DISEÑO DE SISTEMAS DIFUSOS PARA PROCESADO DE IMÁGENES CON XFUZZY 3

I. BATURONE, P. BROX Y R. ARJONA
Departamento de Electrónica y Electromagnetismo. Universidad de Sevilla- Instituto de Microelectrónica de Sevilla, IMSE-CNM-CSIC. España.
{lumi, brox, arjona}@imse-cnm.csic.es

La presente comunicación describe la utilización de un software de libre distribución, Xfuzzy 3, para ilustrar la aplicación de sistemas difusos al procesamiento de imágenes, en concreto, al problema del aumento de resolución. El proceso de diseño de sistemas difusos quedará cubierto por el uso de las herramientas CAD de descripción, verificación, identificación, aprendizaje y simplificación del entorno XFuzzy en su versión 3.3, que facilitan al alumno la comprensión de todos los pasos del proceso.

Palabras clave: Sistemas de lógica difusa, sistemas de procesado de imágenes, software educativo, aprendizaje basado en prácticas.

1. Introducción

Desde sus orígenes con la teoría de conjuntos difusos de Zadeh [1], la lógica difusa se concibió como una teoría para manejar los conceptos ambiguos e imprecisos del mundo real tal y como lo haría un humano a través del lenguaje natural, evitando el tratamiento más rígido de la teoría de conjuntos tradicional. El campo de aplicación por excelencia de la lógica difusa ha sido el de control. Por ello, existen muchas referencias al uso de entornos de desarrollo de sistemas difusos para el diseño de controladores. Por el contrario, son pocas las referencias al diseño automatizado de sistemas difusos en aplicaciones de procesamiento de señal. Específicamente para el procesado de imágenes tan sólo encontramos en la literatura la herramienta propuesta por Russo, que se aplica a un problema de extracción de bordes [2].

En esta comunicación ilustraremos cómo el entorno Xfuzzy 3 (tradicionalmente empleado para el diseño de controladores) también puede emplearse para el diseño de procesadores de imágenes. En concreto, se ilustra una práctica en la que los alumnos diseñan sistemas difusos para el aumento de la resolución de imágenes empleando las herramientas de descripción, verificación, identificación, aprendizaje y simplificación. En primer lugar, se diseña un sistema difuso que aplica cierto conocimiento heurístico basado en el funcionamiento de un algoritmo ampliamente conocido en el procesado de imágenes. La segunda parte de la práctica consiste en extraer directamente el sistema de procesado difuso a partir de datos numéricos obtenidos de distintas imágenes y sus correspondientes imágenes agrandadas.

Esta práctica se realiza en la asignatura “Sistemas neuro-mórficos y difusos: aplicaciones y casos prácticos”, que es optativa dentro del “Máster en Microelectrónica: diseño y aplicaciones de sistemas micro-nanométricos” (un máster que se imparte de forma no presencial en la Universidad de Sevilla). También parte de ella se realiza en la asignatura “Computadores neuronales”, optativa de 5º curso de Ingeniería Informática de la Universidad de Sevilla.

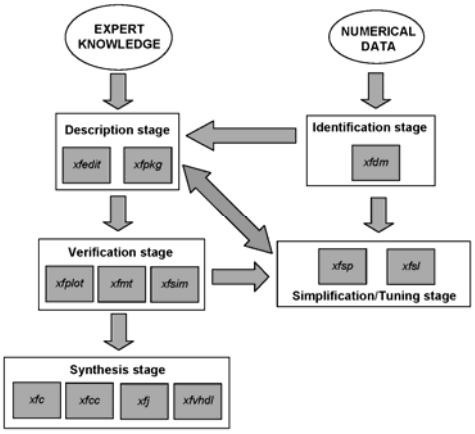


Figura 1. Diagrama de flujo de diseño de sistemas difusos con XFuzzy 3

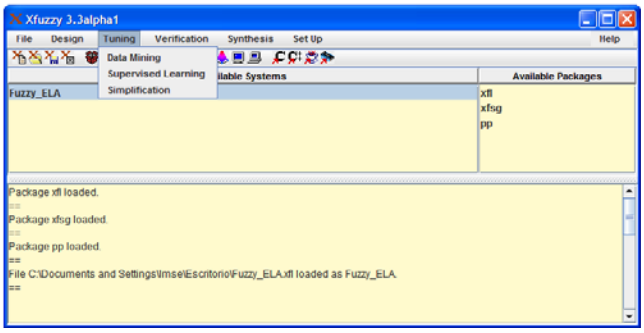


Figura 2. Ventana principal de XFuzzy 3

2. Descripción del entorno XFuzzy 3

XFuzzy 3 es un entorno de desarrollo de sistemas difusos complejos realizado en el Instituto de Microelectrónica de Sevilla [3]. La versión 3.3 contiene herramientas que cubren todo el proceso de diseño de sistemas difusos (descripción, verificación, identificación, aprendizaje, simplificación y síntesis) teniendo como base común el lenguaje de especificación formal XFL3 (Fig. 1) [4]. De ellas, se utilizarán todas en esta práctica, salvo las de síntesis.

Xfuzzy es un entorno en modo gráfico que facilita el tratamiento de sistemas difusos cumpliendo requisitos como portabilidad (está desarrollado en Java y, por tanto, puede ser ejecutado en cualquier plataforma) y usabilidad (a través de la versatilidad que ofrece su interfaz de usuario (Fig. 2)). Estos rasgos lo hacen propios de un entorno didáctico para la enseñanza de sistemas difusos. Cada una de las etapas del diseño queda cubierta por el acceso a una funcionalidad en la ventana principal del sistema o su ejecución como herramienta independiente. La descripción de sistemas difusos con la herramienta *xfedit* permite al alumno editar un sistema difuso sin necesidad de conocer los detalles del lenguaje de especificación *XFL3*. La edición de los paquetes que contienen las funciones que se utilizarán para las definiciones anteriores se puede realizar cómodamente a través de la herramienta *xfpkg*.

La verificación permite el análisis del sistema difuso descrito en su contexto de aplicación. En XFuzzy 3 esta etapa se realiza a través de tres herramientas de diferente naturaleza: *xfplot*, que permite representar gráficamente en dos o tres dimensiones la salida del sistema frente a una o dos de sus entradas; *xfmt*, para monitorizar los grados de pertenencia de las variables involucradas en las reglas a los conjuntos difusos así como los grados de activación y las conclusiones de las reglas para unos valores determinados de entrada, y *xfsim* como herramienta de simulación, que utiliza un modelo del entorno sobre el que se aplica el sistema difuso (tradicionalmente una planta a controlar en lazo cerrado), para evaluar su comportamiento.

El ajuste de un sistema difuso es mejor realizarlo mediante un aprendizaje automático, debido a la gran cantidad de parámetros que se manejan. XFuzzy 3 incluye la posibilidad de configurar y ejecutar el aprendizaje supervisado con la herramienta *xfsl* teniendo en cuenta unos patrones que determinan el comportamiento de entrada/salida deseado. También permite la posibilidad de generar un sistema a partir de unos datos de entrada (*Data Mining*) sin tener previamente que describir la estructura, es decir, se

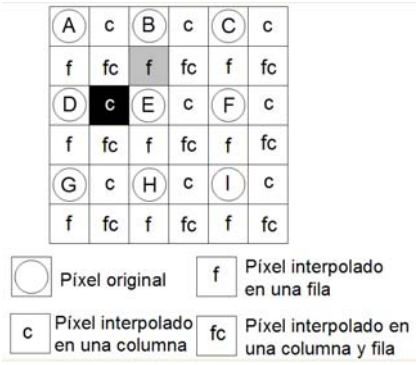


Figura 3. Píxeles que participan en el proceso de aumento de resolución de una imagen para un factor de amplificación igual a 2.

extrae una base de reglas a partir del conjunto de patrones, utilizando la herramienta *xfdm*. Para eliminar el conocimiento redundante que puede ser adquirido en todo proceso de aprendizaje, Xfuzzy posee la herramienta de simplificación *xfsp*.

3. Diseño de un sistema difuso de procesado a partir de heurística

Para agrandar o aumentar la resolución de una imagen hay que introducir nuevos píxeles en la imagen que constituyan nuevas líneas y columnas. Esto se ilustra en la Fig. 3 para un factor de amplificación igual a 2. El aumento de resolución de imágenes, por tanto, puede verse como un problema de interpolación para el cual se han empleado diversas técnicas de interpolación, como las basadas en polinomios de distinto grado o las basadas en redes neuronales. En esta práctica se ilustra a los alumnos el uso de técnicas de interpolación difusa.

Como primer paso en nuestro proceso de diseño vamos a proponer un sistema difuso que se base en conocimiento heurístico expresado lingüísticamente mediante reglas del tipo 'si-entonces'. Para ello, encontramos en la literatura distintos algoritmos que nos sirven de inspiración. En particular, fijaremos nuestra atención en un algoritmo ampliamente conocido como es el ELA (*Edge Line Average*), que realiza una interpolación que se adapta a los bordes de la imagen, utilizando las diferencias entre las luminancias de los píxeles de las líneas superiores e inferiores (Fig. 4a). El pseudocódigo consta de 3 reglas que tienen en cuenta el mínimo de estas diferencias, en la parte antecedente, y realiza un promediado de los valores de los píxeles en la dirección más probable de borde, en la parte consecuente (Fig. 4b) [5].

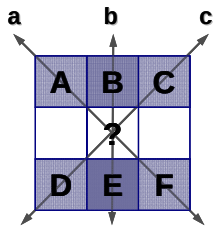


Figura 4a. Píxeles considerados en el ELA

$$a = |A - F| \quad b = |B - E| \quad c = |C - D|$$
$$\text{if} \quad \min\{a,b,c\} = a \Rightarrow \quad X = (A + F)/2$$
$$\text{if} \quad \min\{a,b,c\} = c \Rightarrow \quad X = (C + D)/2$$
$$\text{else} \quad \Rightarrow \quad X = (B + E)/2$$

Figura 4b. Pseudocódigo del ELA

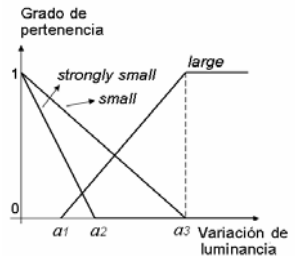


Figura 5a. Definición de conceptos difusos

if	*antecedents*	*consequent*
1)	(a is SMALL) and (b is LARGE) and (c is LARGE)	(A+F)/2
2)	(a is LARGE) and (b is LARGE) and (c is SMALL)	(C+D)/2
3)	(a is strongly SMALL) and (b is LARGE) and (c is strongly SMALL)	(A+F+C+D)/4
4)	otherwise	(B+E)/2

Figura 5b. Pseudocódigo del algoritmo difuso

Para construir un sistema homólogo al ELA pero difuso se tendrá en cuenta no sólo que las diferencias en una dirección sean pequeñas sino que en las otras direcciones deben ser grandes. Los conceptos de "pequeño" y "grande" se entenderán como difusos y se representarán mediante funciones de pertenencia (Fig. 5a). Además, incorporaremos una nueva regla que acote un caso más de los considerados por la rama *else*. El pseudocódigo de este algoritmo difuso de procesado se muestra en la Fig. 5b [6].

3.1. Etapa de descripción

La traducción a XFuzzy de este algoritmo difuso requiere, en primer lugar, la definición de los operadores que vayan a ser utilizados. Para este caso, utilizaremos funciones de pertenencia trapezoidales para realizar la operación de *fuzzification*, el operador mínimo para resolver las conectivas *and* de las reglas y la media ponderada (habitual como operador de *defuzzification* en los sistemas difusos) para calcular la salida global del sistema (Ec. 1):

$$X = \beta_1(\frac{A+F}{2}) + \beta_2(\frac{C+D}{2}) + \beta_3(\frac{A+F+C+D}{4}) + \beta_4(\frac{B+E}{2}) \quad (1)$$

Donde β_1, β_2, β_3 y β_4 son los grados de activación de las reglas correspondientes. En la ecuación anterior no aparece un denominador con la suma de los grados de activación porque esta suma es la unidad. Esto sucede al establecer β_4 como el complemento de los valores del resto de grados de activación (es la manera de definir el operador matemático para el concepto lingüístico *otherwise*).

Las funciones de pertenencia trapezoidales y el operador mínimo pueden ser empleados directamente por el estudiante porque ya están incluidos en el paquete estándar que trae Xfuzzy. Pero el operador de suma ponderada (sin división) no está incluido, por lo que el estudiante debe definir un nuevo paquete que incluya esta funcionalidad. La definición de un operador nuevo se hace seleccionando su nombre (y alias posibles), los parámetros que especifican su comportamiento (así como posibles restricciones sobre estos parámetros), la descripción de su comportamiento en Java (y en C y C++ si se quiere generar síntesis a esos lenguajes) e incluso la descripción de sus derivadas (si se quiere emplear ese operador en métodos de aprendizaje guiados por gradiente). El estudiante puede introducir toda esta información de forma gráfica mediante la herramienta *xfpkg* (Fig. 6). Con esta herramienta se genera automáticamente una clase Java que puede emplearse en los sistemas difusos que se describan.

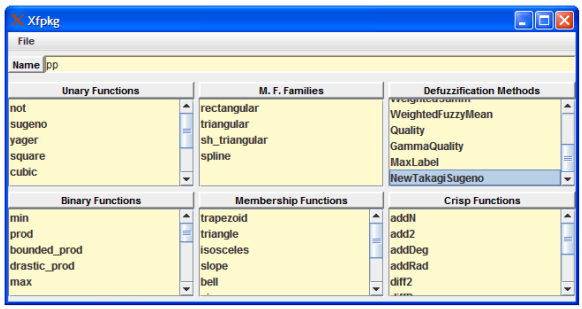
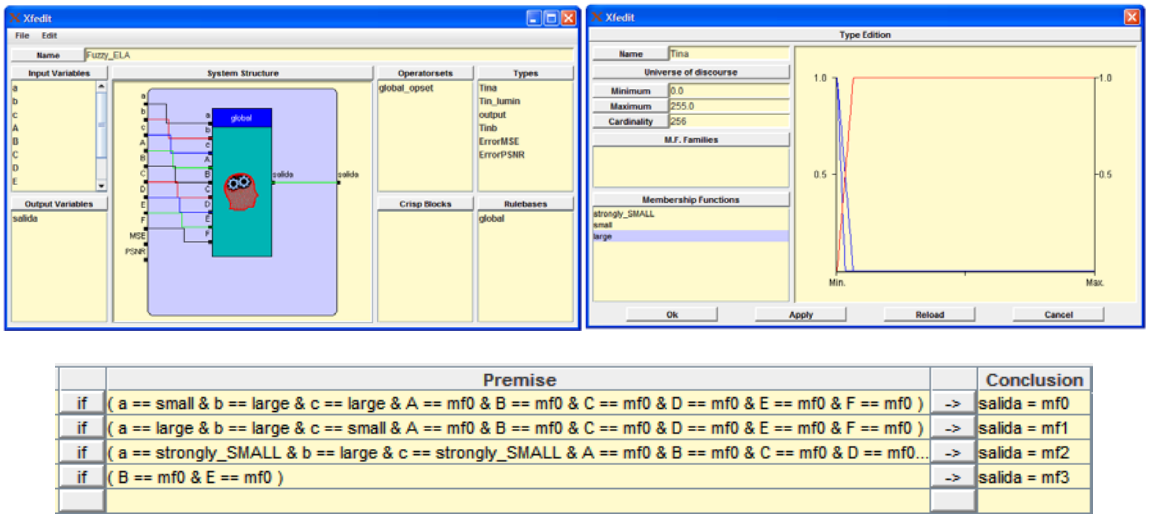


Figura 6. Edición de paquetes de funciones con *xpkg*

Tras esto, el siguiente paso es crear la estructura del sistema utilizando la herramienta *xfedit* (Fig. 7). Esto significa definir las variables de entrada y de salida, las funciones de pertenencia y la base de reglas del sistema global. Los alumnos definirán libremente los parámetros de las funciones de pertenencia en base a su idea de *large*, *small* y *strongly small*.

3.2. Etapa de verificación

La verificación de un sistema difuso tiene como objetivo comprobar su comportamiento. En este caso, se empleará la herramienta *xfsim* que nos permite aplicar una simulación utilizando un modelo del contexto de la aplicación (Fig. 8). Este modelo no es más que una clase Java que se les proporciona a los alumnos y que, en este caso, extrae de la imagen a agrandar los 6 píxeles que se emplean como vecinos para interpolar el nuevo píxel a introducir. Con la base de reglas definida en el sistema se interpolan primero los píxeles tipo ‘f’ (de la Fig. 3) y, de forma análoga, los píxeles tipo ‘c’ (de la Fig. 3). Los píxeles mostrados con el símbolo ‘fc’ en la Fig. 3 se interpolan a continuación utilizando como entrada los cuatro píxeles originales más cercanos y los cuatro píxeles ya interpolados también más cercanos. El resultado final se obtiene como el valor medio de los resultados obtenidos al aplicar la base de reglas del sistema en los 3+3 píxeles de las líneas superior e inferior, y los 3+3 píxeles de las columnas izquierda y derecha. Por otro lado, esta misma clase toma el píxel nuevo proporcionado por el sistema difuso y lo va almacenando para construir la nueva imagen de mayor resolución.



	Premise		Conclusion
if	(a == small & b == large & c == large & A == mf0 & B == mf0 & C == mf0 & D == mf0 & E == mf0 & F == mf0)	->	salida = mf0
if	(a == large & b == large & c == small & A == mf0 & B == mf0 & C == mf0 & D == mf0 & E == mf0 & F == mf0)	->	salida = mf1
if	(a == strongly_SMALL & b == large & c == strongly_SMALL & A == mf0 & B == mf0 & C == mf0 & D == mf0...	->	salida = mf2
if	(B == mf0 & E == mf0)	->	salida = mf3

Figura 7. Descripción global del sistema, de las funciones de pertenencia y de la base de reglas con *xfedit*

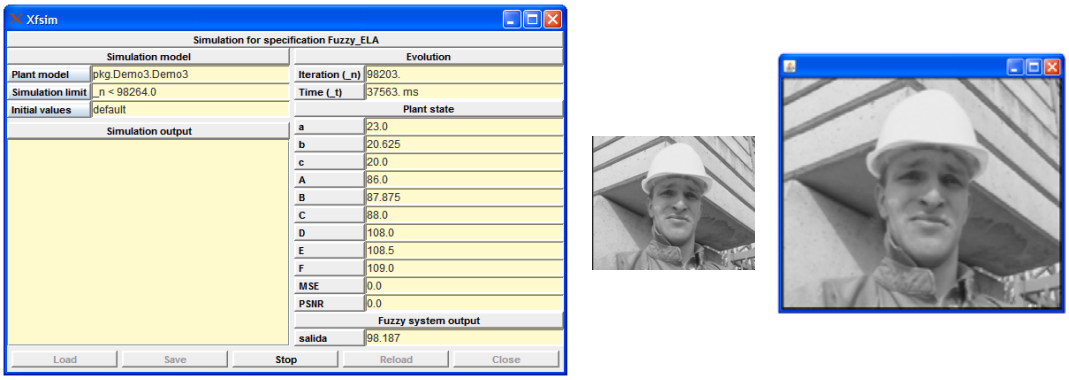


Figura 8. Ventana de ejecución de *xfsim*, imagen original y resultado de la simulación

Se invita a los alumnos a que modifiquen los parámetros que definen las funciones de pertenencia de los conjuntos difusos para que aprecien su influencia en la imagen agrandada. En particular, se les sugiere que definan la etiqueta *small* de la variable *b* ocupando prácticamente todo el universo de discurso para que predomine la regla *otherwise*. En tal caso, la interpolación apenas se adapta a los bordes y se puede apreciar un efecto de escalera en el agrandado de las zonas con bordes.

4. Diseño de un sistema difuso de procesado a partir de datos numéricos

Se supone ahora que desconocemos cualquier tipo de heurística para aplicar en este problema. La otra estrategia que se propone entonces a los alumnos es que definan el sistema de procesado a partir de datos numéricos correspondientes a imágenes con mayor y menor resolución. Se trata de utilizar un conjunto de muestras de diferentes imágenes que tienen como entradas las diferencias de luminancias y las propias luminancias y como salidas las luminancias de los píxeles que se desean interpolar. La herramienta *xfdm* nos permite generar un sistema a partir de datos numéricos utilizando algoritmos orientados a estructura (*Grid*) o bien basados en el agrupamiento de los datos (*Clustering*). La interfaz gráfica de esta herramienta permite además seleccionar el fichero que contiene los datos que se considerarán en el proceso y distintas opciones para las variables de entrada y para el sistema en general. Por comodidad para la realización de la práctica, se les proporciona a los alumnos un fichero de patrones creados a partir de imágenes de distinta naturaleza (con gran contraste entre blanco y negro, con suavidad en los valores de los píxeles, con muchos bordes, etc.).

Tras la identificación, se hace necesario un proceso de simplificación (que se realizará utilizando la herramienta de simplificación *xfsp*) para eliminar la adquisición de conocimiento redundante (funciones de pertenencia similares, variables lingüísticas innecesarias o la creación de un número elevado de reglas) que dificulta la comprensión del sistema. En cualquier caso, se hace necesaria la aplicación de un aprendizaje supervisado que complemente el resultado del proceso de extracción de reglas. En XFuzzy 3 esto es posible mediante la funcionalidad *xfsl*, que se empleará para ajustar los parámetros del sistema a partir del mismo fichero de patrones.

4.1. Extracción de reglas mediante algoritmos de tipo “Grid”

Las técnicas de identificación basadas en algoritmos de tipo “Grid” realizan una partición de tipo matricial o rejilla de los datos de entrada para estructurar el espacio y obtener la base de reglas que soporte el sistema difuso. El inconveniente de estos sistemas es que, aplicado a muchas entradas, genera un sistema muy complejo puesto que el número de reglas aumenta exponencialmente con el número de

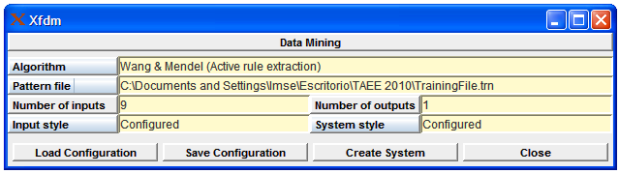


Figura 9a. Ventana de identificación *xfdm*

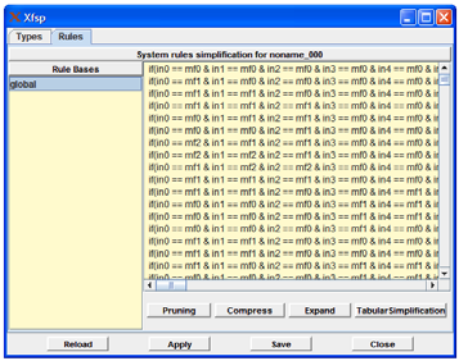


Figura 9b. Simplificación de reglas con *xfsp*

entradas. Por lo tanto, es necesario simplificar posteriormente para encontrar un sentido lingüístico en el resultado.

Con la herramienta *xfdm* (Fig. 9a) realizamos una extracción de reglas aplicando el algoritmo *Wang y Mendel* como algoritmo orientado a estructura [7], con tres funciones de pertenencias de tipo gaussianas como variables de entrada, el producto como operador de conjunción y *Takagi-Sugeno* como método de inferencia.

Como el sistema generado posee información redundante en una base con un número tan elevado de reglas y de funciones de pertenencias en los consecuentes, es necesario simplificar antes de realizar el aprendizaje. La herramienta *xfsp* (Fig. 9b) nos permite realizar dicha tarea hasta obtener una base de reglas con las tres mejores reglas acordes al conjunto de datos. Entonces, podemos aplicar un proceso de aprendizaje a los consecuentes de las reglas (aplicando, por ejemplo, el algoritmo de aprendizaje de *Marquardt-Levenberg*) haciendo uso de la herramienta *xfsl* (Fig. 10a). Verificaremos el comportamiento del sistema obtenido aplicando una simulación en *xfsim* (Fig. 10b) seleccionando una imagen para agrandar.

4.2. Extracción de reglas mediante algoritmos de tipo “Clustering”

En este caso, la extracción de información se realiza por partición y agrupamiento de los datos, identificando conjuntos de datos similares o cercanos entre los datos suministrados. Los algoritmos de tipo clustering, a diferencia de los de tipo rejilla, son buenos para trabajar con muchas entradas porque no

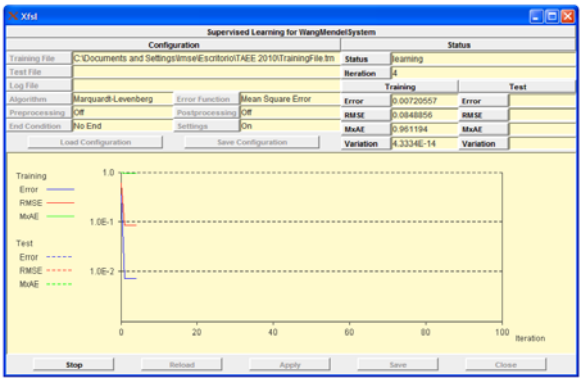


Figura 10a. Herramienta de aprendizaje *xfsl*

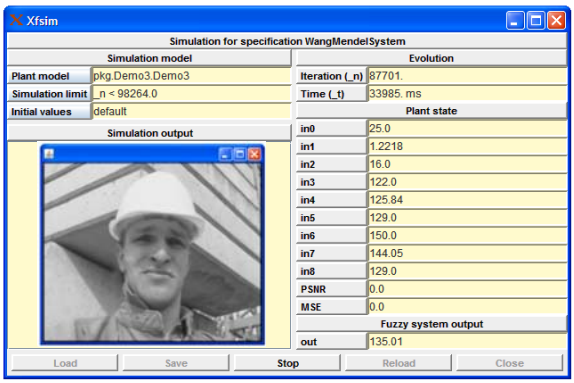


Figura 10b. Simulación del sistema con *xfsim*

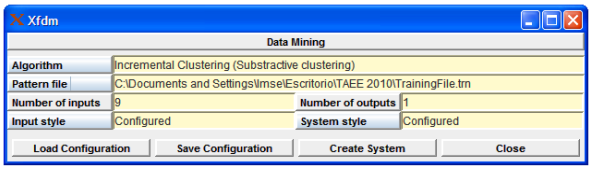


Figura 11a. Ventana de identificación *xfdm*

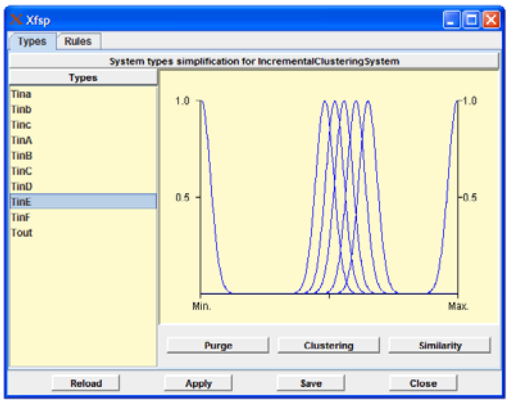


Figura 11b. Simplificación de funciones de pertenencia con *xfsp*

consideran combinaciones de las entradas sino conjuntos de datos, reduciendo el espacio de información. Su limitación surge en la interpretabilidad de las reglas obtenidas, dada la proyección que se hace de los grupos de datos a conjuntos difusos. Según el tipo de algoritmo que se seleccione con la herramienta *xfdm*, será necesario indicar el número de grupos (*clusters*) en los que se desea particionar el espacio de los datos, o el número máximo de ellos y su radio de vecindad.

En esta práctica se propone una extracción de reglas utilizando como algoritmo el *Incremental Clustering*, que va creando grupos de forma sucesiva considerando centros que posean el mayor número de datos cercanos [8]. En la interfaz de la herramienta *xfdm* (Fig. 11a) configuramos una identificación con siete grupos como máximo, seleccionando el mismo fichero de datos con nueve entradas y una salida usado en el apartado anterior, el producto como operador de conjunción y el método *Takagi-Sugeno* como método de inferencia.

La herramienta *xfsp* (Fig. 11b) permite simplificar las funciones de pertenencia de las variables del sistema (por ejemplo, hasta una función de pertenencia para las entradas correspondientes a las diferencias de luminancias, y tres funciones de pertenencia para los de las propias luminancias). A continuación, aplicaremos aprendizaje a los consecuentes del sistema construido utilizando el algoritmo *Marquardt-Levenberg* (Fig. 12a). Se repetirá el proceso de simplificación y posterior entrenamiento de los consecuentes combinando las herramientas *xfsp* y *xfsl*. El objetivo es obtener una base con sólo tres reglas que admita interpretación lingüística a la vez que se pueda comparar con la base de reglas de los apartados anteriores (tanto la de tipo rejilla como la heurística). La simulación con *xfsim* del sistema extraído permitirá evaluar su eficiencia a la hora de agrandar imágenes (Fig. 12b).

5. Comparación de resultados

La última parte de la práctica se centra en comparar los 3 sistemas diseñados. Para ello, lo primero es entender las reglas que se han extraído en los 2 sistemas diseñados a partir de datos numéricos. Aparte de usar la herramienta *xfedit* para visualizar tanto las funciones de pertenencia de antecedentes como los parámetros de los consecuentes y las reglas, se usará la herramienta de monitorización *xfmt* para visualizar cómo se activan las distintas reglas a la hora de proporcionar una determinada salida para unos valores concretos de las entradas.

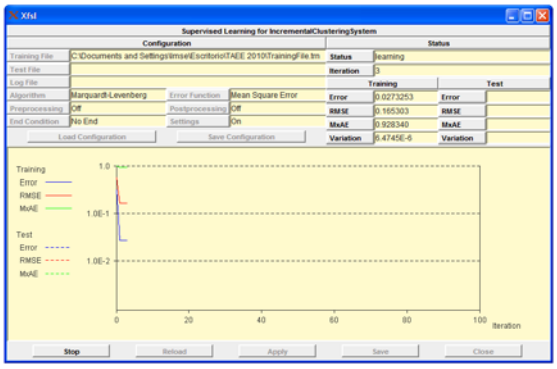


Figura 12a. Herramienta de aprendizaje *xfsl*

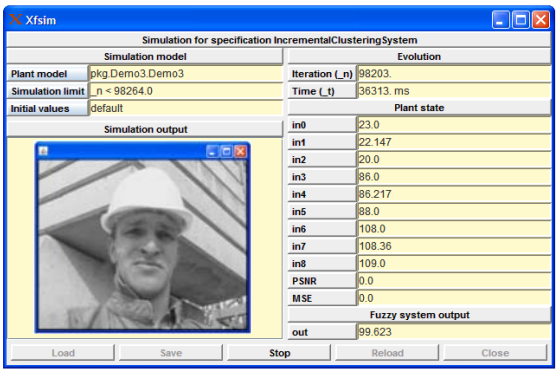


Figura 12b. Simulación del sistema con *xfsim*

Si se monitoriza el sistema creado con el algoritmo de *Incremental Clustering* (Fig. 13) los alumnos observarán que las entradas que más influyen en el valor de la salida, *out*, son las luminancias B y E, es decir, los píxeles vecinos más cercanos al píxel que queremos interpolar (Fig. 3). De hecho, observando la base de reglas y los valores asociados para los consecuentes (mediante *xfedit*) se puede resumir el sistema extraído de una manera similar a la siguiente (donde los conceptos *blanco*, *gris* y *negro* son difusos):

Si los píxeles vecinos son *blancos* entonces out = 0.45|B – E| + 0.46B + 0.52E
Si los píxeles vecinos son *grises* entonces out = 0.5B + 0.5E
Si los píxeles vecinos son *negros* entonces out = -0.25|B – E| + 0.5B + 0.56E

El sistema adquirido mediante el algoritmo de tipo *Grid* tiene un comportamiento similar, de forma que la interpolación prácticamente se realiza siempre como la media de los dos píxeles más cercanos (B y E). Este resultado se corresponde con el caso *otherwise* de la base de reglas propuesta por heurística. Se deduce, por tanto, que los algoritmos de extracción de reglas combinados con los de aprendizaje y simplificación obtienen aproximadamente como regla más representativa la regla por defecto del sistema heurístico. Como resultado, el agrandado de imágenes que proporcionan se ve más afectado por el efecto escalera (Fig. 14a) que el agrandado realizado por el sistema heurístico (Fig. 14b) porque no se tienen en

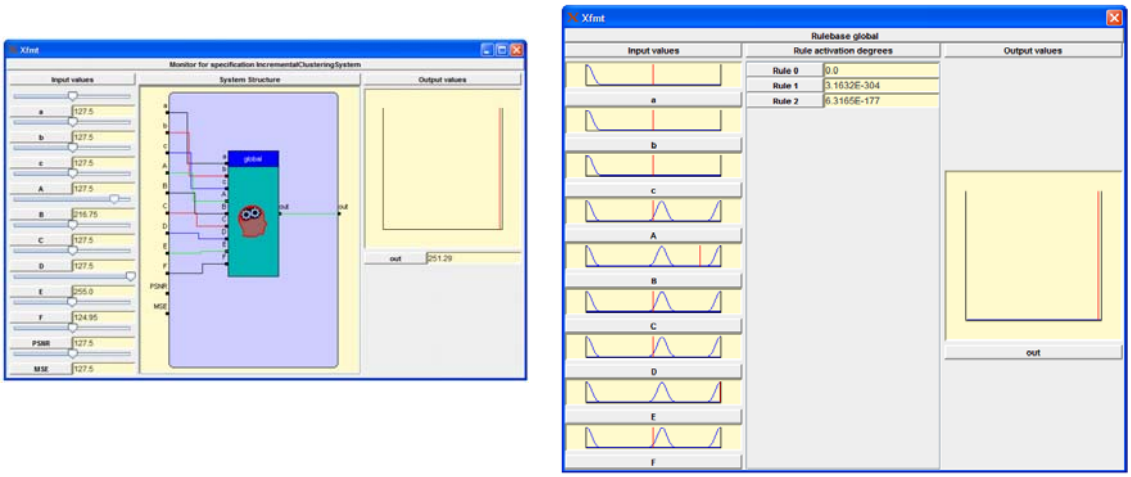


Figura 13. Monitorización del sistema con *xfmt*

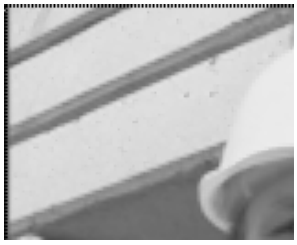


Figura 14a. Detalle de la imagen agrandada por el sistema obtenido con *Incremental Clustering*

Figura 14b. Detalle de la imagen agrandada por el sistema heurístico

cuenta los bordes. Los sistemas difusos obtenidos a partir de datos numéricos ponen de manifiesto que, para minimizar el error cuadrático medio correspondiente al fichero de datos proporcionado, lo mejor es hacer una media de los vecinos más cercanos. Obtienen, por tanto, un algoritmo de agrandado sencillo que, por otra parte, es un algoritmo usado habitualmente por su simplicidad.

6. Conclusiones

Esta práctica ilustra al alumnado el diseño y aplicación de sistemas difusos en el campo del procesado de imágenes, en concreto en una aplicación de aumento de resolución. Las herramientas de Xfuzzy (tradicionalmente empleadas para el diseño de sistemas de control) también permiten cubrir todas las fases de diseño de sistemas de procesado. Por un lado, el alumno aprende cómo algoritmos ampliamente establecidos para procesado de imágenes pueden admitir una versión difusa, pudiendo evaluar sus prestaciones. Por otro lado, aprende estrategias de extracción de conocimiento, aprendizaje y simplificación para obtener algoritmos de procesado a partir de datos numéricos. Estos objetivos se consiguen de forma cómoda e intuitiva gracias a las numerosas interfaces gráficas de usuario. La disponibilidad de Xfuzzy como entorno de libre distribución permite, además, la realización de esta práctica en asignaturas impartidas de forma no presencial.

Referencias

[1] L. A. Zadeh. *Fuzzy sets*. Inf. Contr. (1965).

[2] F. Russo. “A user-friendly research tool for image processing with fuzzy rules”. Proc. *First IEEE Int. Conf. Fuzzy System.* pp. 561-568 (1992).

[3] Xfuzzy home page: http://www.imse-cnm.csic.es/Xfuzzy

[4] I. Baturone et al. “Using Xfuzzy environment for the whole design of fuzzy systems”. Proc. *IEEE Int. Conf. Fuzzy System,* London, July 23-26 (2007).

[5] T. Doyle and M. Looymans. “Progressive scan conversion using edge information”. Signal Processing of HDTV, II. L. Chiariglione, ED., Elsevier Science Publishers, pp.711-721 (1990).

[6] P. Brox, I. Baturone, S. Sánchez Solano and A. Barriga. “Image enlargement using the Fuzzy-ELA algorithm”. Proc. IPMU 2006, Julio (2006).

[7] L-X. Wang. “A course in fuzzy systems and control”. Prentice Hall, Englewood Cliffs, New Jersey, (1996).

[8] S. L. Chiu. “A cluster estimation method with extension to fuzzy model identification”. Proc. *IEEE Conf. of Decision.* pp. 1240-1245 (1994).