventana [Taguchi96c]. Como su propio nombre indica, la salida de las funciones Booleanas está restringida a dos únicos valores (0 ó 1). Intuitivamente, se podrían esperar mejores resultados si la salida de estas funciones variase de manera continua de cero a uno. Este tipo de funciones se denominan funciones Booleanas borrosas. A los filtros que utilizan funciones booleanas borrosas reciben el nombre de filtros borrosos de pila.

El filtro de media y sus variaciones se usan en el procesado de imágenes para suavizar imágenes ruidosas, pero no producen buenos resultados cuando la cantidad de ruido presente en la imagen es elevada. El filtro borroso de media introducido por Lee *et al.* es un filtro de media que opera con tres números borrosos *dark* (DK), *median* (MD) y *bright* (BR) que tienen la siguiente forma:

$$f(x) = \begin{cases} L\left(\frac{m_{LR}-x}{\alpha_{LR}}\right) & para \quad x \leq m, \\ R\left(\frac{x-m_{LR}}{\beta_{LR}}\right) & para \quad x > m \end{cases} \tag{5.3}$$

Se procesan tres reglas borrosas en ventanas de tamaño 3×3 y cada una de ellas produce un subresultado $\bar{y}_1(i,j), \bar{y}_2(i,j)$ y $\bar{y}_3(i,j)$. El resultado final es:

$$y(i,j) = \frac{\sum_{k=1}^{3} w_r \times \bar{y}_k(i,j)}{\sum_{k=1}^{3} w_r} \tag{5.4}$$

La regla borrosa para el conjunto *dark* es la siguiente:

If
$x(i-1,j-1)$ is DK **and** $x(i-1,j)$ is DK **and** $x(i-1,j+1)$ is DK **and** $x(i,j-1)$ is DK **and** $x(i,j)$ is DK **and** $x(i,j+1)$ is DK **and** $x(i+1,j-1)$ is DK **and** $x(i+1,j)$ is DK **and** $x(i+1,j+1)$ is DK,
then

$$\bar{y}_1(i,j) = \frac{\sum_{k=-1}^{1}\sum_{l=-1}^{1} f_{DK}(x(i+k,j+l)) \times x(i+k,j+l)}{\sum_{k=-1}^{1}\sum_{l=-1}^{1} f_{DK}(x(i+k,j+l))} \tag{5.5}$$

Las reglas para los conjuntos *median* y *bright* se definen de manera análoga.

5.3.3. Fusión suave de filtros existentes

En muchas situaciones, el proceso de filtrado debe cumplir diferentes requerimientos. Por ejemplo, la imagen debe ser suavizada pero sin perder los detalles pequeños. A pesar de todos los avances en realce, no es posible desarrollar un *superfiltro* que resuelva todos los conflictos en todas las situaciones posibles. Por lo tanto, a menudo es más apropiado combinar filtros existentes usando sus ventajas y de manera simultánea evitando sus limitaciones. Las técnicas borrosas nos permiten implementar esta fusión de un modo robusto, ya que hacen posible usar *decisiones suaves*. Esta propiedad es fundamental ya que en las imágenes reales es imposible encontrar una respuesta clara a si una zona de la imagen es ruidosa, suave o abrupta.

Choi y Krishnapuram [Choi97] propusieron un esquema de fusión basado en reglas *if-then*, usando un grado de compatibilidad entre el pixel central y sus vecinos:

If the compatibility is *small*, **then** use the filter F_1,

If the compatibility is *medium*, **then** use the filter F_2,

If the compatibility is *large*, **then** use the filter F_3,

Los filtros F_1, F_2 y F_3 se seleccionan teniendo en cuenta la aplicación en concreto.

Capítulo 6

Filtros borrosos diseñados

El objetivo de este proyecto es el diseño de nuevos filtros para el tratamiento de imágenes médicas captadas mediante ultrasonidos. Como ya se señalaba en la introducción, este tipo de imágenes se ven afectadas por la presencia de ruido *speckle*. La mayoría de las técnicas ordinarias no permiten preservar bien los bordes cuando se suavizan las imágenes que contienen ruido, existiendo un compromiso entre la cantidad de ruido eliminado y la conservación de los bordes y estructuras finas. La técnica de realce a aplicar sobre cada píxel debería ser dependiente del contexto local, pudiendo decidir así, si las diferencias de intensidad entre píxeles se deben a la existencia de bordes, o simplemente se trata de ruido. Sin embargo, es extremadamente difícil establecer los condiciones bajo las cuales elegir como se debe de tratar cada píxel en cada momento, ya que las condiciones locales puede que sean evaluadas de forma vaga en algunas partes de la imagen.

Tras un estudio del estado del arte en temas del compromiso *ruido vs. bordes*, se ha observado que la lógica borrosa es una de las opciones más adecuadas con las que afrontar este problema. La naturaleza no-lineal que poseen los filtros no borrosos y su flexibilidad de diseño, permiten realizar un procesado en función de la información vaga que aporta la imagen ruidosa.

Por otra parte, la mayoría de los filtros encontrados presentan buen comportamiento ante el ruido *gaussiano* pero no ante el *speckle* que es el que nos interesa en este proyecto. Sin embargo, tras una profunda revisión de diferentes filtros borrosos, se observa que existe un tipo de filtrado que además de preservar bien los bordes, presenta la propiedad de eliminar ruido de naturaleza multiplicativa, es decir, ruido el *speckle*. Se trata de los filtros IFCF *(filtro iterativos basados en lógica de control borroso)*, desarrollados por Farbiz, Menhaj, Motamedi y Hagan en [Farbiz00]. Los principios en los que se basan estos filtros, así como los diferentes operadores utilizados en ellos, son el punto de partida para el diseño de los filtros desarrollados en este proyecto.

En este capítulo se va a explicar de forma detallada cada uno de los filtros diseñados e implementados, pero primero se analizará el principio de funcionamiento de los filtros IFCF, para pasar posteriormente a explicar los diferentes filtros creados. Estos se organi-

zan en los siguientes *toolboxes*:

- *Toolbox HPFborroso*:

  El objetivo de los filtros implementados en este *toolbox* es el de detectar bordes en imágenes afectadas por ruido *speckle* mediante un filtrado paso alto de naturaleza borrosa.

- *Toolbox Anisótropo*:

  En este *toolbox* se pretende realizar un filtrado anisótropo de naturaleza borrosa, que presente un buen comportamiento ante imágenes afectadas por ruido *speckle*. El algoritmo que se utilizará para el filtrado de las imágenes ya está definido, se trata de el algoritmo de difusión anisótropa propuesto por [Gerig].

- *Toolbox Difusión*:

  El objetivo que se pretende alcanzar en este *toolbox* también es el de realizar un filtrado anisótropo de naturaleza borrosa para imágenes afectadas por ruido *speckle*. Para ello el punto de partida será la regularización espacial de el filtro de Perona-Malik [Perona90] realizada por Caté et al.

En esta descripción se usará, para determinadas estructuras, terminología inglesa ya que se ha considerado más oportuno con vistas a una posible ampliación de los temas que se proponen o simplemente para mayor facilidad de lectura de los artículos originales donde se proponen los operadores.

6.1. Filtros basados en lógica de control borroso

6.1.1. Filtro iterativo basado en lógica de control borroso - IFCF

En este apartado se va a presentar la arquitectura que utiliza el filtro IFCF (*Iterative Fuzzy Control Filter*). La estructura general que adopta se basa en el mecanismo típico *if-then-else*. La idea básica que se maneja es la de no procesar cada punto del área de interés de manera uniforme. Esta idea está ampliamente extendida en aplicaciones de control borroso. Esta no uniformidad reducirá la sensibilidad del filtro al ruido y por lo tanto mejorará el tratamiento de los bordes de la imagen.

Las reglas borrosas que se proponen para este tipo de filtrado son las siguientes mientras que las funciones de pertenencia que se utilizan se pueden ver en la Fig. (6.1). Estas funciones de pertenencia denominadas NB, NM, NS, PS, PM, PB y Z son de naturaleza triangular y trapezoidal.

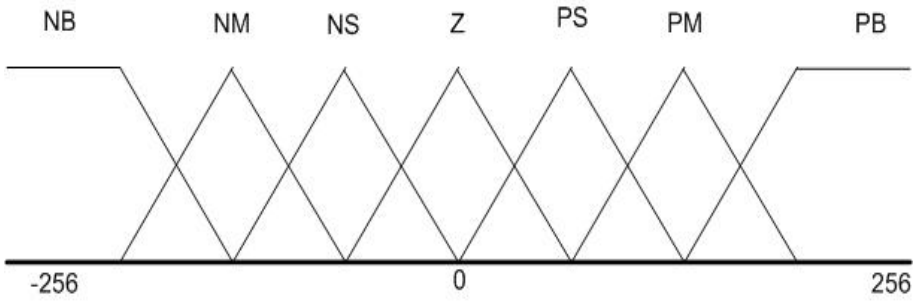


Figura 6.1: Funciones de pertenencia.

R_1:	IF (**more**	of	x_i	are	**NB**)	THEN	y	is	**NB**
R_2:	IF (**more**	of	x_i	are	**NM**)	THEN	y	is	**NM**
R_3:	IF (**more**	of	x_i	are	**NS**)	THEN	y	is	**NS**
R_4:	IF (**more**	of	x_i	are	**PS**)	THEN	y	is	**PS**
R_5:	IF (**more**	of	x_i	are	**PM**)	THEN	y	is	**PM**
R_6:	IF (**more**	of	x_i	are	**PB**)	THEN	y	is	**PB**
R_0:	ELSE	y	is	**Z**					

Arriba, los x_i's son las diferencias de luminancia entre píxeles vecinos, P_i (localizados en una ventana de tamaño NxN), y el píxel central P: $x_i = Pi - P$. La variable de salida y es la cantidad que se ha de añadir a P para producir la luminancia de salida P'. El término "la mayoría" (**more**) representa a una función borrosa de tipo S cuya forma es la descrita en la ecuación (6.1) ó en la ecuación (6.2).

$$\mu_{more}(z) = \frac{1}{1+\exp(-(\alpha_1 z - \beta_1))} \tag{6.1}$$

$$\mu_{more}(z) = \begin{cases} 0 & z \leq a \\ 0{,}5\left\{1 - cos\left[\frac{\pi(t-a)}{b-a}\right]\right\} & a < z < b \\ 1 & z > b \end{cases} \tag{6.2}$$

Valores usuales para los parámetros a y b son 0.2 y 0.8 respectivamente.

Cálculo del grado de activación de las reglas

El grado de activación de la regla R_1 se calcula haciendo uso de la siguiente relación:

$$\lambda_1 = min\{\mu_{NB}(x_i) : x_i \in soporte(NB)\} \times \mu_{more}\left[\frac{\text{número de } x_i / x_i \in \text{soporte(NB)}}{\text{número total de } x_i}\right] \tag{6.3}$$

Evidentemente para el resto de las reglas como ésta (R_2-R_6) el cálculo del grado de activación se realiza de la misma manera. Mientras que para la regla cero, se aplica la siguiente fórmula:

$$\lambda_0 = Max\left\{0, 1 - \sum_{i=0}^{6} \lambda_i\right\} \tag{6.4}$$

Para inferir la salida numéricamente a partir de las reglas R_0-R_6 presentadas anteriormente se emplea el siguiente mecanismo:

$$y = \frac{\sum_{i=0}^{6} C_i w_i \lambda_i}{\sum_{i=0}^{6} w_i \lambda_i} \tag{6.5}$$

donde C_i y w_i son el punto central y la anchura de las funciones de pertenencia usadas en la i-ésima regla borrosa. Como todas las anchuras tienen el mismo valor y $C_o = 0$, la ecuación (6.4) se puede simplificar de la siguiente manera:

$$y = \sum_{i=1}^{6} C_i \lambda_i \tag{6.6}$$

6.1.2. Filtro IFCF modificado - MIFCF

Si se observasen resultados experimentales del filtro IFCF se podría ver cómo cada vez que se incrementa una iteración el procesado, los bordes de la imagen de salida pierden nitidez. Por lo tanto, al incrementar el número de iteraciones se consigue una imagen menos nítida. Especialmente para aquellos casos en los que la cantidad de ruido presente en la imagen original es pequeña, esto hace que la imagen, después de unas pocas iteraciones, acabe siendo más bien degradada que realzada. Para resolver este problema, en cada paso del procesado se sintoniza la función **more** de forma que se vuelva más abrupta en los límites como se puede ver en la Fig. (6.2). La razón que se esconde detrás de este razonamiento es la de forzar al filtro a que se vuelva más activo cuando es necesario. Por otra parte, el filtro se vuelve menos activo en aquellas partes de la imagen que no necesitan ningún cambio.

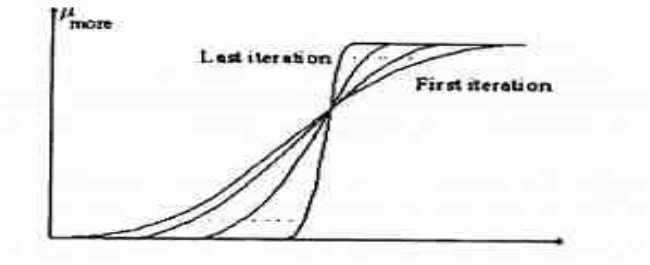


Figura 6.2: Cambios en la forma de la función more con las iteraciones.

Al volverse la forma de la función **more** más abrupta en cada iteración, su rango de actividad disminuye; esto hace que las reglas R_1-R_6 sólo sean activadas en aquellas regiones en las que tengan una mayor posibilidad de ser activadas. De este modo, después de unas pocas iteraciones las reglas no se activarán en la mayoría de los píxeles. Esta

característica permite elevar el número de iteraciones sin tener problemas de nitidez en los bordes.

La utilidad de esta modificación puede verse desde otro punto de vista. Puede ser empleada como criterio de parada del algoritmo de la siguiente forma; cuando el número de píxeles para los cuáles se activa al menos una regla (R_1-R_6) es menor de un cierto valor umbral (que puede establecerse como un porcentaje pequeño del número total de píxeles de la imagen) no tiene sentido realizar más iteraciones ya que no se conseguirá con ello un mejor realce. Por lo tanto este será el momento de finalizar el procesado de la imagen.

6.1.3. Filtro de suavizado basado en control borroso - SFCF

Cambiando un poco la naturaleza de los dos filtros anteriores el filtro de suavizado basado en control borroso ó SFCF (*Smoothing Fuzzy Control Filter*) tiene una naturaleza no iterativa aunque ciertamente está basado en el algoritmo IFCF. Para el diseño de este nuevo algoritmo se introdujeron las siguientes modificaciones al algoritmo IFCF:

- La regla ELSE (R_0) del algoritmo IFCF se reemplaza por esta otra para conseguir dotar al filtro SFCF de una mayor insensibilidad ante el ruido Gaussiano.

  R_0': ELSEIF (**fairly more** of x_i are **Z**) THEN y is $\overline{x_i}$

  donde,

  $$\overline{x_i} = \text{media}(x_i : x_i \in soporte(Z)) \tag{6.7}$$

  El término ”casi la mayoría” (**fairly more**) es un modificador lingüístico como lo era **more** en los algoritmos anteriores pero con una naturaleza más suave:

  $$\mu_{fairlymore}(z) = \frac{1}{1+\exp(-az+b)} \tag{6.8}$$

  Los valores típicos para los parámetros a y b son 6 y 3 respectivamente. Estos parámetros tratan de ajustar la función de forma que sea aproximadamente lineal en el rango de entrada. Con esto es con lo que se consigue el efecto de suavizado.

- El operador **min** utilizado en la ecuación (6.3) se remplaza por el operador **mediana**. En teoría esto sería capaz de procesar la imagen en una sola iteración [Farbiz00]. Con todo esto la salida del filtro (y) se computaría así:

  $$y = k \cdot \overline{x_i} + (1-k)\sum_{i=1}^{6} C_i \lambda_i \tag{6.9}$$

  donde

  $$k = \mu_{fairlymore}\left[\frac{\text{número de } x_i / x_i \in \text{soporte(Z)}}{\text{número total de } x_i}\right] \tag{6.10}$$

6.1.4. Filtro de suavizado y realce de bordes basado en control borroso - SSFCF

Si se añaden dos reglas extra al conjunto de reglas borrosas que se utilizan en el filtro SFCF, se obtiene un filtro que puede suavizar la imagen a la vez que realzar los bordes de la misma. Esto es importante ya que éstas son dos tareas inherentemente conflictivas como es de sobra conocido. El algoritmo que desarrolla estas ideas se conoce como filtro de afilado y suavizado basado en control borroso ó SSFCF (*Sharpening and Smoothing Fuzzy Control Filter*). Por lo tanto, y usando el mismo esquema que en los algoritmos anteriores, las nuevas reglas que se han de añadir son las siguientes:

R_7: IF (**more** of x_i are **Z**) & (**not few** of x_i are A_1) THEN y is B_1
R_8: IF (**more** of x_i are **Z**) & (**not few** of x_i are A_2) THEN y is B_2

donde el término ”poco” (**not few**) es una función borrosa tipo-S como la ecuación (6.8) pero con diferentes parámetros. Las funciones de pertenencia A_1**,** A_2**,** B_1 y B_2 son:.

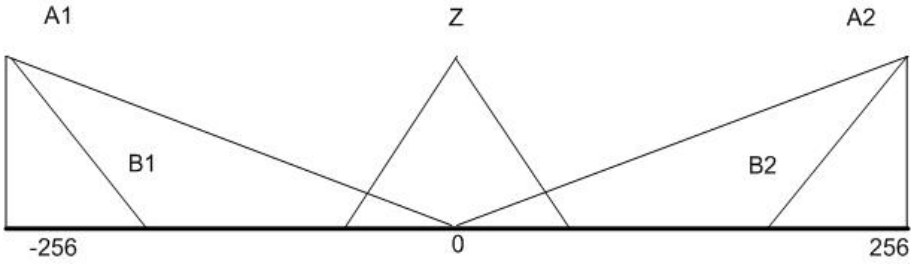


Figura 6.3: Funciones de pertenencia A_1**,** A_2**,** B_1**,** B_2

El grado de activación de la regla R_7, λ_7, se calcula según la ecuación (6.11), mientras que el grado de activación de la regla R_8 se calcula similarmente.

$$\begin{aligned}\lambda_7 = min[\mu_{A1}(x_i) : \mu_{A1} > \mu_Z(x_i)] \times \\ \mu_{more}\left[\frac{\text{número de } x_i \text{ / } \mu_Z(x_i) > \mu_{A1}(x_i)}{\text{número total de } x_i}\right] \times \\ \mu_{not\,few}\left[\frac{\text{número de } x_i \text{/ } \mu_{A1}(x_i) > \mu_Z(x_i)}{\text{número total de } x_i}\right]\end{aligned} \tag{6.11}$$

Finalmente la salida y se calcula:

$$y = k \cdot \overline{x_i} + (1-k)\left[\sum_{i=1}^{6} C_i\lambda_i + SC \cdot \sum_{i=1}^{2} C_{B_i}\lambda_{i+6}\right] \tag{6.12}$$

En la ecuación (6.12), el coeficiente de afilado ó SC (*Sharpening Coefficient*) puede modificar el grado de realce de los bordes de la imagen. La razón de que las reglas R_7 y R_8 hayan sido diseñadas de esta forma es para que estas reglas sólo se activen en las zonas de la imagen donde hay bordes. Así se asegura que sólo los bordes de la imagen sean realzados y no las zonas ruidosas.

6.2. Toolbox HPFborroso: filtrado paso alto borroso

6.2.1. Introducción

Una de las principales aplicaciones en el tratamiento de imágenes médicas es la *segmentación*. Para llevar a cabo una correcta segmentación es necesario disponer de un buen *detector de bordes*. En el apartado [**?**] se hacía una introducción a diferentes métodos de realce para la detección de bordes. Se veía como la primera derivada de una imagen podía dar información de la presencia de un borde, por lo tanto, el *operador gradiente* se utilizaba para la detección de bordes. Sin embargo este operador tiene el defecto de magnificar el ruido subyacente en la imagen, pues es altamente sensible a pequeñas fluctuaciones de luminancia. Un modo de solucionar este problema, será mediante operadores que simultáneamente realicen diferenciación en una dirección y promediado en la dirección ortogonal. Esta propiedad la presentan tanto los operadores de *Sobel* como el resto de operadores de vecindad (*Prewitt, Roberts o Freichen*), es decir, suavizan la imagen eliminando parte del ruido (ruido impulsivo), y por tanto, minimizan la aparición de falsos bordes debido al efecto de magnificación de ruido por parte de los operadores derivada [Pajares03].

El tipo de imágenes bajo estudio, imágenes procedentes de ultrasonidos, presentan ruido, por lo tanto a la hora de diseñar un filtro paso alto se descartará el operador gradiente sencillo. Los operadores de vecindad podrían ser adecuados, dada la propiedad de suavizado de ruido que poseen, pero para el caso de ecografías no es suficiente debido al ruido *speckle*. Por lo tanto se intentará implementar estos operadores siguiendo la base de funcionamiento del algoritmo de los filtros IFCF.

Es decir, se implementará un detector de bordes cuya característica principal sea el correcto funcionamiento para imágenes afectadas por ruido *speckle*, y cuyas bases fundamentales de diseño sean:

- operadores de vecindad → suavizado de ruido.
- algoritmo borroso IFCF → eliminación de ruido speckle.

6.2.2. Arquitectura del Algoritmo

Siguiendo la filosofía del filtrado iterativo basado en lógica de control borroso (IFCF), la estructura que adoptará el nuevo filtro se basará en el mecanismo típico *if-then-else*.

Se aplicará una máscara de tamaño 3*x*3

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

En lugar de aplicar las reglas borrosas a todos los píxeles a la vez como se hacía en el filtro IFCF, se harán cuatro subgrupos y a cada uno de ellos se le aplicarán las reglas correspondientes. Los subgrupos sería:

$$(z_1, z_2, z_3),\ (z_7, z_8, z_9),\ (z_1, z_4, z_7)\ y\ (z_3, z_6, z_9)$$

Las reglas borrosas que se proponen para este tipo de filtrado son las siguientes:

R_1: IF (**more** of z_i are **NB**) THEN y is **NB**
R_2: IF (**more** of z_i are **NM**) THEN y is **NM**
R_3: IF (**more** of z_i are **NS**) THEN y is **NS**
R_4: IF (**more** of z_i are **PS**) THEN y is **PS**
R_5: IF (**more** of z_i are **PM**) THEN y is **PM**
R_6: IF (**more** of z_i are **PB**) THEN y is **PB**
R_0: ELSE y is **Z**

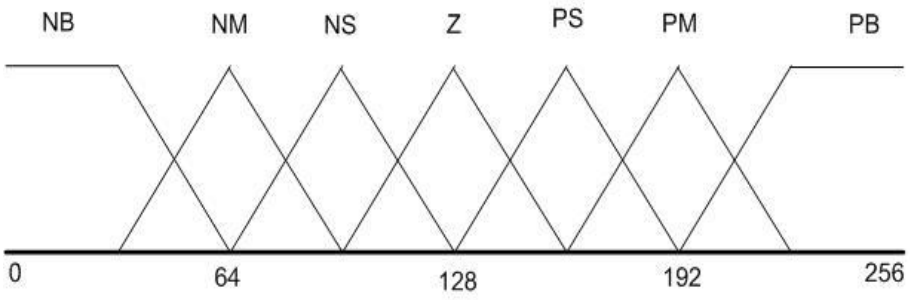


Figura 6.4: Funciones de pertenencia.

Así mismo, las funciones de pertenencia que se utilizan se pueden ver en la Fig. (6.4). Las funciones NM, NS, Z, PS y PM son de naturaleza triangular, mientras NB y PB son trapezoidales.

Se implementarán diferentes operadores de vecindad, siendo el algoritmo similar para todos ellos. La principal diferencia estará en las máscaras utilizadas.

Operador de Prewit

Como ya se veía en el capítulo 4 el operador de *Prewitt* viene dado por las máscaras mostradas en Fig. (6.5) .

Estas máscaras se identifican matemáticamente con:

$$Borde_{vertical} = \frac{1}{K+2}\left[(Z_1 + KZ_4 + Z_7) - (Z_3 + KZ_6 + Z_9)\right]$$

$$Borde_{horizontal} = \frac{1}{K+2}\left[(Z_1 + KZ_2 + Z_3) - (Z_7 + KZ_8 + Z_9)\right]$$

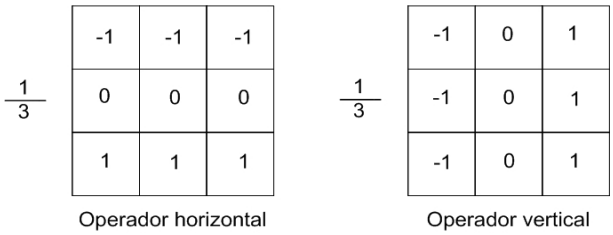


Figura 6.5: Máscaras del operador de Prewitt.

donde $K = 1$. En esta formulación los gradientes son normalizados para proporcionar ganancia unidad.

En el algoritmo desarrollado se distinguen las siguientes fases:

1. **Detección de bordes verticales**.

   Para ello se aplica la máscara correspondiente al operador vertical. El resultado que se obtiene se debe escalar de modo que el máximo valor de los bordes verticales detectados sea 255. En el caso de que no haya ningún borde vertical la salida será cero (imagen completamente negra).

2. **Detección de bordes horizontales**.

   El procedimiento a seguir es totalmente igual al anterior, solo que ahora se trabaja con la máscara horizontal.

3. **Bordes totales**.

   Una vez que se han hallado los bordes verticales y horizontales, la salida total que se corresponde con los bordes totales se calcula mediante:

$$Bordes_{totales} = \sqrt[2]{Bordes^2_{horizontales} + Bordes^2_{verticales}} \tag{6.13}$$

   La salida final también se escalará entre 0 y 255.

El modo de implementar en lógica borrosa cada una de las máscaras es el que se explica a continuación. Se mostrará solo el correspondiente al caso del operador vertical, ya que el horizontal funciona de modo similar. Se han de seguir los siguientes pasos:

1. Se toman los píxeles z_1, z_4 y z_7 y se les aplica cada una de las siete reglas obteniendo por salida y_1.

2. Lo mismo se hace con los píxeles z_3, z_6 y z_9 obteniendo y_2.

3. Por último se calcula y como $|y_1 - y_2|$.

El grado de actividad de cada una de las reglas permite determinar si las transiciones en cada uno de los píxeles se deben a la existencia de un borde o de ruido. El elemento más importante del algoritmo es la incorporación del operador **more**.

Se explicará con un ejemplo como actúa el algoritmo sobre una imagen.

- **Imagen sin ruido**

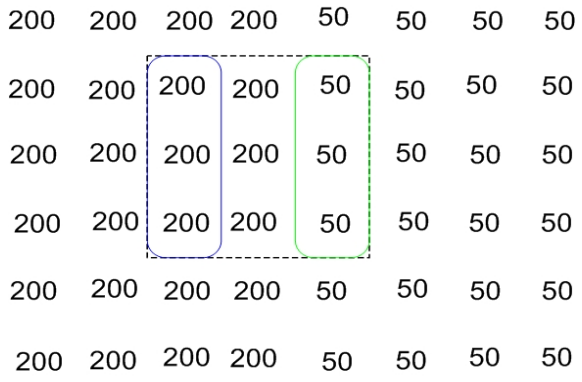


Figura 6.6: Imagen ausente de ruido: zona de transición

- **Borde**
  Todos los píxeles que intervienen en el cálculo de y_1 pertenecen a dos conjuntos borrosos, PM y PB. El primer paso será entonces calcular el grado de actividad de cada regla:

$$\begin{aligned} mu_{PM}(200) &= 0{,}75 \\ mu_{PB}(200) &= 0{,}25 \end{aligned} \tag{6.14}$$

  Por lo tanto:

$$\lambda_{PM} = min\{\mu_{PM}(x_i) : x_i \in soporte(PM)\} \times \mu_{more}\left[\frac{\text{número de } x_i / x_i \in \text{soporte(PM)}}{\text{número total de } x_i}\right]$$

$$\lambda_{PM} = 0{,}75 \times \mu_{more}\left[\frac{3}{3}\right] = 0{,}75 \times 1 = 0{,}75$$

$$\lambda_{PB} = min\{\mu_{PB}(x_i) : x_i \in soporte(NB)\} \times \mu_{more}\left[\frac{\text{número de } x_i / x_i \in \text{soporte(PB)}}{\text{número total de } x_i}\right]$$

$$\lambda_{PB} = 0{,}25 \times \mu_{more}\left[\frac{3}{3}\right] = 0{,}25 \times 1 = 0{,}25$$

$$\begin{aligned}\lambda_Z &= max(0,1-\lambda_{PM}-\lambda_{PB}) \\ \lambda_Z &= max(0,1-0{,}75-0{,}25)=0\end{aligned}$$

Entonces:

$$\begin{aligned}y_1 &= \frac{192\times 0{,}75\times 64+224\times 0{,}25\times 64}{0{,}75\times 64+0{,}25\times 64} \\ y_1 &= \frac{9218+3584}{48+16}=\frac{12800}{64}=200\end{aligned}$$

Siguiendo el mismo procedimiento para el cálculo de y_2 se obtiene:

$$\begin{aligned}mu_{NB}(50)=0{,}44 \\ mu_{NM}(50)=0{,}56\end{aligned}$$

$$\begin{aligned}\lambda_{NB} &= 0{,}44 \\ \lambda_{PB} &= 0{,}56 \\ \lambda_Z &= 0\end{aligned}$$

$$y_1 = \frac{0\times 0{,}44\times 64+64\times 0{,}56\times 64}{0{,}44\times 64+0{,}56\times 64}=35{,}84$$

Por lo tanto

$$y=|y_1-y_2|=|200-35{,}84|=164{,}16\simeq 164$$

- **Ausencia de borde**
  Siguiendo los mismos pasos que en el caso anterior:

$$y=|y_1-y_2|=|200-200|=0$$

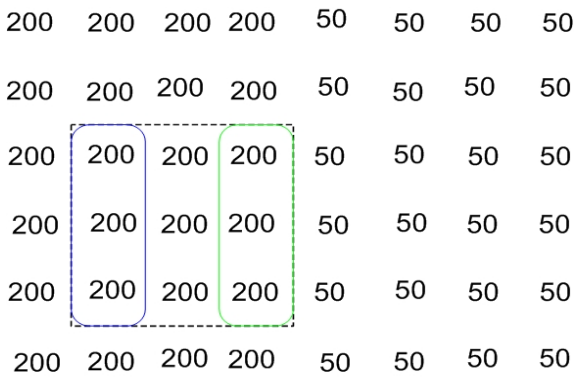


Figura 6.7: Imagen ausente de ruido: zona uniforme

Si se recorren todos los píxeles de la matriz, aplicando la máscara en cada uno de ellos como se ha explicado, se obtiene la imagen resultante. Esta sería la que se puede ver en Fig. (6.8).

Figura 6.8: Bordes detectados en una imagen ausente de ruido

- **Imagen ruidosa**
  - **Borde**

En el cálculo de y_1 se ve que dos de los píxeles pertenecen a los conjuntos borrosos, PM y PL, mientras que el otro debido a la presencia de ruido pertenece al conjunto NS. Realizando las operaciones pertinentes:

$$\begin{aligned} mu_{PM}(200) &= 0{,}75 \\ mu_{PB}(200) &= 0{,}25 \\ mu_{NB}(32) &= 1 \end{aligned}$$

200	200	200	200	50	50	50	50
200	200	200	200	50	50	160	50
200	200	200	32	50	50	50	50
200	200	200	200	50	50	50	50
200	200	200	200	50	50	50	50
200	200	200	200	50	50	50	50

Figura 6.9: Imagen ruidosa

Por lo tanto:

$$\lambda_{PM} = min\{\mu_{PM}(x_i) : x_i \in soporte(PM)\} \times \mu_{more}\left[\frac{\text{número de } x_i / x_i \in \text{soporte(PM)}}{\text{número total de } x_i}\right]$$

$$\lambda_{PM} = 0{,}75 \times \mu_{more}\left[\frac{2}{3}\right] = 0{,}75 \times 0{,}8716 = 0{,}6537$$

$$\lambda_{PB} = min\{\mu_{PB}(x_i) : x_i \in soporte(PB)\} \times \mu_{more}\left[\frac{\text{número de } x_i / x_i \in \text{soporte(PB)}}{\text{número total de } x_i}\right]$$

$$\lambda_{PB} = 0{,}25 \times \mu_{more}\left[\frac{2}{3}\right] = 0{,}25 \times 0{,}8716 = 0{,}2179$$

$$\lambda_{NB} = min\{\mu_{NB}(x_i) : x_i \in soporte(NB)\} \times \mu_{more}\left[\frac{\text{número de } x_i / x_i \in \text{soporte(NB)}}{\text{número total de } x_i}\right]$$

$$\lambda_{NB} = 1 \times \mu_{more}\left[\frac{1}{3}\right] = 1 \times 0{,}1114 = 0{,}1114$$

$$\lambda_Z = max(0, 1 - \lambda_{PM} - \lambda_{PB} - \lambda_{NB})$$
$$\lambda_Z = max(0, 1 - 0{,}6537 - 0{,}2179 - 0{,}1114) = 0{,}017$$

Entonces:

$$y_1 = \frac{0 \times 0{,}1114 \times 64 + 128 \times 0{,}017 \times 64 + 192 \times 0{,}6537 \times 64 + 224 \times 0{,}2179 \times 64}{0{,}1114 \times 64 + 0{,}017 \times 64 + 0{,}6537 \times 64 + 0{,}2179 \times 64}$$

$$y_1 = \frac{139{,}26 + 8032{,}66 + 3123{,}81}{7{,}129 + 1{,}088 + 41{,}84 + 13{,}95}$$

$$y_1 = \frac{11295{,}73}{64} = 176{,}47$$

Del mismo modo, para y_2 resulta:

$$y_2 = 35{,}84$$

Por lo tanto:

$$y = |y_1 - y_2| = |176{,}47 - 35{,}84| = 140{,}65 \simeq 140$$

- **Ausencia de borde**

  Ahora se analizará el caso de una zona uniforme ruidosa. En concreto, se verá como afecta el píxel de valor 160 en aquella zona donde todos los píxeles deberían de tener un valor de 50.
  Para el cálculo de y_1 el ruido no influye:

$$y_1 = 35{,}84 \tag{6.15}$$

  Para el cálculo de y_2

$$\begin{aligned} mu_{NB}(50) &= 0{,}44 \\ mu_{NM}(50) &= 0{,}56 \\ mu_{PS}(160) &= 1 \end{aligned} \tag{6.16}$$

  Por lo tanto:

$$\lambda_{NB} = 0{,}44 \times \mu_{more}\left[\frac{2}{3}\right] = 0{,}44 \times 0{,}8716 = 0{,}3335$$

$$\lambda_{NM} = 0{,}56 \times \mu_{more}\left[\frac{2}{3}\right] = 0{,}56 \times 0{,}8716 = 0{,}4380$$

$$\lambda_{PS} = 1 \times \mu_{more}\left[\frac{1}{3}\right] = 1 \times 0{,}1114 = 0{,}1114$$

$$\begin{aligned} \lambda_Z &= max(0, 1 - \lambda_{NB} - \lambda_{NM} - \lambda_{PS}) \\ \lambda_Z &= max(0, 1 - 0{,}3335 - 0{,}4380 - 0{,}1114) = 0{,}1171 \end{aligned}$$

$$y_2 = \frac{0 \times 0{,}3335 \times 64 + 64 \times 0{,}4380 \times 64 + 160 \times 0{,}1114 \times 64 + 128 \times 0{,}1171 \times 64}{0{,}3335 \times 64 + 0{,}4380 \times 64 + 0{,}1114 \times 64 + 0{,}1171 \times 64}$$

$$y_2 = \frac{1734{,}048 + 1140{,}736 + 953{,}2832}{21{,}344 + 28{,}032 + 7{,}1296 + 7{,}4944}$$

$$y_1 = \frac{3834{,}0672}{64} = 53{,}90$$

$$y = |y_1 - y_2| = |35{,}84 - 53{,}90| = 18{,}06 \simeq 18$$

Recorriendo todos los píxeles de la imagen, se obtiene:

0	0	0	164	164	18	0	18
0	0	0	164	164	18	0	18
0	0	23	164	140	18	0	18
0	0	23	164	140	0	0	0
0	0	23	164	140	0	0	0
0	0	0	164	164	0	0	0

Figura 6.10: Bordes detectados en una imagen ruidosa

Hemos visto como afecta el ruido en la detección de bordes en una imagen. A continuación veremos cual sería el resultado de aplicar el operador de *Prewitt* convencional a la misma imagen ruidosa.

Comparando los resultados obtenidos, se observa que el detector de bordes borroso presenta un mejor comportamiento ante el ruido, pues los errores que presenta son menores. En el capitulo 7 se mostrarán los resultados obtenidos al aplicar el algoritmo a imágenes reales, pudiéndose hacer entonces una mejor valoración del filtro diseñado.

Operador de *Sobel*

El operador de *Sobel* viene dado por:

$$Borde_{vertical} = \frac{1}{K+2}\left[(Z_1 + KZ_4 + Z_7) - (Z_3 + KZ_6 + Z_9)\right]$$

$$Borde_{horizontal} = \frac{1}{K+2}\left[(Z_1 + KZ_2 + Z_3) - (Z_7 + KZ_8 + Z_9)\right]$$

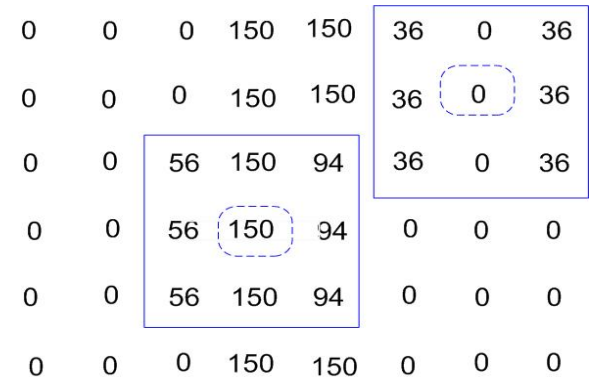


Figura 6.11: Bordes detectados en una imagen ruidosa según el operador Prewitt clásico

donde K=2.

Vemos que la diferencia respecto al operador de *Prewitt* está en que el valor de los píxeles z_2, z_4, z_6 y z_8 se duplica, lo que equivale a que cada uno de estos píxeles se considere dos veces. El motivo de utilizar estos pesos es el de dar igual importancia a cada pixel en términos de su contribución al gradiente espacial. Como consecuencia de ello, aparece la principal diferencia entre *Sobel* y *Prewitt*: *Sobel* es más sensible a transiciones diagonales, mientras que *Prewitt* lo es a los bordes horizontales y verticales.

Las máscaras correspondientes al operador de *Sobel* son (ver Fig. 6.12):

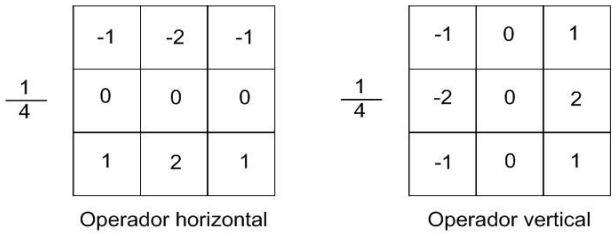


Figura 6.12: Máscaras del operador de Sobel.

El algoritmo que implementa este operador en su versión borrosa presenta las mismas tres fases que el operador de *Prewitt*, es decir, detección de bordes verticales, detección de bordes horizontales y cálculo de los bordes totales.

Se han desarrollados dos versiones del operador de *Sobel*, las cuales se explican a continuación.

1. ***Sobel***

   En esta implementación aplicar la máscara vertical se consigue mediante las siguientes operaciones:

a) Se toman de nuevo los píxeles z_1, z_4 y z_7 , pero ahora el píxel z_4 cuenta doble, es decir, es como si en lugar de tres píxeles se tuviesen cuatro: z_1, z_4, z_4 y z_7 . Se les aplican las siete reglas obteniendo por salida y_1.

b) Lo mismo se hace con los píxeles z_3, z_6 y z_9 , duplicando z_6. La salida es y_2.

c) Finalmente se calcula $y = |y_1 - y_2|$.

2. ***Sobel2***

   Esta versión también se desarrolla en tres fases:

   a) Se tiene los píxeles z_1,z_4 y z_7. El modo de reflejar ahora $2z_4$ se consigue duplicando la intensidad de este píxel. Al igual que siempre, se les aplican las siete reglas obteniendo por salida y_1.

   b) Se actúa de modo similar para z_3,z_6 y z_9 duplicando z_6. Se obtiene y_2.

   c) De nuevo, $y = |y_1 - y_2|$.

En el capítulo 7 se mostrarán los resultados obtenidos al aplicar el algoritmo a imágenes reales.

6.2.3. Otras opciones planteadas

Finalmente se van a explicar otras posibilidades que se han estudiado para la implementación del filtro paso alto borroso y que al final no han sido tenidas en cuenta ya que los resultados que generaban no eran lo suficientemente buenos.

Cálculo de la salida final

Un primer punto a discutir fue como hallar la salida final, es decir, el cálculo de los bordes totales. Se plantearon tres opciones:

1. Combinar el operador horizontal y vertical dando lugar a solo un operador y por lo tanto a una única máscara. Esta máscara sería la suma del operador vertical y horizontal (Fig. 6.13). Esta opción resultó ser no válida, ya que al calcular los grados de actividad de las diferentes reglas se aplica el operador “more”, y éste es no lineal.

2. Aplicar la fórmula:

$$Bordes_{totales} = |Bordes_{horizontales}| + |Bordes_{verticales}| \tag{6.17}$$

   Esta idea tampoco se adoptó ya que los resultados obtenidos fueron peores tal y como se señalaba en [Pratt91].

3. Finalmente como ya se ha comentado anteriormente la salida final se calcula como:

$$Bordes_{totales} = \sqrt[2]{Bordes^2_{horizontales} + Bordes^2_{verticales}} \tag{6.18}$$

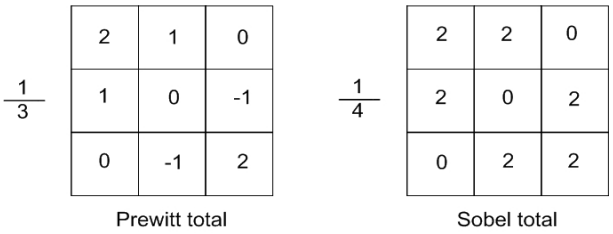


Figura 6.13: Máscaras únicas de los operadores de Prewit y Sobel

Otros operadores

Cuando se hablaba de los filtros IFCF aparecían diferentes operadores: "la mayoria" (**more**), "casi la mayoría" (**fairly more**) y "no poco"(**not few**). Todos estos operadores se aplicaban sobre nueve píxeles. Sin embargo en el filtro diseñado solo actuarían sobre tres píxeles o a lo sumo cuatro, por lo tanto carecería de sentido hablar de "casi la mayoria" o "no poco" cuando se tiene un número tan pequeño de píxeles. De ahí que el nuevo filtro se modele siguiendo los principios del filtro IFCF y no de SFCF o SSFCF.

Otros tamaños de máscaras

Se ha elegido trabajar con máscaras 3*x*3, pero también se estudiaron las opciones de trabajar con máscaras de diferentes tamaños.

1. Máscaras 2*x*2

   Sin duda se trata del operador de Roberts. Este se descartó ya que una máscara tan pequeña no permitía el suavizado del ruido.

2. Máscaras mayores

   Uno de los principales problemas del filtro es el de encontrar bordes en ambientes muy ruidosos, por lo tanto se pensó en aumentar el tamaño de las máscaras. Al aumentar el número de píxeles vecinos el suavizado sería mayor y el ruido debería afectar menos. Sin embargo, los resultados experimentales demostraron que se comportaba mal en los bordes, por lo tanto se optó por mantener la máscara de tamaño 3*x*3.

6.2.4. Posibles mejoras

En este apartado se comentarán algunas posibilidades que no han sido analizadas, pero que podrían suponer una mejora del algoritmo. Serán lineas futuras de investigación los siguientes casos:

Establecimiento de un umbral

Cuando se habla de operadores de *Sobel* o de *Prewitt* se está hablando de operadores de primera derivada. Normalmente cuando se trabaja con este tipo de operadores, el resultado que se obtiene tras aplicar la máscara se somete a un umbral que determina si tenemos o no un borde. Se estaría binarizando la imagen, valor 1 si supera el umbral y 0 en caso contrario. En el algoritmo desarrollado no se ha implementado, pero podría resultar interesante. Para ello será necesario determinar primero el umbral adecuado.

Determinación de los conjuntos borrosos

Como se ha visto el punto de partida del algoritmo es la definición de unos conjuntos borrosos sobre los cuales se van a realizar todas las operaciones. En la implementación realizada del filtro paso alto, se permite variar en cierta medida estos conjuntos. Para ello es necesaria la introducción a través de la linea de comandos de un llamado '*factor de escala*'. Como se verá en el capítulo 7, según cual sea la imagen a tratar se utilizarán unos conjuntos borrosos u otros. Sería interesante hacer un estudio mediante el cual se pudiesen definir los conjuntos borrosos más adecuados para cada imagen, en función de las características de las mismas.

6.3. Toolbox Anisótropo: filtrado anisótropo borroso

6.3.1. Introducción

Cuando se desarrolla un filtro para el procesado de imágenes médicas, aspectos como que las imágenes resultantes estén difusas o borrosas son intolerables. Por consiguiente a la hora de diseñar un filtro se deben perseguir los siguientes ideales:

- Minimizar la pérdida de información, preservando las diferencias entre regiones y los bordes de las estructuras.
- Eliminación de ruido en las regiones homogéneas de manera eficiente.
- Aumentar la definición de las diversas estructuras marcando las discontinuidades existentes.

De entre las técnicas de filtrado espacial, aquellas que han satisfecho de manera más eficaz los anteriores criterios son las basadas en la difusión anisotrópica [Perona90]. Por lo tanto en el desarrollo de este proyecto también nos vamos a preocupar de este tipo de filtrado.

En la sección anterior se explicaba como se hacía uso de la lógica borrosa y del principio de funcionamiento de los filtros IFCF para el desarrollo de diferentes filtros paso alto. El objetivo ahora es el diseño de un filtro de difusión anisótropo para el tratamiento de imágenes captadas por ultrasonidos, por lo tanto tal y como se ha explicado anteriormente, se tomará como base fundamental de diseño el filtrado IFCF.

6.3.2. Arquitectura del algoritmo

Objetivos

En el capítulo 4 se hacía una introducción al filtrado borroso anisótropo extendiéndolo al tratamiento de imágenes. En él se tenía como base la ecuación de difusión del calor (4.6.2), y en ella aparecía un coeficiente llamado *coeficiente de difusión*, que junto al *gradiente* permitía definir los diversos flujos locales (4.46). Este coeficiente se definía como una función del módulo del *gradiente*, que debía ser monótona decreciente para cumplir ciertos requisitos.

Pero el algoritmo de difusión anisótropa definido de esta manera tenía una serie de problemas. El principal problema es que se necesita una gran cantidad de iteraciones para alcanzar el estado estacionario. Esto significa un tiempo de procesado elevado y una influencia importante del efecto de bordes. Además, en entornos muy ruidosos, la imagen resultante tras un gran número de iteraciones no tiene una apariencia muy real.

Otro problema que aparece en esta clase de filtros está en la correcta elección de una función para calcular la constante de difusión.

Por ello surge la idea de definir un nuevo filtro de suavizado, que preserve los bordes, controlado por una serie de reglas borrosas. El algoritmo que se va a utilizar para el filtrado de las imágenes ya está definido: el algoritmo de difusión anisótropa propuesto por [Gerig]. Y es sobre éste donde se va a tratar de aplicar la teoría de lógica borrosa. Los objetivos de esta innovación son los de reducir el número de iteraciones, mejorar los resultados obtenidos con la lógica clásica y permitir tener un mayor control sobre el proceso de difusión, evitando así llegar a resultados no deseados.

La idea de este nuevo método se basa en calcular el *coeficiente de difusión* mediante un sistema de inferencia borrosa en vez de calcularlo de forma exacta con una función dependiente del *gradiente*.

Coeficientes de Difusión borrosos

En este apartado se va a indicar cual es la estructura definitiva que tendrá el filtro propuesto. Como ya se ha comentado, en vez de calcular el *coeficiente de difusión* de forma exacta a través del *gradiente*, lo que se quiere es hacer una aproximación borrosa de dicho coeficiente, para así poder controlar mejor el proceso de difusión. Por tanto la salida del sistema borroso propuesto será tal coeficiente, definiéndose entonces una variable lingüística de salida llamada *coeficiente de difusión*.

En el apartado 4.6.2 se ha visto que una imagen 2D se puede ver como una colección de píxeles (una matriz), y que cada píxel viene caracterizado por un nivel de gris. Como el *coeficiente de difusión* es función del *gradiente*, y éste se puede definir de forma aproximada a partir de las diferencias entre los píxeles en una dirección determinada, entonces, el *coeficiente de difusión* se puede modelar como un proceso de inferencia borrosa en el que se tengan en cuenta únicamente tales diferencias entre píxeles. Por ello, se debe definir una variable lingüística de entrada que represente esa diferencia, a la que se llamará *diferencia de luminancias*.

Se han diseñado dos modos diferentes de calcular este *coeficiente de difusión* dando lugar a los dos filtros siguientes: anisotropo_log y anisotropo_doble. Estos se explican a continuación.

1. **Anisotropo_log**

   Se parte de una ventana de tamaño 3×3, centrada en el píxel de estudio. A partir de esta ventana se obtienen las distancias $D_{1i} = log(|z_5 - z_i|)$, que constituirán las entradas del sistema de inferencia borrosa, el cual, a partir de ellas, debe permitir obtener el *coeficiente de difusión*.

   Al igual que se hacía en el filtro paso alto (apartado 6.2), en lugar de aplicar las reglas borrosas a todos las distancias D_{1i} a la vez como en el caso de los filtros IFCF,

se harán cuatro subgrupos. Cada subgrupo intervendrá en el cálculo del *coeficiente de difusión* correspondiente a cada uno de los cuatro flujos (norte, sur, este, oeste), y a cada uno de ellos se le aplicarán las reglas borrosas correspondientes. Estas expresiones permiten ya expresar los coeficientes en función de los píxeles y por tanto de las distancias correspondientes:

$$\begin{aligned} C_e &= C_e(z_6, z_5, z_9, z_3) = C_e(D_{16}, D_{19}, D_{13}) \\ C_w &= C_w(z_5, z_4, z_7, z_1) = C_w(D_{14}, D_{17}, D_{11}) \\ C_n &= C_n(z_8, z_5, z_9, z_7) = C_n(D_{18}, D_{19}, D_{17}) \\ C_s &= C_s(z_5, z_2, z_3, z_1) = C_s(D_{12}, D_{13}, D_{11}) \end{aligned} \tag{6.19}$$

Una vez conocidas las variables necesarias, se debe tener claro cuales son los valores entre los que pueden variar las variables lingüísticas. En el caso de la variable *coeficiente de difusión* es sencillo determinar que su valor estará comprendido en el rango $[0,1]$, ya que no debe ser negativa, y si superase la unidad daría lugar a resultados inestables. Considerando que se tienen 256 niveles de gris, el rango de la otra variable es fácil de determinar: $[0,1]$.

Definidas las variables de entrada y salida y sus respectivos rangos, el primer paso es seleccionar aquellos *estados lingüísticos* más significativos para cada variable y expresarlos mediante conjuntos borrosos apropiados. Para la variable lingüística *diferencia de luminancias* se van a definir ocho conjuntos borrosos con funciones de pertenencia pseudotrapezoidales, que van a estar centrados aproximadamente en los siguientes valores: 0, 0.15, 0.3, 0.45, 0.6, 0.75, 0.8 y 0.85. Dichos conjuntos borrosos se representan en Fig. (6.14).

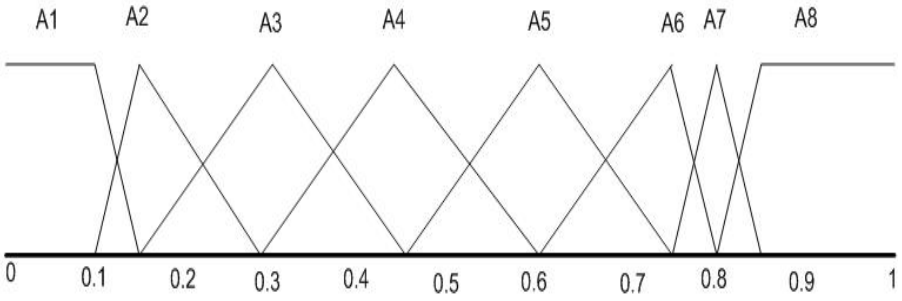


Figura 6.14: Conjuntos borrosos de la variable *diferencia entre luminancias*.

Ahora ya solo falta por definir los conjuntos borrosos que representarán a la variable *coeficiente de difusión*. Como en el caso anterior se tienen ocho conjuntos borrosos. Dichos conjuntos son escogidos de modo que sus centroides sean: 0.025, 0.05, 0.075, 0.1, 0.2, 0.4, 0.6 y 0.8. Se muestran en Fig. (6.15).

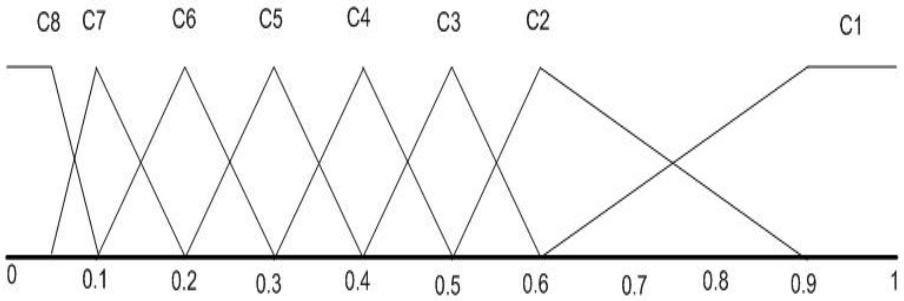


Figura 6.15: Conjuntos borrosos de la variable lingüística *coeficiente de difusión*.

Una vez definidos los conjuntos borrosos que van a representar a las dos variables, el siguiente paso es determinar la base de reglas. Se hará uso de directivas del tipo:

SI la diferencia entre los niveles de gris de los píxeles es pequeña, ENTONCES el coeficiente de difusión es alto.

La idea es usar la base de reglas borrosas para permitir una mayor difusión en las zonas homogéneas, donde la diferencia entre los píxeles es pequeña, y ser más restrictivos en aquellas zonas donde esta diferencia sea mayor, que se corresponderán con los bordes. Las reglas borrosas que se proponen para este tipo de filtrado son las siguientes:

R_1: IF (**more** of D_{1i} are A_1) THEN C is C_1
R_2: IF (**more** of D_{1i} are A_2) THEN C is C_2
R_3: IF (**more** of D_{1i} are A_3) THEN C is C_3
R_4: IF (**more** of D_{1i} are A_4) THEN C is C_4
R_5: IF (**more** of D_{1i} are A_5) THEN C is C_5
R_6: IF (**more** of D_{1i} are A_6) THEN C is C_6
R_7: IF (**more** of D_{1i} are A_7) THEN C is C_7
R_8: IF (**more** of D_{1i} are A_8) THEN C is C_8

Ya solo falta realizar el proceso de desborrificación, que permitirá obtener el valor real de *coeficiente de difusión* con el que se va a trabajar. Siguiendo la filosofía del filtrado IFCF::

$$\lambda_1 = min\{\mu_{A_1}(D_i) : D_{1i} \in soporte(A_1)\} \times \mu_{more}\left[\frac{\text{número de } D_{1i} \text{ / } D_{1i} \in \text{soporte}(A_1)}{\text{número total de } D_{1i}}\right] \tag{6.20}$$

$$C = \frac{\sum_{i=1}^{8} C_i w_i \lambda_i}{\sum_{i=1}^{8} w_i \lambda_i} \tag{6.21}$$

2. **Anisotropo_doble**

En el filtro anterior, el cálculo del *coeficiente de difusión* se hacía en base de la variable *diferencias de luminancias*, D_{1i}, sobre la cual se aplicaban las diferentes reglas borrosas. Ahora, con el objetivo de disminuir los efectos del ruido, se va a definir una nueva variable, *diferencias de luminancias 2*, que se define como:

$$\begin{aligned} D_{2e} &= log(|z_3 - z_9|) \\ D_{2w} &= log(|z_1 - z_7|) \\ D_{2n} &= log(|z_7 - z_9|) \\ D_{2s} &= log(|z_1 - z_3|) \end{aligned} \tag{6.22}$$

El rango en el que puede variar esta nueva variable lingüística es también $[0,1]$ y los conjuntos difusos son los mismos que para D_{1i}.

El sistema usará reglas del tipo:

$$IF\ (\textbf{more}\ of\ D_{1i}\ are\ A_1)\ \ AND\ \ D_2\ is\ A_1\ THEN\ \ \mathrm{C}\ \ is\ \ C_1$$

Para el caso que nos ocupa, se quiere modelar una difusión rápida para diferencias pequeñas y una restricción fuerte a la difusión cuando la diferencia es lo suficientemente grande. Considerando los rangos de valores que pueden tomar D_1 y D_2 y teniendo en cuenta el tipo de difusión que se pretende conseguir, se puede construir una matriz 8*x*8 que constituirá la base de reglas del sistema propuesto (ver Tabla 2).

$D1 \mid D2$	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
A_1	C_1	$*$	$*$	$*$	$*$	$*$	$*$	$*$
A_2	$*$	C_2	$*$	$*$	$*$	$*$	$*$	$*$
A_3	$*$	$*$	C_3	$*$	$*$	$*$	$*$	$*$
A_4	$*$	$*$	$*$	C_4	$*$	$*$	$*$	$*$
A_5	$*$	$*$	$*$	$*$	C_5	$*$	$*$	$*$
A_6	$*$	$*$	$*$	$*$	$*$	C_6	$*$	$*$
A_7	$*$	$*$	$*$	$*$	$*$	$*$	C_7	$*$
A_8	$*$	$*$	$*$	$*$	$*$	$*$	$*$	C_8

Cuadro 6.1: Base de reglas del filtro anisotropo_doble

El proceso de desborrosificación que permitirá obtener el valor real del *coeficiente de difusión* se trata de nuevo de un sistema SAM [Kosko]. La función que permitirá obtener el coeficiente será la siguiente:

$$\lambda_1 = min\{\mu_{A_1}(D_{1i}) : D_{1i} \in soporte(A_1)\} \times \mu_{more}\left[\frac{\text{número de } D_{1i} \text{ / } D_{1i} \in \text{soporte(A}_1\text{)}}{\text{número total de } D_{1i}}\right] \tag{6.23}$$

$$C_j = \frac{\sum_{i=1}^{8} C_i w_i \lambda_i \mu_{A_i}(D_{2j})}{\sum_{i=1}^{8} w_i \lambda_i \mu_{A_i}(D_{2j})} \tag{6.24}$$

donde j puede ser norte, sur, este y oeste.

6.3.3. Algoritmo y ejemplo

Después de ver la estructura del sistema borroso que permite el cálculo del *coeficiente de difusión*, en este apartado se explica detalladamente el algoritmo que sigue el filtro implementado. Además se muestra con un ejemplo la obtención del *coeficiente de difusión* mediante inferencia borrosa. Se verá el caso del filtro *anisotropo_doble* dado que es el más complejo a la hora de obtener el *coeficiente de difusión*, pero una vez obtenido éste, los pasos a seguir son los mismos para los dos filtros.

Se parte de la imagen que se quiere realzar. Esta imagen se puede ver como una matriz de píxeles donde cada píxel tiene un nivel de gris asociado (el valor 0 se correspondería con el color negro y 255 con el blanco). Para cada píxel de la imagen se seguirán los siguientes pasos:

1. Se coge una ventana 3*x*3 centrada en el pixel de estudio. En este primer paso, parece haber un problema con los píxeles situados en las primeras y últimas filas y columnas de la imagen (los situados en los bordes de la imagen). Para resolverlo, simplemente se añaden dos filas, una al principio y una al final, idénticas a la primera y última fila respectivamente, y lo mismo con las columnas. Y en cada una de las cuatro esquinas que faltan se añade un valor, que puede ser una interpolación de los que tiene al lado, o por ejemplo se ponen a negro (valor 0). Si la imagen tenía tamaño $M \times N$ ahora pasa a tamaño $(M+2) \times (N+1)$ (Fig. (6.16)).

2. Se calculan las diferencias entre píxeles D1 y D2 para cada orientación. Por ejemplo, en la orientación *este* se tendría:

$$\begin{aligned} D_{13} &= log(|z_3 - z_5|) \\ D_{16} &= log(|z_3 - z_6|) \\ D_{19} &= log(|z_3 - z_9|) \end{aligned}$$

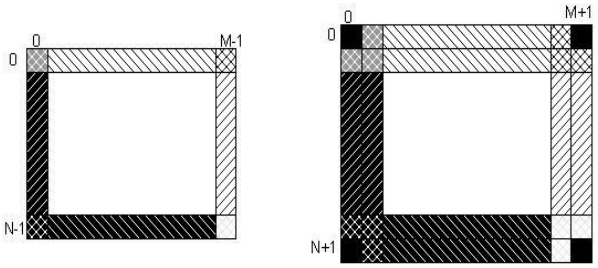


Figura 6.16: Forma de resolver el problema de los bordes.

$$D_{2e} = log(|z_5 - z_9|)$$

3. Se aplica el razonamiento borroso para obtener el *coeficiente* en cada orientación.
4. Se calculan los flujos locales para cada orientación.

   Por ejemplo:

   $$\Phi_e = [C_e(I(x+\Delta x, y, t) - I(x, y, t))]$$

5. Se genera la nueva imagen:

   $$I(x, y, t+\Delta t) = I(x, y, t) + \Delta t(\Phi_e - \Phi_w + \Phi_n - \Phi_s)$$

6. Vuelta al primer punto para una nueva iteración.

El algoritmo es similar al que se tiene en el caso de filtrado anisótropo normal. Sólo se diferencia en el cálculo del *coeficiente de difusión*. En el caso de filtrado normal, hay que calcular los *gradientes* requeridos para cada orientación, y a partir de ellos y a través de una función dada se calculan de forma exacta los coeficientes correspondientes.

A continuación se muestra un ejemplo del cálculo del *coeficiente de difusión* mediante inferencia borrosa (Fig. (6.17)).

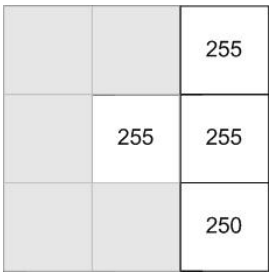


Figura 6.17: Píxeles que intervienen en el cálculo del *coeficiente de difusión este*.

Se calcula el valor de las *diferencias de luminancias*:

$$\begin{aligned} D_{13} &= log(|z_3 - z_5|) = 0 \\ D_{16} &= log(|z_3 - z_6|) = 0 \\ D_{19} &= log(|z_3 - z_9|) = 0{,}29 \\ D_{2e} &= log(|z_3 - z_9|) = 0{,}29 \end{aligned}$$

Según la base de regla de la Tabla 2 se tienen las siguientes reglas:

IF **more** *of* D_{1i} *are* A_2*, and* D_2 *is* A_2 *THEN C is* C_2

IF **more** *of* D_{1i} *are* A_3*, and* D_2 *is* A_3 *THEN C is* C_3

Se calcula el grado de actividad de cada regla:

$$\begin{aligned} mu_{A_2}(D_{13}) &= mu_{A_2}(0) = 0 \\ mu_{A_3}(D_{13}) &= mu_{A_3}(0) = 0 \\ mu_{A_2}(D_{15}) &= mu_{A_2}(0) = 0 \\ mu_{A_3}(D_{15}) &= mu_{A_3}(0) = 0 \\ mu_{A_2}(D_{19}) &= mu_{A_2}(0{,}29) = 0{,}0667 \\ mu_{A_3}(D_{19}) &= mu_{A_2}(0{,}29) = 0{,}0933 \\ mu_{A_2}(D_2) &= mu_{A_2}(0{,}29) = 0{,}0667 \\ mu_{A_3}(D_2) &= mu_{A_2}(0{,}29) = 0{,}0933 \end{aligned}$$

Por lo tanto:

$$\lambda_{A_2} = min\{\mu_{A_2}(D_{1i}) : D_{1i} \in soporte(A_2)\} \times \mu_{more}\left[\frac{\text{número de } D_{1i}/D_{1i} \in \text{soporte(A}_2)}{\text{número total de } D_{1i}}\right]$$

$$\lambda_{A_2} = 0{,}0667 \times \mu_{more}\left[\frac{1}{3}\right] = 0{,}0667 \times 0{,}1114 = 0{,}0743$$

$$\lambda_{A_3} = min\{\mu_{A_3}(D_{1i}) : D_{1i} \in soporte(A_3)\} \times \mu_{more}\left[\frac{\text{número de } D_{1i}/D_{1i} \in \text{soporte(A}_3)}{\text{número total de } D_{1i}}\right]$$

$$\lambda_{A_3} = 0{,}9333 \times \mu_{more}\left[\frac{1}{3}\right] = 0{,}9333 \times 0{,}1114 = 0{,}103369$$

Entonces:

$$
\begin{aligned}
C_e &= \frac{\sum_{i=1}^{8} C_i w_i \lambda_i \mu_{A_i}(D_{2j})}{\sum_{i=1}^{8} w_i \lambda_i \mu_{A_i}(D_{2j})} \\
C_e &= \frac{0{,}6 * 0{,}2 * 0{,}07430 * 0{,}0667 + 0{,}5 * 0{,}1 * 0{,}1033 * 0{,}9333}{0{,}2 * 0{,}07430 * 0{,}0667 + 0{,}1 * 0{,}1033 * 0{,}9333} \\
C_e &= \frac{0{,}00059470 + 0{,}0048}{0{,}00099116 + 0{,}0096} \\
C_e &= \frac{0{,}0054}{0{,}0106} = 0{,}5094
\end{aligned}
$$

6.3.4. Otras opciones planteadas

Hasta ahora se han explicado las características de los filtros que se recogen en el *Toolbox Anisótropo,* pero han sido muchas otras las opciones estudiadas que por falta de resultados satisfactorios se han descartado. A continuación se explican brevemente las más importantes:

Cálculo del coeficiente: conjuntos no logarítmicos

En la sección anterior se explicaba como calcular el *coeficiente de difusión*. Este se calculaba en función de unas distancias definidas como el logaritmo en base 255 de la diferencia de intensidad entre determinados píxeles, por lo cual el rango de estas variables lingüísticas era $[0,1]$ y los conjuntos borrosos se definían según Fig. (6.20).

Al principio del diseño del algoritmo se pensó en trabajar con estas distancias definiéndolas como los valores absolutos de la diferencias de intensidades entre píxeles. Por lo tanto el rango de estas variables era $[0,255]$ y los conjuntos borrosos utilizados se definían también en este rango. Por medio de inferencia borrosa se obtenía una variable, *dif*, en función de la cual se hallaba el *coeficiente de difusión*:

$$C = \exp(\frac{dif^2}{K}) \tag{6.25}$$

siendo K un número entero positivo.

Los resultados obtenidos no fueron los esperados, por lo tanto se pasó a calcular el *coeficiente de difusión* de otro modo.

Eliminación del coeficiente

En todos los casos expuestos se calcula mediante inferencia borrosa un *coeficiente de difusión*, que determina el flujo y por lo tanto cuanto se debe difundir una imagen. Otra de las opciones planteadas inicialmente, fue la de prescindir de este coeficiente, calculando directamente mediante inferencia borrosa los flujos totales, es decir, la salidas del sistema borroso en este caso eran: ϕ_n, ϕ_s, ϕ_e y ϕ_w.

Cálculo del grado de activación

Al igual que en los filtros IFCF desarrollados por Farbiz, Menhaj, Motamedi y Hagan en [Farbiz00], el grado de activación de cada regla se calculaba como:

$$\lambda_1 = min\{\mu_{A_1}(D_i) : D_{1i} \in soporte(A_1)\} \times \mu_{more}\left[\frac{\text{número de } D_{1i} \text{ / } D_{1i} \in \text{soporte(A}_1\text{)}}{\text{número total de } D_{1i}}\right]$$

Otra de las opciones planteadas fue modificar el cálculo del grado de actividad de cada regla, haciéndolas unas veces más restrictivas y otras menos. Se estudiaron las siguientes variaciones:

1. Sustitución de mínimos por productos: se consiguen reglas más restrictivas.

$$\lambda_1 = \prod_i \{\mu_{A_1}(D_i) : D_{1i} \in soporte(A_1)\} \times \mu_{more}\left[\frac{\text{número de } D_{1i} \text{ / } D_{1i} \in \text{soporte(A}_1\text{)}}{\text{número total de } D_{1i}}\right]$$

2. Sustitución de productos por mínimos: en este caso resultan reglas menos restrictivas.

$$\lambda_1 = min\{min\{\mu_{A_1}(D_i) : D_{1i} \in soporte(A_1)\}, \mu_{more}\left[\frac{\text{número de } D_{1i} \text{ / } D_{1i} \in \text{soporte(A}_1\text{)}}{\text{número total de } D_{1i}}\right]\}$$

Otros operadores

Al igual que en el caso del diseño de los filtros contenidos en el *Toolbox HPFborroso*, el número de píxeles sobre los que actúa el operador **more** es muy pequeño, por lo tanto recurrir a operadores del tipo “casi la mayoría” (**fairly more**) o “no poco”(**not few**) carece de sentido.

6.3.5. Posibles mejoras: la base de reglas borrosas

En los filtros implementados se ha visto que los resultados mejoran cuando se trabaja con reglas dobles. Sin embargo, como se muestra en la tabla 2, se están aplicando solo 8 reglas, cuando se podrían tener hasta sesenta y cuatro. Entonces, una de las posibles líneas de investigación, sería la de estudiar qué otras reglas se podrían imponer para la mejora de los filtros borrosos diseñados.

6.4. Toolbox Difusión: difusión borrosa anisótropa

6.4.1. Introducción

En la sección anterior se hacía uso de la lógica borrosa para la implementación de un filtro de difusión anisótropo. En esta sección también se va implementar un filtro de difusión anisótropo, siendo sus criterios de diseño:

- el filtro de Perona-Malik [Perona90].
- los filtros iterativos de lógica de control borroso, particularmente el filtro *SSFCF (Smoothing Fuzzy Control Filter)* [Farbiz00].
- el filtro paso alto HPFborroso implementado en el *Toolbox HPFborroso*.

Regularización espacial de el filtro de Perona-Malik

En el capítulo 4 se hacía una introducción al filtrado de difusión anisótropo y a su evolución a lo largo de los años. En él se explicaban ciertas características del filtro de Perona-Malik como eran sus propiedades y la definición del coeficiente de difusión. Regularización espaciales de este filtro tienen un sólido fundamento matemático. Como prototipo de filtro de difusión anisótropo, ahora nos centraremos en la regularización espacial del filtro de Perona-Malik realizada por Caté et al [Catte92].

El caso *m*-dimensional del filtro de Cate et al. presenta la siguiente estructura:

siendo $\Omega := (0,a_1)x...x(0,a_m)$ el dominio de nuestra imagen entonces el filtro calcula la imagen filtrada $u(x,y)$ de $f(x,y)$ como solución de la ecuación de difusión

$$\partial_t u = div\left(g(|\nabla u_\sigma|^2)|\nabla u|\right) \tag{6.26}$$

con condiciones iniciales, $u(x,y) = f(x)$, y condiciones de contorno $\partial_n = 0$ en $\partial\Omega$, donde *n* denota la normal al borde de la imagen.

- El tiempo, t, es un parámetro escalar: incrementos de t permiten simplificar la representación de la imagen.
- Para reducir la borrosidad de los bordes, la difusividad g es elegida como una función decreciente de el detector $|\nabla u_\sigma|$, donde ∇u_σ es el gradiente de una versión de u que se obtiene mediante la convolución de u con una Gaussiana de desviación estándar σ:

$$\nabla u_\sigma = \nabla(K_\sigma * u) \tag{6.27}$$

$$K_\sigma = \frac{1}{(2\pi\sigma^2)^{m/2}} \exp(-\frac{(|x|^2)}{(2\sigma^2)}) \tag{6.28}$$

$$g(s) = \begin{cases} 1 & s \leq 0 \\ 1 - \exp\left(\frac{-3{,}315}{(s/\lambda)^4}\right) & s > 0 \end{cases}$$

- λ juega el papel de parámetro de contraste. Estructuras con $|\nabla u_\sigma| > \lambda$ son consideradas como bordes, donde la difusividad es cercana a cero, mientras estructuras con $|\nabla u_\sigma| < \lambda$ son consideradas pertenecientes al interior de una región, donde la difusividad es próxima a uno.
- El parámetro σ, hace el filtro insensible a ruido a escalas mayores que su valor (siempre positivo).

6.4.2. Arquitectura del algoritmo

Objetivos

El objetivo de este proyecto es el diseño de filtros para el tratamiento de imágenes médicas, en concreto ecografías, es decir, imágenes afectadas por ruido *speckle*. Por lo tanto, para llevarlo a cabo se va a hacer una vez más uso de los filtros iterativos de lógica de control borroso *(IFCF)*. Se van a probar las siguientes opciones:

- Opción A

  Se centra en el suavizado de la imagen inicial.

  1. nldif_borroso

     La modificación que se introduce es sustituir la convolución de la imagen original con una gaussiana, por un filtrado borroso paso bajo con el filtro *SSFCF*.

- Opción B

  Las modificaciones ahora se introducen en el cálculo del gradiente, de modo que en en el cálculo de la difusividad se hace uso de la lógica borrosa. Es decir, en lugar de calcular el gradiente del modo convencional, se calcula la versión paso alto de la imagen a la entrada según el filtro HPFborroso (opción *Sobel2*).

  1. nldif_borroso4

     La imagen se suaviza mediante la convolución con una gaussiana y posteriormente se calcula la difusividad de la forma explicada.
  2. nldif_borroso5

     Como el filtro HPFborroso funciona bien ante el ruido *speckle*, se probará a calcular la difusividad sin suavizar previamente la imagen..

- Opción C

  1. nldif_borroso3

     Se corresponde con la composición de las opciones A y B, es decir, utilizará la lógica borrosa tanto para el suavizado de la imagen (SSFCF) como para el cálculo del gradiente (HPFborroso).

Otra de las limitaciones que presenta el filtro de Perona-Malik regularizado por Caté et al. sería el cálculo de los valores que deben tomar ciertos parámetros, como son λ, σ, t y el número de iteraciones. Por lo tanto otro de los objetivos es el diseño de un controlador borroso que los determine. En concreto, se diseñará un controlador que en función de la imagen a la entrada del filtro en cada iteración, asignará un valor a λ y determinará si es o no necesaria una nueva iteración. Esto se recoge en el filtro nldif_total.m.

Citar que el controlador borroso solo se añadirá a la versión original del filtro, ya que la composición de varios sistemas borrosos daría lugar a demasiadas no linealidades difíciles de controlar.

Implementación del controlador borroso

El primer paso en el diseño del controlador borroso es calcular el ruido de la imagen que se tiene como entrada en la iteración correspondiente. Esto se recoge en el archivo est_ruido1.m y sería:

1. A la imagen de entrada se le aplica una máscara $5x5$ que se desplaza por todos los píxeles de la imagen. A cada uno de estos le corresponde un valor Cs tal que

$$Cs = \frac{\sigma^2}{\text{med(I)}^2} \tag{6.29}$$

2. Una vez que se tiene la matriz de Cs se calcula Cu como

$$Cu^2 = \text{mediana}_i(Cs(i)) \tag{6.30}$$

Conocido Cu, éste se utilizará como entrada de nuestro sistema de inferencia borrosa para el cálculo de λ y el número de iteraciones.

- **Determinación de λ**

  El parámetro λ es el parámetro de contraste. Estructuras con $|\nabla u_\sigma| > \lambda$ son consideradas como bordes, por lo tanto en esa zona la difusividad debe ser cercana a cero y en las zonas con $|\nabla u_\sigma| < \lambda$ son consideradas pertenecientes al interior de una región, siendo la difusividad cercana a uno. Es decir, λ controla que intensidad

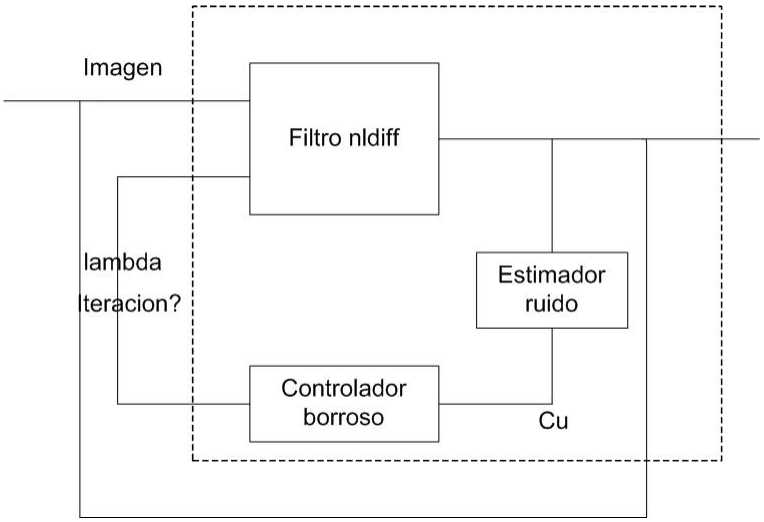


Figura 6.18: Controlador borroso.

de gradiente será difuminada y cual no. Por lo tanto, si la imagen está muy sucia el valor de λ ha de ser pequeño y al contrario, si la imagen está limpia, λ será alto. Esto significa q a medida que el proceso de difusión va llegando al final, λ debe ir aumentando, ya que al utilizar una imagen menos borrosa ésta es más sensible al ruido. Sin embargo, debe existir cierto compromiso en el valor que se le asigna a λ, pues si toma un valor demasiado alto se corre el riesgo de perder los bordes.

Por tanto, las variables lingüísticas que aparecen en este sistema son *Cu* como entrada (rango [0,1]) y λ como salida. Falta por determinar el rango de λ, y basándonos en resultados experimentales tomará [3.5, 8]. Entonces los conjuntos borrosos son los que se muestran a continuación.

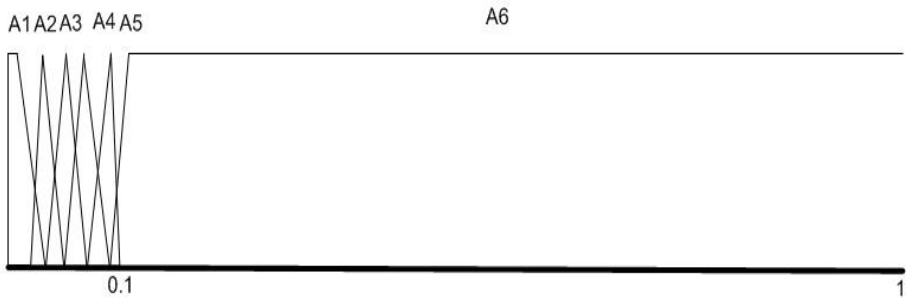


Figura 6.19: Conjuntos borrosos de la variable *Cu*.

Definidos los conjuntos borrosos que van a representar a las dos variables, el siguiente paso es determinar la base de reglas. Se hará uso de directivas del tipo:

SI el nivel de ruido que presenta la imagen es pequeño, ENTONCES el parámetro λ ha de ser alto.

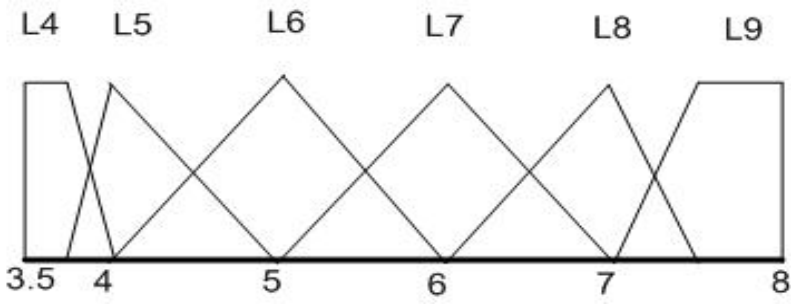


Figura 6.20: Conjuntos borrosos de la variable λ.

R_1: IF Cu are A_1 THEN λ is $L9$
R_2: IF Cu are A_2 THEN λ is $L8$
R_3: IF Cu are A_3 THEN λ is $L7$
R_4: IF Cu are A_4 THEN λ is $L6$
R_5: IF Cu are A_5 THEN λ is $L5$
R_6: IF Cu are A_6 THEN λ is $L4$

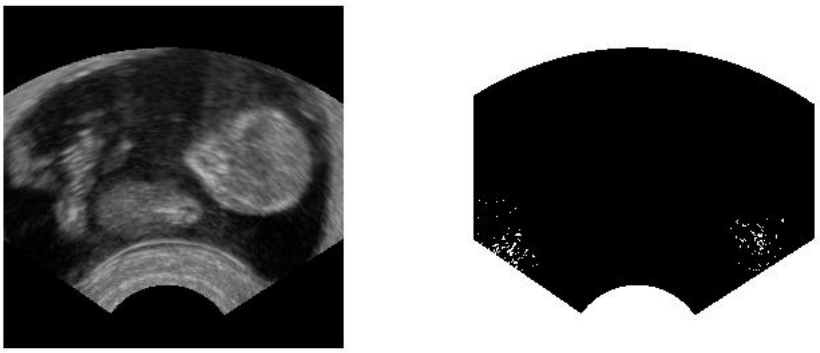
El proceso de desborrificación que permite obtener el valor real del parámetro λ varía un poco respecto al que adoptan los filtros implementados en los *toolboxes* explicados anteriormente. Sería:

$$\lambda = \frac{\sum_{i=1}^{8} C_i w_i \mu_{A_i}(Cu)}{\sum_{i=1}^{8} w_i \mu_{A_i}(Cu)} \tag{6.31}$$

- **Número de iteraciones**

  Estrictamente hablando en lugar de determinar el número de iteraciones, lo que se hace es observar el ruido que presenta la imagen que se obtiene a la salida de cada iteración, y decidir si es necesario o no continuar iterando.

  Cuando el controlador borroso asigna a λ el valor máximo significa que la imagen ya presenta poco ruido, y entonces se inicializa un contador que controla el número de veces que λ toma tal valor. Cuando el contador llega a *MAX*, se decide que el bucle debe detenerse. El valor *MAX* dependerá de cual sea el escalón de tiempo *t* utilizado, que a su vez depende del tipo de imagen. Así en el caso de imágenes sintéticas *t* puede ser relativamente grande, digamos mayor que uno, tomando en este caso *MAX* el valor catorce. Si se trabaja con ecografías reales, el tamaño de *t* ha de ser menor que uno, y entonces el valor que alcanzará el contador como máximo será ocho.



(a) (b)

Figura 6.21: Píxeles que intervienen en el cálculo del ruido que presenta la imagen

El controlador en ecografías

Anteriormente se ha explicado el diseño del controlador borroso, pero en el caso de trabajar con imágenes ecográficas reales es necesario introducir una pequeña modificación.

Observando la ecografía que se muestra en Fig. (6.21.a), se puede ver que una gran parte de la imagen no contiene información. Nos estamos refiriendo a la parte exterior de la imagen que es completamente negra. Esta zona dará lugar a valores de Cs que influyen en el valor que toma Cu. Por lo tanto será interesante que en el cálculo de Cu solo intervengan los Cs calculados a partir de los píxeles que pertenecen a la ecografía real. En Fig. (6.21.b) en negro se muestra la parte de la imagen que se ha de considerar para el cálculo de los Cu .

6.4.3. Otras opciones planteadas

A continuación se van a explicar qué otras alternativas se han estudiado para implementar estos filtros, pero las cuales al final no se han implementado debido a la falta de calidad en los resultados a los que llevan. Solamente se enumeran aquellas que se consideran de mayor interés.

Cálculo del ruido en una zona uniforme de la imagen

El primer estimador de ruido planteado calculaba el ruido de la imagen en una zona uniforme de la misma. Se elegía una región de la imagen de la que previamente se sabía que era una zona uniforme, y se calculaba:

$$R(n) = \frac{\sigma(n)^2}{\text{med}(I(n))^2} \tag{6.32}$$

donde *n* indica la iteración correspondiente.

1. Número de iteraciones:

   Para determinar cuando el número de iteraciones era suficiente, se calculaba el cociente $div = R(n)/R(n-1)$, y si este se mantenía próximo a uno durante un número considerable de iteraciones, se consideraba que realizar más iteraciones no supondría mejora alguna y de detenía el filtrado.

2. λ:

   Para el cálculo de λ se llevaba a cabo un proceso de inferencia borrosa como el que se explica en el apartado 6.4.2, siendo la variable lingüística de entrada *R*.

Estimador de ruido logarítmico.

Otra modo de estimar el ruido de la imagen fue el que se explica a continuación.

1. Se calcula el gradiente del logarimo de la imagen de entrada (f_x y f_y).
2. Se halla el módulo del gradiente, $M1$.
3. Se calcula la mediana de $M1$.
4. Conocida esta mediana, MED, se halla:

$$MAD = \sqrt{(fx - MED)^2 + (fy - MED)^2} \tag{6.33}$$

5. Ahora

$$Cu_2 = 1{,}4826 * \text{median}(MAD) \tag{6.34}$$

6. Finalmente se halla *Cu*, que será de nuevo la entrada del controlador borroso.

$$Cu = 0{,}5 * {Cu_2}^2 \tag{6.35}$$

En este caso conocido *Cu*, el parámetroλ y el número de iteraciones se calculaba como se hace en el filtro finalmente implementado en este *toolbox*.

6.4.4. Posibles mejoras

Dada la variedad de filtros que componen este *toolbox*, así como el gran número de funciones que han sido creadas como soporte a estos filtros, son muchas las posibilidades de mejora que presenta. Entre ellas podría ser interesante estudiar:

1. Implementación de un controlador para el parámetro σ.
2. Utilización de escalones de tiempo variable.
3. Implementación de controladores en todas las versiones de los filtros borrosos, es decir, *nldif_borroso, nldif_borroso3, nldif_borroso4 y nldif_borroso5*.

Capítulo 7

Experimentación y resultados

En el capítulo 6 se explicaban detalladamente los filtros diseñados para el tratamiento de ruido *speckle* implementados en los diferentes *toolboxes* que componen el proyecto. Como allí se señalaba, en este capítulo se va a mostrar los resultados obtenidos durante la experimentación de los mismos. La forma de evaluar dichos resultados es, en la mayoría de los casos, cualitativa. Es decir, se partirá de una imagen ruidosa, se realzará dicha imagen mediante un filtro y se verá en qué medida el preprocesado es adecuado. Evidentemente, para tener claro la validez o no de un filtro, se le comparará con el filtro clásico que se utilice con la misma finalidad, así por ejemplo, si se está tratando con un filtro que está diseñado para detectar bordes, parece claro que la comparación con un filtro de Prewitt o Sobel será sumamente útil. En algunos casos para evaluar en cierta medida la validez del filtro, se hará uso de índices de calidad como sería el *SSIM*.

Como el objetivo del proyecto es la eliminación de ruido *speckle*, las imágenes utilizadas para las pruebas serán fundamentalmente ecografías e imágenes sintéticas afectadas por este tipo de ruido. También se estudiará el comportamiento del filtro para imágenes con ruido *impulsivo* aunque los filtros no hayan sido diseñados para ello. Evidentemente, como la variación de los parámetros que intervienen en cada uno de los filtros puede ser muy amplia, sólo se mostrarán los resultados que han parecido ser los más interesantes.

Una ventaja importante de la implementación de los filtros mediante la API de *matlab*, utilizando archivos mex, es que el tiempo de computación necesario para el filtrado de las imágenes es muy pequeño. Por eso, en el resto del capítulo no se realizarán comparativas temporales.

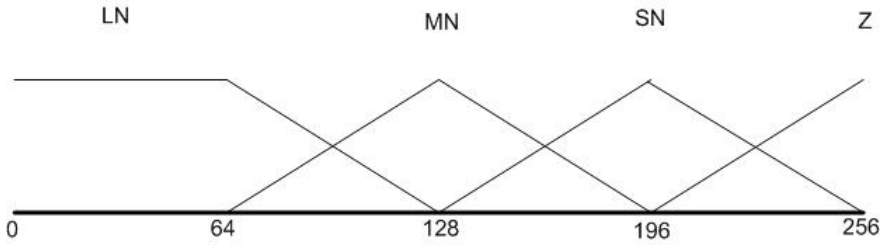
7.1. Toolbox HPFborroso

Los filtros implementados en este *toolbox* tienen como objetivo la detección de bordes. Estos filtros son versiones borrosas de los operadores de Prewitt y Sobel, por lo tanto todos los resultados se compararán con los obtenidos por medio de sus versiones clásicas. Veremos tanto ecografías como imágenes sintéticas para cada uno de los filtros.

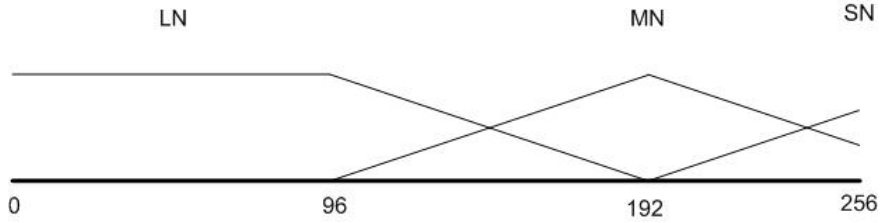
7.1.1. Ecografías: imágenes reales

En esta sección se mostrarán los experimentaciones realizadas con imágenes ecográficas reales. Se han utilizado dos volúmenes de datos de partida. El primero de ellos es el volumen de un feto y el segundo el de un riñón. Para poder trabajar sobre los datos con los filtros 2D realizados, se han pasado cada uno de estos volúmenes a secciones aplicando posteriormente dichos filtros sobre sobre cada una de ellas por separado. Una vez que se tienen todas las secciones procesadas, se han renderizado mediante trazado de rayos usando la herramienta de visualización VTK (*Visualization ToolKit*) [Schroeder00].

También hay que señalar que los conjuntos borrosos utilizados en cada una de las imágenes son diferentes. En el caso del feto se han definidos cuatro conjuntos borrosos, mientras que para el caso del riñón se tienen tres conjuntos (Fig. 7.1).



(a)



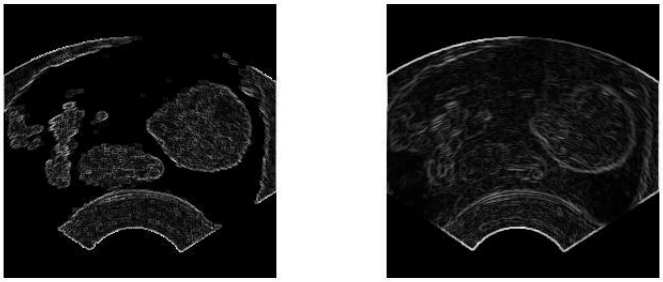
(b)

Figura 7.1: Conjuntos borrosos definidos para las ecografías de: (a) feto; (b) riñón.

Primero se mostrarán los resultados que se obtienen al procesar una sección 2D y posteriormente se verá que es lo que sucede sobre el volumen total.

Tanto en el caso del feto como del riñón, se consiguen mejores resultados cuando se detectan los bordes con los operadores borrosos que con las versiones clásicas. En un

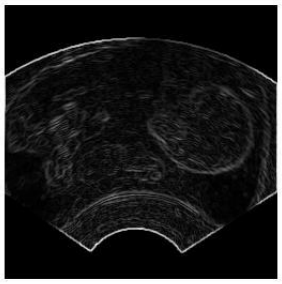
Figura 7.2: Sección 2D del volumen de un feto



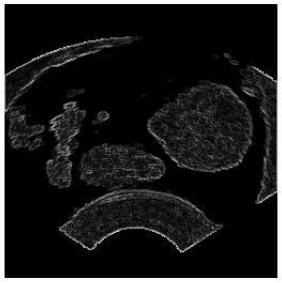
(a) (b)

Figura 7.3: Procesado de sección 2D: (a) Versión borrosa del operador de Prewitt; (b) Prewitt en su versión clásica.

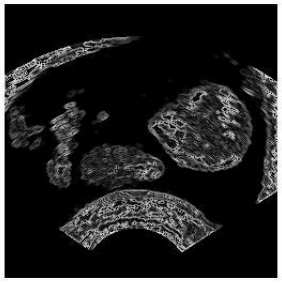
principio no se observa gran diferencia entre utilizar Prewitt borroso o Sobel borroso, sin embargo si las hay respecto el operador borroso Sobel2. Así, mientras que en los primeros los contornos quedan muy bien delimitados, con el segundo aparte de delimitar los contornos se consiguen marcar también aquellas transiciones más acentuadas que pertenecen al interior de los volúmenes.



(a)



(b)



(c)

Figura 7.4: Procesado de sección 2D: (a) Versión clásica del operador de Sobel; (b) Filtro borroso Sobel; (c) Filtro borroso Sobel2.

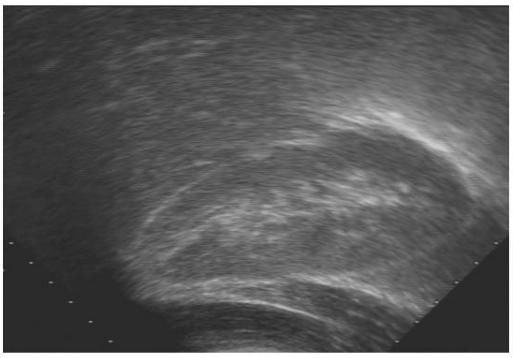
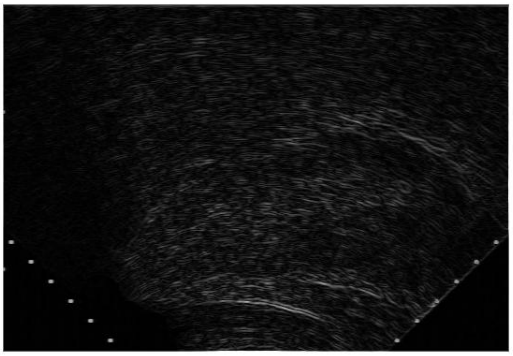
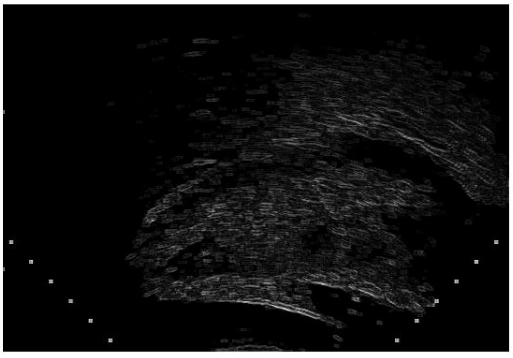


Figura 7.5: (a) Sección 2D del volumen de un riñón.

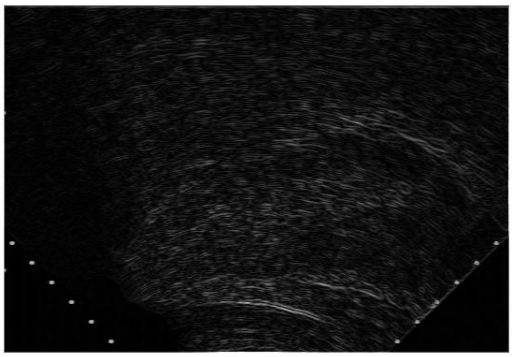


(a)

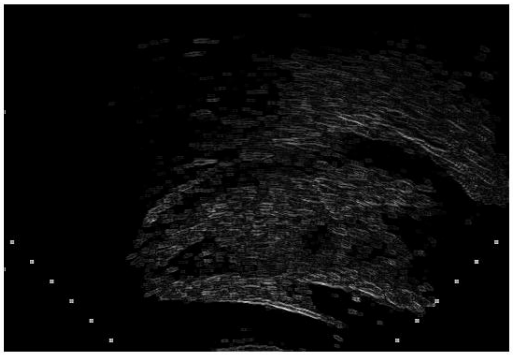


(b)

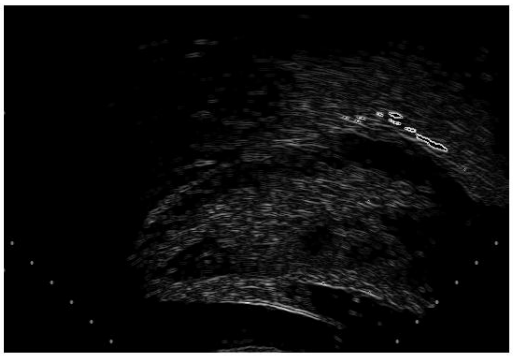
Figura 7.6: Procesado de sección 2D: (a) Versión clásica del operador de Prewitt; (b) Prewitt en su versión borrosa.



(a)



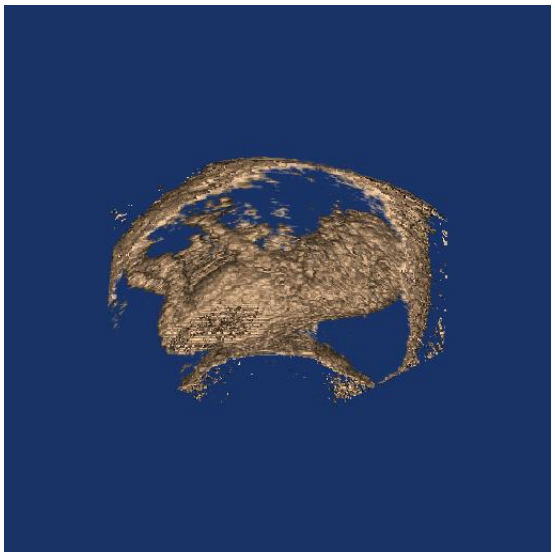
(b)



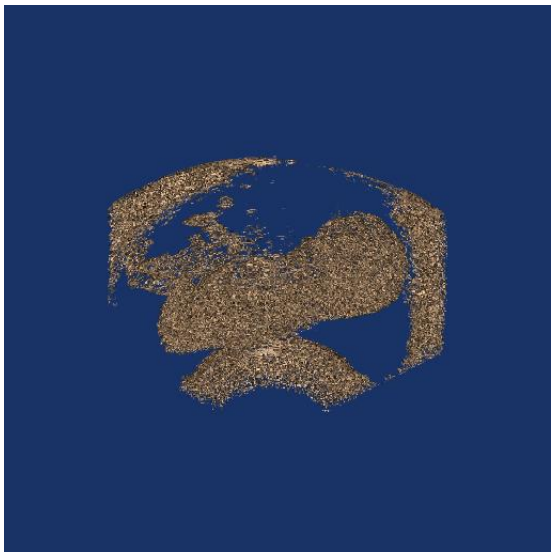
(c)

Figura 7.7: Procesado de sección 2D: (a) Versión clásica del operador de Sobel; (b) Filtro borroso Sobel; (c) Filtro borroso Sobel2.

A continuación se muestra cual es el resultado de aplicar el operador borroso Sobel2 al volumen del feto. Solo se renderizará el volumen para este operador ya que es con el que se consiguen mejores resultados para cada sección 2D. Respecto al volumen sobre el que se trabaja, es necesario aclarar que solo se utiliza aquel dado por las secciones comprendidas entre la 115 y 195, ya que el resto pertenecen a las paredes del útero y los resultados que se obtendrían para la detección de bordes en tal volumen no serían indicativos.



(a)



(b)

Figura 7.8: (a) Volumen original; (b) Volumen procesado con el operador borroso Sobel2.

7.1.2. Imágenes sintéticas

En la sección anterior se ha visto cuales son los resultados de aplicar el detector de bordes creado sobre ecografías, y se ha observado que mejora considerablemente a los operadores clásicos. Ahora se va a estudiar cual es su comportamiento cuando opera sobre cualquier imagen, tanto si se trata de una imagen limpia como si está afectada por ruido *speckle* o *impulsivo*.

Imágenes limpias

Para mostrar el funcionamiento de los filtros cuando éstos se enfrentan a imágenes sintéticas, se utilizarán las imágenes que tienen un tamaño de 256 x 256 *píxeles* y 256 niveles de gris. Se muestran en la Fig. 7.9. El número de conjuntos borrosos definidos son dos en el caso del cameraman, y tres para la casa.

En base a los resultados obtenidos, se puede decir que los nuevos operadores borrosos marcan mayor número de bordes (detalle muy fino), pero la calidad de estos bordes no es muy buena.

Imágenes afectadas por ruido speckle

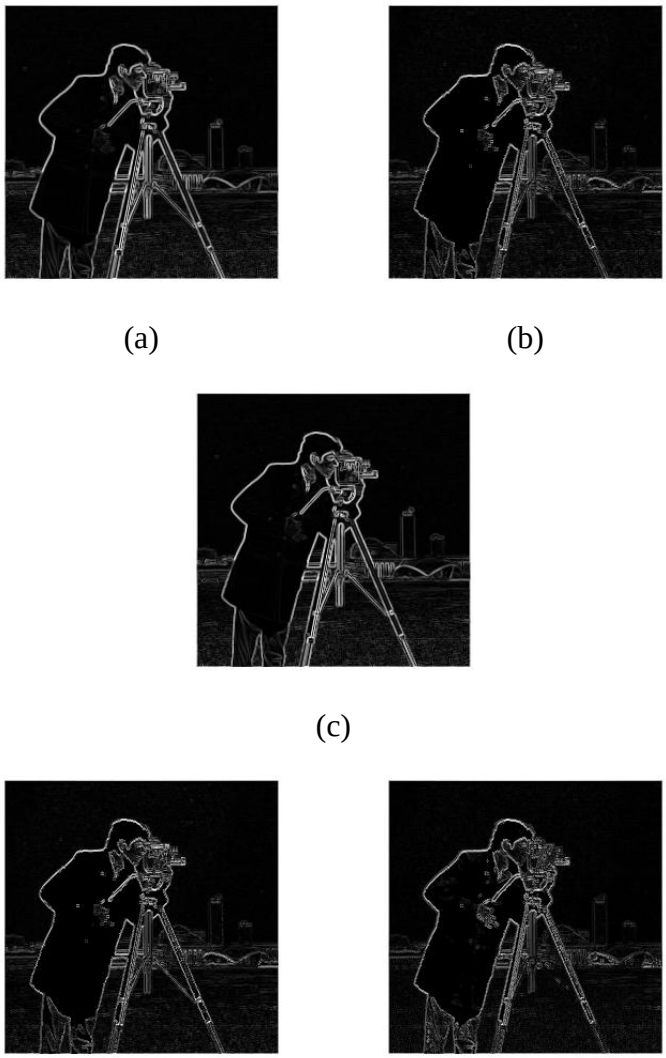
A las imágenes seleccionadas se les añade ruido *speckle*. Se experimentará que sucede cuando se tiene poco (media 0, varianza 0.01) y mucho ruido (media 0 y varianza 0.04) (Figs. 7.12, 7.13).

Imágenes afectadas por ruido gaussiano

Aunque el objetivo de este proyecto no es el tratamiento del ruido *gaussino*, se va a experimentar con imágenes afectadas por este tipo de ruido para analizar las propiedades de los nuevos filtros. Se verá que sucede para ruido *gaussiano* de media 0 y varianza 0.01 y con ruido gaussiano de media 0 y varianza 0.04 (Fig. 7.18, 7.19).

(a) (b)

Figura 7.9: Imágenes ausentes de ruido: (a) Cameraman; (b) Casa.

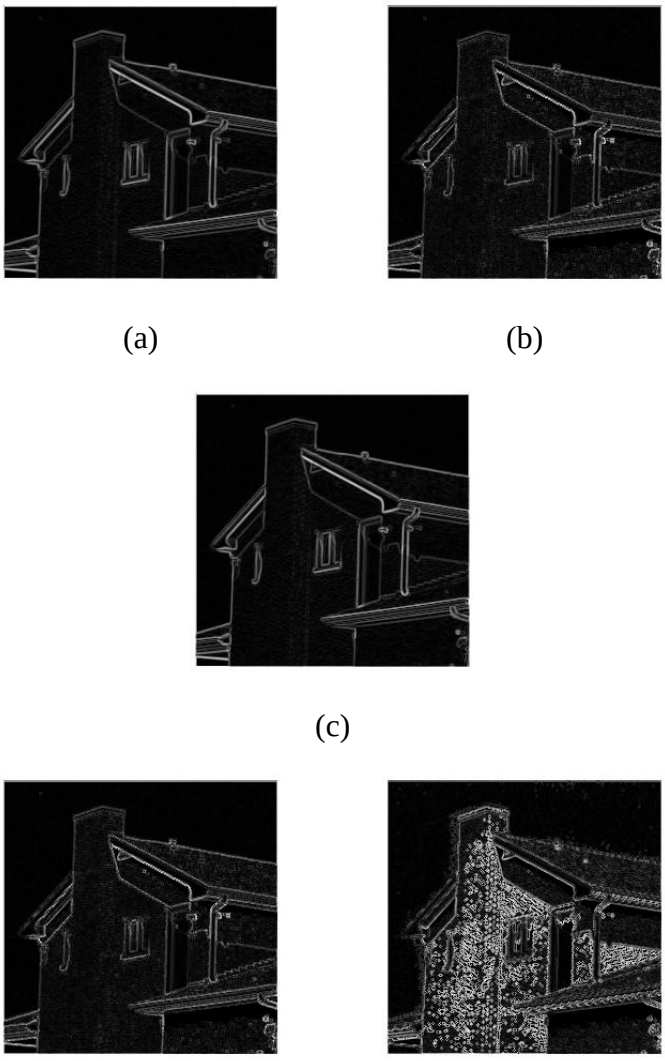


(a) (b)

(c)

(d) (e)

Figura 7.10: Detectores de bordes: (a) Prewitt clásico; (b) Prewitt borroso; (c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.

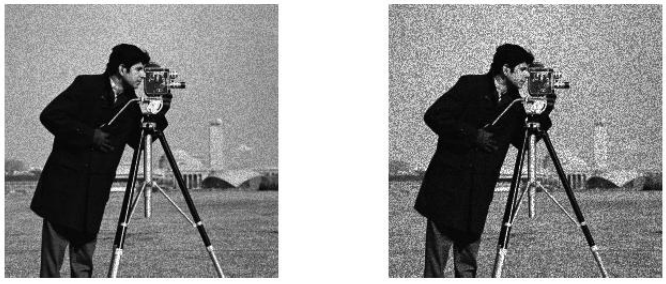


(a) (b)

(c)

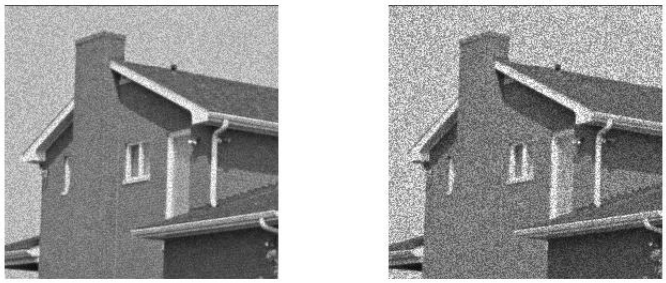
(d) (e)

Figura 7.11: Detectores de bordes: (a) Prewitt clásico; (b) Prewitt borroso; (c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.



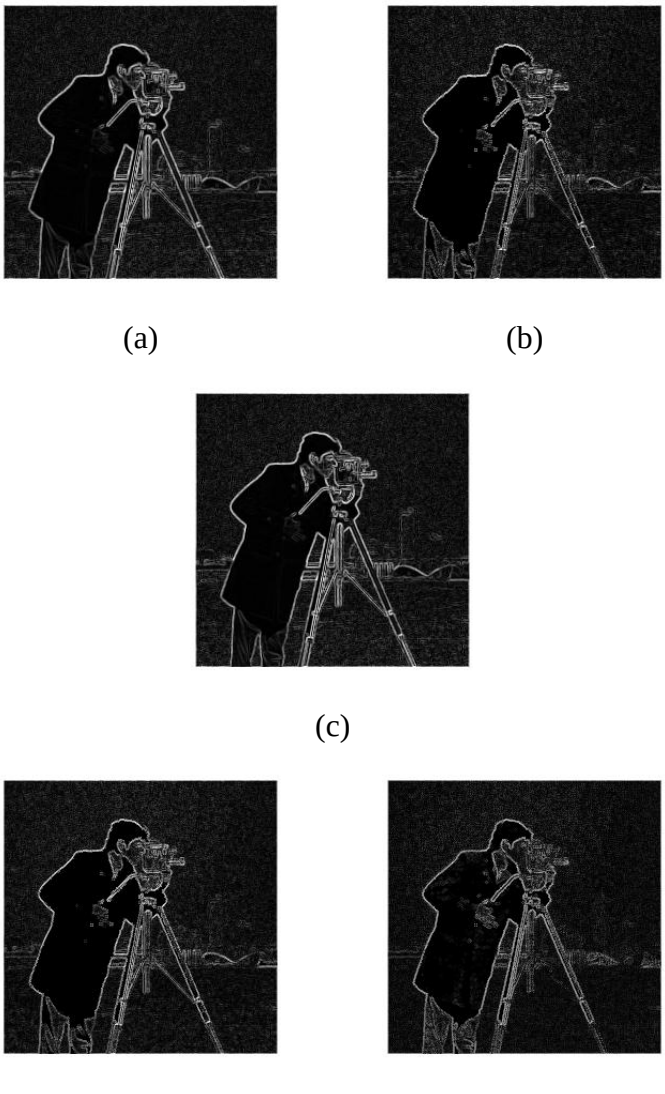
(a) (b)

Figura 7.12: Ruido *speckle*: (a) m=0 var=0.01; (b) m=0 var=0.04.



(a) (b)

Figura 7.13: Ruido *speckle*: (a) m=0 var=0.01; (b) m=0 var=0.04.

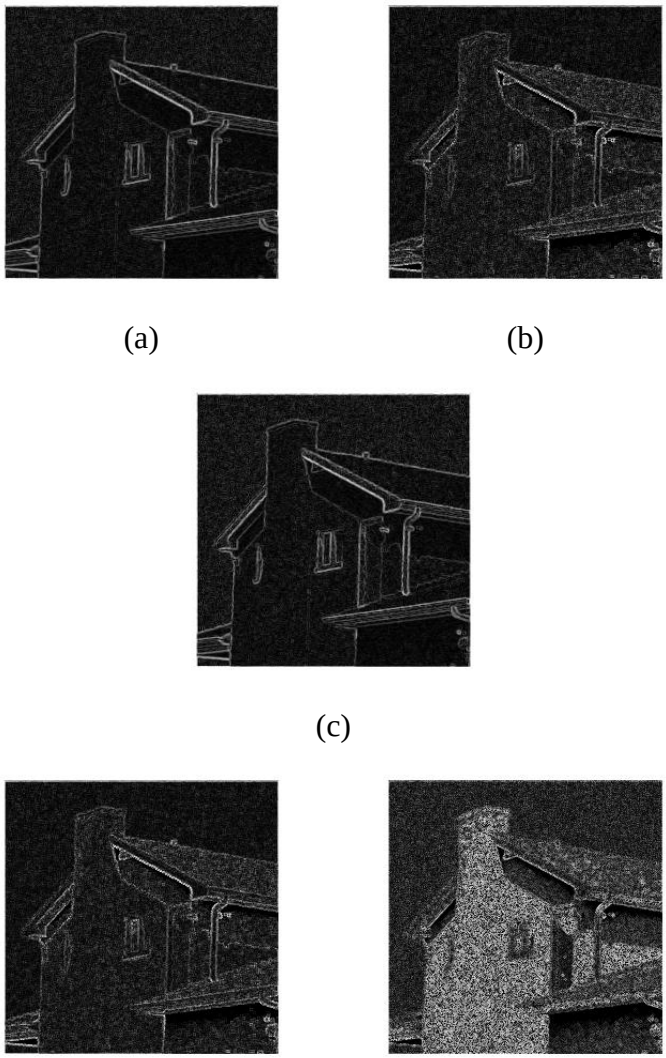


(a) (b)

(c)

(d) (e)

Figura 7.14: Ruido *speckle* (m=0, var 0.01): (a) Prewitt clásico; (b) Prewitt borroso; (c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.

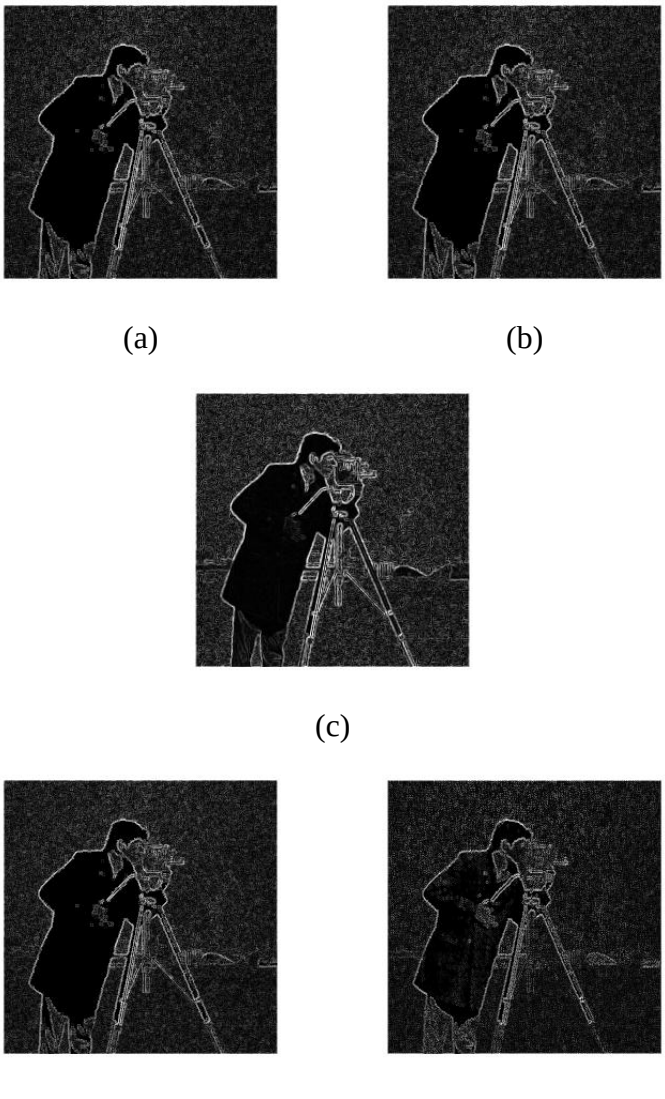


(a) (b)

(c)

(d) (e)

Figura 7.15: Ruido *speckle* (m=0, var 0.01): (a) Prewitt clásico; (b) Prewitt borroso; (c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.

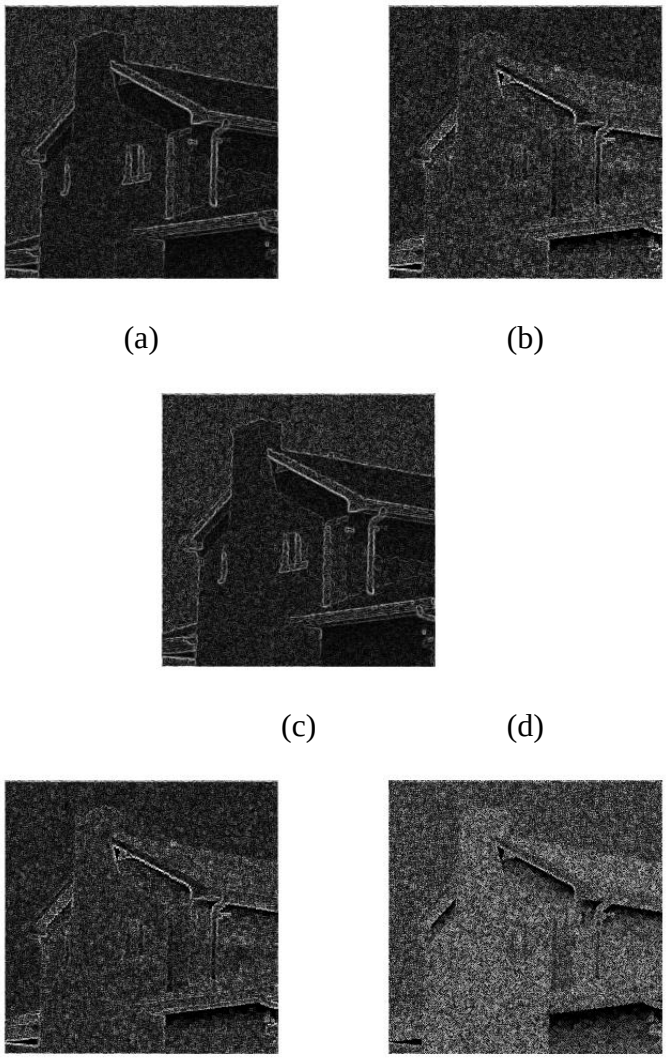


(a) (b)

(c)

(d) (e)

Figura 7.16: Ruido *speckle* (m=0, var 0.04): (a) Prewitt clásico; (b) Prewitt borroso; (c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.



(a) (b)

(c) (d)

(e) (f)

Figura 7.17: Ruido *speckle* (m=0, var 0.04): (a) Prewitt clásico; (b) Prewitt borroso; (c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.

(a)

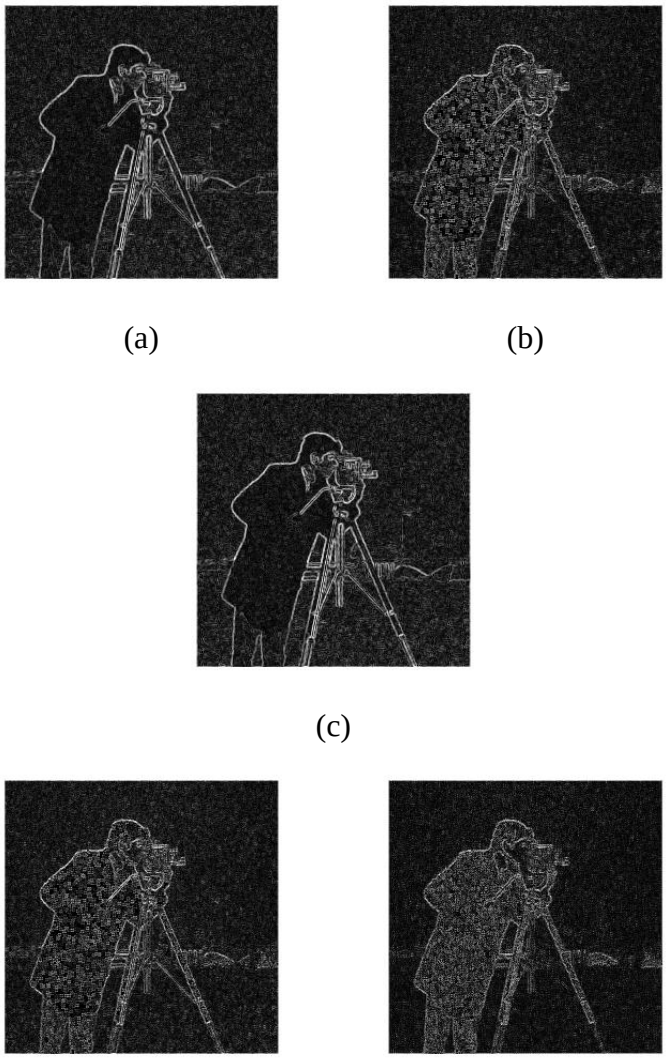
(b)

Figura 7.18: Ruido *gaussiano*: (a) m=0 var=0.01; (b) m=0 var=0.04.

(a)

(b)

Figura 7.19: Ruido *gaussiano*: (a) m=0 var=0.01; (b) m=0 var=0.04.

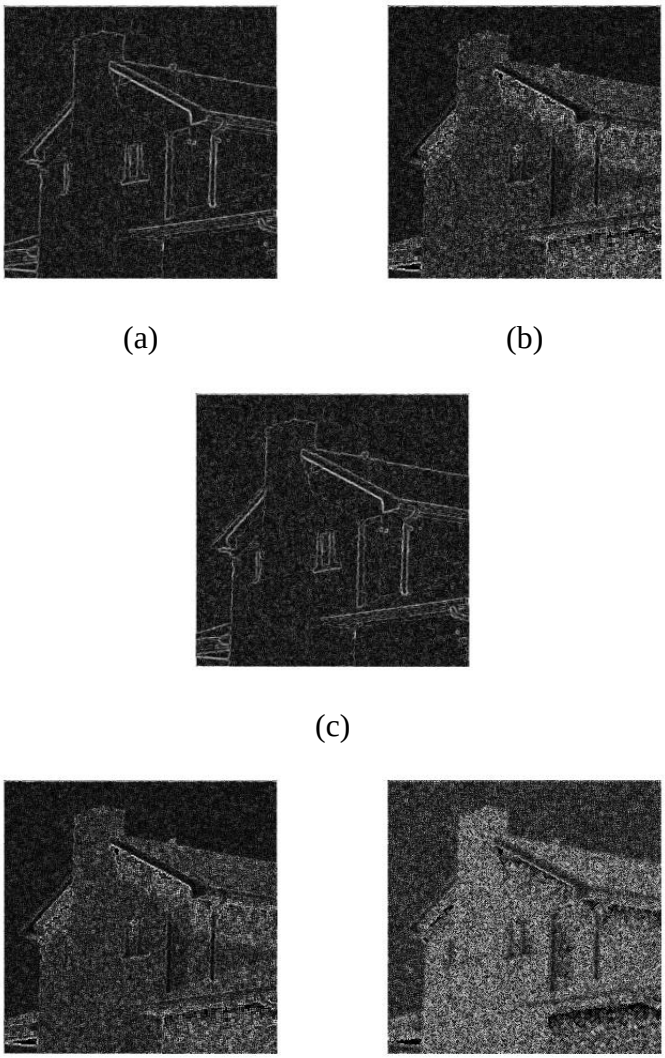


(a) (b)

(c)

(d) (e)

Figura 7.20: Ruido *gaussiano* (m=0, var 0.01): (a) Prewitt clásico; (b) Prewitt borroso; (c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.

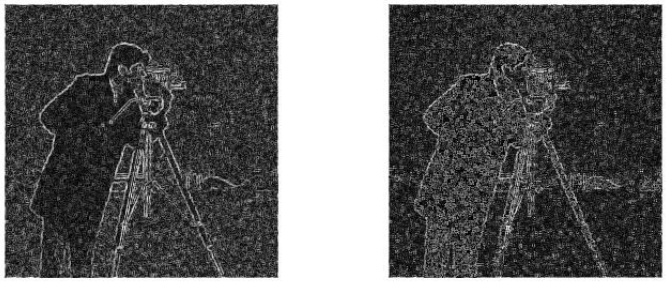


(a) (b)

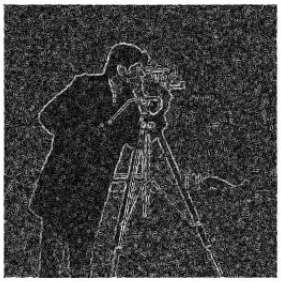
(c)

(d) (e)

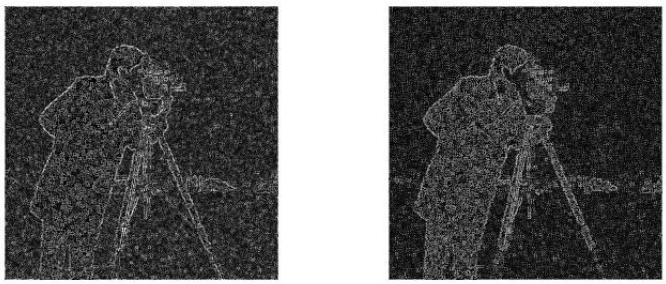
Figura 7.21: Ruido *gaussiano* (m=0, var 0.01): (a) Prewitt clásico; (b) Prewitt borroso;(c)Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.



(a) (b)

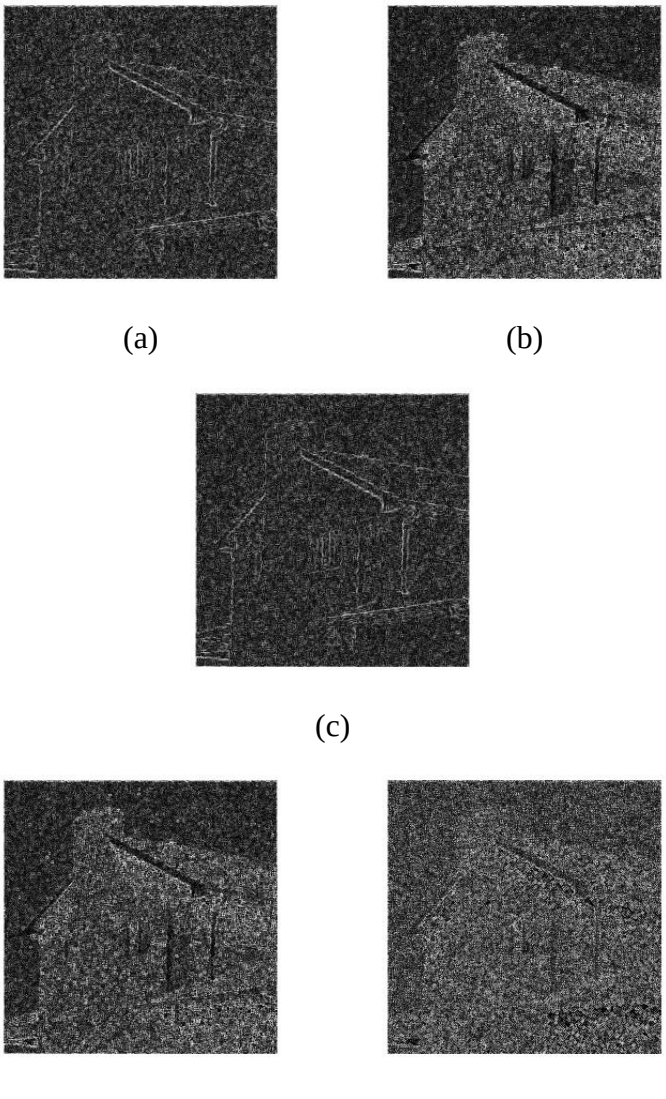


(c)



(d) (e)

Figura 7.22: Ruido *gaussiano* (m=0, var 0.04): (a) Prewitt clásico; (b) Prewitt borroso;(c) Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.



(a) (b)

(c)

(d) (e)

Figura 7.23: Ruido *gaussiano* (m=0, var 0.04): (a) Prewitt clásico; (b) Prewitt borroso; (c)Sobel clásico; (d) Sobel borroso; (e) Sobel2 borroso.

7.2. Toolbox Anisótropo

Después de haber explicado detalladamente en el capítulo anterior los filtros implementados en este *toolbox*, se van a mostrar los resultados obtenidos por medio de la ejecución de los mismos. La forma de evaluar dichos resultados es fundamentalmente cualitativa. Es decir, se partirá de una imagen ruidosa, se realzará dicha imagen mediante un filtro y se verá en qué medida el resultado obtenido es adecuado. En el caso del análisis de imágenes sintéticas, se podrá hacer una evaluación en cierta manera cuantitativa según el índice de calidad *SSIM*. Evidentemente, los resultados obtenidos deben ser comparados con los que se obtienen al aplicar algún filtro clásico, como por ejemplo el filtro de mediana.

7.2.1. Imágenes sintéticas

Para mostrar el comportamiento de los filtros de difusión anisótropa diseñados cuando se enfrentan a imágenes afectadas por ruido *speckle* y ruido *gaussiano* se ha partido de las imágenes sintéticas mostrada en la Fig. (7.24). Estas imágenes tiene un tamaño de 256×256 píxeles y 256 niveles de gris.

Imágenes afectadas por ruido speckle

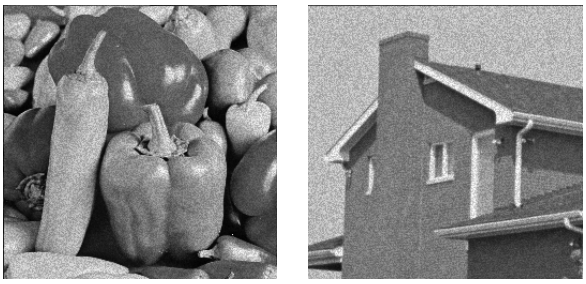
A las imágenes originales se les va a añadir ruido *speckle* según la siguiente expresión:

$$I_s = I_0 + I_0 \Delta r(\sigma_s^2) \tag{7.1}$$

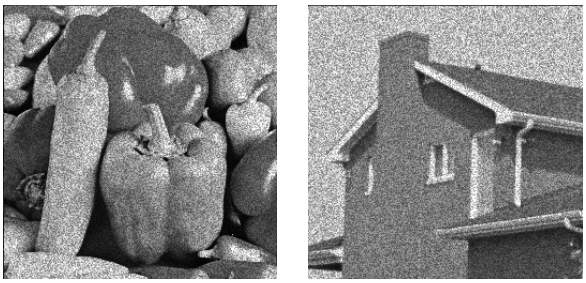
Se va a tomar una varianza $\sigma_s^2 = 0{,}01$ y $\sigma_s^2 = 0{,}04$ para analizar el comportamiento de los filtros en diferentes ambientes.

De esta manera se obtienen la imágenes ruidosas que se muestran en la Fig. (7.25).

Figura 7.24: imágenes originales.



(a) (b)

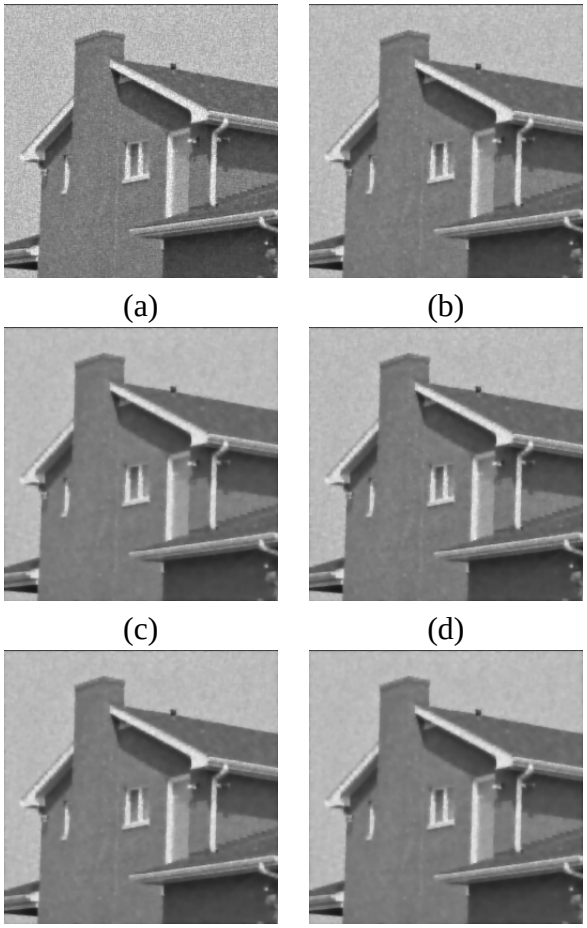


(c) (d)

Figura 7.25: Imágenes ruidosas utilizadas en las pruebas: (a) var = 0.01; (b) var = 0.01; (c) var = 0.04; (d) var = 0.04

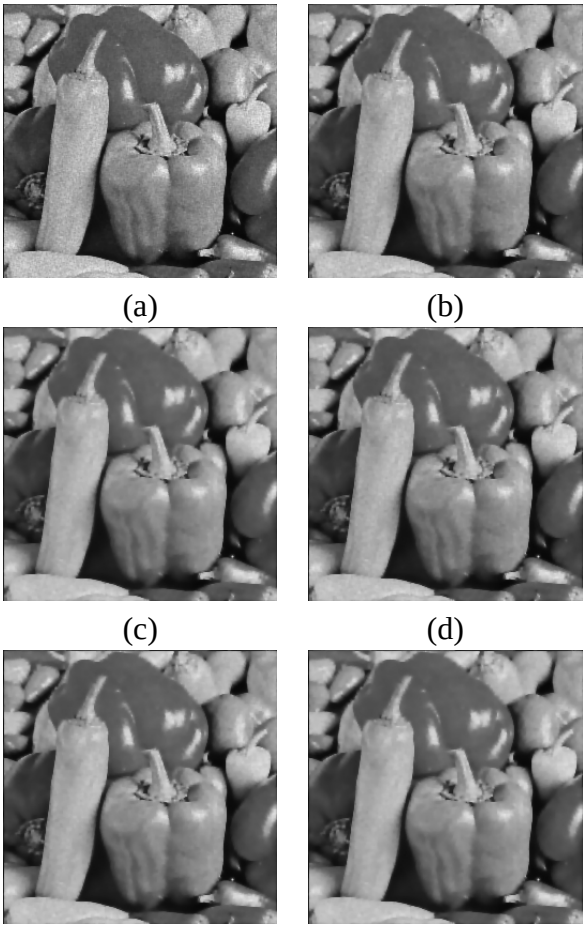
Si estas imágenes se someten a un filtrado de difusión anisótropo borroso se obtienen los resultados que se muestran a continuación:

- anisotropo_log



(a) (b) (c) (d) (e) (f)

Figura 7.26: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

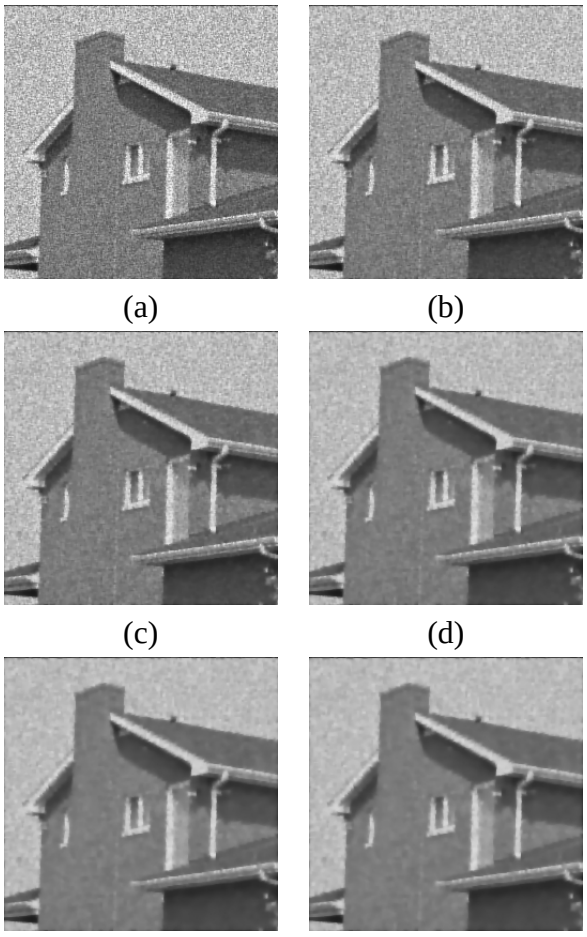


(a) (b)

(c) (d)

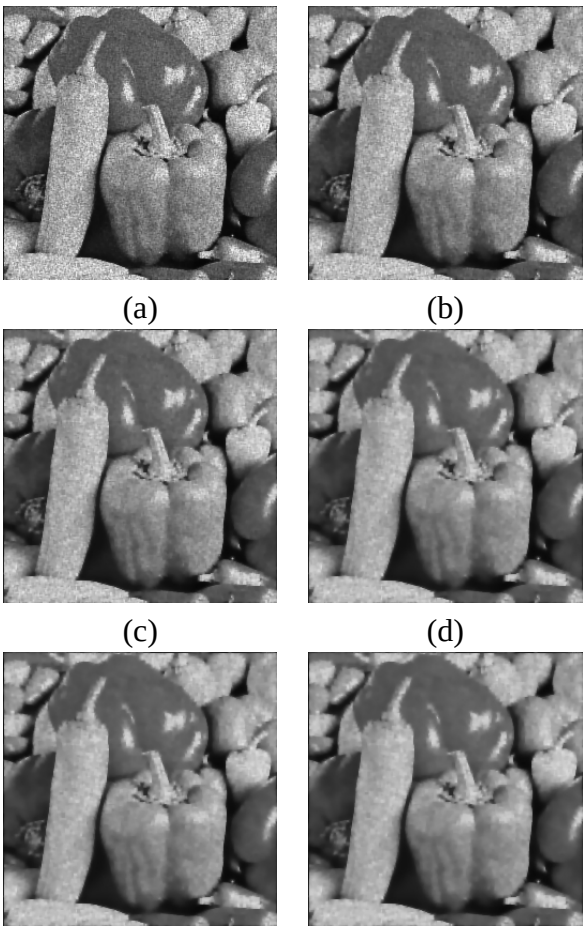
(e) (f)

Figura 7.27: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b) (c) (d) (e) (f)

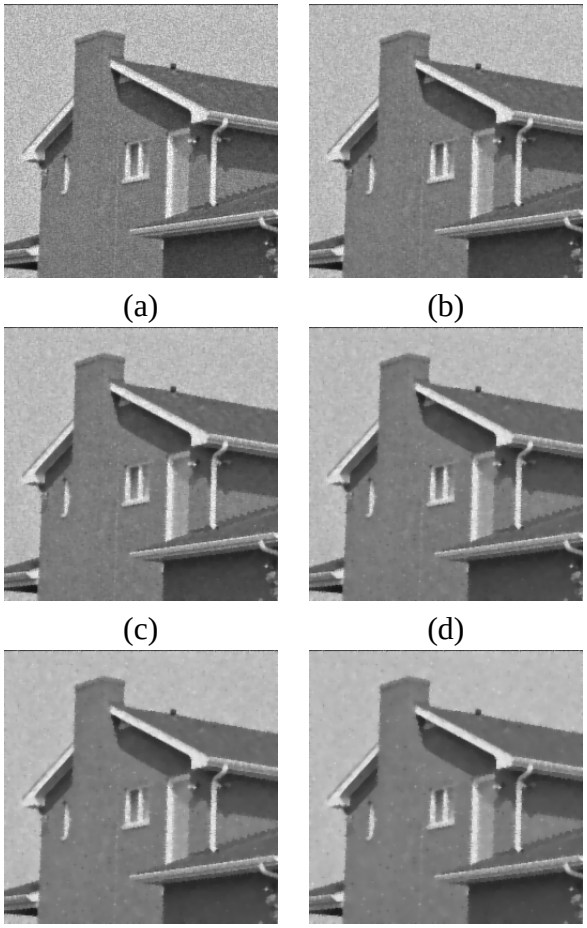
Figura 7.28: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b) (c) (d) (e) (f)

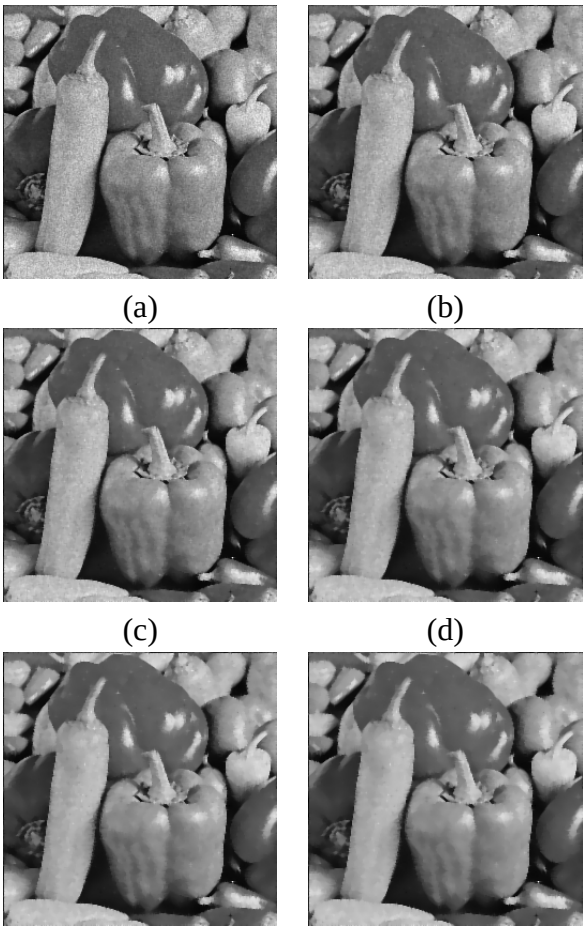
Figura 7.29: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

- anisotropo_doble



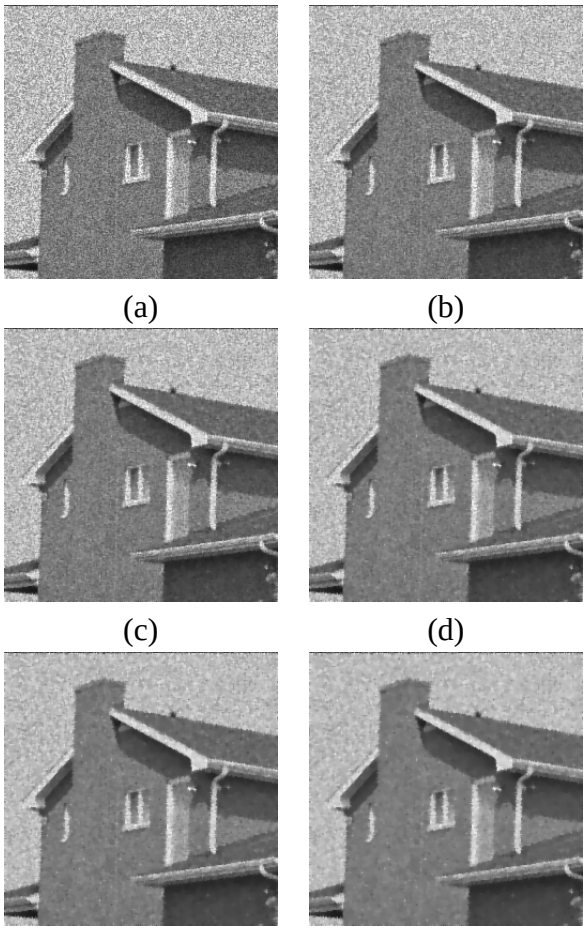
(a) (b) (c) (d) (e) (f)

Figura 7.30: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b) (c) (d) (e) (f)

Figura 7.31: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

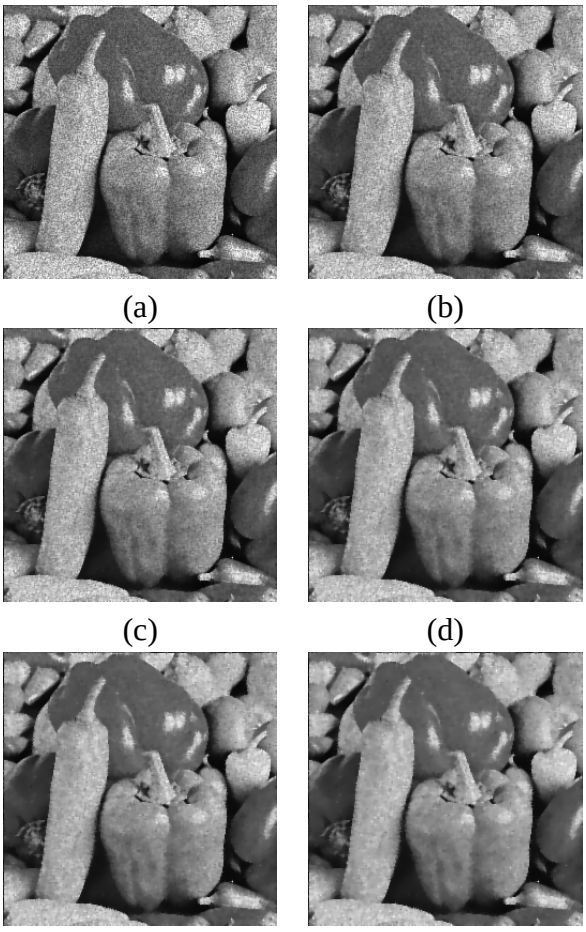


(a) (b)

(c) (d)

(e) (f)

Figura 7.32: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b)

(c) (d)

(e) (f)

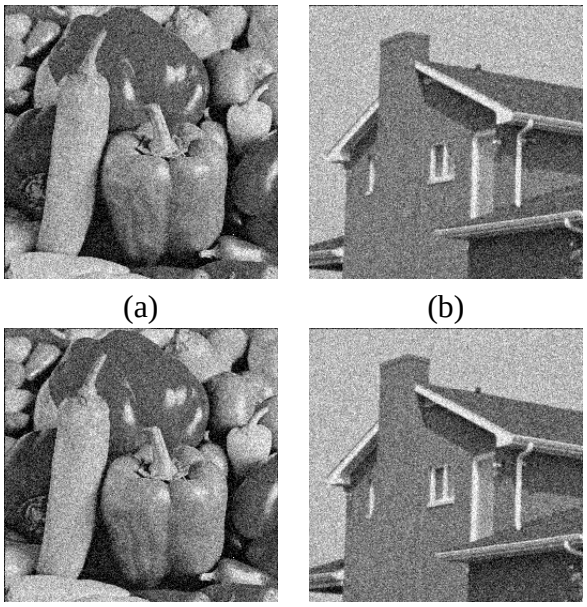
Figura 7.33: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

Imágenes afectadas por ruido gaussiano

Se analizarán también los resultados de añadir a las imágenes ruido *gaussiano*:

$$I_s = I_0 + \eta(\sigma_g^2) \tag{7.2}$$

Se va a tomará una varianza $\sigma_g^2 = 0{,}01$ y $\sigma_g^2 = 0{,}04$ para analizar el comportamiento de los filtros en diferentes situaciones. La media será nula en todos los casos.



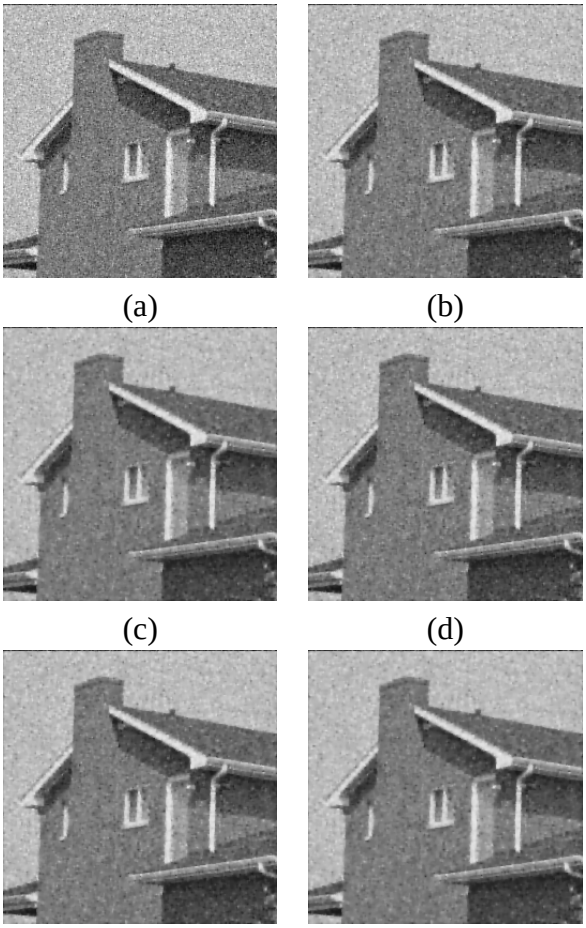
(a) (b)

(c) (d)

Figura 7.34: Imágenes ruidosas utilizadas en las pruebas: (a) var = 0.01; (b) var =0.01; (c) var =0.04; (d) var = 0.04.

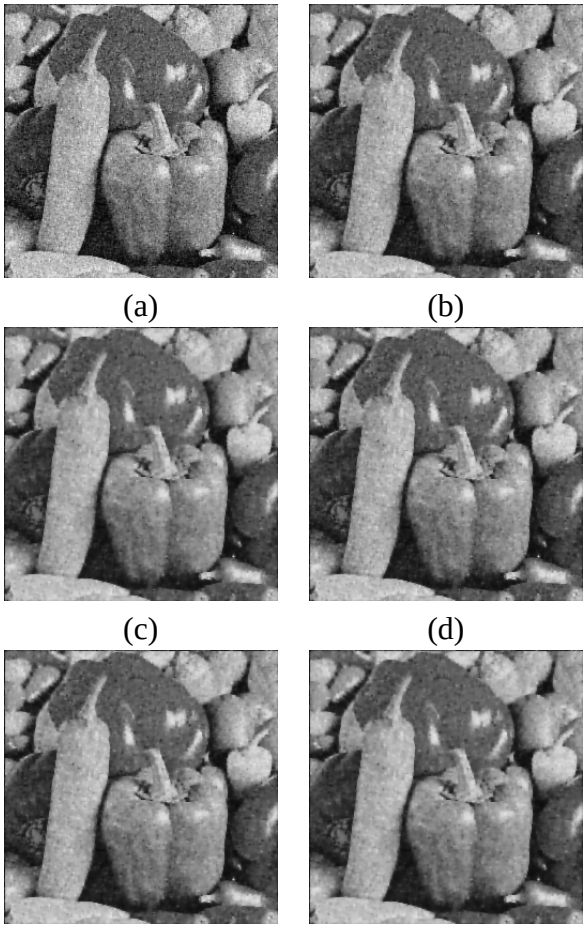
Si estas imágenes se someten a un filtrado de difusión anisótropo borroso se obtienen los resultados que se muestran a continuación:

- anisotropo_log



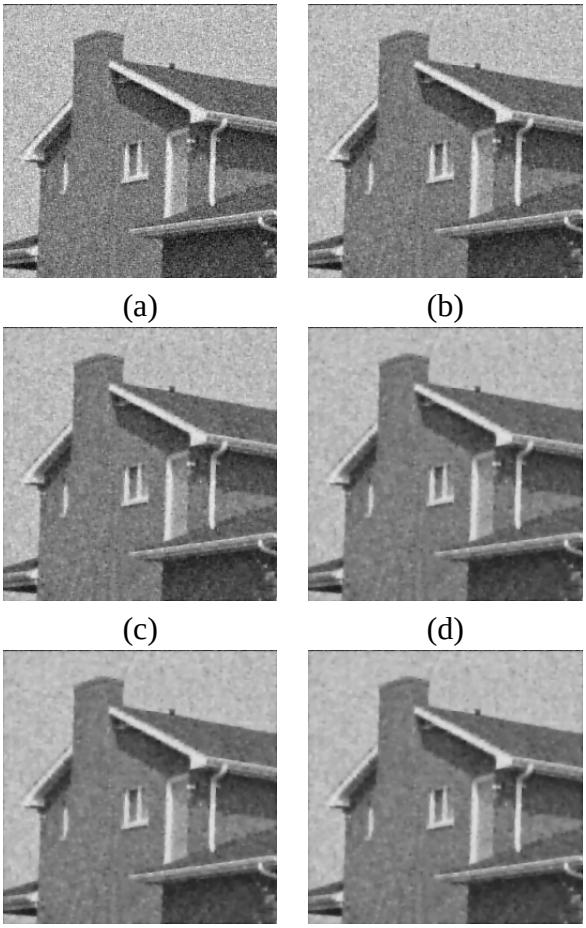
(a) (b) (c) (d) (e) (f)

Figura 7.35: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b) (c) (d) (e) (f)

Figura 7.36: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

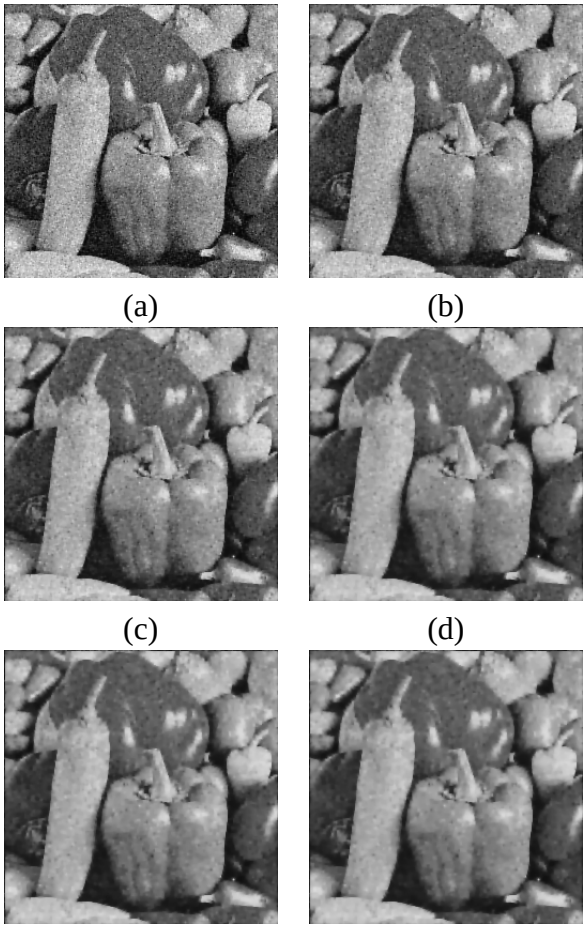


(a) (b)

(c) (d)

(e) (f)

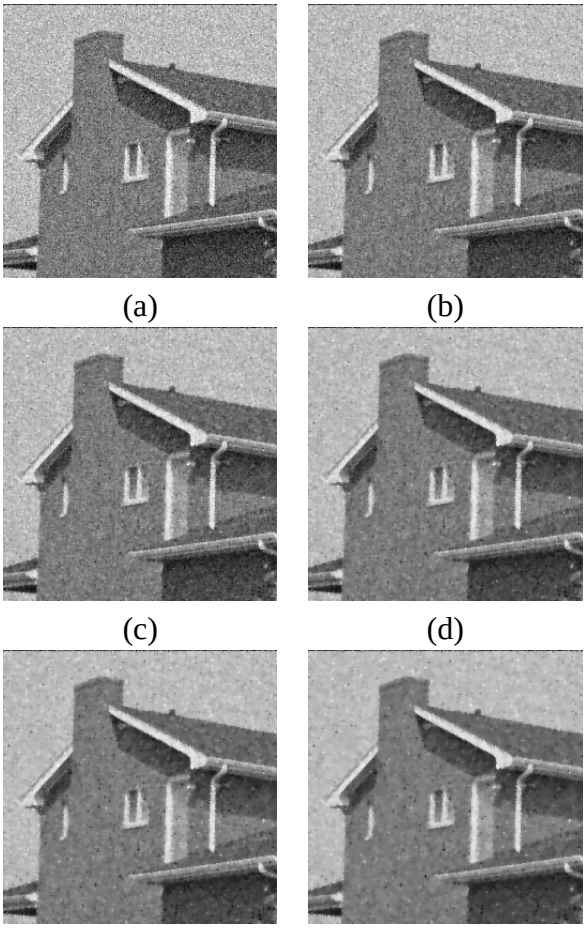
Figura 7.37: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b) (c) (d) (e) (f)

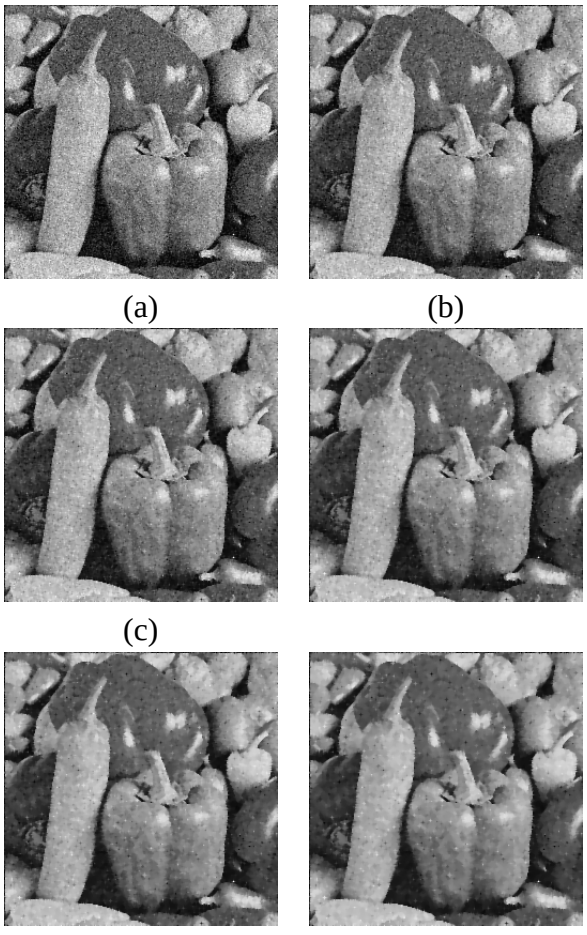
Figura 7.38: Imágenes procesadas con el filtro anisotropo_log (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

- anisotropo_doble



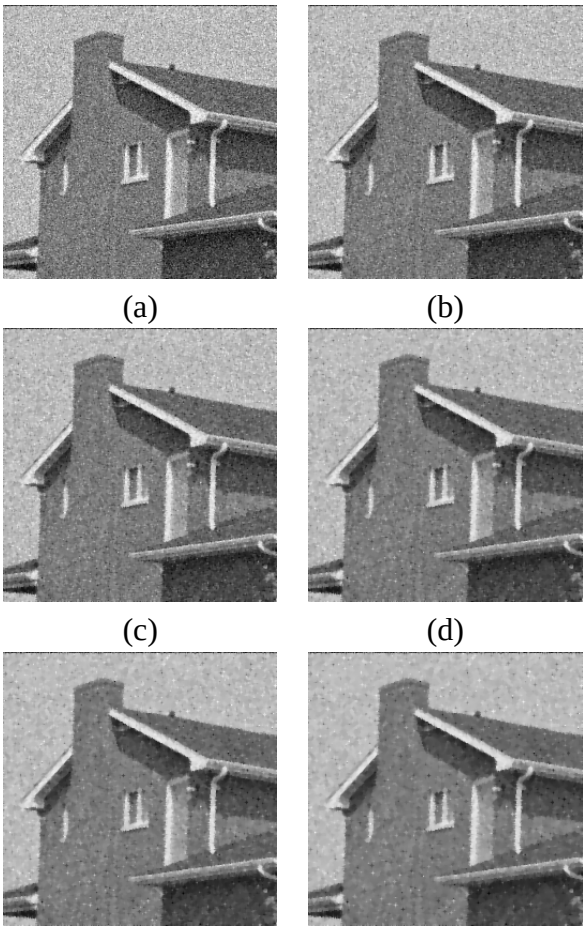
(a) (b) (c) (d) (e) (f)

Figura 7.39: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b) (c) (d) (e)

Figura 7.40: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.01): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones



(a) (b)

(c) (d)

(e) (f)

Figura 7.41: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

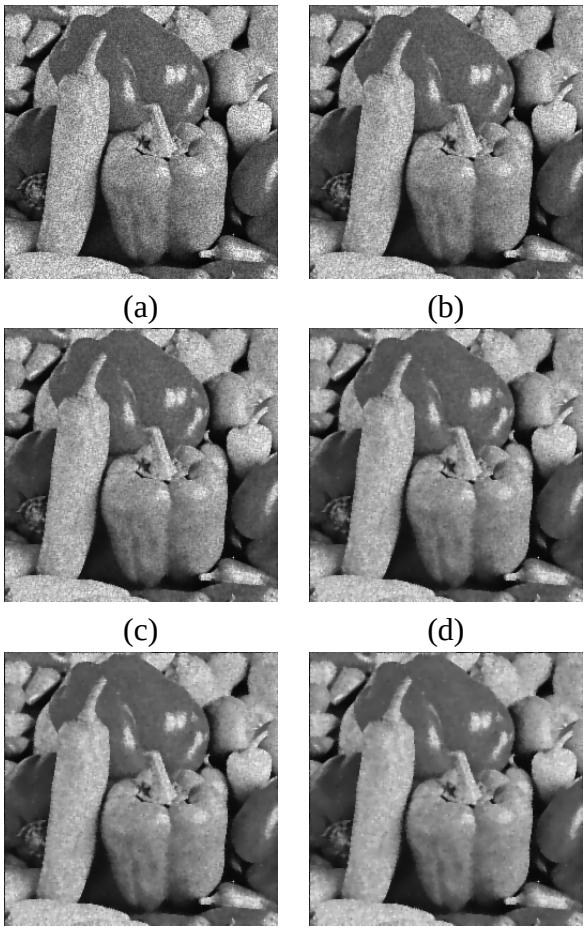
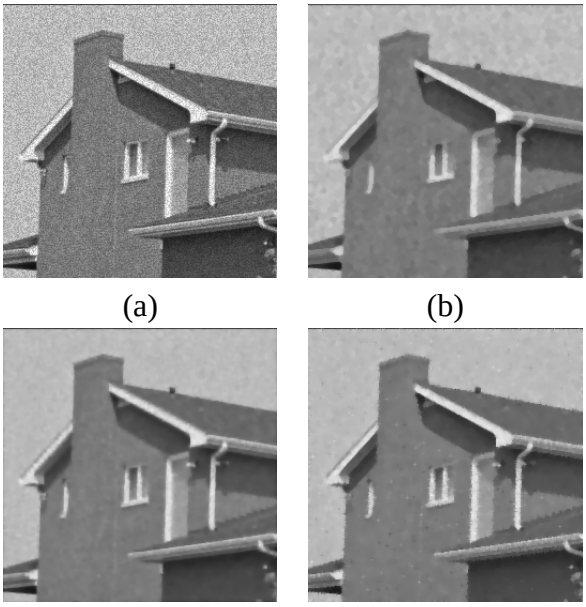


Figura 7.42: Imágenes procesadas con el filtro anisotropo_doble (m = 0, var = 0.04): (a) 10 iteraciones; (b) 20 iteraciones; (c) 30 iteraciones; (d) 40 iteraciones; (e) 50 iteraciones; (f) 60 iteraciones

Análisis de los resultados

Cualitativamente, en base a los imágenes obtenidas como resultados, se puede decir que los filtros diseñados presentan un comportamiento bastante bueno ante el ruido *speckle*. Sin embargo, cuando la imagen está afectada por ruido *gaussiano*, los resultados obtenidos ya no son tan buenos, pues al intentar limpiar este tipo de ruido, aparece en cierta medida ruido *sal y pimienta*. Además hay que señalar que el filtro *anisotropo_log* difumina un poco los bordes, mientras que en el caso de *anisotropo_doble* al tratarse de condiciones dobles y por lo tanto más restrictivas, se comporta mejor en éstos.

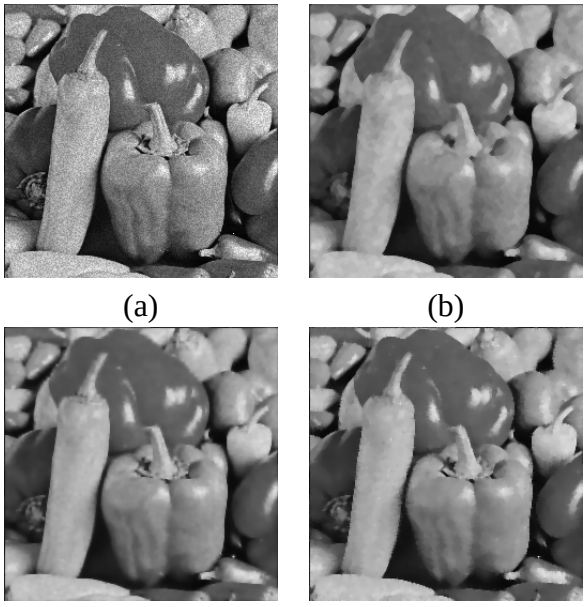
A continuación se va a hacer una comparativa entre los resultados obtenidos para los filtros diseñados y el filtro clásico de mediana.



(a) (b)

(c) (d)

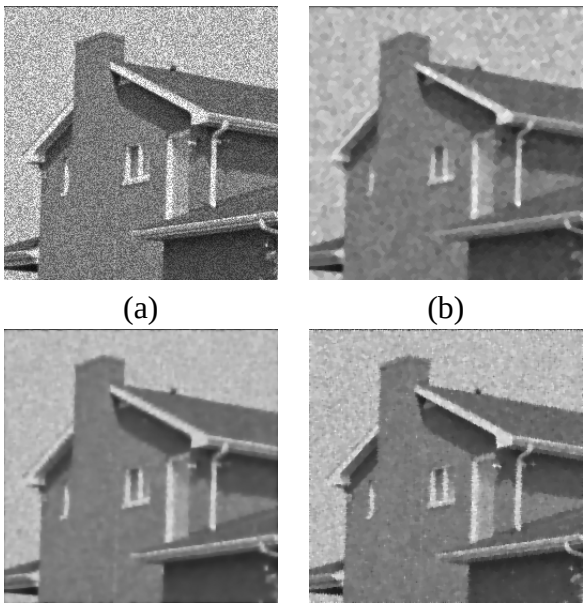
Figura 7.43: Casa. Ruido *speckle* (m = 0, var = 0.01). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

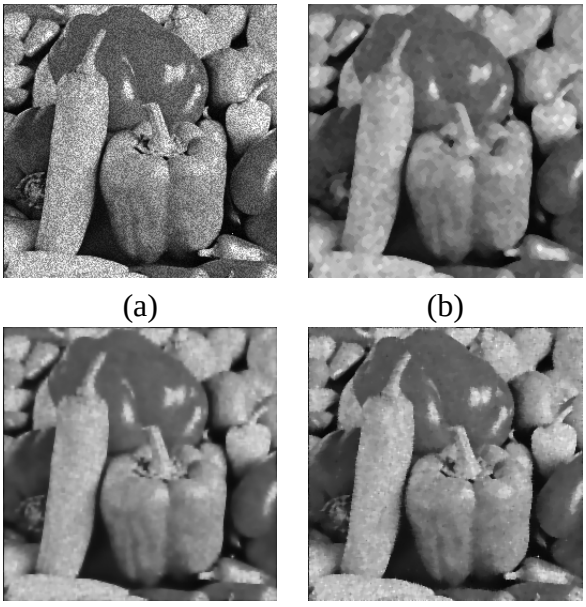
Figura 7.44: Pimiento. Ruido *speckle* (m = 0, var = 0.01). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

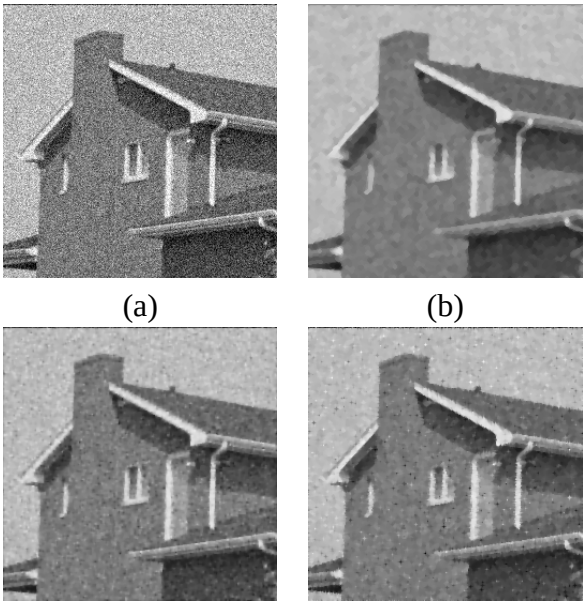
Figura 7.45: Casa. Ruido *speckle* (m = 0, var = 0.04). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

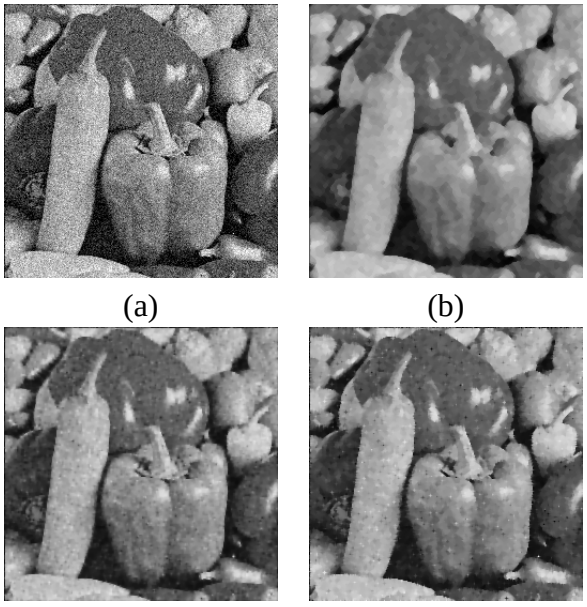
Figura 7.46: Ruido *speckle* (m = 0, var = 0.04). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

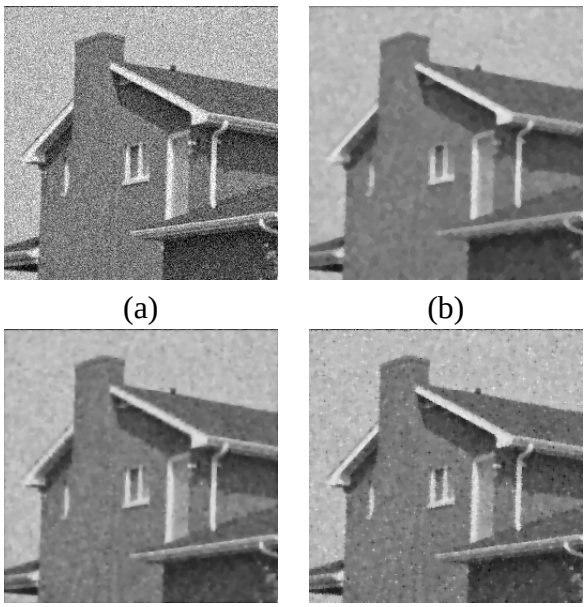
Figura 7.47: Casa. Ruido *gaussiano* (m = 0, var = 0.01). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

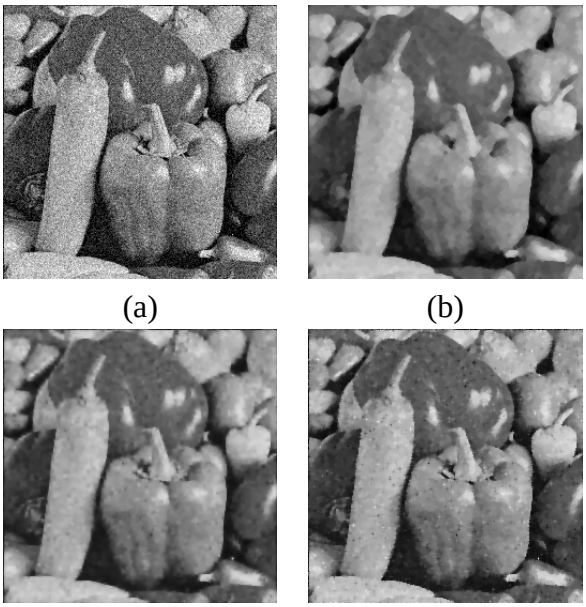
Figura 7.48: Ruido *gaussiano* (m = 0, var = 0.01). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

Figura 7.49: Casa. Ruido *gaussiano* (m = 0, var = 0.04). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

Figura 7.50: Ruido *gaussiano* (m = 0, var = 0.01). (a) Imagen ruidosa; (b) mediana; (c) anisotropo_log; (d) anisotropo_doble

Dados los resultados observados se puede decir que aunque el filtro de mediana también elimina el ruido, lo hace en menor grado. Además hay zonas que no se hacen completamente homogéneas cuando deberían de ser lo, y da lugar a un mayor grado de distorsión en los bordes. Por otra parte las imágenes obtenidas tienen un aspecto más artificial que en el caso de difusión anisótropa borrosa.

Para poder extraer conclusiones cuantitativas se va a utilizar el índice de calidad, *SSIM, (Structural Similarity Index Measure)*. En un principio se consideraron otros índices como el E_{mse}, *Mínimo Error Cuadrático Medio*, o el $factor\ Q$. Sin embargo al final se observó que estos índices no eran adecuados, pues para imágenes que presentaban un mismo E_{mse}, la calidad con la que se percibían era muy diferente; lo mismo sucedía para el $factor\ Q$. Al final se decidió tomar el índice *SSIM* como medida de la calidad de una imagen, pues mejora el $factor\ Q$ al tener en cuenta información estructural.

Los datos recogidos en las tablas que se muestran a continuación, no se corresponden con las conclusiones extraídas visualmente, lo que refleja que el *SSIM* tampoco es un índice del todo adecuado para los objetivos que se persiguen conseguir con los filtros diseñados.

Como se puede ver en los resultados obtenidos con este *toolbox*, concretamente con el filtro anisotropo_doble, se consigue eliminar bastante ruido de la imagen (no todo) conservando los bordes, por lo que este tipo de filtrado se podría utilizar como técnica de restauración.

$NAIteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,8295	0,8702	0,8711	0,8694	0,8658	0,8610
Anisotropo_doble	0,7797	0,8383	0,8607	0,8665	0,8644	0,8586
Mediana	0,8214	0,8193	0,8191	0,8190	0,8190	0,81893

Cuadro 7.1: Pimiento. Ruido *speckle*. Var=0.01, m=0

$N^{o}Iteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,6027	0,7052	0,7661	0,7949	0,7986	0,8005
Anisotropo_doble	0,5367	0,6225	0,6861	0,7322	0,7637	0,7835
Mediana	0,7897	0,7895	0,7890	0,7888	0,7887	0,78866

Cuadro 7.2: Pimiento. Ruido *speckle*. Var=0.04, m=0

$N^{o}Iteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,7457	0,8254	0,8343	0,8390	0,8405	0,8400
Anisotropo_doble	0,6737	0,7574	0,8009	0,8214	0,8288	0,8294
Mediana	0,7341	0,7339	0,7339	0,7338	0,7338	0,7337

Cuadro 7.3: Casa. Ruido *speckle*. Var=0.01, m=0

$N^{o}Iteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,4708	0,5890	0,6716	0,7297	0,7639	0,7699
Anisotropo_doble	0,4008	0,4913	0,5618	0,6168	0,6602	0,69369
Mediana	0,6706	0,6747	0,6749	0,6749	0,6749	0,6749

Cuadro 7.4: Casa. Ruido *speckle*. Var=0.04, m=0

$N^{o}Iteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,5151	0,6222	0,6487	0,6723	0,6929	0,7106
Anisotropo_doble	0,4649	0,5470	0,6121	0,6617	0,6968	0,7209
Mediana	0,76534	0,7646	0,7644	0,7644	0,7644	0,7644

Cuadro 7.5: Pimiento. Ruido *gaussiano*. Var=0.01, m=0

$N^{o}Iteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,5182	0,6244	0,7016	0,7470	0,7552	0,7613
Anisotropo_doble	0,4686	0,5513	0,6166	0,6655	0,6995	0,7222
Mediana	0,7464	0,7483	0,7482	0,7482	0,7481	0,74817

Cuadro 7.6: Pimiento. Ruido *gaussiano*. Var=0.04, m=0

$N^{o}Iteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,4214	0,5419	0,5751	0,6061	0,6344	0,6600
Anisotropo_doble	0,3691	0,4537	0,5267	0,5870	0,6336	0,6682
Mediana	0,7652	0,7646	0,7645	0,7644	0,7643	0,7643

Cuadro 7.7: Casa. Ruido *gaussiano*. Var=0.01, m=0

$N^{o}Iteraciones \vert Filtro$	10	20	30	40	50	60
Anisotropo_log	0,4249	0,5453	0,6458	0,7150	0,7295	0,74159
Anisotropo_doble	0,3730	0,4592	0,5324	0,5909	0,6357	0,6693
Mediana	0,7492	0,7504	0,7499	0,7497	0,7496	0,7496

Cuadro 7.8: Casa. Ruido *gaussiano*.Var=0.04, m=0

7.2.2. Ecografías: imágenes reales

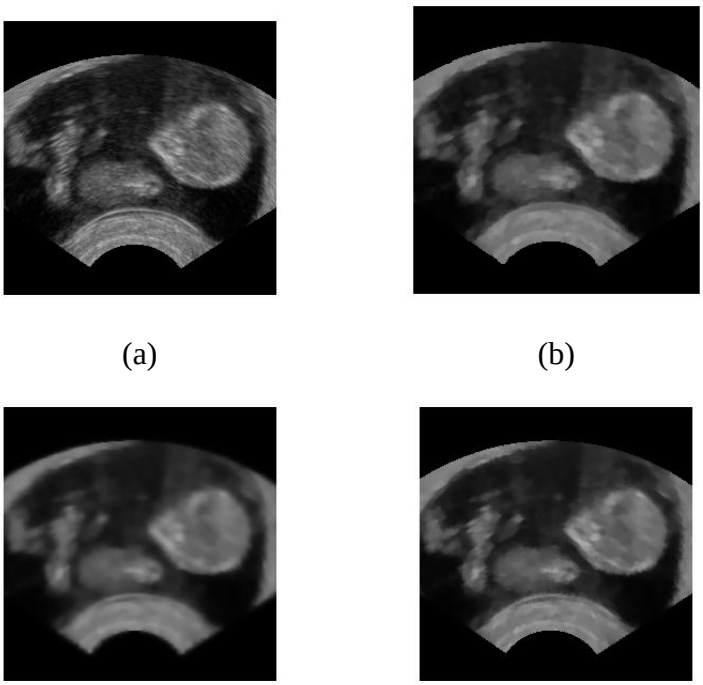
Después de haber visto los resultados obtenidos para imágenes sintéticas, en esta sección se mostrarán los experimentos realizados con imágenes ecográficas reales. Para la realización de tales experimentos se han utilizado dos volúmenes de datos de partida, uno correspondiente a un feto y otro a un riñón.

Como ya se ha señalado anteriormente, se está trabajando con filtros 2D, por lo que para poder trabajar sobre los datos con los filtros realizados se ha pasado cada uno de estos volúmenes a secciones, aplicando posteriormente los filtros sobre cada una de ellas por separado. Si se desea ver el efecto sobre el volumen total, una vez que se tienen todas las secciones procesadas, se han de renderizar mediante trazado de rayos usando la herramienta de visualización VTK (*Visualization ToolKit*). En ambos casos, una vez renderizadas las secciones procesadas se podrá comparar el volumen original (sin preprocesado) con el volumen procesado.

Inicialmente se va a comparar el resultado que se obtiene cuando se procesa una sección del feto mediante un filtro de mediana y el que se obtiene cuando se usa difusión anisótropa borrosa, que es lo que habitualmente se ha hecho hasta ahora, y lo mismo para una sección de riñón (ver Figs. 7.51, 7.52).

En base a los resultados, se puede decir que el comportamiento de los filtros es similar a cuando trabajan con imágenes sintéticas. Se observa como efectivamente el filtro de mediana elimina ruido, pero las zonas no se hacen completamente homogéneas y hay un mayor grado de distorsión en los bordes. Así mismo, el comportamiento del filtro de doble condición, *anisotropo_doble* es mejor, pues elimina ruido, suaviza más las zonas homogéneas y respeta los bordes.

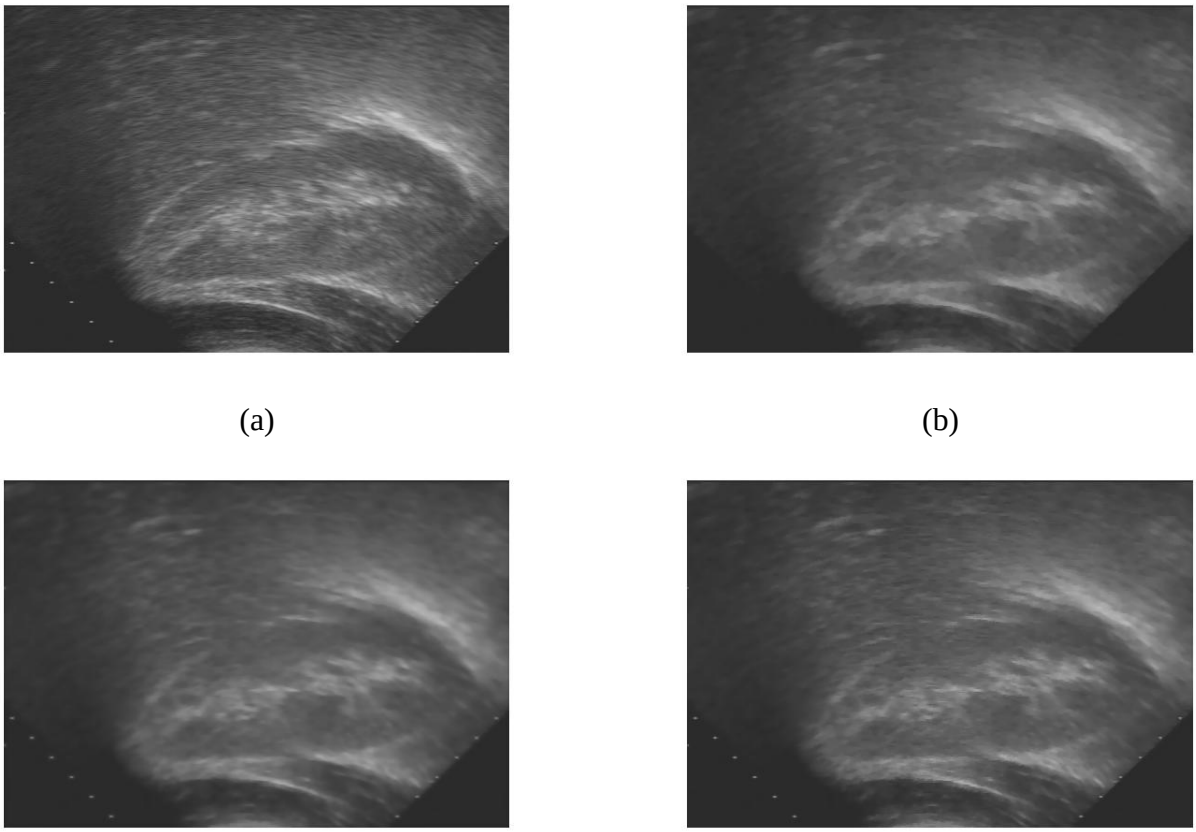
A continuación se muestra los volúmenes de la ecografía del feto (ver Fig. 7.71), que se consiguen renderizando las secciones procesadas. Evidentemente, como el diseño de todas las herramientas implementadas en este proyecto está orientado al trabajo de datos en dos dimensiones, los resultados que aquí se presentan podrían mejorarse si se realizase una extensión de los mismos a 3-D. Esto es así porque tal y como están implementados (2D) no se están considerando relaciones entre secciones consecutivas por lo que no se utiliza toda la información que está implícita en los datos. Aún así, como se verá, los resultados obtenidos son bastantes notables.



(a) (b)

(c) (d)

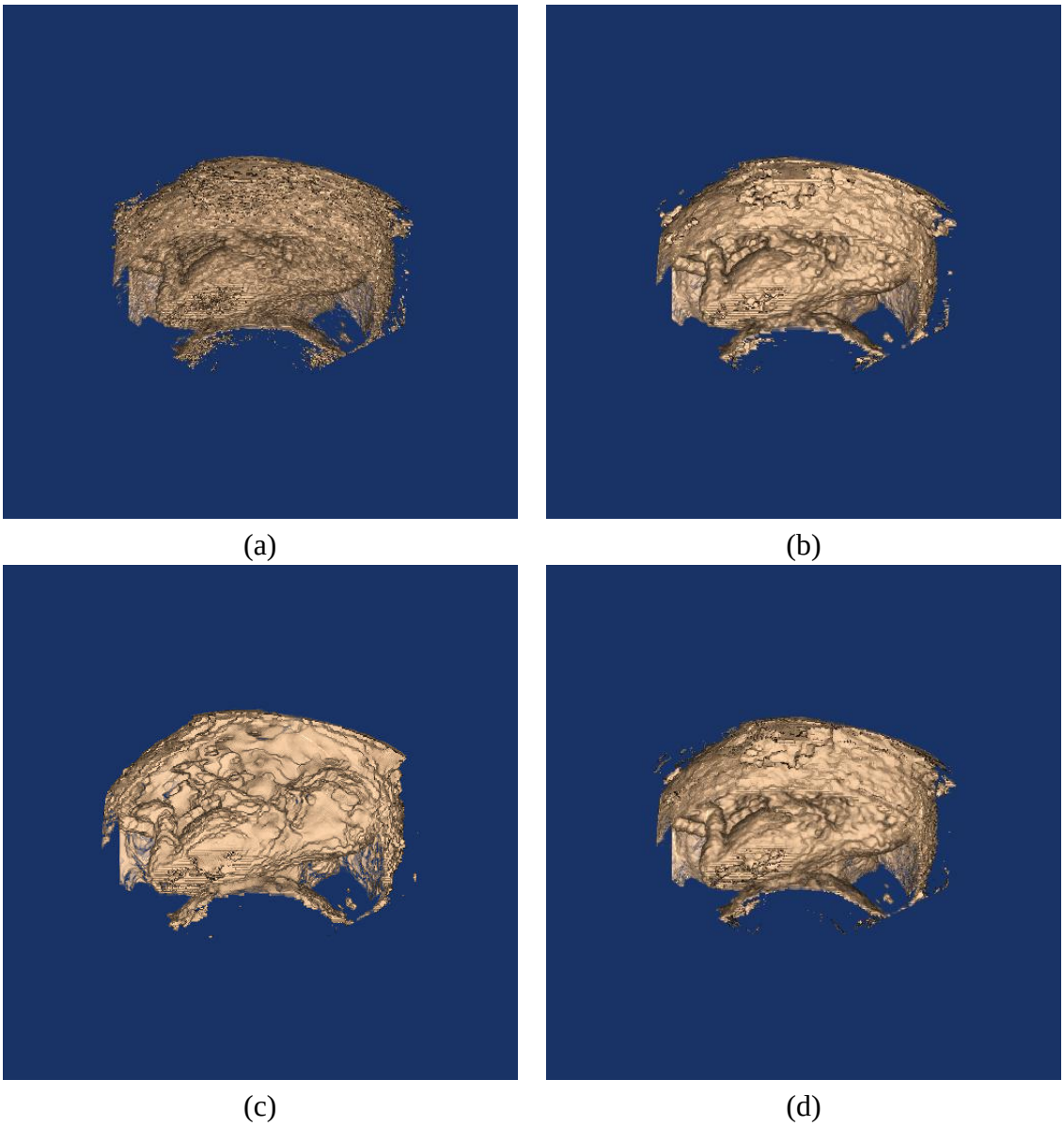
Figura 7.51: Feto. (a) Ecografía original. Ecografía tras ser procesada por el filtro: (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

Figura 7.52: Riñón. (a) Ecografía original. Ecografía tras ser procesada por el filtro: (b) mediana; (c) anisotropo_log; (d) anisotropo_doble



(a) (b)

(c) (d)

Figura 7.53: (a) Volumen del feto original. Resultado de aplicar el filtro: (b) mediana (60 iteraciones); (c) difusión anisótropa 2D (60 iteraciones); (d) anisotropo_doble (60 iteraciones)

7.3. Toolbox Difusión

Como se explicaba en el capítulo 6, en este *toolbox* se han implementado varias versiones borrosas de la regularización espacial del filtro de Perona-Malik realizada por Caté et al [Catte92]. Por lo tanto en este apartado se mostrarán los resultados obtenidos con estos nuevos filtros, y se compararán con aquellos resultantes de tratar tales imágenes con el filtro original de Caté et al.

Al igual que se ha venido haciendo hasta ahora, la forma de evaluar los resultados es fundamentalmente cualitativa, pues se partirá de una imagen ruidosa, se realzará dicha imagen mediante el filtro correspondiente y se verá en qué medida el resultado obtenido es adecuado. En el caso del análisis de imágenes sintéticas, se hará también una evaluación cuantitativa gracias el índice de calidad *SSIM*. Se intentará llegar a una conclusión en la que se decida cúal es la versión del filtro más adecuada, o al menos la más indicada para cada tipo de imagen. Finalmente, se verán las ventajas de utilizar el controlador borroso.

7.3.1. Imágenes sintéticas

Se comenzará analizando el comportamiento de los filtros para imágenes sintéticas. Dado que los resultados obtenidos para diferentes imágenes presentan las mismas características, y que el número de filtros que contiene este *toolbox* es elevado, solo se mostrarán los resultados de la imagen 'cameraman'. Como siempre se analizará la respuesta de los filtros ante el ruido *speckle* y el ruido *gaussiano*.

Imágenes afectadas por ruido speckle

A las imágenes originales se les va a añadir ruido *speckle* tal que $I_s = I_0 + I_0 \Delta r(\sigma_s^2)$ donde $\sigma_s^2 = 0{,}01$ y $\sigma_s^2 = 0{,}04$ para analizar su comportamiento en diferentes ambientes.

El procedimiento seguido, ha sido ejecutar cada filtro para todas las posibles combinaciones de los parámetros λ y *número de iteraciones*. El parámetro *m* se ha mantenido constante e igual a 12, y σ es igual a 1. Una vez recogidos todos los resultados se ha seleccionado el mejor para cada filtro.

En el caso de $\sigma_s^2 = 0{,}01$ los mejores resultados son los que se indican a continuación y se muestran el Fig. (7.54):

- nldif (filtro de Perona-Malik): λ = 4; *número de iteraciones* = 5;
- nldif_borroso: λ = 5; *número de iteraciones* = 4;
- nldif_borroso3: λ = 5; *número de iteraciones*= 20;
- nldif_borroso4: λ = 9; *número de iteraciones* = 20;

- nldif_borroso5: λ = 9; *número de iteraciones* = 20;

Para el caso de $\sigma_s^2 = 0{,}04$ se muestran en Fig. (7.55) y serían:

- nldif (filtro de Perona-Malik): λ = 8; *número de iteraciones* = 8;
- nldif_borroso: λ = 9; *número de iteraciones* = 12;
- nldif_borroso3: λ = 9; *número de iteraciones* = 24;
- nldif_borroso4: λ = 9; *número de iteraciones* = 20;
- nldif_borroso5: λ = 9; *número de iteraciones* = 50;

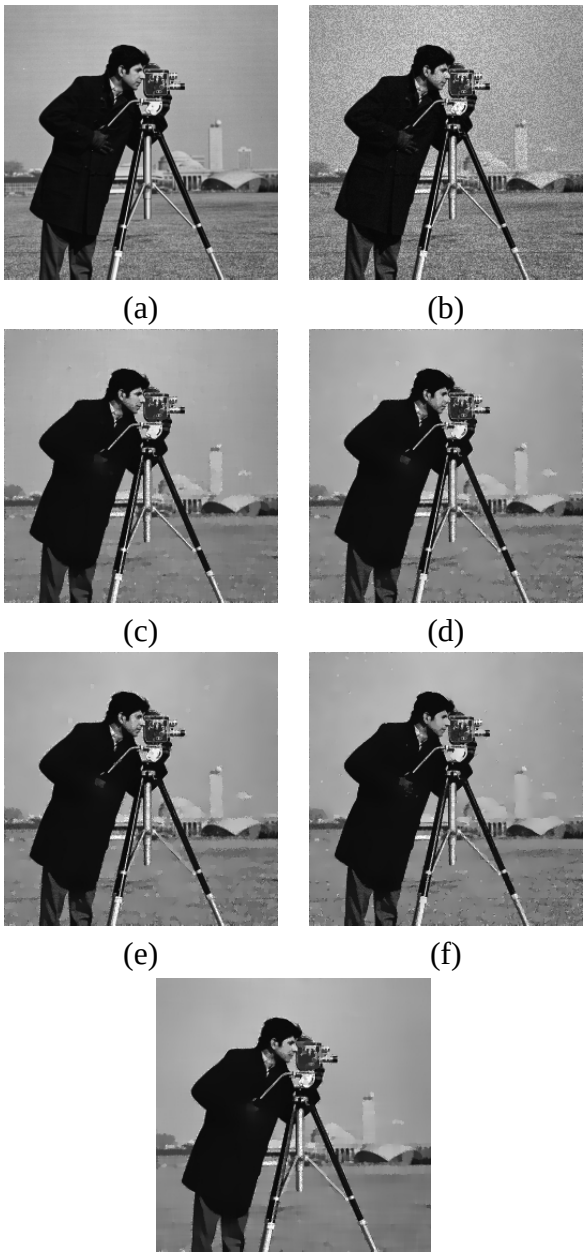
Este análisis se ha realizado manteniendo constante el parámetro λ. Sin embargo, como ya se comentó en el capítulo 6, lo ideal sería variar a lo largo del proceso este parámetro, de modo que a medida que el proceso de difusión llegue al final λ haya aumentado, alcanzando un valor elevado. Para ello se ha diseñado el filtro nldif_total, en el que se incorpora un controlador borroso al filtro de Perona-Malik realizado por Caté et al.

El resultado que se obtiene con este nuevo filtro para el caso de $\sigma_s^2 = 0{,}01$ se muestra en Fig. (7.56); para $\sigma_s^2 = 0{,}04$ ver Fig. (7.57). Además de las imágenes obtenidas se muestra la evolución del parámetro λ y del ruido presente en la imagen. Efectivamente se comprueba que a medida que el número de iteraciones aumenta, la imagen se vuelve más limpia y por lo tanto λ crece. Respecto al ruido, se observa que disminuye rápidamente, y que en el momento en el que nuevas iteraciones no producen mejoras significativas, se detiene la ejecución del bucle.

El *SSIM* para estos casos se recoge en la tabla 7.9.

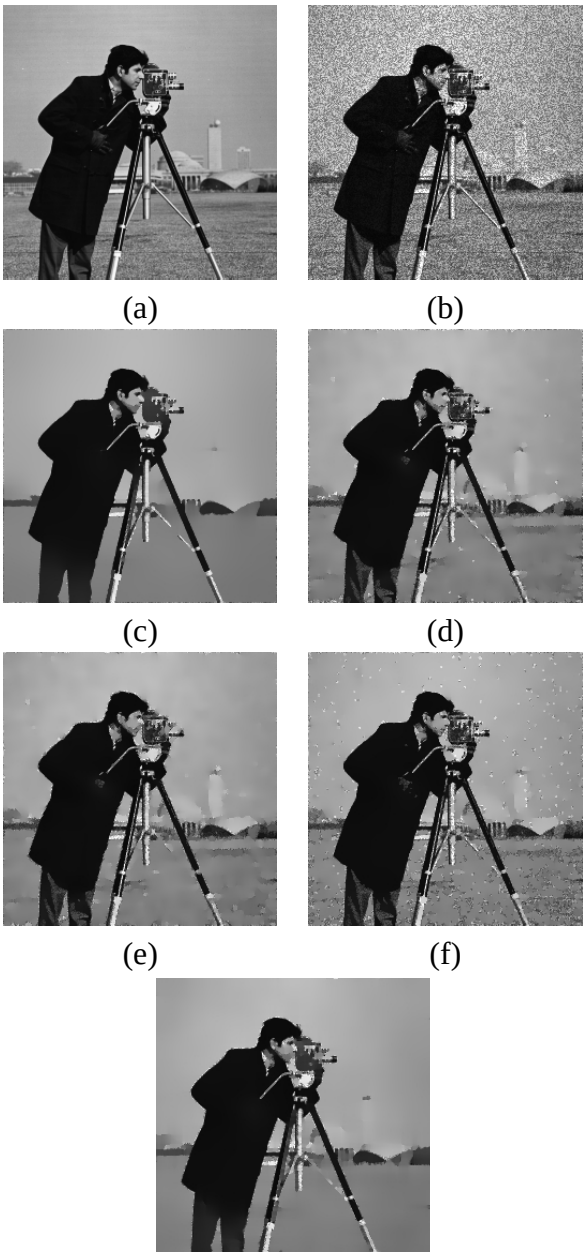
Ruido|Filtro	*nldif*	*nldif_ borroso*	*nldif_ borroso3*	*nldif_ borroso4*	*nldif_ borroso5*	*nldif_ total*
Speckle($\sigma_s^2$0,01)	0,83	0,86	0,86	0,85	0,85	0,77
Speckle($\sigma_s^2$0,04)	0,69	0,77	0,80	0,79	0,75	0,78

Cuadro 7.9: SSIM



(a) (b) (c) (d) (e) (f) (g)

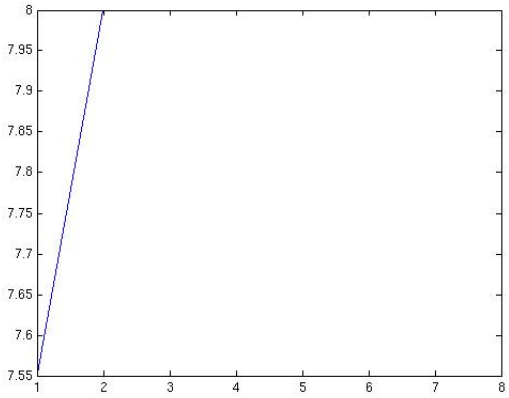
Figura 7.54: (a) Imagen ausente de ruido; (b) Imagen ruidosa (var = 0.04); Imagen filtrada con: (c) nldif_borroso; (d) nldif_borroso3; (e) nldif_borroso4; (f) nldif_borroso5; (g) versión original de *Caté et al.*



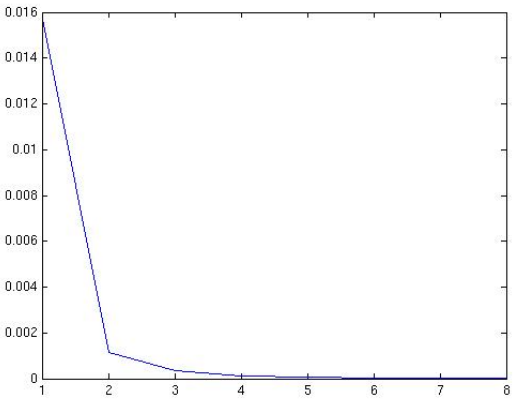
(a) (b) (c) (d) (e) (f) (g)

Figura 7.55: (a) Imagen ausente de ruido; (b) Imagen ruidosa (var = 0.04); Imagen filtrada con: (c) nldif_borroso; (d) nldif_borroso3; (e) nldif_borroso4; (f) nldif_borroso5; (g) versión original de *Caté et al.*

(a)



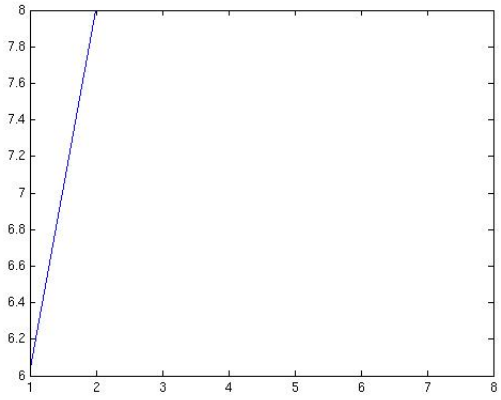
(b)



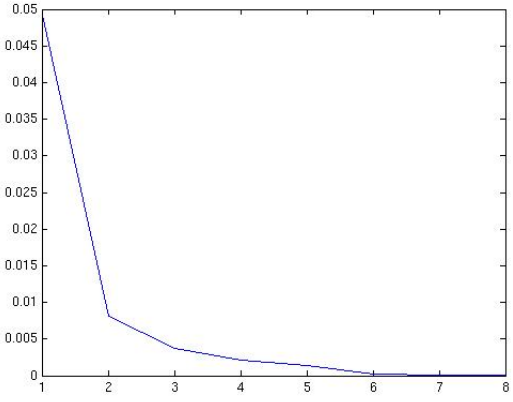
(c)

Figura 7.56: (a) Imagen filtrada con nldif_total (var = 0.01); (b) λ utilizada; (c) Evolución del ruido en la imagen (*Cu*)

(a)



(b)



(c)

Figura 7.57: Imagen filtrada con nldif_total (var = 0.04); (b) λ utilizada; (c) Evolución del ruido en la imagen (*Cu*)

Imágenes afectadas por ruido gaussiano

Ahora se estudiará el tratamiento de imágenes que presentan ruido *gaussiano* tal que $I_s = I_0 + \eta(\sigma_g^2)$. Se tomará una varianza $\sigma_g^2 = 0{,}01$ y $\sigma_g^2 = 0{,}04$. La media será nula en todos los casos.

Se seguirá el mismo procedimiento que en el caso del ruido *speckle*. Por el mismo motivo, solo se mostrarán los mejores resultados obtenidos para el caso del 'cameraman'.

Para el caso de $\sigma_s^2 = 0{,}01$ los mejores resultados son los que se indican a continuación y se muestran el Fig. (7.58):

- nldif (filtro de Perona-Malik): λ = 7; *número de iteraciones* = 8;
- nldif_borroso: λ = 7; *número de iteraciones* = 8;
- nldif_borroso3: λ = 9; *número de iteraciones* = 20;
- nldif_borroso4: λ = 5; *número de iteraciones* = 24;
- nldif_borroso5: λ = 9; *número de iteraciones* = 35;

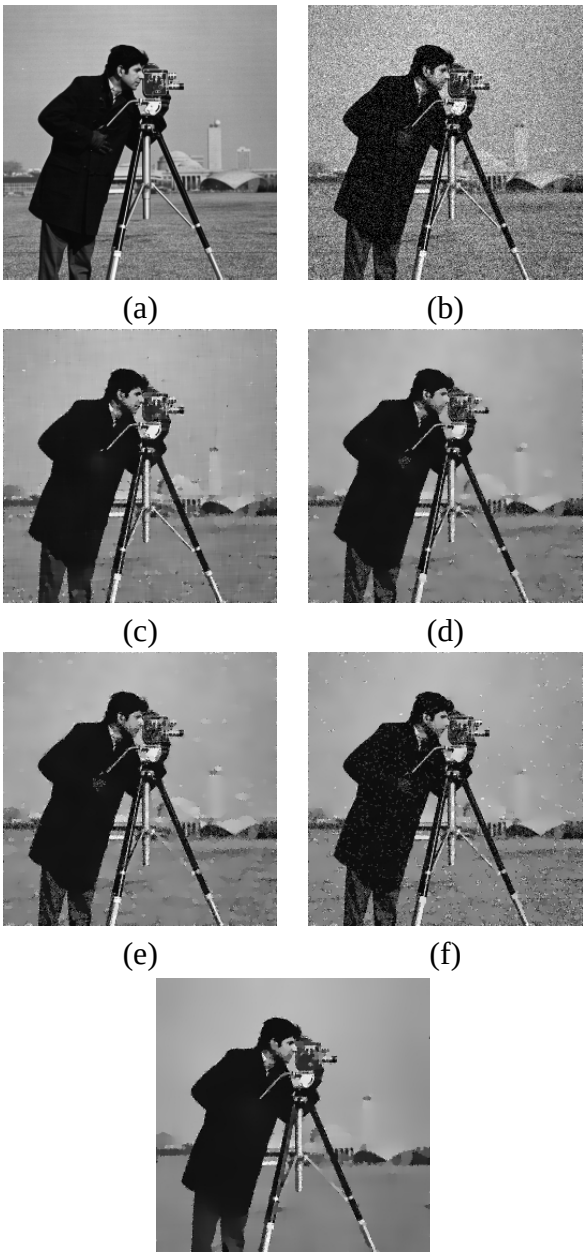
Para el caso de $\sigma_s^2 = 0{,}04$ se muestran en Fig. (7.59) y serían:

- nldif (filtro de Perona-Malik): λ = 8; *número de iteraciones* = 12;
- nldif_borroso: λ = 9; *número de iteraciones* = 50;
- nldif_borroso3: λ = 9; *número de iteraciones* = 50;
- nldif_borroso4: λ = 8; *número de iteraciones*= 40;
- nldif_borroso5: λ = 9; *número de iteraciones* = 50;

Al igual que se ha hecho para el ruido *speckle* se recoge el índice de calidad *SSIM* (tabla 7.10), y se muestran los resultados obtenido con el fitro nldif_total.

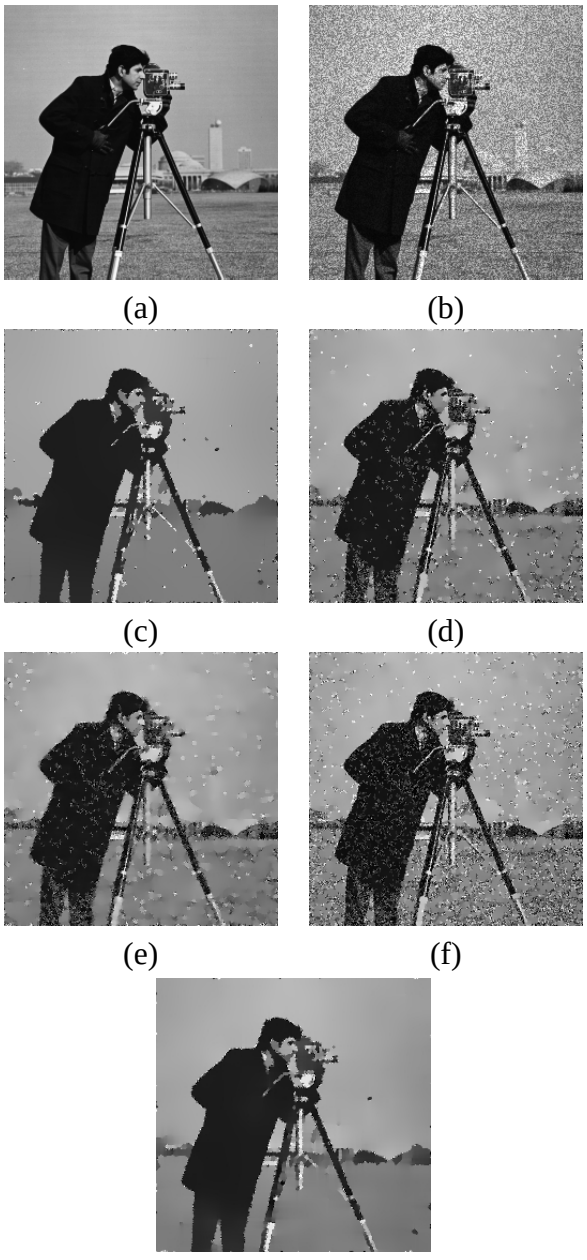
Ruido|Filtro	*nldif*	*nldif_ borroso*	*nldif_ borroso*3	*nldif_ borroso*4	*nldif_ borroso*5	*nldif_ total*
Gaussiano($\sigma_g^2$0,01)	0,76	0,80	0,78	0,78	0,70	0,76
Gaussiano($\sigma_g^2$0,04)	0,77	0,67	0,61	0,60	0,43	0,67

Cuadro 7.10: SSIM



(a) (b) (c) (d) (e) (f) (g)

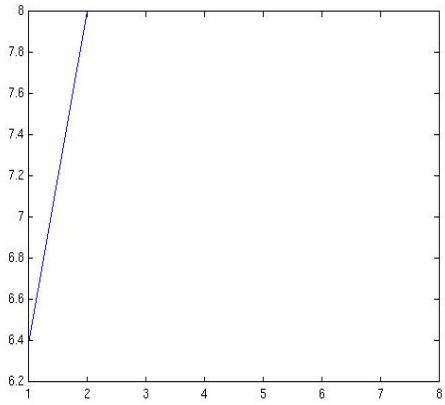
Figura 7.58: (a) Imagen ausente de ruido; (b) Imagen ruidosa (var = 0.04); Imagen filtrada con: (c) nldif_borroso; (d) nldif_borroso3; (e) nldif_borroso4; (f) nldif_borroso5; (g) versión original de *Caté et al.*



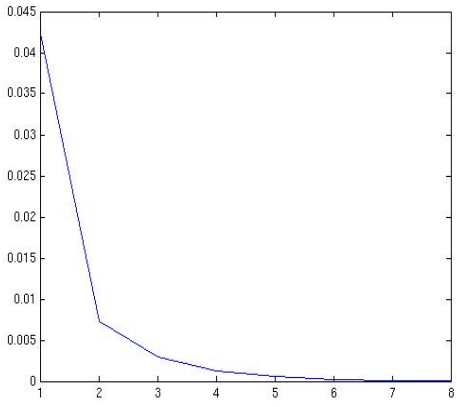
(a) (b) (c) (d) (e) (f) (g)

Figura 7.59: (a) Imagen ausente de ruido; (b) Imagen ruidosa (var = 0.04); Imagen filtrada con: (c) nldif_borroso; (d) nldif_borroso3; (e) nldif_borroso4; (f) nldif_borroso5; (g) versión original de *Caté et al.*

(a)



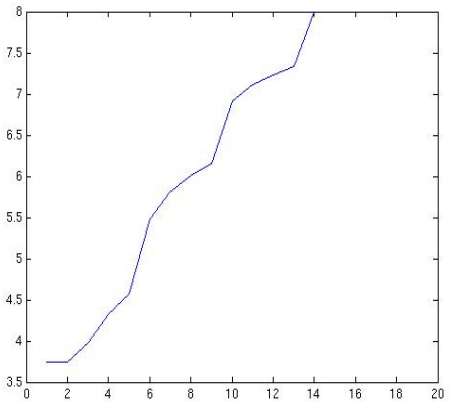
(b)



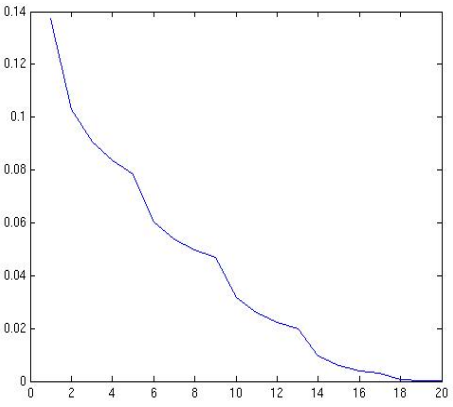
(c)

Figura 7.60: (a) Imagen filtrada con nldif_total (var = 0.01); (b) λ utilizada; (c) Evolución del ruido en la imagen (*Cs*)

(a)



(b)



(c)

Figura 7.61: Imagen filtrada con nldif_total (var = 0.04);(b) λ utilizada;(c) Evolución del ruido en la imagen (*Cs*)

Análisis de los resultados

Se observa que los filtros borrosos diseñados presentan peor comportamiento para el caso de ruido *gaussiano*. Además en base a los resultados obtenidos para las imágenes analizadas, hay que decir que las versiones borrosas nldif_borroso3, nldif_borroso4 y nldif_borroso5 no solo no presentan mejora alguna, sino que los resultados a los que conducen son peores. Esto se debe a que estos tres filtros incluyen en su algoritmo el filtro HPFborroso que como ya se ha visto no era adecuado para el caso de imágenes sintéticas. Sin embargo con el filtro nldif_borroso, que incluye el filtro sscfc, y nldif_total, que implementa el controlador, los resultados son bastantes buenos.

Se va a intentar explicar el comportamiento de los filtros. Para ello se comenzará viendo cual es la imagen suavizada con cada uno de los métodos.

(a) (b) (c)

Figura 7.62: (a) Imagen ruidosa *speckle* (var = 0.01); (b) Convolucionada con una gaussiana; (c) Filtrada con el filtro borroso ssfcf

(d) (e) (f)

Figura 7.63: (a) Imagen ruidosa *gaussiano* (var = 0.01); (b) Convolucionada con una gaussiana; (c) Filtrada con el filtro borroso ssfcf

El suavizado que produce cada uno de los métodos es diferente y debido a ello el modo en el que se comportan los filtros también. Como se puede observar si se comparan los dos tipos de suavizado, aunque los dos suavizan la imagen el filtro ssfcf elimina menos

ruido y respeta más los bordes. Esto se traduce en que el valor del parámetro λ a introducir en cada filtro ha de ser diferente. Así si un valor de λ adecuado para el filtro nldif (filtro de Perona-Malik realizado por Caté el al.) es introducido en el filtro nldif_borroso tratará el ruido como bordes, y al revés, es decir, si un valor idóneo para nldif_borroso se introduce en nldif difuminará los bordes al ser interpretados estos como ruido.

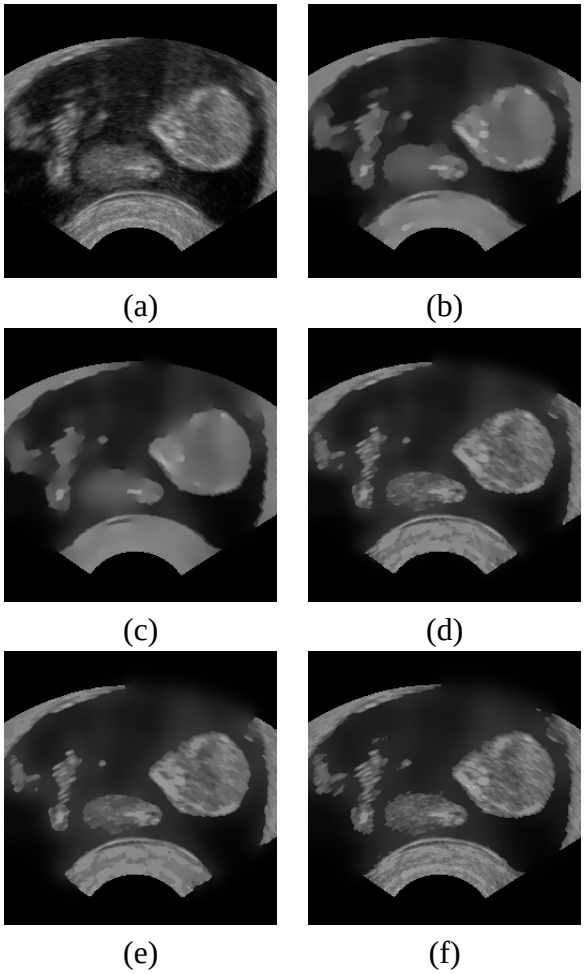
7.3.2. Ecografías: imágenes reales

Después de haber analizado el comportamiento de los filtros de este *toolbox* para imágenes sintéticas, se va a hacer un estudio sobre el tratamiento de imágenes obtenidas mediante ultrasonidos. Para ello se hará uso de los volúmenes de datos que se han venido utilizando hasta ahora, es decir, se tratarán con estos futuros el volumen correspondiente a un feto y a un riñón. El procedimiento a seguir será el usual en este proyecto, pues primero se trabajará en 2D con cada una de las secciones de los volúmenes por separado, y finalmente se renderizarán mediante la herramienta de visualización VTK (*Visualization ToolKit*). Así se podrá comparar el volumen original (sin preprocesado) y el volumen conseguido después de filtrar los datos originales con los métodos implementados.

Se comenzará mostrando los resultados obtenidos al tratar con los filtros nldif_borroso, nldif_borroso3, nldif_borroso4 y nldif_borroso5, una sección del feto. Al igual que se hizo para el tratamiento de las imágenes sintéticas se probarán los filtros para varios valores de los parámetros y se recogerán los resultados mostrando aquí los mejores para cada opción. Estos son los que se muestran en Fig. (7.64).

- nldif (filtro de Perona-Malik): λ = 5; *número de iteraciones* = 25 ;
- nldif_borroso: λ = 7; *número de iteraciones* = 25 ;
- nldif_borroso3: λ = 7; *número de iteraciones* = 30;
- nldif_borroso4: λ = 7; *número de iteraciones*= 40;
- nldif_borroso5: λ = 7; *número de iteraciones* = 50;

De los resultados obtenidos se puede decir que el comportamiento de los filtros borrosos para el tratamiento de ecográficas reales es bastante bueno. En Fig. (7.64) se muestran las imágenes obtenidas al tratar una sección del feto con los diferentes filtros. Se ve que el filtro nldif_borroso no presenta un buen comportamiento para ecografías, sin embargo el resto de los filtros borrosos, en los que se ha sustituido el gradiente por el filtrado HPFborroso, consiguen resultados muy buenos. Así en las imágenes que se obtienen por salida se ha conseguido eliminar gran parte del ruido speckle (en las zonas oscuras completamente), y además se respetan los bordes en el proceso de difusión.



(a) (b)

(c) (d)

(e) (f)

Figura 7.64: (a) Sección original; (b)filtro de Perona-Malik; (c) nldif_borroso (d) nldif_borroso3; (e) nldif_borroso4; (f) nldif_borroso5

Para intentar explicar los resultados a los que se llegan, se va a mostrar tanto la imagen correspondiente a la sección suavizada como la versión paso alto de la imagen para cada uno de los filtros borrosos (Fig. 7.65). Se ve que el problema a la hora de trabajar con ecografías está en la detección de bordes. Si se intenta hallar éstos por medio de un gradiente normal, los resultados obtenidos no son buenos, ya que el gradiente es un detector de bordes malo para imágenes ecográficas. Sin embargo, cuando se utiliza el filtro HPFborroso los bordes se detectan bastante bien, sobre todo si la imagen no presenta ningún tipo de suavizado (nldif_borroso5).

Se mostrarán también los resultados para el caso de una ecografía del riñón (Fig. 7.66). Para esta sección los mejores resultados se consiguen con:

- nldif (filtro de Perona-Malik): λ = 4; *número de iteraciones* = 20;
- nldif_borroso: λ = 4; *número de iteraciones* = 20 ;
- nldif_borroso3: λ = 8; *número de iteraciones* = 20;
- nldif_borroso4: λ = 8; *número de iteraciones*= 30;
- nldif_borroso5: λ = 8; *número de iteraciones* = 30;

En los experimentos realizados hasta este punto el parámetro λ toma el mismo valor en todas las iteraciones. Además, tanto éste como el número de iteraciones son prefijados antes de ejecutar el filtro. A continuación se va a probar el filtro que incorpora el controlador borroso, nldif_total, tanto con la sección del feto (Fig. 7.67) como la del riñón (Fig. 7.68). Los resultados obtenidos no son muy buenos, por lo tanto se va a añadir el controlador borroso al filtro nldif_borroso5 que como se había visto era el que mejor se comportaba ante imágenes reales. Así se tiene el filtro nldif_borroso5_cont cuyos resultados se muestran en Fig. (7.69) y Fig. (7.70).

En vista de los experimentos realizados se puede decir que el comportamiento de las versiones borrosas diseñadas para el tratamiento de ecografías es bastante bueno. Se observa como efectivamente en las imágenes que se obtienen por salida se ha conseguido eliminar gran parte del ruido speckle y se respetan los bordes en el proceso de difusión, en especial el filtro nldif_borroso5. Así mismo cuando a este se le incorpora un controlador borroso, nldif_borroso5_cont, se siguen obteniendo buenos resultados. Por lo tanto se tratará cada una de las secciones del volumen del feto con este filtro, y a continuación se renderizará mediante la herramienta de visualización VTK (*Visualization ToolKit*) mostrando en Fig.(7.71) el volumen resultante.

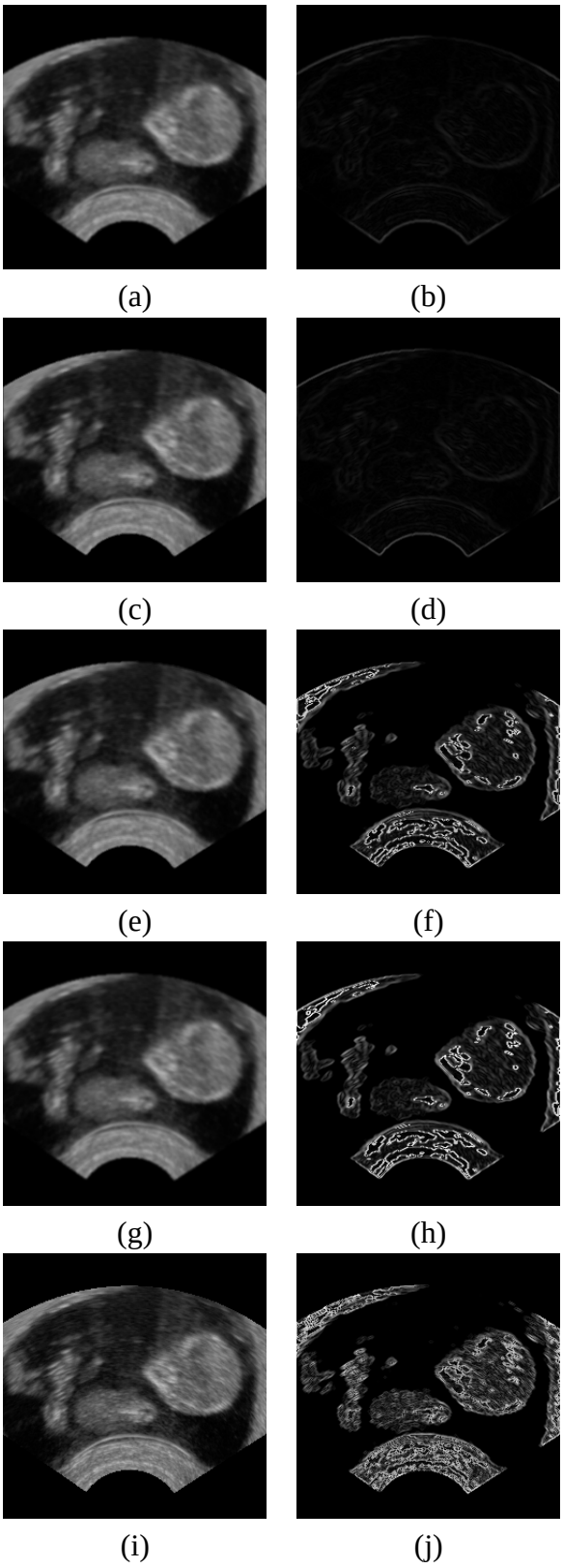
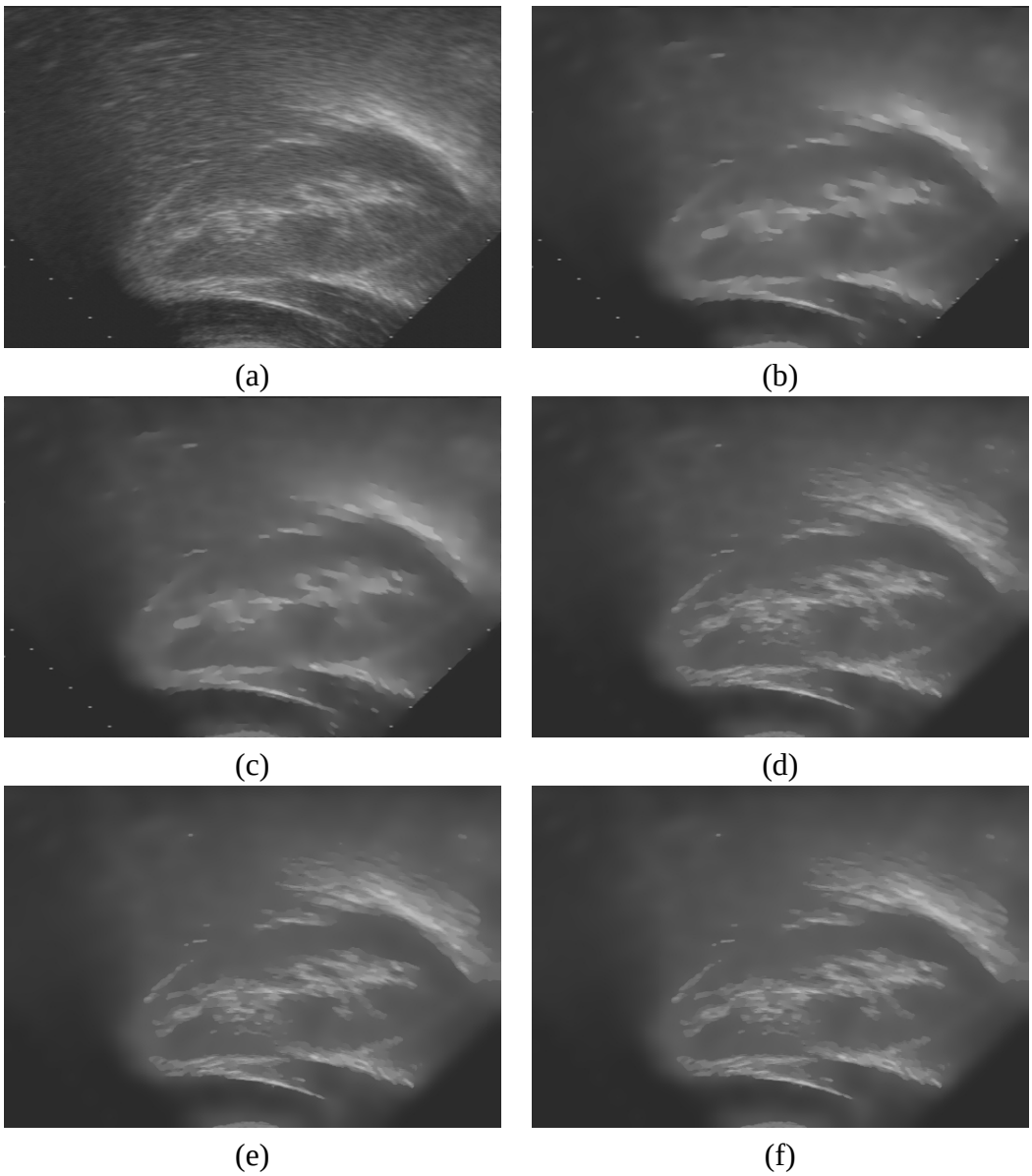


Figura 7.65: Imagen suavizada y paso bajo para: (a,b) filtro de Perona-Malik; (c,d) nldif_borroso; (e,f) nldif_borroso3; (g,h) nldif_borroso4; (c,d) nldif_borroso5

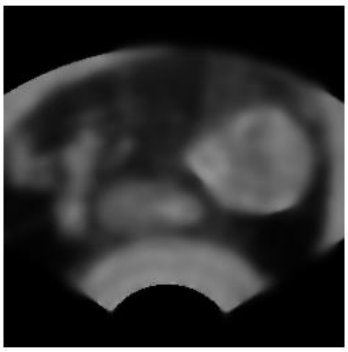


(a) (b)

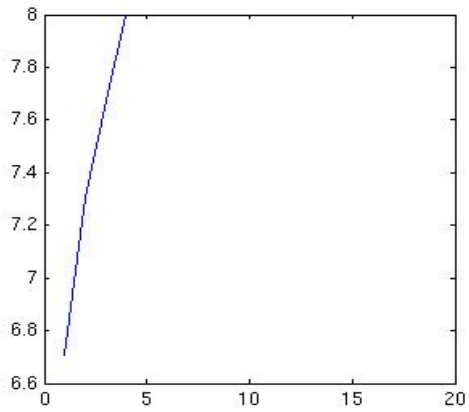
(c) (d)

(e) (f)

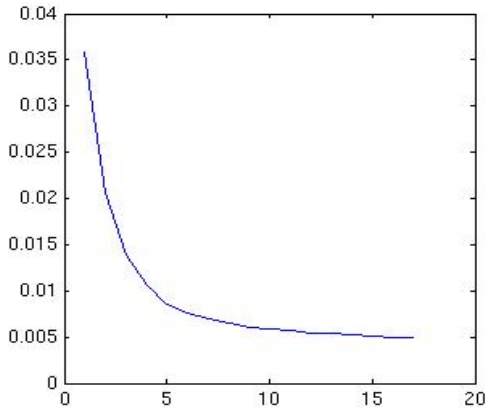
Figura 7.66: (a) Sección original; (b)filtro de Perona-Malik; (c) nldif_borroso (d) nldif_borroso3; (e) nldif_borroso4; (f) nldif_borroso5



(a)

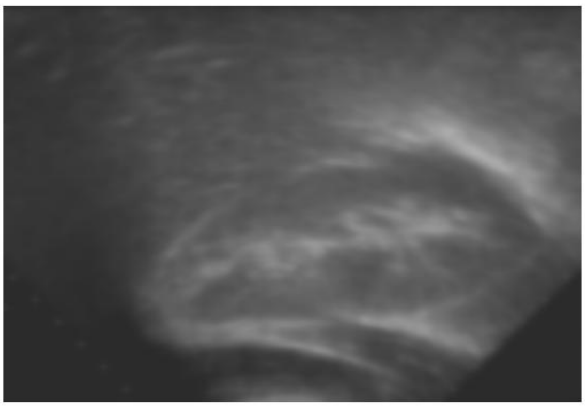


(b)

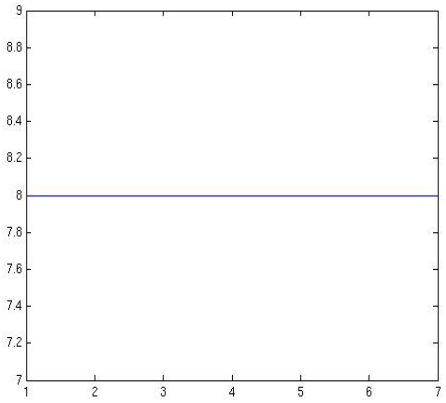


(c)

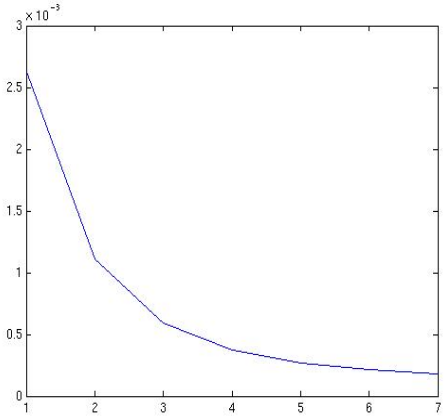
Figura 7.67: (a) Imagen filtrada con nldif_total (var = 0.01); (b) λ utilizada; (c) Evolución del ruido en la imagen (*Cu*)



(a)

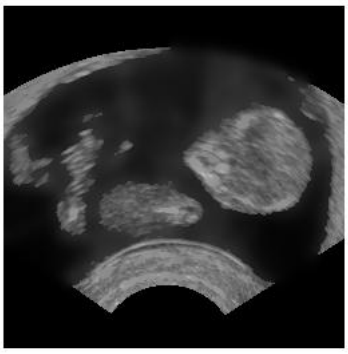


(b)

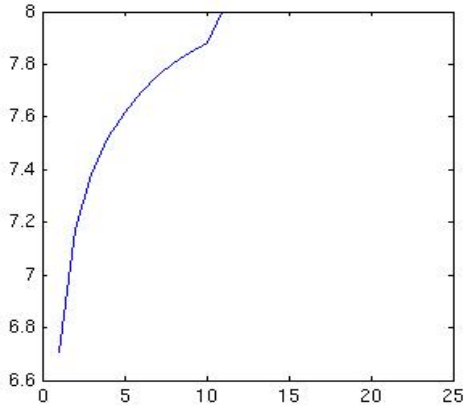


(c)

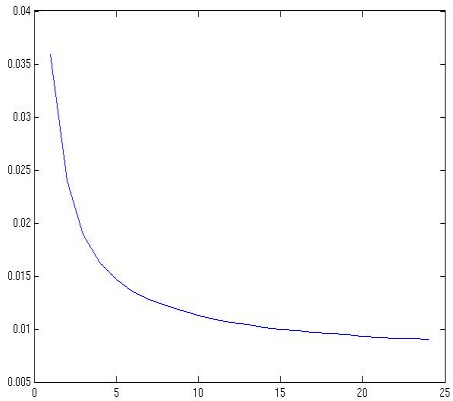
Figura 7.68: (a) Imagen filtrada con nldif_total (var = 0.01); (b) λ utilizada; (c) Evolución del ruido en la imagen (*Cu*)



(a)

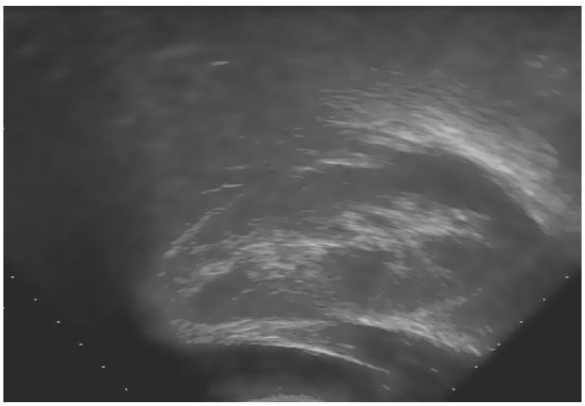


(b)

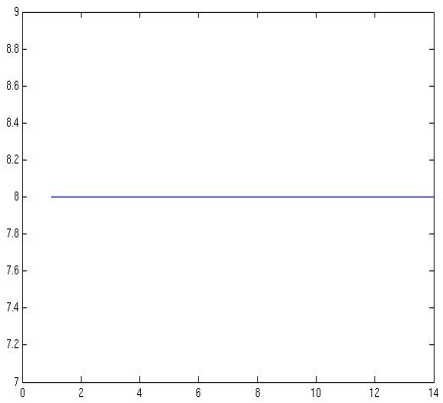


(c)

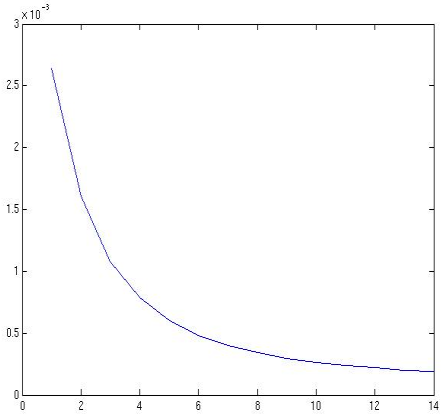
Figura 7.69: (a) Imagen filtrada con nldif_borroso5_cont (var = 0.01); (b) λ utilizada; (c) Evolución del ruido en la imagen (*Cu*)



(a)

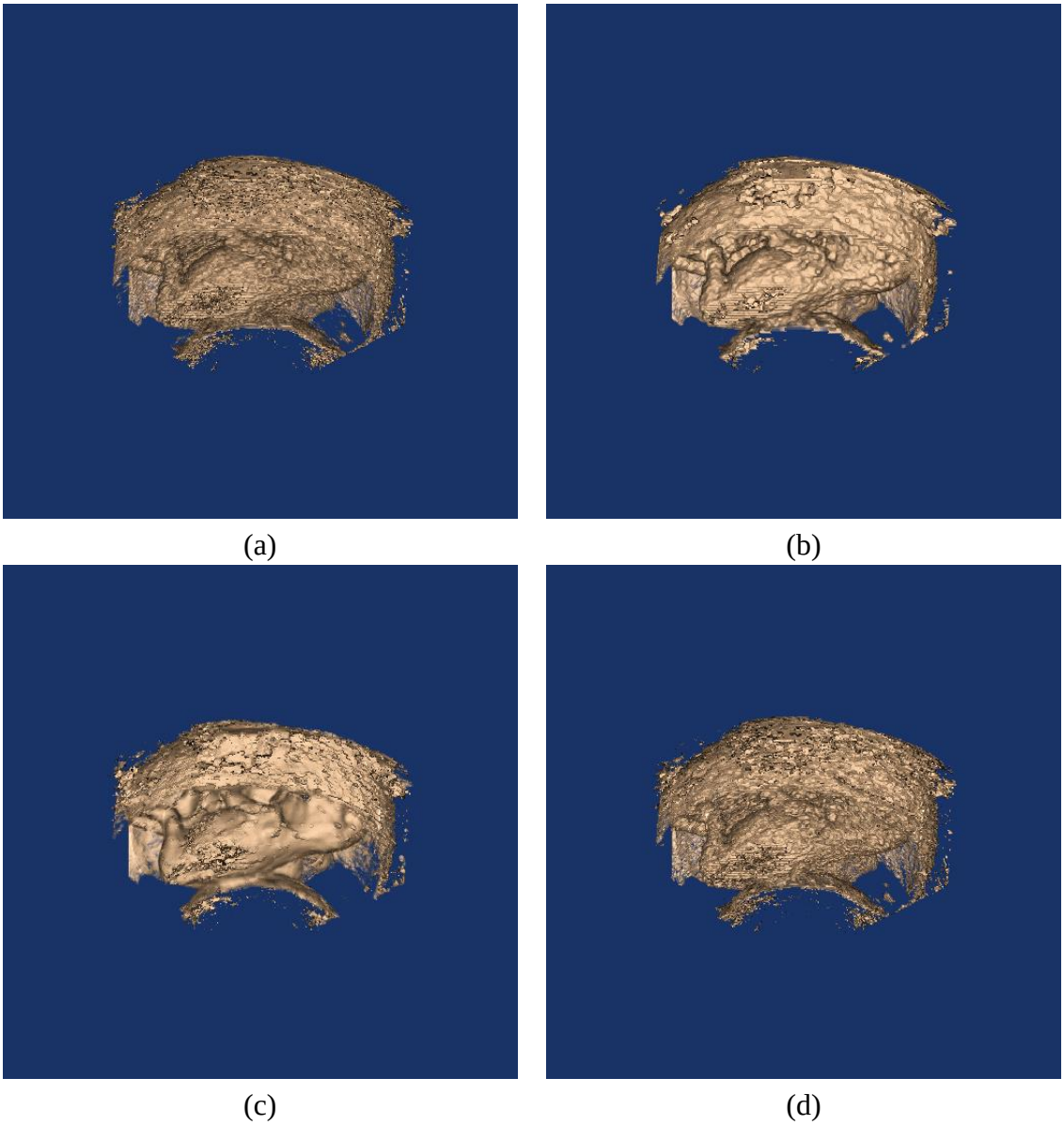


(b)



(c)

Figura 7.70: (a) Imagen filtrada con nldif_borroso5_cont (var = 0.01); (b) λ utilizada; (c) Evolución del ruido en la imagen (*Cu*)



(a) (b)

(c) (d)

Figura 7.71: (a) Volumen del feto original. Resultado de aplicar el filtro: (b) mediana (60 iteraciones); (c) nlfid_borroso5_cont (3 conjuntos borrosos para la detección de bordes); (d) nlfid_borroso5_cont (4 conjuntos borrosos para la detección de bordes)

7.4. Toolbox anisótropo vs Toolbox difusión

En ambos *toolboxes* se han implementado filtros de difusión anisótropa, pero cada uno corresponde a diferentes algoritmos. Hasta aquí se ha realizado un análisis de los resultados obtenidos con los filtros que corresponden a cada *toolbox* por separado, pero no se ha hecho una comparativa entre ellos, lo cual sería razonable ya que con ambos se persiguen los mismos objetivos. Por lo tanto en esta sección se va a realizar tal comparativa, pero tomando de cada *toolbox* el filtro con el que se consigue mejores resultados según el tipo de imagen.

7.4.1. Imágenes sintéticas

Como anteriormente se ha podido comprobar el filtro del *toolbox difusión* que mejor se comporta con imágenes sintéticas es la versión borrosa nldif_borroso. Por otra parte, en el *toolbox anisótropo*, el que conseguía mejores resultados era el filtro anisotropo_doble. Por lo tanto se compararán las imágenes resultantes tras ser tratadas con ambos filtros. Una vez más se utilizarán como imagen el 'cameraman' y 'pimiento'.

Imágenes afectadas por ruido speckle

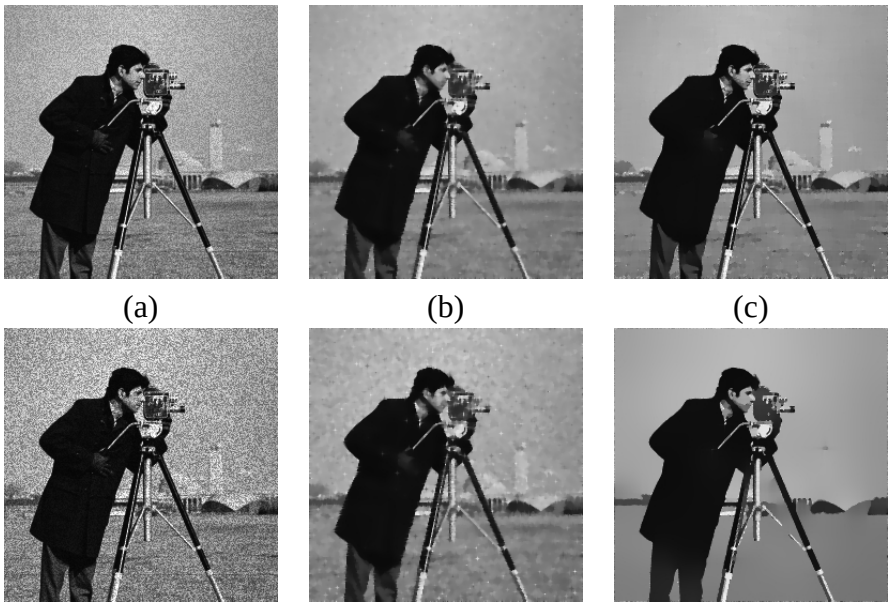
A continuación se muestran los mejores resultados conseguidos con ambos *toolboxes* para las dos imágenes. Como se puede ver cualitativamente (Figs. 7.72, 7.73), el filtro nldif_borroso consigue mejores resultados. Lo mismo indica el índice de calidad *SSIM* que se recoge en la tablas 7.11 y 7.12.

Filtro — Ruido ***speckle***	0,01	0,04
Anisotropo_doble	0,82	0,69
Nldif_borroso	0,86	0,77

Cuadro 7.11: Casa. Ruido *speckle*

Filtro — Ruido ***speckle***	0,01	0,04
Anisotropo_doble	0,85	0,78
Nldif_borroso	0,87	0,78

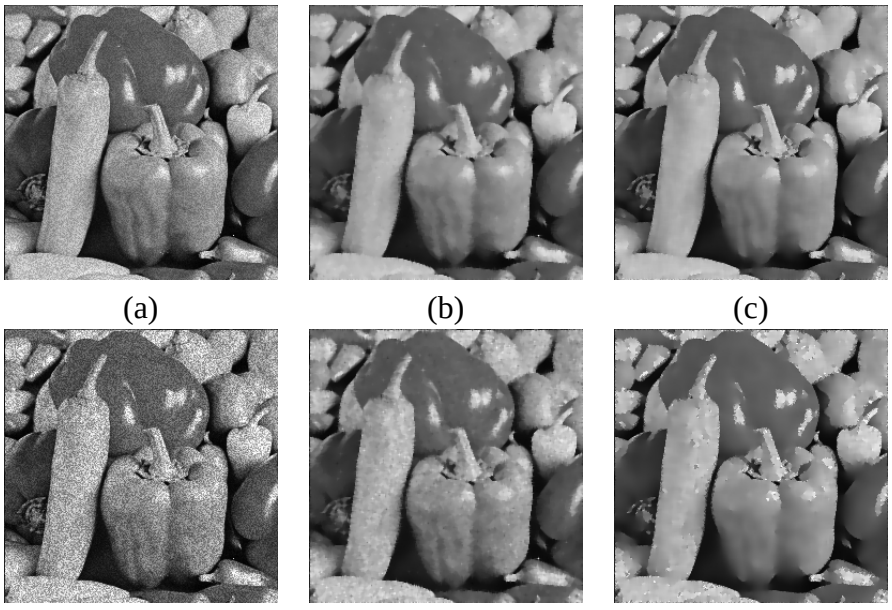
Cuadro 7.12: Pimiento. Ruido *speckle*



(a) (b) (c)

(d) (e) (f)

Figura 7.72: (a) Imagen ruidosa (var 0.01); (b) anisotropo_doble (60 iteraciones); (c) nldif_borroso (15 iteraciones); (d) Imagen ruidosa (var 0.04); (e) anisotropo_doble (80 iteraciones); (f) nldif_borroso (12 iteraciones)



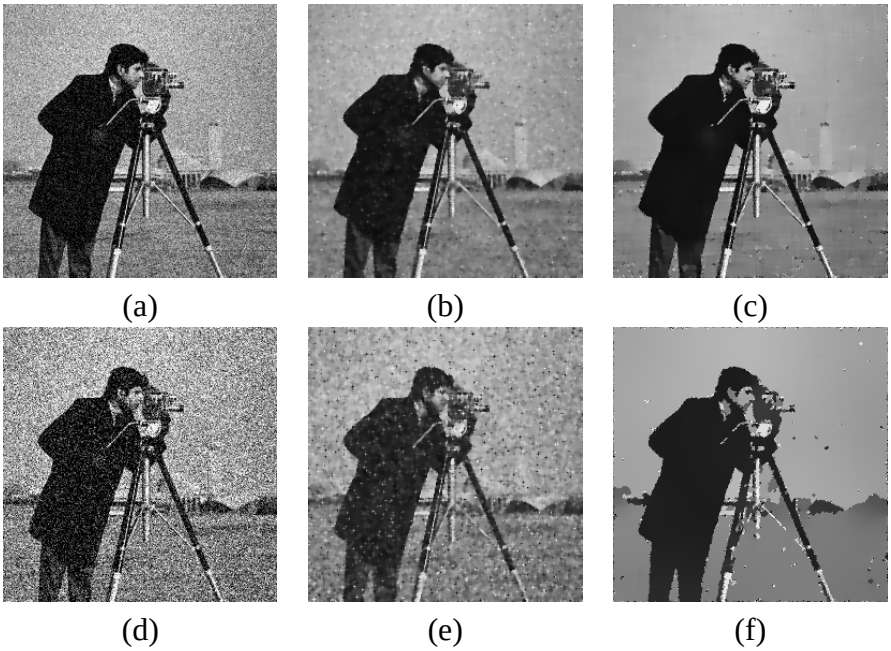
(a) (b) (c)

(d) (e) (f)

Figura 7.73: (a) Imagen ruidosa (var 0.01); (b) anisotropo_doble (60 iteraciones); (c) nldif_borroso (4 iteraciones); (d) Imagen ruidosa (var 0.04); (e) anisotropo_doble (60 iteraciones) ; (f) nldif_borroso (10 iteraciones)

Imágenes afectadas por ruido gaussiano

Se puede observar que al igual que sucedía para el para el caso de ruido *speckle*, también el filtro nldif_borroso consigue mejores resultados.



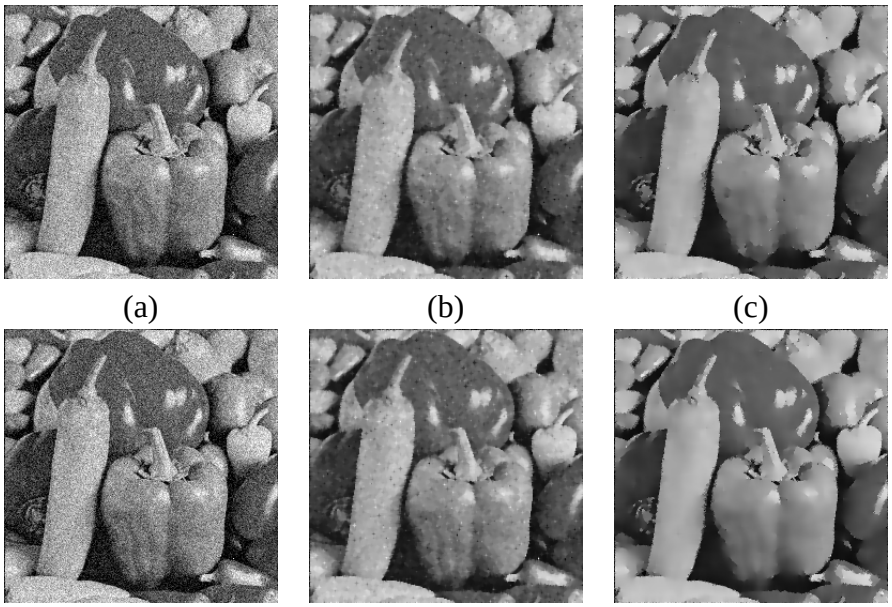
(a) (b) (c)

(d) (e) (f)

Figura 7.74: (a) Imagen ruidosa (var 0.01); (b) anisotropo_doble (60 iteraciones ; (c) nldif_borroso (5 iteraciones); (d) Imagen ruidosa (var 0.04); (e) anisotropo_doble (80 iteraciones); (f) nldif_borroso (50 iteraciones)

Filtro— Ruido ***gaussiano***	0,01	0,04
Anisotropo_doble	0,66	0,66
Nldif_borroso	0,80	0,77

Cuadro 7.13: Casa. Ruido *speckle*



(a) (b) (c)

(d) (e) (f)

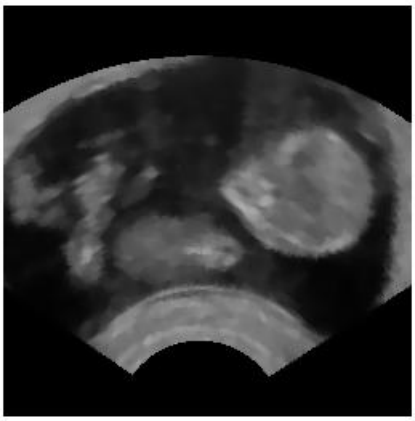
Figura 7.75: (a) Imagen ruidosa (var 0.01); (b) anisotropo_doble (60 iteraciones); (c) nldif_borroso (10 iteraciones); (d) Imagen ruidosa (var 0.04); (e) anisotropo_doble (60 iteraciones) ; (f) nldif_borroso (20 iteraciones)

Filtro— Ruido ***gaussiano***	0,01	0,04
Anisotropo_doble	0,72	0,72
Nldif_borroso	0,80	0,80

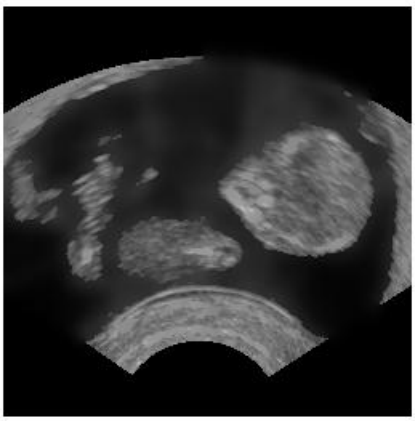
Cuadro 7.14: Pimiento. Ruido *gaussiano*

7.4.2. Ecografías: imágenes reales

También se hará una comparativa para el caso de imágenes reales. En este caso compararemos primero los efectos sobre una sección del feto y del riñón y posteriormente sobre un volumen total, concretamente el del feto. Los filtros a comparar serán de nuevo el filtro anisotropo_doble por parte del *toolbox anisótropo,* y el filtro nldif_borroso5_contador perteneciente al *toolbox difusión.*

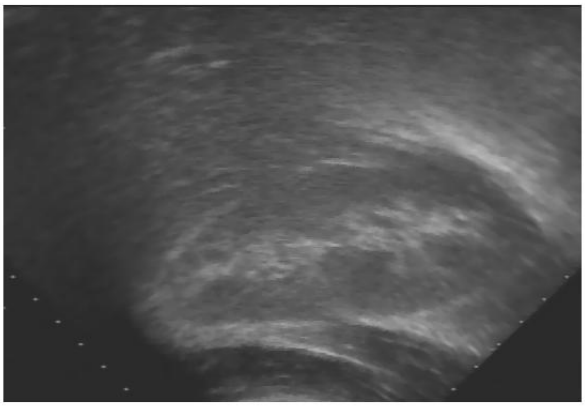


(a)

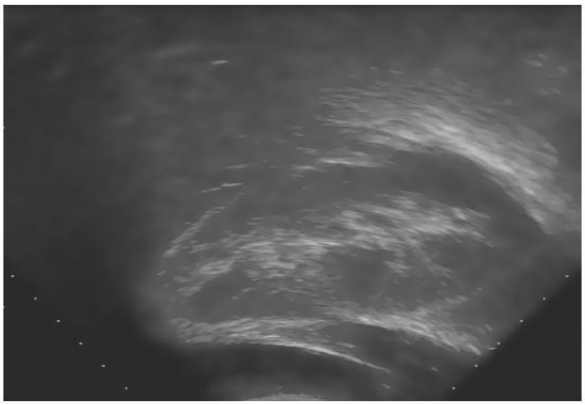


(b)

Figura 7.76: (a) Sección filtrada con anisotropo_doble; (b) Imagen filtrada con nldif_ borroso5_cont

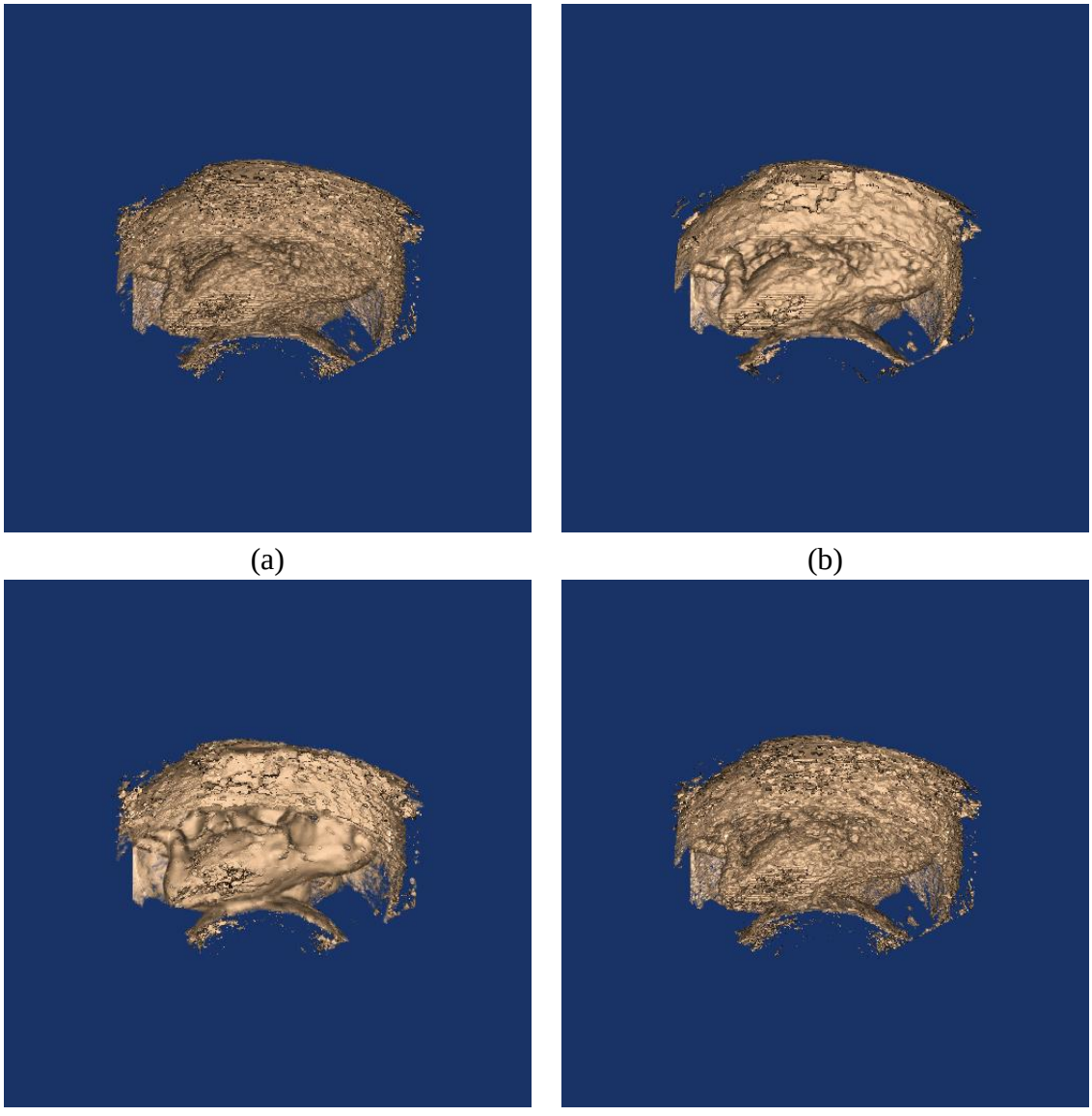


(a)



(b)

Figura 7.77: (a) Sección filtrada con anisotropo_doble; (b) Imagen filtrada con nldi_ borroso5_cont



(a) (b)

(c) (d)

Figura 7.78: (a) Volumen del feto original. Resultado de aplicar el filtro: (b) anisotropo_doble (60 iteraciones); (c) nlfid_borroso5_cont (3 conjuntos borrosos para la detección de bordes); (d) nlfid_borroso5_cont (4 conjuntos borrosos para la detección de bordes)

Filtro	Ecografías		Sintéticas speckle		Sintéticas gaussiano	
	Conservar Bordes	**Eliminar Ruido**	**Conservar Bordes**	**Eliminar Ruido**	**Conservar Bordes**	**Eliminar Ruido**
anisotropo_log	Mal	Regular	Mal	Bien	Mal	Mal
anisotropo_doble	Regular	Regular	Bien	Bien	Bien	Mal

Cuadro 7.15: *Toolbox Anisótropo*

Filtro	Ecografías		Sintéticas speckle		Sintéticas gaussiano	
	Conservar Bordes	**Eliminar Ruido**	**Conservar Bordes**	**Eliminar Ruido**	**Conservar Bordes**	**Eliminar Ruido**
nldif_borroso	Regular	Regular	Regular	Bien	Regular	Regular
nldif_borroso3	Bien	Bien	Regular	Bien	Regular	Mal
nldif_borroso4	Bien	Bien	Regular	Regular	Mal	Mal
nldif_borroso5	Bien	Bien	Regular	Bien	Regular	Mal
nldif_total	Mal	Mal	Mal	Mal	Regular	Mal
nldif_borroso5_cont	Bien	Bien	Mal	Regular	Regular	Mal

Cuadro 7.16: *Toolbox Difusión*

Capítulo 8

Conclusiones y líneas futuras

8.1. Conclusiones

Las imágenes médicas suponen una herramienta de diagnóstico muy importante en el campo de la medicina. Pero estas imágenes captadas a través de las diferentes técnicas vistas en el capítulo 2 no siempre tienen la calidad que uno desea, por este motivo, hay que buscar técnicas de procesado de señal efectivas que permiten mejorar su calidad. El trabajo realizado en este proyecto se ha centrado en el diseño e implementación de un conjunto de métodos borrosos de realce que preparan a las imágenes para un tratamiento posterior como puede ser una segmentación de estructuras. Cada uno de estos métodos opera mejor sobre un tipo de imágenes que sobre otras, es decir, no todos los métodos están indicados para el mismo tipo de realce. Aunque en este proyecto se ha hecho un análisis de los diversos métodos presentados para imágenes captadas por ultrasonidos, los filtros pueden ser perfectamente aplicados a otros tipos de imagen. Al estar integrados en un mismo entorno, y convenientemente documentados en un manual de usuario, el acceso y análisis de cada uno de ellos para ser utilizados en un problema concreto será bastante sencillo.

Cada tipo de imagen es degradada por diferentes factores, por lo tanto cada una de ellas requerirá un procesado diferente. El proyecto se centra en el estudio de imágenes médicas procedentes de ultrasonidos, las cuales se ven afectadas por la presencia de ruido *speckle*. Este ruido dificulta el análisis de la información para la mayoría de las aplicaciones, por lo tanto con los filtros diseñados se pretende bien la eliminación del *speckle* (*Toolbox Anisótropo y Toolbox Difusión*) o bien una correcta detección de bordes asumiendo la existencia de este ruido en la imagen (*Toolbox HPFborroso*).

Uno de los principales requisitos para una correcta segmentación es disponer de un buen detector de bordes. Por lo tanto en el *Toolbox HPFborroso* se ha implementado un detector de bordes indicado para este tipo de imágenes médicas. De los resultados recogidos en el capítulo 7, se puede decir que el comportamiento de este filtro es bueno para las imágenes para las que ha sido creado, es decir, ecografías. Así tanto para el caso del

feto como del riñón analizados, se consiguen mejores resultados cuando se detectan los bordes con los operadores borrosos que con las versiones clásicas. En un principio no se observa gran diferencia entre utilizar *Prewitt borroso o Sobel borroso*, sin embargo, si la hay respecto al operador borroso *Sobel2*. Así, mientras que en los primeros los contornos quedan muy bien delimitados, con el segundo aparte de delimitar los contornos se consiguen marcar también aquellas transiciones más acentuadas que pertenecen al interior de los volúmenes. Sin embargo cuando se intenta aplicar a imágenes sintéticas los resultados que se consiguen son peores que los que se obtienen con los detectores clásicos (*Prewitt o Sobel*), en especial cuando se trata de ruido *gaussiano*, pues aunque se consigue detectar gran número de transiciones (detalle muy fino), la calidad de los bordes detectados es mala.

Como ya se ha comentado, otro de los objetivos es la eliminación del ruido *speckle*, para lo cual se han diseñado los filtros contenidos en los *Toolboxes Anisótropo y Difusión*. A pesar de realizar con todos ellos un filtrado de difusión anisotrópico, los resultados conseguidos con cada uno son muy diferentes. Con los filtros contenidos en el *Toolbox Anisótropo* se consiguen resultados interesantes para el tratamiento de imágenes sintéticas contaminadas con ruido *speckle* pero no con ruido *gaussiano* pues en este caso aparece en cierta medida ruido *sal y pimienta*. Además hay que añadir que el filtro *anisotropo_log* difumina un poco los bordes, mientras que en el caso de *anisotropo_doble* al tratarse de condiciones más restrictivas, se comporta mejor en estos. Tampoco para el tratamiento de ecografías los resultados que se obtienen son satisfactorios.

Sin embargo con el *Toolbox Difusión*, dado la gran variedad de filtros que en él se implementan, se consiguen resultados muy dispares. En el caso de imágenes ecográficas reales, los resultados son muy buenos con los filtros nldif_borroso3, nldif_borroso4 y nldif_borroso5, pues eliminan bien el ruido *speckle* y conservan bien los bordes. Sin embargo para el filtro nldif_total o la versión borrosa nldif_borroso, los resultados ya no son tan satisfactorios, pues aunque se consigue eliminar el ruido *speckle* ya no se respetan los bordes, por lo que la imagen aparece difuminada. Respecto a imágenes sintéticas, decir que para el caso de ruido *gaussiano* los resultados conseguidos con los nuevos filtros son peores. Sin embargo en el caso de sintéticas con ruido *speckle* los resultados son muy buenos para imágenes con poco ruido (del mismo orden que el que presentan las ecografías) y para el caso de imágenes muy sucias ya no son tan buenos, pues aunque eliminan mucho ruido no conservan bien los bordes.

Otra necesidad que se ha de tener en cuenta a la hora de implementar un método de realce es la carga computacional necesaria para su ejecución. La ventaja que ofrecen los *toolboxes* implementados, es el bajo tiempo de ejecución necesario para filtrar una imagen con cualquiera de los métodos. Esto se ha conseguido gracias al API *mex* de matlab. Por la experiencia adquirida durante la realización de este trabajo, la utilización de esta interfaz (en vez de utilizar el lenguaje propio de *matlab*) cuando los algoritmos a implementar llevan una carga elevada de bucles mejora extremadamente su tiempo de ejecución. Sería recomendable que cualquier trabajo futuro que tratase temas relacionados con el tratamiento de imágenes médicas usase esta interfaz debido a su sencillez y mejora

de prestaciones.

Resumiendo, decir que los métodos diseñados e implementados en este proyecto presentan un buen comportamiento para el tratamiento de imágenes afectadas por ruido *speckle*. Así dada una ecografía, se consigue tanto detectar sus bordes como eliminar el ruido *speckle* de modo satisfactorio en un tiempo de ejecución bajo.

8.1.1. Líneas futuras de investigación

La metodología presentada en este trabajo puede ser, por supuesto, extendida y generalizada. El capítulo 6 se estructuraba en tres secciones correspondientes a los diferentes *toolboxes* implementados. En cada una de estas secciones se incluía un apartado denominado, *Posibles mejoras*, en el que se especificaban diferentes alternativas que podrían suponer mejores interesantes en cada *toolbox*. Dado que éstas ya han sido analizadas minuciosamente, ahora simplemente se van a recordar las más generales e interesantes. Destacamos:

- El lenguaje de programación.

  A pesar de la potencia que ofrece la interfaz API *mex* de *matlab* sería conveniente realizar la implementación de los métodos tratados en un lenguaje de programación diferente sobre todo para poder usar los filtros en entornos de procesado de imagen que tengan potencia en este campo. Concretamente, sería interesante utilizar y completar así las herramientas que posee.

- Extensión a 3D.

  Se ha visto en el capítulo 7 que el funcionamiento de los filtros con datos volumétricos tiene un comportamiento bastante aceptable. Pero para el procesado de datos volumétricos se han de seccionar éstos de alguna forma, se han de procesar cada una de estas secciones y posteriormente éstas han de ser renderizadas. Evidentemente en todos estos procesos se está perdiendo algún tipo de información. Entonces parece razonable realizar una extensión a 3D de los métodos de realce implementados para poder aprovechar toda la información que llevan los datos volumétricos ya que al realizar el procesado de cada una de las secciones de manera independiente no se aprovechan posibles relaciones que haya entre secciones consecutivas.

- Análisis del histograma.

  Para mejorar los resultados del filtrado borroso, y hacer que el sistema de inferencia se adecue a la imagen que se va a procesar, se podría pensar en hacer uso del histograma de dicha imagen, ya que este permite ver las diferencias más significativas entre los niveles de gris existentes en la imagen. Y por tanto en función de éste se podrían definir los conjuntos borrosos que representarían a la variable lingüística de entrada, centrándolos en aquellas diferencias más significativas y definiendo su

anchura según la precisión del control que se quiera tener sobre las distintas diferencias. Y en función de los valores que se consideren significativos, se pueden obtener los centroides que se van a usar.

- Integración con otros métodos de realce.

  Finalmente, cualquiera de los métodos desarrollados podrían integrarse dentro de una aplicación dedicada a la segmentación o, en general, a cualquier aplicación de procesado de imagen. Sería el primer paso a realizar dentro de la aplicación y prepararían la imagen para que cualquier procesado posterior fuese mejor realizado. La integración de estos métodos dentro de cualquier aplicación de *matlab* sería muy sencilla y apenas modificaría el tiempo necesario para la ejecución de la aplicación.

Bibliografía

[Acharya98] R. Acharya y R. Wassermanand J. Stevens y C. Hinojos, “Biomedical imaging modalities: an overview”, Biomedical Imaging Group, Department of Electrical and Computer Engineering, State University of New York at Buffalo, 1998.

[Aja00] S. Aja, “Algoritmo Fuzzy de Difusión Anisótropa para Eliminación de Speckle”. *Informe técnico*. ETSI Telecomunicación. Valladolid, Octubre 2000.

[Aja01] S. Aja, C. Alberola y J. Ruiz, “Fuzzy Anisotropic Diffusion for Speckle Filtering”, *Proceedings of the ICASSP 2001*, Salt Lake City, Mayo 2001.

[Alberola01] C. Alberola y S. Aja, “Fuzzy Image Procesing. Estado del Arte y bibliografía”. *Informe técnico*. ETSI Telecomunicación. Valladolid, Febrero 2001.

[Apiguide] *MATLAB Application Program Interface Guide*

[Apiref] *MATLAB Application Program Interface Reference*

[Atlas94] , R. A. Bowerman, “Atlas de anatomía ecográfica del feto normal”, Mosby, División de Times Mirror de España, S. A., 1994.

[Baxes94] G. A. Baxes, “Digital image processing: principles and applications”, Ed. John Wiley Sons, Inc, 1994.

[Beutel00] J. Beutel, H. L. Kundel y R. L. Van Metter, *Handbook of medical imaging*, Vol 1, Physics and Psychophysics, SPIE Press, Bellingham, Washington, 2000.

[Bronzino95] J. D. Bronzino, *The biomedical engineering. Handbook*, CRC Press/IEEE Press, Boca Raton, Florida, USA, 1995.

[Catte92] F. Catté, P. L. Lions, J. M. Morel, y T. Coll “Image selective smoothing and edge detection by nonlinear diffusion”, *SIAM J. Numer. Anal.*, Vol. 29, pp. 182-193, 1992.

[Chan97] K. C. C Chan, V. Lee y H. Leung, “Radar Tracking for Air Surveillance in a Stressful Environment using a Fuzzy-Gain-Filter”, *IEEE Transactions on fuzzy systems*, Vol. 13, No. 4, Abril 2004.

[Choi97] Y. Choi y R. Krishnapuram, "A Robust Approach to Image Enhancement Based on Fuzzy Logic", *IEEE Transactions on Image Processing*, Vol. 6, No. 6, Junio 1997.

[Cho93] Z. Cho, J. P. Jones y M. Singh, "Foundations of medical imaging", Ed. John Wiley Sons, Inc, 1993.

[De98] T. K. De y B. N. Chatterji, "An approach to a generalized technique for image contrast enhancement using the concept of fuzzy set", *Fuzzy Sets and Systems*, Vol. 25, pp. 145-158, 1998.

[Fang93] Fang N. y Cheng M. C, "An automatic crossover point selection technique for image enhancement using fuzzy sets", *Pattern Recognition Letters,* Vol. 14, pp.397-406, 1993.

[Farbiz00] F. Farbiz, M. B. Menhaj, S. A. Motamedi y M. T. Hagan, "A New Fuzzy Logic Filter for Image Enhancement", *IEEE Transactions on Systems, Man and Cybernetics-Parte B: Cybernetics*, Vol. 30, No. 1, Febrero 2000.

[Gerig] G. Gerig, O. Kübler, R. Kikinis y F. A. Jolesz,"Nonlinear Anisotropic Filtering of MRI Data", *IEEE Trans. on Medical Imaging*, Vol. 11, nº 2, pp. 221-224, Junio 1992.

[Gonzalez92] R. C. González y R. E. Woods, *Digital Image Processing*, USA: Addison-Wesley Publishing, 1992.

[Kerre00] E. E. Kerre y M. Nachtegael (Editores), "Fuzzy Techniques in Image Processing", *Studies in fuzziness and soft computing*, Vol. 52, Physica-Verlag, pp. 137-221, Heidelberg, 2000.

[Klir95] G. Klir y B. Yuang, "Fuzzy Sets and Fuzzy Logic", *Prentice-Hall International*, New Jersey, 1995.

[Koenderink84] J. Koenderink, "The Structure of Images", *Biological Cybernetics*, Vol. 50, pp. 363-370, 1984.

[Kosko] B. Kosko, "Fuzzy Engineering", *Prentice-Hall International*, New Jersey, 1997.

[Law96] T. Law, H. Itoh y H. Seki, "Image Filtering, Edge Detection, and Edge Tracing Using Fuzzy Reasoning", *IEEE Trans. Pattern Anal. Mach Intell.*, Vol. 18, No. 5, pp. 481-491, Mayo 1996.

[Mendel95] J. M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial", *Proceedings of the IEEE*, Vol. 83, No. 3, Marzo 1995.

[Pajares03] G. Pajares, J. M. Cruz, J. M. Molina, J. Cuadrado y A. Lòpez, "Imágenes Digitales: procesamiento práctico con Java", Dpto. Arquitectura de Computadores y Automática, Facultad de CC. Físicas, Universidad Complutense de Madrid, 2003.

[Pal81] S. K. Pal y R. A. King, "Image Enhancement Using Smoothing with Fuzzy Sets", *IEEE Transactions On Systems, Man, and Cybernetics*, Vol. SMC-11, No. 7, Julio 1981.

[Pratt91] W. K. Pratt, *Digital Image Processing*, A Wiley-Interscience publication, Segunda Edición, Wiley, 1991.

[Perona90] P. Perona y J. Malik, "Scale-Space and Edge Detection using Anisotropic Diffusion", *IEEE Pattern Anal. Machine Intell.*, Vol. 12, pp. 629-639, Julio 1990.

[Reyero95] R. Reyero, C. F. Nicolás, *Sistemas de control basados en lógica borrosa: fuzzy control*, Omron electronics S.A, Centro de Investigaciones Tecnológicas IKERLAN, 1995.

[Russo95a] F. Russo y G. Ramponi, "A Fuzzy Operator for the Enhancement of Blurred and Noisy Images", *IEEE Transactions On Image Processing*, Vol. 4, no. 8, pp. 1169-1174, Agosto 1995.

[Russo95b] F. Russo y G. Ramponi, "Adaptive Image Smoothing Using Fuzzy Operators", *IEEE Workshop on Nonlinear Signal and Image Proccesing*, pp. 305-308, Junio 1995.

[Russo96a] F. Russo y G. Ramponi, "Removal of Impulse Noise Using a FIRE Filter", *IEEE Int. Conf. on Image Processing, ICIP-96*, Septiembre 1996.

[Russo96c] F. Russo y G. Ramponi, "A Fuzzy Filter for Images Corrupted by Impulse Noise", *IEEE Signal Processing Letters*, Vol. 3, No. 6, Junio 1996.

[Russo98] F. Russo, "Recent Advances in Fuzzy Techniques for Image Enhancement", *IEEE Transactions on instrumentation and measurement*, Vol. 47, No. 6, Diciembre 1998.

[Sakas95] G. Sakas, L. Schreyer y M. Grimm, "Preprocessing and Volume rendering for 3D ultrasound data", *IEEE Computer Graphics and Applications*, pp. 47-54, Julio 1995.

[Schroeder00] W. Schroeder, K. Martin, L. Avila y C. Law, *The VTK user's guide*, Kitware Inc., Mayo 2000.

[Seeram94] Seeram, "Computed Tomography: Physical principles, clinical applications and quality control", Ed. W.B. Saunders Company, 1994.

[Sugeno74] M. Sugeno, *Theory of Fuzzy Integrals and Its Applications*, Dissertation, Tokyo Institute of Technology, Japón 1974.

[Szczepaniak00] P. S. Szczepaniak, P. J. G. Lisboa y J. Kacprzyk (Editores),"Fuzzy Systems in Medicine", *Studies in fuzziness and soft computing*, Vol. 41, Physica-Verlag, pp. 281-335, Heidelberg, 2000.

[Taguchi96c] A. Taguchi y N. Izawa, "Fuzzy Center Weighted Median Filters", *EUSIPCO'96*, pp. 1721-1724, Trieste, Italia, Septiembre 1996.

[Tizhoosh97a] H. R. Tizhoosh, G. Krell y B. Michaelis, "On Fuzzy Enhancement of MegaVoltage Images in Radiation Therapy", *IEEE Conference on Fuzzy Systems*, Vol.3, pp.1399-1404, FUZZ-IEEE'97, Barcelona, España.

[Tizhoosh97b] H. R. Tizhoosh , *Fuzzy Image Processing*, Springer, Heidelberg, 1997.

[Tizhoosh99c] H. R. Tizhoosh y H. Hauβecker , "Fuzzy Image Processing: An Overview", *Handbook on computer vision and applications*, Vol. 2, pp. 683-727, Academic Press, Boston, 1999.

[Trillas92] E. Trillas y J. Gutiérrez Ríos (Editores), *Aplicaciones de la lógica borrosa*, Consejo Superior de Invertigaciones Científicas, Madrid, 1992.

[Wang04] Z. Wang, A. C. Bovik, H. R. Sheikh y E. P. Simoncelli, "Image Quality Assessment From Error Visibility to Struct Similirity", *IEEE Transactions on image processing*, Vol. 13, No. 4, Abril 2004.

[Witkin83] A. Witkin, "Scale-Space Filtering", *Int'l Joint Conf. Artificial Intelligence*, pp. 1019-1021, 1983.

[Weickert98] J. Weickert, B. M. ter H. Romeny y M. A. Viergever, "Efficient and Reliable Schemes for Nonlinear Diffusion Filtering", *IEEE Transactions on image processing*, Vol. 7, No. 3, Marzo 1998.

[Zadeh65] L. A. Zadeh, "Fuzzy sets", *Information and Control*, Vol. 8, pp. 338-353, 1965.

Apéndice A

Manual de usuario

En este apéndice se muestra la documentación creada para facilitar el uso de los filtros implementados en los diferentes *toolboxes*. Para cada uno de estos filtros se muestra el propósito que tienen, la sintáxis que debe ser utilizada, una descripción de las opciones que soporta cada uno de ellos, el tipo de datos que han de ser los argumentos de entrada y salida y un ejemplo de utilización.

A.1. HPFborroso

Propósito

Detectar bordes en imágenes afectadas por ruido *speckle* mediante un filtrado paso alto de naturaleza borrosa.

Sintáxis

K=HPFborroso(I, tipoFiltrado);
K=HPFborroso(I, tipoFiltrado, opcionEscala, N);

Descripción

Se puede elegir realizar el filtrado según los operadores Prewitt y Sobel, existiendo del segundo dos versiones diferentes. Además existe la posibilidad de modificar los conjuntos borrosos, lo cual se debe indicar a través de la línea de comandos.

- **tipoFiltrado:** Indica cual es el tipo de filtro a aplicar. Puede tomar los siguientes valores: *'prewitt'*, *'sobel'* y *'sobel2'*.
- **opcionEscala:** Permite modificar los conjuntos borrosos. Se puede elegir entre: *'escala'* (se modifican los conjuntos) o *'noeescalo'* (no se pueden variar). Por defecto se emplea la opción de no modificar los conjuntos.
- **N:** En el caso de decidir modificar los conjuntos es necesario indicar cuales serán los nuevos conjuntos. Esto se consigue gracias a este factor. Si no se le da ningún valor, toma el valor uno por defecto.

Tipos de datos

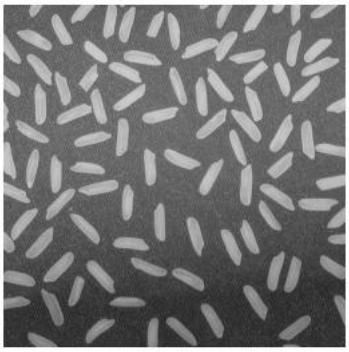
I debe ser una matriz en formato doble, *N* ha de ser un número real y *tipoFiltrado* y *opcionEscala* ha de se una de las cadenas de caracteres antes indicadas.

Notas

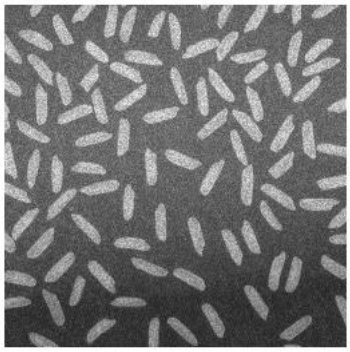
Este filtro se trata de un filtro paso alto, que implementa los operadores de vecindad básicos siguiendo la filosofía del algoritmo IFCF.

Ejemplo

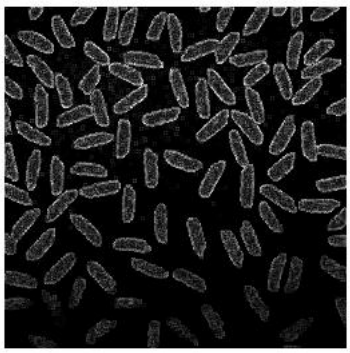
```
    I=imread(rice.tif);
J=imnoise(I, speckle, 0.01);
K=HPFborroso(double(J), sobel, escalo, 4 );
imshow(I);
figure,imshow(J);
figure,imshow(uint8(K));
```



(a)



(b)



(c)

Figura A.1: (a) I: Imagen limpia; (b) J: Imagen afectada por ruido speckle; (c) K: Imagen filtrada.

A.2. Toolbox-Anisótropo

Propósito

Realizar un filtrado anisótropo de naturaleza borrosa, que presente un buen comportamiento ante imágenes afectadas por ruido *speckle*. Los objetivos que se pretenden alcanzar con esta implementación borrosa, son los de reducir el número de iteraciones tratando de reducir el tiempo de procesado, mejorar los resultados obtenidos con la lógica clásica y permitir tener un mayor control sobre el proceso de difusión, evitando así llegar a resultados no deseados.

Sintáxis

Este *toolbox* contiene dos filtros diferentes: *anisotropo_log* y *anisotropo_doble*. La sintáxis para cada uno de ellos será:
K=anisotropo_log(I,num)
K=anisotropo_doble(I,num)

Descripción

Como ya se ha comentado, en este *toolbox* se han implementado dos filtros diferentes. La diferencia entre ellos está en el modo de calcular el coeficiente de difusión, pues cada filtro utiliza reglas diferentes.

Tipos de datos

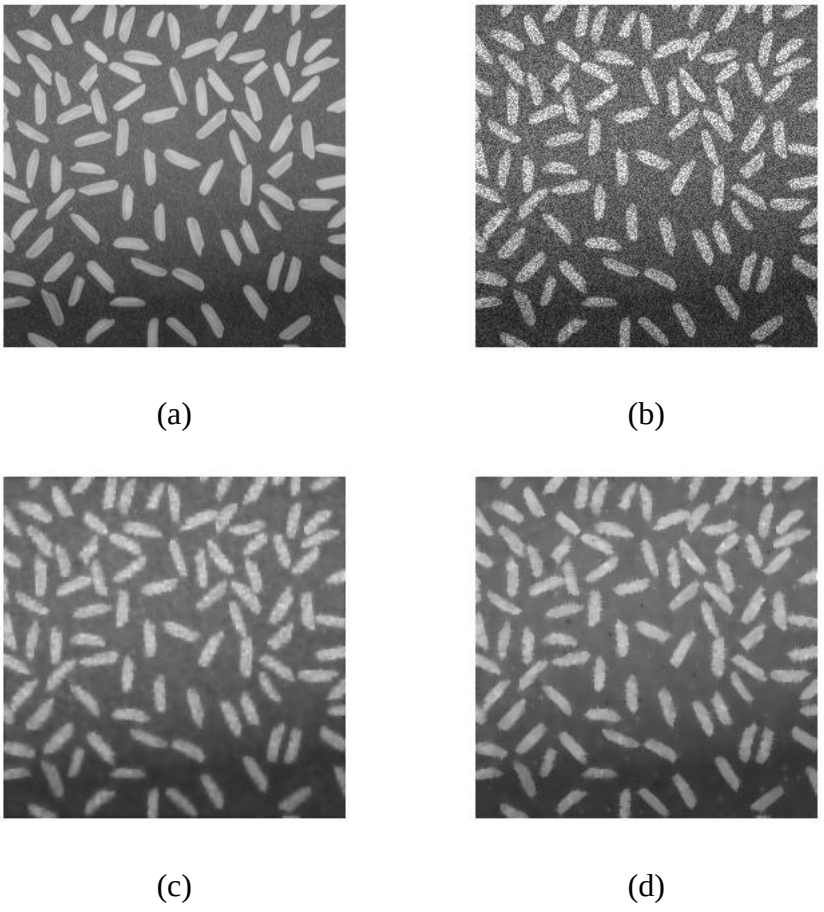
I debe ser una matriz en formato doble y *num* ha de ser un número real pues es el número de iteraciones.

Notas

Este filtro se trata de un filtro anisótropo. El algoritmo que se va a utilizar para el filtrado de las imágenes ya está definido: el algoritmo de difusión anisótropa propuesto por [Gerig]. Y es sobre éste donde se va a tratar de aplicar la teoría de lógica borrosa, siguiendo el principio de funcionamiento de los filtros *IFCF*.

Ejemplos

```
    I=imread(rice.tif);
J=imnoise(I, speckle, 0.04);
K=anisotropo_log(double(J),30);
K2=anisotropo_doble(double(J),80);
imshow(I);
figure,imshow(J);
figure,imshow(uint8(K));
figure,imshow(uint8(K2));
```



(a) (b)

(c) (d)

Figura A.2: (a) I: Imagen limpia; (b) J: Imagen afectada por ruido speckle; (c) K: Imagen filtrada (anisotropo_log) 30 iteraciones; (d) K2: Imagen filtrada (anisotropo_doble) 80 iteraciones.

A.3. Toolbox-Difusión

Propósito

Realizar un filtrado anisótropo de naturaleza borrosa que presente un buen comportamiento ante imágenes afectadas por ruido *speckle*, a partir de la regularización espacial del filtro de Perona-Malik realizada por Caté te al. Los objetivos que se pretenden alcanzar con esta implementación borrosa son mejorar los resultados obtenidos con la lógica clásica y permitir tener un mayor control sobre el proceso de difusión, evitando así llegar a resultados no deseados. Además se implementará un controlador borroso que permita determinar ciertos parámetros como son λ y el número de iteraciones.

Sintáxis

Este *Toolbox* contiene los filtros diferentes: *nldif_borroso, nldif_borroso3, nldif_borroso4, nldif_borroso5, nldif_borroso5_cont* y *nldif_total*. La sintáxis para cada uno de ellos será:
$[y, ssim] = nldif\_borroso(I, lambda, m, stepsize, steps, imagenoriginal, verbose, drawstep)$
$[y, ssim] = nldif\_borroso3(I, lambda, m, stepsize, steps, imagenoriginal, verbose, drawstep)$
$[y, ssim] = nldif\_borroso4(I, lambda, sigma, m, stepsize, steps, imagenoriginal, verbose, drawstep)$
$[y, ssim] = nldif\_borroso5(I, lambda, m, stepsize, steps, imagenoriginal, verbose, drawstep)$
$[y, ssim] = nldif\_borroso5\_cont(I, lambda, m, stepsize, steps, imagenoriginal, verbose, drawstep)$
$[y, ssim] = nldif\_total(I, sigma, m, stepsize, steps, imagenoriginal, verbose, drawstep)$

Además en cada uno de ellos se pueden elegir las siguientes opciones:
$[y, ssim] = nldif*(..., 'grad', 'flux')$ plotea también el flujo y el gradiente de la imagen.
$[y, ssim] = nldif*(..., 'imscale')$ usa el comando 'imagesc' en lugar de 'image' para plotear la difusión.
$[y, ssim] = nldif*(..., 'dfstep', n)$ solo recalcula la difusividad después de n pasos, con lo que se incrementa la velocidad.
$[y, ssim] = nldif*(..., 'aos')$ usa el esquema AOS para actualizar la imagen. Permite que el escalón de tiempo t sea arbitrariamente grande ($t < \infty$)

Descripción

Anteriormente se han nombrado los filtros implementados en este *toolbox* y la sintáxis necesaria para ejecutar cada uno de ellos. A continuación se explica cual es la diferencia entre ellos.

- nldif_borroso

  La característica que lo diferencia está en el suavizado de la imagen inicial. En este filtro se suaviza con un filtrado borroso paso bajo, el filtro *SSFCF*.

- nldif_borroso4

  La imagen se suaviza mediante la convolución con una gaussiana y posteriormente se calcula la difusividad. Para calcular ésta se hace uso de la lógica borrosa. Las modificaciones ahora se introducen en el cálculo del gradiente, pues en lugar de calcular el gradiente del modo convencional, se calcula la versión paso alto de la imagen a la entrada según el filtro HPFborroso (opción Sobel2).

- nldif_borroso5

  Igual que el filtro anterior, solo que ahora como el filtro HPFborroso funciona bien ante el ruido *speckle*, se probará a calcular la difusividad sin suavizar previamente la imagen.

- nldif_borroso3

  Se corresponde con la composición de los filtros nldif_borroso y nldif_borroso4, es decir, utilizará la lógica borrosa tanto para el suavizado de la imagen (SSFCF) como para el cálculo del gradiente (HPFborroso).

- nldif_borroso5_cont

  Al filtro nldif_borroso5 se le añade un controlador borroso que permite determinar ciertos parámetros como son λ y el número de iteraciones.

- nldif_total

  Se corresponde con la versión clásica de la regularización espacial del filtro de Perona-Malik realizada por Caté et al. a la que se la ha incorporado un controlador borroso que determina el parámetro λ y el número de iteraciones.

Tipos de datos

- I: debe ser una matriz en formato doble.

- lambda: parámetro de contraste, un número. Para lambda variable en el tiempo, lambda ha de ser un vector de tamaño el número de pasos.

- sigma: desviación típica estándar de la gaussiana con la que será convolucionada la imagen antes del que se calcule el gradiente. Para sigma variable en el tiempo, sigma ha de ser un vector de tamaño el número de pasos.

- m: define la velocidad de variación de difusividad y de flujo, para una variación en el gradiente. Valores grandes de 'm' hacen que el flujo varíe rápidamente. 'm' ha de ser mayor que 1. Una buena elección sería $8 < m < 16$.

- stepsize: puede ser un escalar o un vector para un 'stepsize' variable. En este último caso, el tamaño del vector has de ser igual al número de pasos del algoritmo. Para

conseguir un sistema estable, el stepsize ha de ser menor que 0,25. En caso de usar *AOS* el 'spetsize' puede ser cualquier número positivo.

- steps: indica el número de iteraciones que ha de ejecutarse el filtro.
- imagen original: matriz en formato doble que contenga a la imagen sin degradar (se utilizará para hacer pruebas con imágenes sintéticas).
- verbose: número entero positivo que indica el número de la figura donde la imagen salida (y) será ploteada.
- drawstep: indica el número de iteraciones que han de pasar para refrescar la imagen ploteada.
- n: numero de iteraciones que han de pasar para que se recalcule la difusividad (solamente en el caso de utilizar la opción 'dfstep').

Nota

Se trata de un filtro de difusión anisótropo para el tratamiento de imágenes afectadas por ruido *speckle*. Sus bases fundamentales de diseño son:

- Regularización espacial de el filtro de Perona-Malik [Perona90].
- los filtros iterativos de lógica de control borroso, particularmente el filtro *SSFCF: Smoothing Fuzzy Control Filter* [Farbiz00].
- el filtro paso alto *HPFborroso* implementado en el *Toolbox HPFborroso*.

Ejemplos

```
    I=imread('rice.tif');
J=imnoise(I, 'speckle', 0.04);
K = nldif_borroso(J,9,12,10,50,I,10,4,'dfstep',4,'aos','imscale')
K2 = nldif_borroso3(J,9,12,10,50,I,10,4,'dfstep',4,'aos','imscale')
K3 = nldif_borroso4(J,9,1,12,10,50,I,10,4,'dfstep',4,'aos','imscale')
K4 = nldif_borroso5(J,9,12,10,50,I,10,4,'dfstep',4,'aos','imscale')
K5 = nldif_borroso5_cont(J,9,12,10,50,I,10,4,'dfstep',4,'aos','imscale')
K6 = nldif_total(J,1,12,10,50,I,10,4,'dfstep',4,'aos','imscale')
imshow(I)
figure,imshow(J)
figure,imshow(uint8(K)); figure,imshow(uint8(K2));
figure,imshow(uint8(K3)); figure,imshow(uint8(K4));
figure,imshow(uint8(K5)); figure,imshow(uint8(K6))
```

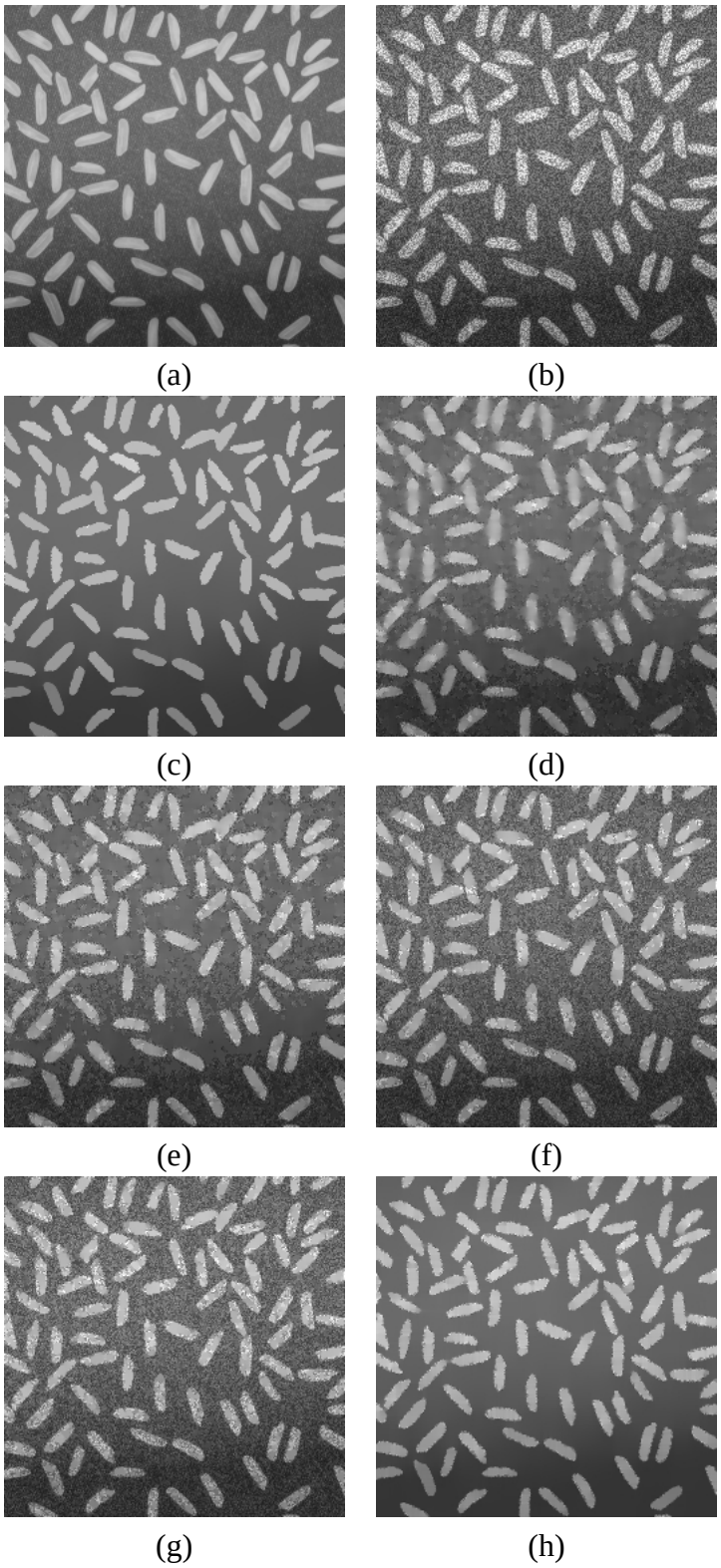


Figura A.3: (a) I: Imagen limpia; (b) J: Imagen afectada por ruido speckle; (c) K: Imagen filtrada con nldif_borroso (9 iteraciones); (d) K2: Imagen filtrada con nldif_borroso3 (10 iteraciones); (e) K3: Imagen filtrada con nldif_borroso4 (12iteraciones); (f) K4: Imagen filtrada con nldif_borroso5(50 iteraciones); (g) K5: Imagen filtrada con nldif_borroso5_cont; (h) K6: Imagen filtrada con nldif_total

Apéndice B

API *mex* de matlab

En este apéndice se describe de forma breve, las características más relevantes de la herramienta empleada para la implementación de los filtros. Aunque para la ejecución de cada uno de los ficheros realizados se ha de hacer dentro de *matlab*, la programación de los mismos ha sido en C. Se ha aprovechado el API que ofrece *matlab* para interactuar con otros programas. Es absurdo explicar de manera detallada cada uno de los ficheros realizados por eso sólo se describirán a fondo aquellos detalles de los mismos que sean más interesantes.

B.1. Introducción

Aunque *matlab* es una herramienta completa y autocontenida para programación y trabajo con datos, es útil poder interactuar con datos y programas externos a *matlab*. *Matlab* tiene una API (*Application Program Interface*) para poder soportar interfaces externas. Las funciones que son soportadas por la API incluyen:

- Llamadas a programas C o Fortram desde *matlab*.
- Importar y exportar datos hacia y desde *matlab*.
- Establecer relaciones cliente/servidor entre *matlab* y otros programas.

Se puede llamar a subrutinas C y Fortran desde *matlab* como si fuesen funciones propias de *matlab*. A estos programas C y Fortran que se pueden llamar desde *matlab* reciben el nombre de archivos **mex**. Los archivos mex son subrutinas enlazadas dinámicamente que el intérprete de *matlab* puede cargar y ejecutar automáticamente. Los archivos mex pueden tener varias aplicaciones (aunque no son apropiados para todas ellas):

- Los programas que hayan sido realizados en Fortran o C pueden ser llamados desde *matlab* sin la necesidad de reescribirlos como archivos *.m*.

Cuadro B.1: Extensiones de los archivos mex.

Plataforma	Extensión del archivo mex
Sun OS 4.x	mex4
HP 9000/series 700	mexhp7
Alpha	mexaxp
SGI	mexsg
SGI64	mexsg64
IBM RS/6000	mexrs6
Linux	mexlx
Solaris	mexsol
Windows	dll
Macintosh	mex

- Aquellos elementos que crean un cuello de botella en las computaciones de *matlab* (por ejemplo, los bucles *for*) pueden ser realizados en C o Fortran de manera eficiente.

Mientras que los archivos propios de *matlab* tienen una extensión independiente de la plataforma (*.m*), *matlab* identifica los archivos mex por extensiones específicas para cada plataforma (ver Tabla. B.1).

Los archivos mex se llaman exactamente igual que si fuesen funciones de *matlab*. Por ejemplo, un archivo mex llamado *conv2.mex* que se encuentra en la toolbox *datafun* realiza la convolución de dos dimensiones de matrices mientras que el archivo *conv2.m* sólo contiene el texto de ayuda. Si se invoca la función *conv2* desde *matlab*, el intérprete busca a lo largo de la lista de directorios. Si encuentra dos archivos con igual nombre, uno de ellos con extensión *.m* y otro con alguna de las especificadas en la Tabla. B.1, carga y ejecuta el segundo de ellos. Es decir, los archivos mex tienen preferencia sobre los propios archivos de *matlab*. Sin embargo, la documentación de ayuda siempre se lee del archivo con extensión *.m*. Basándose en esta propiedad, en la toolbox de filtros realizados se encuentran tanto archivo mex como archivos *.m*. En los primeros se encuentra el código que se ejecuta cuando se invoca al filtro, mientras que en los archivos *.m* se encuentran la documentación de ayuda.

B.2. Datos en matlab

Antes de poder programar cualquier archivo mex, se debe tener clara la forma en que *matlab* representa los tipos de datos que soporta. El lenguaje *matlab* trabaja con un sólo tipo de objetos: el *array de matab*. Todas las variables *matlab*, que incluye escalares, vectores, matrices, cadenas, estructuras, ... se almacenan como arrays de *matlab*. La

declaración `mxArray` corresponde a la estructura de datos interna que *matlab* usa para representar los arrays. La estructura `mxArray` contiene, entre otras cosas:

- El nombre de la variable *matlab*.
- Sus dimensiones.
- Su tipo.
- Si la variable es real o compleja. En el caso de que la variable contenga números complejos, el array de *matlab* incluye vectores que contienen la parte real e imaginaria.
- Si la variable es una matriz cuyos elementos son la mayoría cero.

Las matrices son un subconjunto de los arrays de *matlab*. En este caso la estructura del `mxArray` contiene dos parámetros llamados *pr* y *pi*. El primero de ellos contiene la parte real de la matriz mientras que *pi* contiene la parte imaginaria. Tanto *pr* como *pi* son arrays unidimensionales de números de doble precisión. Esta es la forma en que Fortran almacena las matrices; *matlab* usa esta convención porque originalmente fue escrito en este lenguaje.

B.3. Organización de los archivo mex

Los archivos mex se construyen, no sólo con las herramientas propias que este lenguaje proporciona, sino que se puede utilizar un conjunto de rutinas que proporciona la librería C de *matlab*.

El código fuente para un archivo mex está compuesto de dos partes principales:

- Una rutina computacional que contiene el código para ejecutar las computaciones que se quiera implementar en el archivo.
- Una rutina de puerta (*gateway routine*) que hace de interfaz entre el fichero C y *matlab* mediante el punto de entrada `mexFunction` y sus parámetros `prhs, nrhs, plhs` y `nlhs`. `prhs` es un puntero a los `mxArrays` de entrada mientras que `plhs` es un puntero a que apunta a los `mxArrays de salida`. Los parámetros `nrhs` y `prhs` indican el número de entradas y salidas respectivamente.

En la rutina de puerta, se accede a los datos existentes en la estructura `mxArray` y después éstos se manipulan dentro de la subrutina computacional C. Por ejemplo, la expresión `mxGetPr(prhs[0])` devuelve un puntero de tipo `double *` a la parte real de los datos que son apuntados mediante `prhs[0]`. Entonces ya se puede usar este puntero de

tipo `double *` como cualquier otro puntero cualquiera de este tipo de C. Después de llamar la rutina C computacional desde la rutina de puerta, se puede establecer un puntero de tipo `mxArray` a los datos que devuelve la rutina computacional. De esta forma *matlab* es capaz de reconocer la salida de la rutina computacional como la salida del archivo mex.

El diagrama presentado en la Fig. (B.1) muestra cómo se introducen las entradas en un archivo mex, que funciones ejecuta la rutina de puerta, y cómo se envía la salida a *matlab*.

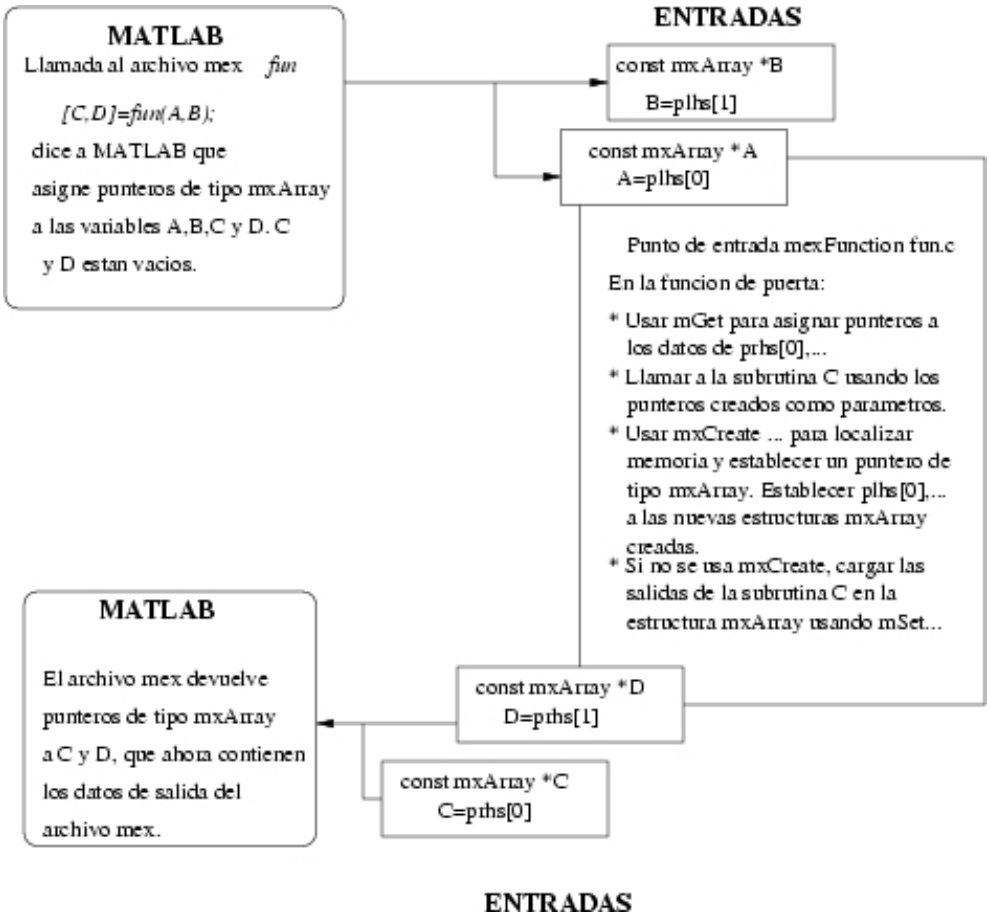


Figura B.1: Proceso de entrada y salida de los archivos mex.

Los dos componentes del archivo mex pueden separarse o combinarse. En cualquiera de los casos, los archivos deben contener el fichero de cabecera *#include "mex.h"* para que las rutinas de entrada e interfaz se declaren de manera apropiada. El nombre de la rutina de puerta debe ser siempre `mexFunction` y debe tener siempre estos parámetros:

```
void mexFunction(
int nlhs, mxArray *plhs [ ],
int nrhs, const mxArray *prhs [ ])
{
```

```
/*mas código C ... */
```

Para obtener más detalles de los presentados hasta ahora sobre los archivos mex o la forma preparar *matlab* para que sea capaz de compilar estos archivos se recomienda la lectura de los manuales de referencia [Apiguide] [Apiref].

B.4. Detalles de implementación

B.4.1. Rutina de puerta (*Gateway routine*)

En este punto se va a mostrar las funciones de la librería C de *matlab* que se han usado a lo largo de la toolbox de filtros. Para comprobar que los elementos que introduce el usuario son correctos se han usado, principalmente, las siguientes funciones:

- `mxIsComplex` determina si el `mxArray` que se está tratando tiene parte imaginaria. En caso de que el `mxArray` no contenga parte compleja, el valor que toma el parámetro `pi` es `NULL`. Sin embargo, si el `mxArray` tiene naturaleza compleja, `pi` apuntará a un arreglo de datos que contenga la parte imaginaria.

- `mxIsDouble` se encarga de determinar si el `mxArray` que se le pasa representa sus datos como números en punto flotante de doble precisión. Es importante el uso de esta función ya que los filtros realizados manejan datos que estén representados en punto flotante de doble precisión.

Además de estas funciones se han utilizado, con objetos diferentes, las siguientes:

- `mexErrMsgTxt` se utiliza para escribir un mensaje de error en la ventana de *matlab*. Después de que se imprime el mensaje de error, *matlab* termina el archivo mex y devuelve el control al *prompt* de *matlab*. Además, si la aplicación C ha utilizado las funciones `mxCalloc` o `mxCreate`, la función `mexErrMsgTxt` automáticamente libera la memoria que previamente haya sido reservada.

- `mxGetM` informa del número de filas que tiene el `mxArray` que se le pasa. Con el término ”filas”se está haciendo referencia a la primera dimensión del arreglo, no importa el número de dimensiones que éste tenga. Por ejemplo, si `array_ptr` apunta a un arreglo de cuatro dimensiones, con dimensiones $8 \times 9 \times 5 \times 3$, entonces la función `mxGetM` devuelve 8.

- `mxGetN` informa del número de columnas que tiene el `mxArray` que se le pasa. Al igual que sucede con la función anterior, çolumnas”hace referencia a la segunda dimensión del arreglo.

- `mxGetPr` determina la dirección de comienzo de la parte real del `mxArray` que se le pasa. Una vez que se tiene la dirección de comienzo, es fácil acceder a cualquier elemento del `mxArray`.

- `mxGetString` copia los caracteres que contiene un `mxArray` de tipo cadena a una cadena propia de C. Esta función devuelve un puntero `buf` que apunta al comienzo de la cadena C. Se sabe que las cadenas en C siempre terminan con el carácter NULL. En el caso de que la cadena de tipo `mxArray` tenga varias filas, éstas se copian, por columnas, en una cadena C larga.

- `mxCalloc`, `mxFree` son funciones que tienen que ver con la asignación y liberación dinámica de memoria. En los archivos mex se suele utilizar más la función `mxCalloc` que la función típica de C `calloc` para asignar dinámicamente memoria. En los archivos mex, la función `mxCalloc` realiza las siguientes funciones de forma automática.

  - Reserva espacio contiguo suficiente para para mantener *n* elementos.
  - Inicializa los *n* elementos a cero.
  - Registra este espacio reservado como si fuese una reserva propia de memoria de *matlab*.

  La parte de *matlab* encargada del manejo de memoria mantiene una lista de toda la memoria que ha sido reservada por `mxCalloc`. Además, cuando el control vuelve al *prompt* de *matlab* la parte que gestiona la memoria libera todas las parcelas que habían sido utilizadas por archivos mex.

  Por defecto, en un archivo mex `mxCalloc` genera datos no persistentes. En otras palabras, la parte de gestión de memoria de *matlab* libera la memoria tan pronto como la ejecución del archivo mex termine. Sin embargo, si se desea que la memoria reservada esté disponible cuando la ejecución de un archivo mex termina, se deben usar las funciones `mxMakeMermor` y `Persistent` después de la llamada a `mxCalloc`. En el caso de que se use memoria persistente, hay que asegurarse de usar la función `mexAllExit` para liberar esta memoria persistente cuando sea preciso.

  Por otra parte, si en algún momento es conveniente liberar memoria dentro de un archivo mex, la función que se debe usar es `mxFree`. Las tareas que realiza esta función son:

  - Llama a la función de C `free`, que libera la memoria reservada dinámicamente.
  - Elimina esta memoria de la lista que la parte de gestión de memoria de *matlab* mantiene.

A continuación se muestra la rutina de puerta de uno de los filtros realizados en el proyecto donde se puede ver en acción las funciones hasta ahora presentadas:

```
//The gateway function.
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
{

//Definiciones.
int N,L,f,c,buflen,status,fv,cv,i;
double *entrada,valores[4],*aux;
char *forma,*buf,*buf2,*tipo_filtro;

//Consideramos los siguientes parámetros por defecto.
L=256;//Niveles de gris
N=3;//Tamaño de la ventana de trabajo.
valores[0]=0;
valores[1]=10;
valores[2]=20;
valores[3]=30;

//Tratamiento de los argumentos de entrada.
if(nrhs==0) mexErrMsgTxt("\n Es necesario, como mínimo,
un argumento de entrada.\n");
if(nrhs>=1)
{
if(!mxIsNumeric(prhs[0])) mexErrMsgTxt("\nEl primer argumento
debe ser una matriz de datos dobles.\n");
   if(!mxIsDouble(prhs[0])) mexErrMsgTxt("\n El formato de la matriz
de entrada ha de ser doble.\n");
f=mxGetM(prhs[0]);
c=mxGetN(prhs[0]);
if((f==1)|(c==1)) mexErrMsgTxt("\nEl primer argumento debe ser
una matriz de datos dobles.\n");
entrada=(double*)mxGetPr(prhs[0]);
}
if(nrhs>=2)
{
if (!mxIsDouble(prhs[1])||mxIsComplex(prhs[1])||mxGetN(prhs[1])
*mxGetM(prhs[1])!=1) mexErrMsgTxt("\n El segundo argumento debe ser
un número escalar\n.");
N=mxGetScalar(prhs[1]);
if(N<=0) mexErrMsgTxt("\nN ha de ser un número entero positivo
e impar.\n");
}
if(nrhs>=3)
{
```

```
buflen=(mxGetM(prhs[2])*mxGetN(prhs[2]))+1;
        tipo_filtro=mxCalloc(buflen,sizeof(char));
status=mxGetString(prhs[2],tipo_filtro,buflen);
if (status!=0) mexErrMsgTxt("No se ha podido convertir la cadena
de datos\n.");
if(!((strcmp(tipo_filtro,"mlmf")==0)||(strcmp(tipo_filtro,"afmmf")==0)))
mexErrMsgTxt("\nLos tipos de filtro permitidos son mlmf y afmmf.\n");
}
if(nrhs==4) mexErrMsgTxt("\n Si introduces el conjunto VH has
de introducir también su forma.\n");
if(nrhs==5)
{
if(!mxIsNumeric(prhs[3])) mexErrMsgTxt("\nEl cuarto argumento
 debe ser un array de datos dobles.\n");
fv=mxGetM(prhs[3]);
cv=mxGetN(prhs[3]);
if(fv!=1) mexErrMsgTxt("\nEl cuarto argumento debe ser un
array de datos dobles.\n");
aux=(double *)mxGetPr(prhs[3]);
for(i=0;i<cv;i++) valores[i]=*(aux+i);
buflen=(mxGetM(prhs[4])*mxGetN(prhs[4]))+1;
forma=mxCalloc(buflen,sizeof(char));
status=mxGetString(prhs[4],forma,buflen);
if (status!=0) mexErrMsgTxt("No se ha podido convertir la
cadena de datos\n.");
if(!((strcmp(forma,"trimf")==0)||(strcmp(forma,"trapmf")==0)))
mexErrMsgTxt("\n El tipo de conjunto sólo puede se triangular
(trimf) o trapezoidal(trapmf).\n");
if(((strcmp(forma,"trimf")==0)&(cv!=3)))
mexErrMsgTxt("\n Si el conjunto es triangular deben
especificarse 3 parámetros.\n");
if(((strcmp(forma,"trapmf")==0)&(cv!=4)))
mexErrMsgTxt("\n Si el conjunto es trapezoidal deben
especificarse 4 parámetros.\n");
}
if(nrhs>5) mexErrMsgTxt("\n Has introducido demasiados parámetros
de entrada.\n");
if(nrhs<3) tipo_filtro="afmmf";
if(nrhs<5) forma="trapmf";

//Llamada a la rutina C
plhs[0]=afmmf(entrada,N,tipo_filtro,f,c,L,valores,forma);
```

```
}
```

B.4.2. Rutina computacional

Como es sabido la rutina computacional es la encargada de realizar las operaciones específicas que el archivo mex implementa. Sin embargo, aunque la mayor parte del código que se escribe en esta rutina es código C usual, se puede hacer uso de las funciones presentes en la librería C de *matlab*. En este proyecto se han usado principalmente las funciones que tienen que ver con la salida de datos a *matlab* y, ocasionalmente, la función que permite hacer llamada a funciones propias de matlab (archivos *.m*). A continuación se explican las funciones más utilizadas en los archivos mex realizados:

- `mxCreateDoubleMatrix` se utiliza para crear un `mxArray` de tamaño $m \times n$. `mxCreateDoubleMatrix` inicializa cada elemento de la parte real del `mxArray` (`pr`) a cero. Si se especifica en la función que el `mxArray` va a tener parte compleja, es decir, se establece el valor del *flag* `ComplexFlag` a `mxCOMPLEX`, entonces también la función `mxCreateDoubleMatrix` inicializa los valores de la parte imaginaria a cero. En caso de que el `mxArray` que se desea crear vaya a ser real, `ComplexFlag` valdrá entonces `mxREAL` esta función reserva memoria para mantener $m \times n$ elementos reales. En caso de ser complejo, la reserva de memoria se hace extensiva a otros $m \times n$ elementos.

- `mexPrintf` imprime una cadena en la pantalla y en el diario (en el caso de que el diario se esté usando). La tarea y la sintaxis que se utiliza es exactamente la misma que la típica función de C `printf`. Sin embargo es conveniente, dentro de un archivo mex, utilizar `mexPrint` en vez de `printf`.

- `mexCallMatlab` se utiliza para invocar funciones numéricas internas de *matlab*, operadores de *matlab*, archivos *.m* u otros archivos mex. Los argumentos de entrada y salida son los mismos que se necesitan en la `mexFunction`.

  Por defecto, si el nombre del comando *matlab* que se invoca es incorrecto, o se produce algún tipo de error en la invocación, *matlab* termina la ejecución del archivo mex y devuelve el control al *promt* de *matlab*. Sin embargo este comportamiento puede modificarse mediante el uso de la función `mexSetTrapFlag`.

Se puede ver que dentro de las funciones presentadas hay unas que tiene el prefijo `mx` mientras que otras empiezan por `mex`. Aquellas funciones cuyo nombre comienza por el prefijo `mx` son rutinas que manipulan los arrays de *matlab*. Si bien en C es posible manipular los `mxArrays` directamente (cosa que en Fortran no es posible), es aconsejable usar este tipo de funciones para evitar complicaciones innecesarias. Es decir, este tipo de rutinas permiten al usuario acceder y/o manipular la parte de la información del `mxArray` que se desee. Por ejemplo, `mxGetPi` devuelve un puntero a la parte imaginaria de los

datos contenidos en el array. Las únicas excepciones a esta regla general son las funciones `mxGetEps`, `mxGetNan` y `mxGetInf`, que no acceden ni manipulan el `mxArray`.

Aquellas rutinas cuyo nombre comienza por el prefijo `mex` ejecutan algún tipo de operación dentro del entorno *matlab*. Por ejemplo, como se ha comentado, la función `mexPrintf` imprime una cadena en el *prompt* de *matlab*, incluso cuando la llamada a la función se efectúa dentro de un archivo mex. Ninguna función con el prefijo `mex` accede o manipula información dentro de la estructura del `mxArray`.

Apéndice C

Desarrollo matemático del gradiente

En este apéndice se muestran los pasos a a seguir para obtener las expresiones de los gradientes requeridos en función de los píxeles de la imagen.

El *coeficiente de difusión*, según se ha visto, se expresa como una función del *gradiente* de la intensidad de la imagen:

$$c(x,y,t) = f(\|\nabla I(x,y,t)\|)$$

Y si se hacía la aproximación del *gradiente* se tenía:

$$\|\nabla I(x,y)\| \approx \sqrt{\frac{1}{\Delta x^2}\left[I(x+\frac{\Delta x}{2},y) - I(x-\frac{\Delta x}{2},y)\right]^2 + \frac{1}{\Delta y^2}\left[I(x,y+\frac{\Delta y}{2}) - I(x,y-\frac{\Delta y}{2})\right]^2}$$

Si se recuerda la ecuación (4.58), en la que se mostraban los 4 flujos locales, se ve que se necesitan los siguientes coeficientes: $c(x+\frac{\Delta x}{2},y,t)$, $c(x-\frac{\Delta x}{2},y,t)$, $c(x,y+\frac{\Delta y}{2},t)$, y $c(x,y-\frac{\Delta y}{2},t)$. Por tanto se necesita calcular: $\|\nabla I(x+\frac{\Delta x}{2},y,t)\|$, $\|\nabla I(x-\frac{\Delta x}{2},y,t)\|$, $\|\nabla I(x,y+\frac{\Delta y}{2},t)\|$ y $\|\nabla I(x,y-\frac{\Delta x}{2},t)\|$. Para el primero se tiene para un t fijo:

$$\begin{aligned}\|\nabla I(x+\frac{\Delta x}{2},y)\| \approx & \sqrt{\frac{1}{\Delta x^2}\left[I(x+\Delta x,y) - I(x,y)\right]^2} \\ + & \sqrt{\frac{1}{\Delta y^2}\left[I(x+\frac{\Delta x}{2},y+\frac{\Delta y}{2}) - I(x+\frac{\Delta x}{2},y-\frac{\Delta y}{2})\right]^2}\end{aligned} \tag{C.1}$$

Si se ve la Fig. (4.13) y considerando que $I(x,y)$ se refiere a P_5 se llega a la expresión definitiva:

$$|\nabla I(x+\frac{\Delta x}{2},y)\| \approx \sqrt{\frac{1}{\Delta x^2}(P_6 - P_5)^2 + \frac{1}{\Delta y^2}(\frac{P_9 - P_3}{2})^2} \tag{C.2}$$

Del mismo modo para los otro casos:

$$\begin{aligned} \|\nabla I(x-\frac{\Delta x}{2},y)\| &\approx \sqrt{\frac{1}{\Delta x^2}\left[I(x,y)-I(x-\Delta x,y)\right]^2} \\ &+ \sqrt{\frac{1}{\Delta y^2}\left[I(x-\frac{\Delta x}{2},y+\frac{\Delta y}{2})-I(x-\frac{\Delta x}{2},y-\frac{\Delta y}{2})\right]^2} \end{aligned} \tag{C.3}$$

$$\|\nabla I(x-\frac{\Delta x}{2},y)\| \approx \sqrt{\frac{1}{\Delta x^2}(P_5-P_4)^2+\frac{1}{\Delta y^2}(\frac{P_7-P_1}{2})^2} \tag{C.4}$$

$$\begin{aligned} \|\nabla I(x,y+\frac{\Delta y}{2})\| &\approx \sqrt{\frac{1}{\Delta x^2}\left[I(x+\frac{\Delta x}{2},y+\frac{\Delta y}{2})-I(x-\frac{\Delta x}{2},y+\frac{\Delta y}{2})\right]^2} \\ &+ \sqrt{\frac{1}{\Delta y^2}\left[I(x,y+\Delta y)-I(x,y)\right]^2} \end{aligned} \tag{C.5}$$

$$\|\nabla I(x,y+\frac{\Delta y}{2})\| \approx \sqrt{\frac{1}{\Delta x^2}(\frac{P_9-P_7}{2})^2+\frac{1}{\Delta y^2}(P_8-P_5)^2} \tag{C.6}$$

$$\begin{aligned} \|\nabla I(x,y-\frac{\Delta y}{2})\| &\approx \sqrt{\frac{1}{\Delta x^2}\left[I(x+\frac{\Delta x}{2},y-\frac{\Delta y}{2})-I(x-\frac{\Delta x}{2},y-\frac{\Delta y}{2})\right]^2} \\ &+ \sqrt{\frac{1}{\Delta y^2}\left[I(x,y)-I(x,y-\Delta y)\right]^2} \end{aligned} \tag{C.7}$$

$$\|\nabla I(x,y-\frac{\Delta y}{2})\| \approx \sqrt{\frac{1}{\Delta x^2}(\frac{P_3-P_1}{2})^2+\frac{1}{\Delta y^2}(P_5-P_2)^2} \tag{C.8}$$