Zoom de Imágenes Digitales usando Esquemas de Subdivisión

Autor: Francisco José Ruiz Díaz
Directores:
Sergio Amat Plata
Sonia Busquier Sáez
Juan Carlos Trillo Moya

Febrero 2008

Autor	Francisco José Ruiz Díaz
E-mail del Autor	fjruiz@gmail.com
Director(es)	Sergio Amat Plata, Sonia Busquier Sáez, Juan Carlos Trillo Moya
E-mail del Director	Sergio.Amat@upct.es, Sonia.Busquier@upct.es, Jc.Trillo@upct.es
Título del PFC	Zoom de Imágenes Digitales usando Esquemas de Subdivisión
Descriptores	Zoom, subdivisión, multirresolución, Lagrange, ENO, WENO, PPH,

Resumen

En la actualidad se utilizan con gran profusión las nuevas tecnologías, y en particular la fotografía digital. Aunque siguen conviviendo los dos tipos de fotografía, analógica y digital, la fotografía digital ha tomado mucha importancia en los últimos tiempos y cubre la mayor parte de los campos de aplicación. Las aplicaciones para la fotografía digital son inmensas, tanto en la industria como a nivel de usuario.

A lo largo de este proyecto se desarrolla un estudio sobre algoritmos de zoom en fotografías digitales. Realizar un zoom en las imágenes es algo que puede sernos de utilidad, imaginemos que queremos ampliar una imagen en el escritorio del ordenador para verla con mayor facilidad. Entonces lo que tenemos que hacer es aumentar la resolución de la imagen con la que trabajamos, y en esto consiste un zoom. También hacer zoom es importante en aplicaciones profesionales como pueda ser la medicina, y en general cuando se requiera apreciar mejor los detalles de una foto. En pocas palabras, hacer un zoom básicamente es ampliar una zona de una imagen.

Las herramientas matemáticas con las que vamos a trabajar son los esquemas de subdivisión, éstos pueden interpretarse a partir del operador de predicción que aparece en la multirresolución de Harten.

Vamos a trabajar con diferentes reconstrucciones lineales (Lagrange) y no lineales (ENO, WENO, PPH y ENO subcell resolution), y por tanto con diferentes algoritmos de subdivisión y de multirresolución, tanto lineales como no lineales.

Titulación	I. T. T. Esp. Telemática
Departamento	Matemática Aplicada y Estadística
Fecha de Presentación	Marzo 2008

Índice de Tablas

Índice de figuras

Índice general

Capítulo 1

Introducción

En la actualidad se utilizan con gran profusión las nuevas tecnologías, y en particular la fotografía digital. Aunque siguen conviviendo los dos tipos de fotografía, analógica y digital, la fotografía digital ha tomado mucha importancia en los últimos tiempos y cubre la mayor parte de los campos de aplicación. Las aplicaciones para la fotografía digital son inmensas, tanto en la industria como a nivel de usuario. Entre dichas aplicaciones podemos citar por poner algún ejemplo la eliminación de ruido en imágenes médicas o de satélite, la ecualización de una imagen para mejorar su visionado, el reconocimiento de patrones, la detección de objetos defectuosos en una cadena de fabricación, o la clasificación de algunos productos en base a unas características concretas. Estos no son más que algunos ejemplos, pero las aplicaciones son inmensas (ver por ejemplo [38],[19],[26]).

A lo largo de este proyecto se desarrolla un estudio sobre algoritmos de zoom en fotografías digitales. Realizar un zoom en las imágenes es algo que puede sernos de utilidad, imaginemos que queremos ampliar una imagen en el escritorio del ordenador para verla con mayor facilidad. Entonces lo que tenemos que hacer es aumentar la resolución de la imagen con la que trabajamos, y en esto consiste un zoom. También hacer zoom es importante en aplicaciones profesionales como puede ser la medicina, y en general cuando se requiera apreciar mejor los detalles de una foto. En pocas palabras, hacer un zoom básicamente es ampliar una zona de una imagen.

Las herramientas matemáticas con las que vamos a trabajar son los esquemas de subdivisión. Estos métodos consisten fundamentalmente en lo siguiente: a partir de un conjunto fijo inicial de puntos, se va creando otro

conjunto nuevo de puntos, más denso que el anterior, mediante la aplicación de unas reglas bien definidas. La aplicación recursiva de estas reglas nos permite realizar varias etapas de zoom, y así sucesivas ampliaciones de la imagen. Los esquemas de subdivisión pueden interpretarse a partir del operador de predicción que aparece en la multirresolución de Harten.

La multirresolución de Harten ([8],[28],[29]), es una herramienta muy eficaz para el procesamiento de imágenes. El objetivo de la multirresolución es obtener una reordenación multiescala de la información contenida en un conjunto de datos discretos. Para realizar la tranformación entre los distintos niveles de multirresolución utilizamos los operadores de decimación y predicción. El operador de predicción es el que nos sirve para llevar a cabo el zoom, ya que se encarga de aumentar la resolución de la escala. Los operadores de predicción y de decimación están intimamente relacionados con los operadores de reconstrucción y de discretización, los cuales conectan los diferentes niveles discretos de resolución con un espacio funcional adecuado, el cual depende de las aplicaciones. La característica que hace más atractiva la multirresolución de Harten con respecto a otras técnicas es el hecho de permitir de manera sencilla la introducción de no linealidad en los esquemas. La clave está en que el operador reconstrucción puede ser no lineal y así mejor adaptado a la imagen en concreto con que se trabaja. Igualmente, esta propiedad de no linealidad es heredada por el operador de predicción.

Un problema que aparece muy frecuentemente en teoría de aproximación es reconstruir una función a partir de un conjunto discreto de datos que da información relevante sobre la función misma. Este problema puede ser tratado desde el punto de vista de valores puntuales o de medias en celda, dependiento de si los datos discretos que se consideran son los valores de una función en un conjunto finito de puntos o de medias en celda de la función, respectivamente. En este proyecto trabajaremos en el entorno de valores puntuales o interpolatorio.
Vamos a trabajar con diferentes reconstrucciones lineales (Lagrange) y no lineales (ENO, WENO, PPH y ENO subcell resolution), y por tanto con diferentes algoritmos de subdivisión y de multirresolución, tanto lineales como no lineales ([19],[28],[8],[6]).

Este proyecto está dividido en dos grandes apartados. El primer apartado es donde propiamente realizamos el zoom de una imagen digital mediante el uso de esquemas de subdivisión. Hacemos uso de diferentes operadores de reconstrucción y por tanto de predicción para la definición de diferentes esquemas de subdivisión, los cuales serán aplicados para realizar el zoom

de una imagen digital permitiendo así resaltar zonas que a simple vista no percibimos. Empleando las técnicas que hemos comentado anteriormente, se verá como la imagen se agrandará.

Un nivel del proceso de subdivisión que se aplica a la matriz (imagen o fotografía) original para hacer zoom consiste en aplicar primero un algoritmo de predicción en una dimensión a cada una de las filas de la imagen para así doblar el número de puntos y después realizar la misma operación por columnas.

La matriz en la esquina superior izquierda de la Figura 1.1 corresponde con la imagen original (donde el número de puntos se corresponden con el número de píxeles), la segunda matriz corresponde con la aplicación del algoritmo 1-dimensional a cada una de las filas. Hemos representado mediante puntos negros los valores iniciales y las estrellas son los valores de las predicciones hechas por filas. La tercera matriz, situada debajo de dichas matrices, corresponde con el resultado de aplicar el algoritmo 1-dimensional a cada una de las columnas del resultado del paso anterior. Los valores predecidos de los nuevos píxeles, quedan representados con estrellas anaranjadas y con triángulos rojos.

En el segundo apartado nuestro objetivo es poder medir la bondad del zoom realizado. Para ello partimos de una versión a baja resolución de una imagen y le aplicamos el zoom hasta llegar a la misma resolución de la imagen original. Una vez las imágenes, original y reconstruida, tienen el mismo tamaño es posible compararlas utilizando el $PSNR$, que no es más que una medida de la calidad de la reconstrucción.

La memoria está organizada de la siguiente forma: En el capítulo 2 se hablará de los conceptos más importantes de la multirresolución Harten y se hará la conexión entre algoritmos de multirresolución y de subdivisión. En el capítulo 3 se presenta con cierto detalle la teoría empleada para realizar zoom en imágenes digitales. En el capítulo 4 se exponen algunos experimentos numéricos extrayendo conclusiones de los mismos. En el capítulo 5 se explica el desarrollo de los programas Matlab utilizados en el proyecto, los cuales son añadidos al final del capítulo. Así mismo se expondrá cómo utilizar las interfaces gráficas que han sido implementadas para facilitar al usuario el acceso a los programas. Por último, en el capítulo 6 se presentan las conclusiones extraídas del proyecto.

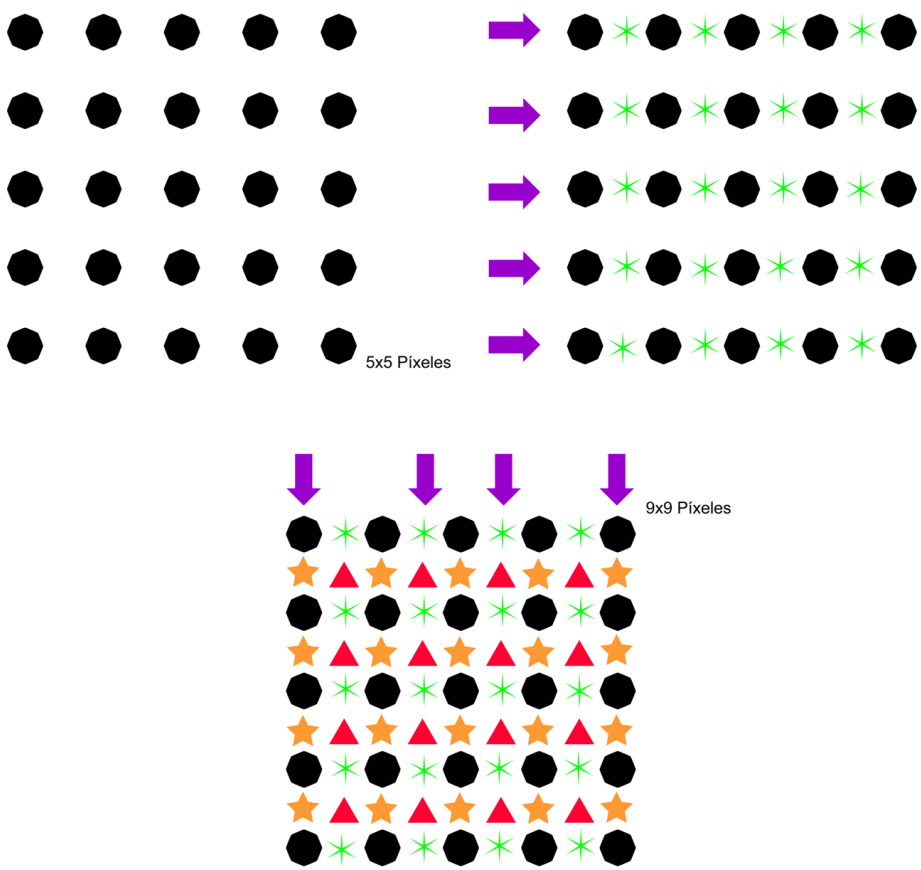

Figura 1.1: Pasos del algoritmo de zoom: imagen original, algoritmo 1D aplicado a las filas y algoritmo 1D aplicado a las columnas.

Capítulo 2

Multirresolución de Harten

El objetivo de la multirresolución es obtener una reordenación multiescala de la información contenida en un conjunto de datos discretos a una cierta resolución. Por ejemplo, esta información puede ser el resultado de discretizar una función, denotada f, y es un elemento perteneciente a un espacio, V^k, en el que k indica el nivel de resolución. Un mayor valor de k indica una mayor resolución. Para realizar la transición entre distintos niveles de resolución se utilizan dos operadores llamados decimación y predicción. El operador decimación proporciona información discreta a un nivel de resolución $k-1$ a partir de la información contenida en el nivel k:

$$D_k^{k-1} : V^k \rightarrow V^{k-1}$$

y debe ser lineal y sobreyectivo. El operador predicción actúa en sentido opuesto, dando una aproximación a la información discreta en el nivel k a partir de la información contenida en el nivel $k-1$:

$$P_{k-1}^k : V^{k-1} \rightarrow V^k$$

Además, al operador predicción no se le exige que sea lineal.

Los datos discretos se obtienen a partir de la discretización de una función f, para lo cual existen distintos operadores. Dependiendo del operador discretización utilizado, la secuencia de datos f^k que resulta es diferente. El objetivo del enfoque propuesto por Harten es la construcción de esquemas de multirresolución adaptados a cada proceso de discretización. Esto se consigue definiendo un operador reconstrucción apropiado. Estos operadores, discretización y reconstrucción, son los elementos a partir de los cuales se

construyen los operadores decimación y predicción del esquema de multirresolución.

Formalmente, sea F un espacio de funciones:

$$F \subset \{f | f : \Omega \subset \mathbb{R}^m \longrightarrow \mathbb{R}\}$$

El operador discretización asigna a cada elemento de este espacio, $f \in F$, una secuencia f^k de datos discretos perteneciente al espacio V^k. De este modo se define el operador discretización:

$$D_k : F \to V^k$$

que ha de ser lineal y sobreyectivo y que a cada $f \in F$ le asocia:

$$f^k = D_k(f)$$

La reconstrucción opera en sentido inverso, tomando una secuencia de datos discretos para reconstruir, a partir de la información proporcionada por dichos datos, la función de la que provienen:

$$R_k : V^k \to F$$

La principal novedad introducida por Harten consiste en que a este operador reconstrucción no se le exige que sea lineal.

Por motivos de consistencia, se requiere que los operadores discretización y reconstrucción satisfagan la siguiente condición:

$$D_k R_k f^k = f^k, \forall f^k \in V^k$$

o expresado de otro modo:

$$D_k R_k = I_{V^k}$$

es decir, si tomamos la información reconstruida a partir de unos datos discretos con una cierta resolución y la discretizamos a ese mismo nivel de resolución, la información discreta obtenida coincide con la original.

En la Figura 2.1 se muestran las relaciones existentes entre los operadores discretización y reconstrucción, y los operadores decimación y predicción. Según estas relaciones, el operador decimación se define del siguiente modo:

$$D_k^{k-1} := D_{k-1} R_k$$

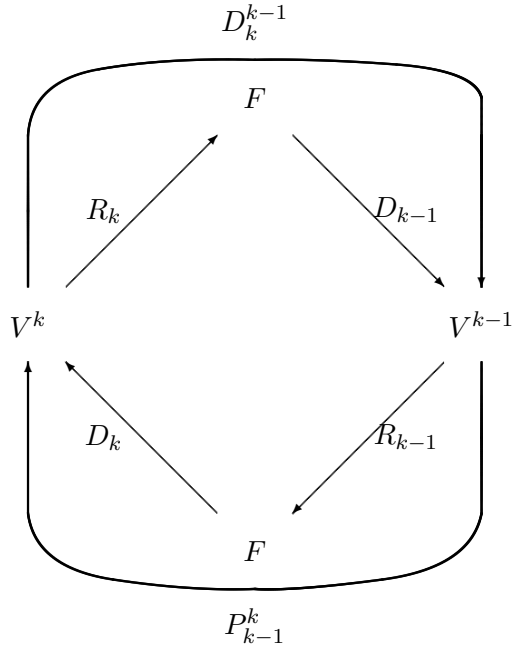

Figura 2.1: Definición de operadores.

Aunque aparentemente el operador decimación depende de la elección del operador reconstrucción, en realidad no es así si (y sólo si) la sucesión de operadores discretización $\{D_k\}$ es anidada, es decir, si se tiene:

$$D_k f = 0 \Longrightarrow D_{k-1} f = 0, \ \ \forall f \in F$$

La propiedad de anidamiento significa que la información contenida en los datos a un cierto nivel de resolución k no será nunca mayor que la información contenida en un nivel de resolución superior.
De forma similar, el operador predicción se construye según la expresión:

$$P_{k-1}^k := D_k R_{k-1}$$

A partir de estas definiciones se obtiene la siguiente relación de consistencia para los operadores decimación y predicción:

$$D_k^{k-1} P_{k-1}^k = D_{k-1} R_k D_k R_{k-1} = D_{k-1} R_{k-1} = I_{V^{k-1}}$$

Esta última relación lo que significa es que cuando utilizamos estos operadores no inventamos información, es decir, si decimamos la información

obtenida a partir de la predicción realizada sobre una información con resolución dada por V^{k-1}, obtenemos exactamente la misma información de partida, sin introducir ningún elemento nuevo.
Consideremos ahora f^k, la información discreta en el nivel de resolución k. Si aplicamos el operador decimación sobre f^k obtenemos f^{k-1}, es decir, la información contenida en el nivel de resolución $k-1$:

$$f^{k-1} = D_k^{k-1} f^k$$

En este caso, podemos interpretar que $P_{k-1}^k f^{k-1}$ es una aproximación a f^k, con un error:

$$e^k := \left(I_{V^k} - P_{k-1}^k D_k^{k-1}\right) f^k =: Q_k f^k \in V^k$$

De esta forma podemos representar la información contenida en f^k en la forma ya descrita, y recíprocamente, conociendo f^{k-1} y e^k se puede calcular f^k mediante la expresión $P_{k-1}^k f^{k-1} + e^k = f^k$.

El problema es que haciendo esto incluimos información redundante, ya que, si suponemos que V^k es un espacio de dimensión finita (como lo es habitualmente en la práctica), $dim\ V^k = N_k$, tenemos por un lado f^k, que contiene la información codificada en N_k elementos, mientras que $\left\{f^{k-1}, e^k\right\}$ contine la misma información codificada en $N_{k-1} + N_k$ elementos. Esta información redundante puede ser eliminada, como consecuencia del siguiente resultado:

$$\begin{aligned} D_k^{k-1} e^k &= D_k^{k-1}\left(I_{V^k} - P_{k-1}^k D_k^{k-1}\right) v^k && (2.1)\\ &= D_k^{k-1} v^k - D_k^{k-1} P_{k-1}^k D_k^{k-1} v^k && (2.2)\\ &= D_k^{k-1} v^k - D_k^{k-1} v^k = 0 && (2.3) \end{aligned}$$

es decir, $e^k \in N\left(D_k^{k-1}\right) = \left\{f^k \in V_k : D_k^{k-1} f^k = 0\right\}$, cuya dimensión es $dimN\left(D_k^{k-1}\right) = dimV^k - dimV^{k-1} = N_k - N_{k-1}$.

Sea $\left\{\mu_i^k\right\}$ el conjunto de elementos que generan el espacio $N\left(D_k^{k-1}\right)$. Podemos expresar el error e^k como:

$$e^k = \sum d_i^k \mu_i^k$$

Si definimos un operador G_k que asocie a cada elemento $e^k \in N\left(D_k^{k-1}\right)$ su correspondiente conjunto de coeficientes $\left\{d_i^k\right\}$, podemos establecer la siguiente equivalencia:

$$f^k \equiv \left\{ f^{k-1}, d^k \right\}$$

mediante las relaciones:

$$f^{k-1} = D_k^{k-1} f^k$$

$$d^k = G_k \left(I - P_{k-1}^k D_k^{k-1} \right) f^k,$$

para obtener $\left\{ f^{k-1}, d^k \right\}$ a partir de f^k, y :

$$P_{k-1}^k f^{k-1} + E_k d^k = f^k,$$

en sentido inverso.

En este caso la equivalencia de información lo es también en cuanto a número de elementos utilizados para codificar dicha información, ya que en $\left\{ f^{k-1}, d^k \right\}$ tenemos $N_{k-1} + (N_k - N_{k-1}) = N_k$ elementos, los mismos que hay en f^k. Decimos entonces que los coeficientes $\left\{ d_i^k \right\}$ contienen la información no redundante del error de predicción, y serán llamados detalles.

Iterando este procedimento en cada nivel de resolución, se consigue la descomposición multiescala que se muestra en la Figura 2.2, y que permite establecer la siguiente equivalencia:

$$f^L \equiv \left\{ f^0, d^L, \ldots, d^1 \right\}$$

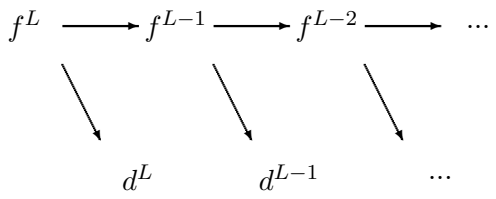

Figura 2.2: Descomposición multiescala.

Por tanto, y para resumir, los algoritmos para las transformaciones directa e inversa de la multirresolución son los siguientes:

(Directa)

$$f^L \to Mf^L = \{f^0, d^1, \ldots, d^L\} \begin{cases} \text{Do} \quad k = L, \ldots, 1 \\ f^{k-1} = D_k^{k-1} f^k \\ d^k = G_k(f^k - P_{k-1}^k f^{k-1}) \end{cases} \tag{2.4}$$

e

(Inversa)

$$Mf^L \to M^{-1}Mf^L \begin{cases} \text{Do} \quad k = 1, \ldots, L \\ f^k = P_{k-1}^k f^{k-1} + E_k d^k \end{cases} \tag{2.5}$$

El paso fundamental en la construcción de un esquema de multirresolución es la definición de un operador reconstrucción apropiado para la discretización que se está considerando. Habitualmente se utilizan dos tipos de discretización: la discretización por valores puntuales y la discretización por medias en celda.

Para este proyecto en cuestión, se utilizará la discretización por valores puntuales, que pasamos a describir en el siguiente apartado.

2.1. Multirresolución para la discretización por valores puntuales en [0,1]

Se considera el conjunto de redes anidadas en el intervalo [0,1] dado por:

$$X^k = \{x_j^k\}_{j=0}^{J_k}, \quad x_j^k = jh_k, \quad h_k = 2^{-k}/J_0, \quad J_k = 2^k J_0,$$

donde J_0 es un entero fijo y X^k una partición uniforme en el intervalo unidad cerrado. La discretización por valores puntuales viene dada por:

$$D_k : \begin{cases} C([0,1]) & \to V^k \\ f & \mapsto f^k = (f_j^k)_{j=0}^{J_k} = (f(x_j^k))_{j=0}^{J_k} \end{cases} \tag{2.6}$$

donde V^k es el espacio de las secuencias reales de dimensión $J^k + 1$. Un operador de reconstrucción para esta discretización es cualquier operador R_k tal que:

$$R_k : V^k \to C([0,1]); \quad \text{y satisface} \quad D_k R_k f^k = f^k, \tag{2.7}$$

lo cual significa que:

$$(R_k f^k)(x_j^k) = f_j^k = f(x_j^k). \tag{2.8}$$

En otras palabras, $(R_k f^k)(x)$ es una función continua que interpola los datos f^k en X^k.

Si se escribe $(R_k f^k)(x) = I_k(x; f^k)$, entonces uno puede definir las transformadas directa (2.4) e inversa (2.5) de la multirresolución como:

$$f^L \to Mf^L \begin{cases} \text{Do} \quad k = L, \ldots, 1 & \\ \quad f_j^{k-1} = f_{2j}^k & 0 \le j \le J_{k-1}, \\ d_j^k \quad = f_{2j-1}^k - I_{k-1}(x_{2j-1}^k; f^{k-1}) & 1 \le j \le J_{k-1}. \end{cases} \tag{2.9}$$

y

$$Mf^L \to M^{-1}Mf^L \begin{cases} \text{Do} \quad k = 1, \ldots, L & \\ f_{2j}^k = f_j^{k-1} & 1 \le j \le J_{k-1}, \\ \quad f_{2j-1}^k = I_{k-1}(x_{2j-1}^k; f^{k-1}) + d_j^k & 0 \le j \le J_{k-1}. \end{cases} \tag{2.10}$$

El valor de los coeficientes d_j^k varía en función de la singularidad, por lo que podemos pensar en el análisis de multirresolución como un análisis de la regularidad de una función. Los mayores coeficientes van asociados a las singularidades de la función, lo que significa que no podemos predecir la información contenida en esas regiones. Más importante aún es el hecho de que si utilizamos una técnica de interpolación independiente de los datos, el conjunto de los intervalos afectados por una singularidad no se reduce únicamente a aquel intervalo en el que se localiza dicha singularidad, sino a todos los intervalos cuyo stencil contenga el intervalo donde se encuentra la singularidad, por lo que habrá una mayor cantidad de coeficientes con valores significativos, y la capacidad de compresión del esquema de multirresolución se verá reducida. Esto es lo que ocurre cuando se utiliza interpolación lineal centrada.

Por otra parte, al utilizar técnicas de interpolación que dependan de los datos, es decir, no lineales, se minimiza la zona afectada por cada singularidad, y la capacidad de compresión del esquema de multirresolución mejora. Esto ocurre cuando se utilizan las técnicas no lineales ENO y ENO-SR.

Las técnicas de interpolación más usuales son los polinomios.

2.2. Relación entre esquemas de Multirresolución y los esquemas de subdivisión (zoom)

Hemos visto que, dados unos datos f^L donde L representa un nivel de resolución, una representación de multirresolución de f^L es cualquier secuencia del tipo $\{f^0, d^1, ..., d^L\}$ donde f^k es una aproximación de f^L en la resolución $k < L$ y d^{k+1} representa los detalles requeridos para conseguir f^{k+1} a partir de f^k (ver Figura 2.2 y los algoritmos (2.4) y (2.5)).

Si en estos algoritmos de multirresolución nos quedamos solamente con la etapa de decodificación de la señal o ascenso por la pirámide de multirresolución, entonces lo que tenemos es un esquema de subdivisión. Por tanto, un esquema de subdivisión S queda definido a través del operador de predicción P_k^{k-1} que se utilice, es decir, que dada una secuencia discreta:

$$f^{k-1} \in V^{k-1},\ f^k = Sf^{k-1}$$

queda definido como:

$$f^k = Sf^{k-1} = P_{k-1}^k f^{k-1}$$

Como en la construcción del operador de predicción, interviene de forma decisiva el operador de reconstrucción, la construcción de éste será una etapa crucial en el diseño del esquema de subdivisión. La posibilidad de considerar reconstrucciones no lineales abre entonces un amplio espectro de posibilidades.

En el caso interpolatorio, en el que la discretización viene dada por los valores en una malla de una función, el esquema de subdivisión tiene la forma:

$$f^k = Sf^k \begin{cases} f_{2j}^k = (Sf^{k-1})_{2j} = f_j^{k-1} & 1 \le j \le J_{k-1}, \\ f_{2j-1}^k = (Sf^{k-1})_{2j-1} = I_{k-1}(x_{2j-1}^k; f^{k-1}) & 0 \le j \le J_{k-1}. \end{cases} \tag{2.11}$$

El hecho de que en este caso las reconstrucciones equivalen a interpolaciones simplifica mucho su diseño, y aún más si consideramos que las interpolaciones más usuales son polinómicas. A continuación vamos a definir una reconstrucción lineal basada en una interpolación de Lagrange con 4 puntos, así como diferentes reconstrucciones no lineales (ENO, WENO, PPH, ENO Subcell Resolution) que premitirán una mejora en los resultados obtenidos debido a su adaptación a los datos.

Capítulo 3

Diferentes reconstrucciones en el entorno de valores puntuales

Dados unos datos f^L donde L representa un nivel de resolución, una representación de multirresolución de f^L es cualquier secuencia del tipo $\{f^0, d^1, ..., d^L\}$ donde f^k es una aproximación de f^L en la resolución $k < L$ y d^{k+1} representa los detalles requeridos para conseguir f^{k+1} a partir de f^k. Las representaciones de multirresolución lineales de los datos, como las descomposiciones 'wavelet', son multirresoluciones que implican operadores lineales de interresolución.

La eficiencia de las descomposiciones de multirresolución lineal está limitada por la presencia de bordes o ejes en las imágenes. Los coeficientes numéricamente significativos d_j^k son principalmente aquellos para los cuales el soporte de la función wavelet asociada cruza las discontinuidades.

El entorno de resolución de Harten ha sido desarrollado para incorporar un tratamiento específico adaptado a las singularidades. La ventaja de este marco general está en su flexibilidad, donde el operador de reconstrucción juega un papel principal. Los niveles de resolución discretos son conectados por operadores de interresolución, llamados decimación (desde un nivel de resolución(k) a un nivel de resolución menor($k-1$)) y predicción (de menor resolución a mayor resolución). Estos operadores entre escalas están directamente relacionados con los operadores de discretización y de reconstrucción, los cuales actúan entre un espacio funcional adecuado (el cual depende de los datos discretos con los que se trabaja) y cada uno de los niveles discre-

tos (donde quedan representadas las secuencias f^k). Este entorno de Harten hace posible considerar técnicas de reconstrucción dependientes de los datos. Se pueden considerar diferentes tipos de algoritmos de multirresolución dependiendo del operador de discretización lineal que produce los datos. En este proyecto se considera la multirresolución por valores puntuales porque es en este entorno donde se obtienen reconstrucciones de manera más fácil. Usando funciones primitivas podemos obtener las reconstrucciones asociadas al caso de discretización por medias en celda.
En el caso de valores puntuales el operador de reconstrucción equivale a hacer interpolación. Normalmente se consideran interpolaciones independientes de los datos. Una opción para obtener una adaptación cerca de las singularidades es considerar interpolación no lineal(dependiente de los datos). Por ejemplo, usando los esquemas ENO (esencialmente no oscilatorios) podemos obtener una alta exactitud en todos los intervalos sin ninguna singularidad.

3.1. *Técnicas de Interpolación*

El problema a resolver consiste en, a partir de una información discreta dada sobre una función, recuperar la función original. El objetivo es encontrar una función sencilla (generalmente polinomios o funciones trigonométricas) que aproxime a la función original, utilizando la información disponible. Para ello se pueden utilizar técnicas lineales, independientes de los datos, que resultan muy sencillas de aplicar, o también técnicas no lineales, que al ser de tipo adaptativo, proporcionan una mejor aproximación.

3.1.1. *Técnicas de reconstrucción lineal: La Interpolación de Lagrange*

Tomando $\mathcal{S}$ como el conjunto:

$$\mathcal{S} = \mathcal{S}(r,s) = \{-s, -s+1, \ldots, -s+r\}, \quad r \geq s > 0,\ r \geq 1,$$

y $\{L_m(y)\}_{m\in S}$ como los polinomios interpoladores de Lagrange de grado r basados en los elementos del conjunto $S(r,s)$,

$$L_m(y) = \prod_{l=-s, l\neq m}^{-s+r} (\frac{y-l}{m-l}), \quad L_m(j) = \delta_j^m, \quad j \in \mathcal{S}$$

La interpolación de Lagrange para:

$\{x^k_{j+m}\}_{m\in\mathcal{S}}$, se escribe

$$\mathcal{I}_k(x, f^k) = \sum_{m=-s}^{-s+r} f^k_{j+m} L_m(\frac{x - x^k_j}{h_k}), \quad x \in [x^k_{j-1}, x^k_j], \quad 1 \le j \le J_k.$$

Es importante observar que si $f(x) = P(x)$, donde $P(x)$ es un polinomio de grado menor o igual que r, entonces $\mathcal{I}_k(x, f^k) = f(x)$ para $x \in [x^k_{j-1}, x^k_j]$. Lo cual significa que, para funciones suaves:

$$\mathcal{I}_k(x, f^k) = f(x) + O(h_k)^{r+1}$$

Por lo tanto, el orden del procedimiento de reconstrucción, que caracteriza su precisión, será: $p = r + 1$. La situación particular para el valor $r = 2s - 1$ se corresponde con un stencil de interpolación simétrico respecto del intervalo $[x^k_{j-1}, x^k_j]$. Por ejemplo, para $s = 2(r = 3)$ se obtiene la siguiente transformada de multirresolución:

$$\begin{cases} \text{Do} \quad k = L, \dots, 1 & \\ \quad f^{k-1}_j = f^k_{2j} & 0 \le j \le J_{k-1}, \\ d^k_j = f^k_{2j-1} - (\frac{-f^{k-1}_{j-2} + 9f^{k-1}_{j-1} + 9f^{k-1}_j - f^{k-1}_{j+1}}{16}), & 1 \le j \le J_{k-1}. \end{cases} \tag{3.1}$$

$$\begin{cases} \text{Do} \quad k = 1, \dots, L & \\ f^k_{2j} = f^{k-1}_j & 0 \le j \le J_{k-1}, \\ f^k_{2j-1} = d^k_j + (\frac{-f^{k-1}_{j-2} + 9f^{k-1}_{j-1} + 9f^{k-1}_j - f^{k-1}_{j+1}}{16}), & 1 \le j \le J_{k-1}. \end{cases} \tag{3.2}$$

Y por tanto, el esquema de subdivisión queda:

$$f^k = Sf^{k-1} \begin{cases} f^k_{2j} = f^{k-1}_j & 0 \le j \le J_{k-1}, \\ f^k_{2j-1} = (\frac{-f^{k-1}_{j-2} + 9f^{k-1}_{j-1} + 9f^{k-1}_j - f^{k-1}_{j+1}}{16}), & 1 \le j \le J_{k-1}. \end{cases} \tag{3.3}$$

Las técnicas de interpolación de Lagrange pierden mucha de su precisión en presencia de singularidades, por ejemplo en presencia de saltos en la función el error se comporta como:

$$f(x) = \mathcal{I}^L_k(x, f^k) + O([f]) \tag{3.4}$$

Cuando se utiliza interpolación lineal centrada, cada stencil (conjunto de puntos usados para interpolar) se elige de forma independiente de los datos, es decir, no se tiene en cuenta la suavidad de la función que se interpola. Esto significa que una singularidad aislada en un intervalo I_k producirá una pérdida de exactitud en varios subintervalos de la partición, ya que los stencils correspondientes a los subintervalos adyacentes a I_k contendrán también la singularidad, con lo que el error de interpolación vendrá dado por la expresión (3.4).

3.1.2. *Técnicas de reconstrucción no lineal: La Interpolación ENO*

La idea de las reconstrucciones ENO es construir trozos o partes polinomiales usando datos pertenecientes a las regiones suaves de la función en la medida de lo posible. El punto clave en la interpolación ENO es el proceso de selección del stencil S (conjunto de datos usados para construir el polinomio interpolador), el cual se intenta elegir dentro de una región suave de la función $f(x)$. De forma más precisa, este proceso de selección trabaja de la siguiente manera: para cada intervalo, $\left[x_{j-1}^k, x_j^k\right]$, se consideran todos los posibles conjuntos S con $r \geq 2$ puntos incluyendo los puntos x_{j-1}^k y x_j^k.

Denominemos el stencil ENO para el jth intervalo:

$$S_j^{ENO} = \left\{x_{s_{j-1}}^k, x_{s_j}^k, ..., x_{s_j+r-1}^k\right\},$$

siendo $r+1$ el orden de la interpolación.

Existen dos estrategias para realizar esta selección: la estrategia jerárquica y la no jerárquica.

La selección de S_j^{ENO} se realiza como sigue:

(1) La selección jerárquica del stencil consiste en, partiendo de los extremos del intervalo $\left[x_{j-1}^k, x_j^k\right]$, ir añadiendo puntos a derecha o izquierda, comparando las diferencias divididas correspondientes a los conjuntos formados por los extremos del intervalo más los puntos añadidos, y escogiendo la de menor valor absoluto. El algoritmo para este procedimiento es el siguiente:

Set $s_0 = j$
for $l = 0, \ldots, r-2$
  if $|f[x^k_{s_l-2}, \ldots, x^k_{s_l+l}]| < |f[x^k_{s_l-1}, \ldots, x^k_{s_l+l+1}]|$
    $s_{l+1} = s_l - 1$
  else
    $s_{l+1} = s_l$
  end
end
$s_j = s_{r-1}$

y,

(2) La selección no jerárquica del stencil considera las diferencias divididas de mayor orden correspondientes a todos los stencils posibles, y calcula el mínimo de entre todos los valores absolutos de dichas diferencias. El algoritmo es el siguiente:

Se elige un s_j tal que

Choose s_j such that
$|f[x^k_{s_j-1}, \ldots, x^k_{s_j+r-1}])| = \text{mín}_{j-r+1 \le l \le j}\{|f[x^k_{l-1}, \ldots, x^k_{l+r-1}]|\}$

Ambos algoritmos, (1) y (2) conducen asintóticamente a conjuntos de puntos de interpolación que se mueven lejos de la discontinuidad. Consecuentemente, el orden de aproximación del operador de predicción ENO sigue siendo $r + 1$ siempre que sea posible evitar la discontinuidad.

Notar además, que no hay diferencia entre ambos algoritmos cuando $r = 2$, pero los stencils obtenidos pueden variar cuando $r > 2$. En cualquier caso, se observa que el stencil siempre contiene los extremos del subintervalo en el que se realiza la interpolación, evitando siempre que sea posible los intervalos que contienen singularidades.

En el caso en que $r = 3$ tenemos tres posibles stencils (ver Figura 3.1), es decir,

$$S_j^1 = \left\{x_{j-3}^k, x_{j-2}^k, x_{j-1}^k, x_j^k\right\},$$
$$S_j^2 = \left\{x_{j-2}^k, x_{j-1}^k, x_j^k, x_{j+1}^k\right\},$$
$$S_j^3 = \left\{x_{j-1}^k, x_j^k, x_{j+1}^k, x_{j+2}^k\right\}.$$

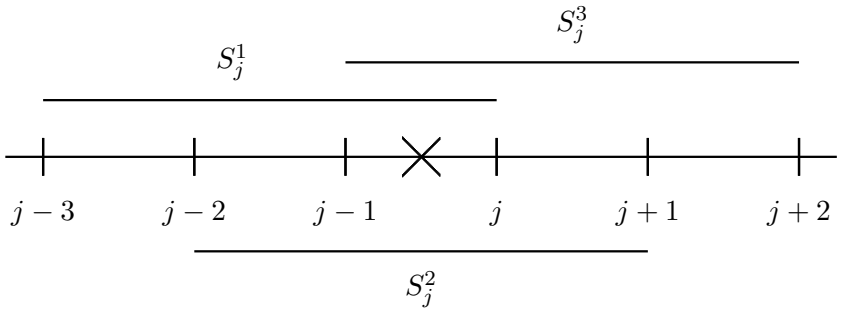

Figura 3.1: Stencil ENO.

El algoritmo de multirresolución queda,

$$\begin{cases} \text{Do} \quad k = L, \ldots, 1 \\ \quad f_j^{k-1} = f_{2j}^k \\ \quad d_j^k = \begin{cases} f_{2j-1}^k - (5f_{j-3}^{k-1} + 15f_{j-2}^{k-1} - 5f_{j-1}^{k-1} + f_j^{k-1}), & si\ S_j^1 \\ f_{2j-1}^k - (\frac{-f_{j-2}^{k-1} + 9f_{j-1}^{k-1} + 9f_j^{k-1} - f_{j+1}^{k-1}}{16}), & si\ S_j^2 \\ f_{2j-1}^k - (5f_{j-1}^{k-1} + 15f_j^{k-1} - 5f_{j+1}^{k-1} + f_{j+2}^{k-1}), & si\ S_j^3 \end{cases} \end{cases} \tag{3.5}$$

$$\begin{cases} \text{Do} \quad k = 1, \ldots, L \\ \quad f_{2j}^k = f_j^{k-1} \\ \quad f_{2j-1}^k = \begin{cases} d_j^k + (5f_{j-3}^{k-1} + 15f_{j-2}^{k-1} - 5f_{j-1}^{k-1} + f_j^{k-1}), & si\ S_j^1 \\ d_j^k + (\frac{-f_{j-2}^{k-1} + 9f_{j-1}^{k-1} + 9f_j^{k-1} - f_{j+1}^{k-1}}{16}), & si\ S_j^2 \\ d_j^k + (5f_{j-1}^{k-1} + 15f_j^{k-1} - 5f_{j+1}^{k-1} + f_{j+2}^{k-1}), & si\ S_j^3 \end{cases} \end{cases} \tag{3.6}$$

El correspondiente esquema de subdivisión es entonces:

$$\begin{cases} \text{Do} \quad k = 1, \ldots, L \\ \quad f_{2j}^k = f_j^{k-1} \\ \quad f_{2j-1}^k = \begin{cases} 5f_{j-3}^{k-1} + 15f_{j-2}^{k-1} - 5f_{j-1}^{k-1} + f_j^{k-1}, & si\ S_j^1 \\ \frac{-f_{j-2}^{k-1} + 9f_{j-1}^{k-1} + 9f_j^{k-1} - f_{j+1}^{k-1}}{16}, & si\ S_j^2 \\ 5f_{j-1}^{k-1} + 15f_j^{k-1} - 5f_{j+1}^{k-1} + f_{j+2}^{k-1}, & si\ S_j^3 \end{cases} \end{cases} \tag{3.7}$$

Los resultados anteriores se pueden mejorar, si conocemos la localización de la singularidad. Esta es la idea básica en la que se basa la técnica de interpolación ENO-SR (Subcell Resolution), introducida por Harten en [27].

3.1.3. *Técnicas de reconstrucción no lineal: La Interpolación ENO-SR*

Supongamos que f es una función continua con una esquina en $\xi \in [x_{j-1}^k, x_j^k]$ y que los polinomios $q_{j\pm1}(x)$ asociados a stencils completamente descentrados satisfacen:

$$f(x) = q_{j-1}(x) + O(h^{r+1}) \quad x \in [x_{j-2}^k, x_{j-1}^k]$$

$$f(x) = q_{j+1}(x) + O(h^{r+1}) \quad x \in [x_j^k, x_{j+1}^k]$$

Definimos la función:

$$G_j(x) := q_{j+1}(x) - q_{j-1}(x)$$

Haciendo los desarrollos de Taylor:

$$q_{j+1}(x) = f(\xi)^+ + f'(\xi)^+(x-\xi) + \frac{f''(\xi)^+}{2}(x-\xi)^2 + \ldots + O(h^{r+1})$$

$$q_{j-1}(x) = f(\xi)^- + f'(\xi)^-(x-\xi) + \frac{f''(\xi)^-}{2}(x-\xi)^2 + \ldots + O(h^{r+1})$$

deducimos que:

$$G_j(x) = [f]_\xi + (x-\xi)[f']_\xi + (x-\xi)^2\frac{[f'']_\xi}{2} + \ldots + O(h^{r+1})$$

y de aquí teniendo en cuenta que f es continua (esta es la razón por la que esta técnica no vale para funciones con salto):

$$G_j(x) = (x-\xi)[f']_\xi + \lambda(x) \qquad \lambda(x) = (x-\xi)^2 \frac{[f'']_\xi}{2} + \ldots + O(h^{r+1}) \quad (3.8)$$

Por tanto para h_k suficientemente pequeño, tal que $\lambda(x_{j-1}^k)$ y $\lambda(x_j^k)$ tiendan a cero mucho más rápido que $min((x_j^k - \xi), (x_{j-1}^k - \xi))$, se tiene que existe una raíz de G_j en la celda (x_{j-1}^k, x_j^k).

Supongamos entonces que existe μ tal que $G_j(\mu) = 0$, y veamos que μ puede ser una buena aproximación a la localización de la esquina de f.
En el caso especial en que la función f viene dada por:

$$\begin{cases} P_I(x) & x \leq \xi \\ P_D(x) & x \geq \xi \end{cases} \qquad \begin{cases} P_I(\xi) & = & P_D(\xi) \\ P_I'(\xi) & \neq & P_D'(\xi) \end{cases}$$

donde P_I y P_D son polinomios tales que $max(grado(P_I), grado(P_D)) \leq r$.
Entonces tendremos que:

$$q_{j-1}(x) = P_I(x) \qquad q_{j+1}(x) = P_D(x)$$

con lo cual $\xi = \mu$.
En el caso general en el que P_I y P_D son funciones suaves, lo que tenemos es que:

$$q_{j+1}(x) = P_I(x) + O(h_k^{r+1})$$
$$q_{j-1}(x) = P_D(x) + O(h_k^{r+1})$$

y por tanto:

$$G_j(x) = P_I(x) - P_D(x) + O(h_k^{r+1})$$

Definiendo ahora $\hat{G}(x) := P_I(x) - P_D(x)$ tenemos que $G_j(x) = \hat{G}(x) + O(h_k^{r+1})$. Sustituyendo por μ, y haciendo un desarrollo de Taylor de $\hat{G}$ alrededor de ξ llegamos a:

$$0 = G_j(\mu) = \hat{G}(\xi) + (\mu-\xi)\hat{G}'(\xi) + (\mu-\xi)^2 \frac{\hat{G}''(\xi)}{2} + \ldots + O(h^{r+1})$$

Y de aquí teniendo en cuenta que $\hat{G}(\xi) = 0$, $\hat{G}'(\xi) \neq 0$

$$(\mu-\xi) = (\mu-\xi)^2 \frac{\hat{G}''(\xi)}{2\hat{G}'(\xi)} + \ldots + O(h_k^{r+1}) \quad (3.9)$$

Exigiendo que $|\mu-\xi|$ sea suficientemente pequeño como para que el término $(\mu-\xi)^2\frac{\hat{G}''(\xi)}{2\hat{G}'(\xi)}$ y sucesivos sean mucho más pequeños, lo cual es equivalente a decir que h_k sea suficientemente pequeño, esto conduce a:

$$|\mu-\xi| = O(h_k^{r+1})$$

Si el factor $\frac{\hat{G}''(\xi)}{2\hat{G}'(\xi)}$ es grande, entonces podremos localizar la esquina ξ con precisión sólo si h_k es suficientemente pequeño.
Esto motiva la definición de la reconstrucción como:

$$\mathcal{I}_k^{SR}(x,f^k) = \begin{cases} q_l(x) & x\in[x_{l-1}^k, x_l^k], \quad l\neq j \\ q_{j-1}(x) & x\in[x_{j-1}^k,\mu] \\ q_{j+1}(x) & x\in[\mu,x_j^k]. \end{cases}$$

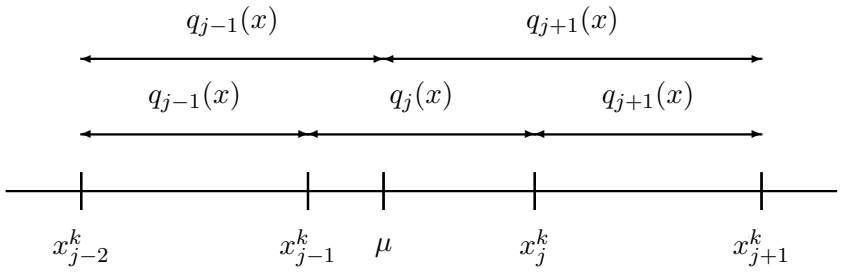

Figura 3.2: Definición del operador de reconstrucción para la técnica ENO-SR.

Entonces:

$$\mathcal{I}_k^{SR}(x_l,f^k) = f(x_l), \qquad \forall\ x_l^k \in X^k$$

$$\frac{d^m}{dx^m}\mathcal{I}_k^{SR}(x,f^k) = \frac{d^m}{dx^m}f(x) + O(h_k^{r+1-m}), \qquad 0\leq m\leq r$$

para todos los puntos excepto para una banda de longitud $O(h_k^{r+1})$ alrededor de ξ, siempre que h_k sea suficientemente pequeño.

Los pasos a seguir para programar este método son los siguientes:

1. Computar los stencils asignados a los subintervalos I_{j-1} y I_{j+1}. Nosotros hemos usado el algoritmo jerárquico, pero como se ha dicho anteriormente podría usarse cualquiera de los dos.

2. Comprobar si:
$$S_{j-1} \cap S_{j+1} = \emptyset$$
si esto es así etiquetar el intervalo como sospechoso de contener una singularidad

3. Definir la función:
$$G(x) = q_{j+1}(x) - q_{j-1}(x)$$
y usarla para decidir si hay una esquina en (x_{j-1}^k, x_j^k) o no. Habrá una esquina si:
$$G(x_{j-1}^k)G(x_j^k) < 0$$
según se deduce inmediatamente de la ecuación (3.8) aplicando el teorema de Bolzano en la celda $[x_{j-1}^k, x_j^k]$ a la función G.

El algoritmo de multirresolución se puede escribir,

$$\left\{\begin{array}{l} \text{Do} \quad k = L, \ldots, 1 \\ \quad f_j^{k-1} = f_{2j}^k \\ \quad d_j^k = \left\{\begin{array}{ll} ENO \left\{\begin{array}{ll} f_{2j-1}^k - (5f_{j-3}^{k-1} + 15f_{j-2}^{k-1} - 5f_{j-1}^{k-1} + f_j^{k-1}), & si\ S_j^1 \\ f_{2j-1}^k - (\frac{-f_{j-2}^{k-1}+9f_{j-1}^{k-1}+9f_j^{k-1}-f_{j+1}^{k-1}}{16}), & si\ S_j^2 \\ f_{2j-1}^k - (5f_{j-1}^{k-1} + 15f_j^{k-1} - 5f_{j+1}^{k-1} + f_{j+2}^{k-1}), & si\ S_j^3 \end{array}\right. \\ ENO - SR \left\{\begin{array}{ll} f_{2j-1}^k - (\frac{-5f_{j-4}^{k-1}+21f_{j-3}^{k-1}-35f_{j-2}^{k-1}+35f_{j-1}^{k-1}}{16}), & si\ q_{j-1} \\ f_{2j-1}^k - (\frac{35f_j^{k-1}-35f_{j+1}^{k-1}+21f_{j+2}^{k-1}-5f_{j+3}^{k-1}}{16}), & si\ q_{j+1} \end{array}\right. \end{array}\right. \end{array}\right. \tag{3.10}$$

$$\left\{\begin{array}{l} \text{Do} \quad k = 1, \ldots, L \\ \quad f_{2j}^k = f_j^{k-1} \\ \quad f_{2j-1}^k = \left\{\begin{array}{ll} ENO \left\{\begin{array}{ll} d_j^k + (5f_{j-3}^{k-1} + 15f_{j-2}^{k-1} - 5f_{j-1}^{k-1} + f_j^{k-1}), & si\ S_j^1 \\ d_j^k + (\frac{-f_{j-2}^{k-1}+9f_{j-1}^{k-1}+9f_j^{k-1}-f_{j+1}^{k-1}}{16}), & si\ S_j^2 \\ d_j^k + (5f_{j-1}^{k-1} + 15f_j^{k-1} - 5f_{j+1}^{k-1} + f_{j+2}^{k-1}), & si\ S_j^3 \end{array}\right. \\ ENO - SR \left\{\begin{array}{ll} d_j^k + (\frac{-5f_{j-4}^{k-1}+21f_{j-3}^{k-1}-35f_{j-2}^{k-1}+35f_{j-1}^{k-1}}{16}), & si\ q_{j-1} \\ d_j^k + (\frac{35f_j^{k-1}-35f_{j+1}^{k-1}+21f_{j+2}^{k-1}-5f_{j+3}^{k-1}}{16}), & si\ q_{j+1} \end{array}\right. \end{array}\right. \end{array}\right. \tag{3.11}$$

El correspondiente esquema de subdivisión es entonces:

$$\begin{cases} \text{Do} \quad k = 1, \ldots, L \\ \quad f_{2j}^k = f_j^{k-1} \\ f_{2j-1}^k = \begin{cases} ENO \begin{cases} 5f_{j-3}^{k-1} + 15f_{j-2}^{k-1} - 5f_{j-1}^{k-1} + f_j^{k-1}, & si\ S_j^1 \\ \frac{-f_{j-2}^{k-1} + 9f_{j-1}^{k-1} + 9f_j^{k-1} - f_{j+1}^{k-1}}{16}, & si\ S_j^2 \\ 5f_{j-1}^{k-1} + 15f_j^{k-1} - 5f_{j+1}^{k-1} + f_{j+2}^{k-1}, & si\ S_j^3 \end{cases} \\ ENO - SR \begin{cases} \frac{-5f_{j-4}^{k-1} + 21f_{j-3}^{k-1} - 35f_{j-2}^{k-1} + 35f_{j-1}^{k-1}}{16}, & si\ q_{j-1} \\ \frac{35f_j^{k-1} - 35f_{j+1}^{k-1} + 21f_{j+2}^{k-1} - 5f_{j+3}^{k-1}}{16}, & si\ q_{j+1} \end{cases} \end{cases} \end{cases} \tag{3.12}$$

Partiendo de distintos puntos de vista, se han propuesto otras muchas técnicas de interpolación que intentan mejorar los resultados proporcionados por la técnica ENO. Una de estas características es la llamada WENO (Weighted ENO), que se describe en la siguiente sección.

3.1.4. *Técnicas de reconstrucción no lineal: La Interpolación WENO*

Mediante la interpolación ENO se consigue una aproximación con un buen orden de exactitud en todos los intervalos excepto en aquellos que contienen singularidades. Pero existen algunas características de esta técnica de interpolación que se podrían mejorar:

En primer lugar, el proceso de selección del stencil es muy sensible a las perturbaciones: si los valores de las diferencias divididas que se comparan en el criterio de selección son muy similares, una pequeña perturbación, como un error de redondeo, por ejemplo, podría hacer que el stencil seleccionado cambiase.

Por otra parte, en las regiones suaves de la función no es necesario realizar esta selección de stencil, ya que el stencil centrado que utiliza la interpolación lineal producirá la misma aproximación.

Por último, se podría aumentar el orden de exactitud de la aproximación, ya que el método ENO consiste en seleccionar uno de entre todos los stencils posibles, para hacer la interpolación en cada subintervalo. Tomando stencils formados por r intervalos, esto significa que la interpolación ENO se

hará seleccionando un stencil entre r posibles, dando una aproximación con un orden de exactitud igual a r. Hay $2r-1$ subintervalos contenidos en los r stencils, por lo que se pierde la información proporcionada por $r-1$ de estos intervalos. Si la función es suave en estas regiones, esta información podría servir para obtener una mejor aproximación, de manera que utilizando la información dada por los $2r-1$ subintervalos contenidos en los distintos stencils, se podría obtener un orden de exactitud máximo igual a $2r$ (en el caso de valores puntuales).

Para solucionar los dos primeros problemas, se presentó en [24] y en [39] una estrategia de sesgo, que consiste en tomar como base un stencil preferido, que será el stencil centrado en el intervalo donde se realiza la interpolación, y utilizarlo para modificar el criterio de selección de stencil con un parámetro de sesgo. La idea es no alejarse de este stencil, excepto en el caso de que en el stencil alternativo la función sea mucho más suave, y este mucho viene dado por el sesgo.

Posteriormente, Liu et al. [37] introdujeron, en el contexto de las leyes de conservación, la técnica WENO, como mejora de la interpolación ENO. Una versión más eficiente de esta técnica fue propuesta por Jiang y Shu en [32]. La diferencia entre ENO y WENO radica en la forma en que se construye el interpolante. El interpolante ENO se construye seleccionando un stencil para cada subintervalo, mientras que en el método WENO, a cada subintervalo se le asignan todos los stencils posibles, y el polinomio interpolador se calcula como una combinación lineal convexa de los polinomios correspondientes a dichos stencils. En esta combinación lineal, se da más importancia a los polinomios construidos a partir de stencils en los que la función es suave, de forma que los polinomios que cruzan alguna singularidad tienen un efecto prácticamente nulo. De este modo, se conserva el efecto ENO, es decir, la interpolación en intervalos próximos a singularidades se hace tomando información sólo de regiones donde la función es suave, y además, como se utiliza una combinación convexa de polinomios interpoladores, los errores cometidos por unos se pueden cancelar con los de otros, resultando en un orden de aproximación mayor.

El algoritmo de multirresolución en este caso es

$$\begin{cases} \text{Do} \quad k = L, \ldots, 1 \\ \quad f_{j-1}^{k-1} = f_{2j}^{k} \\ d_j^k = f_{2j-1}^k - (w_{j-1}^{k-1} s_{j-1}^{k-1} + w_j^{k-1} s_j^{k-1} + w_{j+1}^{k-1} s_{j+1}^{k-1}) \end{cases} \tag{3.13}$$

$$\begin{cases} \text{Do} \quad k = 1, \ldots, L \\ \quad f_{2j}^{k} = f_{j-1}^{k-1} \\ f_{2j-1}^k = d_j^k + (w_{j-1}^{k-1} s_{j-1}^{k-1} + w_j^{k-1} s_j^{k-1} + w_{j+1}^{k-1} s_{j+1}^{k-1}) \end{cases} \tag{3.14}$$

donde

$$s_{j-1}^{k-1} = 5f_{j-3}^{k-1} + 15f_{j-2}^{k-1} - 5f_{j-1}^{k-1} + f_j^{k-1}$$

$$s_j^{k-1} = \frac{-1f_{j-2}^{k-1} + 9f_{j-1}^{k-1} + 9f_j^{k-1} - 1f_{j+1}^{k-1}}{16}$$

$$s_{j+1}^{k-1} = 5f_{j-1}^{k-1} + 15f_j^{k-1} - 5f_{j+1}^{k-1} + f_{j+2}^{k-1}$$

es decir, los valores en el punto x_{2j-1}^k de los polinomios interpoladores basados en los stencils S_j^1, S_j^2 y S_j^3 respectivamente (ver Figura 3.1).

Y los pesos w_{j-1}^{k-1}, w_j^{k-1} y w_{j+1}^{k-1} se calculan mediante las expresiones

$$w_{j-1}^{k-1} = \frac{\alpha_{j-1}^{k-1}}{\alpha_{j-1}^{k-1} + \alpha_j^{k-1} + \alpha_{j+1}^{k-1}}$$

$$w_j^{k-1} = \frac{\alpha_j^{k-1}}{\alpha_{j-1}^{k-1} + \alpha_j^{k-1} + \alpha_{j+1}^{k-1}}$$

$$w_{j+1}^{k-1} = \frac{\alpha_{j-1}^{k-1}}{\alpha_{j-1}^{k-1} + \alpha_j^{k-1} + \alpha_{j+1}^{k-1}}$$

Los valores de α_{j-1}^{k-1}, α_j^{k-1} y α_{j+1}^{k-1} son

$$\alpha_{j-1}^{k-1} = \frac{\frac{3}{16}}{\epsilon + IS_{j-1}^{k-1}}, \quad \alpha_j^{k-1} = \frac{\frac{10}{16}}{\epsilon + IS_j^{k-1}}, \quad \alpha_{j+1}^{k-1} = \frac{\frac{3}{16}}{\epsilon + IS_{j+1}^{k-1}}.$$

con ϵ una constante positiva que se introduce para evitar que el denominador se anule, y suele tener el valor $\epsilon = 10^{-5}$ o $\epsilon = 10^{-6}$.

Finalmente necesitamos saber cómo calcular los indicadores de suavidad IS_{j-1}^{k-1}, IS_j^{k-1} y IS_{j+1}^{k-1}. Una posibilidad es (ver Liu et al. [37]).

$$\begin{aligned} IS_{j-1}^{k-1} &= \frac{1}{2}\left[\left(f\left[x_{j-2}^{k-1}, x_{j-1}^{k-1}\right] - f\left[x_{j-3}^{k-1}, x_{j-2}^{k-1}\right]\right)^2 + \left(f\left[x_{j-1}^{k-1}, x_j^{k-1}\right] - f\left[x_{j-2}^{k-1}, x_{j-1}^{k-1}\right]\right)^2\right] \\ &+ \left(f\left[x_{j-1}^{k-1}, x_j^{k-1}\right] - 2f\left[x_{j-2}^{k-1}, x_{j-1}^{k-1}\right] + f\left[x_{j-3}^{k-1}, x_{j-2}^{k-1}\right]\right)^2 \end{aligned}$$

$$\begin{aligned} IS_j^{k-1} &= \frac{1}{2}\left[\left(f\left[x_{j-1}^{k-1}, x_j^{k-1}\right] - f\left[x_{j-2}^{k-1}, x_{j-1}^{k-1}\right]\right)^2 + \left(f\left[x_j^{k-1}, x_{j+1}^{k-1}\right] - f\left[x_{j-1}^{k-1}, x_j^{k-1}\right]\right)^2\right] \\ &+ \left(f\left[x_j^{k-1}, x_{j+1}^{k-1}\right] - 2f\left[x_{j-1}^{k-1}, x_j^{k-1}\right] + f\left[x_{j-2}^{k-1}, x_{j-1}^{k-1}\right]\right)^2 \end{aligned}$$

$$\begin{aligned} IS_{j+1}^{k-1} &= \frac{1}{2}\left[\left(f\left[x_j^{k-1}, x_{j+1}^{k-1}\right] - f\left[x_{j-1}^{k-1}, x_j^{k-1}\right]\right)^2 + \left(f\left[x_{j+1}^{k-1}, x_{j+2}^{k-1}\right] - f\left[x_j^{k-1}, x_{j+1}^{k-1}\right]\right)^2\right] \\ &+ \left(f\left[x_{j+1}^{k-1}, x_{j+2}^{k-1}\right] - 2f\left[x_j^{k-1}, x_{j+1}^{k-1}\right] + f\left[x_{j-1}^{k-1}, x_j^{k-1}\right]\right)^2 \end{aligned}$$

El esquema de subdivisión será

$$\begin{cases} \text{Do} \quad k = L, \ldots, 1 \\ \quad f_{2j}^k = f_{j-1}^{k-1} \\ f_{2j-1}^k = (w_{j-1}^{k-1} s_{j-1}^{k-1} + w_j^{k-1} s_j^{k-1} + w_{j+1}^{k-1} s_{j+1}^{k-1}) \end{cases} \tag{3.15}$$

3.1.5. *Técnicas de reconstrucción no lineal: La reconstrucción PPH*

En esta sección se describe un esquema de interpolación no lineal de cuarto orden dependiente de los datos, el cual está basado en una interpolación a trozos polinomial denominada PPH (ver [6]). Esta técnica de interpolación no lineal conduce a un operador de reconstrucción con varias características deseables. Primero, cada parte está construida con un stencil fijo centrado

de cuatro puntos. Segundo, la reconstrucción es tan exacta como su equivalente lineal en las regiones suaves. En tercer lugar, la exactitud se reduce cerca de las singularidades, pero no se pierde completamente como ocurre en su contraparte lineal.
A continuación se describe el operador de reconstrucción PPH, el cual denotamos como $\mathcal{I}_k^P(x, f^k)$. Al igual que con todas las otras técnicas de interpolación, dado $x \in \mathbb{R}$, tomemos j tal que

$$x \in [x_{j-1}^k, x_j^k]$$

Entonces, $\mathcal{I}_k^P(x, f^k) = \tilde{P}_j(x, f^k)$, donde $\tilde{P}_j(x, f^k)$ es un polinomio construido a partir de los datos centrados $f_{j-2}^k, f_{j-1}^k, f_j^k, f_{j+1}^k$ y tal que $\tilde{P}_j(x_{j-1}^k, f^k) = f_{j-1}^k$ y $\tilde{P}_j(x_j^k, f^k) = f_j^k$. En lo que sigue suprimiremos el superíndice k por claridad. Se considera el conjunto de puntos $f_{j-2}, f_{j-1}, f_j, f_{j+1}$. Y vamos a describir la predicción para el punto medio $f_{j-\frac{1}{2}}$. Según lo expuesto arriba,si la función no tiene ninguna singularidad de salto en el intervalo $[x_{j-2}, x_{j+1}]$ una interpolación centrada proporciona una buena aproximación. No obstante, cuando la señal muestra singularidades la aproximación pierde su exactitud. En lo siguiente se discute la modificación propuesta cuando se detecta una singularidad en $[x_j, x_{j+1}]$. Supongamos que la diferencia dividida $f[x_{j-1}, x_j, x_{j+1}]$ es mayor o igual que $f[x_{j-2}, x_{j-1}, x_j]$ en valor absoluto. Esto indica la posible presencia de una singularidad en un punto $x_d \in [x_j, x_{j+1}]$. Se considera el trozo polinomial para $[x_{j-1}, x_j]$ escrito como,

$$P_j(x) = a_0 + a_1(x - x_{j-\frac{1}{2}}) + a_2(x - x_{j-\frac{1}{2}})^2 + a_3(x - x_{j-\frac{1}{2}})^3. \tag{3.16}$$

Para un esquema lineal centrado las cuatro condiciones de interpolación en los puntos x_{j-2}, x_{j-1}, x_j y x_{j+1} son

$$\begin{cases} a_0 - a_1\frac{3}{2}h + a_2(\frac{3}{2}h)^2 - a_3(\frac{3}{2}h)^3 & = f_{j-2}, \\ a_0 - a_1\frac{1}{2}h + a_2(\frac{1}{2}h)^2 - a_3(\frac{1}{2}h)^3 & = f_{j-1}, \\ a_0 + a_1\frac{1}{2}h + a_2(\frac{1}{2}h)^2 + a_3(\frac{1}{2}h)^3 & = f_j, \\ a_0 + a_1\frac{3}{2}h + a_2(\frac{3}{2}h)^2 + a_3(\frac{3}{2}h)^3 & = f_{j+1}. \end{cases} \tag{3.17}$$

Es fácil comprobar que

$$a_1 = \frac{f_{j-2} - 27f_{j-1} + 27f_j - f_{j+1}}{24h}.$$

De ese modo, el sistema de ecuaciones anterior es equivalente a

$$\begin{cases} a_0 - a_1\frac{3}{2}h + a_2(\frac{3}{2}h)^2 - a_3(\frac{3}{2}h)^3 & = f_{j-2}, \\ a_0 - a_1\frac{1}{2}h + a_2(\frac{1}{2}h)^2 - a_3(\frac{1}{2}h)^3 & = f_{j-1}, \\ a_0 + a_1\frac{1}{2}h + a_2(\frac{1}{2}h)^2 + a_3(\frac{1}{2}h)^3 & = f_j, \\ a_1 & = \frac{f_{j-2}-27f_{j-1}+27f_j-f_{j+1}}{24h}. \end{cases}$$

Se introducen las diferencias divididas definidas por $e_{j-\frac{3}{2}} = f[x_{j-2}, x_{j-1}]$, $e_{j-\frac{1}{2}} = f[x_{j-1}, x_j]$, $e_{j+\frac{1}{2}} = f[x_j, x_{j+1}]$, $D_{j-1} = f[x_{j-2}, x_{j-1}, x_j]$ y $D_j = f[x_{j-1}, x_j, x_{j+1}]$. Después de algunas manipulaciones algebráicas se llega fácilmente a

$$a_1 = \frac{-e_{j-\frac{3}{2}} + 13e_{j-\frac{1}{2}}}{12} - \frac{1}{12}\frac{D_{j-1} + D_j}{2}h.$$

En particular, se observa que en presencia de una discontinuidad de salto en $[x_j, x_{j+1}]$, $a_1 = O(\frac{1}{h})$, ya que $D_j = O(\frac{1}{h^2})$. Este comportamiento es debido a la mala aproximación de la reconstrucción en presencia de discontinuidades. Destacando que D_{j-1} sigue siendo de orden $O(1)$, se sustituye la media aritmética

$$\frac{D_{j-1}+D_j}{2}$$

por la media armónica

$$\frac{2D_{j-1}D_j}{D_{j-1}+D_j}$$

siempre que $D_{j-1}D_j > 0$. Se obtiene así la siguiente expresión modificada para a_1,

$$\tilde{a}_1 := \frac{-e_{j-\frac{3}{2}} + 13e_{j-\frac{1}{2}}}{12} - \frac{1}{12}\frac{2D_{j-1}D_j}{D_{j-1} + D_j}h.$$

Por un lado, debido al hecho que

$$|2\frac{D_{j-1}D_j}{D_{j-1} + D_j}| \leq 2\,\text{mín}(|D_{j-1}|, |D_j|) = O(1), \tag{3.18}$$

asumiendo $D_{j-1}D_j > 0$, la media armónica está bien adaptada a la presencia de singularidades porque, cuando $|D_{j-1}|$ es $O(1)$ y $|D_j|$ es $O(\frac{1}{h^2})$, la media armónica permanece siendo $O(1)$, y en consecuencia, $\tilde{a}_1 = O(1)$. Por otro lado, en las regiones suaves $a_1 - \tilde{a}_1 = O(h^3)$, ya que la diferencia entre la media armónica y la aritmética original es $O(h^2)$. Por consiguiente, la reconstrucción es de cuarto orden, y en particular $(f_{j-\frac{1}{2}} - \tilde{P}_j(x_{j-\frac{1}{2}}) = O(h^4))$. Si $D_{j-1}D_j \leq 0$ la media armónica no está bajo control, ya que en algunos casos $D_{j-1} + D_j \approx 0$. Se considera por lo tanto en esta situación

$$\tilde{\tilde{a}}_1 := \frac{-e_{j-\frac{3}{2}} + 13e_{j-\frac{1}{2}}}{12}. \tag{3.19}$$

Entonces se tiene $a_1 - \tilde{\tilde{a}}_1 = O(h)$. La reconstrucción está adaptada en este caso a la presencia de singularidades aunque la exactitud se reduce hasta grado dos. El operador de reconstrucción PPH estará dado por la expresión polinomial de la ecuación (8) con los nuevos coeficientes $\tilde{a_0}$, $\tilde{a}_1$, $\tilde{a}_2$ y $\tilde{a}_3$ si $D_{j-1}D_j > 0$, o con $\tilde{\tilde{a}}_0$, $\tilde{\tilde{a}}_1$, $\tilde{\tilde{a}}_2$ y $\tilde{\tilde{a}}_3$ si $D_{j-1}D_j \leq 0$. Es fácil comprobar que la predicción se convierte en

$$\begin{aligned} f_{j-\frac{1}{2}} &\approx \frac{-f_{j-2}+18f_{j-1}-9f_j}{8} - \frac{1}{8}12\tilde{a}_1 h \\ &= \frac{f_j+f_{j-1}}{2} - \frac{1}{4}\frac{(f_j-2f_{j-1}+f_{j-2})(f_{j+1}-2f_j+f_{j-1})}{(f_{j+1}-f_j-f_{j-1}+f_{j-2})}, \end{aligned} \tag{3.20}$$

ó

$$\begin{aligned} f_{j-\frac{1}{2}} &\approx \frac{-f_{j-2}+18f_{j-1}-9f_j}{8} - \frac{1}{8}12\tilde{\tilde{a}}_1 h \\ &= \frac{f_j+f_{j-1}}{2}, \end{aligned} \tag{3.21}$$

respectivamente. Por razones de simetría la modificación es la misma cuando la singularidad pertenece a $[x_{j-2}, x_{j-1}]$. En este método, la no linealidad aparece en el proceso de selección entre una interpolación de Lagrange y una interpolación no lineal así como en la propia interpolación. Al contrario que en la interpolación ENO, la reconstrucción PPH siempre usa un stencil centrado. La meta de los operadores de reconstrucción no lineales es mejorar la exactitud de la predicción en los alrededores de las singularidades aisladas. Así se espera un mejor tratamiento de las singularidades correspondientes a los ejes de las imágenes.

El algoritmo de multirresolución queda,

$$\begin{cases} \text{Do} \quad k = L, \ldots, 1 \\ \quad f_j^{k-1} = f_{2j}^k \\ \quad d_j^k = \begin{cases} f_{2j-1}^k - (\frac{f_{j-1}^k + f_j^k}{2} - \frac{1}{4}\frac{Df_{j-1}^k Df_j^k}{Df_{j-1}^k + Df_j^k}), & si\ Df_{j-1}^k Df_j^k > 0 \\ f_{2j-1}^k - (\frac{f_{j-1}^k + f_j^k}{2}), & en\ otro\ caso \end{cases} \end{cases} \tag{3.22}$$

$$\begin{cases} \text{Do} \quad k = 1, \ldots, L \\ \quad f_{2j}^k = f_j^{k-1} \\ f_{2j-1}^k = \begin{cases} d_j^k + (\frac{f_{j-1}^k + f_j^k}{2} - \frac{1}{4}\frac{Df_{j-1}^k Df_j^k}{Df_{j-1}^k + Df_j^k}), & si\ Df_{j-1}^k Df_j^k > 0 \\ d_j^k + (\frac{f_{j-1}^k + f_j^k}{2}), & en\ otro\ caso \end{cases} \end{cases} \tag{3.23}$$

donde $Df_{j-1}^k = f_{j-2}^k - 2f_{j-1}^k + f_j^k$ y $Df_j^k = f_{j-1}^k - 2f_j^k + f_{j+1}^k$.

El correspondiente esquema de subdivisión es entonces:

$$\begin{cases} \text{Do} \quad k = 1, \ldots, L \\ \quad f_{2j}^k = f_j^k \\ f_{2j-1}^k = \begin{cases} (\frac{f_{j-1}^k + f_j^k}{2} - \frac{1}{4}\frac{Df_{j-1}^k Df_j^k}{Df_{j-1}^k + Df_j^k}), & si\ Df_{j-1}^k Df_j^k > 0 \\ \frac{f_{j-1}^k + f_j^k}{2}, & en\ otro\ caso \end{cases} \end{cases} \tag{3.24}$$

3.1.6. *Zoom de imágenes digitales mediante Producto Tensor*

Para generalizar los algoritmos de zoom anteriormente definidos para secuencias de datos dos dimensionales tales como las imágenes, se ha usado la estrategia del producto tensor, la cual presentamos brevemente a continuación. Representando el array bidimensional $f = (f_{i,j}^0)_{(i,j)=0}^{J_0}$ por la matriz $A = A^0$, la representación de un nivel de zoom de la imagen A^0 puede observarse en la Figura 3.3, donde en primer lugar se realiza un proceso de zoom en una dimensión a todas las filas y posteriormente a todas las columnas.

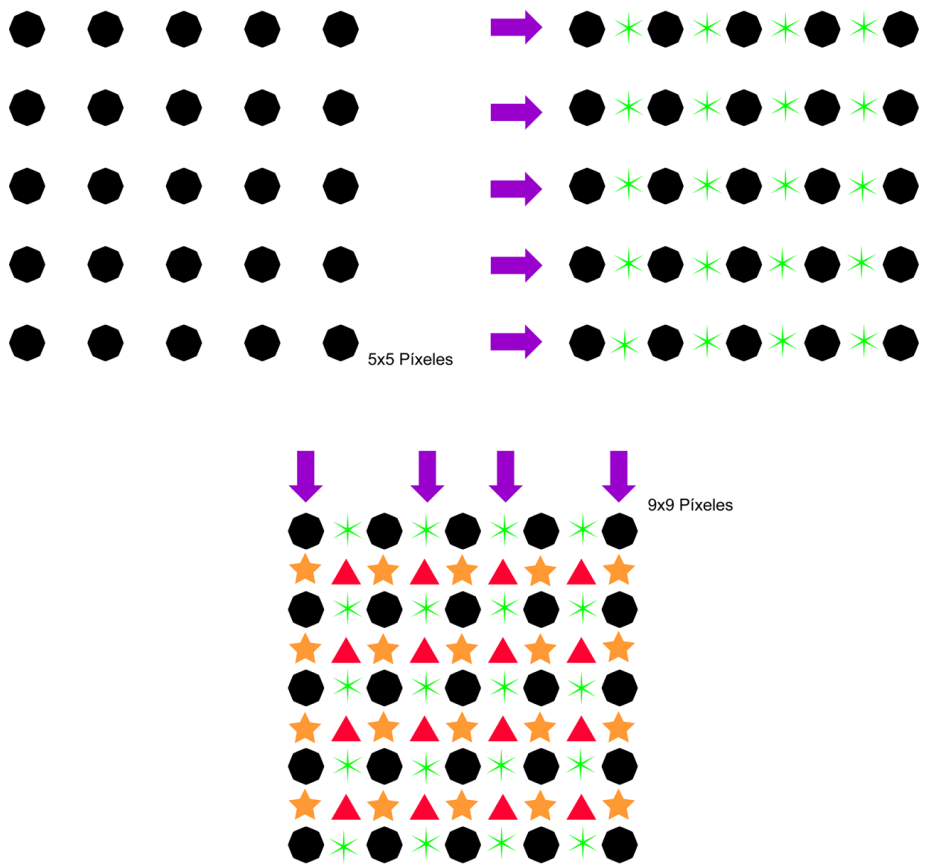

Figura 3.3: Pasos del algoritmo de zoom: imagen original, algoritmo 1D aplicado a las filas y algoritmo 1D aplicado a las columnas.

Capítulo 4

Experimentos realizados sobre imágenes digitales empleando algoritmos lineales y no lineales

En esta sección vamos a comentar los dos experimentos realizados en el proyecto. Ambos tienen una interfaz gráfica para su propósito, además de la posible ejecución del programa directamente desde la ventana de comandos del programa Matlab. Se explicará también como se realiza la ejecución de las GUI's, sin necesidad de Matlab. Se expondrán los comandos pertinentes para conseguir dicho objetivo. Daremos también una pequeña explicación teórica sobre cada uno de los experimentos. Haremos algunos ejemplos con imágenes reales y extraeremos conclusiones.

4.1. *Desarrollo del experimento 1: Ejecución y ejemplos*

En el primer experimento que trata el presente proyecto, realizamos el zoom de una imagen digital mediante el uso de esquemas de subdivisión. Hacemos uso de diferentes operadores de reconstrucción y por tanto de predicción en la definición de los esquemas de subdivisión, los cuales serán aplicados para agrandar una zona deseada de una imagen digital permitien-

do así resaltar los detalles que a simple vista no percibimos.

Como ya se comento en el capítulo anterior, un nivel del proceso de subdivisión que se aplica a la matriz (imágen o fotografía) original para hacer zoom consiste en aplicar primero un algoritmo de predicción en una dimensión a cada una de las filas de la imagen para así doblar el número de puntos y después realizar la misma operación por columnas. Este proceso puede verse de manera gráfica en la Figura 3.3.

4.1.1. Ejecución de los programas Matlab

Para la ejecución de la experimento 1, se han creado tres caminos diferentes:

1. Mediante la ejecución de una interfaz gráfica con la necesidad de tener instalado el programa Matlab.

   Para ejecutar la interfaz gráfica, tenemos que introducir en el intérprete de comandos de matlab

   $$>> GUIdemo$$

   Antes de ejecutar dicha instrucción tenemos que situarnos en el directorio donde está contenida la GUI. Una vez hecho esto, se nos cargará la interfaz de la Figura 4.1.

   El primer paso será cargar la imagen, bien escribiendo en la casilla correspondiente, o bien, desde la barra de menu escogiendo *Archivo*, *Abrir imagen*.

   Una vez que ya está cargada debemos ver en qué zona queremos hacer zoom (nótese que al abrir la imagen nos muestra sus dimensiones, así como un mallado para que nos sirvan de ayuda). La manera de escribir la zona de zoom $[x1, x2] \times [y1, y2]$ es

   $$x1\ x2\ y1\ y2$$

   Si por algún motivo, se introduce incorrectamente, el programa nos avisará del error(ver Figura 4.3).

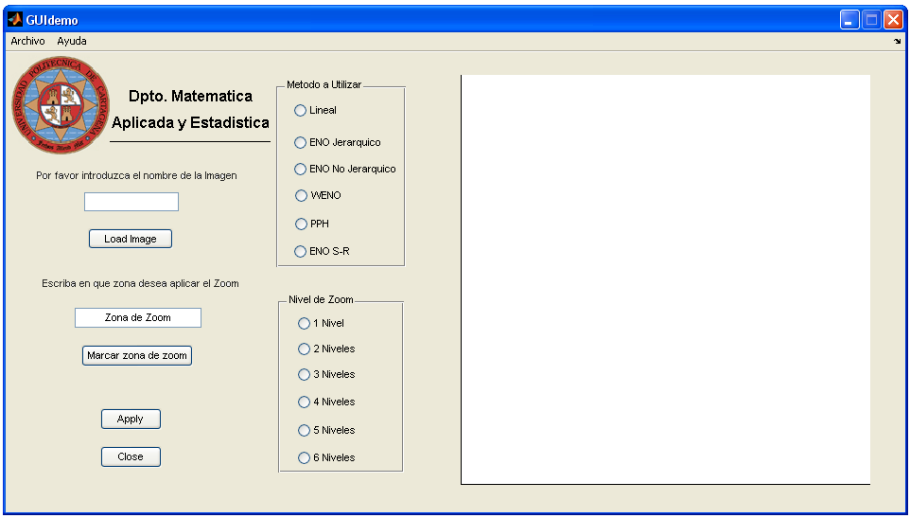

Figura 4.1: Interfaz gráfica de usuario para el experimento 1, sin introducir ningún dato.

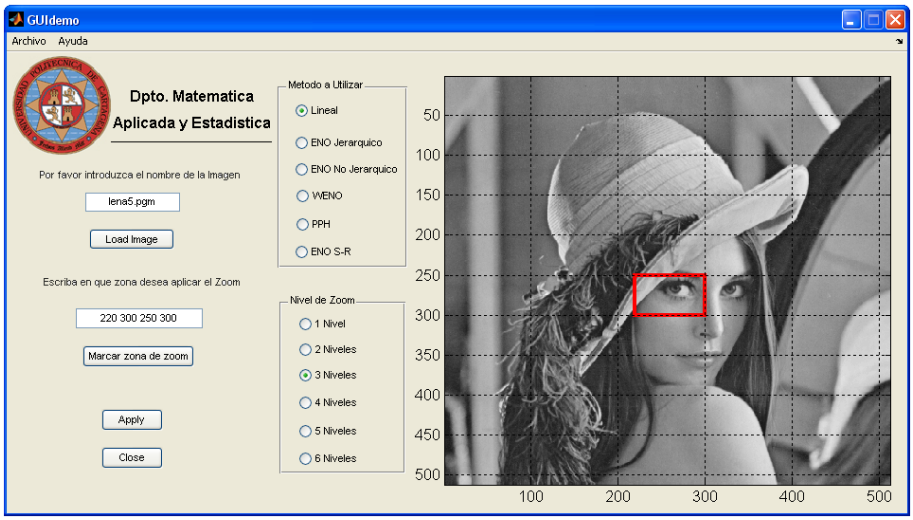

Figura 4.2: Interfaz gráfica de usuario para el experimento 1, una vez introducidos todos los datos necesarios.

Despúes de seleccionar la zona de zoom, podemos hacer 'click' con el botón izquierdo del ratón en *Marcar zona de zoom* y ver cual es la zona que hemos seleccionado. Ya sólo falta elegir el método y los

niveles de zoom. (ver Figura 4.2).

Una vez completado todo el proceso, hacer 'click' en el botón *Apply*.

El programa está diseñado para evitar cualquier error que el usuario pueda cometer a la hora de su utilización mostrando por pantalla ventanas emergentes que nos indican como subsanarlo (ver Figura 4.3).

Figura 4.3: Posibles mensajes de errores para la interfaz gráfica del experimento 1.

2. Mediante la ejecución de una interfaz gráfica sin la necesidad de tener instalado el programa Matlab.

   Para la ejecución de la interfaz gráfica sin el programa Matlab, primero debemos de instalar el *Matlab Component Runtime (MCR)* en el PC sin Matlab en alguna carpeta, por ejemplo en C:/Carpeta_de_MCR. A continuación, hay que crear una variable de entorno de usuario que se llame Path y su valor debe ser

   $$C : /Carpeta\_de\_MCR/v70/runtime/win32$$

   Eso se hace en Propiedades de Mi PC ('click' derecho en *Mi PC* y luego 'click' en *Propiedades*). A continuación un 'click' en *Opciones Avanzadas* y un 'click' en *Variables de entorno*. Después se agrega una nueva variable de entorno de usuario (ver Figura 4.4). En las variables de sistema ya hay una que se llama Path, pero no hay que tocarla.

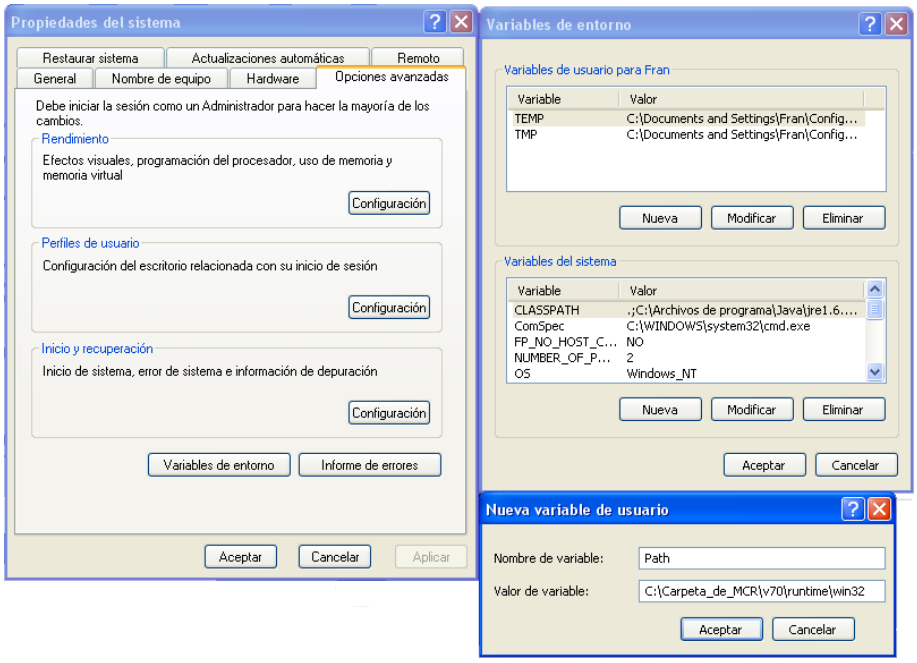

Figura 4.4: Explicación de cómo agregar la variable de sistema Path.

Una vez realizado el paso anterior, ya podemos ejecutar la interfaz gráfica de usuario, en el icono *GUIdemo.exe*.

3. Mediante la ejecución en el intérprete de comandos del programa Matlab.

Para la debida ejecución en la consola de comandos matlab, antes de nada vamos a considerar la siguiente función:

$$function \quad [a, c] = zoom\_tensor\_exp1(l, im, met, v),$$

donde l indica los niveles de zoom, im es la imagen con la que vamos a trabajar, met representa el método de predicción a utilizar (ver capítulo 3), y v es el vector donde introducimos las coordenadas de la zona ($[x1, x2, y1, y2]$) que se quiere ampliar. El programa nos devuelve la imagen original a y el zoom de la imagen c. En la Figura 4.5 se ve un ejemplo de cómo se escribiría en la línea de comandos.

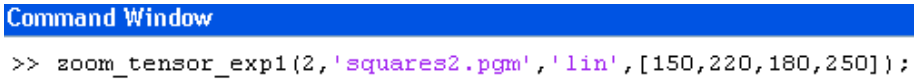

Figura 4.5: Intérprete de comandos Matlab.

4.1.2. Ejemplos de zoom empleando esquemas de subdivisión

A continuación mostramos una relación de ejemplos para mostrar los resultados del experimento 1. Vamos a trabajar con las cinco imágenes digitales de la Figura 4.6: la primera de ellas es una sencilla imagen geométrica 'squares2' con cuatro tonalidades de gris, la segunda 'geo5' es una imagen también geométrica con diferentes formas, y por último tres imágenes reales 'camera2', 'lena5' y 'crowd5'.

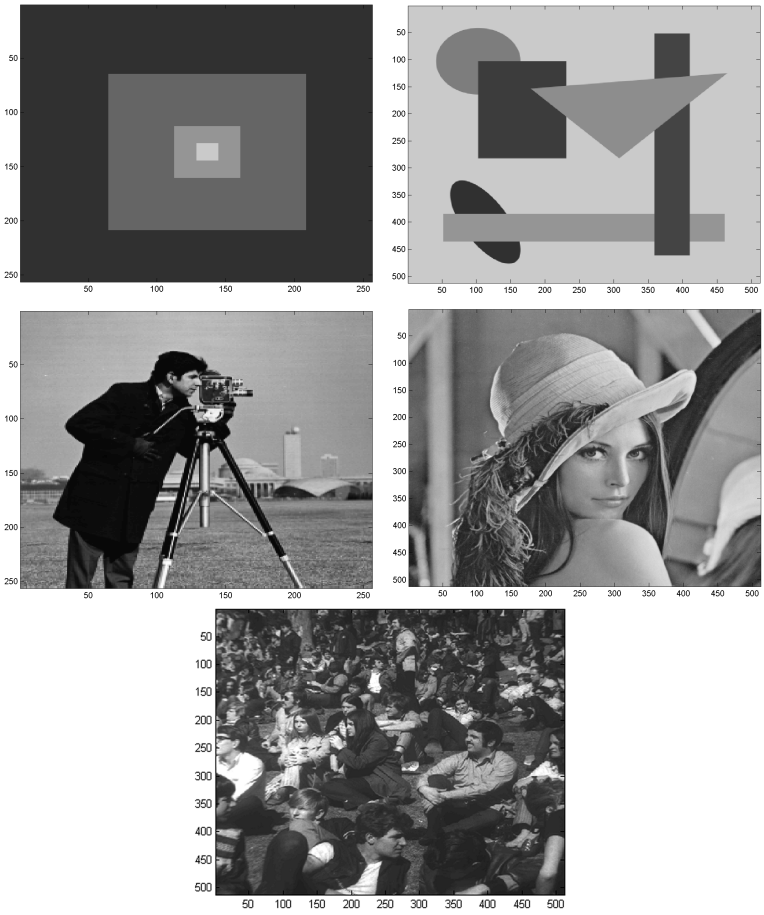

Figura 4.6: Imágenes utililizadas para el experimento 1.

La primera imágen 'squares2.pbm' como se ha descrito anteriormente es una imagen geométrica, con cuatro tonalidades de gris. Hacemos zoom en la zona $x_1 = 125$, $x_2 = 150$, $y_1 = 125$ e $y_2 = 150$, con 4 niveles, como se muesta en la Figura 4.7. El zoom aplicando distintos métodos se muestra en las Figuras 4.8, 4.9, 4.10, 4.11, 4.12, 4.13.

En la Figura 4.8 vemos el efecto de un zoom lineal en los ejes de la imagen. Se produce un efecto difusivo que aumenta con el orden de la interpolación utilizada. Para el caso que nos ocupa, interpolación de cuarto orden, hay tres intervalos afectados y eso se puede observar en la Figura 4.8 donde tres líneas de píxeles alrededor de los ejes están afectados del efecto difusivo.

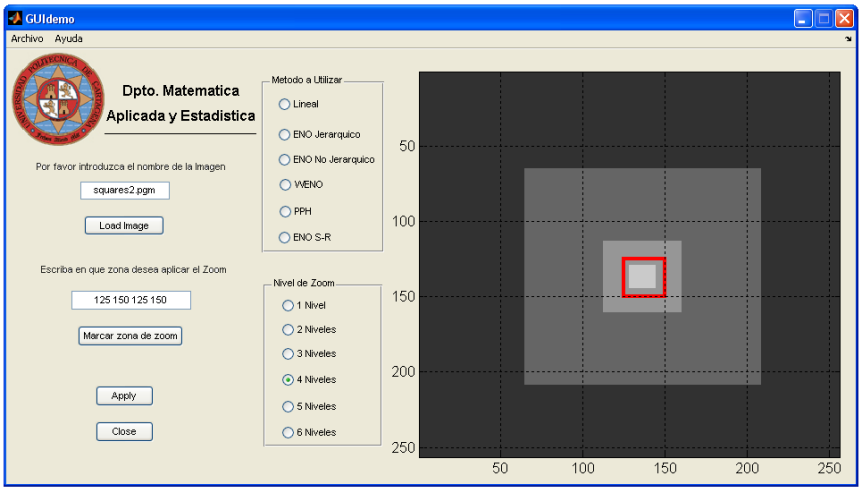

Figura 4.7: Interfaz gráfica con 'squares2.pgm'.

En las Figuras 4.9, 4.10, 4.11, 4.12, 4.13 se ve el resultado de aplicar métodos no lineales. Vemos como el efecto de la difusión se reduce en este caso a un único intervalo.

En la Figura 4.13 observamos el método ENO-SR el cúal es adecuado para discontinuidades en derivada. Como en imágenes tratamos en gran medida con discontinuidades de salto, este método no logra sus mayores potenciales cuando trabajamos en el entorno de valores puntuales.

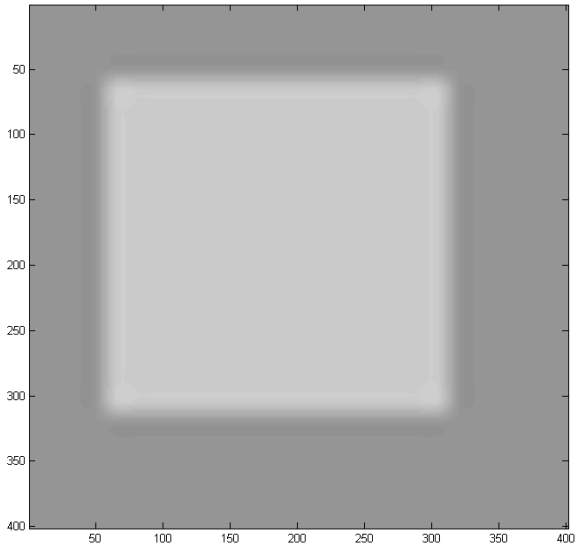

Figura 4.8: Zoom de 'squares2.pgm' en la zona [$x_1 = 125$, $x_2 = 150$, $y_1 = 125$, $y_2 = 150$] aplicando el método Lineal con 4 niveles.

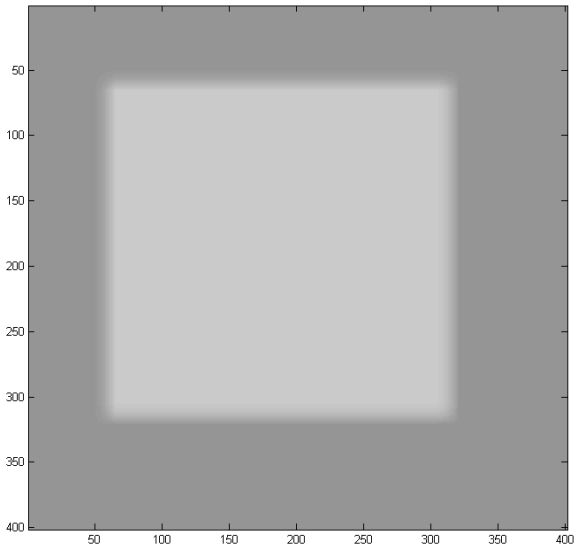

Figura 4.9: Zoom de 'squares2.pgm' en la zona [$x_1 = 125$, $x_2 = 150$, $y_1 = 125$, $y_2 = 150$] aplicando el método ENO jerárquico con 4 niveles.

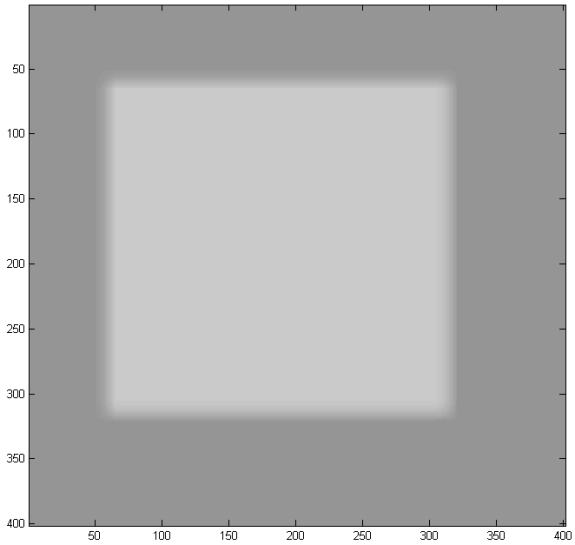

Figura 4.10: Zoom de 'squares2.pgm' en la zona [$x_1 = 125$, $x_2 = 150$, $y_1 = 125$, $y_2 = 150$] aplicando el método ENO no jerárquico con 4 niveles.

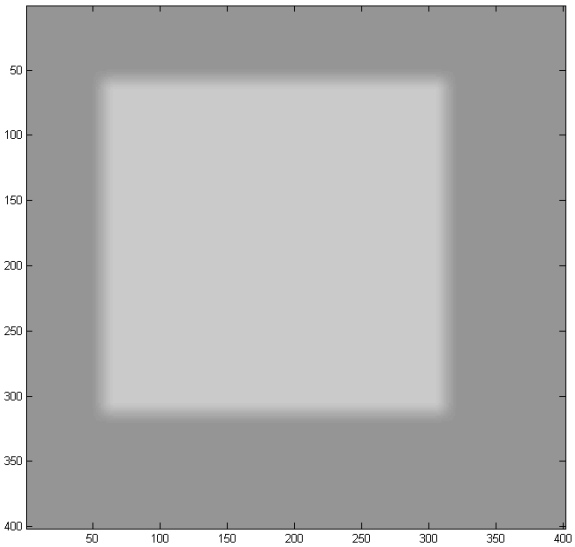

Figura 4.11: Zoom de 'squares2.pgm' en la zona [$x_1 = 125$, $x_2 = 150$, $y_1 = 125$, $y_2 = 150$] aplicando el método WENO con 4 niveles.

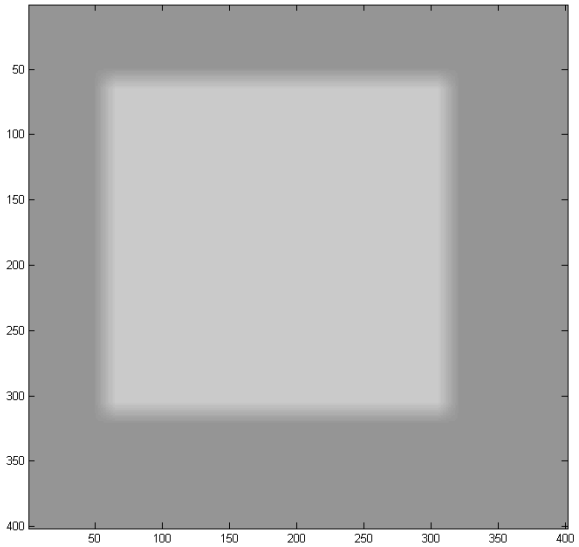

Figura 4.12: Zoom de 'squares2.pgm' en la zona [$x_1 = 125$, $x_2 = 150$, $y_1 = 125$, $y_2 = 150$] aplicando el método PPH con 4 niveles.

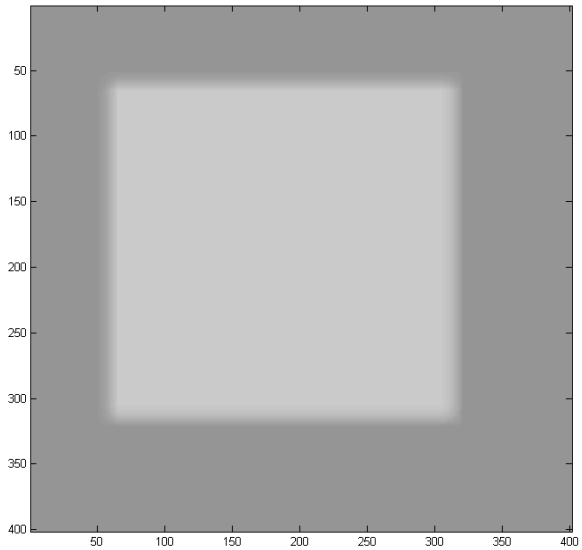

Figura 4.13: Zoom de 'squares2.pgm' en la zona [$x_1 = 125$, $x_2 = 150$, $y_1 = 125$, $y_2 = 150$] aplicando el método ENO-SR con 4 niveles.

La siguiente imágen 'geo5.pbm' como se ha descrito anteriormente es una imagen geométrica, con diferentes formas. Hacemos zoom en la zona $x_1 = 45$, $x_2 = 90$, $y_1 = 350$ e $y_2 = 400$, con 3 niveles, como se muesta en la Figura 4.7. El zoom aplicando distintos métodos se muestra en las Figuras 4.15, 4.16, 4.17, 4.18, 4.19, 4.20.

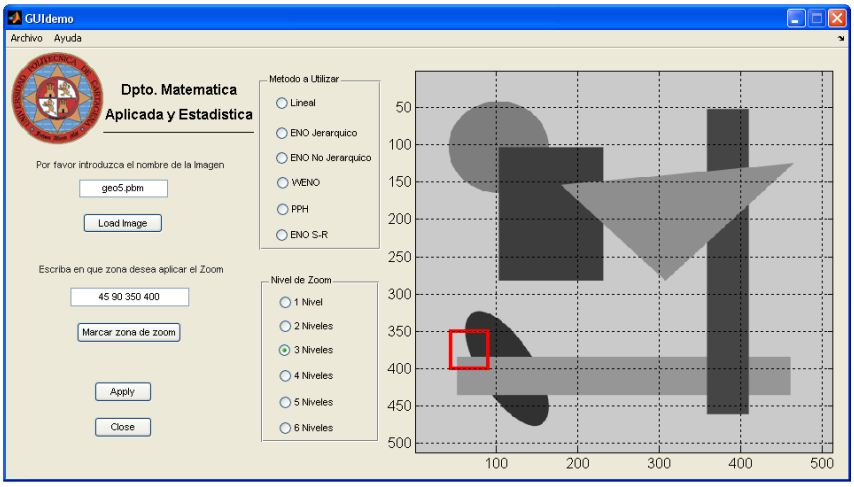

Figura 4.14: Interfaz gráfica con 'geo5.pbm'.

Podemos observar los mismos efectos comentados en el caso anterior, debido a que la imagen es geométrica. Además la resolución en las zonas curvas no es tan buena en este tipo de zoom basado en la aplicación primero a filas y después a columnas del algoritmo uno dimensional.

En la Figura 4.15 volvemos a tener el efecto de Gibbs y la difusión típica de los esquemas lineales. Estos efectos indeseables se reducen con los métodos no lineales.

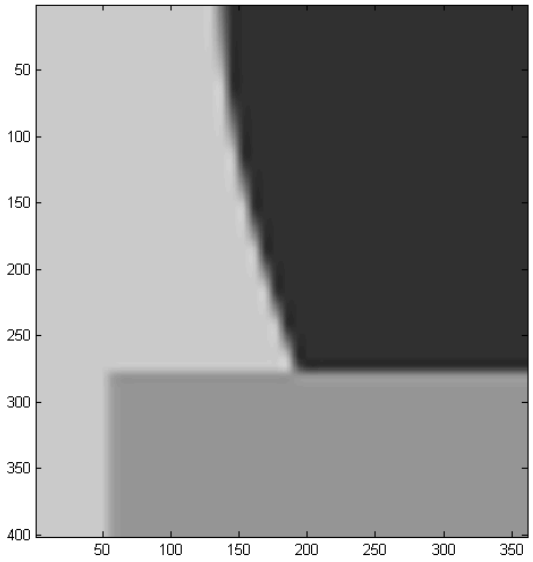

Figura 4.15: Zoom de 'geo5.pbm' en la zona [$x_1 = 45$, $x_2 = 90$, $y_1 = 350$, $y_2 = 400$] aplicando el método Lineal con 3 niveles.

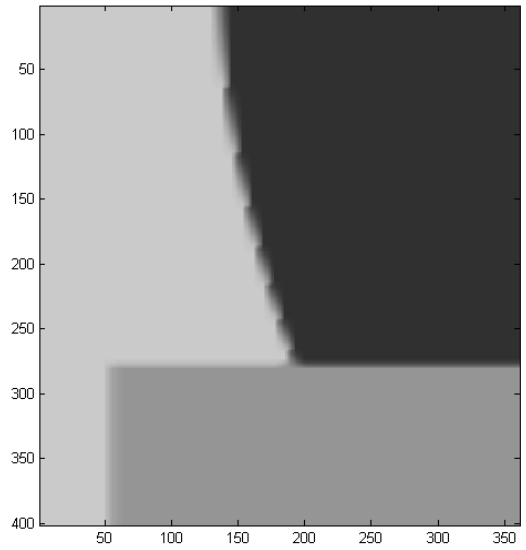

Figura 4.16: Zoom de 'geo5.pbm' en la zona [$x_1 = 45$, $x_2 = 90$, $y_1 = 350$, $y_2 = 400$] aplicando el método ENO jerárquico con 3 niveles.

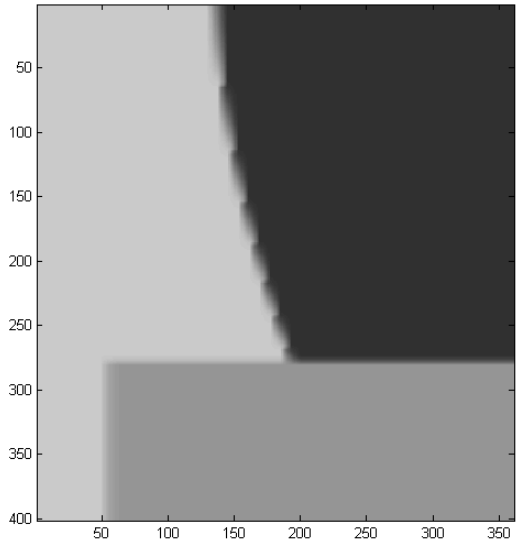

Figura 4.17: Zoom de 'geo5.pbm' en la zona [$x_1 = 45$, $x_2 = 90$, $y_1 = 350$, $y_2 = 400$] aplicando el método ENO no jerárquico con 3 niveles.

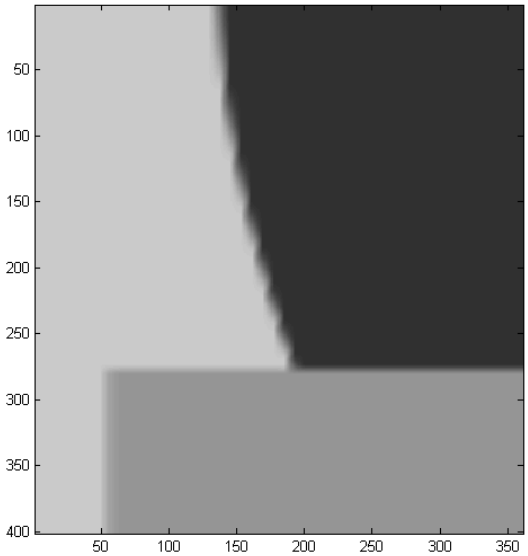

Figura 4.18: Zoom de 'geo5.pbm' en la zona [$x_1 = 45$, $x_2 = 90$, $y_1 = 350$, $y_2 = 400$] aplicando el método WENO con 3 niveles.

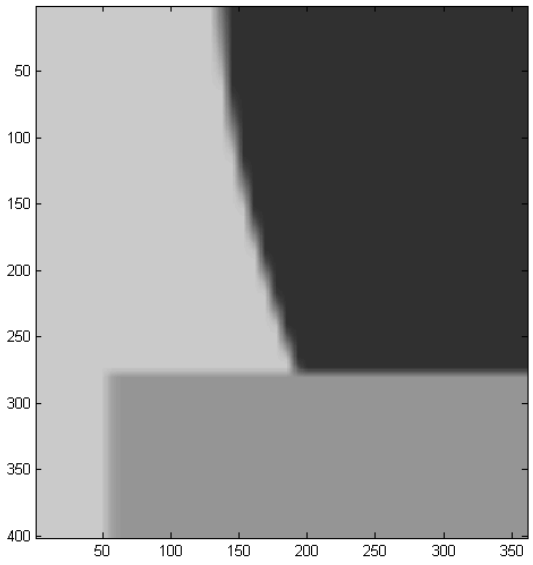

Figura 4.19: Zoom de 'geo5.pbm' en la zona [$x_1 = 45$, $x_2 = 90$, $y_1 = 350$, $y_2 = 400$] aplicando el método PPH con 3 niveles.

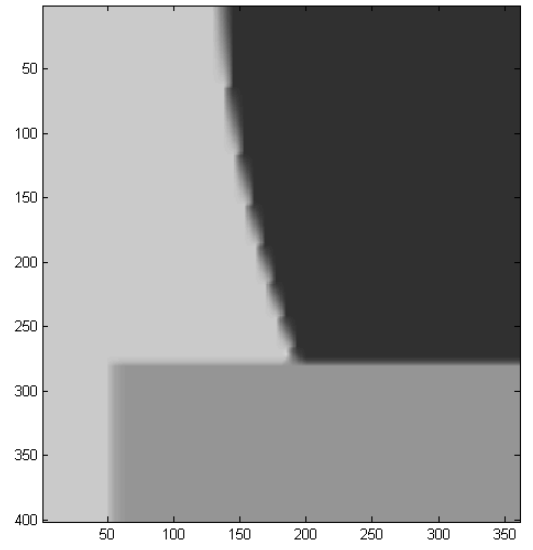

Figura 4.20: Zoom de 'geo5.pbm' en la zona [$x_1 = 45$, $x_2 = 90$, $y_1 = 350$, $y_2 = 400$] aplicando el método ENO-SR con 3 niveles.

La siguiente imagen 'camera2.pgm' corresponde a la imagen real de un cámara. Hacemos zoom en la zona $x_1 = 90$, $x_2 = 150$, $y_1 = 35$ e $y_2 = 100$, con 5 niveles, como se muesta en la Figura 4.21. El zoom aplicando distintos métodos se muestra en las Figuras 4.22, 4.23, 4.24, 4.25, 4.26, 4.27.

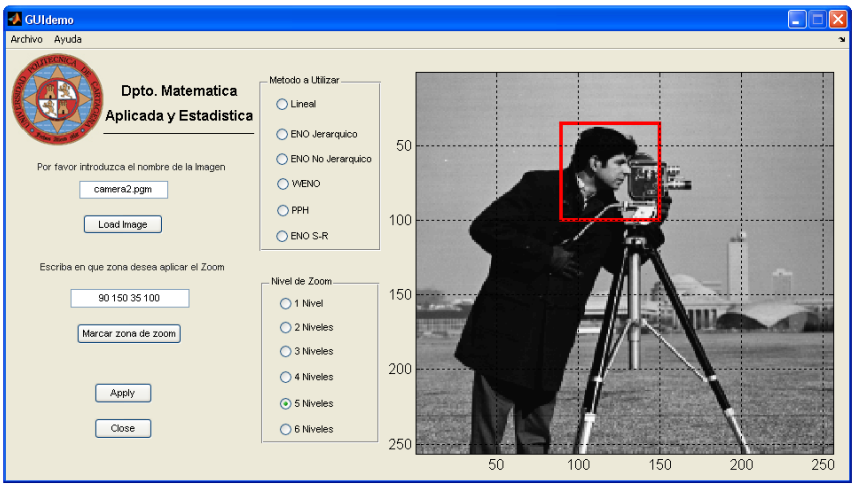

Figura 4.21: Interfaz gráfica con 'camera2.pgm'.

En este caso se trata de una imagen real, pero que contiene varios objetos con una estructura bastante geométrica. En la imágenes geométricas es donde se espera una mejora notable de los esquemas no lineales sobres sus relativos lineales ya que estas imágenes tienen los ejes bien marcados. Las diferencias visuales de los métodos quedan claras en las Figuras 4.22, 4.23, 4.24, 4.25, 4.26 y 4.27.

Figura 4.22: Zoom de 'camera2.pgm' en la zona [$x_1 = 90$, $x_2 = 150$, $y_1 = 35$, $y_2 = 100$] aplicando el método Lineal con 5 niveles.

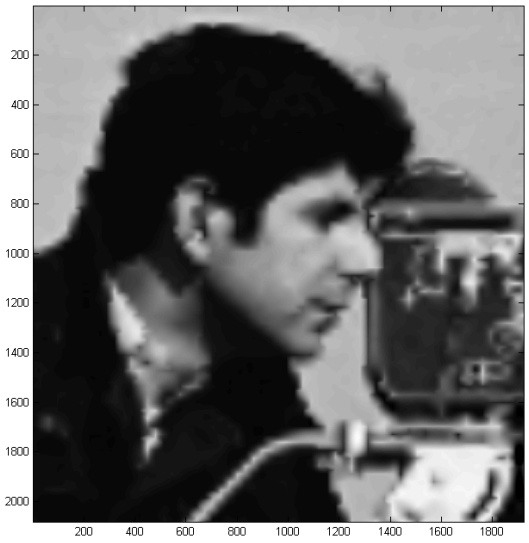

Figura 4.23: Zoom de 'camera2.pgm' en la zona [$x_1 = 90$, $x_2 = 150$, $y_1 = 35$, $y_2 = 100$] aplicando el método ENO jeráquico con 5 niveles.

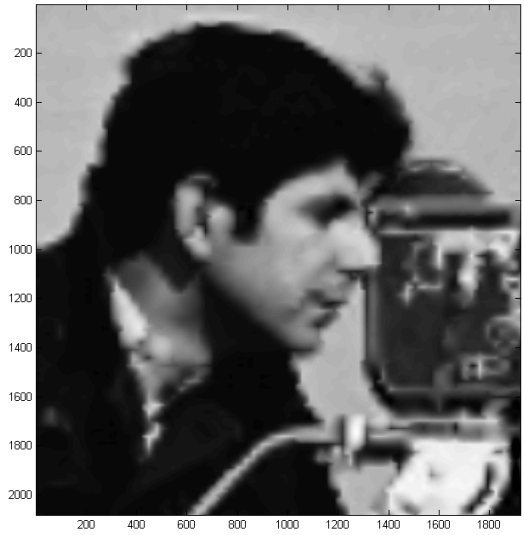

Figura 4.24: Zoom de 'camera2.pgm' en la zona [$x_1 = 90$, $x_2 = 150$, $y_1 = 35$, $y_2 = 100$] aplicando el método ENO no jerárquico con 5 niveles.

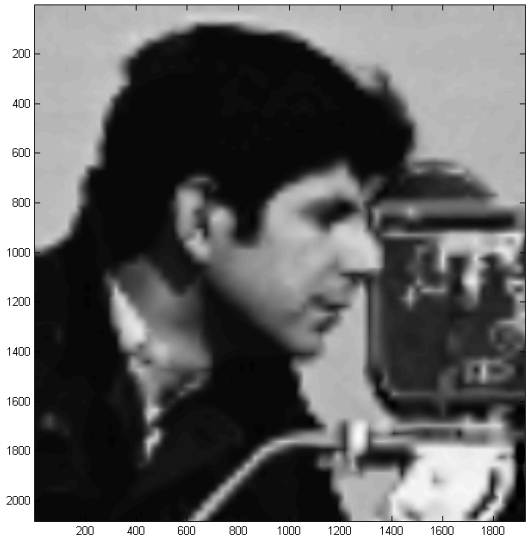

Figura 4.25: Zoom de 'camera2.pgm' en la zona [$x_1 = 90$, $x_2 = 150$, $y_1 = 35$, $y_2 = 100$] aplicando el método WENO con 5 niveles.

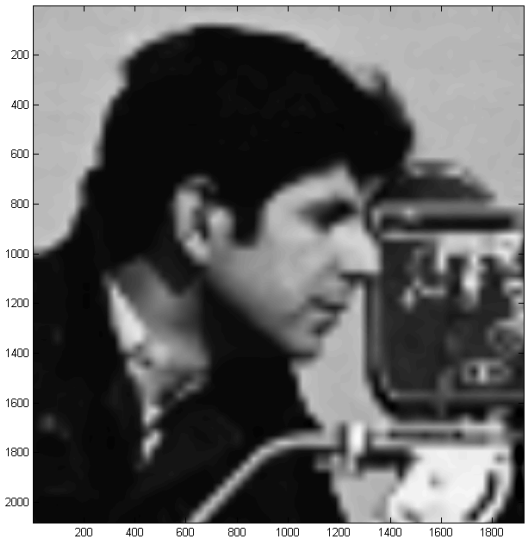

Figura 4.26: Zoom de 'camera2.pgm' en la zona [$x_1 = 90$, $x_2 = 150$, $y_1 = 35$, $y_2 = 100$] aplicando el método PPH con 5 niveles.

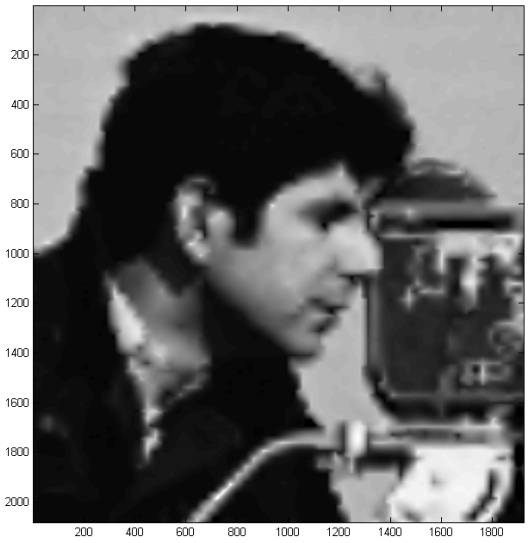

Figura 4.27: Zoom de 'camera2.pgm' en la zona [$x_1 = 90$, $x_2 = 150$, $y_1 = 35$, $y_2 = 100$] aplicando el método ENO S-R con 5 niveles.

La siguiente imagen 'lena5.pgm' es una imagen real, muy utilizada como imagen test en el mundo del procesamiento de imágenes digitales. Hacemos zoom en la zona $x_1 = 245$, $x_2 = 295$, $y_1 = 250$ e $y_2 = 285$, con 4 niveles, como se muesta en la Figura 4.28. La zona se corresponde con el ojo derecho de la modelo Lena. Aquí aparece un área con diferentes texturas que representa bien la complejidad de las imágenes reales. El zoom aplicando distintos métodos se muestra en las Figuras 4.29, 4.30, 4.31, 4.32, 4.33, 4.34.

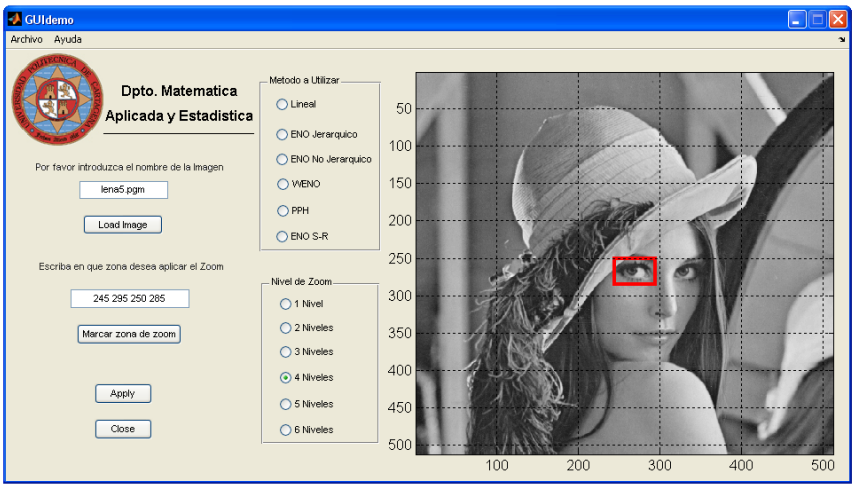

Figura 4.28: Interfaz gráfica con 'lena5.pgm'.

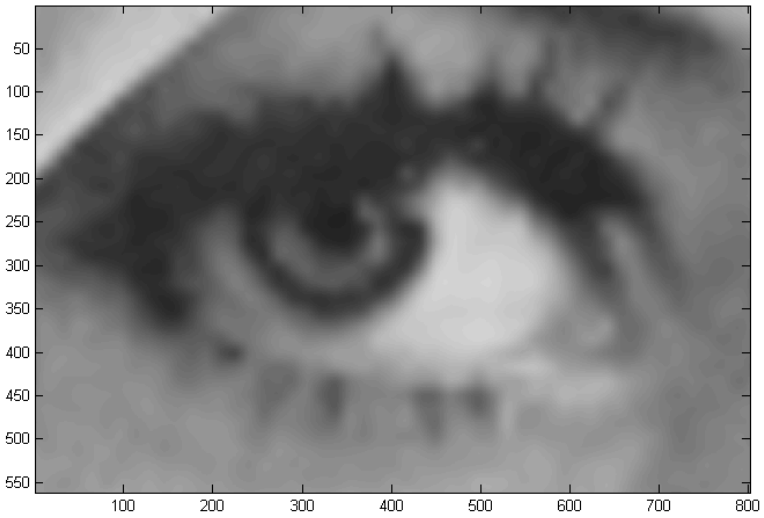

Figura 4.29: Zoom de 'lena5.pgm' en la zona [$x_1 = 245$, $x_2 = 295$, $y_1 = 250$, $y_2 = 285$] aplicando el método Lineal con 4 niveles.

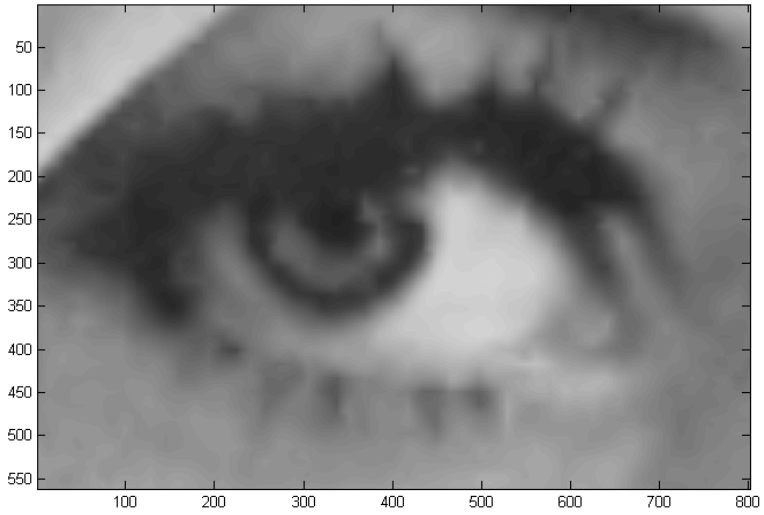

Figura 4.30: Zoom de 'lena5.pgm' en la zona [$x_1 = 245$, $x_2 = 295$, $y_1 = 250$, $y_2 = 285$] aplicando el método ENO jerárquico con 4 niveles.

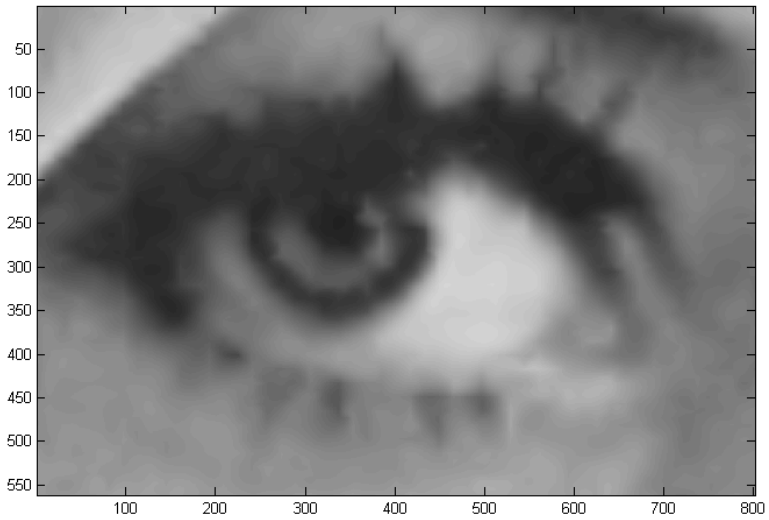

Figura 4.31: Zoom de 'lena5.pgm' en la zona [$x_1 = 245$, $x_2 = 295$, $y_1 = 250$, $y_2 = 285$] aplicando el método ENO no jerárquico con 4 niveles.

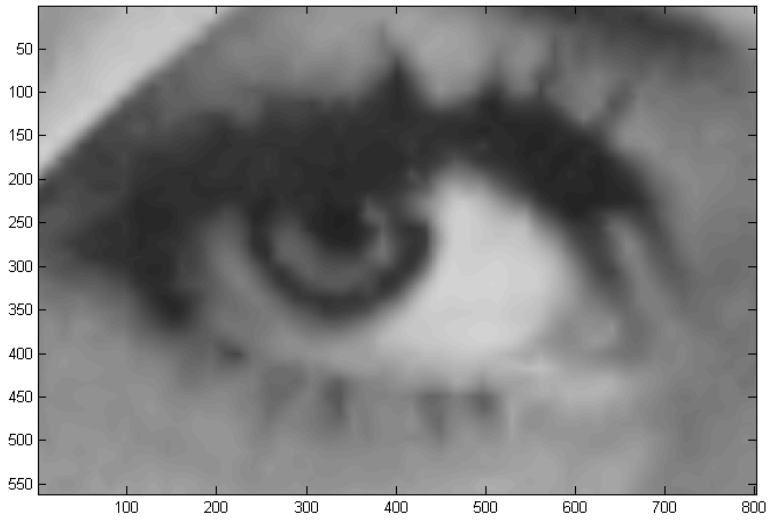

Figura 4.32: Zoom de 'lena5.pgm' en la zona [$x_1 = 245$, $x_2 = 295$, $y_1 = 250$, $y_2 = 285$] aplicando el método WENO con 4 niveles.

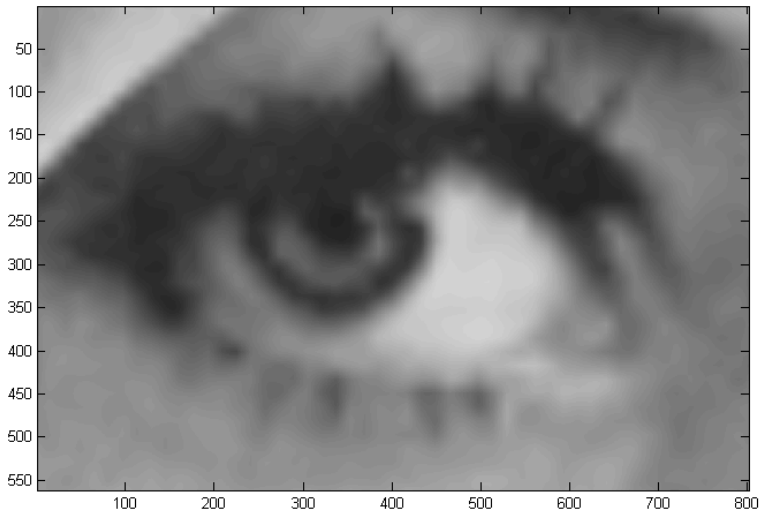

Figura 4.33: Zoom de 'lena5.pgm' en la zona [$x_1 = 245$, $x_2 = 295$, $y_1 = 250$, $y_2 = 285$] aplicando el método PPH con 4 niveles.

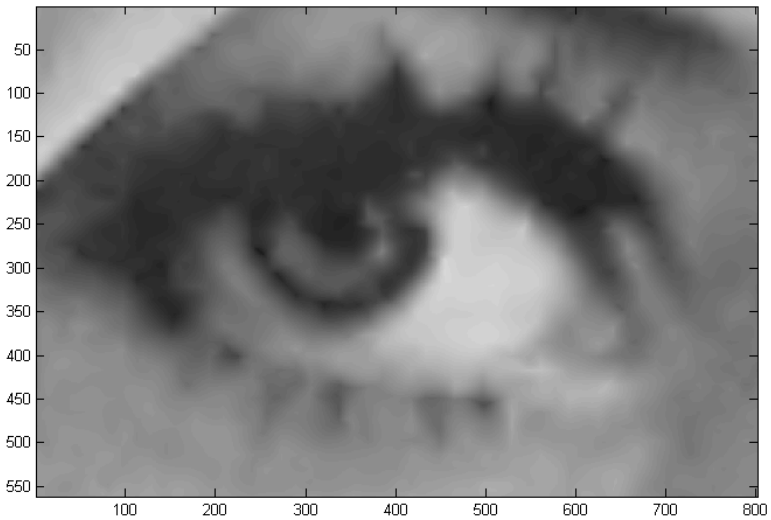

Figura 4.34: Zoom de 'lena5.pgm' en la zona [$x_1 = 245$, $x_2 = 295$, $y_1 = 250$, $y_2 = 285$] aplicando el método ENO S-R con 4 niveles.

Si comparamos todas las imágenes de las Figuras 4.29, 4.30, 4.31, 4.32, 4.33, 4.34 con la imagen de la Figura 4.35, vemos como en el zoom de Matlab "pixela"mucho la imagen. En cambio con el zoom aplicado vemos como hay un suavizado y por tanto, la calidad visual de la imagen es mucho mejor.

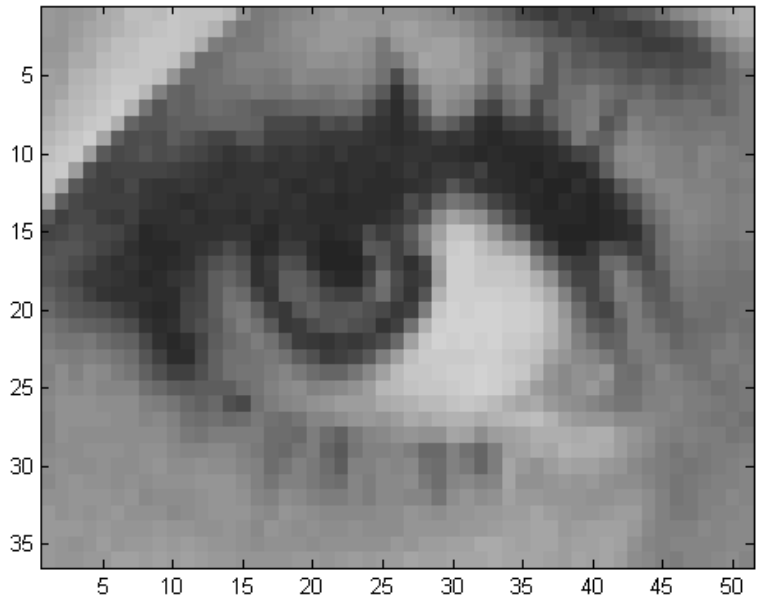

Figura 4.35: Zoom de 'lena5.pgm' en la zona [$x_1 = 245$, $x_2 = 295$, $y_1 = 250$, $y_2 = 285$] aplicando zoom de Matlab.

Las diferencias entre los métodos lineales y los no lineales son ahora menos perceptibles siendo los resultados comparables.

La siguiente y última imágen 'crowd5.pgm' es una imagen real, con mayor complejidad que la imagen de Lena. Hacemos zoom en la zona $x_1 = 255$, $x_2 = 285$, $y_1 = 10$ e $y_2 = 45$, con diferentes niveles, como se muesta en la Figura 4.36. Debido que el zoom es de una zona muy pequeña, se ha decidido hacerlo con el método que nos proporciona un mejor resultado, PPH. Se aprecia como con la progresión de niveles de zoom la imagen aumenta y permite observar mejor ciertos detalles. El zoom aplicado con dicho método se muestra en las Figuras 4.37, 4.38, 4.39 y 4.40.

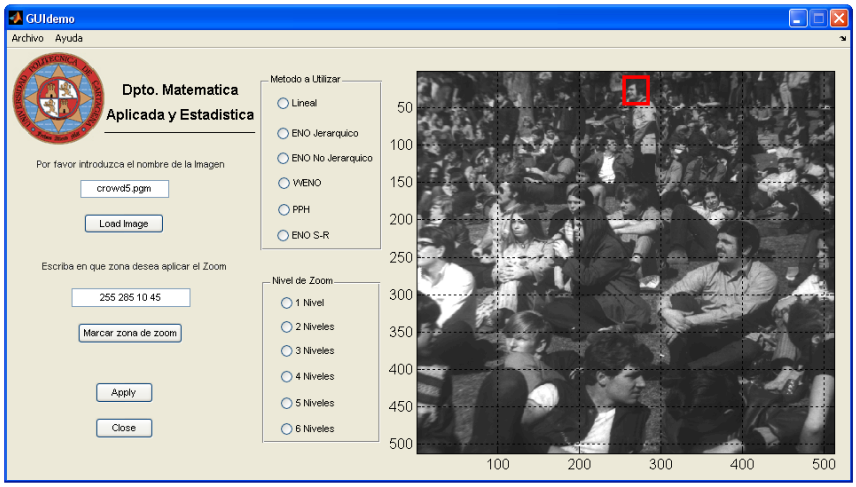

Figura 4.36: Interfaz gráfica con 'crowd5.pgm'.

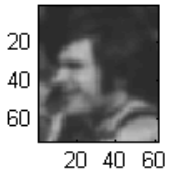

Figura 4.37: Zoom de 'crowd5.pgm' en la zona [$x_1 = 255$, $x_2 = 285$, $y_1 = 10$, $y_2 = 45$] aplicando el método PPH con 1 nivel.

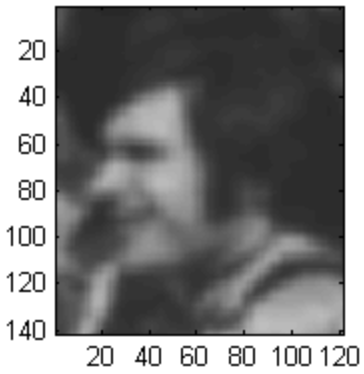

Figura 4.38: Zoom de 'crowd5.pgm' en la zona [$x_1 = 255$, $x_2 = 285$, $y_1 = 10$, $y_2 = 45$] aplicando el método PPH con 2 niveles.

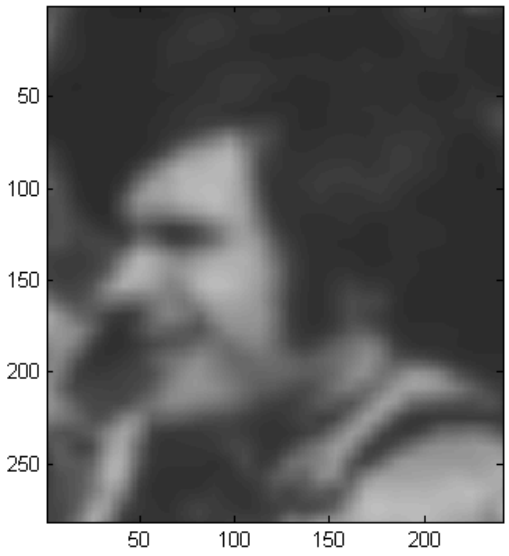

Figura 4.39: Zoom de 'crowd5.pgm' en la zona [$x_1 = 255$, $x_2 = 285$, $y_1 = 10$, $y_2 = 45$] aplicando el método PPH con 3 niveles.

Figura 4.40: Zoom de 'crowd5.pgm' en la zona [$x_1 = 255$, $x_2 = 285$, $y_1 = 10$, $y_2 = 45$] aplicando el método PPH con 4 niveles.

Es interesante comparar otra vez las imágenes de las Figuras 4.37, 4.38, 4.39 y 4.40 con la imagen de la Figura 4.41. Vemos como el zoom de Matlab "pixela"mucho la imagen, en cambio el zoom basado en esquemas de subdivisión da imágenes más nítidas.

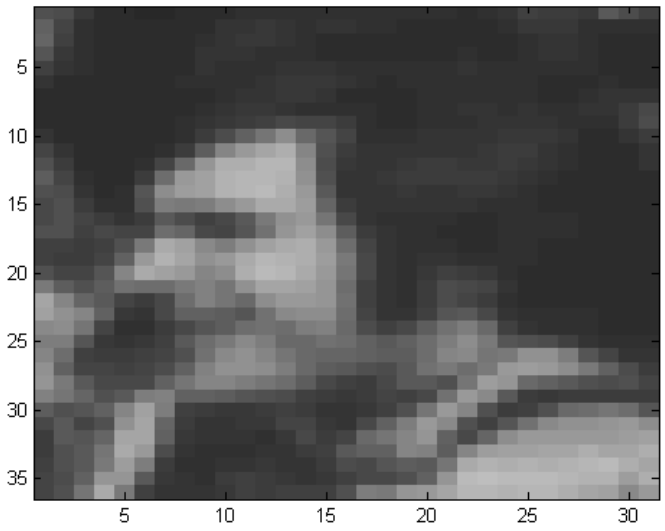

Figura 4.41: Zoom de 'crowd5.pgm' en la zona [$x_1 = 255$, $x_2 = 285$, $y_1 = 10$, $y_2 = 45$] aplicando zoom de Matlab.

4.2. *Desarrollo del experimento 2: Ejecución y ejemplos*

En el segundo experimento que trata el presente proyecto realizamos el zoom de una imagen digital mediante el uso de esquemas de subdivisión al igual que en el experimento 1. Hacemos uso de operadores de reconstrucción y de predicción para la definición de los diferentes esquemas de subdivisión, los cuales son aplicados para realizar el zoom de una imagen digital. La diferencia con el experimento 1 es que ahora vamos a partir de una imagen $A = A^L$ original y vamos a decimarla L niveles de resolución para obtener una imagen A^0 de baja resolución. Después aplicaremos los diferentes algoritmos de zoom tal cúal fueron descritos en el capítulo 3 y en el experimento 1 a la imagen A^0 para subir L niveles y obtener una aproximación $\widetilde{A}^L$ a A^L. El objetivo de esta sección es poder medir la diferencia entre A^L y $\widetilde{A}^L$, y para ello utilizaremos la medida $PSNR$. Esta medida nos da una indicación de la calidad de la aproximación $\widetilde{A}^L$. Cuanto mayor sea mejor.

Como también ya se comentó en el capítulo 1 y en el experimento 1, un nivel del proceso de subdivisión que se aplica a la matriz (imágen o fotografía) original para hacer zoom consiste en aplicar primero un algoritmo de predicción en una dimensión a cada una de las filas de la imagen para así doblar el número de puntos y despúes realizar la misma operación por columnas.

4.2.1. Ejecución de los programas Matlab

Al igual que para la ejecución del experimento 1 se han creado tres caminos diferentes:

1. La ejecución de una interfaz gráfica con la necesidad de tener instalado el programa Matlab.
   Para ejecutar la interfaz gráfica, tenemos que introducir en el intérprete de comandos de matlab

   $$>> GUIdemo$$

   Antes de ejecutar dicha instrucción tenemos que situarnos en el directorio donde está contenida la GUI. Se nos cargará la interfaz de la

Figura 4.42.

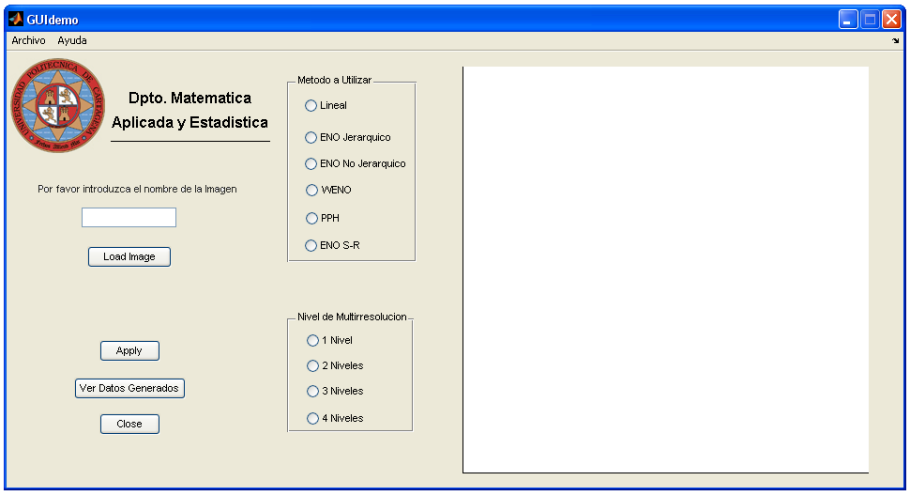

Figura 4.42: Interfaz gráfica de usuario para el experimento 2, sin haber introducido ningún dato.

El primer paso será cargar la imagen, bien escribiendo en la casilla correspondiente, o bien, desde la barra de menu *Archivo*, *Abrir imagen* (ver Figura 4.43).

Una vez que está cargada, podemos seleccionar el botón, *Ver datos generados*, el cual nos muestra en una ventana la fecha y hora de las simulaciones, acompañadas de las medidas pertinentes, tales como el $PSNR$ (ver Figura 4.44). Si por algún motivo no existe el archivo (porque no hemos realizado ninguna simulación, o porque lo hemos borrado) donde se guardan los datos de simulaciones, el programa nos avisará del error (ver Figura 4.45).

A continuación podemos elegir el método y los niveles de multirresolución (ver Figura 4.43).

Y una vez completado todo el proceso, podemos hacer 'click' en el botón, *Apply* (ver Figura 4.43).

El programa está diseñado para evitar cualquier error que el usuario pueda cometer a la hora de su utilización mostrando por pantalla ventanas emergentes que nos indican en algunos casos como subsanar el error. En la Figura 4.45, se muestran los errores que el programa puede

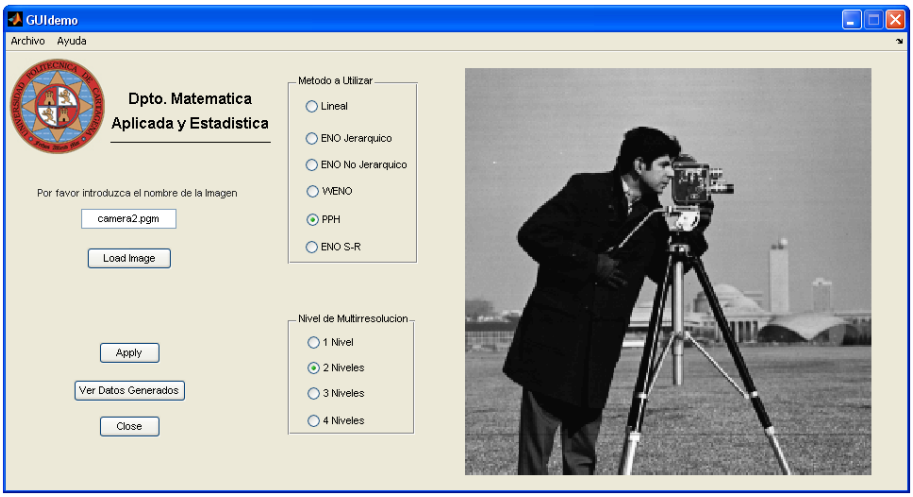

Figura 4.43: Interfaz gráfica de usuario para el experimento 2, con la imagen cargada.

desprender.

2. Mediante la ejecución de una interfaz gráfica sin la necesidad de tener instalado el programa Matlab.

Al igual que para el experimento 1, para la ejecución de la interfaz gráfica sin el programa Matlab, primero debemos instalar el *Matlab Component Runtime (MCR)* en el PC en alguna carpeta, por ejemplo en C:/Carpeta_de_MCR. A continuación, hay que crear una variable de entorno de usuario que se llame Path y su valor debe ser

$$C : /Carpeta\_de\_MCR/v70/runtime/win32$$

Eso se hace en Propiedades de Mi PC ('click' derecho en *Mi PC* y luego 'click' en *Propiedades*). A continuación un 'click' en *Opciones Avanzadas* y un 'click' en *Variables de entorno*. Después se agrega una nueva variable de entorno de usuario (ver Figura 4.4). En las variables de sistema ya hay una que se llama Path, pero no hay que tocarla.

Una vez realizado el paso anterior, ya podemos ejecutar la interfaz

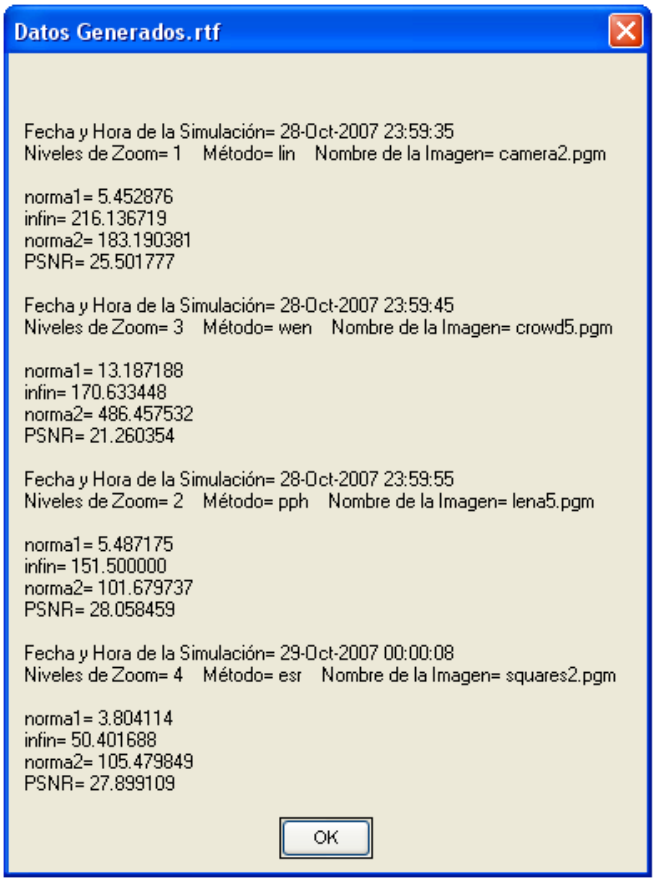

Figura 4.44: Panel de la interfaz gráfica, con fecha y hora de simulación, método seleccionado, niveles, nombre de la imagen y medidas ($norma1$, $infinito$, $norma2$ y $PSNR$).

gráfica de usuario, en el icono $GUIdemo.exe$.

3. Mediante la ejecución en el intérprete de comandos del programa Matlab.

   Para la debida ejecución en la consola de comandos matlab, antes de nada debemos considerar la siguiente función:

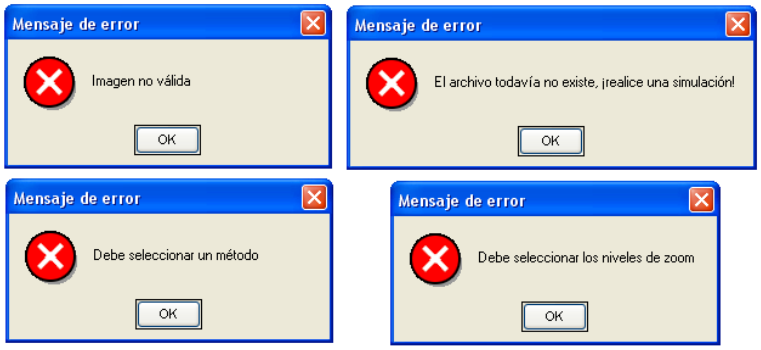

Figura 4.45: Posibles mensajes de errores para la interfaz gráfica del experimento 2.

$$function \quad [a, c] = zoom\_tensor\_exp2(l, im, met),$$

donde l indican los niveles de zoom, im es la imagen con la que vamos a trabajar y met representa el método de predicción a utilizar (ver capítulo 3). El programa nos devuelve la imagen original a y el zoom de la imagen c. En la Figura 4.46 se ve un ejemplo de cómo se escribiría en la línea de comandos.

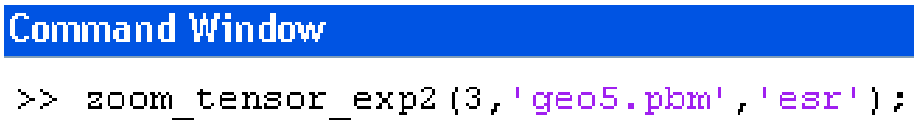

Figura 4.46: Intérprete de comandos Matlab.

4.2.2. Ejemplos de zoom empleando esquemas de subdivisión

A continuación mostramos una relación de ejemplos para mostrar los resultados del experimento 2. Vamos a trabajar con las cuatro imágenes digitales de la Figura 4.47: La primera de ellas 'squares2' es una sencilla imagen geométrica con cuatro tonalidades de gris, 'geo5' es también una imagen geométrica con la característica de incluir diferentes formas. Las dos siguientes son imágenes reales 'camera2' y 'lena5'.

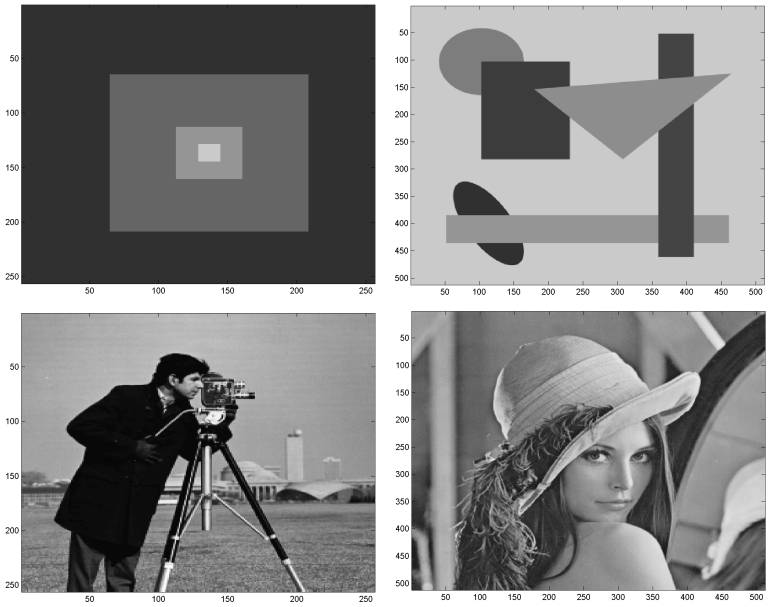

Figura 4.47: Imágenes utililizadas para el experimento 2.

A continuación vemos la imagen original de 'squares2.pgm' (ver Figura 4.48) y su imagen a baja resolución con un nivel de multirresolución (ver Figura 4.49) y con cuatro niveles de multirresolución (ver Figura 4.50). Por la simplicidad de la imagen, aunque sigamos subdividiendo niveles, la imagen nunca perderá calidad.

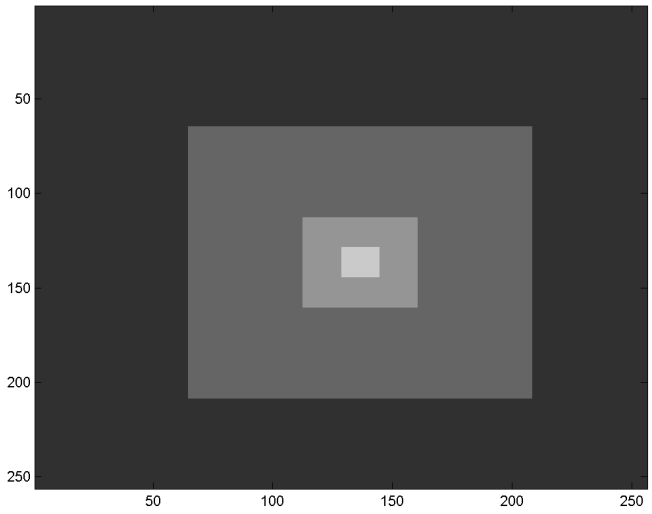

Figura 4.48: Imagen original de Squares.

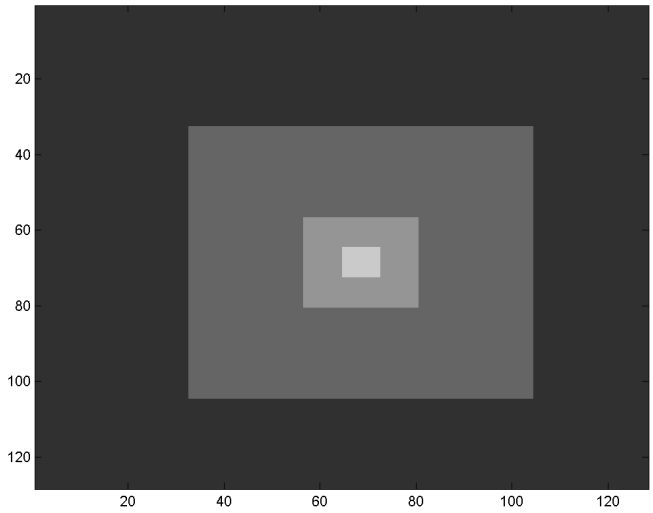

Figura 4.49: Squares a baja resolución con 1 nivel.

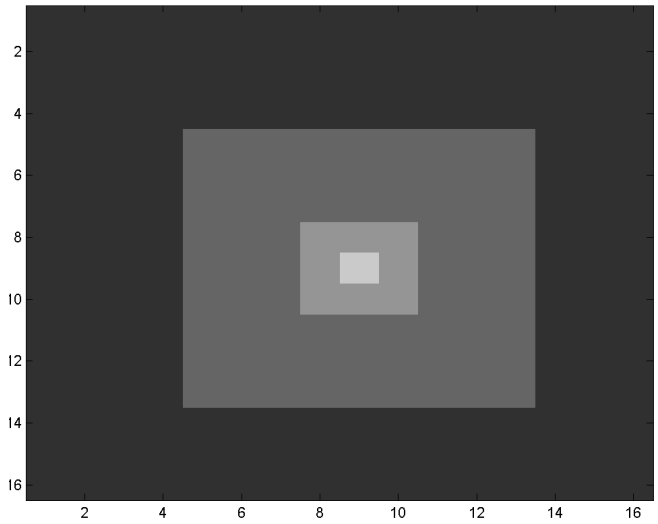

Figura 4.50: Squares a baja resolución con 4 niveles.

Ya que, todos los métodos no lineales nos dan un resultado a simple vista similar para esta imagen, se mostrará solamente la secuencia de imágenes con el método ENO S-R (ver Figuras 4.51, 4.52 , 4.53 y 4.54).

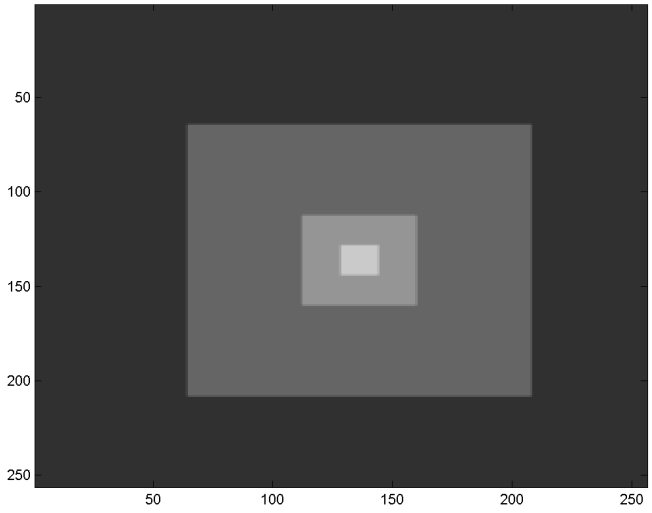

Figura 4.51: Prueba realizada con la imagen Squares con 1 nivel de zoom aplicando el método ENO S-R.

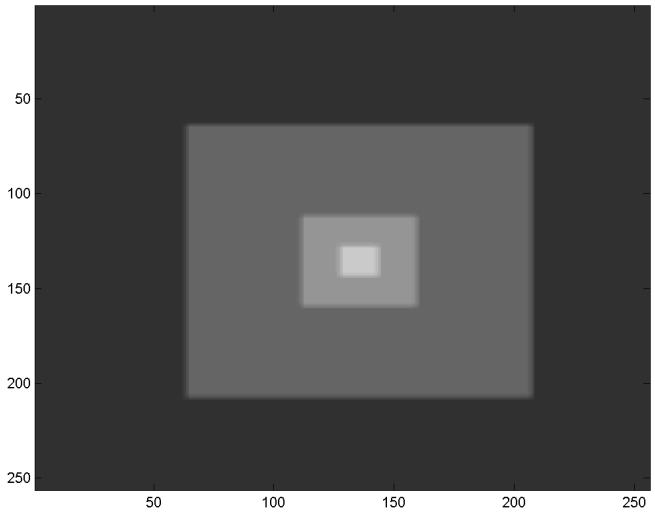

Figura 4.52: Prueba realizada con la imagen Squares con 2 niveles de zoom aplicando el método ENO S-R.

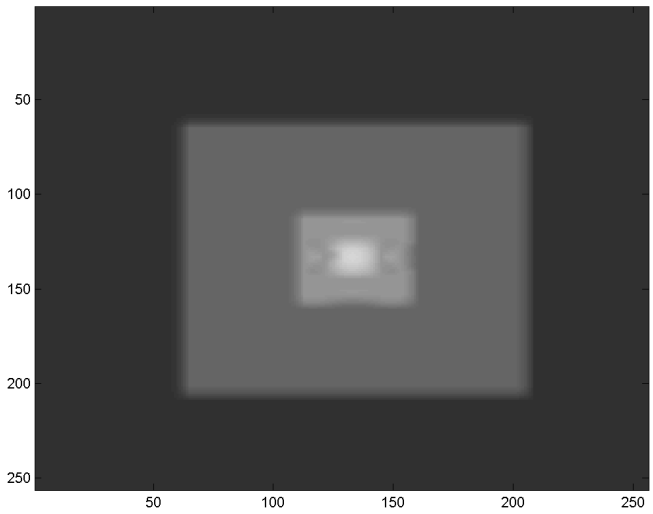

Figura 4.53: Prueba realizada con la imagen Squares con 3 niveles de zoom aplicando el método ENO S-R.

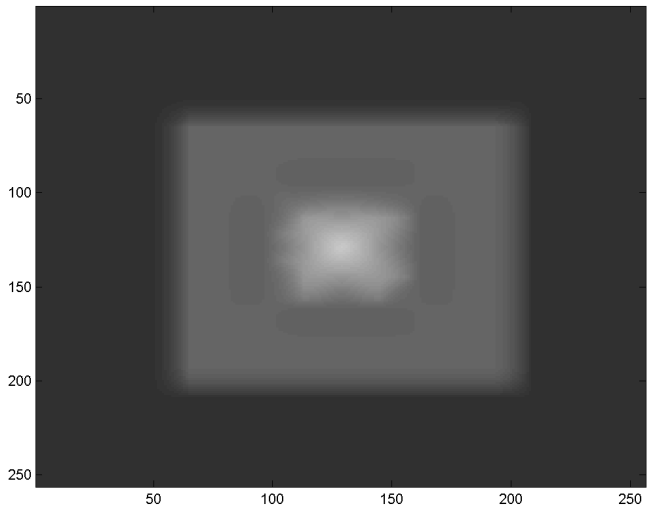

Figura 4.54: Prueba realizada con la imagen Squares con 4 niveles de zoom aplicando el método ENO S-R.

En este apartado nos interesa ofrecer resultados cuantitativos de la calidad del zoom. Para ello ofrecemos en la Tabla 4.1 los resultados del $PSNR$ entre la imagen original 'squares2' y la imagen obtenida mediante zoom a partir de una versión a baja resolución de la imagen original. En la Tabla 4.1 podemos observar cómo la calidad disminuye al aumentar el número de niveles de zoom, aunque aumente el tamaño. También debemos indicar que la mayor calidad visual de los métodos no lineales no queda reflejada con claridad en la Tabla.

Tabla de 'squares2.pgm'.

Método	Niveles de Zoom	$PSNR$
Lineal	1	$39,02$
	2	$33,51$
	3	$29,59$
	4	$26,28$
ENO no jerárquico	1	$43,22$
	2	$36,36$
	3	$31,42$
	4	$27,89$
ENO jerárquico	1	$43,22$
	2	$36,36$
	3	$31,69$
	4	$27,91$
WENO	1	$39,15$
	2	$33,52$
	3	$29,60$
	4	$26,31$
PPH	1	$39,15$
	2	$33,74$
	3	$29,83$
	4	$26,54$
ENO S-R	1	$43,22$
	2	$36,36$
	3	$31,69$
	4	$27,90$

Tabla 4.1: **Imagen original squares2.pgm**, PSNR entre la imagen original y la imagen obtenida mediante zoom a partir de una versión a baja resolución de la imagen original. Resultados para los distintos métodos con 1, 2, 3 y 4 niveles de zoom.

Nuestra siguiente imagen de trabajo es la imagen geométrica 'geo5.pbm' (ver Figura 4.55). En la Figura 4.56 vemos la imagen a baja resolución con tres niveles de multirresolución menos que la original.

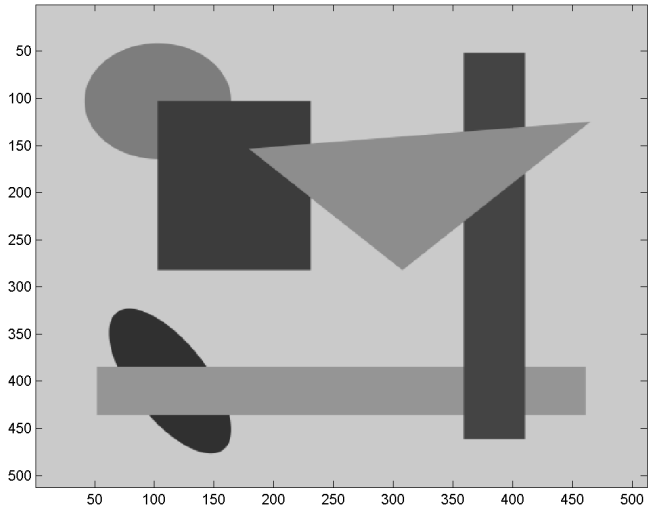

Figura 4.55: Imagen original geométrica.

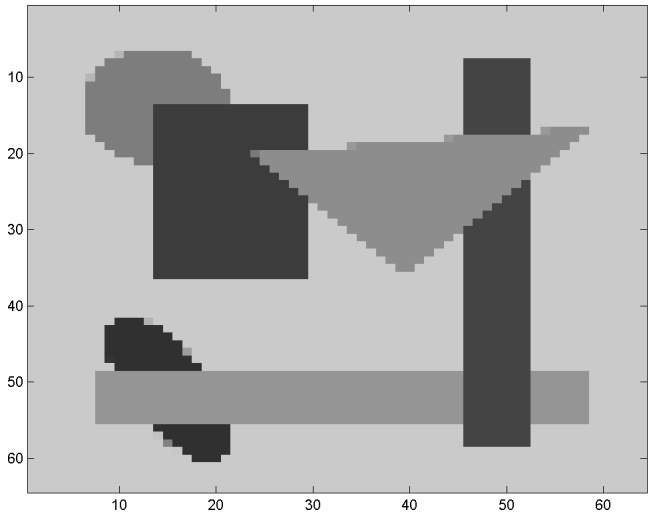

Figura 4.56: Imagen geométrica a baja resolución con 3 niveles.

A continuación mostramos el zoom de la imagen con el método lineal y con el PPH que fueron estudiados en el capítulo 3 (ver Figuras 4.57 y 4.58).

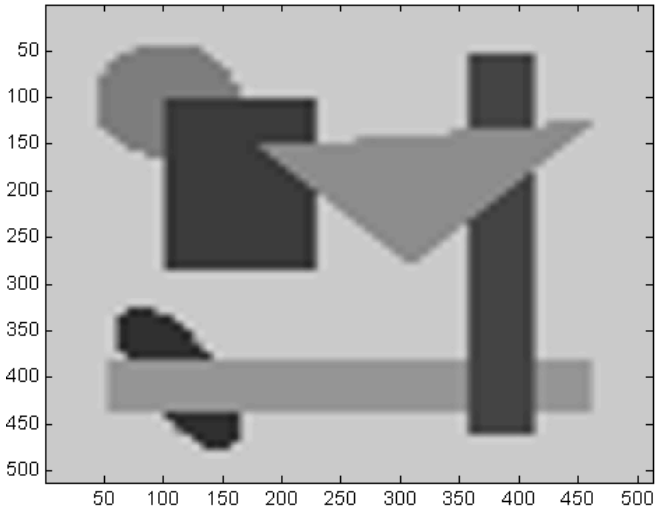

Figura 4.57: Prueba realizada con la imagen geométrica con 3 niveles de zoom aplicando el método Lineal.

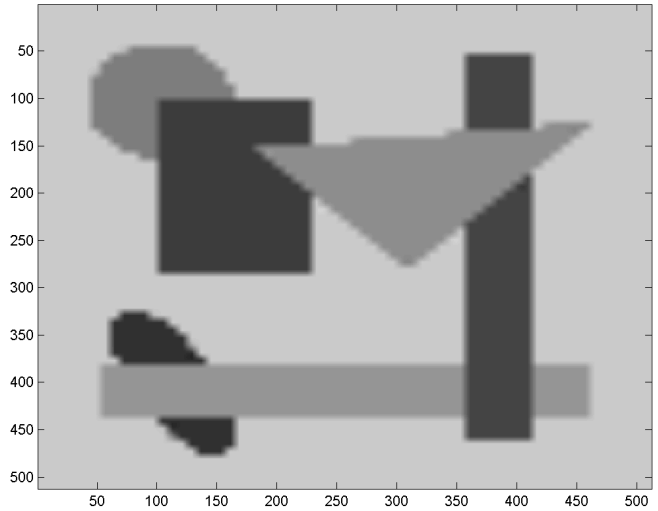

Figura 4.58: Prueba realizada con la imagen geométrica con 3 niveles de zoom aplicando el método PPH.

La Tabla 4.2 nos muestra los resultados similares para los diferentes métodos, destacando ligeramente el método Lineal y PPH. Sin embargo la calidad visual nos dice que es preferible el método no lineal PPH.

Tabla de 'geo5.pbm'.

Método	Niveles de Zoom	$PSNR$
Lineal	1	$37,46$
	2	$32,08$
	3	$26,53$
	4	$24,43$
ENO no jerárquico	1	$36,26$
	2	$30,58$
	3	$26,04$
	4	$23,58$
ENO jerárquico	1	$36,24$
	2	$30,57$
	3	$26,06$
	4	$23,42$
WENO	1	$36,89$
	2	$32,20$
	3	$26,57$
	4	$24,44$
PPH	1	$37,27$
	2	$32,10$
	3	$26,66$
	4	$24,40$
ENO S-R	1	$36,23$
	2	$30,58$
	3	$26,06$
	4	$23,41$

Tabla 4.2: **Imagen original geo5.pbm**, PSNR entre la imagen original y la imagen obtenida mediante zoom a partir de una versión a baja resolución de la imagen original. Resultados para los distintos métodos con 1, 2, 3 y 4 niveles de zoom.

Consideremos la imagen original de 'camera2.pgm' (ver Figura 4.59) y su imagen a baja resolución (ver Figura 4.60).

Figura 4.59: Imagen original del cámara.

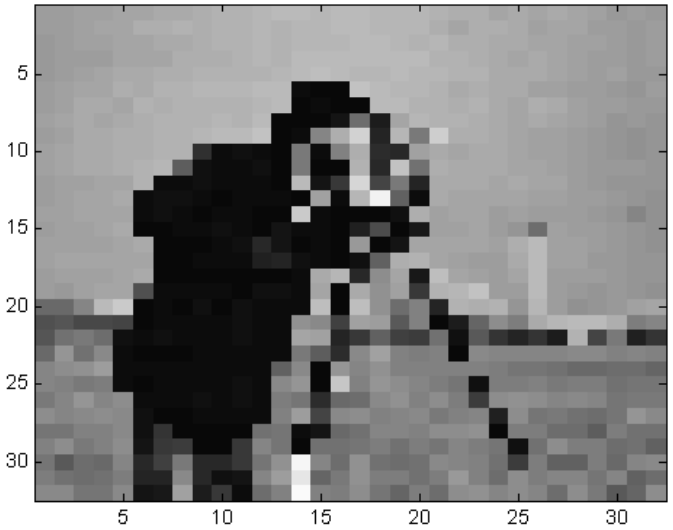

Figura 4.60: Imagen a baja resolución con 3 niveles del cámara.

A continuación mostramos el zoom de la imagen con el método Lineal y PPH estudiados en el capítulo 3 (ver Figuras 4.61 y 4.62).

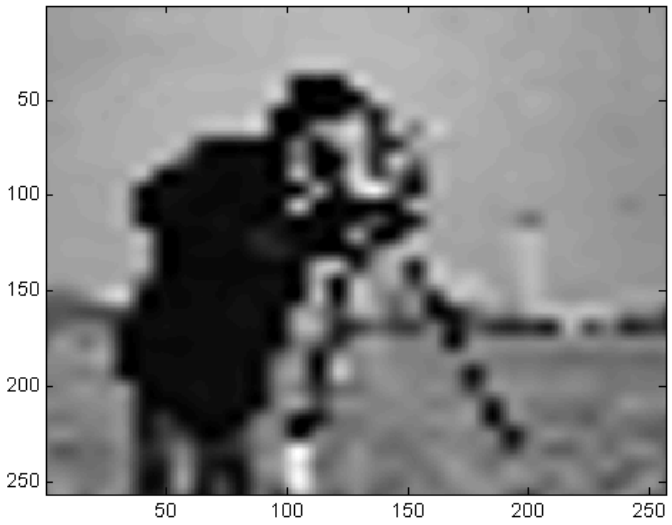

Figura 4.61: Prueba realizada con la imagen cámara con 3 niveles de zoom aplicando el método Lineal.

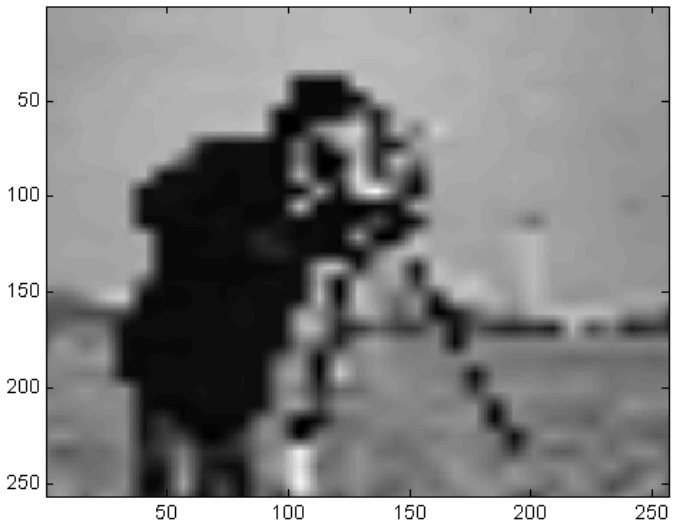

Figura 4.62: Prueba realizada con la imagen cámara con 3 niveles de zoom aplicando el método PPH.

Tabla de 'camera2.pgm'.

Método	Niveles de Zoom	$PSNR$
Lineal	1	$25,50$
	2	$21,33$
	3	$18,87$
	4	$16,44$
ENO no jerárquico	1	$24,91$
	2	$21,19$
	3	$18,85$
	4	$16,55$
ENO jerárquico	1	$24,93$
	2	$21,22$
	3	$18,95$
	4	$16,74$
WENO	1	$25,26$
	2	$21,42$
	3	$19,01$
	4	$16,71$
PPH	1	$25,54$
	2	$21,50$
	3	$19,09$
	4	$16,71$
ENO S-R	1	$24,90$
	2	$21,18$
	3	$18,93$
	4	$16,73$

Tabla 4.3: **Imagen original camera2.pgm**, PSNR entre la imagen original y la imagen obtenida mediante zoom a partir de una versión a baja resolución de la imagen original. Resultados para los distintos métodos con 1, 2, 3 y 4 niveles de zoom.

Las observaciones realizadas anteriormente también son válidas para la imagen 'camera2.pgm' según puede verse en las Figuras 4.59, 4.60, 4.61 y 4.62 y en la Tabla 4.3. Igualmente ofrecemos los resultados obtenidos para la imagen 'lena5'. Ver Figuras 4.63, 4.64, 4.65, 4.66 y la Tabla 4.4.

Figura 4.63: Imagen original de Lena.

Figura 4.64: Lena: Imagen a baja resolución con 3 niveles.

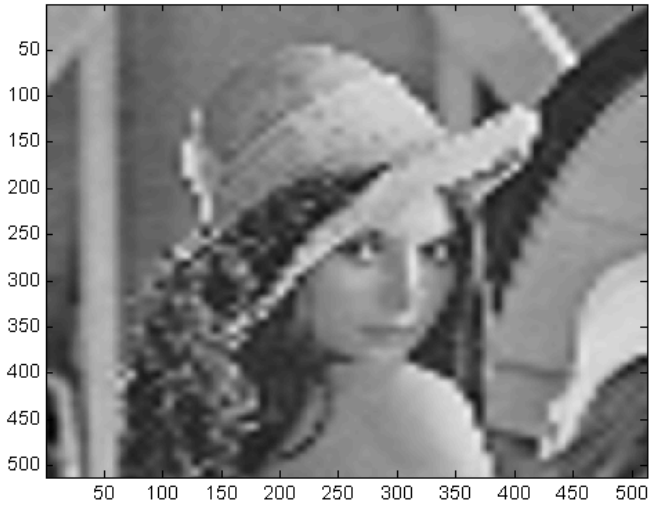

Figura 4.65: Prueba realizada con la imagen Lena con 3 niveles de zoom aplicando el método Lineal.

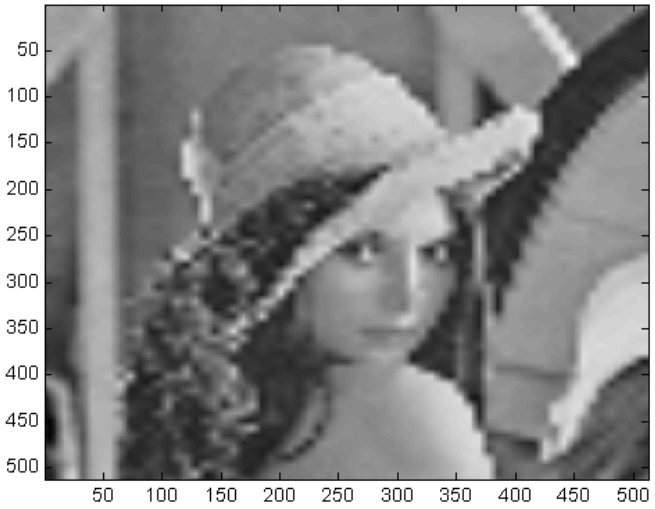

Figura 4.66: Prueba realizada con la imagen Lena con 3 niveles de zoom aplicando el método PPH.

Tabla de 'lena5.pgm'.

Método	Niveles de Zoom	$PSNR$
Lineal	1	$34,00$
	2	$28,08$
	3	$23,92$
	4	$20,78$
ENO no jerárquico	1	$32,88$
	2	$27,48$
	3	$23,63$
	4	$20,79$
ENO jerárquico	1	$32,87$
	2	$27,49$
	3	$23,66$
	4	$20,67$
WENO	1	$33,30$
	2	$27,77$
	3	$23,84$
	4	$20,94$
PPH	1	$33,77$
	2	$28,06$
	3	$24,02$
	4	$20,91$
ENO S-R	1	$32,76$
	2	$27,44$
	3	$23,63$
	4	$20,65$

Tabla 4.4: **Imagen original lena5.pgm**, PSNR entre la imagen original y la imagen obtenida mediante zoom a partir de una versión a baja resolución de la imagen original. Resultados para los distintos métodos con 1, 2, 3 y 4 niveles de zoom.

Capítulo 5

Desarrollo de los programas en Matlab

El desarrollo del proyecto ha sido realizado mediante programación en Matlab de las diferentes funciones necesarias para llevar a cabo cada uno de los pasos hasta la obtención de los resultados.

5.1. Experimento 1: Explicación de los algoritmos

En este apartado del proyecto se va explicar que es lo que hace cada una de las funciones programadas en matlab para el experimento 1, así como dar una visión general de las mismas.

Describamos de forma esquemática y breve el orden en que se van ejecutando cada una de las funciones para conseguir una visión global del procedimiento seguido.

La función principal se denomina zoom_tensor_exp1 y es la que se encarga de realizar, a partir de los parámetros de entrada introducidos, los pasos necesarios para calcular y mostrar el zoom de la imagen. En los parámetros de entrada debemos indicar: el número de niveles de zoom, la imagen que deseamos estudiar, la zona de zoom deseada y el método mediante el cual queremos llevar a cabo nuestro trabajo.

El flujograma, de como se conectan todas las funciones creadas en Mat-

lab y la explicación a groso modo del proceso llevado a cabo, se observa en las Figuras 5.1, 5.2, 5.3 y 5.4.

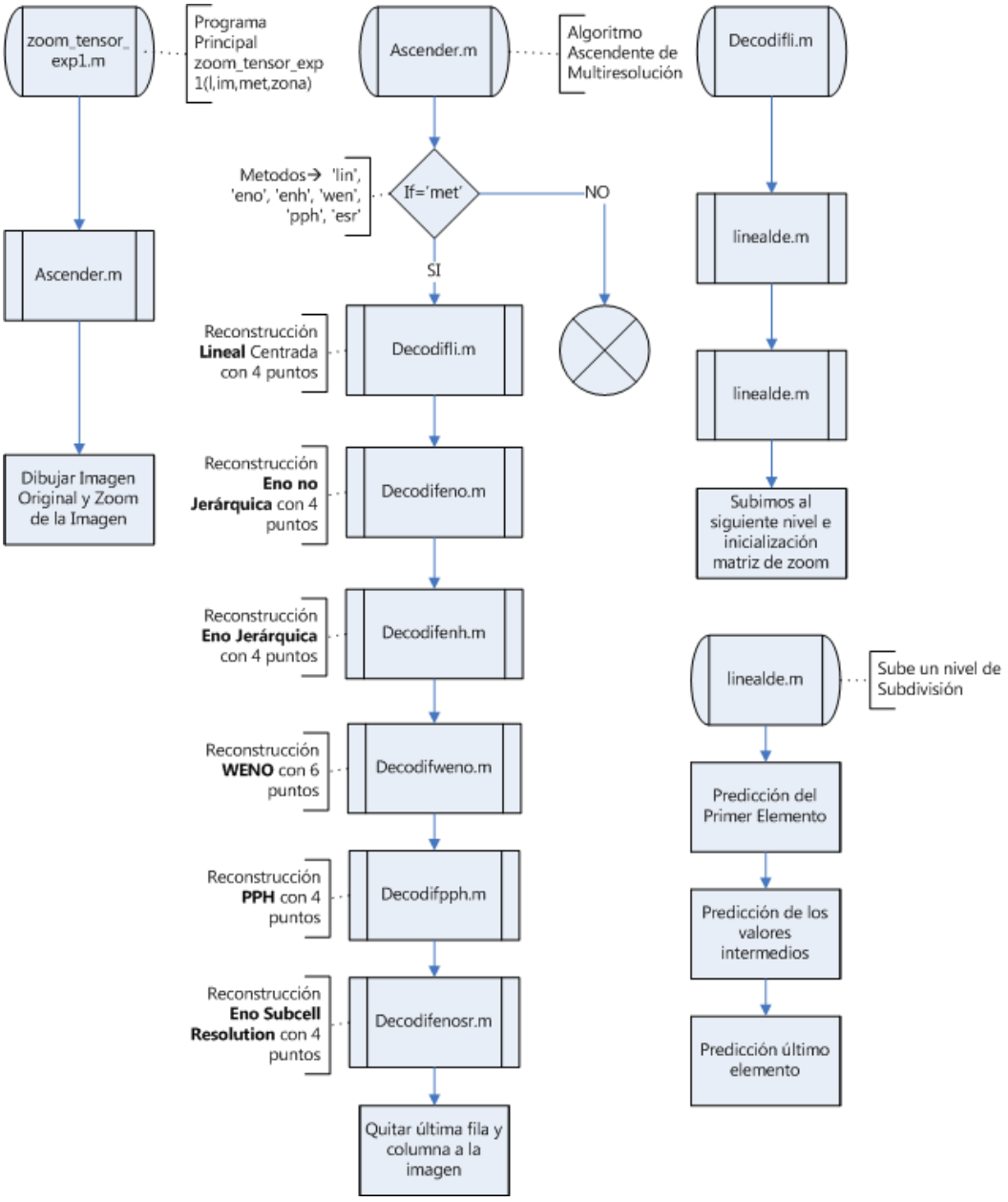

Figura 5.1: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 1, parte 1 de 4.

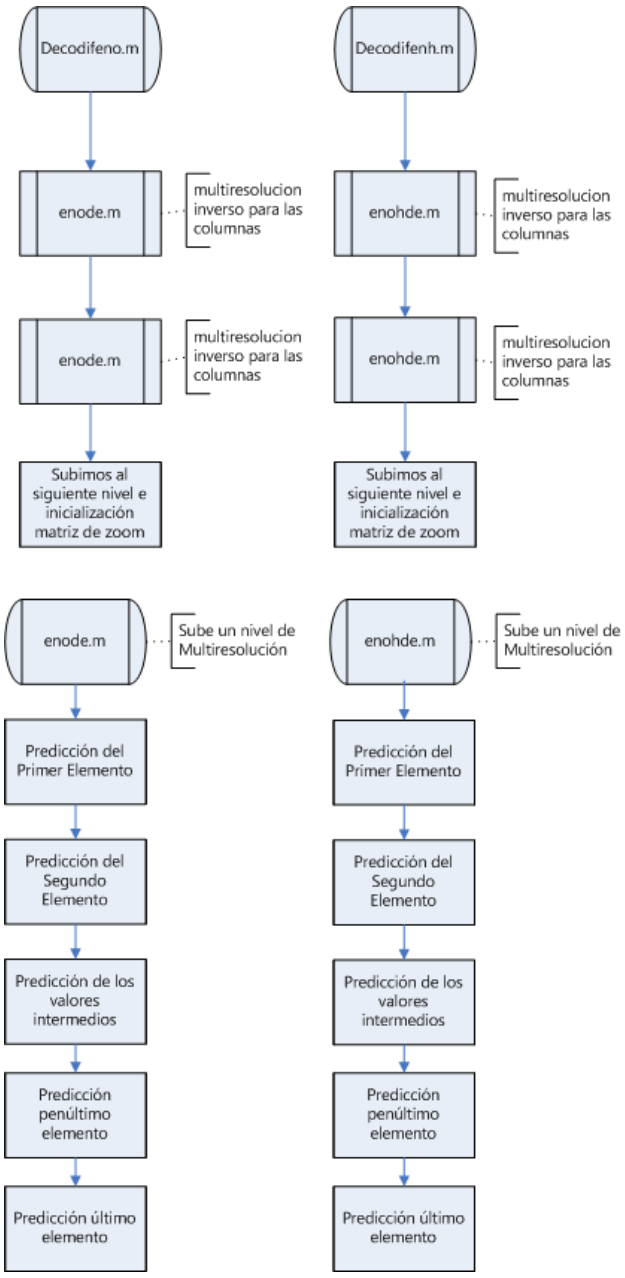

Figura 5.2: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 1, parte 2 de 4.

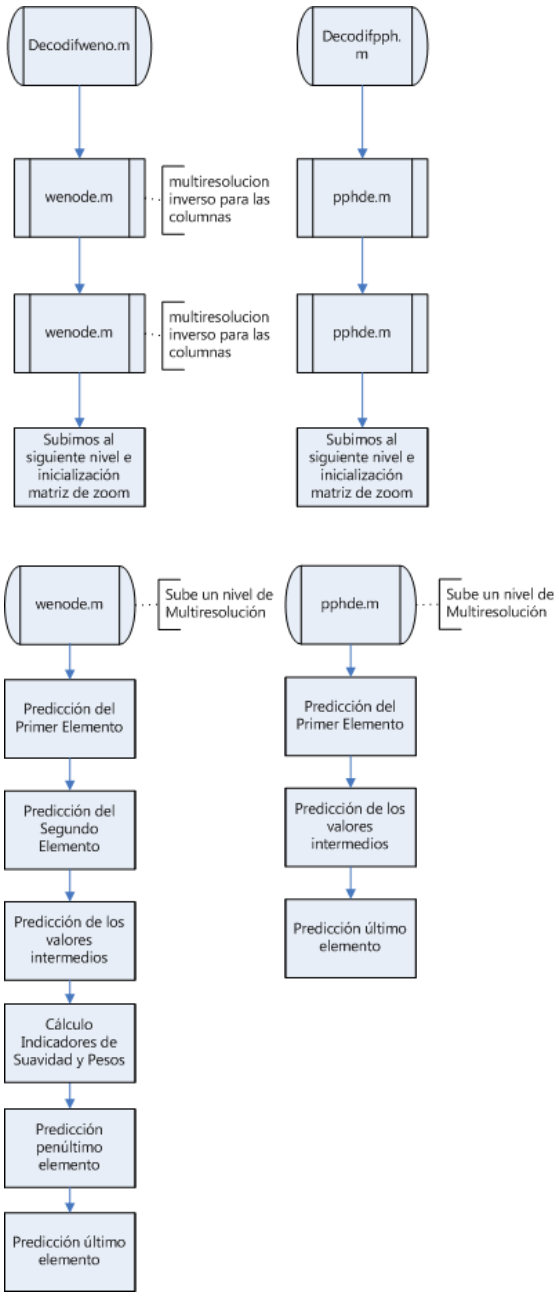

Figura 5.3: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 1, parte 3 de 4.

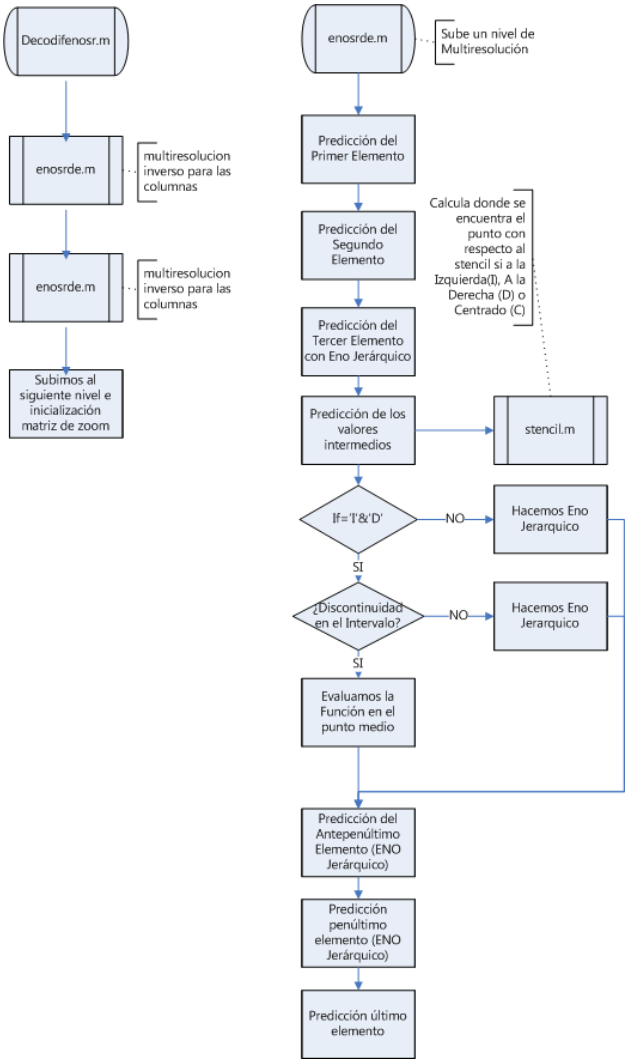

Figura 5.4: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 1, parte 4 de 4.

En este punto vamos a explicar el procedimiento que siguen cada una de las funciones programadas en Matlab enumeradas en el diagrama de flujo anterior (ver Figuras 5.1 , 5.2, 5.3 y 5.4).

Comenzamos con la función *zoom_tensor_exp1.m*. Como se dijo con anterioridad es la función principal de nuestro programa, es la encargada de recibir las características que nosotros imponemos al ejecutar el programa y devolvernos los resultados. La cabecera de la función es de la siguiente manera:

```
function [a,c]=zoom_tensor_exp1(l,im,met,v)
```

Como se observa en la cabecera, la función tensor recibe como entrada cuatro parámetros *l,im,met,v* correspondientes a los niveles de zoom, imagen, método y zona de zoom respectivamente y nos devuelve como salida otros dos parámetros *a,c* que corresponden a la zona de zoom de la imagen original y al zoom de dicha zona. A partir de estos parámetros delega en la función ascender para subir en la pirámide de resolución y llevar a cabo los pasos necesarios para poder calcular las diferentes matrices y por tanto los resultados que el usuario desea. Una vez que la función ascender ha terminado, devuelve el control a la función *zoom_tensor_exp1* encargándose ésta de analizar los datos recibidos y redibujar la imagen de zoom y mostrarla.

A continuación nos encontramos con la función *ascender.m* que tiene la siguiente cabecera:

```
function c=ascender(a,l,met)
```

La función recibe como parámetros de entrada la matriz a la que aplicamos el algoritmo ascendente de multirresolución, el número de niveles de multirresolución que vamos a aplicar y el método mediante el cual queremos realizar la reconstrucción respectivamente y nos devuelve como variable de salida el zoom de la imagen. Esta función se encarga de delegar en una de las siguientes seis funciones (métodos a aplicar) el trabajo a realizar para construir la matriz dependiendo del valor del parámetro de entrada *met*. Según este parámetro se accede a *decodifli.m, decodifeno.m, decodifenoh.m, decodifweno.m, decodifpph.m* o *decodifenosr.m*.

El primero de todos ellos y al que se accedería si el valor de *met* fuese 'lin', *decodifli.m*, tiene la siguiente cabecera:

```
function b=decodifli(a,l)
```

Recibe como parámetros de entrada la matriz a la que vamos a aplicar el algoritmo y el número de niveles de zoom a aplicar y nos produce como variable de salida la matriz reconstruida. El procedimiento que sigue para cada nivel de zoom es la predicción tanto para las filas de la matriz como para las columnas, llamando a la función *linealde.m*. A continuación actualiza el nivel de zoom.

El segundo método al que se accedería si el valor de *met* fuese 'eno' (ENO no jerárquico) tiene la siguiente cabecera:

```
function b=decodifeno(a,l)
```

Como ocurría durante el proceso de ascenso en la pirámide de multirresolución los parámetros de entrada y variables de salida de esta función, igual que en la función anterior y en las dos siguientes, son los mismos, ya que aunque cambie la manera de calcular o de obtener los resultados dependiendo del procedimiento de cada método lo que me interesa es que me devuelvan la matriz reconstruida a partir de unos parámetros que son comunes para todos los métodos. Por lo tanto, *decodifeno.m* recibe también como parámetros de entrada la matriz y los niveles de zoom y a partir de éstos se encarga de realizar el algoritmo inverso de multirresolución por columnas y por filas mediante la función *enode.m* y después actualiza al siguiente nivel de zoom.

La función que se tiene para realizar este paso en el que nos encontramos del programa si el valor de *met* fuese 'enh' (ENO jerárquico) sería *decodifenh.m*, y tiene la siguiente cabecera:

```
function b=decodifenh(a,l)
```

Realiza los mismos pasos que los anteriores accediendo en este caso a la función *enohde.m*.

Si el valor de *met* fuese 'wen' (WENO), nos encontraríamos con el archivo *decodifweno.m* que tiene la siguiente cabecera:

```
function b=decodifweno(a,l)
```

Realiza también los mismos pasos que los anteriores y accede a la función *wenode.m*.

En la opción en que la variable *met* tenga el valor 'pph' y por tanto que ascender delegue el funcionamiento en la función *decodifpph.m* para llevar a cabo el ascenso en la pirámide de multirresolución tenemos:

```
function b=decodifpph(a,l)
```

Esta función se encarga al igual que en los cuatro casos anteriores de realizar la decodificación de la multirresolución por columnas y por filas mediante la función *pphde.m* y de actualizar el nivel de zoom para repetir la operación tantas veces como niveles debamos ascender en la pirámide de multirresolución.

Y por último, nos queda la opción de que la variable *met* tenga el valor 'esr' y por tanto se acceda a la función: *decodifesr.m*, que tiene como cabecera:

```
function b=decodifesr(a,l)
```

Su funcionamiento es igual que los anteriores y para realizar la multirresolución por columnas y por filas tiene la función *enosrde.m* y después se actualiza el nivel de zoom, como en las otras funciones ya mencionadas.

Nos quedaría por último explicar para las funciones *linealde.m, enode.m, enohde.m, wenode.m, pphde.m* y *enosrde.m* a las que se accede desde *decodifli.m, decodifeno.m, decodifenoh.m, decodifweno.m, decodifpph.m* y *decodifenosr.m* respectivamente. Vamos a dar una breve explicación de estos algoritmos ya deducidos previamente en el capítulo 3 de este proyecto. La mayor diferencia entre ellos está en el uso de diferentes máscaras para el cálculo de las predicciones.

La primera función *linealde.m*, que es llamada desde *decodifli.m*, tiene la siguiente cabecera:

```
function f=linealde(v,n)
```

Esta función recibe como parámetros de entrada el vector v que contiene los valores significativos de la escala inferior y la dimensión de v y produce una variable de salida f que corresponde con los valores significativos de la escala superior. Se encarga de calcular todos los valores superiores mediante una reconstrucción lineal de cuatro puntos, calculando el primer y último valor por separado y el resto de los valores intermedios de f mediante un bucle for.

La segunda función *enode.m*, que deriva de *decodifeno.m* tiene la siguiente cabecera:

```
function f=enode(v,n)
```

Recibe como parámetros de entrada el vector v que contiene los valores significativos de la escala inferior y la dimensión de v. En este caso, utilizamos una reconstrucción ENO no jerárquico de cuatro puntos. Los valores de los dos primeros y últimos elementos se calculan desplazándose hacia el lado donde se dispone de datos. Los elementos intermedios mediante un bucle for. Los datos son almacenados en f.

La tercera opción *enohde.m* corresponde a la función que depende de *decodifenoh.m* cuya cabecera es:

```
function f=enohde(v,n)
```

A partir de los valores de entrada calcula los valores significativos del nivel superior y los devuelve almacenados en su variable de salida. Es como la anterior función pero aplicando una reconstrucción ENO jerárquico de cuatro puntos.

La cuarta opción *wenode.m* que se llama desde método WENO, tiene la siguiente cabecera:

```
function f=wenode(v,n)
```

Es similar a las funciones anteriores. En este caso, utilizamos una reconstrucción WENO de cuarto orden. La mayor diferencia está en determinar los elementos intermedios, ya que requieren del cálculo de los indicadores de suavidad y pesos descritos en el capítulo 3. Los datos de salida quedan almacenados en f.

La quinta opción desde que seleccionamos en método 'PPH', es *pphde.m* y tiene como cabecera:

```
function f=pphde(v,n)
```

Al igual que los anteriores, su misión es calcular los valores significativos del nivel superior a partir de los valores de una escala inferior. Una vez calculados a través del método PPH nos lo devuelve en la variable de salida f.

Y como última función nos encontramos *enosrde.m* y su cabecera es la siguiente:

```
function f=enosrde(v,n)
```

Como en las funciones anteriores, recibe los parámetros de entrada ya descritos y devuelve un parámetro de salida que tenemos que calcular mediante una reconstrucción ENO Subcell Resolution. Los valores significativos para los dos primeros elementos y los dos últimos se calculan por separado. Además el tercer y el antepenúltimo elemento se predicen con ENO jerárquico. Para los valores intermedios, llamamos a la función *stencil.m*, que se encarga de determinar si q_{i-1} se descentra a la izquierda y q_{i+1} se descentra a la derecha (ver Figura 5.5). Si esto no se cumple, entonces se aplica ENO jerárquico. Si por el contrario sí que se cumple, entonces se comprueba si $G(x_i)G(x_{i+1}) < 0$ con $G(x) = q_{i+1}(x) - q_{i-1}(x)$. Si no se cumple aplicamos ENO jerárquico. Si se cumple determinamos si el punto medio entre x_i y x_{i+1} está a un lado u otro del punto de corte entre $q_{i-1}(x)$ y $q_{i+1}(x)$, es decir

$$Si\ G(x_i)G(x_{2i+1}) \leq 0 \Rightarrow \text{Predicción} = \text{q}_{i+1}(x_{2i+1})$$
$$Si\ G(x_i)G(x_{2i+1}) > 0 \Rightarrow \text{Predicción} = \text{q}_{i-1}(x_{2i+1})$$
$$\text{donde } x_{2i+1} \text{ es el punto medio.}$$

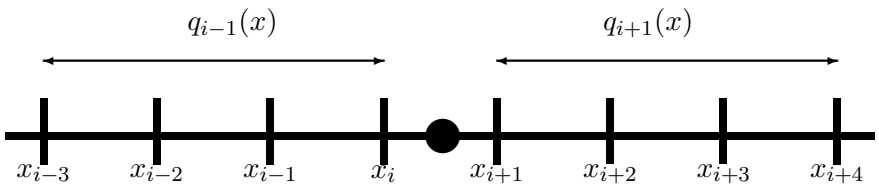

Figura 5.5: Polinomios $q_{i-1}(x)$ y $q_{i+1}(x)$ en la predicción con ENO Subcell Resolution.

5.2. Experimento 2: Explicación de los algoritmos

En este apartado del proyecto se va explicar que es lo que hace cada una de las funciones matlab para el experimento 2, así como dar una visión general de las mismas.

El flujograma, de como se conectan todas las funciones creadas en Matlab y la explicación a groso modo del proceso llevado a cabo se observa en las Figuras 5.6 , 5.7, 5.8 y 5.9.

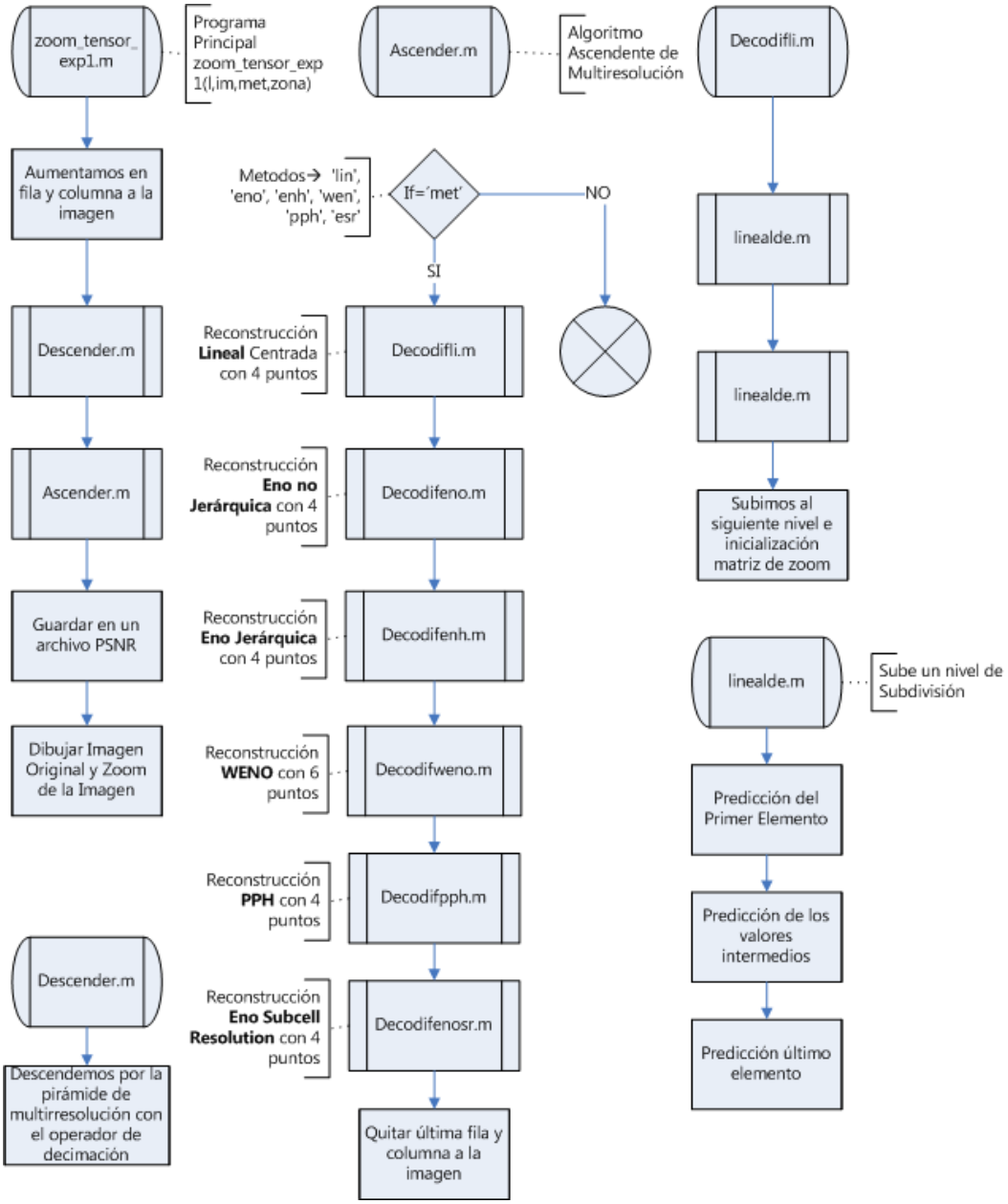

Figura 5.6: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 2, parte 1 de 4.

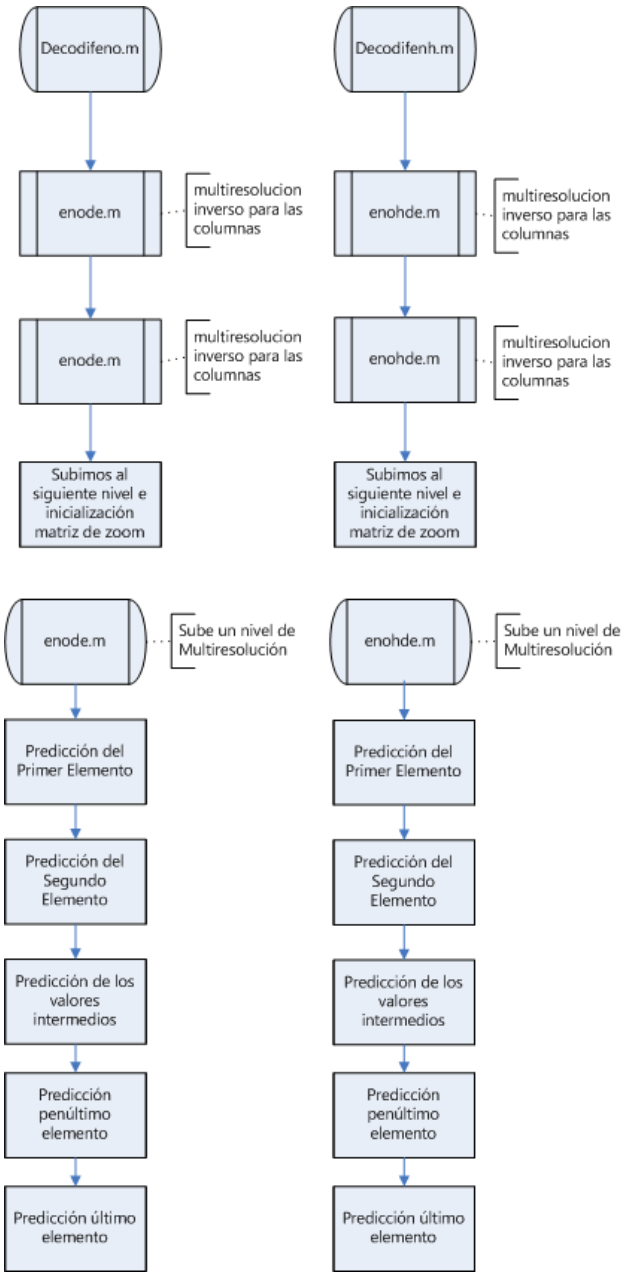

Figura 5.7: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 2, parte 2 de 4.

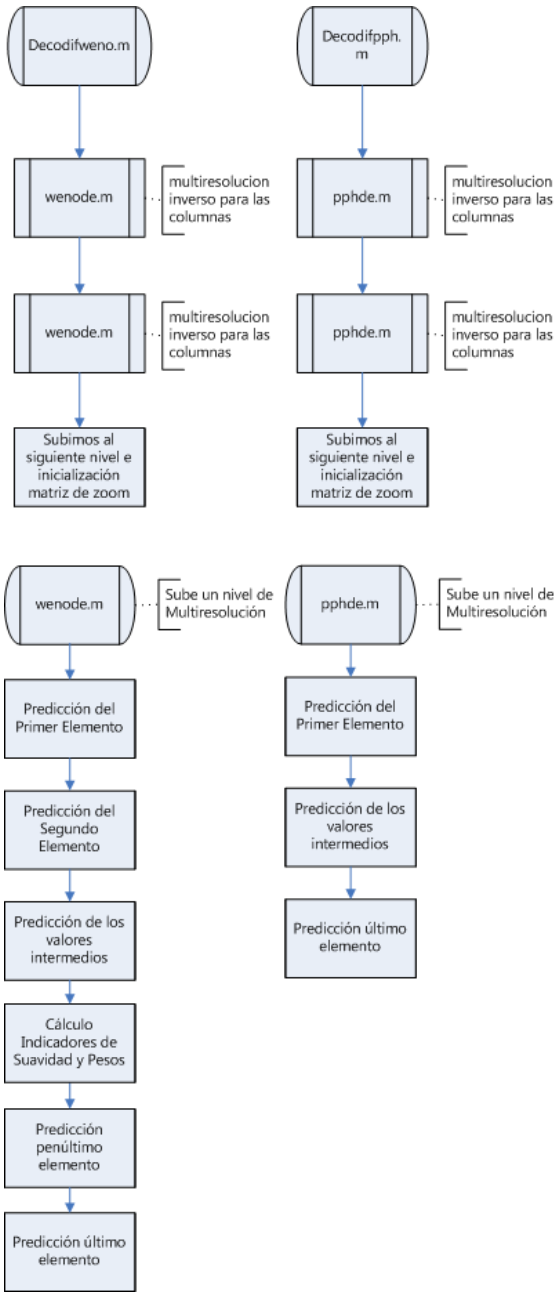

Figura 5.8: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 2, parte 3 de 4.

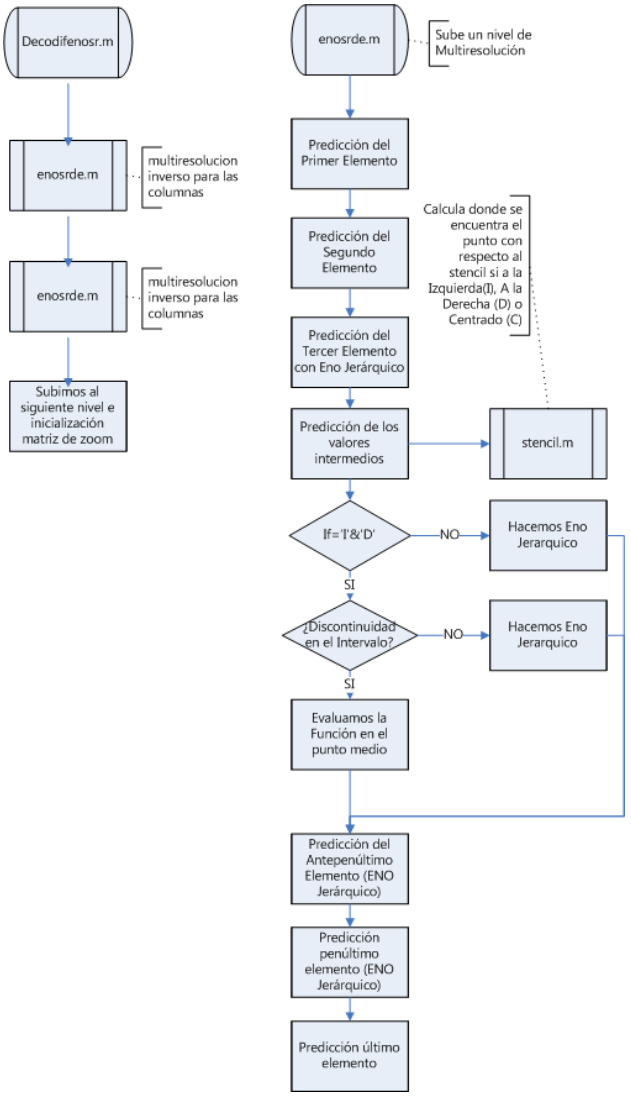

Figura 5.9: Diagrama de flujo de los pasos realizados para hacer zoom en el experimento 2, parte 4 de 4.

En el experimento 2, nos centraremos en las funciones que no sean iguales a las del experimento 1, dando una breve explicación del descenso en la pirámide de multirresolución (que en el experimento 1 no haciamos uso de ella). La funciones iguales a las del experimento 1 son: decodifli, decodifeno, decodifenh, decodifweno, decodifpph, decodifenosr, lenealde, enode, enohde, wenode, pphde, enosrde, stencil (para enosrde). Por lo que se describirán sólo las funciones *zoom_tensor_exp2* y *descender*.

Comenzamos con la función *zoom_tensor_exp2.m*. Es la función principal de nuestro programa, es la encargada de recibir las características que nosotros imponemos al ejecutar el programa y devolvernos los resultados más el fichero donde se nos guardan las simulaciones *datos generados.rtf*. La cabecera de la función es de la siguiente manera:

```
function [a,c]=zoom_tensor_exp2(l,im,met)
```

Como se observa en la cabecera, la función tensor recibe como entrada tres parámetros *l,im,met* correspondientes a los niveles de zoom, imagen y método y nos devuelve como salida *a,c* que corresponden a la matriz original a baja resolución y al zoom de la imagen original, calculada a partir de la imagen a baja resolución. A partir de los parámetros de entrada delega en la función descender para bajar en la pirámide de multirresolución, descendiendo l niveles. Después de descender pasamos a la función ascender para subir en la pirámide de resolución. Una vez que la función ascender a terminado, devuelve el control a la función *zoom_tensor_exp2* esta se encarga de analizar los datos recibidos, redibujar la imagen de zoom y mostrala. En el fichero *datos generados.rtf* se guarda la fecha y hora de la simulación, el número de niveles de zoom, el nombre de la imagen y el método elegido por el usuario, además de calcularnos $norma1$, $infinito$, $norma2$ y el $PSRN$ de la diferencia entre la imagen original y el zoom.

A continuación nos encontramos con la función *descender.m*. Esta función tiene la siguiente cabecera:

```
function aux1=descender(aux,l)
```

Como parámetros de entrada recibe dos *aux,l* correspondientes a la matriz de la imagen original y los niveles de multirresolución que se desciende respectivamente. Produce como variable de salida *aux1* siendo la versión de la imagen original ya en forma de matriz a baja resolución después de haber descendido l niveles con el operador de decimación.

Una vez terminado el proceso de descenso por la pirámide de multirresolución, se siguen los mismos pasos que en el experimento 1.

Para visualizar el contenido completo de las funciones programadas con Matlab, se incluye a continuación su código completo, para ambos experimentos.

5.3. Experimento 1: Código fuente de los algoritmos

zoom_tensor_exp1.m

```
function [a,c]=zoom_tensor_exp1(l,im,met,v)

% [a,c]=zoom_tensor_exp1(l,im,met,v);
% -------variables de salida-------
% a imagen original
% c zoom de la imagen original
% ------variables de entrada-------
% l son los niveles de zoom
% im imagen a utilizar:
% 'squares2.pgm', 'geo5.pbm', 'camera2.pgm', 'lena5.pgm', etc...
% met método a utilizar 'lin', 'eno', 'enh', 'wen', 'pph', 'esr'
% v = zona de la imagen a hacer zoom por ejemplo: [130,165,130,165]
% definición de la matriz, la guardamos en a

b=imread(im);
[n,m]=size(b);
b=double(b);

% Dibujamos la imagen original

figure('Tag','Original','Name',[blanks(6),'IMAGEN ORIGINAL'],...
    'Position',[722 188 560 420])

% axes('Position',[0 0 1 1],'Visible','off');

imagesc(b,[0 255])
title('imagen original');
colormap gray
rect_posicion=[v(1),v(3),v(2)-v(1),v(4)-v(3)];
rectangle('Position',rect_posicion,...
    'LineWidth',3,'EdgeColor',[1 0 0],'LineStyle','-');

a=b(v(3):v(4),v(1):v(2));
clear b;
```

```
%t0=clock;
%t=cputime;

% ascendemos por la piramide de multiresolución

c=ascender(a,l,met);

%tiempo=etime(clock,t0);
%tiempo1=cputime-t;

% Dibujamos una nueva figura

figure('Tag','zoom','Name',[blanks(6),'ZOOM DE LA IMAGEN',...
    blanks(2),'METODO',blanks(2),upper(met),blanks(2),'NIVELES',...
    blanks(2),num2str(l)],'Position',[6 100 710 690])
h=axes('Position',[0 0 1 1],'Visible','off');

% mostramos el zoom de la imagen dentro de la nueva figura

dx=v(2)-v(1);
dy=v(4)-v(3);

ax=(0.87*dx)/n;
ay=(0.87*dy)/m;

for i=1:l
    ax=2*ax;
    ay=2*ay;
end

if ax>ay
    if ax>=0.87
        ax=0.87;
        ay=(dy*ax)/dx;
    end
else
    if ay>0.87
        ay=0.87;
        ax=(dx*ay)/dy;
    end
end
```

```
% Centrar imagen

centerX= 0.075+(0.87-ax)/2;
centerY= 0.035+(0.87-ay)/2;

axes('Position',[centerX centerY ax  ay])
imagesc(c,[0 255])
title('zoom de la imagen','Color','r');
colormap gray

% incluimos un texto explicativo en la nueva figura

str=['Método:',blanks(2),met,blanks(10),'Imagen:',blanks(2),im,...
    blanks(10),'Zoom:',blanks(2),num2str(l),...
    blanks(10),'Zona:',blanks(2),'[',blanks(1),...
    num2str(v),blanks(1),']'];
set(gcf,'CurrentAxes',h)
text(.025,.97,str,'FontSize',12)
```

ascender.m

```
function c=ascender(a,l,met)

% c=ascender(a,l,met);
% c aproximación a la imagen
% a matriz a la que le aplicamos el algorimo ascendente
% l son los niveles de zoom
% met método que quieres utilizar 'lin', 'eno','enh', 'pph', 'esr'


% llamamos al método

if(met=='lin')
    c=decodifli(a,l);
end
if(met=='eno')
    c=decodifeno(a,l);
end
if(met=='enh')
    c=decodifenh(a,l);
```

```
end
if(met=='wen')
    c=decodifweno(a,l);
end
if(met=='pph')
    c=decodifpph(a,l);
end
if(met=='esr')
    c=decodifenosr(a,l);
end
```

decodifli.m

```
function b=decodifli(a,l)

% b=decodifli(a,l);
% b zoom utilizando una reconstrucción lineal centrada
% con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% l son los niveles de subdivisión (zoom)

[n,m]=size(a);

for k=1:l

% se trata del bucle para los niveles de zoom

nf=2*n-1; nc=2*m-1;
b=zeros(nf,nc);
b(1:2:nf,1:2:nc)=a(1:n,1:m);

% se hace el algoritmo de predicción para las columnas

    for j=1:2:nc
        b(1:nf,j)=linealde(b(1:2:nf,j),n)';
    end

% se hace el algoritmo de predicción para las filas

    for i=1:nf
        b(i,1:nc)=linealde(b(i,1:2:nc),m);
    end
```

```
% se actualizan n y m para subir al siguiente nivel

    n=nf;
    m=nc;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

linealde.m

```
function f=linealde(v,n)

% f=linealde(v,n);
% programa que sube un nivel de subdivision utilizando
% una reconstrucción lineal centrada de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos
% de la escala inferior
% n dimensión de v

f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción de los valores intermedios de f

for j=4:2:2*n-4
    f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
end

% predicción del ultimo elemento

f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

decodifeno.m

```
function b=decodifeno(a,l)

% b=decodifeno(a,l);
% b zoom utilizando una reconstrucción
% eno no jerárquica con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% l son los niveles de subdivisión (zoom)

[n,m]=size(a);

for k=1:l

% se trata del bucle para los niveles de zoom

nf=2*n-1; nc=2*m-1;
b=zeros(nf,nc);
b(1:2:nf,1:2:nc)=a(1:n,1:m);

% se hace el algoritmo de predicción para las columnas

    for j=1:2:nc
        b(1:nf,j)=enode(b(1:2:nf,j),n)';
    end

% se hace el algoritmo de predicción para las filas

    for i=1:nf
        b(i,1:nc)=enode(b(i,1:2:nc),m);
    end

% se actualizan n y m para subir al siguiente nivel

    n=nf;
    m=nc;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

enode.m

```
function f=enode(v,n)
```

```
% f=enode(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción eno no jerarquica de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de la escala
% inferior y los detalles para subir
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción del segundo elemento

at2=abs(-v(1)+3*v(2)-3*v(3)+v(4));
at3=abs(-v(2)+3*v(3)-3*v(4)+v(5));

if(at2<=at3)
    f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
    f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
end

% predicción de los valores intermedios de f

for j=6:2:2*n-6
  at1=abs(-v(j/2-2)+3*v(j/2-1)-3*v(j/2)+v(j/2+1));
  at2=abs(-v(j/2-1)+3*v(j/2)-3*v(j/2+1)+v(j/2+2));
  at3=abs(-v(j/2)+3*v(j/2+1)-3*v(j/2+2)+v(j/2+3));

  if(at2<=at1 & at2<=at3)
      f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
  else
      if(at1<=at3)
         f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
      else
         f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);
      end
  end
end
```

```
% predicción del penúltimo elemento

at1=abs(-v(n-4)+3*v(n-3)-3*v(n-2)+v(n-1));
at2=abs(-v(n-3)+3*v(n-2)-3*v(n-1)+v(n));

if(at2<=at1)
    f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
    f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
end

% predicción del último elemento

f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

decodifenh.m

```
function b=decodifenh(a,l)

% b=decodifenh(a,l);
% b zoom utilizando una reconstrucción eno jerárquica
% con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% l son los niveles de subdivisión (zoom)

[n,m]=size(a);

for k=1:l

% se trata del bucle para los niveles de zoom

    nf=2*n-1; nc=2*m-1;
    b=zeros(nf,nc);
    b(1:2:nf,1:2:nc)=a(1:n,1:m);

% se hace el algoritmo de predicción para las columnas

    for j=1:2:nc
        b(1:nf,j)=enohde(b(1:2:nf,j),n)';
    end

% se hace el algoritmo de predicción para las filas
```

```
    for i=1:nf
        b(i,1:nc)=enohde(b(i,1:2:nc),m);
    end

% se actualizan n y m para subir al siguiente nivel

    n=nf;
    m=nc;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

enohde.m

```
function f=enohde(v,n)

% f=enohde(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción eno jerárquica de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de la escala
% inferior y los detalles para subir
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción del segundo elemento

p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
s1=(p2-p1); as1=abs(s1);
s2=(p3-p2); as2=abs(s2);
```

```
if(as1<=as2)
   f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
   s3 =(p4-p3);
   t1=(s2-s1); at1=abs(t1);
   t2=(s3-s2); at2=abs(t2);
   if(at1<=at2)
      f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
   else
      f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
   end
end

% predicción de los valores intermedios de f

for j=6:2:2*n-6
  p1=(v(j/2-1)-v(j/2-2));
  p2=(v(j/2)-v(j/2-1));
  p3=(v(j/2+1)-v(j/2));
  p4=(v(j/2+2)-v(j/2+1));
  p5=(v(j/2+3)-v(j/2+2));
  s1=(p2-p1);
  s2=(p3-p2); as2=abs(s2);
  s3=(p4-p3); as3=abs(s3);
  s4=(p5-p4);

  if(as2<=as3)
      t1=(s2-s1); at1=abs(t1);
      t2=(s3-s2); at2=abs(t2);
      if(at2<=at1)
         f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
      else
         f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
      end
  else
      t2=(s3-s2); at2=abs(t2);
      t3=(s4-s3); at3=abs(t3);
      if(at2<=at3)
         f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
      else
```

```
         f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);
      end
  end
end

% predicción del penúltimo elemento

p1=(v(n-3)-v(n-4));
p2=(v(n-2)-v(n-3));
p3=(v(n-1)-v(n-2));
p4=(v(n)-v(n-1));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);

if(as3<=as2)
   f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
   t1=(s2-s1); at1=abs(t1);
   t2=(s3-s2); at2=abs(t2);
   if(at2<=at1)
      f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
   else
      f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
   end
end

% predicción del último elemento

f(2*n-2)=(3/48)*v(n-3)-(15/48)*v(n-2)+(45/48)*v(n-1)+(15/48)*v(n);
```

decodifweno.m

```
function b=decodifweno(a,l)

% b=decodifweno(a,l);
% b zoom utilizando una reconstrucción eno de cuarto orden
% a matriz a la que se le va a aplicar el zoom
% l son los niveles de subdivisión (zoom)
```

```
[n,m]=size(a);

for k=1:l

% se trata del bucle para los niveles de zoom

    nf=2*n-1; nc=2*m-1;
    b=zeros(nf,nc);
    b(1:2:nf,1:2:nc)=a(1:n,1:m);

% se hace el algoritmo de predicción para las columnas

    for j=1:2:nc
        b(1:nf,j)=wenode(b(1:2:nf,j),n)';
    end

% se hace el algoritmo de predicción para las filas

    for i=1:nf
        b(i,1:nc)=wenode(b(i,1:2:nc),m);
    end

% se actualizan n y m para subir al siguiente nivel

    n=nf;
    m=nc;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

wenode.m

```
function f=wenode(v,n)

% f=wenode(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción weno de cuarto orden
% f vector de la escala superior
% v vector que contiene los valores significativos de la escala
% inferior y los detalles para subir
% n dimensión de v
```

```
f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción del segundo elemento

at2=abs(-v(1)+3*v(2)-3*v(3)+v(4));
at3=abs(-v(2)+3*v(3)-3*v(4)+v(5));

if(at2<=at3)
    f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
    f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
end

% predicción de los valores intermedios de f

epsil=10^(-6);

for j=6:2:2*n-6
    at1=v(j/2-1)-v(j/2-2);
    at2=v(j/2)-v(j/2-1);
    at3=v(j/2+1)-v(j/2);
    at4=v(j/2+2)-v(j/2+1);
    at5=v(j/2+3)-v(j/2+2);

% cálculo de los indicadores de suavidad

    inleft=0.5*((at2-at1)^2+(at3-at2)^2)+(at3-2*at2+at1)^2;
    incenter=0.5*((at3-at2)^2+(at4-at3)^2)+(at4-2*at3+at2)^2;
    inright=0.5*((at4-at3)^2+(at5-at4)^2)+(at5-2*at4+at3)^2;

% cálculo de los numeradores de los pesos

    alfaleft=3./16/(epsil+inleft)^3;
    alfacenter=10./16/(epsil+incenter)^3;
    alfaright=3./16/(epsil+inright)^3;

% cálculo de los pesos
```

```
    alfasum=alfaleft+alfacenter+alfaright;
    wleft=alfaleft/alfasum;
    wcenter=alfacenter/alfasum;
    wright=alfaright/alfasum;

% sumamos los tres polinomios multiplicados por
% sus correspondientes pesos

    qcenter=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
    qleft=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
    qright=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);

    f(j)=wleft*qleft+wcenter*qcenter+wright*qright;
end

% predicción del penúltimo elemento

at1=abs(-v(n-4)+3*v(n-3)-3*v(n-2)+v(n-1));
at2=abs(-v(n-3)+3*v(n-2)-3*v(n-1)+v(n));

if(at2<=at1)
    f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
    f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
end

% predicción del último elemento

f(2*n-2)=(3/48)*v(n-3)-(15/48)*v(n-2)+(45/48)*v(n-1)+(15/48)*v(n);
```

decodifpph.m

```
function b=decodifpph(a,l)

% b=decodif1(a,l);
% b zoom utilizando una reconstrucción pph
% con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% l son los niveles de subdivisión (zoom)

[n,m]=size(a);

for k=1:l
```

```
% se trata del bucle para los niveles de zoom

nf=2*n-1; nc=2*m-1;
b=zeros(nf,nc);
b(1:2:nf,1:2:nc)=a(1:n,1:m);

% se hace el algoritmo de predicción para las columnas

    for j=1:2:nc
        b(1:nf,j)=pphde(b(1:2:nf,j),n)';
    end

% se hace el algoritmo de predicción para las filas

    for i=1:nf
        b(i,1:nc)=pphde(b(i,1:2:nc),m);
    end

% se actualizan n y m para subir al siguiente nivel

    n=nf;
    m=nc;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

pphde.m

```
function f=pphde(v,n)

% f=pphde(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción pph de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de la
% escala inferior y los detalles para subir
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);
```

```
% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción de los valores intermedios de f

for j=4:2:2*n-4
    d1=v(j/2-1)-2*v(j/2)+v(j/2+1);
    d2=v(j/2)-2*v(j/2+1)+v(j/2+2);
    d=d1*d2;
    s=(v(j/2)+v(j/2+1))/2;
    if (d>0)
        f(j)=s-d/(4*(d1+d2));
    else
        f(j)=s;
    end
end

% predicción del último elemento

f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

decodifenosr.m

```
function b=decodifenosr(a,l)

% b=decodifenosr(a,l);
% b zoom utilizando una reconstrucción eno S-R con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% l son los niveles de subdivisión (zoom)

[n,m]=size(a);

for k=1:l

% se trata del bucle para los niveles de zoom

nf=2*n-1; nc=2*m-1;
b=zeros(nf,nc);
b(1:2:nf,1:2:nc)=a(1:n,1:m);

% se hace el algoritmo de predicción para las columnas
```

```
    for j=1:2:nc
        b(1:nf,j)=enosrde(b(1:2:nf,j),n)';
    end

% se hace el algoritmo de predicción para las filas

    for i=1:nf
        b(i,1:nc)=enosrde(b(i,1:2:nc),m);
    end

% se actualizan n y m para subir al siguiente nivel

    n=nf;
    m=nc;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

enosrde.m

```
function f=enosr(v,n)

% f=enosrde(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción eno S-R de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de la
% escala inferior
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción del segundo elemento
```

```
p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
s1=(p2-p1); as1=abs(s1);
s2=(p3-p2); as2=abs(s2);

if(as1<=as2)
   f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
   s3 =(p4-p3);
   t1=(s2-s1); at1=abs(t1);
   t2=(s3-s2); at2=abs(t2);
   if(at1<=at2)
      f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
   else
      f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
   end
end

% predicción del tercer elemento: con ENO jerárquico

p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
p5=(v(6)-v(5));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);
s4=(p5-p4);

if(as2<=as3)
       t1=(s2-s1); at1=abs(t1);
       t2=(s3-s2); at2=abs(t2);
       if(at2<=at1)
           f(6)=(-v(2)+9*v(3)+9*v(4)-v(5))/16;
       else
           f(6)=(3/48)*v(1)-(15/48)*v(2)+(45/48)*v(3)+(15/48)*v(4);
       end
else
```

```
        t2=(s3-s2); at2=abs(t2);
        t3=(s4-s3); at3=abs(t3);
        if(at2<=at3)
            f(6)=(-v(2)+9*v(3)+9*v(4)-v(5))/16;
        else
            f(6)=(15/48)*v(3)+(45/48)*v(4)-(15/48)*v(5)+(3/48)*v(6);
        end
end

% predicción de los valores intermedios de f

for j=8:2:2*n-8

% PASO 1: Para cada intervalo determinamos los stencil a la izquierda
% y a la derecha. Si no se descentra aplicamos ENO-H.

   stencil_izq=stencil(v(j/2-3:j/2+2));
   stencil_der=stencil(v(j/2-1:j/2+4));

   if(stencil_izq=='I' & stencil_der=='D')

% PASO 3: Continuamos con el método de resolución subcelda
% comprobamos si g(x_i) * g(x_{i+1}) < 0
% valores de los polinomios q_{i-1}(x_i), q_{i-1}(x_{i+1}),
% q_{i+1}(x_i), q_{i+1}(x_{i+1}) respectivamente

        q11=v(j/2);
        q12=-v(j/2-3)+4*v(j/2-2)-6*v(j/2-1)+4*v(j/2);
        q21=4*v(j/2+1)-6*v(j/2+2)+4*v(j/2+3)-v(j/2+4);
        q22=v(j/2+1);

% valores de la función g en x_i y x_{i+1}

        g1=q21-q11;
        g2=q22-q12;

        if(g1*g2 < 0)

% PASO 4: Efectivamente hay una discontinuidad en el intervalo y
% extendemos q_{i-1}(x) y q_{i+1}(x)
% polinomios q_{i-1}(punto medio), q_{i+1}(punto medio)
```

```
            q1m=-5/16*v(j/2-3)+21/16*v(j/2-2)-35/16*v(j/2-1)+35/16*v(j/2);
            q2m=35/16*v(j/2+1)-35/16*v(j/2+2)+21/16*v(j/2+3)-5/16*v(j/2+4);

% función g evaluada en el punto medio

            gm=q2m-q1m;

            if (g1*gm <0)
                 f(j)=q2m;
             else
                 f(j)=q1m;
             end

       else

% PASO 2: Hace ENO jerárquico

      p1=(v(j/2-1)-v(j/2-2));
      p2=(v(j/2)-v(j/2-1));
      p3=(v(j/2+1)-v(j/2));
      p4=(v(j/2+2)-v(j/2+1));
      p5=(v(j/2+3)-v(j/2+2));
      s1=(p2-p1);
      s2=(p3-p2); as2=abs(s2);
      s3=(p4-p3); as3=abs(s3);
      s4=(p5-p4);

     if(as2<=as3)
        t1=(s2-s1); at1=abs(t1);
        t2=(s3-s2); at2=abs(t2);
        if(at2<=at1)
          f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
        else
          f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
        end
     else
         t2=(s3-s2); at2=abs(t2);
         t3=(s4-s3); at3=abs(t3);
         if(at2<=at3)
           f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
         else
```

```
            f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);
          end
      end

     end

   else

% PASO 2: Aplicamos ENO-H

   p1=(v(j/2-1)-v(j/2-2));
   p2=(v(j/2)-v(j/2-1));
   p3=(v(j/2+1)-v(j/2));
   p4=(v(j/2+2)-v(j/2+1));
   p5=(v(j/2+3)-v(j/2+2));
   s1=(p2-p1);
   s2=(p3-p2); as2=abs(s2);
   s3=(p4-p3); as3=abs(s3);
   s4=(p5-p4);

   if(as2<=as3)
          t1=(s2-s1); at1=abs(t1);
          t2=(s3-s2); at2=abs(t2);
          if(at2<=at1)
           f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
          else
           f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
          end
   else
          t2=(s3-s2); at2=abs(t2);
          t3=(s4-s3); at3=abs(t3);
          if(at2<=at3)
           f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
          else
           f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);
          end
   end

end

end
```

```
% predicción del antepenúltimo elemento: Con ENO jerárquico

p1=(v(n-4)-v(n-5));
p2=(v(n-3)-v(n-4));
p3=(v(n-2)-v(n-3));
p4=(v(n-1)-v(n-2));
p5=(v(n)-v(n-1));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);
s4=(p5-p4);

if(as2<=as3)
       t1=(s2-s1); at1=abs(t1);
       t2=(s3-s2); at2=abs(t2);
       if(at2<=at1)
           f(2*n-6)=(-v(n-4)+9*v(n-3)+9*v(n-2)-v(n-1))/16;
       else
           f(2*n-6)=(3/48)*v(n-5)-(15/48)*v(n-4)+(45/48)*v(n-3)+(15/48)*v(n-2);
       end
else
       t2=(s3-s2); at2=abs(t2);
       t3=(s4-s3); at3=abs(t3);
       if(at2<=at3)
           f(2*n-6)=(-v(n-4)+9*v(n-3)+9*v(n-2)-v(n-1))/16;
       else
           f(2*n-6)=(15/48)*v(n-3)+(45/48)*v(n-2)-(15/48)*v(n-1)+(3/48)*v(n);
       end
end

% predicción del penúltimo elemento

p1=(v(n-3)-v(n-4));
p2=(v(n-2)-v(n-3));
p3=(v(n-1)-v(n-2));
p4=(v(n)-v(n-1));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);

if(as3<=as2)
```

```
   f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
   t1=(s2-s1); at1=abs(t1);
   t2=(s3-s2); at2=abs(t2);
   if(at2<=at1)
        f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
   else
        f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
   end
end

% predicción del último elemento

f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

stencil.m

```
function op=stencil(v)

% op=stencil(v);
% op descentramiento del método ENO jerárquico: 'I' izquierda,
% 'C' centrado, 'D' derecha
% v vector de 6 elementos


p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
p5=(v(6)-v(5));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);
s4=(p5-p4);

if(as2<=as3)
       t1=(s2-s1); at1=abs(t1);
       t2=(s3-s2); at2=abs(t2);
       if(at2<=at1)
           op='C';
```

```
        else
            op='I';
        end
else
        t2=(s3-s2); at2=abs(t2);
        t3=(s4-s3); at3=abs(t3);
        if(at2<=at3)
            op='C';
        else
            op='D';
        end
end
```

5.4. Experimento 2: Código fuente de los algoritmos

zoom_tensor_exp2.m

```
function [a,c]=zoom_tensor_exp2(l,im,met)

% [a,c]=zoom_tensor_exp1(l,im,met,v);
% -------variables de salida-------
% a imagen original
% c zoom de la imagen original
% ------variables de entrada-------
% l son los niveles de zoom
% im imagen a utilizar:
% '../Imagenes_Test/squares2.pgm','../Imagenes_Test/geo5.pbm',
% '../Imagenes_Test/camera2.pgm', '../Imagenes_Test/lena5.pgm'
% met método a utilizar 'lin', 'eno', 'enh', 'wen', 'pph', 'esr'
% Los cálculos realizado son guardados en el fichero datos generados.rtf,
% que está en el directorio ../Fichero_Datos

a=imread(im);
a=double(a);

n=max(size(a));

figure('Tag','Original','Name',[blanks(6),'IMAGEN ORIGINAL'],...
    'Position',[2 305 560 420])
imagesc(a,[0 255])
title('imagen original');
colormap gray

% aumentamos la matriz a en una columna

y1=a(1:n,n);
aux=[a,y1];
clear y1;

% aumentamos la matriz a en una fila

y1=aux(n,1:n+1);
```

```
aux=[aux;y1];

aux1=descender(aux,l);
naux1=max(size(aux1))-1;

figure('Tag','Original','Name',[blanks(6),'IMAGEN A BAJA RESOLUCIÓN'],...
    'Position',[353 194 560 420])
imagesc(aux1(1:naux1,1:naux1),[0 255])
title('imagen a baja resolucion');
colormap gray

%t0=clock;
%t=cputime;

% ascendemos por la piramide de multiresolución

c=ascender(aux1,l,met);

%tiempo=etime(clock,t0);
%tiempo1=cputime-t;

% calculamos los errores

norma1=sum(sum(abs(a-c)))/n/n
infin=max(max(abs(a-c)))
norma2=sum(sum((a-c).^2))/n/n

% medimos cuanto de buena es la reconstrucción con MSE y PSNR
% MSE = (norm(a - a1,' fro')^2)/n1/n1 que es igual a norma2


MSE=norma2;
PSNR=20*log10(255/sqrt(MSE))

fid = fopen('../Fichero_Datos/datos generados.rtf','a');
fprintf(fid, 'Fecha y Hora de la Simulación= %s\n',datestr(now));
fprintf(fid, 'Niveles de Zoom= %d ',l);
fprintf(fid, 'Método= %s ',met);
fprintf(fid, 'Nombre de la Imagen= %s\n\n',im);
fprintf(fid,'norma1= %f\n',norma1);
fprintf(fid,'infin= %f\n',infin);
fprintf(fid,'norma2= %f\n',norma2);
```

```
fprintf(fid,'PSNR= %f\n\n',PSNR);
fclose(fid);

figure('Tag','Original','Name',[blanks(6),'ZOOM DE LA IMAGEN'],...
    'Position',[718 35 560 420])
imagesc(c,[0 255])
title('zoom de la imagen');
colormap gray
```

descender.m

```
function aux1=descender(aux,l)

% aux1=descender(aux,l);
% aux matriz original ampliada con fila y columna
% para aplicar subdivisión interpolatoria
% aux1 versión a menor resolución de la matriz:
% después de bajar l niveles
% l son los niveles de resolución que se desciende


nk=max(size(aux));

% descendiendo con decimación correspondiente al caso interpolatorio

for k=1:l
    nk1=(nk-1)/2+1;
    aux1=zeros(nk1);
    aux1(1:nk1,1:nk1)=aux(1:2:nk,1:2:nk);
    clear aux;
    aux=aux1;
    clear aux1;
    nk=nk1;
end

aux1=aux;
clear aux;
```

ascender.m

```
function c=ascender(a,l,met)

% c=ascender(a,l,met);
% c aproximación a la imagen
% a matriz a la que le aplicamos el algorimo ascendente de multiresolución
% l son los niveles de multiresolución
% met método que quieres utilizar 'lin', 'eno','enh', 'pph', 'esr'

% llamamos al método

n=max(size(a))-1;

if(met=='lin')
    c=decodifli(a,n,l);
end
if(met=='eno')
    c=decodifeno(a,n,l);
end
if(met=='enh')
    c=decodifenh(a,n,l);
end
if(met=='wen')
    c=decodifweno(a,n,l);
end
if(met=='pph')
    c=decodifpph(a,n,l);
end
if(met=='esr')
    c=decodifenosr(a,n,l);
end

% le quitamos la última fila y columna a la imagen

nl=2^l*n;
b=c(1:nl,1:nl);
clear c;
c=b;
clear b;
```

decodifli.m

```
function b=decodifli(a,nl,l)

% b=decodifli(a,nl,l);
% b zoom utilizando una reconstrucción lineal centrada
% con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% nl número de filas y columnas de la matriz a original
% l son los niveles de subdivisión (zoom)


for k=1:l

% se trata del bucle para los niveles de multiresolución

    nb=2*nl+1;
    b=zeros(nb);
    b(1:2:nb,1:2:nb)=a(1:nl+1,1:nl+1);

% se hace el algoritmo de multiresolución inverso para las columnas

    for j=1:2:nb
        b(1:nb,j)=linealde(b(1:2:nb,j),nl+1)';
    end

% se hace el algoritmo de multiresolución inverso para las filas

    for i=1:nb
        b(i,1:nb)=linealde(b(i,1:2:nb),nl+1);
    end

% se calcula el nl para subir al siguiente nivel

    nl=2*nl;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

linealde.m

```
function f=linealde(v,n)
```

```
% f=linealde(v,n);
% programa que sube un nivel de subdivisión utilizando una
% reconstrucción lineal centrada de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de
% la escala inferior
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción de los valores intermedios de f

for j=4:2:2*n-4
    f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
end

% predicción del último elemento

f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

decodifeno.m

```
function b=decodifeno(a,nl,l)

% b=decodifeno(a,nl,l);
% b zoom utilizando una reconstrucción eno no jerárquica
% con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% nl numero de filas y columnas de la matriz a original
% l son los niveles de subdivisión (zoom)


for k=1:l

% se trata del bucle para los niveles de multiresolución
```

```
    nb=2*nl+1;
    b=zeros(nb);
    b(1:2:nb,1:2:nb)=a(1:nl+1,1:nl+1);

% se hace el algoritmo de multiresolución inverso para las columnas

    for j=1:2:nb
        b(1:nb,j)=enode(b(1:2:nb,j),nl+1)';
    end

% se hace el algoritmo de multiresolución inverso para las filas

    for i=1:nb
        b(i,1:nb)=enode(b(i,1:2:nb),nl+1);
    end

% se calcula el nl para subir al siguiente nivel

    nl=2*nl;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

enode.m

```
function f=enode(v,n)

% f=enode(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción eno no jerárquica de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de la
% escala inferior y los detalles para subir
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f
```

```
f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción del segundo elemento

at2=abs(-v(1)+3*v(2)-3*v(3)+v(4));
at3=abs(-v(2)+3*v(3)-3*v(4)+v(5));

if(at2<=at3)
    f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
    f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
end

% predicción de los valores intermedios de f

for j=6:2:2*n-6
   at1=abs(-v(j/2-2)+3*v(j/2-1)-3*v(j/2)+v(j/2+1));
   at2=abs(-v(j/2-1)+3*v(j/2)-3*v(j/2+1)+v(j/2+2));
   at3=abs(-v(j/2)+3*v(j/2+1)-3*v(j/2+2)+v(j/2+3));

   if(at2<=at1 & at2<=at3)
       f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
   else
     if(at1<=at3)
       f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
     else
       f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);
     end
   end
end

% predicción del penúltimo elemento

at1=abs(-v(n-4)+3*v(n-3)-3*v(n-2)+v(n-1));
at2=abs(-v(n-3)+3*v(n-2)-3*v(n-1)+v(n));

if(at2<=at1)
   f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
   f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
end

% predicción del último elemento
```

```
f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

decodifenh.m

```
function b=decodifenh(a,nl,l)

% b=decodifenh(a,nl,l);
% b zoom utilizando una reconstrucción eno jerarquica
% con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% nl número de filas y columnas de la matriz a original
% l son los niveles de subdivisión (zoom)


for k=1:l

% se trata del bucle para los niveles de multiresolución

    nb=2*nl+1;
    b=zeros(nb);
    b(1:2:nb,1:2:nb)=a(1:nl+1,1:nl+1);

% se hace el algoritmo de multiresolución inverso para las columnas

    for j=1:2:nb
        b(1:nb,j)=enohde(b(1:2:nb,j),nl+1)';
    end

% se hace el algoritmo de multiresolución inverso para las filas

    for i=1:nb
        b(i,1:nb)=enohde(b(i,1:2:nb),nl+1);
    end

% se calcula el nl para subir al siguiente nivel

    nl=2*nl;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

enohde.m

```
function f=enohde(v,n)

% f=enohde(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción eno jerarquica de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de la
% escala inferior y los detalles para subir
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción del segundo elemento

p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
s1=(p2-p1); as1=abs(s1);
s2=(p3-p2); as2=abs(s2);

if(as1<=as2)
   f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
   s3 =(p4-p3);
   t1=(s2-s1); at1=abs(t1);
   t2=(s3-s2); at2=abs(t2);
   if(at1<=at2)
      f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
   else
      f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
   end
end
```

```
% predicción de los valores intermedios de f

for j=6:2:2*n-6

 p1=(v(j/2-1)-v(j/2-2));
 p2=(v(j/2)-v(j/2-1));
 p3=(v(j/2+1)-v(j/2));
 p4=(v(j/2+2)-v(j/2+1));
 p5=(v(j/2+3)-v(j/2+2));
 s1=(p2-p1);
 s2=(p3-p2); as2=abs(s2);
 s3=(p4-p3); as3=abs(s3);
 s4=(p5-p4);

 if(as2<=as3)
      t1=(s2-s1); at1=abs(t1);
      t2=(s3-s2); at2=abs(t2);
      if(at2<=at1)
        f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
      else
        f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
      end
 else
      t2=(s3-s2); at2=abs(t2);
      t3=(s4-s3); at3=abs(t3);
      if(at2<=at3)
         f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
      else
          f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);
      end
 end

end

% predicción del penúltimo elemento

p1=(v(n-3)-v(n-4));
p2=(v(n-2)-v(n-3));
p3=(v(n-1)-v(n-2));
p4=(v(n)-v(n-1));
s1=(p2-p1);
```

```
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);

if(as3<=as2)
   f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
   t1=(s2-s1); at1=abs(t1);
   t2=(s3-s2); at2=abs(t2);
   if(at2<=at1)
        f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
   else
        f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
   end
end

% predicción del último elemento

f(2*n-2)=(3/48)*v(n-3)-(15/48)*v(n-2)+(45/48)*v(n-1)+(15/48)*v(n);
```

decodifweno.m

```
function b=decodifweno(a,nl,l)

% b=decodifweno(a,nl,l);
% b zoom utilizando una reconstrucción eno de cuarto orden
% a matriz a la que se le va a aplicar el zoom
% nl numero de filas y columnas de la matriz a original
% l son los niveles de subdivisión (zoom)


for k=1:l

% se trata del bucle para los niveles de multiresolución

    nb=2*nl+1;
    b=zeros(nb);
    b(1:2:nb,1:2:nb)=a(1:nl+1,1:nl+1);

% se hace el algoritmo de multiresolución inverso para las columnas

       for j=1:2:nb
          b(1:nb,j)=wenode(b(1:2:nb,j),nl+1)';
       end
```

```
% se hace el algoritmo de multiresolución inverso para las filas
       for i=1:nb
          b(i,1:nb)=wenode(b(i,1:2:nb),nl+1);
       end
% se calcula el nl para subir al siguiente nivel
    nl=2*nl;
% se inicializa la matriz de la que se hace zoom
    a=b;
end
```

wenode.m

```
function f=wenode(v,n)
% f=wenode(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción weno de cuarto orden
% f vector de la escala superior
% v vector que contiene los valores significativos de la
% escala inferior y los detalles para subir
% n dimensión de v

f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);
% predicción del primer elemento de f
f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);
% predicción del segundo elemento
at2=abs(-v(1)+3*v(2)-3*v(3)+v(4));
at3=abs(-v(2)+3*v(3)-3*v(4)+v(5));

if(at2<=at3)
   f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
   f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
end
```

```
% predicción de los valores intermedios de f

epsil=10^(-6);

for j=6:2:2*n-6

% cálculo de las diferencias divididas

  at1=v(j/2-1)-v(j/2-2);
  at2=v(j/2)-v(j/2-1);
  at3=v(j/2+1)-v(j/2);
  at4=v(j/2+2)-v(j/2+1);
  at5=v(j/2+3)-v(j/2+2);

% cálculo de los indicadores de suavidad

  inleft=0.5*((at2-at1)^2+(at3-at2)^2)+(at3-2*at2+at1)^2;
  incenter=0.5*((at3-at2)^2+(at4-at3)^2)+(at4-2*at3+at2)^2;
  inright=0.5*((at4-at3)^2+(at5-at4)^2)+(at5-2*at4+at3)^2;

% cálculo de los numeradores de los pesos

  alfaleft=3./16/(epsil+inleft)^3;
  alfacenter=10./16/(epsil+incenter)^3;
  alfaright=3./16/(epsil+inright)^3;

% cálculo de los pesos

  alfasum=alfaleft+alfacenter+alfaright;
  wleft=alfaleft/alfasum;
  wcenter=alfacenter/alfasum;
  wright=alfaright/alfasum;

% sumamos los tres polinomios multiplicados por sus pesos

  qcenter=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
  qleft=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
  qright=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);

  f(j)=wleft*qleft+wcenter*qcenter+wright*qright;

end

% predicción del penúltimo elemento
```

```
at1=abs(-v(n-4)+3*v(n-3)-3*v(n-2)+v(n-1));
at2=abs(-v(n-3)+3*v(n-2)-3*v(n-1)+v(n));

if(at2<=at1)
   f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
   f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
end

% predicción del último elemento

f(2*n-2)=(3/48)*v(n-3)-(15/48)*v(n-2)+(45/48)*v(n-1)+(15/48)*v(n);
```

decodifpph.m

```
function b=decodifpph(a,nl,l)

% b=decodif1(a,nl,l);
% b zoom utilizando una reconstrucción pph con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% nl número de filas y columnas de la matriz a original
% l son los niveles de subdivisión (zoom)


for k=1:l

% se trata del bucle para los niveles de multiresolución

    nb=2*nl+1;
    b=zeros(nb);
    b(1:2:nb,1:2:nb)=a(1:nl+1,1:nl+1);

% se hace el algoritmo de multiresolución inverso para las columnas

    for j=1:2:nb
        b(1:nb,j)=pphde(b(1:2:nb,j),nl+1)';
    end

% se hace el algoritmo de multiresolución inverso para las filas

    for i=1:nb
        b(i,1:nb)=pphde(b(i,1:2:nb),nl+1);
    end
```

```
% se calcula el nl para subir al siguiente nivel

    nl=2*nl;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

pphde.m

```
function f=pphde(v,n)

% f=pphde(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción pph de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos de la escala
% inferior y los detalles para subir
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción de los valores intermedios de f

for j=4:2:2*n-4
    d1=v(j/2-1)-2*v(j/2)+v(j/2+1);
    d2=v(j/2)-2*v(j/2+1)+v(j/2+2);
    d=d1*d2;
    s=(v(j/2)+v(j/2+1))/2;
    if (d>0)
        f(j)=s-d/(4*(d1+d2));
    else
        f(j)=s;
    end
end
```

```
% predicción del último elemento

f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

decodifenosr.m

```
function b=decodifenosr(a,nl,l)

% b=decodifenosr(a,nl,l);
% b zoom utilizando una reconstrucción eno S-R con 4 puntos
% a matriz a la que se le va a aplicar el zoom
% nl número de filas y columnas de la matriz a original
% l son los niveles de subdivisión (zoom)


for k=1:l

% se trata del bucle para los niveles de multiresolución

    nb=2*nl+1;
    b=zeros(nb);
    b(1:2:nb,1:2:nb)=a(1:nl+1,1:nl+1);

% se hace el algoritmo de multiresolución inverso para las columnas

    for j=1:2:nb
        b(1:nb,j)=enosrde(b(1:2:nb,j),nl+1)';
    end

% se hace el algoritmo de multiresolución inverso para las filas

    for i=1:nb
        b(i,1:nb)=enosrde(b(i,1:2:nb),nl+1);
    end

% se calcula el nl para subir al siguiente nivel

    nl=2*nl;

% se inicializa la matriz de la que se hace zoom

    a=b;
end
```

enosrde.m

```
function f=enosrde(v,n)

% f=enosrde(v,n);
% programa que sube un nivel de multiresolución utilizando una
% reconstrucción eno S-R de 4 puntos
% f vector de la escala superior
% v vector que contiene los valores significativos
% de la escala inferior
% n dimensión de v


f=zeros(1,2*n-1);
f(1:2:2*n-1)=v(1:n);

% predicción del primer elemento de f

f(2)=(15/48)*v(1)+(45/48)*v(2)-(15/48)*v(3)+(3/48)*v(4);

% predicción del segundo elemento

p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
s1=(p2-p1); as1=abs(s1);
s2=(p3-p2); as2=abs(s2);

if(as1<=as2)
    f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
else
    s3 =(p4-p3);
    t1=(s2-s1); at1=abs(t1);
    t2=(s3-s2); at2=abs(t2);
    if(at1<=at2)
        f(4)=(-v(1)+9*v(2)+9*v(3)-v(4))/16;
    else
        f(4)=(15/48)*v(2)+(45/48)*v(3)-(15/48)*v(4)+(3/48)*v(5);
    end
end
```

```
% predicción del tercer elemento: con ENO jerarquico

p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
p5=(v(6)-v(5));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);
s4=(p5-p4);

if(as2<=as3)
    t1=(s2-s1); at1=abs(t1);
    t2=(s3-s2); at2=abs(t2);
    if(at2<=at1)
        f(6)=(-v(2)+9*v(3)+9*v(4)-v(5))/16;
    else
        f(6)=(3/48)*v(1)-(15/48)*v(2)+(45/48)*v(3)+(15/48)*v(4);
    end
else
    t2=(s3-s2); at2=abs(t2);
    t3=(s4-s3); at3=abs(t3);
    if(at2<=at3)
        f(6)=(-v(2)+9*v(3)+9*v(4)-v(5))/16;
    else
        f(6)=(15/48)*v(3)+(45/48)*v(4)-(15/48)*v(5)+(3/48)*v(6);
    end
end

% predicción de los valores intermedios de f

for j=8:2:2*n-8

% PASO 1:Para cada intervalo determinamos los stencil a la izquierda
% y a la derecha. Si no se descentra aplicamos eno-h.

    stencil_izq=stencil(v(j/2-3:j/2+2));
    stencil_der=stencil(v(j/2-1:j/2+4));

    if(stencil_izq=='I' & stencil_der=='D')
```

```
% PASO 3: Continuamos con el método de resolución subcelda
% comprobamos si g(x_i) * g(x_{i+1}) < 0
% valores de los polinomios q_{i-1}(x_i), q_{i-1}(x_{i+1}),
% q_{i+1}(x_i), q_{i+1}(x_{i+1}) respectivamente

        q11=v(j/2);
        q12=-v(j/2-3)+4*v(j/2-2)-6*v(j/2-1)+4*v(j/2);
        q21=4*v(j/2+1)-6*v(j/2+2)+4*v(j/2+3)-v(j/2+4);
        q22=v(j/2+1);

% valores de la función g en x_i y x_{i+1}

        g1=q21-q11;
        g2=q22-q12;

        if(g1*g2 < 0)

% PASO 4: Efectivamente hay una discontinuidad en el intervalo y
% extendemos q_{i-1}(x) y q_{i+1}(x)
% polinomios q_{i-1}(punto medio), q_{i+1}(punto medio)


            q1m=-5/16*v(j/2-3)+21/16*v(j/2-2)-35/16*v(j/2-1)+35/16*v(j/2);
            q2m=35/16*v(j/2+1)-35/16*v(j/2+2)+21/16*v(j/2+3)-5/16*v(j/2+4);

% función g evaluada en el punto medio

            gm=q2m-q1m;

            if (g1*gm <0)
                f(j)=q2m;
            else
                f(j)=q1m;
            end

   else

% PASO 2: Hace eno jerárquico

    p1=(v(j/2-1)-v(j/2-2));
    p2=(v(j/2)-v(j/2-1));
    p3=(v(j/2+1)-v(j/2));
    p4=(v(j/2+2)-v(j/2+1));
```

```
    p5=(v(j/2+3)-v(j/2+2));
    s1=(p2-p1);
    s2=(p3-p2); as2=abs(s2);
    s3=(p4-p3); as3=abs(s3);
    s4=(p5-p4);

    if(as2<=as3)
      t1=(s2-s1); at1=abs(t1);
      t2=(s3-s2); at2=abs(t2);

      if(at2<=at1)
        f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
      else
        f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
      end

    else
       t2=(s3-s2); at2=abs(t2);
       t3=(s4-s3); at3=abs(t3);

       if(at2<=at3)
         f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
       else
         f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3);
       end
    end

 end

   else
% PASO 2: Aplicamos eno-h

    p1=(v(j/2-1)-v(j/2-2));
    p2=(v(j/2)-v(j/2-1));
    p3=(v(j/2+1)-v(j/2));
    p4=(v(j/2+2)-v(j/2+1));
    p5=(v(j/2+3)-v(j/2+2));
    s1=(p2-p1);
    s2=(p3-p2); as2=abs(s2);
```

```
    s3=(p4-p3); as3=abs(s3);
    s4=(p5-p4);

    if(as2<=as3)
       t1=(s2-s1); at1=abs(t1);
       t2=(s3-s2); at2=abs(t2);
       if(at2<=at1)
          f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
       else
          f(j)=(3/48)*v(j/2-2)-(15/48)*v(j/2-1)+(45/48)*v(j/2)+(15/48)*v(j/2+1);
       end
    else
        t2=(s3-s2); at2=abs(t2);
        t3=(s4-s3); at3=abs(t3);
        if(at2<=at3)
           f(j)=(-v(j/2-1)+9*v(j/2)+9*v(j/2+1)-v(j/2+2))/16;
        else
           f(j)=(15/48)*v(j/2)+(45/48)*v(j/2+1)-(15/48)*v(j/2+2)+(3/48)*v(j/2+3)
        end
    end
  end
end

% predicción del antepenúltimo elemento: Con ENO jerárquico

p1=(v(n-4)-v(n-5));
p2=(v(n-3)-v(n-4));
p3=(v(n-2)-v(n-3));
p4=(v(n-1)-v(n-2));
p5=(v(n)-v(n-1));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);
s4=(p5-p4);

if(as2<=as3)
   t1=(s2-s1); at1=abs(t1);
   t2=(s3-s2); at2=abs(t2);
   if(at2<=at1)
     f(2*n-6)=(-v(n-4)+9*v(n-3)+9*v(n-2)-v(n-1))/16;
   else
```

```
    f(2*n-6)=(3/48)*v(n-5)-(15/48)*v(n-4)+(45/48)*v(n-3)+(15/48)*v(n-2);
   end

else
   t2=(s3-s2); at2=abs(t2);
   t3=(s4-s3); at3=abs(t3);
   if(at2<=at3)
     f(2*n-6)=(-v(n-4)+9*v(n-3)+9*v(n-2)-v(n-1))/16;
   else
     f(2*n-6)=(15/48)*v(n-3)+(45/48)*v(n-2)-(15/48)*v(n-1)+(3/48)*v(n);
   end
end

% predicción del penúltimo elemento

p1=(v(n-3)-v(n-4));
p2=(v(n-2)-v(n-3));
p3=(v(n-1)-v(n-2));
p4=(v(n)-v(n-1));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);

if(as3<=as2)
  f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
else
  t1=(s2-s1); at1=abs(t1);
  t2=(s3-s2); at2=abs(t2);
  if(at2<=at1)
    f(2*n-4)=(-v(n-3)+9*v(n-2)+9*v(n-1)-v(n))/16;
  else
    f(2*n-4)=(3/48)*v(n-4)-(15/48)*v(n-3)+(45/48)*v(n-2)+(15/48)*v(n-1);
  end
end

% predicción del último elemento

f(2*n-2)=(15/48)*v(n)+(45/48)*v(n-1)-(15/48)*v(n-2)+(3/48)*v(n-3);
```

stencil.m

```
function op=stencil(v)

% op=stencil(v);
% op descentramiento del método ENO jerarquico: 'I' izquierda,
% 'C' centrado, 'D' derecha
% v vector de 6 elementos


p1=(v(2)-v(1));
p2=(v(3)-v(2));
p3=(v(4)-v(3));
p4=(v(5)-v(4));
p5=(v(6)-v(5));
s1=(p2-p1);
s2=(p3-p2); as2=abs(s2);
s3=(p4-p3); as3=abs(s3);
s4=(p5-p4);

if(as2<=as3)
       t1=(s2-s1); at1=abs(t1);
       t2=(s3-s2); at2=abs(t2);
       if(at2<=at1)
           op='C';
       else
           op='I';
       end
else
       t2=(s3-s2); at2=abs(t2);
       t3=(s4-s3); at3=abs(t3);
       if(at2<=at3)
           op='C';
       else
           op='D';
       end
end
```

5.5. Experimento 1: Código fuente de la interfaz gráfica

Sobre el código fuente de la interfaz gráfica los programas anteriormente vistos, se utilizan también para el propósito de esta sección, en el que todos los ficheros fuente son iguales a excepción de zoom_tensor_exp1.m y además de GUIdemo.m que se utiliza para la interfaz.

zoom_tensor_exp1.m

```
function [a,c]=zoom_tensor_exp1(l,im,met,v)

% [a,c]=zoom_tensor_exp1(l,im,met,v);
% -------variables de salida-------
% a imagen original
% c zoom de la imagen original
% ------variables de entrada-------
% l son los niveles de zoom
% im imagen a utilizar:
% 'squares2.pgm', 'geo5.pbm', 'camera2.pgm', 'lena5.pgm', etc...
% met método a utilizar 'lin', 'eno', 'enh', 'wen', 'pph', 'esr'
% v = zona de la imagen a hacer zoom



% comprobación de que se ha introducido correctamente el
% nombre de la imagen

if strcmp(im,'')
    a=0; c=0;
    return
end

% comprobación de que se han introducido correctamente los
% niveles de zoom

if l==-1
    a=0; c=0;
    return
end
```

```
% comprobación de que se ha introducido correctamente el método

if met=='Err'
    a=0; c=0;
    return
end

% comprobación de que se ha introducido correctamente
% la zona para hacer zoom

if length(find(v<0)~=0)
    a=0; c=0;
    return
end

% definición de la matriz, la guardamos en a

b=imread(im);
[n,m]=size(b);
b=double(b);
a=b(v(3):v(4),v(1):v(2));
clear b;

% ascendemos por la piramide de multiresolución

c=ascender(a,l,met);

% Dibujamos una nueva figura

figure('Tag','zoom','Name',[blanks(6),'ZOOM DE LA IMAGEN',...
    blanks(2),'METODO',blanks(2),upper(met),blanks(2),'NIVELES',...
    blanks(2),num2str(l)],'Position',[6 100 710 690])
h=axes('Position',[0 0 1 1],'Visible','off');

% mostramos el zoom de la imagen dentro de la nueva figura

dx=v(2)-v(1);
dy=v(4)-v(3);

ax=(0.87*dx)/n;
ay=(0.87*dy)/m;

for i=1:l
    ax=2*ax;
```

```
    ay=2*ay;
end

if ax>ay
    if ax>=0.87
        ax=0.87;
        ay=(dy*ax)/dx;
    end
else
    if ay>0.87
        ay=0.87;
        ax=(dx*ay)/dy;
    end
end

% Centrar imagen

centerX= 0.075+(0.87-ax)/2;
centerY= 0.035+(0.87-ay)/2;

axes('Position',[centerX centerY ax  ay])
imagesc(c,[0 255])
title('zoom de la imagen','Color','r');
colormap gray

% incluimos un texto explicativo en la nueva figura

str=['Método:',blanks(2),met,blanks(10),'Imagen:',blanks(2),im,...
    blanks(10),'Zoom:',blanks(2),num2str(l),...
    blanks(10),'Zona:',blanks(2),'[',blanks(1),...
    num2str(v),blanks(1),']'];
set(gcf,'CurrentAxes',h)
text(.025,.97,str,'FontSize',12)
```

GUIdemo.m

```
function varargout = GUIdemo(varargin)

%function varargout = GUIdemo(varargin)
% GUIDEMO M-file for GUIdemo.fig
% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUIdemo_OpeningFcn, ...
                   'gui_OutputFcn',  @GUIdemo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

function GUIdemo_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for GUIdemo

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% Colocar imagen de fondo

background=imread('background.jpg');
axes (handles.background);
axis off;
imshow (background);
handles.output=hObject;
guidata (hObject, handles);

function varargout = GUIdemo_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure

varargout{1} = handles.output;
```

```
function inEdit_CreateFcn(hObject, eventdata, handles)

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function inEdit_Callback(hObject, eventdata, handles)

% Hints: get(hObject,'String') returns contents of inEdit as text
% str2double(get(hObject,'String')) returns contents inEdit double

function loadPush_Callback(hObject, eventdata, handles)

image_file = get(handles.inEdit,'String');
image_file=fliplr(deblank(fliplr(deblank(image_file))));

file_dir_struct=dir('../Gui_Zoom_Exp1');
file_dir_cell=struct2cell(file_dir_struct);
file_dir_line=file_dir_cell(1,:);
coincidences=strcmp(image_file,file_dir_line);

if (nnz(coincidences)==0)
    uiwait(msgbox('Imagen no válida','Mensaje de error','error','modal'))
else
    im_original=imread(char(image_file));
    set(handles.orgIm,'HandleVisibility','ON');
    axes(handles.orgIm);
    imagesc(im_original,[0,255]); colormap gray;
    axis tight;
    grid on;
end

% ------ BOTÓN VER ZONA ZOOM ------
% --- Executes on button press in buttonverzona.

function buttonverzona_Callback(hObject, eventdata, handles)
```

```
% Volvemos a dibujar la imagen

image_file = get(handles.inEdit,'String');
im_original=imread(char(image_file));
[n,m]=size(im_original);
set(handles.orgIm,'HandleVisibility','ON');
axes(handles.orgIm);
imagesc(im_original,[0 255]); colormap gray;
axis tight;
grid on;

% Lee el texto de la casilla del zoom

zonazoom1 = get(handles.edit5,'String');
zonazoom=sscanf(zonazoom1, '%f');

if (length(zonazoom)~=4)
    uiwait(msgbox('Debe introduzir la zona de zoom [x1,x2,y1,y2]...
        apropiadamente', 'Mensaje de error','error','modal'))
    zonazoom=-1*ones(1,4);
elseif (length(find(zonazoom<0))~=0 | zonazoom(1)>=zonazoom(2) |...
        zonazoom(3)>=zonazoom(4) | zonazoom(2)>n | zonazoom(4)>m)
    uiwait(msgbox('Debe introduzir la zona de zoom [x1,x2,y1,y2]...
        apropiadamente', 'Mensaje de error','error','modal'))
    zonazoom=-1*ones(1,4);
else
    zonazoom=round(zonazoom);
    rect_posicion=[zonazoom(1),zonazoom(3),zonazoom(2)-zonazoom(1),...
        zonazoom(4)-zonazoom(3)];
    rectangle('Position',rect_posicion,...
       'LineWidth',3,...
       'EdgeColor',[1 0 0],...
       'LineStyle','-');
end

% ------ BOTÓN APPLY ------

function appPush_Callback(hObject, eventdata, handles)

% Nombre de la imagen a hacer zoom y rutina para corregir espacios
% y si la imagen se encuentra dentro del directorio
```

```
image_file = get(handles.inEdit,'String');
image_file=fliplr(deblank(fliplr(deblank(image_file))));

file_dir_struct=dir('../Gui_Zoom_Exp1');
file_dir_cell=struct2cell(file_dir_struct);
file_dir_line=file_dir_cell(1,:);
coincidences=strcmp(image_file,file_dir_line);

if (nnz(coincidences)==0)
    uiwait(msgbox('Imagen no válida','Mensaje de error','error','modal'))
    image_file='';
    [n,m]=zeros(1,2);
else
    im_original=imread(char(image_file));
    [n,m]=size(im_original);
    set(handles.orgIm,'HandleVisibility','ON');
    axes(handles.orgIm);
    imagesc(im_original,[0 255]); colormap gray;
    axis tight;
    grid on;
end

% Rutina para comprobar el método utilizado

if (get(handles.radiobutton1,'Value')==1)
    metodo='lin';
elseif (get(handles.radiobutton2,'Value')==1)
    metodo='enh';
elseif (get(handles.radiobutton3,'Value')==1)
    metodo='eno';
elseif (get(handles.radiobutton4,'Value')==1)
    metodo='wen';
elseif (get(handles.radiobutton5,'Value')==1)
    metodo='pph';
elseif (get(handles.radiobutton6,'Value')==1)
    metodo='esr';
else
    uiwait(msgbox('Debe seleccionar un método','Mensaje de error',...
        'error','modal'))
    metodo='Err';
end
```

```
% Rutina para comprobar el nivel de zoom

if (get(handles.radiobutton7,'Value')==1)
    nivel=1;
elseif (get(handles.radiobutton8,'Value')==1)
    nivel=2;
elseif (get(handles.radiobutton9,'Value')==1)
    nivel=3;
elseif (get(handles.radiobutton10,'Value')==1)
    nivel=4;
elseif (get(handles.radiobutton11,'Value')==1)
    nivel=5;
elseif (get(handles.radiobutton12,'Value')==1)
    nivel=6;
else
    uiwait(msgbox('Debe seleccionar los niveles de zoom ',...
        'Mensaje de error','error','modal'))
    nivel=-1;
end

% Extraemos los datos de la zona a la que vamos a hacer zoom

zonazoom1 = get(handles.edit5,'String');
zonazoom=sscanf(zonazoom1, '%f');

if (length(zonazoom)~=4)
    uiwait(msgbox('Debe introduzir la zona de zoom [x1,x2,y1,y2]...
        apropiadamente','Mensaje de error','error','modal'))

    zonazoom=-1*ones(1,4);
elseif (length(find(zonazoom<0))~=0 | zonazoom(1)>=zonazoom(2) |...
        zonazoom(3)>=zonazoom(4) | zonazoom(2)>n | zonazoom(4)>m)

    uiwait(msgbox('Debe introduzir la zona de zoom [x1,x2,y1,y2]...
        apropiadamente','Mensaje de error','error','modal'))

    zonazoom=-1*ones(1,4);
else
    zonazoom=round(zonazoom);
    rect_posicion=[zonazoom(1),zonazoom(3),zonazoom(2)-zonazoom(1),...
        zonazoom(4)-zonazoom(3)];
```

```
    rectangle('Position',rect_posicion,...
        'LineWidth',3,...
        'EdgeColor',[1 0 0],...
        'LineStyle','-');
end

% Llamada a la funcion a hacer zoom

[a,c]=zoom_tensor_exp1(nivel,image_file,metodo,zonazoom);

% ------ PANEL METODO A UTILIZAR:Lineal,Eno.... ------
% --- Executes on button press in radiobuttonX.
% ------ LINEAL ------

function radiobutton1_Callback(hObject, eventdata, handles)

set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ ENO JERÁRQUICO ------

function radiobutton2_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ ENO NO JERÁRQUICO ------

function radiobutton3_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ WENO ------
```

```
function radiobutton4_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ PPH ------

function radiobutton5_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ ENO SUBCELL RESOLUTION ------

function radiobutton6_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);

% ------ PANEL NIVEL DE ZOOM ------
% --- Executes on button press in radiobuttonX.
% ------ 1 NIVEL DE ZOOM ------

function radiobutton7_Callback(hObject, eventdata, handles)

set (handles.radiobutton8, 'value',0);
set (handles.radiobutton9, 'value',0);
set (handles.radiobutton10, 'value',0);
set (handles.radiobutton11, 'value',0);
set (handles.radiobutton12, 'value',0);

% ------ 2 NIVELES DE ZOOM ------
```

```
function radiobutton8_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
set (handles.radiobutton9, 'value',0);
set (handles.radiobutton10, 'value',0);
set (handles.radiobutton11, 'value',0);
set (handles.radiobutton12, 'value',0);

% ------ 3 NIVELES DE ZOOM ------

function radiobutton9_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
set (handles.radiobutton8, 'value',0);
set (handles.radiobutton10, 'value',0);
set (handles.radiobutton11, 'value',0);
set (handles.radiobutton12, 'value',0);

% ------ 4 NIVELES DE ZOOM ------

function radiobutton10_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
set (handles.radiobutton8, 'value',0);
set (handles.radiobutton9, 'value',0);
set (handles.radiobutton11, 'value',0);
set (handles.radiobutton12, 'value',0);

% ------ 5 NIVELES DE ZOOM ------

function radiobutton11_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
set (handles.radiobutton8, 'value',0);
set (handles.radiobutton9, 'value',0);
set (handles.radiobutton10, 'value',0);
set (handles.radiobutton12, 'value',0);

% ------ 6 NIVELES DE ZOOM ------

function radiobutton12_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
```

```
set (handles.radiobutton8, 'value',0);
set (handles.radiobutton9, 'value',0);
set (handles.radiobutton10, 'value',0);
set (handles.radiobutton11, 'value',0);

   % Función donde introducimos la zona a aplicar el Zoom

function edit5_Callback(hObject, eventdata, handles)

% Hints: get(hObject,'String') returns contents of edit5 as text
% str2double(get(hObject,'String')) returns contents edit5 (double)

function edit5_CreateFcn(hObject, eventdata, handles)

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.

if ispc
  set(hObject,'BackgroundColor','white');
else
  set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% ------ BOTÓN DE SALIR DEL PROGRAMA ------

function closePush_Callback(hObject, eventdata, handles)

closePush=questdlg('¿Desea Salir del Programa?',...
    'Salida del Programa','Si','No','No');

switch closePush
    case 'Si'
        close all;
    case 'No'
        return;
end

% --------------------------------------------------------------------

function archivo_Callback(hObject, eventdata, handles)

% hObject handle to archivo (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
```

```
function ayuda_Callback(hObject, eventdata, handles)

% hObject handle to ayuda (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------

function abrir_Callback(hObject, eventdata, handles)

% hObject handle to abrir (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)


fileName=uigetfile( ...
{'*.pgm;*.pbm','Archivos de Imagen (*.pgm,*.pbm)';
   '*.pgm', 'Portable Graymap (*.pgm)'; ...
   '*.pbm', 'Portable Bitmap (*.pbm)'},'Pick a file');

set (handles.inEdit,'String',fileName);
image_file = get(handles.inEdit,'String');
if ~isempty(image_file)
   im_original=imread(char(image_file));
   set(handles.orgIm,'HandleVisibility','ON');
   axes(handles.orgIm);
   imagesc(im_original,[0,255]); colormap gray;
   axis tight;
   grid on;
end;

% --------------------------------------------------------------------

function salir_Callback(hObject, eventdata, handles)

% hObject handle to salir (see GCBO) % eventdata reserved - to
be defined in a future version of MATLAB % handles structure with
handles and user data (see GUIDATA)

salir=questdlg('¿Desea Salir del Programa?',...
    'Salida del Programa','Si','No','No');

switch salir
    case 'Si'
```

```
        close all;
    case 'No'
        return;
end

% --------------------------------------------------------------------

function acerca_Callback(hObject, eventdata, handles)

% hObject handle to acerca (see GCBO) % eventdata reserved - to
be defined in a future version of MATLAB % handles structure with
handles and user data (see GUIDATA)

fid = fopen('acerca de.txt');
str = fread(fid,inf,'*char').';
fclose(fid);
f = figure('menubar','none','NumberTitle','off',...
    'Name','Acerca de ZoomProgram');

h=uicontrol('style','edit','parent',f,'min',0,'max',...
    2,'enable','inactive','units','normalized','position',...
    [0 0 1 1],'string',str,'horizontalalignment','left','fontsize',10);
drawnow

% --------------------------------------------------------------------

function ayudaprog_Callback(hObject, eventdata, handles)

% hObject handle to ayudaprog (see GCBO) % eventdata reserved -
to be defined in a future version of MATLAB % handles structure
with handles and user data (see GUIDATA)

fid = fopen('ayuda.txt');
str = fread(fid,inf,'*char').';
fclose(fid);
f = figure('menubar','none','NumberTitle','off',...
    'Name','Ayuda ZoomProgram');

h=uicontrol('style','edit','parent',f,'min',0,'max',2,...
    'enable','inactive','units','normalized','position',...
    [0 0 1 1],'string',str,'horizontalalignment','left','fontsize',10);
drawnow
```

5.6. Experimento 2: Código fuente de la interfaz gráfica

Sobre el código fuente de la interfaz gráfica los programas anteriormente vistos, se utilizan también para el propósito de esta sección, en el que todos los ficheros fuente son iguales a excepción de zoom_tensor_exp2.m y además de GUIdemo.m que se utiliza para la interfaz.

zoom_tensor_exp2.m

```
function [a,c]=zoom_tensor_exp2(l,im,met)

% [a,c]=zoom_tensor_exp1(l,im,met,v);
% -------variables de salida-------
% a imagen original
% c zoom de la imagen original
% ------variables de entrada-------
% l son los niveles de zoom
% im imagen a utilizar:
% 'squares2.pgm', 'geo5.pbm', 'camera2.pgm', 'lena5.pgm', etc...
% met método a utilizar 'lin', 'eno', 'enh', 'wen', 'pph', 'esr'
% Los cálculos realizado son guardados en el fichero datos generados.rtf,
% que está en el directorio ../Fichero_Datos


% Definición de la matriz, la guardamos en a

a=imread(im);
a=double(a);

n=max(size(a));

figure('Tag','Original','Name',[blanks(6),'IMAGEN ORIGINAL'],...
    'Position',[2 305 560 420])
imagesc(a,[0 255])
title('imagen original');
colormap gray
```

```
% aumentamos la matriz a en una columna

y1=a(1:n,n);
aux=[a,y1];
clear y1;

% aumentamos la matriz a en una fila

y1=aux(n,1:n+1);
aux=[aux;y1];

aux1=descender(aux,l);
naux1=max(size(aux1))-1;

figure('Tag','Original','Name',[blanks(6),'IMAGEN A BAJA RESOLUCIÓN'],...
    'Position',[353 194 560 420])
imagesc(aux1(1:naux1,1:naux1),[0 255])
title('imagen a baja resolucion');
colormap gray

% ascendemos por la piramide de multiresolución

c=ascender(aux1,l,met);

% calculamos los errores

norma1=sum(sum(abs(a-c)))/n/n;
infin=max(max(abs(a-c)));
norma2=sum(sum((a-c).^2))/n/n;

% medimos cuanto de buena es la reconstrucción con MSE y PSNR


MSE=norma2;
PSNR=20*log10(255/sqrt(MSE))

fid = fopen('../Fichero_Datos/datos generados.rtf','a');
fprintf(fid, 'Fecha y Hora de la Simulación= %s\n',datestr(now));
fprintf(fid, 'Niveles de Zoom= %d ',l);
fprintf(fid, 'Método= %s ',met);
fprintf(fid, 'Nombre de la Imagen= %s\n\n',im);
fprintf(fid,'norma1= %f\n',norma1);
fprintf(fid,'infin= %f\n',infin);
```

```
fprintf(fid,'norma2= %f\n',norma2);
fprintf(fid,'PSNR= %f\n\n',PSNR);
fclose(fid);

figure('Tag','Original','Name',[blanks(6),'ZOOM DE LA IMAGEN'],...
    'Position',[718 35 560 420])
imagesc(c,[0 255])
title('zoom de la imagen');
colormap gray
```

GUIdemo.m

```
function varargout = GUIdemo(varargin)

%function varargout = GUIdemo(varargin)
% GUIDEMO M-file for GUIdemo.fig
% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUIdemo_OpeningFcn, ...
                   'gui_OutputFcn',  @GUIdemo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

function GUIdemo_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for GUIdemo
```

```
handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% Colocar imagen de fondo

background=imread('background.jpg');
axes (handles.background);
axis off;
imshow (background);
handles.output=hObject;
guidata (hObject, handles);

function varargout = GUIdemo_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure

varargout{1} = handles.output;

function inEdit_CreateFcn(hObject, eventdata, handles)

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function inEdit_Callback(hObject, eventdata, handles)

% Hints: get(hObject,'String') returns contents of inEdit as text
% str2double(get(hObject,'String')) returns contents inEdit double

function loadPush_Callback(hObject, eventdata, handles)

image_file = get(handles.inEdit,'String');
image_file=fliplr(deblank(fliplr(deblank(image_file))));

file_dir_struct=dir('../Gui_Zoom_Exp1');
file_dir_cell=struct2cell(file_dir_struct);
```

```
file_dir_line=file_dir_cell(1,:);
coincidences=strcmp(image_file,file_dir_line);

if (nnz(coincidences)==0)
    uiwait(msgbox('Imagen no válida','Mensaje de error','error','modal'))
else
    im_original=imread(char(image_file));
    set(handles.orgIm,'HandleVisibility','ON');
    axes(handles.orgIm);
    imagesc(im_original,[0,255]); colormap gray;
    axis tight;
    grid on;
end

% ------ BOTÓN APPLY ------
% --- Executes on button press in appPush.

function appPush_Callback(hObject, eventdata, handles)}}

% hObject handle to appPush (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Nombre de la imagen a hacer zoom y rutina para corregir
% espacios y si la imagen se encuentra dentro del directorio

image_file = get(handles.inEdit,'String');
image_file=fliplr(deblank(fliplr(deblank(image_file))));

file_dir_struct=dir('../Gui_Zoom_Exp2');
file_dir_cell=struct2cell(file_dir_struct);
file_dir_line=file_dir_cell(1,:);
coincidences=strcmp(image_file,file_dir_line);

if (nnz(coincidences)==0)
    uiwait(msgbox('Imagen no válida','Mensaje de error','error','modal'))
    image_file='';
end

% Rutina para comprobar el método utilizado

if (get(handles.radiobutton1,'Value')==1)
    metodo='lin';
```

```
elseif (get(handles.radiobutton2,'Value')==1)
    metodo='enh';
elseif (get(handles.radiobutton3,'Value')==1)
    metodo='eno';
elseif (get(handles.radiobutton4,'Value')==1)
    metodo='wen';
elseif (get(handles.radiobutton5,'Value')==1)
    metodo='pph';
elseif (get(handles.radiobutton6,'Value')==1)
    metodo='esr';
else
    uiwait(msgbox('Debe seleccionar un método','Mensaje de error',...
        'error','modal'))
    metodo='Err';
end

% Rutina para comprobar el nivel de multirresolución

if (get(handles.radiobutton7,'Value')==1)
    nivel=1;
elseif (get(handles.radiobutton8,'Value')==1)
    nivel=2;
elseif (get(handles.radiobutton9,'Value')==1)
    nivel=3;
elseif (get(handles.radiobutton10,'Value')==1)
    nivel=4;
else
    uiwait(msgbox('Debe seleccionar los niveles de multirresolución ',...
        'Mensaje de error','error','modal'))
    nivel=-1;
end

% Hasta aquí tenemos almacenado
% image_file -->Nombre de la imagen a hacer zoom
% metodo -->El metodo a utilizar
% nivel -->Nivel de multirresolucion
% [a,c]=zoom_tensor_exp2(nivel,image_file,metodo); % Llamada a la
función a hacer zoom

if (nivel~= -1 & ~(strcmp(metodo,'Err')) & nnz(coincidences)~=0)
[a,c]=zoom_tensor_exp2(nivel,image_file,metodo);
end
```

```
% ------ PANEL MÉTODO A UTILIZAR:Lineal,Eno.... ------
% --- Executes on button press in radiobuttonX.
% ------ LINEAL ------

function radiobutton1_Callback(hObject, eventdata, handles)

set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ ENO JERÁRQUICO ------

function radiobutton2_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ ENO NO JERÁRQUICO ------

function radiobutton3_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ WENO ------

function radiobutton4_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton5, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ PPH ------
```

```
function radiobutton5_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton6, 'value',0);

% ------ ENO SUBCELL RESOLUTION ------

function radiobutton6_Callback(hObject, eventdata, handles)

set (handles.radiobutton1, 'value',0);
set (handles.radiobutton2, 'value',0);
set (handles.radiobutton3, 'value',0);
set (handles.radiobutton4, 'value',0);
set (handles.radiobutton5, 'value',0);

% ------ PANEL NIVEL DE MULTIRRESOLUCIÓN ------
% --- Executes on button press in radiobuttonX.
% ------ 1 NIVEL DE MULTIRRESOLUCIÓN ------

function radiobutton7_Callback(hObject, eventdata, handles)

set (handles.radiobutton8, 'value',0);
set (handles.radiobutton9, 'value',0);
set (handles.radiobutton10, 'value',0);

% ------ 2 NIVELES DE MULTIRRESOLUCIÓN ------

function radiobutton8_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
set (handles.radiobutton9, 'value',0);
set (handles.radiobutton10, 'value',0);

% ------ 3 NIVELES DE MULTIRRESOLUCIÓN ------

function radiobutton9_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
set (handles.radiobutton8, 'value',0);
set (handles.radiobutton10, 'value',0);
```

```
% ------ 4 NIVELES DE MULTIRRESOLUCIÓN ------

function radiobutton10_Callback(hObject, eventdata, handles)

set (handles.radiobutton7, 'value',0);
set (handles.radiobutton8, 'value',0);
set (handles.radiobutton9, 'value',0);

% ------ BOTÓN PARA VISUALIZAR LOS DATOS GENERADOS ------
% --- Executes on button press in datos.

function datos_Callback(hObject, eventdata, handles)

% hObject handle to datos (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

archivo = fopen ('datos generados.rtf','r');
if archivo==-1
    uiwait(msgbox('El archivo todavía no existe,...
        ¡realice una simulación!','Mensaje de error',...
        'error','modal'))
else
    str = fread(archivo,inf,'*char','native').';
    uiwait(msgbox(str,'Datos Generados.rtf','none','modal'))
    fclose(archivo);
end

% ------ BOTÓN DE SALIR DEL PROGRAMA ------

function closePush_Callback(hObject, eventdata, handles)

closePush=questdlg('¿Desea Salir del Programa?',...
    'Salida del Programa','Si','No','No');

switch closePush
    case 'Si'
        close all;
    case 'No'
        return;
end

% --------------------------------------------------------------------
```

```
function archivo_Callback(hObject, eventdata, handles)

% hObject handle to archivo (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------

function ayuda_Callback(hObject, eventdata, handles)

% hObject handle to ayuda (see GCBO) % eventdata reserved - to
be defined in a future version of MATLAB % handles structure with
handles and user data (see GUIDATA) % ----------------------------------------

function abrir_Callback(hObject, eventdata, handles)

% hObject handle to abrir (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

fileName=uigetfile( ...
{'*.pgm;*.pbm','Archivos de Imagen (*.pgm,*.pbm)';
   '*.pgm', 'Portable Graymap (*.pgm)'; ...
   '*.pbm', 'Portable Bitmap (*.pbm)'},'Pick a file');

set (handles.inEdit,'String',fileName);
image_file = get(handles.inEdit,'String');
if ~isempty(image_file)
   im_original=imread(char(image_file));
   set(handles.orgIm,'HandleVisibility','ON');
   axes(handles.orgIm);
   imagesc(im_original,[0,255]); colormap gray;
   axis tight;
   grid on;
end;

% --------------------------------------------------------------------

function salir_Callback(hObject, eventdata, handles)

% hObject handle to salir (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
salir=questdlg('¿Desea Salir del Programa?',...
    'Salida del Programa','Si','No','No');

switch salir
    case 'Si'
        close all;
    case 'No'
        return;
end

% --------------------------------------------------------------------

function acerca_Callback(hObject, eventdata, handles)

% hObject handle to acerca (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

fid = fopen('acerca de.txt');
str = fread(fid,inf,'*char').';
fclose(fid);
f = figure('menubar','none','NumberTitle','off',...
    'Name','Acerca de ZoomProgram');

h=uicontrol('style','edit','parent',f,'min',0,'max',...
    2,'enable','inactive','units','normalized','position',...
    [0 0 1 1],'string',str,'horizontalalignment','left','fontsize',10);
drawnow

% --------------------------------------------------------------------

function ayudaprog_Callback(hObject, eventdata, handles)

% hObject handle to ayudaprog (see GCBO) % eventdata reserved -
to be defined in a future version of MATLAB % handles structure
with handles and user data (see GUIDATA)

fid = fopen('ayuda.txt');
str = fread(fid,inf,'*char').';
fclose(fid);
f = figure('menubar','none','NumberTitle','off',...
    'Name','Ayuda ZoomProgram');
```

```
h=uicontrol('style','edit','parent',f,'min',0,'max',2,...
    'enable','inactive','units','normalized','position',...
    [0 0 1 1],'string',str,'horizontalalignment','left','fontsize',10);
drawnow
```

Capítulo 6

Conclusiones

En este proyecto se han desarrollado y analizado algoritmos de zoom de imágenes digitales basados en la teoría de esquemas de subdivisión. Se han comparado esquemas lineales y no lineales. Más concreto, el estudio comprende los métodos resultantes del uso de una reconstrucción de Lagrange de cuarto orden, de una reconstrucción ENO jerárquica, ENO no jerárquica, ENO subcell resolution, WENO y PPH.

Los experimentos desarrollados muestran que el método utilizado consigue calidades de imágenes aceptables y que incluso mejora otros métodos de zoom utilizados por programas comerciales.

Las diferencias entre los algoritmos de subdivisión lineales y no lineales cuando son aplicados al zoom de una imagen digital son los siguientes: con imágenes geométricas los algoritmos no lineales definen mejor los contornos evitando efectos indeseables como el fenómeno de Gibbs y/o reduciendo difusión de ejes; por otro lado cuando trabajamos con imágenes más reales las diferencias entre los métodos lineales y no lineales son mucho menos perceptibles. En este caso hay que tener cuidado de si se elige un método lineal o no lineal, entonces éste debe ser aceptable, ya que si no, se pueden producir artefactos espúreos en las imágenes. El método entre los estudiados que parece producir mejores resultados es el PPH.

El software que se a utilizado ha sido, para la implementación de los algoritmos e interfaces gráficas y su posterior aplicación a las imágenes digitales, Matlab. Para la elaboración de la memoria, el editor de texto científico Latex y en particular el paquete Beamer para la realización de la presentación.

Bibliografía

[1] S. Amat, F. Aràndiga , Cohen A. and R. Donat. Tensor product multiresolution analysis with error control for compact image representation. *Signal Processing*, **82**(4), 587-608, 2002.

[2] S. Amat, F. Aràndiga, A. Cohen, R. Donat, G. García and M. Von Oehsen. Data compression with ENO schemes: A case study. *Applied and Computational Harmonic Analysis*, **11**, 273-288, 2001.

[3] S. Amat, S. Busquier and J.C. Trillo. Stable Interpolatory Multiresolution in 3D. *Applied Numerical Analysis and Computational Mathematics*, **2**(2), 177-188, 2005.

[4] S. Amat, S. Busquier and J.C. Trillo. Non-linear Hartens's Multiresolution on the Quincunx Pyramid. *Journal of Computational and Applied Mathematics*, **189**, 555-567, 2006.

[5] S. Amat, R. Donat, J.Liandrat and J.C.Trillo. Proving convexity preserving properties of interpolatory subdivision schemes through reconstruction operators. In preparation.

[6] S. Amat, R. Donat, J. Liandrat and J.C. Trillo. Analysis of a new nonlinear subdivision scheme. Applications in image processing. *Foundations of Computational Mathematics*, **6**(2), 193-226, 2006.

[7] S. Amat and J. Liandrat. On the stability of the PPH nonlinear multiresolution. *Applied Computational Harmonic Analysis*, **18**(2), 198-206, 2005.

[8] F. Aràndiga and R. Donat. Nonlinear Multi-scale Decomposition: The Approach of A.Harten. *Numerical Algorithms*, **23**, 175-216, 2000.

[9] F. Aràndiga, R. Donat and A. Harten. Multiresolution Based on Weighted Averages of the Hat Function I: Linear Reconstruction Operators, *SIAM Journal on Numerical Analysis*, **36**, 160-203, 1999.

[10] F. Aràndiga, R. Donat and A. Harten. Multiresolution Based on Weighted Averages of the Hat Function II: Nonlinear Reconstruction Operators, *SIAM Journal on Scientific Computing*, **20**(3), 1053-1099, 1999.

[11] F. Aràndiga and J.C. Trillo. Resultados de Interpolación No Lineal. *Revista Matemàtica de la Universitat de València*, **1**(1).

[12] S. Bachelli and S. Papi. Filtered wavelet thresholding mehthods. *Journal of Computational and Applied Mathematics*, **164-165**, 39-52, 2004.

[13] A.S. Cavaretta, W. Dahmen and C.A Micchelli. Stationary subdivision. *Memoirs of the American Mathematical Society*, **93**(453):186, 1991.

[14] F. Chen. Estimating subdivision depths for rational curves and surfaces. *ACM Transactions Graphics*, **11**(2), 140-151, 1992.

[15] A. Cohen, I. Daubechies and J.C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications in Pure and Applied Mathematics*, **45**, 485-560, 1992.

[16] A. Cohen, N. Dyn and B. Matei. Quasilinear subdivision schemes with applications to ENO interpolation. *Applied Computational Harmonic Analysis*, **15**, 89-116, 2003.

[17] G. Delauries and S. Dubuc. Symmetric Iterative Interpolation Scheme. *Constructive Approximations*, **5**, 49-68, 1989.

[18] D. Donoho. Interpolating wavelet transforms. *Prepint Standford University*, 1994.

[19] D. Donoho. Denoising by soft thresholding, *IEEE Transactions on Informations Theory*, **41**(3), 613-627, 1995.

[20] N. Dyn. Subdivision schemes in computer aided geometric design. *Advances in Numerical Analysis II., Subdivision algorithms and radial functions, W.A. Light (ed.), Oxford University Press*, 36-104. Prentice-Hall, 1992.

[21] N. Dyn, A. Gregory and D. Levin. A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, **4**, 257-268, 1987.

[22] N. Dyn, J.A. Gregory and D. Levin. Analysis of linear binary subdivision schemes for curve design. *Constructive Approximations*, **7**, 127-147, 1991.

[23] N. Dyn, F. Kuijt, D. Levin and R. Van Damme. Convexity preservation of the four-point interpolatory subdivision scheme. *Computer Aided Geometric Design*, **16**(8), 789-792, 1999.

[24] E. Fatemi, J. Jerome and S. Osher. Solution of the hydrodynamic device model using high order non-oscillatory shock capturing algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **10**, 232-244, 1991.

[25] M.S. Floater and C.A. Micchelli. Nonlinear stationary subdivision. *Approximation theory: in memory of A.K. Varna, edt: Govil N.K, Mohapatra N., Nashed Z., Sharma A., Szabados J.*, 209-224, 1998.

[26] J. Gomes and L. Velho. Computação Gráfica: Imagem, *Série Computação e Matemática*, 2002.

[27] A. Harten. ENO Schemes with subcell resolution, *Journal of Computational Physics*, **83**, 148-184, 1989.

[28] A. Harten. Discrete multiresolution analysis and generalized wavelets, *Journal Applied Numerical Mathematics*, **12**, 153-192, 1993.

[29] A. Harten. Multiresolution representation of data II, *SIAM Journal on Numerical Analysis*, **33**(3), 1205-1256, 1996.

[30] A. Harten, B. Engquist, S.J. Osher and S.R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes III, *Journal of Computational Physics*, **71**, 231-303, 1987.

[31] A. Harten, S.J. Osher, B. Engquist and S.R. Chakravarthy. Some results on uniformly high-order accurate essentially non-oscillatory schemes, *Applied Numerical Mathematics*, **2**, 347-377, 1987.

[32] G.-S. Jiang and C.-W. Shu, Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, **126**, 202-228, 1996.

[33] F. Kuijt and R. van Damme. Convexity preserving interpolatory subdivision schemes. *Constructive Approximations*, **14**, 609-630, 1998.

[34] A. Marquina. Local piecewise hyperbolic reconstruction of numerical fluxes for nonlinear scalar conservation laws. *SIAM Journal on Scientific Computing*, **15**(4), 892-915, 1994.

[35] C.A. Micchelli. Interpolatory subdivision schemes and wavelets. *Journal of Approximation Theory*, **86**, 41-71, 1996.

[36] G. Mustafa, F. Chen and J. Deng. Estimating error bounds for binary subdivision curves/surfaces. *Journal of Computational and Applied Mathematics*. To appear.

[37] X.-D. Liu, S. Osher and T. Chan, Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, **115**, 200-212, 1994.

[38] M. Rabbani and P.W. Jones. Digital Image Compression Techniques. Tutorial Text, *Society of Photo-Optical Instrumentation Engineers (SPIE)*, TT07, 1991.

[39] C.-W. Shu. Numerical experiments on the accuracy of ENO and modified ENO schemes. *Journal of Scientific Computing*, **5**, 127-149, 1990.