E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Segmentación de imágenes a través de lógica difusa y funciones REF, de Dombi y penalti

Grado en Ingeniería Informática

Trabajo Fin de Grado

Iñigo Aguas Ardaiz

Humberto Bustince Sola
Fco. Javier Fernández Fernández

Pamplona, 25 de junio de 2015

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

"Las máquinas me sorprenden
con mucha frecuencia."

Alan M. Turing

Gracias a Javier, a Humberto y a todo el Grupo de Inteligencia Artificial y Razonamiento Aproximado (GIARA). Todos han conseguido que este proyecto avanzase y que, aun con situaciones de desánimo total, siguiera ahí, dándolo todo, para llegar al final. Sin duda alguna, si algo me han demostrado es que la investigación no es nada fácil, y que las alegrías se sirven en frascos pequeños en este mundo.

Gracias a todos mis compañeros, a Luis, a Álvaro, a Kevin, a Iñaki, a Paul, al otro Álvaro, etc. Soy una persona que puedo ser un poco singular en mi forma de trabajar, pero tended por seguro que sea cual haya sido mi acción nunca habrá sido con maldad. Y sí, lo sé, quizá me debería centrar más en lo importante.

Gracias al grupo que formamos Edu, Alfon, Marcos, Pedro y alguno más. Fueron muchas horas juntos y muchas idas de olla. Muchos momentos que recordar y mucha #EuforiaColectiva que no olvidar.

Gracias a todos mis compañeros de la Comunidad Arrupe y Universitaria del Colegio San Ignacio - Jesuitas de Pamplona. El poder estar con los chavales es un lujo, pero el ver cómo avanza algo que casi vi nacer (por lo menos como es hoy en día) y ver que es imparable, que se sigue poniendo todo el esfuerzo y todas las ganas es maravilloso. Y poder apoyarse en personas como Nacho, Nuño o Maite es estupendo.

Gracias a todos los amigos que he hecho durante estos años. Alex, Uxue, Dani, Jorge y todos los demás, porque la lista es muy larga. Es un placer poder decir que participo en RITSI y poder decir lo mucho que me ha aportado, aunque no niego que me haya traido, también, dolores de cabeza. Sin duda, el poder conocer a tanta gente por todo el país es algo que no cambio por nada.

Gracias a mis padres, Juan y Lourdes. Ellos tienen bastante culpa de que hoy esté donde estoy, de que hoy sea quien soy, simplemente de que todo haya funcionado y funcione en mi vida. Y a mi hermano, que siempre aparece de la forma más inoportuna, pero no por ello deja de ser alguien importante y en el fondo, nos queremos.

Gracias a todos aquellos que en mayor o menor medida me he topado en estos cuatro años de universidad, sean estudiantes, profesores, PAS, amigos o incluso «enemigos».

Este trabajo pone fin a 4 años, 4 años de estudio en la UPNA pero también de muchas más cosas. Si algo tengo ya claro es que no soy el mismo que entró por la puerta el día 2 de septiembre de 2011. No puedo serlo. Y es que estos años han dado para mucho y para mucha gente. Son años que no volverán pero que no se olvidarán.

«No hay ninguna razón para no seguir su corazón. [...] Tu tiempo es limitado, no lo desperdicies viviendo la vida de otros» (Steve Jobs, discurso de Standford, 2005)

Abstract

Many techniques used for solving specific problems in computer science depend strongly on functions employed for processing the information and determinating of the best parameters for the problem-solving techniques. In particular, it usually happens with processing images (segmentation) tasks, which are applied through fuzzy techniques in the use of functions and variables for the problem. Precisely, this project fosters the utilization of Dombi´s funtions in substitution of REF functions, in the field of black and white images so as to find the best path to segment through the use of tresholding of a group of images and to reuse generally this method. Finally, the use of penalty functions allows us to get the most accurate threshold. OWA functions for the construction of fuzzy sets are also studied.

Keywords: image segmentation, restricted equivalence functions, Dombi's functions, penalty functions, OWA functions.

Resumen

Muchas técnicas para la resolución de problemas específicos de computación dependen fuertemente de las funciones utilizadas para procesar la información y de la determinación de los mejores parámetros para las mismas. Esto es así, en particular, en problemas de procesamiento de imagen (segmentación) con técnicas difusas en los que, en gran medida, las funciones y variables dependen del problema considerado. En concreto, en este proyecto se propone utilizar funciones de Dombi para sustituir las funciones REF en la segmentación de imágenes en blanco y negro de forma que se encuentre la mejor forma de segmentar a través de la umbralización un conjunto de imágenes y poder llevar este proceso a la práctica de forma general. Posteriormente, por medio de funciones penalti se intenta obtener una umbralización lo más parecida posible a todas las llevadas a cabo. Se estudia, también, utilizar funciones OWA en la construcción de los conjuntos difusos.

Palabras clave: segmentación de imagen, funciones de equivalencia restringida (REF), funciones de Dombi, funciones penalti y funciones OWA.

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Índice de extractos de código

Capítulo 1.

Introducción

En este capítulo se da una visión general del problema que se ha estudiado así como la motivación para investigar y contribuir en este área. Se explica, también, la relevancia del problema en el marco del procesamiento de imagen y la segmentación. Finalmente, se especifican el propósito y los objetivos que se han guiado esta investigación

1.1. Motivación

El desarrollo de mecanismos que permiten a las máquinas el aprendizaje de técnicas que les capacitan para la resolución automática de problemas es un campo fundamental dentro del área de la Computación y la Inteligencia Artificial. Este tipo de procesos se utilizan en la vida diaria de muchos seres humanos y resultan innatos en ellos, en cambio, su implementación para que las máquinas los lleven a cabo se convierte en todo un reto debido a la dificultad de la imitación de los procesos cerebrales de las personas.

Muchos autores [3, 12, 28, 32] han propuesto ideas sobre esta cuestión pero debe destacarse a R. Penrose [26] que en su libro *La nueva mente del Emperador*, se proclama un claro detractor de la idea de que las máquinas puedan llegar a tener la opción de discernir de una forma similar al cerebro humano. Llega incluso a preguntarse "¿cómo podríamos siquiera *empezar* a explicar la substancia de tales problemas a una entidad que no sea ella misma consciente…?"

En definitiva, en este trabajo se sustenta en la idea de hacer posible lo que muchos creen imposible, lo que muchos creen que no será posible en mucho tiempo, incluso que no será posible sin conocernos a nosotros mismos. En este sentido, se trata el problema de la segmentación a través de segmentación de imágenes para intentar mejorar los métodos actuales y llegar a un método que pudiera ser utilizado de forma general para cualquier entrada.

1.2. Definición del problema

El problema de la segmentación consiste en poder conocer dentro de una imagen qué parte de ella pertenece al objeto u objetos y cual al fondo. Así, por ejemplo, en la figura 1.1 el lector podrá diferenciar claramente el molino del pueblo que se ve en el fondo de la imagen. El hecho de diferenciar un objeto del fondo de una imagen es un proceso que no supone ninguna dificultad para la mente humana; es algo que realizamos inconscientemente cientos de veces durante el día, sin que nos cueste ningún esfuerzo. Sin embargo, la misma tarea puede resultar muy complicada o imposible de realizar a través de un programa informático. Podría ser como aquel hidalgo que veía gigantes en lugar molinos[1].

Figura 1.1.: Distinguir el molino del pueblo del fondo no es difícil para un humano aunque sí para una máquina.

Este problema tiene una especial importante en la automatización de tareas, como el reconocimiento de direcciones postales, el realzamiento de textos o sistemas de calidad en grandes industrias, por ejemplo. También se utiliza en campos tan importantes como la medicina para poder ver en tiempo real la situación del tumor de un paciente. En definitiva, es importante ya que hace que un programa pueda tener como entrada una imagen en vez de valores discretos y obtener otra imagen que nos da muchísima información de la primera y que es facilmente interpretable por personas y más sencilla para las máquinas.

Autores como González y Woods [16] enuncian el problema de "segmentación de imágenes no triviales como una de las tareas más dificiles en el procesamiento de imágenes". En este sentido, insisten en que "la exactitud de la segmentación determina el éxito o error de los procesos de análisis computerizados". Otros autores [29] hablan de la sementación como una "técnica en al que se divide la imagen en partes que tienen una correlación con objetos o áreas del mundo real contenidas en la imagen".

[1] En el IV centenario de la publicación de la segunda parte de "El Quijote".

Definición 1.2.1. Dada una imagen Q que se puede subdividir en n regiones $R_1, \dots, R_n$, y conocida P que es una cierta propiedad booleana que cumplen todos los píxeles de la región $R_i, \forall i = 1, \dots, n$, se deberá cumplir siempre que:

(1) $\bigcup_{i=1}^{n} R_i = Q$;

(2) En una región $R_i, \forall i = 1, \dots, n$ todos sus píxeles están conectados;

(3) $R_i \cap R_j = \varnothing, \forall i, j : i \neq j$;

(4) $P(R_i) =$ VERDADERO $, \forall i = 1, \dots, n$;

(5) $P(R_i \cup R_j) =$ FALSO $, \forall i = 1, \dots, n$.

Se hablará de segmentación monoumbral cuando se obtenga únicamente un objeto y el fondo con un único umbral t. Para más de un objeto se dirá que el problema es multiumbral, con un vector de umbrales $(t_1, \dots, t_{n-1})$.

En conclusión, en este proceso se lleva a cabo la división de la imagen en regiones donde cada una (desde 2 hasta n, este número dependerá del problema que estemos resolviendo) harán mención al fondo y a cada objeto. Además, todas las regiones serán independientes entre sí, esto es, un pixel pertenecerá solamente a la región i cuando hablemos de segmentación completa. Centrando el problema únicamente en aquellas imágenes en escala de grises, nuestra pretensión será obtener aquellas regiones que contienen a un objeto a través de su histograma (frecuencia de los tonos de gris), esto es, por medio de técnicas de umbralización.

1.3. Solución propuesta

Para llevar a cabo este estudio se ha utilizado la representación de las imágenes por medio de conjuntos difusos. Estas técnicas, en comparación con las habituales de lógica clásica, consiguen evitar que haya perdida de información en las imágenes. Esto se debe a que en la obtención y discretización de las imágenes se llevan a cabo simplificaciones sobre la definición real con intención de poder ser representado digitalmente.

Asimismo, la incertidubre que puede haber en torno a los problemas relacionados con el procesamiento de imagen es alto. En particular, la segmentación busca separar de forma excluyente los objetos del fondo, algo que se hace imposible de definir de forma certera. Se conoce que en un cambio de intensidad suficientemente fuerte estará un borde pero, justamente, por el hecho de que el propio enunciado sea de por sí difuso (suficientemente

fuerte), será necesario poder representar esta incertidumbre, algo que haremos por medio de los conjuntos que definió Zadeh.

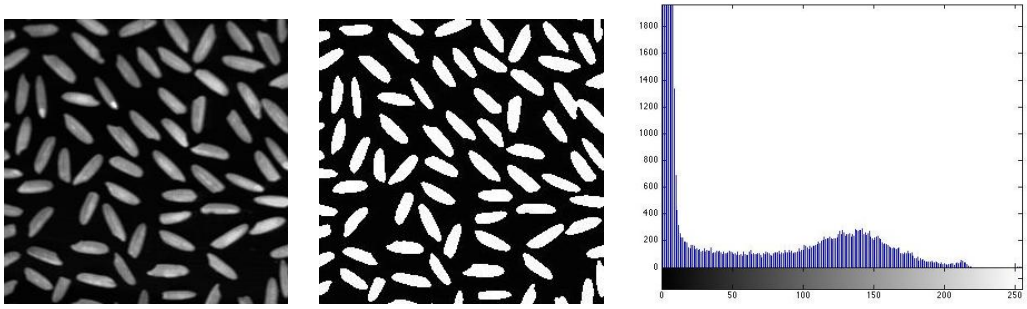
Además, existen tres técnicas para poder llevar a cabo la segmentación de una imagen [16]:

a) *Segmentación basada en umbralización.* A través de uno o varios umbrales se obtiene una forma de clasificar todos los píxeles de la imagen en una única región buscando que las regiones tengan un cierto significado.

b) *Técnicas basada en agrupamiento de píxeles en regiones.* Se procede clasificando cada uno de los píxeles de la imagen dentro de las regiones buscando que estas se creen con los píxeles más similares. Se suele indicar el número de regiones que se desea obtener.

c) *Técnicas basadas en la detección de bordes.* Basan su técnica en la identificación de los píxeles en la frontera. Se detecta el objeto y el fondo, ya que de esta forma queda definido también su contorno.

En este trabajo se va a centrar la segmentación por medio de las técnicas de umbralización. Para ello lo que tendremos que hacer será calcular los umbrales que separen las regiones que se hayan encontrado, de forma que situaremos las regiones entre un umbral t_i y otro t_{i+1}. Para ejemplificar esto, tomaremos la umbralización binaria en la cual se dispondrá de un único umbral t. De esta forma, todos los elementos de la imagen que se encuentren por encima del umbral pertenecerán a una región y los que estén por debajo a la segunda. Esta técnica se basa únicamente en detectar los diferentes tonos de gris de la imagen, así que mirando el histograma de la imagen (fig. 1.2), podríamos ver cómo existe una frontera entre la intensidad $t = 71$ y el resto creando diferencias en los niveles de gris. [5]



(a) Imagen original. (b) Imagen segmentada. (c) Histograma.

Figura 1.2.: Proceso de segmentación. Se puede ver que en $t = 71$ se produce un mínimo con respecto a los otros niveles de gris y que por eso se escoge como umbral.

Se debe tener en cuenta que la umbralización es un método rápido y de coste

computacional bajo por lo que se puede realizar en tiempo real. Como sólo basa su algorítmica en el histograma de la imagen hace que sea un método sencillo e intuitivo, aunque esto también hace que tenga problemas ante el ruido y objetos o fondos que no sean uniformes.

La solución que se presenta en este trabajo se obtiene por medio de lógica difusa. Se han utilizado funciones REF, de Dombi, de agregación y penalti. En el capítulo 2 se hace mención a todos estos conceptos y se explican poníendose en práctica a partir del capítulo 3. En el capítulo 5 se presentan todas las conclusiones obtenidas así como las líneas de trabajo futuro.

1.4. Relevancia

En el campo de la medicina [29, 30] se ha experimentado una mejora sustancial en la efectividad de las pruebas médicas gracias a diferentes opciones como los rayos X, la tomografía computerizada, la resonancia magnética, la tomografía por emisión de positrones (PET), imágenes de ultrasonidos y otras. La revolución digital y el gran poder de procesamiento que disponen los ordenadores ha conseguido que los profesionales comprendan la compleja anatomía humana, aunque esto no ha sido suficiente ya que se ha visto necesario poder obtener los bordes, las superficies y la segmentación de los órganos. Estos órganos segmentados y sus bordes son clave para poder conseguir que un especialista médico pueda hacer una cirugía adecuada para muchas ramas de la medicina, debido a la importancia de tener datos en tiempo real [18].

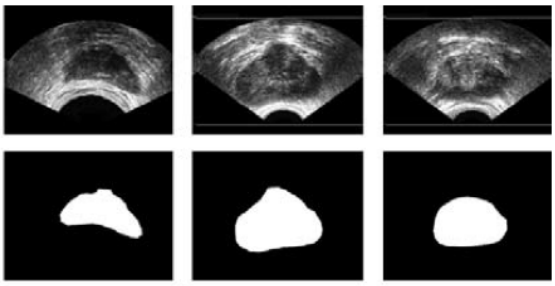


Figura 1.3.: Segmentación de una imagen de próstata [8].

Por otra parte, en los sistemas de seguridad hoy en día se utilizan factores biométricos como clave. Un claro ejemplo es el uso de la huella dactilar. En este sentido, antes de llevar a cabo el reconocimiento de la huella se hace imprescindible aplicarle la segmentación para poder obtener la mayor información posible de la huella. Se utiliza también la segmentación en procesos de calidad, sobre todo de cadenas de producción, donde todos los productos de salida son iguales. En este caso, será sencillo detectar diferencias entre

la salida adecuada y la defectuosa para poder retirarla. Esto se generaliza en la detección de errores en diferentes campos de la industria y la sociedad.

Figura 1.4.: Utilización de técnicas de segmentación para extraer la información de una matrícula y poder ser procesada por un sistema de IA.[19].

En definitiva, el problema de la segmentación de la imagen se hace importante ya que busca que la entrada del problema no sean datos concretos sino imágenes que después se interpretarán. Para que esta interpretación se correcta, el poder disponer de una información certera, sencilla y limpia se hace fundamental.

1.5. Propósito y objetivos

Este proyecto se centra en estudiar técnicas de segmentación para un único umbral en imágenes en blanco y negro e intentar mejorar las técnicas de las que actualmente se disponen intentando generalizarlas de forma que los parámetros no dependan del problema.

Además, para poder conseguir el propósito anterior se estipularon los siguiente objetivos:

- Investigar y conocer técnicas actuales de segmentación de imagen de forma que estas sean el punto de partida.
- Analizar y evaluar las funciones que J. Dombi propone en [13]. Comparar estas con las funciones REF y sustituirlas en la construcción de los conjuntos difusos para conocer sus efectos.
- Implementar diferentes algoritmos de segmentación con el conocimiento adquirido anteriormente evaluando su mejora y haciendo las correcciones necesarias con la intención de generalizar el método de forma máxima.
- Dirimir cual es el mejor umbral a través de la agregación de resultados y obtención del mejor con funciones penalti.

- Implementar diferentes algoritmos que incluyan la agregación OWA de las diferentes funciones estudiadas para la segmentación en los puntos anteriores comprobando si esto mejora los resultados anteriores.

- Analizar todos los puntos anteriores a fin de concluir los resultados del trabajo así como dirimir si se ha podido conseguir cumplir el propósito inicial.

Capítulo 2.

Conceptos básicos

Este capítulo pretende ser una introducción a todos los conceptos teóricos necesarios para la correcta comprensión del trabajo que se detalla en esta memoria.

2.1. Procesamiento digital de imagen

Un imagen Q puede ser entendida como una función de dos variables, $q(x, y)$, donde x e y son las coordenadas en el plano de cada elemento (píxel) y la función da como resultado el nivel de gris o intensidad asociada a ese píxel. En concreto, cuando x e y así como la intensidad $q(x, y)$ son finitas y discretas hablaremos de que tenemos una imagen digital. Por eso mismo, el procesamiento digital de imagen se puede definir como aquel que se hace con imágenes digitales a través de un ordenador. Tiene su origen con la introducción de las primeras imágenes digitales a principios de 1920, donde se enviaban imágenes entre Londres y Nueva York por medio de un cable submarino.

Es claro que los seres humanos disponemos del sentido de la vista como el sentido más desarrollado para interactuar con el medio, pero está limitada a la banda visible dentro del espectro electromagnético. En cambio, el procesamiento digital de imagen puede cubrir el análisis de todo el espectro electromagnético, lo que incluye los ultrasonidos, el infrarrojos, rayos X, etc. En consecuencia, el procesamiento digital es aplicado en numerosos ámbitos de la sociedad y la vida diaria.

Dentro del procesamiento de imagenes digitales se encuentra la visión artificial. Esta es un subcampo de la inteligencia artificial (IA) y pretende emular la visión humana, incluyendo la capacidad de aprender así como el manejo directo de imágenes como datos de entrada a un problema dado. Actualmente, este campo está poco desarrollado [16] ya que el avance está siendo mucho más lento de lo que se esperaba en un primer momento. El análisis de imagen, que pretende entender cómo está formada la imagen, está situada a

caballo entre la visión artificial y el procesamiento de imágen, y será lo que utilizaremos en este trabajo.

No hay una clara frontera entre el procesamiento de imagen y la visión artificial, esto hace que se hable de un paradigma que incluye 3 tipos de procesos; de bajo nivel, medio nivel y alto nivel. Dentro del nivel bajo se pueden incluir todos los procesos primitivos que tienen como objetivo reducir el ruido, realzar el contraste o ajustar la nitidez, por ejemplo. En un segundo nivel, más avanzado, se encuentran todos los procesos que tienen como entrada una imagen y obtienen atributos de esta. Pueden ser procesos como la segmentación (que se tratará aquí), descripción de objetos de la imagen o recocimiento de los mismos. En el último nivel se encuentran todas aquellas técnicas que hacen que todo "tenga sentido" ya que hacen que el análisis imite a la forma cognitiva de la mente humana.

2.1.1. Imágenes digitales

Como ya se ha explicado en el apartado anterior, se habla de imagen digital cuando se puede ser capaz de determinar todos sus elementos (píxeles), es decir, estos son de forma finita y discreta. A partir de esta idea, se dispone de dos formas de representación para las imágenes en niveles de gris, en los rangos $[0,1] \in \mathbb{R}$ y $[0,255] \in \mathbb{N}$. En el primer caso diremos que la imagen está normalizada. Además, las imágenes digitales serán representadas por una matriz que contendrá cada uno de sus píxeles de manera ordenada.

$$\begin{pmatrix} 87 & 192 & 178 \\ 255 & 65 & 227 \\ 57 & 129 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0.3412 & 0.7529 & 0.6980 \\ 1 & 0.2549 & 0.8902 \\ 0.2235 & 0.5059 & 0 \end{pmatrix}$$

(a) Niveles de gris (b) Normalizada (c) Gráfica

Figura 2.1.: Imagen digital en diferentes representaciones.

Como se puede apreciar en la figura 2.1 cuanto más cercano a 0 sea el número, más negro será el nivel de gris, y cuanto más cercano a 1 ó 255, más blanco. En el caso de imágenes en color, se utiliza el formato RGB (*Red*, *Green* y *Blue*) donde existe una matriz como las anteriores por cada uno de los colores y al superponerlas se obtiene la imagen que vemos habitualmente.

Definición 2.1.1. Se define el histograma de una imagen Q con niveles de gris en el intervalo [0, 255] como la función $h(q) = n_q$ donde n_q es el número de píxeles en la imagen con la intensidad q.

Un ejemplo gráfico de la función de histograma para una imagen puede encontrarse en la figura 1.2.

2.1.2. Herramienta para el procesamiento digital: MatLab

MatLab (abreviatura de *Matrix Laboratory*) es una herramienta de software matemático que ofrece un lenguaje de programación (lenguaje M) y un entorno de desarrollo integrado. Fue creado en 1984 por el informático y matemático Cleve Moler el cual buscaba una forma alternativa de ejecutar programas de álgebra en *Fortran*.

Entre sus prestaciones básicas están la manipulación de matrices, la representación de datos y funciones o la implementación de algoritmos así como la creación de interfaces de usuario (GUI). Además, el paquete básico de MatLab puede ser expandido por medio de *toolboxes*, como es el caso de este trabajo, donde se utilizará la relativa a procesamiento de imagen. Se puede disponer también del software *Simulink* para trabajar conjuntamente a MatLab.

2.1.3. Contraste

El contraste es aquel concepto que nace de la diferencia de la intensidad más alta y más baja de una imagen. Esto se puede apreciar de manera clara en el histograma. Así, cuando una imagen dispone de un gran rango dinámico (intervalo para el cual el histograma es diferente a 0) entonces tendrá un contraste alto y viceversa. En la figura 2.2 se puede ver un ejemplo.

(a) Imagen original (b) Imagen con poco contraste (c) Imagen con muy poco contraste

Figura 2.2.: Imagen del fotógrafo con diferentes contrastes

Para llevar a cabo este ajuste en MatLab, se utiliza la función `imadjust(img_var, extremos_entrada, extremos_salida)` donde mintinlinematlabimg_var es la variable donde está almacenada la imagen en tipo `uint8`, mintinlinematlabextremos_entrada es un vector que indica entre cuales están los valores que se deben modificar (en nuestro caso será vacío para que coja todo el histograma) y de manera simétrica mintinlinematlabextremos_salida es el rango donde debe encontrarse el histograma de la imagen de salida.

2.1.4. Ruido en las imagenes

Se condidera que una imagen está degradada cuando tiene ruido, esto es, cuando tiene defectos con respecto a la imagen original (fig. 2.3). La principal fuente de ruido en images digitales se da en la adquisición o transmisión de las imagenes. Esto se puede deber a las condiciones ambientales o a la calidad de los sensores durante la toma de la imagen. Por ejemplo, la luminosidad, el polvo en el ambiente, etc. pueden ser determinantes. Por otra parte, en el caso de la transmisión, las imágenes pueden ser corrompidas por interferencias en el medio de transmisión, principalmente. Esto es claro cuando se transmite una imagen de un satélite, ya que la atmósfera puede interferir y provocar que la imagen recibida en la Tierra no sea exactamente igual.

(a) Imagen original

(b) Imagen con ruido 'sal y pimienta'

(c) Imagen con ruido gausiano

Figura 2.3.: Imagen de Lena con diferentes tipo de ruido

Las imágenes con ruido también se pueden crear artificialmente. Para ello utilizamos una cierta función H de degradación, a la que añadimos un ruido al que llamaremos η de forma que obtendremos una nueva imagen $g(x, y)$ que tendrá una serie de imperfecciones. Todo el ruido creado de esta forma se aplicará directamente sobre los píxeles de la imagen lo que provocará cambios en su histograma en una forma u otra. Aun así, debemos tener en cuenta que no hay relación directa entre la intensidad nueva que tienen los píxeles y la anterior.

Existen muchos tipos de ruido como el exponencial, el gamma, el uniforme, etc. pero en esta memoria se han utilizado, el gausiano y el de tipo impulsivo o de 'sal y pimienta'.

Ruido gausiano

Este puede ser uno de los modelos de ruido más utilizado en la práctica debido a que tiene una gran maleabilidad matemática. Basa su forma de actuar sobre la imagen en la función gausiana de probabilidad:

$$p(z) = \frac{1}{\sqrt{1}2\pi\sigma} e^{\frac{-(z-\mu)^2}{2\sigma^2}}$$

donde z representa la intensidad, μ la media de z y σ la desviación estándar. Este tipo de ruido puede crearse en MatLab a través de la función `imnoise(img_var, 'gaussian')` sabiendo que `img_var` es la variable donde está almacenada la imagen en tipo `uint8`.

Ruido de tipo 'sal y pimienta'

Este tipo de ruido, formalmente llamado impulsivo, hace que se conviertan píxeles de forma aleatoria en píxeles con intensidad 0 y 255. Esta es la razón de su particular nombre. De nuevo, fijándonos en MatLab, podremos crear imágenes con este tipo de ruido por medio del comando `imnoise(img_var, salt & pepper', prob_val)`, de nuevo `img_var` es la variable donde se almacena la imagen y `prob_val` es un valor entre 0 y 1 que hace las veces de probabilidad de la aparición del ruido.

En este trabajo, se han utilizado imágenes con ruido para comprobar la adecuación o no de los algoritmos de segmentación al ruido. Ejemplos de ambos tipos de ruido se han presentado en la figura 2.3.

2.2. Lógica difusa

La lógica difusa fue introducida por el matemático L. A. Zadeh [35] en 1965 con la intención de poder extender la lógica clásica o *crips* de forma que permitiera manejar y procesar la información que se compone de términos inexactos, imprecisos o subjetivos. Es, al fin y al cabo, un intento de imitar la forma de deducción del cerebro humano para trasladarlo a las máquinas. Se comenzará recordando la lógica clásica para después extender los conceptos a la difusa.

En primer lugar, definiremos el universo finito, U, con el que se trabajará, de forma que contenga todos aquellos elementos con los que se desee trabajar, esto es, $U = \{u_1, u_2, \dots, u_n\}$. En la lógica clásica simplemente asignamos verdadero o falso a cada uno de los elementos del conjunto que se estudia. Por esta razón, al definir un conjunto A podremos decir que este contiene o no a los elementos del universo U con una certeza absoluta. De esta manera, daremos un valor de 1 cuando u_i esté incluido en A (verdadero) y un valor de 0 cuando no lo esté (falso). En esta última idea se refleja por medio de la función de pertenencia al conjunto A, μ_A (ecuación 2.1).

$$\begin{aligned} A &= \{(u_i, \mu_A(u_i)) | u_i \in U\} \\ \mu_A &: U \to \{0, 1\} \text{ tal que} \\ \mu_A(u) &= \begin{cases} 1 & \text{si} \quad u \in A \\ 0 & \text{si} \quad u \notin A \end{cases} \end{aligned} \tag{2.1}$$

Podemos considerar el problema de determinar si una persona es alta o no. Para la lógica clásica, este razonamiento se simplificará en buscar un valor a partir del cual podamos definir que cierta persona es alta. Para el siguiente ejemplo, tomaremos 1,75 metros como referencia, donde A es el conjunto de las personas altas. De este modo, $\mu_A = 1$ si $u > 1,75$ y en otro caso $\mu_A = 0$. En la figura 2.4 se puede ver la representación del concepto 'alto' para las posibles alturas que se pueden presentar.

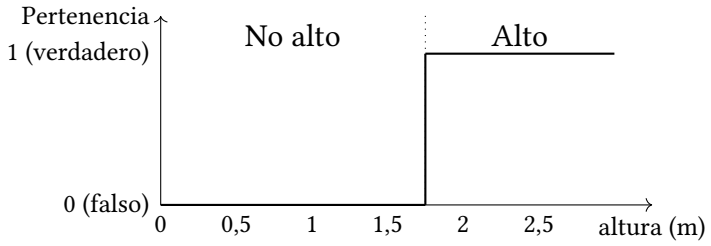


Figura 2.4.: Representación de la pertenencia de los elementos para la lógica clásica.

Por otra parte, la lógica difusa dispone de la función de pertenencia más compleja, lo que nos hace poder decir que alguien es 'poco alto' o 'bastante alto' ya que no daremos un par de valores ($\{0, 1\}$) sino cualquiera de los contenidos en el intervalo que definen. Así μ_A será una función que asigna un valor entre 0 y 1 a cada elemento de A.

$$\begin{aligned} A &= \{(u_i, \mu_A(u_i)) | u_i \in U\} \\ \mu_A &: U \to [0, 1] \end{aligned} \tag{2.2}$$

Si continuamos con el ejemplo se verá que el conjunto alto (A) esta vez se define como explica la ecuación 2.3.

$$\mu_A(u) = \begin{cases} 1 & \text{si} \quad u \geq 2 \\ 2u - 3 & \text{si} \quad 1.5 > u > 2 \\ 0 & \text{si} \quad u \leq 1.5 \end{cases} \tag{2.3}$$

Esta nueva función de pertenencia hace que podamos distinguir 3 zonas dentro de su representación (figura 2.5). Se tendrá de nuevo la etiqueta 'alto' y 'no alto' que hacen que su pertenencia sea certera. Se dispondrá, también, una parte de la pertenencia a la que llamaremos 'difuso' donde el conjunto no afirma ser ni 'alto' ni 'no alto' sino que está en una situación intermedia. En este caso se habla de que el elemento a_i que se encuentra ahí pertenece a A con grado $\mu_A(a_i)$.

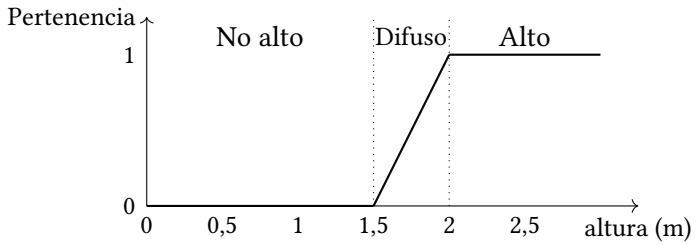


Figura 2.5.: Representación de la pertenencia de los elementos para la lógica difusa.

Más formalmente, denotaremos al conjunto que incluye a todos los conjuntos difusos como $\mathscr{F}(u)$ a todos aquellos que se encuentran definidos sobre un referencial finito ($|U| = n$) por lo que U no será un conjunto vacío.

2.3. Funciones de equivalencia restringida (REF)

El concepto de función de equivalencia restringida (REF por sus siglas en inglés) surge del concepto de equivalencia y el de similitud [5]. Este concepto es muy utilizado para la comparación de imágenes y su intención es dar una medida de cómo de iguales o similares son dos elementos x e y. Para poder definir el concepto de REF, necesitamos varios previamente.

Definición 2.3.1. Una negación estricta es aquella función $c : [0,1] \to [0,1]$ que cumple que $c(0) = 1$ y $c(1) = 0$ y es estrictamente decreciente y continua. Además, si c es involutiva se considera que se habla de una negación fuerte.

Definición 2.3.2. Llamaremos automorfismo (φ) a todas aquellas funciones del intervalo unidad tal que $\varphi : [0,1] \to [0,1]$ sean continuas y estrictamente crecientes propiciando que $\varphi(0) = 0$ y $\varphi(1) = 1$.

En 1979, E. Trillas [31] enunció el siguiente teorema.

Teorema 2.3.3 (Teorema de Trillas). *Una función $c : [0,1] \to [0,1]$ es una negación fuerte si, y sólo si, existe un automorfismo del intervalo unidad tal que $c(x) = \varphi^{-1}(1 - \varphi(x))$.*

Definición 2.3.4. Una función $REF : [0,1] \to [0,1]$ es llamada de equivalencia restringida cuando cumple que:

(1) $REF(x,y) = REF(y,x), \forall x,y \in [0,1]$;

(2) $REF(x,y) = 1$, si y sólo si, $x = y$;

(3) $REF(x,y) = 0$, si y sólo si, $x = 1$ e $y = 0$ ó si $x = 0$ e $y = 1$;

(4) $REF(x,y) = REF(c(x),c(y)), \forall x,y \in [0,1]$, siendo c una negación fuerte.

(5) $\forall x,y,z \in [0,1]$, si $x \leq y \leq z$, entonces $REF(x,y) \geq REF(x,z)$ y $REF(y,z) \leq REF(x,z)$

Proposición 2.3.5. *Sean dos automorfismos φ_1 y φ_2, se llamará función REF a la construcción que cumpla que:*

$$REF(x,y) = \varphi_1^{-1}(1 - |\varphi_2(x) - \varphi_2(y)|) \quad con \quad c(x) = \varphi_2^{-1}(1 - \varphi_2(x)).$$

Además, si tenemos una REF y un automorfismo en $[0,1]$, la aplicación de estos ($F = \varphi \circ REF$) es otra REF.

Teorema 2.3.6. *Una función continua $REF : [0,1]^2 \to [0,1]$ que cumpla que $REF(1,x) = x, \forall x \in [0,1]$, es una función REF asociada con la función $I : [0,1]^2 \to [0,1]$ con I continua y asociativa si y sólo si existe un automorfismo φ en $[0,1]$ tal que:*

$$REF(x,y) = \varphi^{-1}(1 - \left|(\varphi(x) - \varphi(y))\right|) \quad y \quad c(x) = \varphi^{-1}(1 - \varphi(x))$$

2.4. Funciones de agregación

Las funciones de agregación tienen como propósito reducir las dimensiones de la información a partir de la combinación de los datos de entrada obteniendo una salida que los represente [10, 22]. Su aplicación se extiende en muchos casos prácticos y teóricos como la lógica multievaluada, control difuso, la toma de decisión, etc [25]. Se definirá una función de agregación como sigue:

Definición 2.4.1. Se dice que $M : [0,1]^n \to [0,1]$ es una función de agregación de dimensión n siempre que satisfaga:

(1) $M(x_1, \dots, x_n) = 0$ si y sólo si $x_1 = \dots = x_n = 0$;

(2) $M(x_1, \dots, x_n) = 1$ si y sólo si $x_1 = \dots = x_n = 1$;

(3) M es una función estrictamente creciente.

Definición 2.4.2. Una función de agregación M será llamada media si

$$min(x_1, \dots, x_n) \leq M(x_1, \dots, x_n) \leq max(x_1, \dots, x_n).$$

En este proyecto se utilizarán mayoritariamente las funciones de agregación idempotentes, esto es, que cumplen que $M(x, \dots, x) = x, \forall x$. En particular, una función de agregación idempotente es una media. Entre las funciones que utilizaremos encontramos la media aritmética, el mínimo, el máximo o la media geométrica. Además, se definen a continuación otras que no son de uso tan común.

2.4.1. Funciones OWA

En esta sección se introduce el concepto de las funciones de media ponderada ordenada (OWA, *ordered weighted averaging*) [6, 25, 34]. Se basan en la idea de generar una media, ordenando primero los elementos a agregar, para luego darles mayor relevancia a una parte de ellos. Este tipo de función generaliza la media aritmética, siendo esta el OWA en el que todos los elementos del vector de pesos son iguales.

Definición 2.4.3. Una función $F : [0,1]^n \to [0,1]$ será una función OWA de dimensión n si existe un vector $w = (w_1, w_2, \dots, w_n) \in [0,1]^n$ tal que $\sum_i w_i = 1$ de forma que

$$F(x_1, \dots, x_n) = \sum_{j=1}^{n} w_j x_{\sigma(j)}$$

donde $x_{\sigma(j)}$ es el j-ésimo mayor elemento del vector $(x_1, \dots, x_n)$.

Por tanto para poder obtener un resultado adecuado, deberá utilizarse un vector de pesos w que se adecúe a las necesidades del problema. En este trabajo se emplea la versión 'al menos la mitad' que tiene como vector de pesos aquel que se obtiene con la ecuación 2.4. Esto generará una agregación donde destaquen todos aquellos elementos que se encuentren por detras de la mediana.

$$w_i = Q\left(\frac{i}{t+1}\right) - Q\left(\frac{i+1}{t+1}\right), \forall i \in \{1, \dots, n\}, \text{ sabiendo que} \tag{2.4}$$

$$Q(r) = \begin{cases} 0 & \text{si} \quad r < 0,5 \\ \dfrac{r - 0,5}{0,5} & \text{si} \quad 0,5 \leq r \leq 1 \\ 1 & \text{si} \quad r > 1 \end{cases}$$

Además, en la segunda parte del estudio, se emplea el OWA 'la mayoría de'. Este OWA pretende destacar aquellos elementos que son mayores en el vector x. Esto se hace realidad por medio de un vector de pesos donde son nulos los pesos relativos a aquellas x_i más pequeñas. Para la construcción del vector de pesos, se utilizará, también, la ecuación 2.4

2.4.2. Integral Choquet

Para este otro tipo de función [11, 23] de agregación se pretende dar una nueva de forma de representar un conjunto de datos en una única salida. De esta forma, se define primeramente una medida a través de la cual se calculará la forma en la que cada elemento del conjunto de datos tendrá reelevancia en la agregación final. Para ello definimos el concepto de medida difusa.

Definición 2.4.4. Dado U un universo finito; $\mathscr{P}(U)$ el conjunto de todos los subcojuntos de U. Una medida difusa es una función $\mu : \mathscr{P}(U) \to [0,1]$ que satisface que:

(1) $\mu(\varnothing) = 0$ y $\mu(U) = 1$.

(2) $A \subseteq B \Rightarrow \mu(A) \leq \mu(B), \forall A, B \subseteq U$.

A continuación definimos la función integral Choquet conociendo que tomaremos su versión discreta por el contexto en el que se está trabajando.

Definición 2.4.5. Dado un vector $(x_1, \dots, x_n)$, sea σ una permutación de $\{1, ..., n\}$ tal que $x_{\sigma(j)}$ es el j-ésimo mayor elemento del vector $(x_1, \dots, x_n)$. La integral discreta de Choquet con respecto a la medida difusa μ es

$$Ch_\mu(x) = \sum_{i=1}^{n} x_{\sigma(i)}(\mu(\{\sigma(i), \dots, \sigma(n)\}) - \mu(\{\sigma(i+1), \dots, \sigma(n)\}))$$

tomando la convención de que $\{\sigma(n+1), \sigma(n)\} = \varnothing$.

Proposición 2.4.6. *Si denotamos a* $w^{\mu}_{\sigma,i} = \mu(\{\sigma(i), \dots, \sigma(n)\}) - \mu(\{\sigma(i+1), \dots, \sigma(n)\})$ *se obtiene la siguiente definición de la integral Choquet en función de los operadores OWA definidos en 2.4.3:*

$$\sum_{i=1}^{n} w^{\mu}_{\sigma,i} \cdot x_{\sigma(i)}.$$

2.5. Funciones de similitud

El concepto de similitud [14, 15] surge cuando queremos medir cómo de parecidos son dos conjuntos difusos. Este concepto es muy parecido al de REF, pero en este caso se consideran conjuntos y no elementos. Por esta razón, utilizamos las REF como base para definir las funciones de similitud.

Definición 2.5.1. Dada una función M de agregación (definición 2.4.1) y una función REF (definión 2.3.4) llamaremos a SM función de similitud si $SM : \mathscr{F}(X) \times \mathscr{F}(X) \to [0,1]$ está definida tal que

$$SM(A,B) = M_{i=1}^{n} REF(\mu_A(x_i), \mu_B(x_i))$$

y satisface las siguientes condiciones:

(1) $SM(A,B) = SM(B,A), \forall A, B \in \mathscr{F}(X)$;

(2) $SM(A, A_c) = 0$, si y sólo si A no es difuso;

(3) $SM(A,B) = 1$ si y sólo si A = B;

(4) Si $A \leq B \leq C$, entonces $SM(A,B) \geq SM(A,C)$ y $SM(C,B) \geq SM(C,A)$;

(5) $SM(A_c, B_c) = SM(A,B)$

Observación 2.5.2. Durante el desarrollo de este trabajo se buscará la similitud entre conjuntos difusos y el conjunto $\tilde{1}$. Se define el conjunto $\tilde{1} = \{(u_i, \mu_{\tilde{1}}(x) = 1) | u_i \in U\}$, esto es, aquel en el que todos sus elementos tienen pertenencia absoluta.

2.6. Funciones penalti

Una función penalti [9] es una función de agregación, por lo que dispone de un vector de entradas del que devuelve un único resultado. Si todos los elementos del vector son iguales, entonces, claramente, la salida será eso mismo. Ahora bien, el problema surge cuando hay algún elemento diferente ya que en ese momento la idea será buscar una salida todo lo parecida posible a la entrada. En este sentido, es elemento o elementos diferentes tendrán una cierta discrepancia con los demás y justamente esto es lo que se pretende minimizar, la discrepancia, para dar una salida adecuada.

Definición 2.6.1. La función $P : [a,b]^{n+1} \to \mathbb{R}^+ = [0, \infty]$ es una función penalti si y sólo si satiface que:

(1) $P(x,y) \geq 0, \forall x, y$

(2) $P(x,y) = 0$ si $x_i = y \forall i = 1, \dots, n$

(3) $P(x,y)$ es cuasiconvexa en y para cualquier x, esto es, $P(x, \lambda \cdot y_1 + (1-\lambda) \cdot y_2) \leq max(P(x,y_1), P(x,y_2))$.

La función en la que se basan las penalti es

$$f(x) = arg\min_{y} P(x, y)$$

si y es el único mínimo e $y = \frac{a+b}{2}$ si el conjunto de minimizadores es el intervalo (a, b).

Teorema 2.6.2. *Todas las funciones de agregación llamadas medias pueden ser escritas como una función basada en una función penalti expresada en la definición 2.6.1.*

2.7. Funciones de Dombi

Las funciones de Dombi [13] son operadores de equivalencia con una definición diferente a las REF presentadas en la sección 2.3. Con esta relación lo que se pretende conocer es como de iguales son dos elementos de un conjunto difuso dado, pero aplicando nuevas ideas para la construcción del resultado.

Definición 2.7.1. Dados $x = (x_1, x_2, \dots, x_n)$ y $w = (w_1, w_2, \dots, w_n)$, denotaremos D como una función de equivalencia de Dombi cuando tengamos que

$$D(w, x) = \frac{1}{2}\left(1 + \prod (1 - 2x_i)^{w_i}\right)$$

Lema 2.7.2. *La función de equivalencia de Dombi, D, cumple las siguientes propiedades:*

(1) $D : [0,1] \times [0,1] \to [0,1]$ *es continua;*

(2) $D((w_1, w_2), (0,0)) = 1; \quad D((w_1, w_2), (1,1)) = 1;$

(3) $D((w_1, w_2), (0,1)) = 0; \quad D((w_1, w_2), (1,0)) = 0;$

(4) $D((w_1, w_2), (x, c(x))) = 0.$

2.8. Notación

A lo largo del trabajo se asumirá la notación que sigue para estos conceptos.

Imagen : la denotaremos con Q.

Coordenadas de un pixel : (x, y).

Máximo nivel de gris : L.

Número de filas de Q : N.

Número de columnas de Q : M.

Intensidad de un pixel : $q(x, y)$ de forma que $0 \leq q(x, y) \leq L - 1, \forall (x, y) \in Q$.

Histograma : $h(q)$. Función para conocer el número de pixeles con la intensidad q.

Media de una imagen :

$$m_Q = \frac{\sum_{q=0}^{L-1} qh(q)}{\sum_{q=0}^{L-1} h(q)}$$

Área de una imagen :

$$A(Q) = \sum_{q=0}^{L-1} qh(q)$$

Capítulo 3.

Segmentación de imágenes con un solo umbral con funciones de Dombi

En este capítulo en primer lugar se presentarán todas las herramientas algorítmicas que se van a utilizar para llevar a cabo los experimentos de este trabajo, haciendo énfasis en la forma de utilización para funciones de Dombi. Seguidamente se mostarán los resultados obtenidos para la segmentación monoumbral con estas funciones.

3.1. Métodos algorítmicos de segmentación de un único umbral

3.1.1. Algoritmo general maximizando la similitud

Este algoritmo, que se presenta en [4], pretende conseguir obtener un único umbral a partir de la maximización de la similitud.

Algoritmo 1 Maximización de la similitud

Entrada: Una imagen Q en escala de grises donde sus píxeles estén entre 0 y $L-1$.
Salida: El umbral t a partir del cual se divide Q en objeto y fondo.

1: **para** t:=0 hasta $L-1$ **hacer**
2: Divisón de la imagen en dos clases $C_b(t)$ y $C_o(t)$. Para cada una de estas clases, calcular su media: $m_b(t)$ y $m_o(t)$.
3: Construcción del conjunto difuso Q_t.
4: Calcular la $SM(\tilde{1}, Q_t)$.
5: **fin para**
6: **devolver** $\{t \mid \max(SM)\}$

En el algoritmo anterior se necesitan varias definiciones que se explican a continuación. En primer lugar, se describe el método de creación de los conjuntos difusos para lo que se explica previamente el cálculo de las medias para el fondo y el objeto.

Definición 3.1.1. Teniendo en cuenta la definición de la media de una imagen que se ha dado en el apartado 2.8, y disponiendo del histograma de la imagen $h(q)$ para un cierto nivel $q, \forall q \in Q$, se define la media de los píxeles del fondo como:

$$m_b(t) = \frac{\sum_{q=0}^{t} qh(q)}{\sum_{q=0}^{t} h(q)};$$

y para los píxeles del objeto como:

$$m_o(t) = \frac{\sum_{q=t+1}^{L-1} qh(q)}{\sum_{q=t+1}^{L-1} h(q)}.$$

Definición 3.1.2. Dada Q, una imagen en la escala de L niveles de gris, y t, un nivel de gris de forma que $0 \leq t \leq L-1$. Teniendo en cuenta que F es una función REF ya que la $REF \circ \varphi$ lo es, se define el conjunto

$$Q_t = \{(q, \mu_{Q_t}(q)) | q \in \{0, 1, \dots, L-1\}\}$$

teniendo en cuenta que

$$\mu_{Q_t}(q) = \begin{cases} F\left(\frac{q}{L-1}, \frac{m_b(t)}{L-1}\right) = \varphi\left(REF\left(\frac{q}{L-1}, \frac{m_b(t)}{L-1}\right)\right) & \text{si } q \leq t, \\ F\left(\frac{q}{L-1}, \frac{m_o(t)}{L-1}\right) = \varphi\left(REF\left(\frac{q}{L-1}, \frac{m_o(t)}{L-1}\right)\right) & \text{si } q > t. \end{cases}$$

Observación 3.1.3. Como parece lógico a la vista de la definición 3.1.2, es necesario poder enumerar aquellas funciones que se utilizarán como REF y como automorfismo φ. Para todos los casos se tendrá $\varphi = x$ y, además, se distinguirán las siguientes funciones REF:

(1) $REF(x, y) = 1 - |x - y|$

(2) $REF(x, y) = 1 - |x - y|^2$

(3) $REF(x, y) = 1 - |x - y|^{0.5}$

(4) $REF(x, y) = (1 - |x - y|)^2$

(5) $REF(x, y) = (1 - |x - y|)^{0.5}$

Debido a la formulación de este tarbajo, se sustituirá la aplicación anterior que actúa como función F en la definición 3.1.2 por la función de Dombi [13] definidas en 2.7.1. De esta forma, en este caso, los conjuntos difusos quedarán como explica la ecuación 3.1.

$$\mu_{Q_t}(q) = \begin{cases} F\left(w, \left(\frac{q}{L-1}, \frac{m_b(t)}{L-1}\right)\right) = \frac{1}{2}\left(1 + \left(1 - 2\frac{q}{L-1}\right)^w \cdot \left(1 - 2\frac{m_b(t)}{L-1}\right)^w\right) & \text{si } q \leq t, \\ F\left(w, \left(\frac{q}{L-1}, \frac{m_o(t)}{L-1}\right)\right) = \frac{1}{2}\left(1 + \left(1 - 2\frac{q}{L-1}\right)^w \cdot \left(1 - 2\frac{m_o(t)}{L-1}\right)^w\right) & \text{si } q > t. \end{cases} \tag{3.1}$$

Además, el último paso del bucle, en la línea 4 (Algoritmo 1=, se lleva a cabo la busqueda de la similitud frente al conjunto $\tilde{1}$. Para poder llevar a cabo este cálculo, tal y como se especifica en la definición 2.5.1, necesitamos utilizar una función REF, que será $REF_2 = 1 - \left|x - y\right|^2$, y la agregación M, la media aritmética ($M = \frac{i=1}{n} \sum_1^n x_i$). De esta forma, quedará como sigue:

$$SM(\tilde{1}, Q_t) = M_{q=0}^{L-1}(h(q) \cdot REF_2(1, \mu_{Q_t}(q))) \tag{3.2}$$

3.1.2. Algoritmo del área

Este algoritmo, que se presenta en [4] también, pretende hayar un nuevo umbral a través de la creación de una función REF, tal y como se explica en el teorema 2.3.5.

Algoritmo 2 Umbralización del área

Entrada: Una imagen Q en escala de grises donde sus píxeles estén entre 0 y $L-1$.
Salida: El umbral t a partir del cual se divide Q en objeto y fondo.

1: **para** t:=0 hasta $L-1$ **hacer**
2:

$$A(Q_t) = \sum_{q=0}^{t} h(q)\varphi_1^{-1}\left(1 - \left|\varphi_2\left(\frac{q}{L-1}\right) - \varphi_2\left(\frac{m_b(t)}{L-1}\right)\right|\right) + \sum_{q=t+1}^{L-1} h(q)\varphi_1^{-1}\left(1 - \left|\varphi_2\left(\frac{q}{L-1}\right) - \varphi_2\left(\frac{m_o(t)}{L-1}\right)\right|\right)$$

3: **fin para**
4: **devolver** $\{t \mid \max(A(Q_t))\}$

Esta es, realmente, una nueva versión del algoritmo de maximización de similitud anterior. Por esta razón, se prepara una formulación para conocer la relación de la similitud en función del área (A) que se ha calculado. Se intentará, por tanto, simplificarlo retirando la función de similitud SM.

Proposición 3.1.4. *Dada una agregación M, la media aritmética, y la función REF_2, una función de equivalencia restringida formada según la definición 2.3.4. Si se construyen dos*

conjuntos difusos Q_{t_1} y Q_{t_2} de acuerdo a definición 3.1.2 de la imagen Q. Si disponemos de una medida de similitud presentada en 2.5.1 podemos afirmar que:

$$SM(\tilde{1}, Q_{t_1}) \leq SM(\tilde{1}, Q_{t_1}) \quad \textit{si y sólo si} \quad A(Q_{t_1}) \leq A(Q_{t_1})$$

Prueba. Sabiendo que $REF(1, x) = x, \forall x \in [0, 1]$, entonces:

$$SM(\tilde{1}, Q_{t_1}) = \frac{1}{\sum_{q=0}^{L-1} h(q)} \sum_{q=0}^{L-1} \left(h(q) REF_2(1, \mu_{Q_{t_1}}(q)) \right) = \frac{1}{\sum_{q=0}^{L-1} h(q)} \sum_{q=0}^{L-1} \left(h(q) \mu_{Q_{t_1}}(q) \right) = \frac{A(Q_{t_1})}{\sum_{q=0}^{L-1} h(q)}.$$

Por esta razón, también se podrá decir que $SM(\tilde{1}, Q_{t_2}) = \frac{A(Q_{t_2})}{\sum_{q=0}^{L-1} h(q)}$ lo que prueba la proposición anterior. □

En este algoritmo, se utilizan 3 pares de automorfismos para darle forma al área.

(1) Se toma $\varphi_1(x) = \varphi_2(x) = x, \forall x \in [0, 1]$.

$$A(Q_t) = sum_{q=0}^{L-1} h(q) - \left(\sum_{q=0}^{t} \left(1 - \left| \frac{q}{L-1} - \frac{m_b(t)}{L-1} \right| \right) - \sum_{q=t+1}^{L-1} \left(1 - \left| \frac{q}{L-1} - \frac{m_o(t)}{L-1} \right| \right) \right)$$

(2) Se toma $\varphi_1(x) = x^d$ con $d \neq 0, \forall x \in [0, 1]$ y $\varphi_2(x) = x, \forall x \in [0, 1]$.

$$\sum_{q=0}^{t} h(q) \left(1 - \left| \frac{q}{L-1} - \frac{m_b(t)}{L-1} \right| \right) - \sum_{q=t+1}^{L-1} h(q) \left(1 - \left| \frac{q}{L-1} - \frac{m_o(t)}{L-1} \right| \right)$$

(3) Se toma $\varphi_1(x) = 1 - \sqrt{1-x}, \forall x \in [0, 1]$ y $\varphi_2(x) = x, \forall x \in [0, 1]$

$$\sum_{q=0}^{L-1} h(q) - \left(\sum_{q=0}^{L-1} h(q) \left(\frac{q}{L-1} - \frac{m_b(t)}{L-1} \right)^2 - \sum_{q=t+1}^{L-1} h(q) \left(\frac{q}{L-1} - \frac{m_o(t)}{L-1} \right)^2 \right)$$

En total, se dispondrá de 4 versiones diferentes. Se debe tener en cuenta que en la en el par (2) de automorfismos se dispondrá de $d = 0,5$ y $d = 2$.

3.1.3. Algoritmo de selección del umbral óptimo

Definición 3.1.5. Dada Q, una imagen en la escala de L niveles de gris, y t, un nivel de gris de forma que $0 \leq t \leq L-1$, se calcula su conjunto $H(Q_t)$ como

$$H(Q_t) = \{(q, \mu_{H(Q_t)}(q) | q \in \{0, 1, \dots, L-1\}\}$$

teniendo en cuenta que

$$\mu_{Q_t}(q) = \begin{cases} \dfrac{m_b(t)}{L-1} & \text{si } q \leq t, \\ \dfrac{m_o(t)}{L-1} & \text{si } q > t. \end{cases}$$

En el algoritmo, necesitaremos, después, calcular la similitud que existe entre el conjunto Q_t y el conjunto $H(Q_t)$. Para ello aplicamos la definición 2.5.1:

$$SM(Q_t, H(Q_t)) = M_{q=0}^{L-1}(h(q)REF(\mu_{Q_t}(q), \mu_{H(Q_t)}(q)))$$

Tendremos en cuenta que M seguirá siendo la media aritmética y $REF = 1 - \left|x - y\right|^2 \forall x \in [0, 1]$. Si a través de la uniformidad simplificamos la expresión anterior, obtendremos:

$$SM(Q_t, H(Q_t)) = \frac{1}{\sum_{q=0}^{L-1} h(q)} \sum_{q=0}^{L-1} \left(h(q)(1 - (\mu_{Q_t}(q) - \mu_H(Q_t)(q))^2)\right)$$

Algoritmo 3 Selección del umbral óptimo

Entrada: Una imagen Q en escala de grises donde sus píxeles estén entre 0 y $L-1$.
Salida: El umbral óptimo t^* a partir del cual se divide Q en objeto y fondo.

1: **para** t:=0 hasta $L-1$ **hacer**
2: Calcular los conjuntos Q_t como se describe en la definición 3.1.2.
3: Calcular los conjuntos H como se muestra en la definición 3.1.5.
4: Calcular la $SM(Q_t, H(Q_t))$.
5: **fin para**
6: **devolver** $\{t* \mid \max(SM)\}$

3.1.4. Otros algoritmos

Algoritmo de umbralización global

Entre los algoritmos que se encargan de segmentar la imagen con un único umbral, este es el más simple aunque en ocasiones podemos obtener resultados suficientemente buenos según que aplicación queramos darle.

En este sentido, para el buen funcionamiento del algoritmo es necesario que la imagen tenga una clara separación entre el objeto y el fondo, expresada con un valle en el histograma de la imagen. Se debe tener cuidado al elegir el parámetro ε ya que de él depende el número de iteraciones del bucle. Normalmente, conforme este crece, menor es el número de iteraciones necesarias para un resultado adecuado.

Algoritmo 4 Umbralización global.

Entrada: Una imagen Q en escala de grises donde sus píxeles estén entre 0 y $L-1$.
Salida: El umbral óptimo t a partir del cual se divide Q en objeto y fondo.

1: $t = 128$; {Se selecciona un umbral inicial cualquiera, aquí se tomará el valor medio de los posibles}
2: **repetir**
3: $tant = t$;
4: $G1 = \{(x,y)|q(x,y) > t0\}$
5: $G2 = \{(x,y)|q(x,y) \leq t0\}$
6: $m1 = \frac{1}{|G1|}\sum_{i \in G1} i$
7: $m2 = \frac{1}{|G2|}\sum_{i \in G2} i$
8: $t = \frac{m1+m2}{2}$
9: **hasta que** $|tant - t| < \varepsilon$ {El valor ε es un cierto error dispuesto por el programador.}
10: **devolver** t

Algoritmo de Otsu

El algoritmo de Otsu [24] basa su técnica de umbralización, también, en el histograma de la imagen que recibe. Para ello, trata las probabilidades que se le ortorgan a cada uno de los niveles de gris de los que dispone la imagen. Si tomamos una imagen Q que tenga sus intensidades entre $[1, 2, \dots, L]$, podremos expresar la probablidad de cada intensidad como

$$p_i = \frac{n_i}{N}, \quad p_i \geq 0 \quad \text{sabiendo que} \quad \sum_{i=0}^{L-1} p_i = 1$$

siendo n_i el número de píxeles de intensidad i y N el número total de píxeles en la imagen Q.

Ahora, supondremos que podemos clasificar los elementos de la imagen en dos clases, C_0 y C_1 que tendrán a los píxeles $[0, \dots, k]$ y $[k+1, \dots, L-1]$ respectivamente. Definirán su probabilidad como

$$\omega_0 = Pr(C_0) = \sum_{i=0}^{t} p_i = \omega_0(t) \qquad \text{y} \qquad \omega_1 = Pr(C_1) = \sum_{i=t+1}^{L-1} p_i = \omega_1(t)$$

y su media como

$$\mu_0 = \sum_{i=0}^{t} i\, Pr(i|C_0) = \sum_{i=0}^{t} \frac{ip_i}{\omega_0} = \frac{\mu(t)}{\omega(t)} \qquad \text{y} \qquad \mu_1 = \sum_{i=t+1}^{L-1} i\, Pr(i|C_1) = \sum_{i=t+1}^{L-1} \frac{ip_i}{\omega_1} = \frac{\mu_T - \mu(t)}{1-\omega(t)}$$

donde

$$\omega(k) \sum_{i=0}^{k} ip_i \qquad \text{y} \qquad \mu(k) = \sum_{i=0}^{k} ip_i$$

Por último, se puede definir

$$\mu_T = \mu(L-1) = \sum_{i=0}^{L-1} ip_i$$

y todas estas definiciones podemos asegurar que cumplirán que

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T \qquad \text{y} \qquad \omega_0 + \omega_1 = 1$$

Con la intención de evaluar lo adecuado o no que es cada umbral t, podemos definir la varianza de cada uno como

$$\sigma_B^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2$$

Con todo ello, para encontrar el umbral óptimo t^* sólo deberemos encontrar aquella varianza que maximice a las demás.

$$\sigma_B^2(t^*) = \max_{1\leq k<L} \left\{\sigma_B^2(t)\right\}$$

Al resumir todo lo anterior en lenguaje algorítmico se encuentra:

Algoritmo 5 Selección del umbral óptimo según Otsu.

Entrada: Una imagen Q en escala de grises donde sus píxeles estén entre 0 y $L-1$.
Salida: El umbral óptimo t^* a partir del cual se divide Q en objeto y fondo.

1: **para** t:=0 hasta $L-1$ **hacer**
2: $\omega_0(\mathrm{t}) = \sum_{i=1}^{t} p_i$
3: $\mu_0(\mathrm{t}) = \sum_{i=1}^{t} \frac{ip_i}{\omega_0}$
4: $\omega_1(\mathrm{t}) = \sum_{i=t+1}^{L} p_i$
5: $\mu_1(\mathrm{t}) = \sum_{i=t+1}^{L} \frac{ip_i}{\omega_1}$
6: $\sigma_B^2(t) = \omega_0\omega_1(\mu_1 - \mu_0)^2$
7: **fin para**
8: **devolver** $\{t^* \mid max_t(\sigma_B^2)\}$

Algoritmo de maximización de la entropía de Renyi

Este algoritmo, enunciado en [27], busca el umbral óptimo de la imagen a través de la maximización de la entropía que se produce en la imagen definida por Renyi. Al igual que en el algoritmo de Otsu se empieza definiendo dos clases, la correspondiente al fondo y la del objeto. Para cada una de ellas se define su probabilidad

$$p(C_0) = \sum_{i=0}^{t} p_i \qquad \text{y} \qquad p(C_1) = \sum_{i=t+1}^{L-1} p_i$$

donde con ambas probabilidades se cumple que $p(C_0) + p(C_1) = 1$.

A continuación se define la entropía para ambas clases sabiendo que $\alpha \neq 1$.

$$H^{\alpha}_{C_0}(t) = \frac{1}{1-\alpha}\, ln \sum_{i=0}^{t} \left(\frac{p_i}{p(C_0)}\right)^{\alpha} \qquad \text{y} \qquad H^{\alpha}_{C_1}(t) = \frac{1}{1-\alpha}\, ln \sum_{i=t}^{L-1} \left(\frac{p_i}{p(C_1)}\right)^{\alpha}$$

Llegado este punto, se puede encontrar el umbral óptimo $t^*(\alpha)$ aquel que maximice la suma de las entropías de las clases:

$$t^*(\alpha) = \underset{0 \leq t < L}{max} \left\{ H^{\alpha}_{C_0}(t) + H^{\alpha}_{C_1}(t) \right\}.$$

Por medio de experimentación se ha comprobado que el umbral anterior no es igual para todos los valores de α. Este hecho se formaliza de la siguiente forma:

$$t^*(\alpha) = \begin{cases} t_1^* \text{ si } 0 < \alpha < 1, \\ t_2^* \text{ si } \alpha \to 1, \\ t_3^* \text{ si } 1 < \alpha < \infty. \end{cases}$$

Por esta razón, se hace necesario emcontrar un umbral t que no sea dependiente de α. Para ello se deben calcular un umbral para cada una de las 3 situaciones enumeradas y ordenarlos de mayor a menor, algo que simbolizaremos como $t_{(1)} \leq t_{(2)} \leq t_{(3)}$. De estas forma, definiremos el umbral óptimo de la imagen como

$$t_c^* = t_{(1)} \left(p(t_{(1)}) + \frac{1}{4}\omega\beta_1 \right) + \frac{1}{4} t_{(2)} \omega\beta_2 + t_{(3)} \left(1 - p(t_{(3)}) + \frac{1}{4}\omega\beta_3 \right)$$

en donde tendremos en cuenta que $p(t) = \sum_{i=1}^{t}(p_i)$ y $\omega = p(t_{(3)}) - p(t_{(1)})$ además del vector $(\beta_1, \beta_2, \beta_3)$ que se toma así:

$$(\beta_1, \beta_2, \beta_3) = \begin{cases} (1,2,1) & \text{si} \quad \left|t_{(1)} - t_{(2)}\right| \leq 5 \text{ y } \left|t_{(2)} - t_{(3)}\right| \leq 5, \\ (1,2,1) & \text{si} \quad \left|t_{(1)} - t_{(2)}\right| > 5 \text{ y } \left|t_{(2)} - t_{(3)}\right| > 5, \\ (0,1,3) & \text{si} \quad \left|t_{(1)} - t_{(2)}\right| \leq 5 \text{ y } \left|t_{(2)} - t_{(3)}\right| > 5, \\ (3,1,0) & \text{si} \quad \left|t_{(1)} - t_{(2)}\right| > 5 \text{ y } \left|t_{(2)} - t_{(3)}\right| \leq 5, \end{cases}$$

En definitiva, el umbral t_c^* puede verse como una media ponderada de los valores de t_1^*, t_2^*, t_3^* por lo que se puede afirmar

$$min\{t_1^*, t_2^*, t_3^*\} \leq t_c^* \leq max\{t_1^*, t_2^*, t_3^*\} = t_{(1)} \leq t_c^* \leq t_{(3)}$$

esta agregación consigue evitar los malos resultados que podrían darse si no se utilizara un parámetro α correcto al igual que integra todas las características de la imagen.

Algoritmo 6 Selección del umbral óptimo maximizando la entropía de Renyi.

Entrada: Una imagen Q en escala de grises donde sus píxeles estén entre 0 y $L-1$.
Salida: El umbral óptimo $t*$ a partir del cual se divide Q en objeto y fondo.

$\alpha = [0.3, 0.99999, 10]$
{Para facilitar la implementación, se tendrá en cuenta únicamente un alfa de cada uno de los casos, cuando se estabilizan cada uno de los casos}
para $\alpha_i \in \alpha$ **hacer**
  $H_T = 0$
  para $t' = 0$ hasta $L-1$ **hacer**
    $p_{C_0} = \sum_{i=0}^{t'} p_i$
    $p_{C_1} = \sum_{i=t'+1}^{L-1} p_i$
    $H_{C_0}^{\alpha}(t') = \frac{1}{1-\alpha} ln \sum_{i=0}^{t'} \left(\frac{p_i}{p(C_0)}\right)^{\alpha}$
    $H_{C_1}^{\alpha}(t') = \frac{1}{1-\alpha} ln \sum_{i=t'}^{L-1} \left(\frac{p_i}{p(C_1)}\right)^{\alpha}$
    $H_T(t') = H_T(t') + H_{C_0} + H_{C_1}$
  fin para
  $t_{mejor}(i) = max_{0 \le j < L} \left\{H_T(j)\right\}$
fin para
t = ordenar(t_{mejor});
$\omega = \sum_{i=0}^{t_{(3)}} p_i - \sum_{i=0}^{t_{(1)}} p_i$
si $\left|t_{(1)} - t_{(2)}\right| \le 5$ y $\left|t_{(2)} - t_{(3)}\right| \le 5$ **entonces**
  $\beta = [1, 2, 1]$;
si no $\left|t_{(1)} - t_{(2)}\right| > 5$ y $\left|t_{(2)} - t_{(3)}\right| > 5$ **entonces**
  $\beta = [1, 2, 1]$;
si no $\left|t_{(1)} - t_{(2)}\right| \le 5$ y $\left|t_{(2)} - t_{(3)}\right| > 5$ **entonces**
  $\beta = [0, 1, 3]$;
si no $\left|t_{(1)} - t_{(2)}\right| > 5$ y $\left|t_{(2)} - t_{(3)}\right| \le 5$ **entonces**
  $\beta = [3, 1, 0]$;
fin si
$t_c^* = t_{(1)} \left(p(t_{(1)}) + \frac{1}{4}\omega\beta_1\right) + \frac{1}{4} t_{(2)} \omega\beta_2 + t_{(3)} \left(1 - p(t_{(3)}) + \frac{1}{4}\omega\beta_3\right)$
devolver t_c^*

Algoritmo *K-means*

Lo primero que se debe destacar del algoritmo de *K-means* es que, a diferencia de los algoritmos anteriores que segmentaban las imágenes con tecnicas de umbralización, este lo hace por medio de la agrupación de píxeles en regiones. Esto hace que este algoritmo sea muy difente del resto y que no base sus fundamentos teóricos en el manejo del histograma. Incluye, también, la posibilidad directa de hacer segmentación con más de un objeto ya que la K indica el número de regiones en las que se dividirá la imagen.

Para poder obtener cada una de las regiones, deberemos encontrar los píxeles (x, y) que pertenecen a cada una. Este se consigue con la minimización de la función de coste que dados los centros $\mu_1, \dots, \mu_n$ se define como

$$J = \sum_{i=1}^{N \cdot M} \left\| q(x,y)_i - \mu_{c_i} \right\|^2 .$$

En vista de lo anteior, se definirá la pertenencia de un pixel a un centro c_k como

$$\min_{k=1,\dots,K} \left\| q(x,y) - \mu(k) \right\|^2 .$$

Para poder actualizar los centros deberemos hacer la minimización de la función de coste que, evidentemente, se obtendrá en aquel lugar donde la derivada se haga 0.

$$\frac{d}{d\mu_k} = \sum_{i \in cluster_k} \left\| q(x,y)_i - \mu_k \right\|^2 = -2 \sum_{i \in cluster_k} (q(x,y) - \mu_k = 0$$

por lo que, cada uno de los centros se encontrará como

$$\mu_k = \frac{1}{|cluster_k|} \sum_{i \in cluster_k} q(x,y)_i$$

Se debe tener en cuenta que al obtener una región tenemos que buscar un método para poder representarla y operar con ella. Debido a la gran cantidad de datos que dispondríamos de cada una se estima que la mejor forma de identificarla es con su centro μ_R.

Como se podrá observar, para el cálculo de los centros se necesita conocer la pertenencia de cada uno de los píxeles y para obtener la pertenencia es necesario disponer de los centros. Esta paradoja hace que nos sea imposible comenzar el proceso si no es asignando unos primeros datos al azar y esperando a que el algoritmo converja conforme el número de iteraciones crezca.

Por último, debemos señalar que para que el bucle tenga una condición de parada esta será puesta en función del error, ya que como se ha mostrado este irá disminuyendo hasta estabilizarse. Así mismo, al terminar de procesar la imagen, deberemos devolverla ya que no de dispone de un umbral t que divida la misma.

Algoritmo 7 Segmentación por medio de *k-means*.

Entrada: Una imagen Q en escala de grises donde sus píxeles estén entre 0 y $L-1$ y el número de clases K que se desean obtener.

Salida: La imagen umbralizada en dos regiones con dos tonos de gris diferenes, *imgSegmentada*.

```
1: N = filas(Q);
2: M = columnas(Q);
3: μ = aleatorios(K);
4: J = 0; {Coste}
5: repetir
6:    Jant = J;
7:    para i = 1 hasta N hacer
8:       para j = 1 hasta M hacer
9:          J = J + min_{k=1,...,K} ||q(x,y) − μ(k)||^2 ;
10:      fin para
11:   fin para
12:   para k = 1 hasta K hacer
13:      μ(j) = 1/|cluster_k| Σ_{i∈cluster_k} q(x,y)_i
14:   fin para
15: hasta que J ≠ Jant
16: para j := 1 hasta K hacer
17:   imgSegmentada = imgSegmentada + Σ_{i∈cluster_j} μ(i)
18: fin para
19: devolver imgSegmentada;
```

3.2. Experimentos y resultados con funciones de equivalencia de Dombi

3.2.1. Experimento 1: sustitución de la función REF por la función de Dombi en el algoritmo de maximización de la similitud

Explicación del experimento

En este primer experimento pretendemos conocer cómo se pueden adaptar las funciones de Dombi que hemos descrito en 2.7.1 para la sustitución de las funciones REF en la creación de los conjuntos difusos. Se han llevado a cabo experimentos con casi 30 imágenes. Entre todos estos experimentos, a continuación se presentan los más reelevantes. Para ello, se tomarán las imágenes que se muestran en la figura 3.1. Se muestran junto con sus histogramas así como las imágenes ya umbralizadas por parte de un experto (si existen).

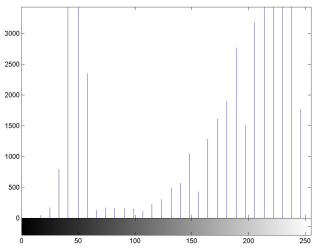
En primer lugar, se procesarán las imágenes tomando el parámetro $w = 1$ tanto para x como para y. Después, y a la vista de unos resultados variables, se toma la determinación de ampliar el rango de pesos por lo que se prepara una prueba para los valores 0,1; 0,5; 0,75; 1; 1,25; 1,5; 2 y 5. Por último, para poder conocer la efectividad de todos estos resultados, son comparados contra el algoritmo de maximización de la similitud original tomando como equivalencia restringida todas las funciones que se han enumerado anteriormente en 2.5.2, dándole especial importancia a la $REF(x,y) = 1 - |x - y|$ así como con todos los demás algoritmos explicados en el apartado 3.1.4 de la sección anterior.

Para poder conocer de forma empírica la diferencia entre dos imagenes segmentadas, se calcula para todas ellas el error cuadrático medio. Se puede expresar el error como

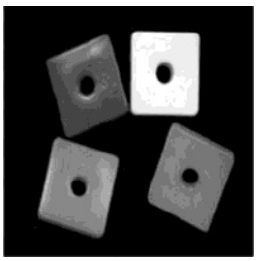
$$ECM(Q,Q') = \frac{\sum_{x=1}^{N}\sum_{y=1}^{M}\left(q(x,y) - q'(x,y)\right)^2}{N \cdot M}.$$

Para poder conocer cómo actua el algoritmo frente al ruido, se han preparado imágenes con ruido gausiano con media $\mu = 0$ y varianza $\sigma^2 = 0,01$. También, se han preparado con ruido impulsivo, de 'de sal y pimienta' con densidad de probabilidad del ruido $d = 0,05$ y $d = 0,2$. Se presentan las imágenes ya tratadas en la figura 3.2. Por otra parte, se prueba a ver cómo resiste el algoritmo con imágenes de bajo y alto contraste.

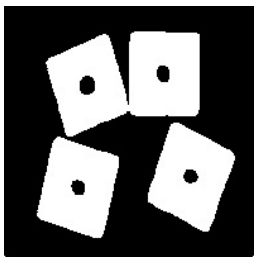
(a) 'Silla'



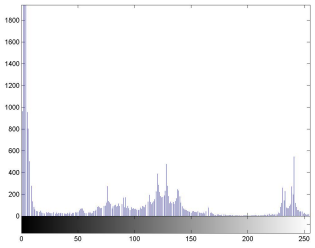
(b) Histograma



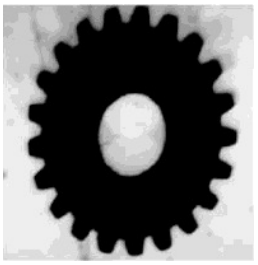
(c) 'Bloques'



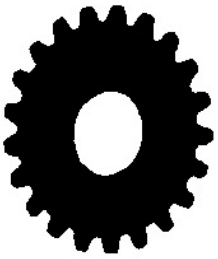
(d) Umbralización del experto.



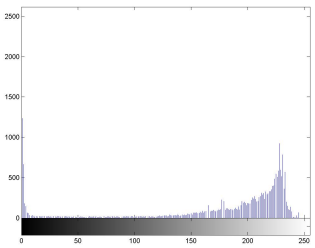
(e) Histrograma



(f) 'Engranaje'

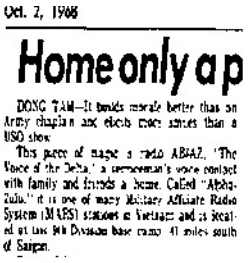


(g) Umbralización del experto.

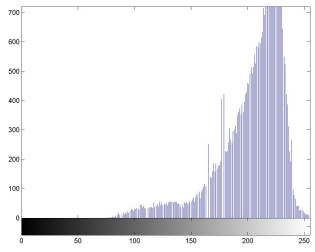


(h) Histrograma

(i) 'Letras'



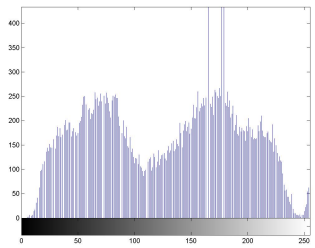
(j) Umbralización del experto.



(k) Histrograma

(l) 'Sombra'

(m) Umbralización del experto.



(n) Histrograma

Figura 3.1.: Imágenes utilizadas en el experimento 1 y siguientes.

Resultados

En la tabla 3.1 se pueden apreciar los resultados para la umbralización de cada una de las imágenes y diferentes w. Se dispone, también de la tabla 3.2 donde se expresan los umbrales para otros algoritmos. En la comparación de ambas tablas se obtiene un resultado claro, el parámetro w influye de forma determinante en la segmentación de las imágenes. Así, por ejemplo, en el caso de la 'silla' $w = 1$ es una opción totalmente acertada mientras que para para las 'letras' habría que doblar el valor para obtener una umbralización igual de buena en comparación con los otros métodos.

w	Silla	Bloques	Engranaje	Letras	Sombra
0,1	218	255	250	142	200
0,5	226	255	250	39	230
0,75	95	119	115	103	111
1	127	123	137	160	125
1,25	70	97	96	80	91
1,5	45	79	0	39	64
2	144	76	138	197	96
5	218	31	59	216	219

Tabla 3.1.: Umbrales para cada imagen con la función de Dombi y diferentes w.

	Silla	Bloques	Engranaje	Letras	Sombra
Alg. 1 con $REF_1 = 1 - \lvert x - y \rvert$	127	79	104	187	123
Alg. 1 con $REF_1 = 1 - \lvert x - y \rvert^2$	127	97	140	179	126
Alg. 1 con $REF_1 = 1 - \lvert x - y \rvert^{0.5}$	119	47	84	200	121
Alg. 1 con $REF_1 = (1 - \lvert x - y \rvert)^2$	127	70	105	190	123
Alg. 1 con $REF_1 = (1 - \lvert x - y \rvert)^{0.5}$	127	82	104	186	124
U. Global	130	79	105	187	124
U. de Otsu	123	79	104	187	123
Máx. la entropía de Renyi	170	32	131	160	135

Tabla 3.2.: Umbrales para cada imagen con otras versiones de algoritmos.

Por otra parte, cabe destacar que este resultado se ha obtenido en un tiempo de computación de alrededor de 0,0025 segundos, tanto para la versión con función de Dombi como para la de REF. Con la intención de que el lector pueda juzgar adecuadamente estos resutlados, se presentan los resultados, en forma gráfica, para aquellos valores de w más reelevantes de forma gráfica en la tabla 3.3. Se facilita la posibilidad de poder compararlo directamente con el algoritmo original. Además, se dispone en el apéndice, en la tabla B.2, la umbralización de las imágenes con otros algoritmos para que se pueda llevar a cabo también la comparación.

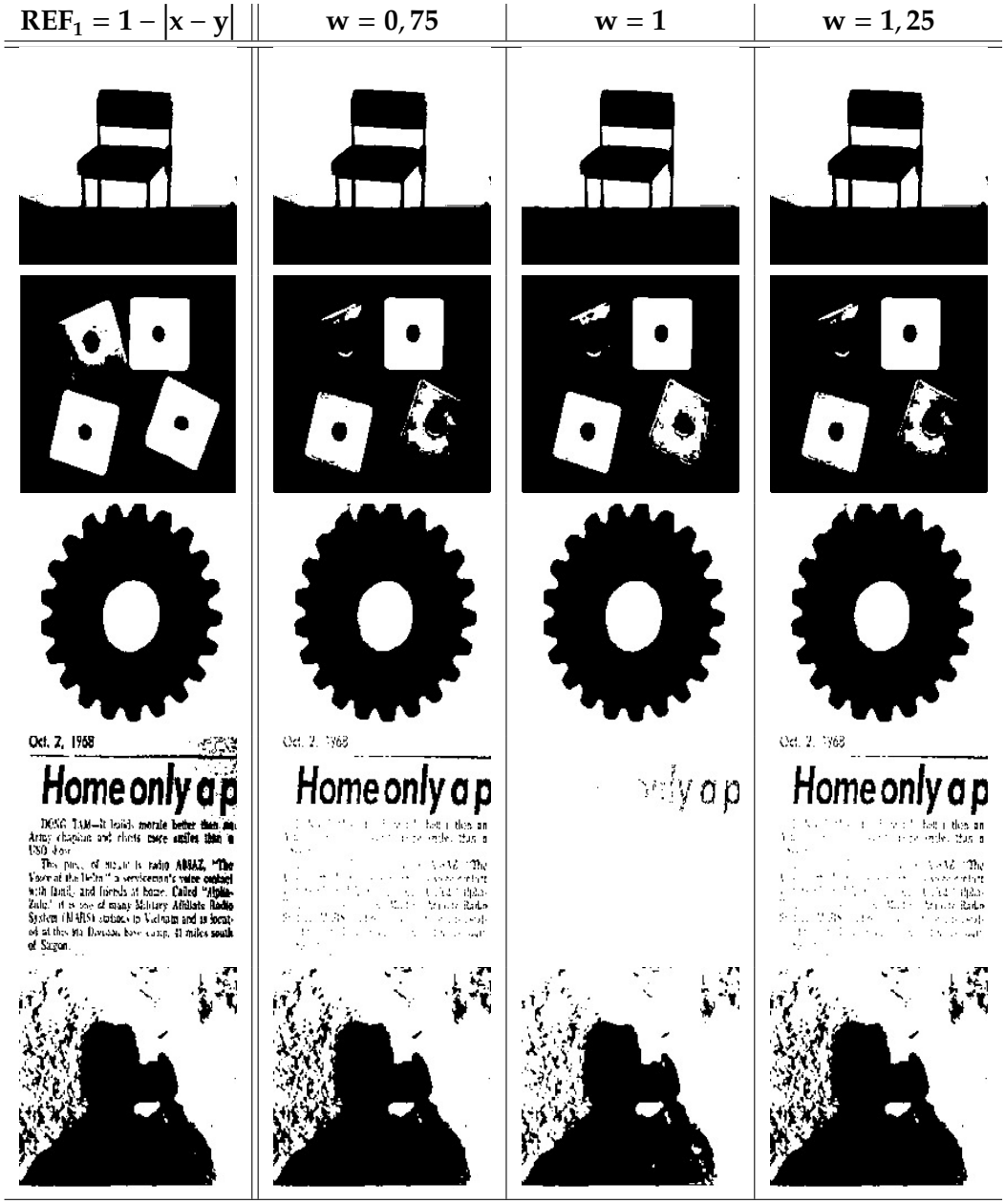


Tabla 3.3.: Resultado de las segmentaciones para el algoritmo con REF y Dombi con varios $w = \{0,75; 1; 1,25\}$.

Para poder comprobar y comparar formalmente todos los resultados, se calcula el error cuadrático medio para las imágenes. En concreto, se hace con todas las versiones obtenidas con otros los algoritmos presentados sin obtener ningún resultado reelevante. Se presentan en la tabla 3.4, la comparación que se hace entre las segmentaciones que se obtienen con varias w y la imagen umbralizada de forma exacta por un experto, resultados que se hacen más interesantes que los anteriores. De nuevo, de esta tabla se desprende de que en función de la imagen que se tome, el parámetro w para obtener una mejor umbralización.

w	Bloques	Engranaje	Letras	Sombra
Alg. 1 con 0,75	43,3750	33,0080	33,3617	59,2420
Alg. 1 con 1	49,8760	36,3914	29,6418	52,0672
Alg. 1 con 1,25	50,0241	32,0153	31,0001	55,0322

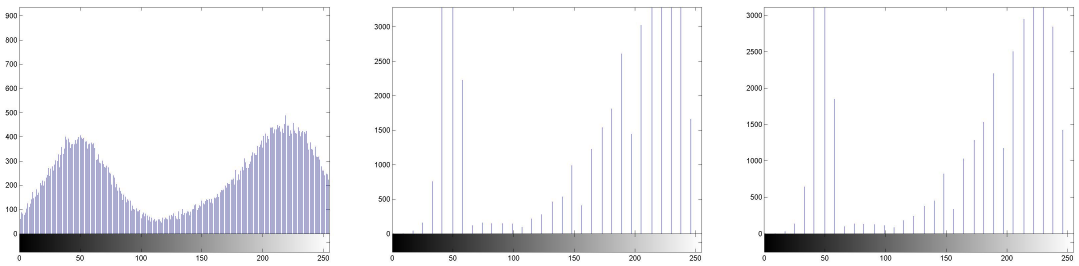
Tabla 3.4.: Errores para las imágenes con ruido con otras versiones de algoritmos.

Con la intención de poder llevar a cabo experimentos sobre imágenes con ruido, se crean estas de forma sintética, tal y como se ha presentando en la sección 2.1.4. Se pueden ver los resultados sobre la imagen 'silla', y sus histogramas, en la figura 3.2.

(a) Ruido gausiano

(b) Ruido 'sal y pimienta' con $p = 0,05$

(c) Ruido 'sal y pimienta' con $p = 0,2$



(d) Histograma de ruido gausiano

(e) Histograma de ruido 'sal y pimienta' con $p = 0,05$

(f) Histograma de ruido 'sal y pimienta' con $p = 0,2$

Figura 3.2.: Imágenes con ruido utilizadas en el experimento 1 y siguientes.

Los umbrales que se han obtenido se expresan en la tabla 3.5. Comparándola de nuevo con otros algoritmos (tablas 3.6), se puede apreciar que se pueden obtener resultados

similiares también e, igualmente, que el parámetro w es decisivo para diferentes imágenes. Es curioso comprobar el hecho de que con la misma w se obtienen resultados similares en comparación con los umbrales de la versión original con REF.

	R. gausiano	**R. impulsivo 0.05**	**R. impulsivo 0.2**
0,1	219	226	226
0,5	234	226	242
0,75	99	95	95
1	128	127	127
1,25	74	70	62
1,5	35	45	37
2	147	136	127
5	226	218	226

Tabla 3.5.: Umbrales para las imágenes con ruido con la función de Dombi y diferentes valores de w.

	R. gausiano	**R. impulsivo 0.05**	**R. impulsivo 0.2**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	132	127	127
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	131	127	127
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	131	136	152
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	132	127	127
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	131	127	127
U. Global	132	130	128
U. de Otsu	132	123	123
Máx. la entropía de Renyi	154	170	170

Tabla 3.6.: Umbrales para las imágenes con ruido con otras versiones de algoritmos.

En cuanto al tiempo de cómputo, de nuevo vuelven a estar todos los valores entorno a 0,0025 segundos. Esto hace ver que el método presentado sigue siendo igual de eficiente que la versión original. Así mismo, se muestran los resultados gráficamente en la tabla 3.7.

En relación con los errores con imagenes umbralizadas con otros algoritmos, no se obtiene ningún resultado concluyente ya que en ningún momento el error pasa de 0,1.

Además, se han hecho experimentos con imágenes con mucho y poco contraste. En este caso, la solución tenía el error de nuevo en un rango de diferencia de una décima, aunque permite llegar a opciones con mucho más bajo contraste que el algoritmo de la maximización de la entropía de Renyi, que es el que mejor lo hace entre los otros algoritmos presentados. En definitiva, con el algoritmo de maximización de la similitud se pueden llegar a segmentar imágenes con intervalos de menos de 15 niveles de gris. Se

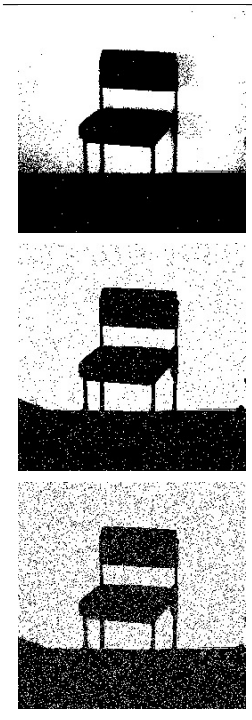
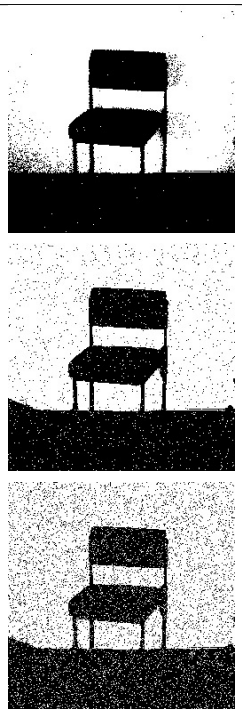
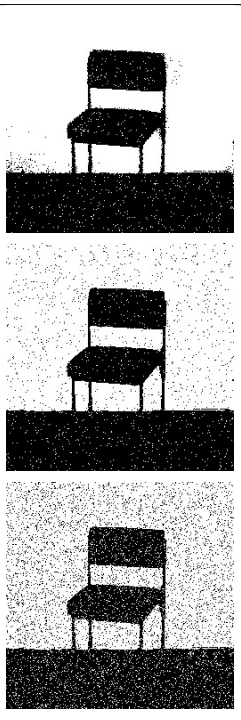
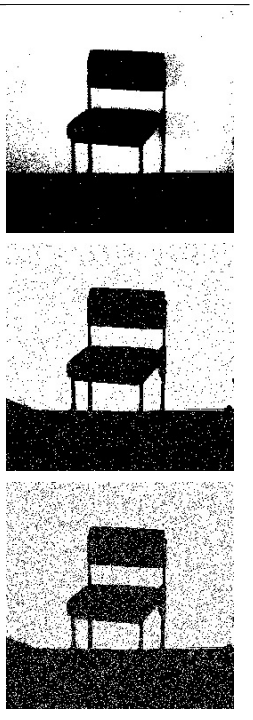
$\mathbf{REF_1 = 1 - \lvert x - y \rvert}$	$\mathbf{w = 0,75}$	$\mathbf{w = 1}$	$\mathbf{w = 1,25}$
			

Tabla 3.7.: Umbrales para las imágenes con ruido con otras versiones de algoritmos.

encuentra, también, el hechod de que en el caso de crear dos imágenes con alto y bajo contraste que sean "contrarias", la segmentación que se obtiene es la misma.

3.2.2. Experimento 2: busqueda del mejor umbral a través de funciones penalti

Explicación del experimento

En vista del experimento anteior donde había momentos en los que la función de dombi no actuaba de forma correcta, se propone una nueva forma para poder elegir entre la función de Dombi o volver al método con la función REF. En concreto, se intentará dilucidar esta situación por medio de una función penalti. El proceso de la obtención del conjunto difuso cambiará, ya que no será una única función y tendrá el siguiente orden:

(1) Creación de todas las pertenencias con las 5 funciones REF presentadas y la función de Dombi con $w = 1$.

(2) Agregación de todas las pertenencias anteriores con varias funciones, a saber:

   a) Media aritmética.

   b) Media geométrica.

   c) Máximo.

   d) Mínimo.

   e) Integral Choquet.

   f) OWA de 'al menos la mitad'.

(3) Con todos estos resultados se actua con la función penalti para cada uno de los elementos $a_1, \dots a_n$ agregados y las pertenencias obtenidas en (1) $r = (r_1, \dots, r_m)$, obtendremos $p(a_i, r) = \sum_{j=1}^{m} |a_i - r_j|$.

(4) Cogeremos como pertenencia aquel resultado de la agregación, a_i, que haga que el resultado de la función penalti sea máximo.

En vista de que los resultados de la idea presentada anteriormente no presentaba a penas diferencia con el experimento 1. Por esta razón, en una segunda versión, se añadirá también la función de agregación siguiente con la intención de crear más *outlayers* que creen más posibles umbrales.

$$M_{zadeh}(r_1, \dots, r_m) = max\left(0, \sum r - (n-1)\right) \cdot \sqrt[n]{\prod_{i=1}^{n} r_i}$$

Esta segunda versión son los resultados que se presentan a continuación.

Resultados

En la tabla 3.8 se recogen los umbrales t creados a través de la función penalti. De nuevo, en comparación con los presentados en la tabla 3.2 se puede observar que siempre son valores que se encuentran en el intervalo que pueden definir. Esto hace suponer que las imágenes tiene una umbralización correcta, algo que se puede comprobar con las imágenes del cuadro 3.9. En relación al error (tabla 3.8)sorprende la diferencia que existe frente

Además, en la tabla 3.11 se pueden observar los umbrales obtenidos para las imágenes con ruido. De forma similar a las imágenes normales, el umbral vuelve a estar en el

Silla	Bloques	Engranaje	Letras	Sombra
127	68	92	196	123

Tabla 3.8.: Umbrales para cada imagen con el algoritmo 1 calculando la función de pertenencia a través de una función penalti.

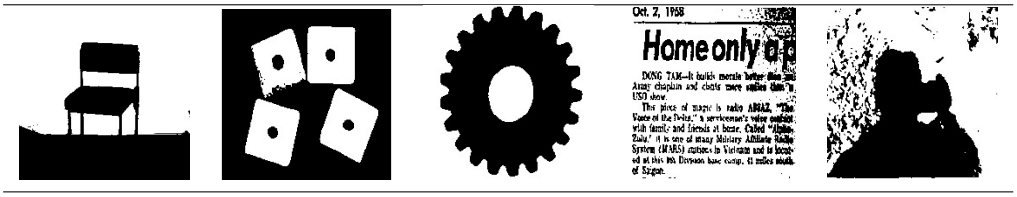


Tabla 3.9.: Resultado para el nuevo algoritmo a través de penalti con todas las funciones propuestas.

intervalo que se crea con los umbrales originales del experimento anterior. Tal y como se puede ver en la tabla 3.12 el resultado para las imágenes con ruido gausiano y con ruido 'sal y pimienta' con $p = 0,05$ mientras que subiendo la probabilidad para el ruido, el umbral comienza a ser bastante inadecuado. Como es lógico, el ruido impulsivo no se consigue resolver aunque tampoco es así en el caso del gausiano, que segmenta de forma bastante adecuada aun no apreciandose un mínimo claro.

3.2.3. Experimento 3: sustitución de la función REF por la función de Dombi en el algoritmo para la obtenición del umbral óptimo

Explicación del experimento

En este experimento se ha tomado el algoritmo de obtención del umbal óptimo maximizando la similitud (algoritmo 3A) para comprobar el efecto de la función de Dombi en él. Por esta razón, y en vista de los resultados del experimento 1, se plantean dos variantes. En primer lugar, se incorpora la función de dombi a las 5 funciones REF que ya estaban para poder ver qué umbral se obtiene, intentando escoger el mejor (algoritmo 3B). Después, se eligen los diferentes valores de w que se han utilizado antes (algoritmo 3C). De esta forma, se intenta buscar cual de los umbrales calculados anteriormente de

Bloques	Engranaje	Letras	Sombra
90,3727	124,9112	140,5241	194,3489

Tabla 3.10.: Errores en comparación con las imágenes segmentadas por el experto.

R. gausiano	R. impulsivo 0.05	R. impulsivo 0.2
132	127	152

Tabla 3.11.: Umbrales para cada imagen ruido a través del algoritmo 1 calculando la función de pertenencia a través de una función penalti.

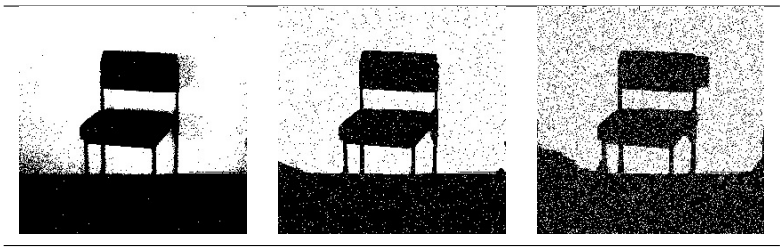


Tabla 3.12.: Resultado para el nuevo algoritmo a través de penalti con todas las funciones propuestas para imágenes con ruido.

forma individual es el mejor.

Resultados

Hecho todo el cómputo, en la tabla 3.13 se muestran los umbrales obtenidos tanto con el algoritmo original como por parte de las dos versiones que se han propuesto. Por un lado, es muy visible el hecho de que para la mayor parte de las imágenes la versión C no tiene utilidad, ya que elige umbrales totalmente fuera de la lógica. Por otra parte, se ve que la versión B es muy similar a la versión original. Esto hace pensar que esta versión podría hacer que las funciones de Dombi fueran interesantes para algunos casos. Los resultados gráficos se presentan en la tabla 3.14

	Silla	Bloques	Engranaje	Letras	Sombra
Alg. 3A	115	80	88	199	121
Alg. 3B	127	123	84	200	125
Alg. 3C	218	254	250	142	219

Tabla 3.13.: Umbrales para cada imagen con el algoritmo 3 en todas sus nuevas versiones.

En cuanto al tiempo de cómputo, aunque crece con respecto a los algoritmos de los experimentos anteriores, este sigue estando en la misma media del algoritmo original, siempre que se tome como versión original la que definen las 5 REF del algoritmo 1. Todos los resultados gráficos son resumidos en la tabla 3.14.

Para poder conocer cómo de buenos son los resultados, se comparan las imágenes obteniendo el error cuadrático medio (tabla3.15). En esta se puede apreciar cómo se

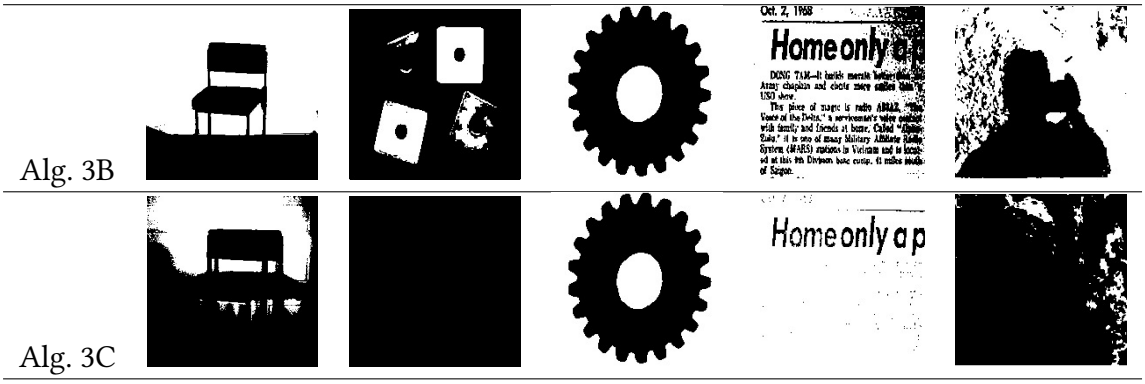


Tabla 3.14.: Resultado gráfico con el algoritmo 3 con las nuevas propuestas.

mantiene un error bastante similar con todos los algoritmos.

	Silla	Bloques	Engranaje	Sombra
Alg. 3A (Original)	72,8822	93,8135	126,5836	124,7987
Umb. Global	71,7070	81,0453	124,7637	122,4343
K-means	71,7070	81,0453	124,9112	123,2863
Método de Otsu	71,7070	81,0453	124,9112	123,2863
Máx. Entropía Renyi	56,9290	96,4543	121,5597	112,4747

Tabla 3.15.: Erorres de las imágenes obtenidas con otros algoritmos para cada imagen con el algoritmo 3 en todas sus versiones.

Se comprueba que las tendencias anteriores son también seguidas por aquellas imágenes que disponen de ruido. Es decir, la versión que se propone con la función de dombi con diferentes valores de w. También, los resultados con la versión B son más bajos con respecto a las versión original, algo que como se ha visto es bastante común desde el momento en el que se introducen las funciones de Dombi.

	R. gausiano	R. impulsivo 0.05	R. impulsivo 0.2
Alg. 3A	131	136	144
Alg. 3B	128	127	127
Alg. 3C	226	218	226

Tabla 3.16.: Umbrales para cada imagen ruidosa con el algoritmo 3 en todas sus versiones.

3.2.4. Repetición de los experimentos con la corrección oportuna.

Vistos los resultados de los experimentos anteriores, y en especial el que se refiere al experimento 3, se plantea el porqué se obtiene un resultado tan malo. Este hecho es

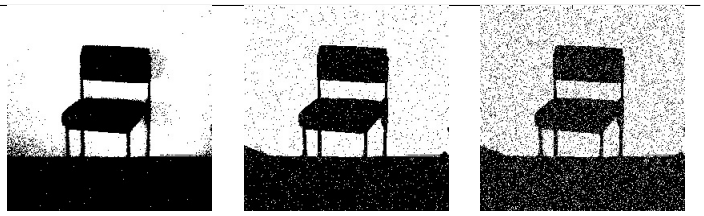
Alg. 3B	
Alg. 3C	

Tabla 3.17.: Representación gráfica del resultado .

	R. gausiano	R. impulsivo 0.05	R. impulsivo 0.2
Alg. 3 Original	95,1075	66,4737	74,2907
Umb. Global	94,8740	68,2635	77,5124
K-means	95,1074	68,2635	77,5124
Método de Otsu	94,8740	68,2635	77,5124
Máx. Entropía Renyi	85,9364	54,1976	65,7772

Tabla 3.18.: Errores para cada imagen ruidosa con otros algoritmos tomando como referencia la versión 3B.

extraño sobre todo por el hecho de que a través de los diferentes valores de w, el algoritmo 1 con las funciones de dombi siempre tiene buenos resultados. Se comprueba el vector de similitudes que se obtiene para dilucidar qué umbral escoger en el algoritmo 3, lo que muestra que la similitud mayor no se encuentra en el mejor umbral.

Reestudiando la situación, se encuentra que en las propiedades de las funciones que propone Dombi (Lema 2.7.2) se ha omitido que una función de equivalencia e cumple $e(x,x) = 1$. En particular, las funciones REF son un ejemplo de esta situación. La razón por la que no se puede cumplir esta propiedad es porque existe otra que dice $e(x, n(x)) = 0$. Como es evidente, el intentar mantener ambas a la vez es una contradicción. Por esta razón, el autor dice que sus funciones cumplirán en parte ambas en función de un parámetro. Esto hace que para dos elementos iguales no tengamos como resultado 1.

Vista la situación y sabiendo que la construcción que se utiliza para los conjuntos difusos hace necesario que la condición anterior se cumpla de forma plena, se intenta reescribir la función de pertenencia a los conjuntos difusos. De esta forma, se aplica a todos los experimentos que se han mostrado anteriormente esta nueva función (ecuación 3.3)

$$\mu_{Q_t}(q) = \begin{cases} min\left(1, \frac{\frac{1}{2}\left(1+\left(1-\frac{2q}{L-1}\right)\left(1-\frac{2m_b}{L-1}\right)\right)}{\frac{1}{2}\left(1+\left(1-\frac{2m_b}{L-1}\right)\left(1-\frac{2m_b}{L-1}\right)\right)}\right) & \text{si} \quad q \leq t \\ min\left(1, \frac{\frac{1}{2}\left(1+\left(1-\frac{2q}{L-1}\right)\left(1-\frac{2m_o}{L-1}\right)\right)}{\frac{1}{2}\left(1+\left(1-\frac{2m_o}{L-1}\right)\left(1-\frac{2m_o}{L-1}\right)\right)}\right) & \text{si} \quad q > t \end{cases} \tag{3.3}$$

Resultados

La opción que se ha propuesto anteriormente ha tenido resultados muy malos, de hecho, mucho peores que las presentadas en las secciones anteirores. Por esa razón y únicamente a modo de ejemplo, en las tablas 3.19 y 3.20 se pueden ver algunos resultados obtenidos. Este algoritmo, es por tanto, desechado de forma completa. Se acaba concluyendo que las funciones de Dombi no pueden sustituir directamente a funciones REF tal y como el autor sugería. Por esta razón, la umbralización con los métodos anteriores presentará resultados hetereogéneos lo que hace que no sea un método útil para su utilización habitual.

		Silla	Bloques	Engranaje	Letras	Sombra
$w = 0,1$	**Umbral (t)**	234	8	0	157	250
	Tiempo (s)	0,002	0,675	0,686	0,689	0,688
$w = 0,5$	**Umbral (t)**	226	8	111	207	254
	Tiempo (s)	0,002	0,649	1,093	1,233	1,13
$w = 0,75$	**Umbral (t)**	226	8	122	182	238
	Tiempo (s)	0,002	0,677	0,686	0,688	0,679
$w = 1$	**Umbral (t)**	226	8	90	191	246
	Tiempo (s)	0,002	0,63	0,642	0,646	0,642
$w = 1,25$	**Umbral (t)**	226	8	250	203	252
	Tiempo (s)	0,002	0,682	0,684	0,688	0,686
$w = 1,5$	**Umbral (t)**	242	6	250	213	254
	Tiempo (s)	0,002	0,666	1,106	1,246	1,138
$w = 2$	**Umbral (t)**	250	3	250	215	66
	Tiempo (s)	0,002	0,631	0,640	0,644	0,641
$w = 5$	**Umbral (t)**	250	1	250	254	66
	Tiempo (s)	0,002	0,657	0,671	0,672	0,673

Tabla 3.19.: Umbrales y tiempo de cómputo para cada imagen con el algoritmo 1 reescrito a través de funciones de Dombi.

	Ruido gausiano	Ruido impulsivo 0.05	Ruido impulsivo 0.2
Alg. 3A	131	136	144
Alg. 3B	131	234	152
Alg. 3C	254	234	250

Tabla 3.20.: Umbrales para versión del algoritmo 3 reescrito con imágenes ruidosas.

Capítulo 4.

Segmentación de imágenes con agregando con funciones OWA

En este capítulo se llevarán a cabo otro tipo de pruebas sobre los algoritmos presentados. Para ello, en primer lugar, se explicará el modo en el que se modificarán los algoritmos presentados en la sección 3.1. Después se presentarán los resultados obtenidos con estas fórmulas, haciendo hincapié en los más interesantes.

4.1. Experimentos y resultados con funciones de agregación OWA

4.1.1. Experimento 4: sustitución de la media aritmética por funciones OWA en la construcción de los conjuntos difusos

Para todos los experimentos que se disponen en este capítulo, tomaremos la determinación de sustituir la media aritmética que se utiliza en la construcción de los conjuntos difusos (def. 3.1.1 y 3.1.2) por una función OWA de tipo 'la mayoría de' (def. 2.4.3). Para ello es importante destacar el hecho de que habrá que introducir todos los datos en forma normalizada (con valores en $[0, 1]$).

En concreto, se utilizarán dos posibilidades de funciones OWA. La primera de ellas, OWA (1), crea los pesos según la ecuación 2.4 habiendo ordenado el vector en función del histograma. En el segundo caso, OWA (2), crearemos los pesos de igual manera, pero en el momento de aplicarlos los multiplicaremos por la función $h(q)$ que les corresponda.

Explicación del experimento

En este primer experimento se sustituirá la media aritmética en la creación de los conjuntos difusos en el algoritmo 1. Se hará con las dos opciones de OWA presentadas. Se conoce de antemano, que debido a la necesidad del cálculo de los pesos, el tiempo de computación crecerá bastante frente a la versión original.

Resultados

En las tablas 4.1 y 4.2 se muestran los resultados obtenidos para todas las imágenes que se toman como muestra. Se ha de destacar que este experimento, al igual que todos, se ha llevado a cabo con un conjunto de casi 30 imágenes de las que se presentan las más significativas.

Se puede observar que claramente los resultados obtenidos son extremadamente alejados de la realidad. Se crean únicamente umbrales que no permiten llevar a cabo ninguna segmentación de forma adecuada por lo que se omite la presentación de los resultados gráficos así como de los errores en este experimento.

4.1.2. Experimento 5: sustitución de la media aritmética por funciones OWA en la construcción de los conjuntos difusos para el algoritmo con función penalti

Explicación del experimento

Para este experimento se sustituirá la media aritmética en la creación del numerador necesario en el algoritmo 2. De nuevo, se hará con las dos opciones de OWA presentadas. De forma recurrente también, se tiene en cuenta que el tiempo de cómputo subirá de forma extraordinaria, más aun si recordamos que este algoritmo inicialmente hacía pocos cálculos y, por tanto, utiliza poco tiempo.

Resultados

Si se estudian las tablas 4.3 y 4.4 se encontrarán de nuevo unos resultados decepcionantes para la umbralización llevada a cabo. Excepto en casos concretos (todos ellos dependientes de $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$)

Silla	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	127	50	50
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	127	50	246
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	119	50	50
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	127	50	50
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	127	50	50

Bloques	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	79	12	7
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	97	11	10
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	47	13	4
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	70	14	7
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	82	12	10

Engranaje	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	104	7	4
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	104	13	4
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	84	4	1
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	105	5	4
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	104	11	4

Letras	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	187	255	239
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	174	255	239
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	200	255	239
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	190	255	236
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	186	255	255

Sombra	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	123	255	231
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	126	255	255
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	121	46	85
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	123	255	202
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	124	255	255

Tabla 4.1.: Umbrales para todas las versiones del algoritmo 1 con la aplicación de OWA.

R. gausiano	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	131	50	0
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	131	12	0
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	132	56	0
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	131	56	0
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	131	43	0

R. impulsivo 0,05	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	127	50	50
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	127	50	50
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	144	50	50
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	127	50	50
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	127	50	50

R. impulsivo 0,2	**Media**	**OWA (1)**	**OWA (2)**
Alg. 1 con $REF_1 = 1 - \|x - y\|$	127	50	50
Alg. 1 con $REF_1 = 1 - \|x - y\|^2$	127	50	0
Alg. 1 con $REF_1 = 1 - \|x - y\|^{0.5}$	152	50	0
Alg. 1 con $REF_1 = (1 - \|x - y\|)^2$	127	50	50
Alg. 1 con $REF_1 = (1 - \|x - y\|)^{0.5}$	127	50	0

Tabla 4.2.: Umbrales para todas las versiones del algoritmo 1 con la aplicación de OWA para imágenes con ruido.

Se puede observar que claramente los resultados obtenidos son extremadamente alejados de la realidad. Se crean únicamente umbrales que no permiten llevar a cabo ninguna segmentación de forma adecuada por lo que se omite la presentación de los resultados gráficos así como de los errores en este experimento.

Silla	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	119	50	50
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	119	58	50
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	103	114	172
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	127	50	50

Bloques	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	47	13	4
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	39	13	4
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	30	11	65
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	79	12	7

Engranaje	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	84	4	1
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	78	4	0
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	147	89	193
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	104	7	4

Letras	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	121	46	85
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	121	54	86
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	101	136	136
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	123	255	231

Sombra	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	200	255	239
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	201	255	235
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	178	171	168
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	187	255	239

Tabla 4.3.: Umbrales para todas las versiones del algoritmo 2 con la aplicación de OWA.

R. gausiano	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	132	56	0
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	132	59	0
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	99	154	159
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	131	50	0

R. impulsivo 0,05	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	144	50	50
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	144	58	50
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	111	122	172
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	127	50	50

R. impulsivo 0,2	**Media**	**OWA (1)**	**OWA (2)**
Alg. 2 con $\varphi_1 = \varphi_2 = x$	152	50	0
Alg. 2 con $\varphi_1 = x^2$ y $\varphi_2 = x$	160	50	0
Alg. 2 con $\varphi_1 = x^{0,5}$ y $\varphi_2 = x$	127	131	172
Alg. 2 con $\varphi_1 = 1 - \sqrt{1-x}$ y $\varphi_2 = x$	127	50	50

Tabla 4.4.: Umbrales para todas las versiones del algoritmo 2 con la aplicación de OWA para imágenes con ruido.

4.1.3. Experimento 6: sustitución de la media aritmética por funciones OWA en la construcción de los conjuntos difusos para el algoritmo con función penalti

Explicación del experimento

En este caso, debido a que este algoritmo es una modificación del primero, se volverá a usar funciones OWA para la creación de los conjuntos difusos que representen las imágenes.

Resultados

Es curioso observar como el resultado cambia de forma radical. En la Tabla 4.5 se pueden observar los umbrales que se encuentran aplicados en las imágenes en la siguiente (tabla 4.6). En cualquier caso, este resultado tiene aún un problema, pues se gasta mucho tiempo de computación para, como mucho, igualar el resultado sino no empeorarlo como sucede con algún ejemplo.

	Media	OWA (1)	OWA (2)
Silla	136	50	50
Bloques	82	12	10
Engranaje	102	7	4
Letras	193	255	239
Sombra	122	255	205

Tabla 4.5.: Umbrales para la versión agregada del algoritmo 1 con la aplicación de OWA.

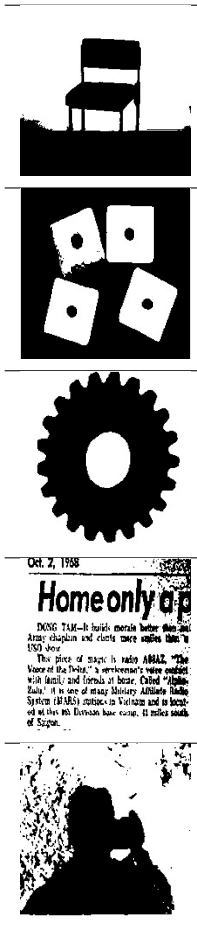
Media	OWA (1)	OWA (2)
		

Tabla 4.6.: Resultados gráficos para la versión agregada del algoritmo 1 con la aplicación de OWA.

Se presentan, también, los resultados de aplicar sobre imágenes con ruido. Se ve que, claramente, existe una tendencia de que la segunda forma de crear el OWA haga que el umbral se dispare hacia un extremo. Aun así, y de nuevo se puede apreciar claramente con la imagen segmentada de la tabla 4.8, que la segmentación es bastante peor, sin contar el tiempo de computación extra que es necesario.

	Media	**OWA (1)**	**OWA (2)**
R. gausiano	131	56	0
R. impulsivo 0,05	127	50	50
R. impulsivo 0,2	136	50	50

Tabla 4.7.: Resultados gráficos para la versión agregada del algoritmo 1 con la aplicación de OWA.

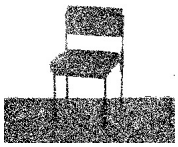
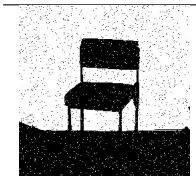
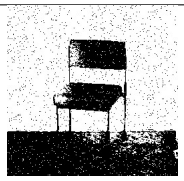
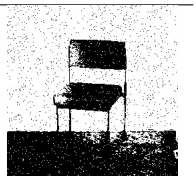
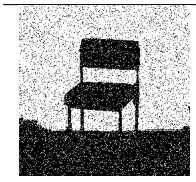
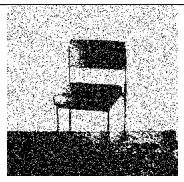
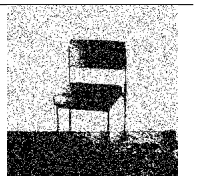
Media	**OWA (1)**	**OWA (2)**
		
		
		

Tabla 4.8.: Resultados gráficos para la versión agregada del algoritmo 1 con la aplicación de OWA en imágenes con ruido.

4.1.4. Experimento 7: sustitución de la media aritmética por funciones OWA en la construcción de los conjutos difusos para el algoritmo del umbral óptimo por similitud

Explicación del experimento

En este caso, se cambiará la media aritmética que se encuentra en la creación del conjunto H que se utiliza para conocer la similitud de las imágenes. Se mantendrá de esta forma

durante todo el experimento. Únicamente cambiará la media que se encuentra en la creación de los conjuntos difusos que se comparan contra H.

Se distinguirán dos casos, por medio de las dos funciones OWA que se han explicado. Aquel que acoge a la OWA (1) es el Alg. 3 (a) y el que multiplica por el histograma será el Alg. 3 (b).

Resultados

De nuevo se vuelve a observar que no se dan buenos resultados para aquellas opciones que siguen que siguen teniendo la construcción del conjunto Q_t por medio de OWA. En cambio, se observa un cambio muy interesante, ya que parece que la versión que se utiliza con la media aritmética pero con OWA para crear el conjunto H obtiene grandes resultados, para el caso en el cual no se multiplica por el histograma (Alg 3 (a)). Tampoco se desdiñen los resultados obtenidos con e OWA (2), la versión (b), aunque no son tan buenos.

Silla	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	119	50	50
Alg. 3 (b)	103	114	172

Bloques	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	47	13	4
Alg. 3 (b)	30	11	65

Engranaje	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	84	4	1
Alg. 3 (b)	147	89	193

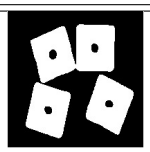
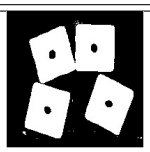
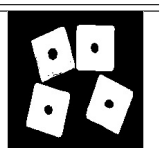
Letras	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	200	255	236
Alg. 3 (b)	178	171	168

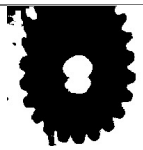
Sombra	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	121	255	85
Alg. 3 (b)	101	136	136

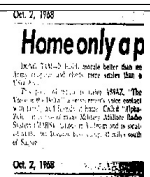
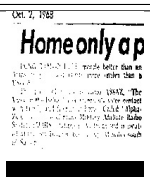
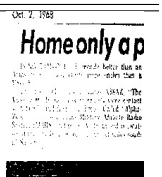
Tabla 4.9.: Umbrales para cada imagen con el algoritmo 3 a través la aplicación de OWA.

En el caso de las imágenes con ruido, la tendencia es exactamente la misma. Como ha ocurrido en todos los experimentos, no se diluye el ruido ya que con un umbral este no

Silla	Media	OWA (1)	OWA (2)
Alg. 3 (a)			
Alg. 3 (b)			

Bloques	Media	OWA (1)	OWA (2)
Alg. 3 (a)			
Alg. 3 (b)			

Engranaje	Media	OWA (1)	OWA (2)
Alg. 3 (a)			
Alg. 3 (b)			

Letras	Media	OWA (1)	OWA (2)
Alg. 3 (a)			
Alg. 3 (b)			

Sombra	Media	OWA (1)	OWA (2)
Alg. 3 (a)			
Alg. 3 (b)			

Tabla 4.10.: Resultados gráficos para el algoritmo 3 con la aplicación de OWA.

puede desaparecer. Esto se se muy claro en el ruido impulsivo. Se solucionaría con un filtro.

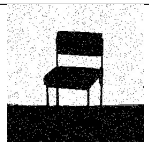
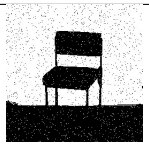
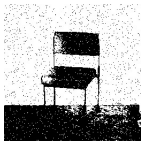
R. gausiano	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	132	56	0
Alg. 3 (b)	99	154	159

R. impulsivo 0,05	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	144	50	50
Alg. 3 (b)	111	122	172

R. impulsivo 0,2	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)	152	50	0
Alg. 3 (b)	127	131	172

Tabla 4.11.: Umbrales para cada imagen con el algoritmo 3 a través la aplicación de OWA en imágenes con ruido.

R. gausiano	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)			
Alg. 3 (b)			

R. impulsivo 0,05	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)			
Alg. 3 (b)			

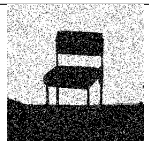
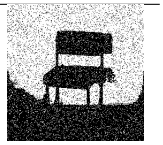
R. impulsivo 0,2	**Media**	**OWA (1)**	**OWA (2)**
Alg. 3 (a)			
Alg. 3 (b)			

Tabla 4.12.: Resultados gráficos para el algoritmo 3 con la aplicación de OWA en imágenes con ruido.

Capítulo 5.

Conclusiones y líneas de futuro

En primer lugar, con respecto a las funciones de Dombi, podemos comentar que estas parecían funcionar bien con una imagen y que sería determinante el parámetro w, siendo diferente para cada imagen. Esto parecía indicar que sería necesario poder entrenar con un conjunto de imágenes un sistema experto para poder obtener siempre la mejor umbralización en función de unos parámetros dados.

Al utilizar la función de Dombi en conjunto con otras o con ella misma es cuando se concluye que esta función no es adecuada para poder hacer segmentación de imagen. No al menos con la construcción de conjuntos difusos que se ha presentado en este estudio. Esto se debe a que las funciones de Dombi no cumplen de forma absoluta que para la función de equivalencia e, $e(x, x) = 1$. Entran a solucionar la paradoja que se produce con la propiedad $e(x, c(x)) = 0$ lo que hace que se convierta en inutil para todas nuestras conclusiones.

Con respecto al método que se ha presentado que utiliza las funciones penalti (sección 3.2.2), se debe decir que consume mucho más tiempo de procesamiento. Ahora bien, ciertamente los errores comparados con los resultados obtenidos por otros métodos hacen ver el método satisface de manera muy buena la solución del problema. Se tiene que tener en cuenta que en la versión que se ha implementado con funciones de Dombi, estas, en vez de empeorar el resultado le hacen abarcar una mayor amplitud de posibilidades algo que hace mejorar este con respecto al algoritmo original (*outlayers*). Si bien es cierto que se nota una cierta mejoría de la propuesta cuando en el histograma, que dispone de varios mínimos, son más similares entre sí.

Por otra parte, con relación a las funciones OWA para sustituir a la media aritmética se comprueba experimentalmente que no es una buena idea para obtener una buena segmentación. Quizá esto se deba a que al ordenar el OWA, si hay dos píxeles diferentes con igual frecuencia de $h(q)$ se elige el de tonalidad más alta. Por esta razón, si el rango dinámico de la imagen es muy pequeño, habrá una serie de píxeles con tonos cercanos a 255 que obtengan un gran beneficio de ello.

Esta situación cambia cuando se utilizan las funciones penalti para obtener el mejor umbral. Aun así, esta situación no es la mejor para estas pruebas. Se ha de reconocer que el algoritmo de maximización de la similitud para obtener el umbral óptimo, si se utiliza la media aritmética para crear los conjuntos Q_t y el OWA sin multiplicar por el valor $h(q)$, se obtiene el mejor resultado para toda esta batería de experimentos.

Ninguno de los algoritmos presentados funciona mejor con imágenes ruidosas que los ya presentados por otros autores. Esto es lógico ya que, por ejemplo, en el caso de tener ruido impulsivo. Si un cierto píxel que originalmente pertenecía a una región ahora ha cambiado de forma que la tonalidad le hace caer en la otra. Esto es irresoluble, ya que aunque existen técnicas para conocer si un píxel es adecuado a su entorno, estas no son aplicadas por la segmentación. En este caso, se podría probar cómo afecta el aplicar un filtro a la imagen para luego segmentarla, lo que podría solucionar el problema.

Por todo ello, acabado el estudio se concluye que tanto el método de funciones penalti analizado como la variación que se hace sobre el algoritmo 3 son una buena forma de de segmentar y que esto mejora los resultados que se obtenian con las versiones originales de ambos algoritmos. Por esta razón, una forma de mejorar estas la versión que utiliza la función penalti sería dirimir qué funciones de agregación son las que más influyen en la decisión con intención de quedarse solo con ellas. Esto provocaría una reducción del tiempo de cómputo.

Otra posible mejora, en ambos algoritmos, tal y como planteaba este trabajo, sería sustituir o añadir más funciones para calcular la pertenencia de los conjuntos difusos, además de las REF. Esto en un principio podría crear un programa más pesado computacionalmente hablando pero si se eligen de forma adecuada las funciones que más ayudan a la decisión. Parece instantáneo pensar algo similiar a lo propuesto anteriormente de buscar un subconjunto de funciones para llevar a cabo menos cálculos.

Se hace curioso comprobar que para una imagen con y sin ruido como la silla se obtengan los umbrales más similares a otros métodos con el mismo valor para w. Esto hace suponer, por tanto que sea aquí quizá donde la función alcance el pleno de la propiedad $e(x, x) = 1$. En cualquier caso, como ya se ha dicho, esto es inutil ya que cada imagen necesita un valor de w diferente por lo que no es posible llegar a un algoritmo universal salvo, quizá, a través de un sistema experto y algoritmos bioinspirados.

Anexo A.

Implementaciones de los algoritmos

Este apéndice pretende recoger todo el código que se ha utilizado para implementar los algoritmos que se han presentado para este trabajo. En particular, para cada algoritmo se detalla su función principal y las auxiliares, si existieran.

A.1. Algoritmo general maximizando la similitud

En este primer algoritmo, se ha implementado una función principal `alg1(I, tipoREF1, d)` (extracto A.1) en la cual se introduce como entrada la imagen en formato `uint8` junto con un número del 1 al 6 que significará el tipo de función que se utiliza para crear los conjuntos difusos.

Extracto de código A.1: Función principal de la implementación del algoritmo 1.

```
% TRABAJO FINAL DE GRADO - UMBRALIZACION GLOBAL
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Funcion que implementa el algoritmo 1 para las funciones de Dombi.
function [tseg, segImg] = alg1(I, tipoREF1, d)

    % Si no se introduce d, autom?ticamente se utiliza 1.
    if nargin < 3
        d = 1;
    end

    % Cardinal total de intensidades de gris.
    L = 256;
```

```
    Lminus1 = 255;

    % Muestro la pertenencia de cada elemento a la imagen (histograma).
    hq = [(0:L-1)' imhist(I)]; % Vectores columna, [t, hist(t)].

    % 1. Creamos los L conjuntos difusos para una t dada.
    Qt = zeros(L);
    for t=0:Lminus1
        % 1.1. Dividimos la imagen en dos clases, Cb y Co y calculamos su media.
        % 1.2. Calculamos la funcion de pertenencia a traves de las medias de 1.1.
        for q=0:t
            Qt(q+1, t+1) = REF1(q/Lminus1, mb(hq, t)/Lminus1, tipoREF1, d);
        end
        for q = t+1:Lminus1
            Qt(q+1, t+1) = REF1(q/Lminus1, mo(hq, t, L)/Lminus1, tipoREF1, d);
        end
    end
    Qt(isnan(Qt))=0; % Se retiran los NaN que se producen.

    % 2. Seleccionamos la REF2 como x y M como la media aritmetica.
    % Se dispone de la funcion REF2(x,y) que implementa 1-|x-y|^2.

    % 3. Calcular la similitud entre el conjunto Qt y el conjunto 1.
    sumHist = sum(hq(:,2));
    similitud = zeros(1, L);
    for t = 0:L-1
        similitud(t+1) = sum(hq(:,2).*REF2(1,Qt(:,t+1))) / sumHist;
    end

    % 4. Tomar como umbral el mejor valor de similud para el apartado
    % anterior.
    invt = find(similitud == max(similitud))-1;
    tseg = round(mean(invt));

    % Se crea la imagen resultado.
    segImg=I;
    segImg(I>tseg(1))=255;
    segImg(I<=tseg(1))=0;
end
```

Para poder hacer uso del programa que se acaba de presentar, se necesitan varias funciones auxiliares que se enumeran en los extractos de código A.2 A.3, A.4 y A.5.

Extracto de código A.2: Función auxiliar para obtener la media del fondo.

```
% Media de los pixeles del fondo (background).
function m = mb(Q, t)
    m = sum(Q(1:t+1,2).*Q(1:t+1,1))/sum(Q(1:t+1,2));
    if isnan(m)
        m=0;
    end
end
```

Extracto de código A.3: Función auxiliar para obtener la media del objeto.

```
% Media de los pixeles del objeto.
function m = mo(Q, t, L)
    m = sum(Q(t+2:L,2).*Q(t+2:L,1))/sum(Q(t+2:L,2));
    if isnan(m)
        m=0;
    end
end
```

Extracto de código A.4: Función auxiliar para poder utilizar la REF1.

```
% Funciones REF utilizadas en el primer lugar.
function eql = REF1(x, y, tipo, d)
    if tipo==1
        eql = 1 - abs(x-y);
    elseif tipo==2
        eql = 1 - abs(x-y).^2;
    elseif tipo==3
        eql = 1 - abs(x-y).^0.5;
    elseif tipo==4
        eql = (1 - abs(x-y)).^2;
    elseif tipo==5
        eql = (1 - abs(x-y)).^0.5;
    elseif tipo==6 % Con los operadores de equivalencia de Dombi d=0.5 (No es una REF)
        eql = .5*(1+((1-2*x)^d)*((1-2*y)^d));
    end
end
```

Extracto de código A.5: Función auxiliar para poder utilizar la REF2.

```
%Funciones REF utilizadas en el segundo lugar.
function eql = REF2(x, y)
    eql = 1 - abs(x-y).^2;
end
```

A continuación, se presenta el código que se ha utilizado para implementar el algoritmo que basa sue elección en las funciones penalti. No se mostarán aquellas partes del código comunes con el algoritmo original.

Extracto de código A.6: Función principal de la implementación del algoritmo 1 con funciones penalti.

```
% TRABAJO FINAL DE GRADO - UMBRALIZACION GLOBAL
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Funcion que implementa el algoritmo 1 para las funciones de Dombi
% utilizando las funciones penalti.
function [tseg, segImg] = alg1agregate(I, tipoREF1)

% Cardinal total de intensidades de gris.
L = 256;
Lminus1 = 255;
% Muestro la pertenencia de cada elemento a la imagen.
hq = [(0:L-1)' imhist(I)]; % Vectores columna, [t, hist(t)].

    % 1. Creamos los L conjuntos difusos pa  ra una t dada.
    for t=0:Lminus1
        % 1.1. Dividimos la imagen en dos clases, Cb y Co y calculamos su media.
        % 1.2. Calculamos la funci  n de pertenencia a trav  s de las medias de 1.1.
        for q=0:t
            Qt(q+1, t+1) = agregateREF1(q/Lminus1, mb(hq, t)/Lminus1);
        end
        for q = t+1:Lminus1
            Qt(q+1, t+1) = agregateREF1(q/Lminus1, mo(hq, t, L)/Lminus1);
        end
    end
    % Para retirar el NaN que se produce
    Qt(isnan(Qt))=0;

    % 2. Seleccionamos la REF2 como x y M como la media aritm  tica. TODO!
```

```
    % En realidad ya esta seleccionada. REF2(x,y)=1-|x-y|^2.

    % 3. Calcular la similitud entre el conjunto Qt y el conjunto 1.
    sumHist = sum(hq(:,2));
    for t = 0:L-1
        similitud(t+1) = sum(hq(:,2).*REF2(1,Qt(:,t+1))) / sumHist;
    end
    % 4. Tomar como umbral el mejor valor de similud para el apartado
    % anterior.
    invt = find(similitud == max(similitud))-1; % Funciona siempre que exista un solo maximo local.
    tseg = round(mean(invt));

    % Muestra de la imagen y el resultado.
    segImg=I;
    segImg(I>tseg(1))=255;
    segImg(I<=tseg(1))=0;
end
```

Extracto de código A.7: Función auxiliar para utilizar la REF1 con las funciones penalti implementadas.

```
% Agregación de las posibles funciones REF.
function eql = agregateREF1(a, b)
    % Sacamos todas las REF posibles
    tipos = [1:5 6];
    for i=1:length(tipos)
        r(i) = REF1(a,b,tipos(i));
    end
    r(isnan(r))=0;

    % Agregamos los resultados anteiores en todas sus formas
    numAgregate = 7;
    for i=1:numAgregate
        ag(i) = agregate(r,i);
    end

    n = 6;
    numPenalty = numAgregate;
    for i=1:numPenalty
        p(i) = sum(abs(r-ag(i))) / n;
    end

    where = find(p == min(p));
    eql = ag(where(1));
end
```

Extracto de código A.8: Función auxiliar para poder llevar a cabo todas las agregaciones que se necesitan para calcular una funcion penalti.

```
function eql = agregate(r, tipo)
    n=6;
    if tipo==1
        eql = mean(r);
    elseif tipo==2
        %w = [1/3, 1/3, 1/3, 0, 0, 0];
        rsort = sort(r);
        eql = sum(w.*rsort);
    elseif tipo==3
        [xsigma, sig] = sort(r);
        m0 = (sig(1:n)/n).^2;
        m1 = [(sig(2:n)/n).^2 0];
        w = m0-m1;
        eql = sum(w.*xsigma);
    elseif tipo==4
        eql = min(r);
    elseif tipo==5
        eql = max(r);
    elseif tipo==6
        eql = max(0, sum(r-(n-1))) * geomean(r);
    elseif tipo==7
        eql = geomean(r);
    end
end
```

Seguidamente, se muestra la versión que implementa la posibilidad de incorporar funciones OWA en los conjuntos difusos.

Extracto de código A.9: Función principal que implementa el algoritmo 1 creando los conjuntos difusos con funciones OWA.

```
% TRABAJO FINAL DE GRADO - ALGORITMO 1 AGREGADO Y OWA
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Algoritmo 1 para segmentar la imagen utilizando owa para crear los conjuntos difusos.
function [tseg, segImg] = alg1owa(I, tipoREF1, tipoOWA)

    % Si no se introduce el owa, automaticamente se utiliza el 2.
    if nargin < 3
        tipoOWA = 2;
```

```
    end

    % Cardinal total de intensidades de gris.
    L = 256;
    Lminus1 = 255;

    % Muestro la pertenencia de cada elemento a la imagen (histograma).
    hq = [(0:L-1)' imhist(I)]; % Vectores columna, [t, hist(t)].

    % 1. Creamos los L conjuntos difusos para una t dada.
    Qt = zeros(L);
    for t=0:Lminus1
        % 1.1. Dividimos la imagen en dos clases, Cb y Co y calculamos su media.
        % 1.2. Calculamos la funci  n de pertenencia a trav  s de las medias de 1.1.
        for q=0:t
            Qt(q+1, t+1) = REF1(q/Lminus1, owab(hq, t, tipoOWA), tipoREF1);
        end
        for q = t+1:Lminus1
            Qt(q+1, t+1) = REF1(q/Lminus1, owao(hq, t, L, tipoOWA), tipoREF1);
        end
    end
    Qt(isnan(Qt))=0; % Se retiran los NaN que se producen.

    % 2. Seleccionamos la REF2 como x y M como la media aritmetica.
    % Se dispone de la funcion REF2(x,y) que implementa 1-|x-y|^2.

    % 3. Calcular la similitud entre el conjunto Qt y el conjunto 1.
    sumHist = sum(hq(:,2));
    similitud = zeros(1, L);
    for t = 0:L-1
        similitud(t+1) = sum(hq(:,2).*REF2(1,Qt(:,t+1))) / sumHist;
    end

    % 4. Tomar como umbral el mejor valor de similud para el apartado
    % anterior.
    invt = find(similitud == max(similitud))-1;
    tseg = round(mean(invt));

    % Se crea la imagen resultado.
    segImg=I;
    segImg(I>tseg(1))=255;
    segImg(I<=tseg(1))=0;
end
```

Extracto de código A.10: Función auxiliar para obtener la media u OWA del fondo.

```
% Función para obtener los owa relativos al objeto.
function m = owab(Q, t, tipo)

    if tipo == 1 % Media de la imagen habitual

         m = sum(Q(1:t+1,2).*Q(1:t+1,1))/sum(Q(1:t+1,2));
         m = m/255;

    elseif tipo == 2 % OWA sin frecuencia.

        % t tiene que estar normalizado.
        intervalo = Q(1:t+1,2);
        tplus1norm = (t+1)/255;
        norm1 = 1/255;

        [~, orden] = sort(intervalo);
        ordenNorm = (orden-1)./255;

        i=1;
        w = zeros(1,t+1);
        for j= norm1 : norm1 : tplus1norm
            w(i) = funcPesos(j/tplus1norm)-funcPesos((j-norm1)/tplus1norm);
            i=i+1;
        end

        m = w*ordenNorm;

    else % OWA con frecuencia.

        % t tiene que estar normalizado.
        intervalo = Q(1:t+1,2);
        tplus1norm = (t+1)/255;
        norm1 = 1/255;

        [frec, orden] = sort(intervalo);
        ordenNorm = (orden-1)./255;
        totalFrec = sum(frec);
        normFrec = frec/totalFrec;

        i=1;
        w = zeros(1,t+1);
        for j= norm1 : norm1 : tplus1norm
            w(i) = funcPesos(j/tplus1norm)-funcPesos((j-norm1)/tplus1norm);
            i=i+1;
        end

```

```
        m = w*(ordenNorm.*normFrec);
    end

    if isnan(m) %Retiramos todos los problemas.
        m=0;
    end
end
```

Extracto de código A.11: Función auxiliar para obtener la media u OWA del objeto.

```
% Función para obtener los owa relativos al objeto.
function m = owao(Q, t, L, tipo)

    if tipo == 1 % Media de la imagen habitual

         m = sum(Q(t+2:L,2).*Q(t+2:L,1))/sum(Q(t+2:L,2));
         m = m/255;

    elseif tipo == 2 % OWA sin frecuencia

        % t tiene que estar normalizado.
        intervalo = Q(t+2:L,2);
        tplus1norm = (t+1)/255;
        norm1 = 1/255;

        [~, orden] = sort(intervalo);
        ordenNorm = (orden-1)./255;

        i=1;
        w = zeros(1,L-(t+1));
        for j= tplus1norm : norm1 : 1
            w(i) = funcPesos(j/1)-funcPesos((j-norm1)/1);
            i=i+1;
        end

        m = w*ordenNorm;

    else % OWA con frecuencia

        % t tiene que estar normalizado.
        intervalo = Q(t+2:L,2);
        tplus1norm = (t+1)/255;
        norm1 = 1/255;

        [frec, orden] = sort(intervalo);
        ordenNorm = (orden-1)./255;
```

```
        totalFrec = sum(frec);
        normFrec = frec/totalFrec;

        i=1;
        w = zeros(1,L-(t+1));
        for j= tplus1norm+norm1 : norm1 : 1+norm1
            w(i) = funcPesos(j/(1+norm1))-funcPesos((j-norm1)/(1+norm1));
            i=i+1;
        end

        m = w*(ordenNorm.*normFrec);
    end

    if isnan(m) %Retiramos todos los problemas.
        m=0;
    end
end
```

Extracto de código A.12: Función auxiliar para calcular los pesos de la función owa.

```
% Función auxiliar para calcular los pesos de la funcion owa.
function Q = funcPesos(r)
        if r < 0.5
            Q = 0;
        else
            Q = (r-0.5)/0.5; % Introduzco la t normalizada.
        end
end
```

Por último, se muestra la versión agregada y que dispone de funciones OWA para crear los conjuntos difusos.

Extracto de código A.13: Función principal que implementa el algoritmo 1 que utiliza funciones penalti y crea los conjuntos difusos con funciones OWA.

```
% TRABAJO FINAL DE GRADO - ALGORITMO 1 AGREGADO Y OWA
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Algoritmo 1 para segmentar la imagen agregado con funciones penalti
% y utilizando owa para crear los conjuntos difusos.
function [tseg, segImg] = alg1agregateowa(I, tipoOWA)
```

```
8
9  % Cardinal total de intensidades de gris.
10     L = 256;
11     Lminus1 = 255;
12     % Muestro la pertenencia de cada elemento a la imagen.
13     hq = [(0:L-1)' imhist(I)]; % Vectores columna, [t, hist(t)].
14
15     % 1. Creamos los L conjuntos difusos para una t dada.
16     for t=0:Lminus1
17         % 1.1. Dividimos la imagen en dos clases, Cb y Co y calculamos su media.
18         % 1.2. Calculamos la funci  n de pertenencia a trav  s de las medias de 1.1.
19         for q=0:t
20             Qt(q+1, t+1) = agregateREF1(q/Lminus1, owab(hq, t, tipoOWA)); % TODO OPTIMIZAR!!
21         end
22         for q = t+1:Lminus1
23             Qt(q+1, t+1) = agregateREF1(q/Lminus1, owao(hq, t, L, tipoOWA)); % TODO OPTIMIZAR!!
24         end
25     end
26     % Para retirar el NaN que se produce
27     Qt(isnan(Qt))=0;
28
29     % 2. Seleccionamos la REF2 como x y M como la media aritm  tica. TODO!
30     % En realidad ya esta seleccionada. REF2(x,y)=1-|x-y|^2.
31
32     % 3. Calcular la similitud entre el conjunto Qt y el conjunto 1.
33     sumHist = sum(hq(:,2));
34     for t = 0:L-1
35         similitud(t+1) = sum(hq(:,2).*REF2(1,Qt(:,t+1))) / sumHist;
36     end
37     % 4. Tomar como umbral el mejor valor de similud para el apartado
38     % anterior.
39     invt = find(similitud == max(similitud))-1; % Funciona siempre que exista un solo maximo local.
40     tseg = round(mean(invt));
41
42
43     % Muestra de la imagen y el resultado.
44     segImg=I;
45     segImg(I>tseg(1))=255;
46     segImg(I<=tseg(1))=0;
47 end
```

A.2. Algoritmo del área

Para la implementación del algoritmo 2, se muestra directamente la que se lleva a cabo con funciones OWA ya que si en la entrada de este parámetro se coloca un 1, entonces se tendrá el algoritmo original.

Extracto de código A.14: Función auxiliar para obtener la media u OWA del objeto.

```
% TRABAJO FINAL DE GRADO - ALGORITMO 2, DEL AREA
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Algoritmo 2 para segmentar la imagen utilizando funciones OWA
% para la creacion de los conjuntos difusos.
function [tseg, segImg] = alg2owa(I, tipoREF, tipoOWA)

    % Cardinal total de intensidades de gris.
    L = 256;
    Lminus1 = L-1;

    % Se muestrea la pertenencia de cada elemento a la imagen.
    hq = [(0:Lminus1)' imhist(I)]; % Vectores columna, [t-1, hist(t-1)].
    sumHist = sum(hq(:,2)); % Constante para acelerar el c  mputo.

    % 1. Seleccionamos los automorfismos 1 y 2. (Se indican en cada tipo).

    % 2. Bucle sobre los umbrales t
    A = zeros(0,L);
    % 2.1. Calculamos el area
    if(tipoREF==1)
        % (i) aut1=x y aut2=x
        for t=0:Lminus1
            A(t+1) = sumHist - (sum(hq(1:t+1,2)'.* abs((0:t)/Lminus1 - owab(hq,t,tipoOWA))) + ...
                sum(hq(t+2:L,2)'.* abs((t+1:Lminus1)/Lminus1 - owao(hq,t,L,tipoOWA))));
        end
    elseif(tipoREF==2)
        % (ii) aut1=x^d, aut2=x y d=0.5
        d=1/2;
        for t=0:Lminus1
           A(t+1) = sum(hq(1:t+1,2)'.*(1-abs((0:t)/Lminus1 - owab(hq,t,tipoOWA))).^(1/d)) + ...
                sum(hq(t+2:L,2)'.*(1-abs((t+1:Lminus1)/Lminus1 - owao(hq,t,L,tipoOWA))).^(1/d));
        end
    elseif(tipoREF==3)
        % (ii) aut1=x^d, aut2=x y d=2
```

```
        d=2;
        for t=0:Lminus1
            A(t+1) = sum(hq(1:t+1,2)'.*(1-abs((0:t) - owab(hq,t,tipoOWA))).^(1/d)) + ...
                sum(hq(t+2:L,2)'.*(1-abs((t+1:Lminus1) - owao(hq,t,L,tipoOWA))).^(1/d));
        end
    else
        % (iii) aut1=1-(1-x)^0.5 y aut2=x
        for t=0:Lminus1
            A(t+1) = sumHist - (sum(hq(1:t+1,2)'.*((0:t)/Lminus1 - owab(hq,t,tipoOWA)).^2) + ...
                sum(hq(t+2:L,2)'.*((t+1:Lminus1)/Lminus1 - owao(hq,t,L,tipoOWA)).^2));
        end
    end

    % 3. Tomar como umbral el valor con mayor area en 2.1.
        invt = find(A == max(A))-1;
        tseg = round(mean(invt));

        segImg=I;
        segImg(I>tseg(1))=255;
        segImg(I<=tseg(1))=0;
end
```

A.3. Algoritmo de selección del umbral óptimo

Extracto de código A.15: Algoritmo 3 implementado con la función original (alg. 2) y que dispone de funciones OWA.

```
% TRABAJO FINAL DE GRADO - ALGORITMO 3, DEL UMBRAL OPTIMO
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Algoritmo 3 para segmentar la imagen utilizando el algoritmo 2 como base.
function [tseg, segImg, similitud] = alg3owa(I, tipoOWA)
    % Cardinal total de intensidades de gris.
    L = 256;
    Lminus1 = L-1;

    % Muestreo la pertenencia de cada elemento a la imagen.
    hq = [(0:L-1)' imhist(I)]; % Vectores columna, [t, hist(t)].

    % 0. Hacemos el algoritmo 1 con diferentes REF y guardamos los mejores
    % umbrales.
    allt = zeros(1,4);
```

```
    similitud = zeros(1, 6);
    HQt = zeros(1, L);
    for i=1:4
        [t, ~] = alg2owa(I, i, tipoOWA);
        allt(i) = t;
        Qt = fuzzySets(hq, i, t, L, tipoOWA);

        % 1. Seleccionamos un operador M y una REF3.
          % M = Media aritm  tica.
          % REF3 = 1-|x-y|^2.

        % 2. Para cada uno de los mejores umbrales t
            % 2.1. Construir el conjunto Ht
            for q=0:L-1
                if q <= t
                    HQt(q+1) = owab(hq, t, tipoOWA);
                else
                    HQt(q+1) = owao(hq, t, L, tipoOWA);
                end
            end
            HQt(isnan(HQt))=0;

            % 2.2. Calcular la similitud
            similitud(i) = SM(hq, Qt, HQt);
    end

    % 3. Elegir aquel con maxima similitud.
    invt = find(similitud == max(similitud)); % Funciona siempre que exista un s  lo m  ximo local
    tseg = allt(invt(1));

    % Muestra de la imagen y el resultado.
    segImg=I;
    segImg(I>tseg(1))=255;
    segImg(I<=tseg(1))=0;
end
```

Extracto de código A.16: Algoritmo 3 implementado con el algorimo 1 y que dispone de funciones OWA.

```
% TRABAJO FINAL DE GRADO - ALGORITMO 3, DEL UMBRAL OPTIMO
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Algoritmo 3 para segmentar la imagen utilizando el algoritmo 1 como base.
function [tseg, segImg, similitud] = alg3bowa(I, tipoOWA)
```

```
7
8       % Cardinal total de intensidades de gris.
9       L = 256;
10
11      % Muestreo la pertenencia de cada elemento a la imagen.
12      hq = [(0:L-1)' imhist(I)]; % Vectores columna, [t, hist(t)].
13
14      % 0. Hacemos el algoritmo 1 con diferentes REF y guardamos los mejores
15      % umbrales.
16      allt = zeros(1, 6);
17      similitud = zeros(1, 6);
18      HQt = zeros(1, L);
19      for i=1:5
20          [t, ~] = alg1owa(I, i, tipoOWA);
21          allt(i) = t;
22          Qt = fuzzySets(hq, i, t, L, tipoOWA);
23
24          % 1. Seleccionamos un operador M y una REF3.
25            % M = Media aritm  tica.
26            % REF3 = 1-|x-y|^2.
27
28          % 2. Para cada uno de los mejores umbrales t
29              % 2.1. Construir el conjunto HQt
30              for q=0:L-1
31                  if q <= t
32                      HQt(q+1) = owab(hq, t, tipoOWA);
33                  else
34                      HQt(q+1) = owao(hq, t, L, tipoOWA);
35                  end
36              end
37              HQt(isnan(HQt))=0;
38
39              % 2.2. Calcular la similitud
40              similitud(i) = SM(hq, Qt, HQt);
41              %disp('.');
42      end
43
44      % 3. Elegir aquel con maxima similitud.
45      invt = find(similitud == max(similitud));
46      tseg = allt(invt(1));
47
48      % Crea la imagen final.
49      segImg=I;
50      segImg(I>tseg(1))=255;
51      segImg(I<=tseg(1))=0;
52  end
```

Extracto de código A.17: Algoritmo 3 implementado con el algoritmo 1 que dispone de la opción para probar con diferentes valores w para la función de Dombi.

```
% TRABAJO FINAL DE GRADO - ALGORITMO 3, DEL UMBRAL OPTIMO
% Inigo Aguas
% UPNA, 25 de junio de 2015.

 % Funci  n que implementa el algoritmo 3 para varios valores w de la funcion
 % de Dombi.
function [tseg, segImg, similitud] = alg3c(I)

    % Cardinal total de intensidades de gris.
    L = 256;
    Lminus1 = L-1;

    % Muestreo la pertenencia de cada elemento a la imagen.
    hq = [(0:L-1)' imhist(I)]; % Vectores columna, [t, hist(t)].

    % 0. Hacemos el algoritmo 1 con diferentes REF y guardamos los mejores
    % umbrales.
    allt = zeros(1, 6);
    similitud = zeros(1, 6);
    HQt = zeros(1, L);
    d = [0.1, 0.5, 0.75, 1, 1.25, 1.5, 2, 5];
    for i=1:length(d)
        [t, ~] = alg1(I, 6, d(i));
        allt(i) = t;
        Qt = fuzzySets(hq, 6, t, d(i), L);

        % 1. Seleccionamos un operador M y una REF3.
          % M = Media aritm  tica.
          % REF3 = 1-|x-y|^2.

        % 2. Para cada uno de los mejores umbrales t
            % 2.1. Construir el conjunto HQt
            for q=0:L-1
                if q <= t
                    HQt(q+1) = mb(hq, t) / Lminus1;
                else
                    HQt(q+1) = mo(hq, t, L) / Lminus1;
                end
            end
            HQt(isnan(HQt))=0;

```

```
            % 2.2. Calcular la similitud
            similitud(i) = SM(hq, Qt, HQt);
    end

    % 3. Elegir aquel con maxima similitud.
    invt = find(similitud == max(similitud));
    tseg = allt(invt(1));

    % Crea la imagen final.
    segImg=I;
    segImg(I>tseg(1))=255;
    segImg(I<=tseg(1))=0;
end
```

Se debe tener en cuenta que cuando se lleva a cabo el cambio en las funciones de Dombi que se ha explicado en la sección 3.2.4, se cambiarán la línea que codifica la función de Dombi en las funciones A.4 y A.7 por las líneas del extracto A.18

Extracto de código A.18: Cambio que se debe llevar a cabo para intentar solucionar el problema que presentan las funciones de Dombi.

```
numer = .5*(1+((1-2*x)^d)*((1-2*y)^d));
denom = .5*(1+((1-2*y)^d)*((1-2*y)^d));
dombi = numer/denom;
eql = min(1, dombi);
```

A.4. Algoritmo de umbralización global

Se presenta la implementación del algoritmo de umbralización global. Se ha preparado la función `globalThresholding(I)` a la que se le intruduce una imagen en `uint8` y devuelve una tupla con el umbral obtenido y la imagen segmentada. (Extracto A.19).

Extracto de código A.19: Función principal del algoritmo de umbralización global.

```
% TRABAJO FINAL DE GRADO - UMBRALIZACION GLOBAL
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Este sencillo metodo propone una forma rapida de obtener un umbral.
```

```
function [t, segImg] = globalThresholding(I)

    % 1. Seleccionamos un umbral inicial.
    t0 = 128;

    % 2. Se segmenta la imagen usando t0. Esto producira dos grupos de pixeles,
    % G1 y G2.
    G1 = I(I>t0);
    G2 = I(I<=t0);

    % 3. Se computa la media de la intensidad de los valores de ambos grupos.
    m1 = mean(G1(:));
    m2 = mean(G2(:));

    % 4. Se recalcula el valor del umbral, t:
    t = 0.5*(m1+m2);

    % 5. Se repiten los pasos anteriores hasta comprobar que la diferencia
    % entre los umbrales t0 y t es menor que un cierto error.
    error = 0;
    while (abs(t0-t) > error)

        t0 = t;

        G1 = I(I>t);
        G2 = I(I<=t);

        m1 = mean(G1(:));
        m2 = mean(G2(:));

        t = 0.5*(m1+m2);
    end

    t=round(t);

    segImg=I;
    segImg(I>t)=255;
    segImg(I<=t)=0;
end
```

A.5. Algoritmo de Otsu

En la función que se presenta en el extracto A.20 solo es necesario introducir una imagen en formato `uint8`. Se devolverá por su parte, un vector con el umbral `t` y la imagen segmentada [24].

Extracto de código A.20: Función principal del algoritmo de Otsu.

```
% TRABAJO FINAL DE GRADO - UMBRALIZACION UTILIZANDO EL METODO DE OTSU
% Inigo Aguas
% UPNA, 25 de junio de 2015.

% Para calcular este metodo utilizamos la prueba de toda la combinatoria
% diferente de todas las divisiones diferentes que se pueden dar. Es decir,
% hacemos los grupos de 0 y L-1... De eso calculamos la varianza y de aquel
% grupo que sea maxima es del que podemos decir que se aplica la t.


function [tseg, Iseg] = otsu(I)
    % Se asegura que la imagen esta en formato uint8 para tener la escala
    % de grises adecuada.
    I=im2uint8(I);

    % Se obtiene el histograma de la imagen y las probabilidades.
    [conteo, intensidad] = imhist(I);
    N = sum(conteo);
    probi = conteo / N;
    probixint = probi.*intensidad;

    % Hay dos bucles, desde 1 hasta t y desde t+1 hasta L
    L = 256;
    vars = zeros(1, L);
    for t=1:L
        w1 = sum(probi(1:t));
        w2 = sum(probi(t+1:L));
        m1 = sum(probixint(1:t))/w1;
        m2 = sum(probixint(t+1:L))/w2;
        mT = w1*m1+w2*m2;
        vari = w1*(m1-mT)^2 + w2*(m2-mT)^2;
        vars(t) = vari;
    end

    % Hayamos el umbral con la varianza maxima
    % (En caso de haber varios, cogeremos el umbral mas bajo)
```

```
37      [~, tseg] = max(vars);
38      tseg = tseg - 1;
39
40      % Se obtiene la imagen que diferencia objeto y fondo.
41      Iseg=I;
42      Iseg(I>tseg)=255;
43      Iseg(I<=tseg)=0;
44  end
```

A.6. Algoritmo de maximización de la entropía de Renyi

La función `renyi(I)` permite llevar a cabo la umbralización por medio del método propuesto por Sahoo et al. en [27]. Se devolverá por su parte, un vector con el umbral `t`, la imagen segmentada y los tres valores para el umbral que se han calculado internamente.

Extracto de código A.21: Función principal del algoritmo de maximización de la entropía de Renyi.

```
1  % TRABAJO FINAL DE GRADO - UMBRALIZACION UTILIZANDO LA ENTROPIA DE RENYI
2  % Inigo Aguas
3  % UPNA, 25 de junio de 2015.
4
5  % La siguiente funcion lleva a cabo la umbralizacion considerando la
6  % entropia de Renyi para cualquier imagen que se le introduzca como
7  % argumento. Esta funci  n es la principal para el programa presentado.
8
9  function [tc, I, ta] = renyi(im)
10
11      % Llevamos a cabo la lectura si es necesario; si no, actuamos como si
12      % la imagen ya estuviera leida.
13      if ischar(im)
14          x = im2double(imread(im)); % Leemos la imagen con formato double.
15          nom = im;
16      else
17          x = im2double(im); % Pasamos a double por si acaso.
18          nom = '';
19      end
20
21      L = 256; % Niveles de grises, constante.
```

```
22
23      [conteo, intensidad] = imhist(x); % Calculamos el histograma de la imagen,
24      % obtenemos el numero de pixeles para cada intensidad (conteo).
25      N = sum(conteo);
26      p = conteo/N; % Calculamos la probabilidad de cada intensidad de gris.
27
28      % Como Sahoo afirma que tomando un alfa cualquiera dentro de unos rangos
29      % delimitados (pag 74, primer parrafo) siempre se obtiene el mismo
30      % umbral definimos una alfa dentro de cada rango.
31      alfa = [0.3, 0.99999, 10];
32
33      % Bucle para iterar sobre los valores de alfa.
34      for i=1:3
35          a = alfa(i); % a es el valor del parametro alfa en cada momento.
36          % Bucle para iterar sobre cada uno de los umbrales posibles para
37          % tomarlos en consideracion y poder decidir de forma acertada.
38          for t=0:L-2
39              % Calculo de las probabilidades de cada uno de los grupos
40              % formados en el umbral.
41              pA1 = sum(p(1:t+1));
42              pA2 = sum(p(t+2:L));
43              % Calculo de la entropia de Renyi para cada una de las
44              % particiones formadas por el umbral.
45              HA1(t+1) = (log(sum((p(1:t+1)/pA1).^a)))/(1-a);
46              HA2(t+1) = (log(sum((p(t+2:L)/pA2).^a)))/(1-a);
47          end
48          H = HA1 + HA2; % Calculamos la entropia total para cada alfa.
49
50          % Buscamos el primer umbral que maximice la entropia para cada uno
51          % de los alfas propuestos.
52         ta(i) = find(H==max(H), 1, 'first');
53      end
54
55      % Ordenamos segun el estadistico de orden (de menor a mayor).
56      ta = sort(ta);
57      t1 = ta(1);
58      t2 = ta(2);
59      t3 = ta(3);
60
61      % Calculamos las probabilidades para cada uno de los umbrales siendo
62      % estas un cumulo desde t=1 hasta t=tx. Calculamos tambien w.
63      pt1 = sum(p(1:t1+1));
64      pt3 = sum(p(1:t3+1));
65      w = pt3 - pt1;
66
67      % Segun la circunstancia de umbrales que se muestre asignamos los beta
```

```
68      % para el calculo del umbral definitivo.
69      if abs(t1 - t2)<=5 && abs(t2 - t3)<=5
70          beta = [1, 2, 1];
71      elseif abs(t1 - t2)>5 && abs(t2 - t3)>5
72          beta = [1, 2, 1];
73      elseif abs(t1 - t2)<=5 && abs(t2 - t3)>5
74          beta = [0, 1, 3];
75      elseif abs(t1 - t2)>5 && abs(t2 - t3)<=5
76          beta = [3, 1, 0];
77      end
78
79      % Calculamos el umbral definitivo segun la ecuacion propuesta.
80      tc = t1*(pt1+.25*w*beta(1))+.25*t2*w*beta(2)+t3*(1-pt3+.25*w*beta(3));
81      tc=round(tc);
82
83      % Preparamos la imagen de salida.
84      I = im2uint8(x);
85      I(I<=tc) = 0; % Asignamos los nuevos valores a los pixeles del fondo
86      I(I>tc) = 255; % Asignamos los nuevos valores a los pixeles del objeto
87
88  end
```

A.7. Algoritmo *K-means*

La función `kmeans(I)` implementa el método de segmentación por regiones *K-means*. Se devolverá por su parte, un vector con el umbral `t` y el coste que ha tenido calcular la segmentación.

Extracto de código A.22: Función principal del algoritmo *K-means*.

```
1  % TRABAJO FINAL DE GRADO - UMBRALIZACION UTILIZANDO EL METODO KMEANS
2  % Inigo Aguas
3  % UPNA, 25 de junio de 2015.
4
5  % Funcion para la umbralizacion por medio de clasificacion de los pixeles
6  % Definimos la funcion con par  metros de entrada I, la imagen, y k, el
7  % numero de clusters.
8  function [ISeg, J] = kmeans(I, k)
9
10     % Calculo constantes del proceso e inicializo variables.
11     [M, N] = size(I);
12     I=im2double(I);
```

```
    coste = zeros(M, N);
    pertenencia = zeros(M, N);

    % Inicializo los costes para entrar en el bucle.
    J=1;
    Jant=0;

    % Inicializo los centros de forma aleatoria para llegar a su convergencia
    % en el bucle.
    centros = rand(1, k);

    % Continuo el proceso hasta conseguir que el coste converja.
    while J~=Jant

        % Calculo la pertenencia de los pixeles a los centros.
        for i=1:M
            for j=1:N
                [coste(i,j), pertenencia(i,j)] = min((I(i,j)-centros).^2);
            end
        end

        % Recalculo el nuevo coste
        Jant=J;
        J=sum(sum(coste));

        % Recalculo los nuevos centros en funcion de la pertenencia a ellos.
        for j=1:k
            if sum(sum(pertenencia==j)) == 0
                centros(j) = 0;
            else
                centros(j) = sum(sum(I(pertenencia==j)))/sum(sum(pertenencia==j));
            end
        end
    end

    % Segmento la imagen para obtener el resultado final.
    ISeg = zeros(M, N);
    for j=1:k
        ISeg = ISeg+centros(j)*(pertenencia==j);
    end
end
```

Con intención de poder llevar a acabo la comparación de las imágenes que se obtienen con otros métodos, se ha preparado la función `normalicekmeans(I)` que tiene como entrada una imagen segmentada con el método *K-means* y como salida otra que ha cambiado las tonalidades para que se correspondan con las que le daría un algoritmo

de segmentación.

Extracto de código A.23: Función para convertir el resultado de *K-means* y hacer posible su comparación.

```
% TRABAJO FINAL DE GRADO - FUNCION AUXILIAR PARA TRASLADAR LA UMBRALIZACION DE KMEANS
% Iñigo Aguas
% UPNA, 25 de junio de 2015.

% Esta funcion permite que cuando se utilice la función k-means sea posible
% convertir su resultado al mismo juego de colores que usarian los demas algoritmos.


function imgClean = normalicekmeans(img)
    [h, gray] = imhist(img);
    grays = unique(gray(h(:)~=0));
    numGrays = length(grays);
    newGrays = floor(0:255/(numGrays-1):255);
    imgClean = img;
    for i=1:numGrays
        imgClean(imgClean==grays(i)) = newGrays(i);
    end
end
```

Además, se han preparado una serie de programas para ejecutar de forma continuada y eficiente los experimentos que no se adjuntan a esta memoria ya que consisten únicamente en repeticiones de los algoritmos ya presentados.

Anexo B.

Segmentaciones con otros algoritmos

Los cuadros que se presentan en este apéndice pretenden ayudar al lector a interpretar los resultados presentados a lo largo de la memoria, conociendo el resultado de otros métodos de segmentación. Con ayuda de estos se puede hacer un análisis visual de los resultados enaminado a facilitar las conclusiones a buscar.

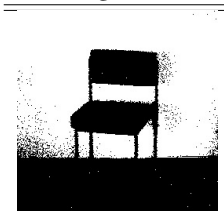
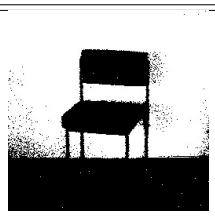
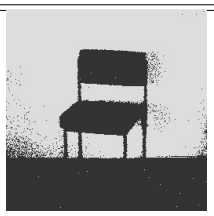
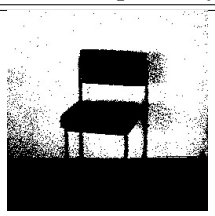
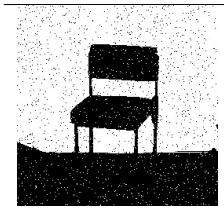
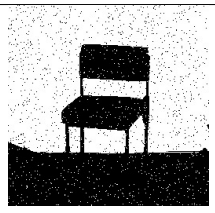
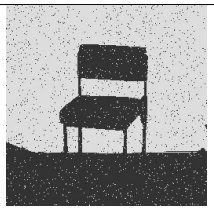
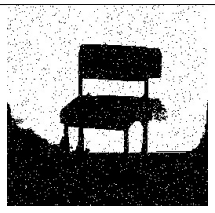
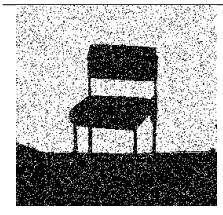
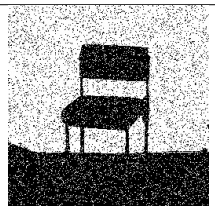
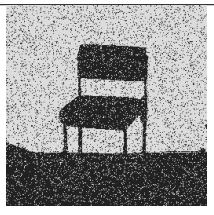
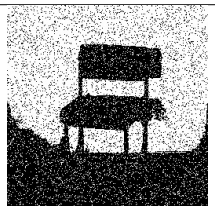
U. global	U. de Otsu	*K-means*	Max. entropía Renyi
			
			
			

Tabla B.1.: Segmentaciones para las imágenes con ruido a través de los algoritmos de otros autores.

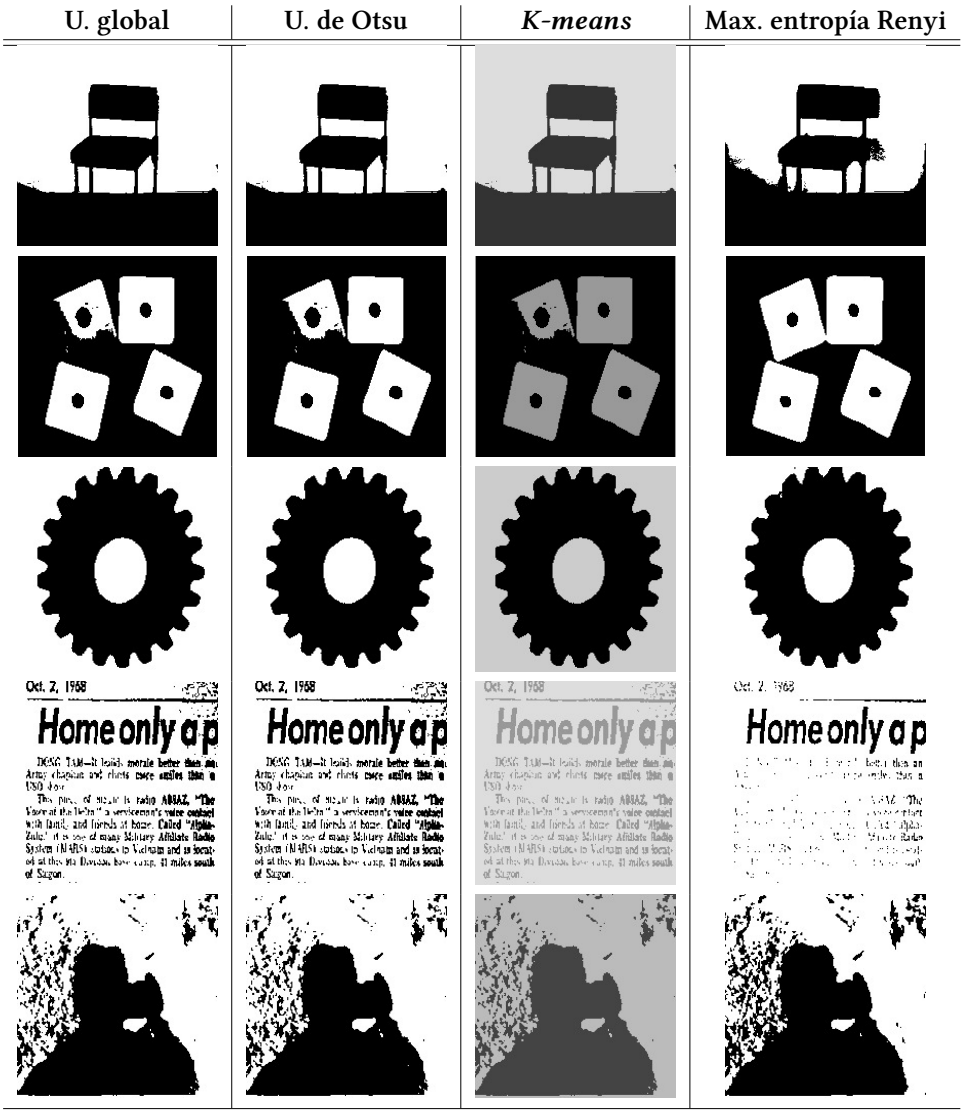


Tabla B.2.: Segmentaciones para las imágenes con los algoritmos de otros autores.

Bibliografía

[1] G. Beliakov, A. Pradera y T. Calvo. *Aggregation functions: a guide for practitioners.* Springer, 2007.

[2] B. Bhanu y S. Lee. *Genetic learning for adaptative image segmentation.* Kluwer Academic Publishers, 1994.

[3] M. A. Boden. *Inteligencia artificial y hombre natural.* Tecnos, 1984 (vid. pág. 1).

[4] H. Bustince, E. Barrenechea y M. Pagola. «Image thresholding using restricted equivalence functions and maximizing the measures of similarity». En: *Fuzzy sets and systems* 158.5 (2007), págs. 496-516 (vid. págs. 23, 25).

[5] H. Bustince, E. Barrenechea y M. Pagola. «Restricted equivalence functions». En: *Fuzzy sets and systems* 157.17 (2006), págs. 2333-2346 (vid. págs. 4, 15).

[6] H. Bustince y col. «A class of aggregation functions encompassing two-dimensional OWA operators». En: *Information Sciences* 180 (2010), págs. 1977-1989 (vid. pág. 17).

[7] H. Bustince y col. «Generalization of the weighted voting method using penalty functions constructed via faithful restricted dissimilarity functions». En: *European Journal of operational research* 225 (2013), págs. 472-478.

[8] H. Bustince y col. «Ignorance functions. An application to the calculation of the threshold in prostate ultrasound images». En: *Fuzzy sets and systems* 161 (2010), págs. 20-36 (vid. pág. 5).

[9] T. Calvo y G. Beliakov. «Aggregation functions based on penalties». En: *Fuzzy sets and systems* 161 (2010), págs. 1420-1436 (vid. pág. 19).

[10] T. Calvo y R. Mesiar. «Aggregation operators: ordering and bounds». En: *Fuzzy sets and systems* 139 (2003), págs. 685-697 (vid. pág. 16).

[11] G. Choquet. «Theory of capacities». En: *Ann. Inst. Fourier* 5 (1955), págs. 131-295 (vid. pág. 18).

[12] P. M. Churchland y P. S. Churchland. «¿Podría pensar una máquina?» En: *Investigación y ciencia* 162 (mar. de 1990), págs. 18-25 (vid. pág. 1).

[13] J. Dombi. «Equivalence operators that are associative». En: *Information Sciences* 281 (2014), págs. 281-294 (vid. págs. 6, 20, 25).

[14] J. Fan y W. Xie. «Some notes on similarity measure and proximity measure». En: *Fuzzy sets and systems* 101 (1999), págs. 403-412 (vid. pág. 18).

[15] J. Fan, W. Xie y J. Pei. «Subsethood measures: new definitions». En: *Fuzzy sets and systems* 106 (1999), págs. 201-209 (vid. pág. 18).

[16] R. C. Gonzalez y R. E. Woods. *Digital image procesing*. Pearson Prentice Hall, 2008 (vid. págs. 2, 4, 9).

[17] R. C. Gonzalez, R. E. Woods y S. L. Eddins. *Digital image procesing using MATLAB*. Gatesmark Publishing, 2009.

[18] *Investigadores de la UPNA desarrollan un método que delimita automáticamente zonas del cerebro en imágenes médicas*. Sep. de 2013. URL: http://www.unavarra.es/actualidad/noticias?contentId=173589 (vid. pág. 5).

[19] Ondrej Martinsky. «Algorithmic and mathematical principles of automatic number plate recognition systems». Tesis doct. Brno University of technology, 2007 (vid. pág. 6).

[20] F. M. McNeill y E. Thro. *Fuzzy logic. A practical approach.* AP Professional, 1994.

[21] L. Molina. «Procesamiento de textos científicos en alta calidad. Apuntes de LaTeX». Universidad de Valladolid. URL: http://metodos.fam.cie.uva.es/~latex/.

[22] J. Montero y col. «Sobre funciones y reglas de agregación». ESTYLF. 2010 (vid. pág. 16).

[23] T. Murofushi y M. Sugeno. «A theory of fuzzy measures: representations, teh Choquet Integral and Null Sets». En: *Journal of mathematical analysis and applications* 159 (1990), págs. 532-549 (vid. pág. 18).

[24] N. Otsu. «A threshold selection method from gray-level histograms». En: *IEEE Trans. Systems, Man, and Cybernetics* 9.1 (1979), págs. 62-66 (vid. págs. 28, 81).

[25] D. Paternain y col. «Construction of image reduction operators using averaging aggregation functions». En: *Fuzzy sets and systems* 261 (2015), págs. 87-111 (vid. págs. 16, 17).

[26] R. Penrose. *La nueva mente del emperador*. Mondadori, 1991 (vid. pág. 1).

[27] P. Sahoo, C. Wilkins y J. Yeager. «Threhold selection using Renyi's entropy». En: *Pattern recognition* 30.1 (1997), págs. 71-84 (vid. págs. 29, 82).

[28] J. R. Searle. «¿Es la mente un programa informático?» En: *Investigación y ciencia* 162 (mar. de 1990), págs. 10-17 (vid. pág. 1).

[29] M. Sonka, V. Halvac y R. Boyle. *Image processing, analysis and machine vision*. Thomson, 2008 (vid. págs. 2, 5).

[30] J. S. Suri, S. K. Setarehdan y S. Singh. *Advanced algorithmic approaches to medical image segmentation*. Springer, 2002 (vid. pág. 5).

[31] E. Trillas. «Sobre las funciones de negación en la teoría de conjuntos difusos». En: *Stochastica* III.1 (1979), págs. 47-59 (vid. pág. 15).

[32] A. M. Turing y col. *Controversia sobre mentes y máquinas*. Ed. por A. Ross Anderson. Ediciones Orbis, 1985 (vid. pág. 1).

[33] Alan Watt y Fabio Policarpo. *The computer image*. Addison-Wesley, 1998.

[34] R. R. Yager. «Families of OWA operators». En: *Fuzzy sets and systems* 59 (1993), págs. 125-148 (vid. pág. 17).

[35] L. A. Zadeh. «Fuzzy sets». En: *Inform. Control 8* (1965), págs. 338-353 (vid. pág. 13).