

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

# Enhanced Local Adaptive Binarization.



Grado en Ingeniería Informática

Trabajo Fin de Grado

Juan Uriarte Barragán

Director: Francisco Javier Fernández Fernández

Codirector: Javier Fumanal Idocin

Pamplona, 09/06/2021

upna

Universidad Pública de Navarra  
Nafarroako Unibertsitate Publikoa



## **Agradecimientos**

En primer lugar, me gustaría agradecer a los profesores de Ingeniería Informática de la Universidad Pública de Navarra por formarme a lo largo de estos cuatro años de carrera. En especial agradecer a mis tutores Francisco Javier Fernández Fernández y Javier Fumal Idocin, por ayudarme a solucionar los problemas que han surgido y guiarme durante este trabajo. No quiero olvidarme de agradecer al profesor Humberto Bustince Sola por su asesoramiento a la hora de elegir el TFG.

También quiero dar las gracias a todos mis amigos y familiares por su apoyo diario en este camino.



**Resumen:**

La umbralización de una imagen es un problema muy conocido en el que se realiza una segmentación solo con dos categorías: claro y oscuro. Cada píxel se clasifica como claro u oscuro; comparando su intensidad con una intensidad de referencia dada denominada umbral. Los algoritmos de umbralización adaptativa funcionan teniendo en cuenta los píxeles adyacentes al píxel a umbralizar por lo que limitan la intensidad de los píxeles con respecto a los píxeles adyacentes que explotan las imágenes integrales. A su vez, las imágenes integrales generalmente se calculan de manera óptima utilizando el algoritmo de tabla de áreas sumadas (SAT). Conociendo la metodología denominada como FLAT (Fuzzy Local Adaptive Thresholding), que es una técnica de binarización adaptativa basada en imágenes integrales difusas mediante un diseño eficiente de un SAT modificado para integrales difusas, en este trabajo proponemos distintos experimentos en cuanto a la configuración de las variables, las funciones de agregación utilizadas y el uso de umbral adaptativo y toma de decisión para obtener unos resultados óptimos. Los resultados experimentales muestran que nuestro algoritmo Enhanced Local Adaptive Binarization (ELAB), gracias a las metodologías propuestas, obtiene una mejora en la precisión de la binarización con respecto a los algoritmos tradicionales y al método FLAT.

**Palabras claves:** Umbral Adaptativo, Imagen Integral, Funciones De Agregación, Binarización, Toma De Decisión, Procesamiento de Imágenes.



---

# Índice de contenidos

---

<b>1. Introducción</b>	<b>1</b>
<b>2. Preliminares</b>	<b>3</b>
2.1. Funciones de agregación . . . . .	3
2.1.1. T-normas . . . . .	3
2.1.2. T-conormas . . . . .	4
2.1.3. Integrales difusas . . . . .	4
2.1.4. Integral de Sugeno . . . . .	5
2.1.5. F-Sugeno . . . . .	5
2.1.6. Integral Choquet . . . . .	5
2.1.7. CF . . . . .	5
2.1.8. CF12 . . . . .	6
2.1.9. Overlaps . . . . .	6
2.1.10. OWAs . . . . .	6
2.2. Algoritmo OTSU . . . . .	7
2.3. Adaptative Thresholding Using Integral Image . . . . .	8
2.4. Adaptive binarization based on fuzzy integrals . . . . .	9
2.4.1. Cálculo de la imagen integral con SAT . . . . .	9
2.4.2. Cálculo de la integral difusa en la imagen integral . . . . .	10
2.4.3. Aplicación del algoritmo de ATUII a la imagen integral . . . . .	12
2.4.4. Binarización adaptativa con con la imagen integral . . . . .	12
<b>3. Algoritmo de umbralización: Enhanced Local Adaptive Binarization</b>	<b>15</b>
3.1. Parámetros estudiados para el algoritmo ELAB . . . . .	15
3.2. Combinación de múltiples soluciones para el algoritmo ELAB . . . . .	16
3.3. Aplicaciones del umbral dependiente de ventana en el algoritmo ELAB . . . . .	16
<b>4. Experimentación</b>	<b>19</b>
4.1. Datos . . . . .	19
4.2. Criterios de evaluación . . . . .	19
4.3. Estudio de umbrales y tamaños de ventanas . . . . .	20
4.4. Resultados con distintas funciones de agregación . . . . .	22
4.5. Resultados con toma de decisión . . . . .	23
4.6. Resultados con umbral dinámico . . . . .	25
4.7. Comparación con otros métodos . . . . .	27
<b>5. Resultados</b>	<b>29</b>
5.1. Conclusiones y líneas futuras . . . . .	29

**Bibliografía**

**31**





# CAPÍTULO 1

---

## Introducción

---

La binarización es un problema clásico en el tratamiento de imagen, que ha sido utilizado en aplicaciones donde se requiere rendimiento en tiempo real y una representación estructural y semántica simple que obtenga una gran precisión. En la literatura, se proponen varios algoritmos de binarización de imágenes basados en modelos de redes tradicionales y neuronales para diferentes problemas de aplicación [1] que proporcionan una revisión ampliada de las metodologías tradicionales basadas en métodos de binarización local y global para la compresión de hologramas, [2] que presentan una comparación entre las técnicas de umbral para la segmentación de imágenes de resonancia magnética del mundo real y del cerebro.

Además, [3] proveen un estudio comparativo de las técnicas adaptativas más comunes. Recientemente, se adoptaron modelos basados en redes convolucionales para binarización y más. En particular, una de las evoluciones naturales de los enfoques de binarización se basa en el estudio de la percepción visual, mejor definida como prominencia visual, y en la capacidad de distinguir y mantener impreso un objeto, una persona o un grupo de píxeles en los que el ser humano presta atención. En varios niveles, los enfoques tradicionales están integrados en modelos deep learning que muestran un justo equilibrio entre precisión, poder de generalización y costos de tiempo computacional. La binarización en redes complejas [4], redes bayesianas [5] y redes / vías biológicas [6]. Además, los modelos que combinan las técnicas locales y globales procesan la misma imagen varias veces y, por culpa de esto, acaban obteniendo una falta de rendimiento [7], [8].

Existen muchos métodos de optimización en la literatura. Generalmente, los algoritmos de binarización local son mejores que los globales. En particular, el método de umbral adaptativo de [8] (conocido como algoritmo de Adaptive Thresholding Using Integral Image) explota el poder de representación de las imágenes integrales. Sin embargo, [9] las integrales difusas, en el contexto de la adaptabilidad local, muestran un desempeño superior a las metodologías basadas en imágenes integrales simples. En este trabajo, hemos realizado modificaciones del algoritmo [10] y centramos la atención tanto en las funciones de agregación, como en los métodos de umbral adaptativos locales. En general, estos últimos son más precisos que los globales y podrían ajustarse de forma automática. Asimismo, las perturbaciones que pueden afectar a una imagen son heterogéneas (cambios de iluminación, ruido experimental, contraste variable, etc.) y dependen de los sujetos representados. En varios niveles de procesamiento digital, como por ejemplo en el muestreo por compresión y la compresión con pérdida, los diferentes tipos y grados de degradación

digital podrían influir en la precisión de la binarización. Los objetos analizados en las imágenes pueden ser estáticos o en movimiento, múltiples o únicos. La binarización tradicional o basada en redes neuronales del mundo real [11] podría usarse para establecer relaciones entre marcos [12].

La idea, detrás del umbral adaptativo local, se basa en considerar un valor de umbral para cada intensidad de píxel o región de intensidades de píxel basando el análisis en sus píxeles vecinos en una ventana local fija o variable. Después de todo, la noción de adaptación tiene sus raíces en el concepto de análisis multiescala, análisis variacional estructural y representación de valores de intensidad diferencial. Uno de los métodos de umbral adaptativo local [8] más eficientes proviene de una extensión del método [13] y es una forma generalizada del algoritmo [14].

En este trabajo nos basamos en una modificación del algoritmo [8], en concreto en el algoritmo definido en el artículo [10]. En la literatura ya ha habido intentos de modificar el algoritmo [8]. En particular, en estos casos, se utiliza una modificación en el cálculo de las intensidades medias de los píxeles vecinos, por ejemplo, considerando una imagen integral ponderada [15]. En el caso del algoritmo [10], proponen un LAT novedoso, y lo definen como FLAT, que es el acrónimo del algoritmo Fuzzy Local Adaptive Thresholding FLAT, se basa en la lógica del algoritmo [8]. En particular, FLAT mejora la precisión del umbral aprovechando una forma generalizada de las imágenes integrales difusas. Las imágenes integrales difusas se calculan a partir de las imágenes integrales con un nuevo algoritmo eficiente basado en una modificación del algoritmo de tabla de áreas sumadas (SAT) [16] que muestra rendimientos en tiempo real. Nuestras modificaciones se basan en la prueba de distintas funciones de agregación en el FLAT para, de esta forma, maximizar el rendimiento y obtener una mayor precisión a la hora de clasificar los píxeles como fondo u objeto.

El documento está organizado de la siguiente manera: las funciones de agregación en la sección 2.1 con una subsección explicando brevemente algunas propiedades de las T-normas, T-conormas, integrales difusas, overlaps y OWAs; los aspectos teóricos de la umbralización y la explicación del algoritmo de OTSU 2.2; la explicación de las imágenes integrales y el algoritmo Adaptive Thresholding Using Integral Image (ATUII) en la sección 2.3. En la sección [FLAT] se explica el algoritmo FLAT de forma teórica. Sin embargo, en la sección 3 se presentan los cambios al algoritmo FLAT: parámetros analizados en la sección 3.1 y los distintos algoritmos probados como son la toma de decisión y el umbral dependiente de ventana en las secciones 3.2 y 3.3 respectivamente. En la sección 4, se explica el dataset utilizado y la función con la que obtendremos los resultados. Además, se muestran los resultados producidos por los algoritmos utilizados con tablas y gráficas y se realiza una comparación entre los distintos algoritmos. Finalmente, en la sección 5, mostramos los resultados y conclusiones obtenidas viendo unos resultados muy precisos y rendimientos óptimos. Vemos que obtenemos una mayor capacidad de binarización con las modificaciones y funciones probadas que en algoritmo [10].

# CAPÍTULO 2

---

## Preliminares

---

En esta sección vamos a repasar algunos conceptos referidos a funciones de agregación y algoritmos de umbralización de imagen.

### 2.1. Funciones de agregación

Las funciones de agregación se utilizan para fusionar información de  $n$  fuentes en una sola salida. Una función  $A: [0, 1]^n \rightarrow [0, 1]$  se dice que es una función de agregación  $n$ -aria si cumple las siguientes condiciones [17]:

- $A$  es creciente en cada argumento:  $\forall i \in \{1, \dots, n\}$ , algunos  $x_i < y$ ,  $A(x_1, \dots, x_i, \dots, x_n) \leq A(x_1, \dots, y, \dots, x_n)$
- $A(0, \dots, 0) = 0$
- $A(1, \dots, 1) = 1$

Algunos ejemplos de funciones clásicas de agregación  $n$ -aria son:

- Media aritmética:  $A(x) = \frac{1}{n} \sum_{i=1}^n x_i$
- Máximo:  $A(x) = \max(x_1, \dots, x_n)$
- Mínimo:  $A(x) = \min(x_1, \dots, x_n)$

#### 2.1.1. T-normas

Las T-Normas son funciones de agregación que para todo  $x, y, z \in [0, 1]$ , satisface las siguientes propiedades [18]:

- Conmutativa:  $T(x, y) = T(y, x)$
- Asociativa:  $T(x, T(y, z)) = T(T(x, y), z)$
- Creciente en cada argumento.
- Condición limite:  $T(x, 1) = 1$

Algunos ejemplos de T-normas son:

- Lukasiewicz:  $TL(x, y) = \max(x + y - 1, 0)$

- Drastic:

$$T_D(x, y) = \begin{cases} y & \text{if } x = 1 \\ x & \text{if } y = 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

- Nilpotent:

$$T_N(x, y) = \begin{cases} \min(x, y) & \text{if } x + y > 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

- Hamacher

$$T_H(x, y) = \begin{cases} 0 & \text{if } x = y = 0 \\ \frac{xy}{x+y-xy} & \text{otherwise} \end{cases} \quad (2.3)$$

### 2.1.2. T-conormas

Una T-conorma es una operación binaria  $S : [0, 1]^2 \rightarrow [0, 1]$ , que cumple para todo  $x, y, z \in [0, 1]$ , satisface las siguientes propiedades [18]:

- Conmutativa:  $T(x, y) = T(y, x)$
- Asociativa:  $T(x, T(y, z)) = T(T(x, y), z)$
- Creciente en cada argumento.
- Elemento neutro 0:  $T(x, 0) = x$ .

Algunos ejemplos de T-conormas son:

- Bounded sum:  $S(x, y) = \min(x + y, 1)$

- Drastic t-conorm:

$$S_D(x, y) = \begin{cases} y & \text{if } x = 0 \\ x & \text{if } y = 0 \\ 1 & \text{otherwise} \end{cases} \quad (2.4)$$

- Nilpotent maximum:

$$S_N(x, y) = \begin{cases} \max(x, y) & \text{if } x + y < 1 \\ 1 & \text{otherwise} \end{cases} \quad (2.5)$$

- Einstein sum:  $S_E(x, y) = \frac{x+y}{1+xy}$

### 2.1.3. Integrales difusas

Dado  $n \in \mathbb{N}$ ,  $[n] = \{1, \dots, n\}$ . Una función  $\mu: 2^{[n]} \rightarrow [0, 1]$  es una medida difusa, si satisface las condiciones [19]:

- $\mu(A) \leq \mu(B)$  cuando  $A \subseteq B$ ,
- $\mu(\emptyset) = 0$ ,  $\mu([n]) = 1$ .

Una medida difusa  $\mu$  es simétrica, si para cualquier  $A, B \subseteq [n]$ ,  $|A| = |B|$  implica  $\mu(A) = \mu(B)$  (aquí  $|E|$  representa la cardinalidad del conjunto  $E$ ). La medida difusa uniforme  $\mu_{uni}$  viene dada por

$$\mu_{uni}(E) = \frac{|E|}{n}, \quad (2.6)$$

para  $E \subseteq [n]$ , es simétrica.

Las Integrales difusas son un tipo de integrales con respecto a una medida difusa.

### 2.1.4. Integral de Sugeno

La integral de Sugeno es una función tal que para un vector  $X$  y una medida difusa  $\mu$

$$Su_{\mu}(X) = \bigvee_{i=1}^n (X_{(i)} \wedge \mu(E_{(i)})) \quad (2.7)$$

### 2.1.5. F-Sugeno

Remplazando los operadores máximo y mínimo por otras operaciones mas generales obtenemos la F-sugeno. Para un vector  $X$  y una medida difusa  $\mu$ ,  $F: [0, \infty[ \times [0, 1] \rightarrow [0, \infty[$  una función binaria y  $G: [0, \infty[^n \rightarrow [0, \infty[$  una función  $n$ -aria. La función F-sugeno viene dada por [10]:

$$A(X) = G(F(x_{(1)}, \mu(E_{(1)})), \dots, F(x_{(n)}, \mu(E_{(n)}))) \quad (2.8)$$

### 2.1.6. Integral Choquet

Mientras que la integral Sugeno se basa en operadores no lineales, mínimo y máximo, la integral de Choquet se basa en operadores lineales, suma y producto, en la función 2.9 la definimos.

$$Ch_{\mu}(X) = \sum_{i=1}^n (X_i - X_{(i-1)}) * \mu(E_{(i)}) \quad (2.9)$$

### 2.1.7. CF

Dada una función  $F: [0, 1]^2 \rightarrow [0, 1]$  y la medida difusa  $\mu$  obtenemos la función CF 2.10 [20]

$$C_{\mu}^F(X) = \sum_{i=1}^n F(X(x_{(i)}) - X(x_{(i-1)})) * \mu(A_{(i)}) \quad (2.10)$$

### 2.1.8. CF12

Otra generalización de la integral Choquet es la CF12 2.11 [20]

$$C_{\mu}^{min,min}(X) = \sum_{i=1}^n \min(X(x_{(i)}), \mu(A_{(i)})) - \min(X(x_{(i-1)}), \mu(A_{(i)})) \quad (2.11)$$

### 2.1.9. Overlaps

Una función  $O: [0, 1]^2 \rightarrow [0, 1]$  es un Overlap si satisface las condiciones [21]:

- $O$  es conmutativa.
- $O(x, y) = 0$  si y solo si  $xy = 0$ .
- $O(x, y) = 1$  si y solo si  $xy = 1$ .
- $O$  es creciente.
- $O$  es continua.

Algunos ejemplos de Overlaps para el vector  $X$  de tamaño  $n$  son:

- Media geométrica:  $\sqrt[n]{\prod_{i=1}^n X_i}$
- Media armónica:  $\frac{n}{\sum_{i=1}^n \frac{1}{X_i}}$
- Overlap seno:  $\sin(\frac{\pi}{2} * \prod_{i=1}^n X^{1/2n})$

### 2.1.10. OWAs

Un operador OWA (2.12) de dimensión  $n$  es un mapeo:  $R^n \rightarrow [R]$ , que tiene un vector de pesos asociado  $w = (w_1, w_2, \dots, w_n)$  tal que  $w_j \in [0, 1]$  y  $\sum_{j=1}^n (w_j) = 1$ .

$$OWA_w(a_1, a_2, \dots, a_n) = \sum_{j=1}^n (w_j b_j) \quad (2.12)$$

donde  $b_j$  is es el  $j$ -ésimo elemento más grande de la colección de objetos agregados  $a_1, a_2, \dots, a_n$ .

Un de los aspectos más importantes del operador OWA es determinar sus pesos asociados. [22] Un enfoque interesante para obtener las ponderaciones OWA mediante el uso de cuantificadores lingüísticos, que, en el caso de un cuantificador lingüístico difuso proporcional  $Q$  no decreciente, viene dado por la función 2.13.

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right), i = 1, 2, 3, \dots, n \quad (2.13)$$

[23] define  $Q$  como en la formula 2.14.

$$Q(r) = \begin{cases} 0 & \text{if } r < a \\ \frac{r-a}{b-a} & \text{if } a \leq r \leq b \\ 1 & \text{if } r > b \end{cases} \quad (2.14)$$

Según que valores le asignemos a ‘a’ y ‘b’ obtenemos distintos operadores OWAs:

- OWA1: Parámetros (a=0,b=0.5).
- OWA2: Parámetros (a=0.5,b=1).
- OWA3: Parámetros (a=0.3,b=0.8).

## 2.2. Algoritmo OTSU

La segmentación o binarización de imágenes es un proceso fundamental dentro de las aplicaciones de procesamiento de imagen y visión artificial, tanto para compresión de imagen como para estudiar su contenido. Se basa en la partición de una imagen en regiones separadas que corresponden o equivalen a representaciones de diferentes objetos del mundo real.

La umbralización consiste en dividir los datos de una imagen en regiones basándose en los valores de intensidad y las propiedades de estos valores.

El método de Otsu [24], es un método que utiliza un análisis discriminante donde se evalúa la precisión de varios umbrales para seleccionar el umbral óptimo. Este algoritmo selecciona los umbrales maximizando la varianza entre clases, mediante una búsqueda que permite maximizar la distancia entre estas.

Partimos de una imagen en niveles de gris con N píxeles y L posibles niveles diferentes.

Calculamos la probabilidad de ocurrencia del nivel de gris i en la imagen con la función

$$p_i = \frac{f_i}{N} \quad (2.15)$$

siendo  $f_i$  la frecuencia de repetición del nivel de gris i-ésimo con  $i = 1, 2, \dots, L$ .

En el caso particular de la binarización, los píxeles se dividen en dos clases C1 y C2, con niveles de gris  $[1, 2, \dots, t]$  y  $[t+1, t+2, \dots, L]$  respectivamente, donde las distribuciones de probabilidad de ambas clases se calculan con las funciones 2.16 y 2.17

$$C1 = \frac{p_1}{w_1(t)}, \dots, \frac{p_t}{w_1(t)} \quad (2.16)$$

$$C2 = \frac{p_{t+1}}{w_2(t)}, \dots, \frac{p_L}{w_2(t)} \quad (2.17)$$

donde  $w_1$  y  $w_2$  están definidos por las ecuaciones 2.18 y 2.19

$$w_1(t) = \sum_{i=1}^t p_i \quad (2.18)$$

$$w_2(t) = \sum_{i=t+1}^L p_i \quad (2.19)$$

Las medias para cada una de las clases se definen en 2.20 y 2.21

$$\mu_1 = \sum_{i=1}^t \frac{i * p_i}{w_1(t)} \quad (2.20)$$

$$\mu_2 = \sum_{i=t+1}^L \frac{i * p_i}{w_2(t)} \quad (2.21)$$

Haciendo uso de un análisis discriminante, Otsu define la varianza entre clases de una imagen umbralizada como 2.22

$$\sigma_B^2 = w_1(\mu_1 - \mu_t)^2 + w_2(\mu_2 - \mu_t)^2 \quad (2.22)$$

Y por ultimo buscamos el umbral, t, que maximice la varianza (Otsu demostró que este era el umbral óptimo) y eso lo hacemos con 2.23

$$t = \underset{t}{\text{máx}} [\sigma_B^2(t)] \quad \text{donde } 1 \leq t \leq L \quad (2.23)$$

### 2.3. Adaptive Thresholding Using Integral Image

La imagen integral es una técnica para acelerar el cálculo de operaciones que incluyan la suma del valor de los píxeles de un área. La imagen integral contiene en cada píxel la suma de todos píxeles contenidos en un rectángulo cuya esquina superior izquierda es el vértice 0,0 de la imagen, y cuya esquina inferior derecha es el propio píxel.

ATUII [8] obtiene la imagen integral de la imagen original y posteriormente le aplica un umbral. Gracias a esto, conseguimos una técnica más resistente a los cambios de iluminación.

Para calcular la imagen integral almacenamos en cada ubicación,  $I(x,y)$ , la suma de todos los términos  $f(x,y)$  a la izquierda y arriba del píxel  $(x,y)$ . Esto se logra en tiempo lineal utilizando la ecuación (2.24) para cada píxel (teniendo en cuenta los casos fronterizos)

$$I(x, y) = f(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1) \quad (2.24)$$

Para clasificar los píxeles los comparamos con una media de los píxeles que los rodean. En concreto, con el promedio de una ventana  $s \times s$  de píxeles centrada alrededor de cada píxel a estudiar. Para calcular este promedio en tiempo lineal usamos la imagen integral.

A continuación mostramos el pseudocódigo de Adaptive Threshold (1)

---

**Algoritmo 1** AdaptiveThreshold

---

```

1: procedure ADAPTATIVE_THRESHOLD(in, out, w, h) ▷ Imagen de entrada, imagen de
   salida, ancho y alto de la imagen
2:   for i = 0 to w do
3:     sum ← 0
4:     for j = 0 to h do
5:       sum ← sum + in[i, j]
6:       if i = 0 then
7:         intImg[i, j] ← sum
8:       else
9:         intImg[i, j] ← sum + intImg[i - 1, j]
10:      end if
11:    end for
12:  end for
13:  for i = 0 to w do
14:    for j = 0 to h do
15:      x1 ← i - s/2
16:      x2 ← i + s/2
17:      y1 ← j - s/2
18:      y2 ← j + s/2
19:      count ← (x2 - x1) * (y2 - y1)
20:      sum ← intImg[x2, y2] - intImg[x2, y1 - 1] - intImg[x1 - 1, y2] +
        intImg[x1 - 1, y1 - 1]
21:      if (in[i, j] * count) ≤ (sum * (100 - t)/100) then
22:        out[i, j] ← 0
23:      else
24:        out[i, j] ← 255
25:      end if
26:    end for
27:  end for
28:  return out
29: end procedure

```

---

## 2.4. Adaptive binarization based on fuzzy integrals

### 2.4.1. Cálculo de la imagen integral con SAT

Dado  $n, m \in \mathbb{N}$ ,  $[n] = \{1, \dots, n\}$ ,  $[m] = \{1, \dots, m\}$  y la imagen original  $I$  de  $n \times m$  píxeles y siendo  $p(x, y)$  la intensidad del píxel  $(x, y) \in [n] \times [m]$ .

Aprovechando el algoritmo SAT, el cálculo de  $S(x, y)$  puede mantenerse constante y la complejidad del tiempo SAT permanece fija a  $O(n \cdot m)$ . El SAT se puede desarrollar de manera eficiente computando para cada píxel  $(x, y)$  las sumas de prefijo en columna y las sumas de prefijo en fila (2.25)

$$S(x, y) = p(x, y) + S(x, y - 1) + S(x - 1, y) - S(x - 1, y - 1) \quad (2.25)$$

con convención  $S(0, k) = 0$ , para cada  $k = 0, \dots, m$  y  $S(l, 0) = 0$ , para cada  $l = 0, \dots, n$ .

### 2.4.2. Cálculo de la imagen difusa en la imagen integral

Las integrales difusas se utilizan para tener en cuenta la correlación y las interacciones de los datos. La principal desventaja de las integrales difusas, como de las integrales de Choquet, radica en asignar más esfuerzo computacional a la clasificación de elementos. A pesar de esta última observación y mirando de cerca la construcción en cascada de una imagen integral, en un tiempo de clasificación constante, es posible adoptar el procedimiento aplicado en el SAT y generar de manera óptima la imagen integral difusa  $F$ . Una vez que se calcula la imagen integral  $S$ , para cada píxel  $(x, y)$  en la ventana operativa, es posible calcular la imagen integral difusa  $F_a$  (2.26)

$$F_{A_i}(x, y) = A_i(S(x, y), S(x, y - 1), S(x - 1, y), S(x - 1, y - 1)) \quad (2.26)$$

donde  $A_i : [0, \infty]^4 \rightarrow [0, \infty]$  for  $i = 1, 2, 3, 4$  es uno de las funciones basadas en integrales difusas (la integral de Sugeno, Sugeno-like FG-funcionales y la integral Choquet). Como medida difusa adoptamos la medida difusa uniforme  $\mu_{uni}$ .

El procedimiento aprovecha el ordenamiento natural de los cuatro elementos agregados, obteniendo un vector de valores ordenados  $\vec{\sigma}$  y un vector estático asociado de medidas difusas  $\vec{m}$ .

De hecho, el valor máximo es el elemento presente en la esquina inferior derecha  $S(x, y)$  de la ventana operativa (es por esto que el algoritmo de Adaptive binarization based on fuzzy integrals con la función máximo devuelve la misma imagen integral que obtenemos en Adaptive binarization based on fuzzy integrals), mientras que el valor mínimo es el elemento  $S(x-1, y-1)$ .

Pseudocódigo de cálculo de  $F_{A_i}$  (2)

---

**Algoritmo 2** : Computation of  $F_{A_i}$ . (*FLAT* - Step 1)

---

**Require:** Gray-scale image  $I$  with intensities in  $[0, 1]$ .

**function** FLAT-  $F_{A_i}$  ( $I$ )

$n, m \leftarrow \dim(I)$

▷ Dimension of  $I$

$S \leftarrow$  allocate a zero-matrix with size  $(n, m)$

$F_{A_i} \leftarrow$  allocate a zero-matrix with size  $(n, m)$

$\vec{m}_1 \leftarrow [1, 0, 75, 0, 50, 0, 25]$ ;  $\vec{m}_2 \leftarrow [1, 0, 50]$

**for**  $r \leftarrow 1$  to  $n$  **do**

**for**  $c \leftarrow 1$  to  $m$  **do**

**if**  $r \neq 1 \wedge c \neq 1$  **then**

$v_1 \leftarrow S[r-1, c-1]$ ;  $s_1 \leftarrow S[r, c-1]$ ;  $s_2 \leftarrow S[r-1, c]$ ;

$S[r, c] \leftarrow I[r, c] + s_1 + s_2 - v_1$ ;  $v_4 \leftarrow S[r, c]$ ;

$v_2, v_3 \leftarrow P_{\text{swap}}(s_1, s_2)$ ; - Procedure 2.30

$v_0 \leftarrow 0$ ;  $\vec{o}v \leftarrow [v_0, v_1, v_2, v_3, v_4]$ ;

$F_{A_i}[r, c] \leftarrow f_i(\vec{o}v, m_1)$ ;

▷  $i = 1, 2, 3, 4$

**else if**  $r \neq 1$  **then**

$v_1 \leftarrow S[r-1, c]$ ;  $S[r, c] \leftarrow I[r, c] + v_1$ ;

$v_4 \leftarrow S[r, c]$ ;  $\vec{o}v \leftarrow [v_0, v_1, v_4]$ ;

$F_{A_i}[r, c] \leftarrow f_i(\vec{o}v, m_2)$ ;

▷  $i = 1, 2, 3, 4$

**else if**  $c \neq 1$  **then**

$v_1 \leftarrow S[r, c-1]$ ;  $S[r, c] \leftarrow I[r, c] + v_1$ ;

$v_4 \leftarrow S[r, c]$ ;  $\vec{o}v \leftarrow [v_0, v_1, v_4]$ ;

$F_{A_i}[r, c] \leftarrow f_i(\vec{o}v, m_2)$ ;

▷  $i = 1, 2, 3, 4$

**else**

$S[r, c] \leftarrow I[r, c]$ ;  $F_{A_i}[r, c] \leftarrow S[r, c]$ ;

**end if**

**end for**

**end for**

**return**  $F_{A_i}$

**end function**

---

### 2.4.3. Aplicación del algoritmo de ATUII a la imagen integral

Una vez obtenida la imagen integral  $S$ , se debe umbralizar esta imagen para obtener la imagen final. El algoritmo se puede ver en la figura 2.

Para esto calculamos la suma de las intensidades de los píxeles de las ventanas (2.27)

$$p_s(x_1, y_1, x_2, y_2) = S(x_2, y_2) - S(x_2, y_1) - S(x_1, y_2) + S(x_1, y_1) \quad (2.27)$$

donde  $1 \leq x_1 \leq x_2 \leq m, 1 \leq y_1 \leq y_2 \leq n$ .

Se calcula la media de las intensidades de los píxeles de la ventana (2.28)

$$p_a(x_1, y_1, x_2, y_2) = \frac{p_s(x_1, y_1, x_2, y_2)}{(x_2 - x_1) * (y_2 - y_1)} \quad (2.28)$$

donde  $1 \leq x_1 \leq x_2 \leq m, 1 \leq y_1 \leq y_2 \leq n$ .

En la segunda parte del proceso, la intensidad de píxel de la imagen original  $I$  se compara píxel a píxel con el valor medio de las intensidades de píxel en la ventana local alrededor del píxel actual. El algoritmo de ATUII binariza iterativamente la imagen original y proporciona los valores binarios  $I_b(x, y)$  por cada píxel  $(x, y)$ , como se describe en la siguiente fórmula (2.29)

$$I_b(x, y) = \begin{cases} 1 & \text{if } p(x, y) \leq p_a(x_1, y_1, x_2, y_2) * t \\ 0 & \text{otro caso} \end{cases} \quad (2.29)$$

donde  $(x_1, y_1, x_2, y_2) = (x - s, y - s, x + s, y + s)$ .

Para completar la clasificación, necesitamos intercambiar  $s1=S(x,y-1)$  y  $s2=S(x-1,y)$  con la operación de intercambio (2.30)

$$P_{swap}(s1, s2) = \begin{cases} v_2 = s_1, v_3 = s_2 & \text{if } s_1 < s_2 \\ v_2 = s_2, v_3 = s_1 & \text{otro caso} \end{cases} \quad (2.30)$$

### 2.4.4. Binarización adaptativa con con la imagen integral

Versión modificada del algoritmo de ATUII de acuerdo con nuestras restricciones. El tamaño y las coordenadas de la ventana local de píxeles del vecino más cercano deslizando se establece con el parámetro dimensional  $n_a$ . Este último se calcula a través de 2 parámetros empíricos:  $a_1$  y  $a_2$ . Se utiliza para binarizar localmente los píxeles centrales de las ventanas. El área de la ventana es igual  $n_a^2$ . El parámetro dimensional se calcula con la ecuación (2.31)

$$n_a = \frac{\min(n, m)}{a_1 \times a_2}. \quad (2.31)$$

siendo  $n$  y  $m$  las dimensiones de la imagen  $I$ . Los parámetros  $a_1, a_2$  y  $t$  se puede variar iterativamente para mejorar la precisión.

---

**Algoritmo 3** : Binarization based on  $F_{A_i}$  (*FLAT* - Step 2)

---

**Require:** Gray-scale image  $I$  with intensities in  $[0, 1]$ .

**Require:** The fuzzy integral image  $F_{A_i}$ .

**Require:** The parameters  $a_1$  and  $a_2$ .

**Require:** The sensitivity parameter  $t$

**function** *Flat* –  $I_b(I, F_{A_i}, a_1, a_2, t)$

$n, m \leftarrow \dim(I)$

▷ Dimension of  $I$

$I_b \leftarrow$  allocate a zero-matrix with size  $(n, m)$

$n_a \leftarrow$  Defined in Formula 2.31 with  $a_1$  and  $a_2$

**for**  $r \leftarrow 1$  to  $n$  **do**

**for**  $c \leftarrow 1$  to  $m$  **do**

$y_0 \leftarrow \max(r - n_a, 0)$

▷ Set the  $w_n$

$y_1 \leftarrow \min(r + n_a, r)$

$x_0 \leftarrow \max(c - n_a, 0)$

$x_1 \leftarrow \max(c + n_a, c)$

$p_{area} \leftarrow (y_1 - y_0) * (x_1 - x_0)$

$p_s \leftarrow F_{A_i}[y_1, x_1] - F_{A_i}[y_0, x_1] - F_{A_i}[y_1, x_0] + F_{A_i}[y_0, x_0]$

$p_a \leftarrow \frac{p_s}{p_{area}}$

**if**  $I[r, c] \leq p_a \times (1 - t)$  **then**

$I_b[r, c] \leftarrow 1$

**else**

$I_b[r, c] \leftarrow 0$

**end if**

**end for**

**end for**

**return**  $I_b$

**end function**

---



## CAPÍTULO 3

---

### Algoritmo de umbralización: Enhanced Local Adaptive Binarization

---

En esta sección desarrollamos nuestro nuevo algoritmo: Enhanced Local Adaptive Binarization (ELAB). El algoritmo ELAB consiste en el algoritmo FLAT utilizando distintas funciones de agregación para sustituir la F-Sugeno utilizada en el original. Concretamente, hemos probado las funciones ya descritas en la sección 2.1.

Lo que queremos conseguir, con todas estas funciones de agregación, es mejorar los resultados obtenidos con este algoritmo respecto al algoritmo de Adaptive Thresholding Using Integral Image. Posteriormente aplicamos la toma de decisión y el umbral dinámico

#### 3.1. Parámetros estudiados para el algoritmo ELAB

ELAB tiene distintos parámetros que son el tamaño de la ventana ( $a_1$  y  $a_2$ ) y el umbral ( $t$ ). En la tabla 3.1 mostramos los distintos valores estudiados para el tamaño de la ventana y los valores del umbral van de 0,05 hasta 0,9 en intervalos de 0,05.

Tamaño de la ventana	
2 x 2	3 x 1
3 x 3	2 x 3
5 x 5	3 x 5
7 x 7	5 x 7

Figura 3.1: Ventanas.

Con estos dos pasos lo que conseguimos es saber, para cada imagen y con cada función, cual es su mejor configuración del tamaño de la ventana y el umbral para obtener su umbralización óptima. Utilizamos estos dos parámetros ya que son las variables principales del método FLAT y le aplicamos esos valores puesto que en el artículo [10] mencionan que la ventana que les optimizaba los resultados era la 3 x 1 pero al utilizar las funciones descritas en la sección 2.1 queremos comprobar si con distintos parámetros obtenemos mejores resultados. No ponemos ventanas mayores a 7x7 porque conforme aumentábamos las ventanas empeoraban los resultados.

## 3.2. Combinación de múltiples soluciones para el algoritmo ELAB

En vez de coger solo una solución, otra posibilidad es calcular la umbralización con distintos métodos y luego combinarlos para ver como mejora.

Escogemos  $N$  funciones de agregación. Calculamos la ‘puntuación’ para cada píxel de la imagen con cada función de agregación. Con esto obtenemos  $N$  puntuaciones para cada píxel. Combinamos para cada píxel las  $N$  puntuaciones usando todas las funciones de agregación que hemos estudiado hasta ahora.

Elegimos un representante de cada función visto en la sección 2.1: Integrales difusas modernas (F-Sugeno y CF), clásicas (Media aritmética y máximo), una T-norma (Lukasiewicz) y un Overlap generalizado (media geométrica). La elección de las tres funciones se realiza al azar entre las seis elegidas.

Para elegir las  $N$  funciones de agregación decidimos coger tres de entre las siguientes seis funciones:

- F-sugeno.
- CF.
- Media arimética.
- Máximo.
- Lukasiwewicz.
- Media geométrica.

Una vez que obtenemos la imagen integral, calculándola con la toma de decisión, lo único que tenemos que hacer es umbralizarla.

## 3.3. Aplicaciones del umbral dependiente de ventana en el algoritmo ELAB

Otra posibilidad es utilizar un umbral distinto para cada ventana, lo que nos permite adecuar cada umbral a sus circunstancias. Esto se denomina Umbral adaptativo.

Para aprender el umbral, necesitamos un conjunto de datos de entrenamiento y unas etiquetas. En nuestro caso, el problema de aprender el mejor threshold es un problema de clasificación, y para ello utilizamos una regresión logística.

Esta implementación (4) necesita unos datos ‘ $X$ ’ y unas etiquetas ‘ $y$ ’. La ‘ $X$ ’ es cada ventana que generamos al umbralizar, las etiquetas ‘ $y$ ’ son 1 si es objeto y 0 si es fondo. Pasamos los valores de las imágenes a estas configuraciones. Para ello, lo que hacemos es que ‘ $X$ ’ sea el resultado de pasar la imagen integral a un vector de tamaño  $[n \times m, 9]$ , ya que estamos utilizando ventanas  $3 \times 3$ . Para la ‘ $y$ ’ lo que hacemos es pasar la imagen solución a un vector de dimensiones  $[n \times m]$ .

Para poder realizar el umbral adaptativo con todos los píxeles, añadimos un borde a la imagen de un píxel, porque la ventana es 3x3, por los cuatro lados con una réplica de los lados de la imagen original.

---

**Algoritmo 4** UmbralAdaptativo

---

```

1: procedure UMBRALADAPTATIVO(imgInt, imgSol, n, m, VX, VY)  ▷ Img integral,
   img solución, ancho y alto img integral, ancho y alto ventana
2:   imgBorde ← copiarBorde(imgInt, VY//2, VY//2, VX//2, VX//2, borderReplicate)
3:   X ← zeros[nxm, 9]
4:   y ← pasarAVector(imgSol)
5:   for r = 1 to n + 1 do
6:     for c = 1 to m + 1 do
7:       for k = 0 to VX do
8:         for z = 0 to VY do
9:           X[m * (r - 1) + (c - 1), 3 * k + z] ← imgBorde[r + k - 1, c + z - 1]
10:        end for
11:       end for
12:     end for
13:   end for
14:   X_train, X_test, y_train, y_test ← train_test_split(X, y, test_size = 0,2)
15:   clf ← sklearn.linear_model.LogisticRegression()
16:   clf.fit(X_train, y_train)
17:   y_pred ← clf.predict(X_test)
18:   return f1_score(y_pred, y_test)
19: end procedure

```

---



# CAPÍTULO 4

## Experimentación

### 4.1. Datos

Utilizamos el dataset del artículo de Adaptive binarization based on fuzzy integrals, compuesto por diez imágenes con sus respectivas imágenes umbralizadas, que se muestran en la figura 4.1 y 4.2 respectivamente. Las dimensiones rondan los 195-198 píxeles de ancho y 197-202 píxeles de alto. Trabajamos con imágenes con un rango de píxeles  $[0,1]$  dividiendo entre 255 las imágenes originales.

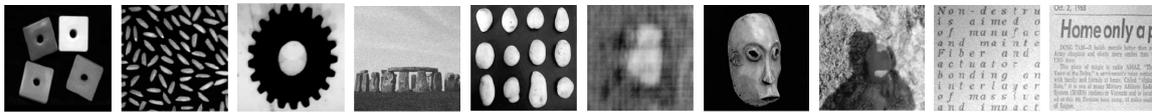


Figura 4.1: Imágenes originales del dataset.

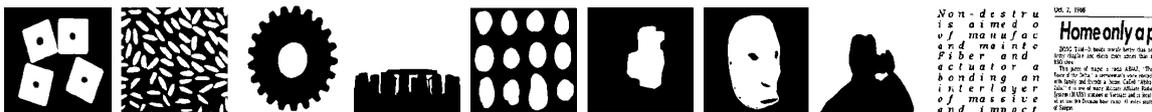


Figura 4.2: Imágenes solución del dataset.

Para obtener los resultados de la umbralización lo hacemos comparando con la imagen solución. Lo que hacemos es pasar nuestra imagen original a un rango de valores entre 0 y 1 dividiendo todos los píxeles entre 255.

Tras aplicarle alguno de los algoritmos obtenemos la imagen umbralizada. Esta imagen umbralizada la comparamos con la imagen solución que tenemos en el dataset para poder comprobar que porcentaje de acierto conseguimos.

### 4.2. Criterios de evaluación

Para conseguir el porcentaje de acierto utilizamos el F1 score. Para realizar este F1 score vamos comparando píxel a píxel la imagen que hemos obtenido nosotros con la imagen solución.

Una vez obtenido la cantidad de TP, FP, TN y FN calculamos la precisión (4.1) y el recall (4.2)

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (4.1)$$

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (4.2)$$

Y con estos dos datos podemos calcular la F1 score que se calcula haciendo la media armónica entre la precisión y el recall (4.3)

$$F1\_score = \frac{2 * precision * recall}{precision + recall} \quad (4.3)$$

---

**Algoritmo 5** F1Score

---

```

1: procedure F1SCORE(imgUmb, imgSol, n, m)    ▷ Img umbralizada, img solución,
   ancho y alto de la imagen
2:   TP ← TN ← FP ← FN ← 0
3:   for r = 0 to n do
4:     for c = 0 to m do
5:       if imgUmb[r, c] == 1 and imgSol[r, c] == 1 then
6:         TP ← TP + 1
7:       else
8:         if imgUmb[r, c] == 0 and imgSol[r, c] == 0 then
9:           TN ← TN + 1
10:        else
11:          if imgUmb[r, c] == 1 and imgSol[r, c] == 0 then
12:            FP ← FP + 1
13:          else
14:            FN ← FN + 1
15:          end if
16:        end if
17:      end if
18:    end for
19:  end for
20:  precision ← Defined in Formula 4.1
21:  recall ← Defined in Formula 4.2
22:  F1_score ← Defined in Formula 4.3
23:  return F1_score
24: end procedure

```

---

### 4.3. Estudio de umbrales y tamaños de ventanas

Como hemos comentado en el apartado 3.1, vamos modificando los valores  $a_1$  y  $a_2$  desde  $[1x3]$  a  $[7x7]$  y el parámetro  $t$  desde 0.05 a 0.9.

Primero realizamos el cálculo de F1 score para distintas ventanas, tal y como lo hacíamos anteriormente.

En la modificación del umbral, para cada par imagen-función calculamos el F1 score con cada umbral y hacemos un plot con todos los resultados. Observamos la figura 4.3 y vemos que obtenemos un F1 score distinto para cada umbral, en la mayoría de los casos, y nos quedamos con el que nos da el mejor valor.

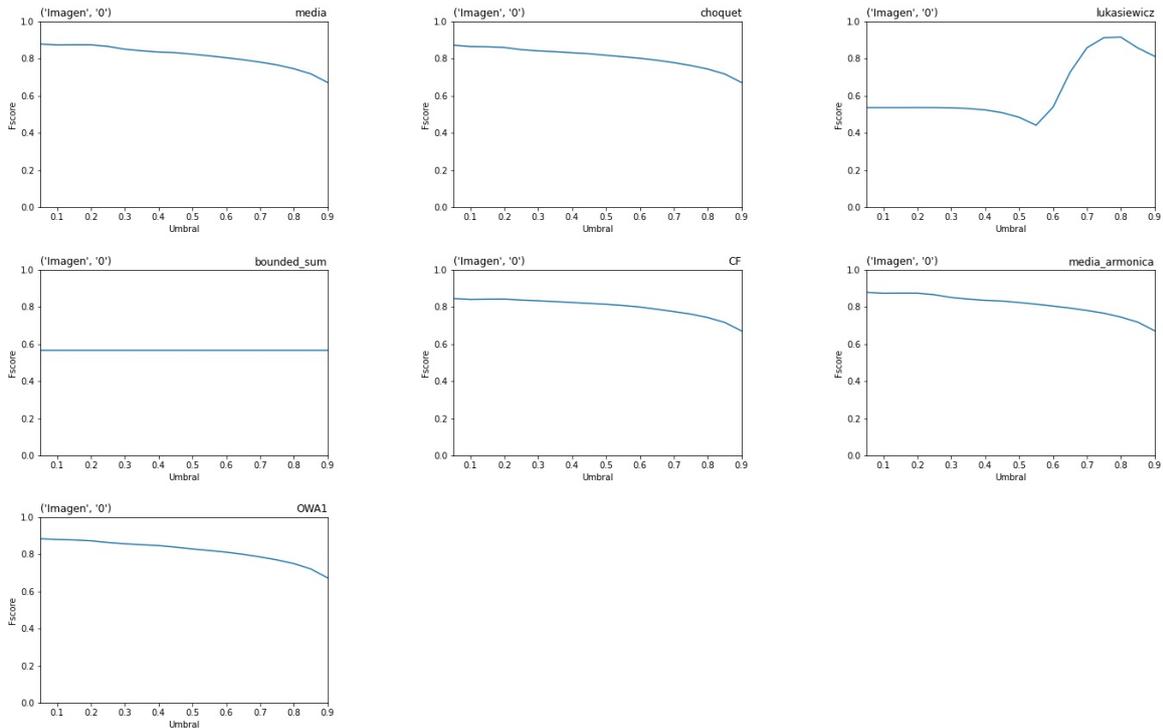


Figura 4.3: Plots Adaptive binarization based on fuzzy integrals con distintos umbrales, de la imagen 0.

Realizamos el cálculo del F1 score para cada conjunto imagen-función con cada una de las combinaciones ventana-umbral y hacemos un plot con los F1 score obtenidos para cada umbral en cada grupo imagen-función-ventana. Vemos en la figura 4.4 que el F1 score es distinto según el umbral y seleccionamos el que nos maximiza el resultado.

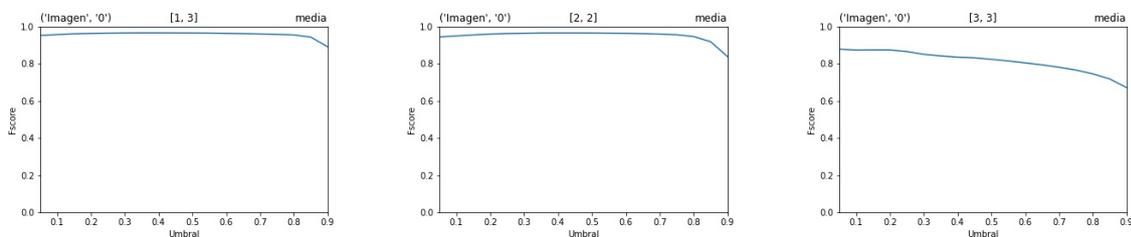


Figura 4.4: Plots Adaptive binarization based on fuzzy integrals con distintos umbrales y ventanas, de la imagen 0 con la función media.

Tras analizar los plots obtuvimos dos claros resultados. En cuanto al estudio de umbrales comprobamos que con la mayoría de las funciones de agregación este dato era clave puesto que modificaba mucho el resultado simplemente cambiando su valor. Además, comprobamos que no había ningún rango de umbrales que nos maximizara el resultado, sino que

dependiendo de la imagen y las funciones utilizadas el mejor umbral variaba mucho. Es por esto que, como explicamos en la sección 3.3, realizamos el cálculo del umbral adaptativo para adecuar cada umbral a sus circunstancias. La segunda conclusión que obtenemos de las gráficas, en el caso del tamaño de las ventanas, es que observamos que claramente los mejores tamaños de ventanas son  $1 \times 3$  y  $3 \times 3$  y además que conforme aumentamos el tamaño la precisión va disminuyendo por lo que el resto de las pruebas se pueden realizar con menos cantidad de ventanas.

#### 4.4. Resultados con distintas funciones de agregación

En la tabla 4.5 se muestran los resultados obtenidos, al calcular el F1 Score de las nueve imágenes, tras aplicarles el algoritmo Adaptive binarization based on fuzzy integrals con las funciones de la sección 2.1.

Imagen \ Función	0	1	2	3	4	5	6	7	8	9	media
Media	0.82	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Mediana	0.82	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Mínimo	0.81	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	0.66	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Máximo	<b>0.83</b>	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	<b>0.77</b>	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Producto	0.53	0.53	0.67	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.65
Sugeno	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.92	<b>0.89</b>	0.68
Choquet	0.82	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Lukasiewicz	0.48	0.84	0.64	0.77	0.47	0.27	0.52	0.77	<b>0.93</b>	<b>0.89</b>	0.66
Drastic	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.68
Nilpotent	0.81	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	0.66	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Hamacher	0.53	0.53	0.68	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.65
Bounded sum	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.68
Drastic t-conorm	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.68
Nilpotent maximum	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.68
Einstein sum	0.53	0.53	0.68	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.65
CF12	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.68
F-Sugeno	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.68
CF	0.81	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	0.66	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Media geométrica	0.82	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Media armónica	0.82	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
Overlap seno	0.55	0.56	0.78	0.77	0.55	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
OWA1	<b>0.83</b>	<b>0.89</b>	0.97	<b>0.78</b>	0.76	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
OWA2	0.81	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	0.67	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>
OWA3	0.82	<b>0.89</b>	<b>0.98</b>	<b>0.78</b>	0.76	<b>0.29</b>	<b>0.68</b>	<b>0.79</b>	<b>0.93</b>	<b>0.89</b>	<b>0.78</b>

Figura 4.5: Resultados F1 score del algoritmo Adaptive binarization based on fuzzy integrals. Parametros: ventana  $3 \times 3$ . Umbral 0,5

Si analizamos la tabla 4.5 observamos que cambiando exclusivamente las funciones de agregación no obtenemos un resultado mejor que el algoritmo [10] y de hecho en la mitad de los casos igualamos la precisión media y en la otra mitad perdemos precisión. La única

función que obtiene los mismos resultados que [10] es la función de agregación máximo y esto es algo obvio ya que tanto con la función máximo como con la integral difusa utilizada por [10] sobre el vector  $\vec{v}$  siempre obtendremos el valor  $v_4$  del vector.

## 4.5. Resultados con toma de decisión

Para el caso de la toma de decisión la evaluamos con el F1 score y creamos los plots, que se muestran en la figura 4.6. Tras obtener la imagen umbralizada la comparamos con la imagen solución y obtenemos el porcentaje de acierto.

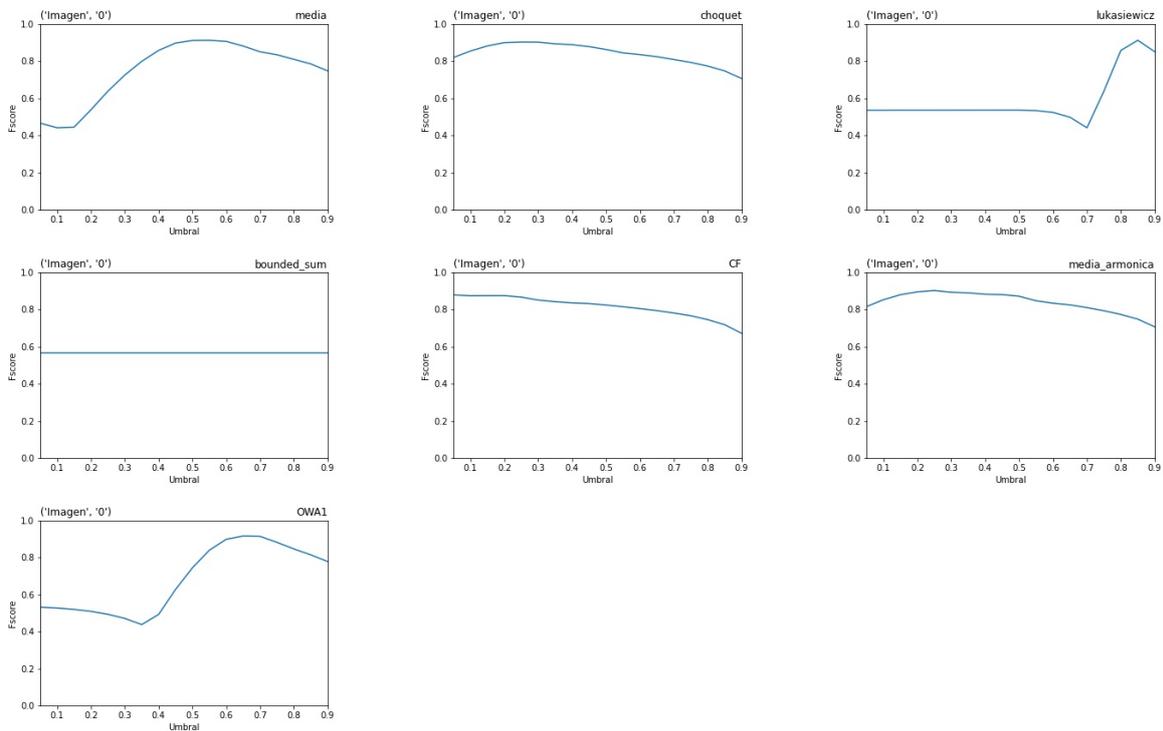


Figura 4.6: Plots Adaptive binarization based on fuzzy integrals con toma de decisión, de la imagen 0. Las funciones obtenidas de la elección al azar son: media geométrica, Media y Lukasiewicz

En la tabla 4.7 mostramos los resultados obtenidos al calcular el F1 Score de las nueve imágenes tras aplicarles la toma de decisión.

Imagen \ Función	0	1	2	3	4	5	6	7	8	9	media
Media	<b>0.91</b>	<b>0.90</b>	0.95	0.67	<b>0.95</b>	<b>0.41</b>	<b>0.76</b>	0.70	0.89	0.83	0.80
Mediana	0.82	0.89	<b>0.98</b>	0.78	0.76	0.29	0.68	0.79	0.93	0.89	0.78
Mínimo	0.82	0.89	<b>0.98</b>	0.78	0.76	0.29	0.68	0.79	0.93	0.89	0.78
Máximo	0.48	0.84	0.64	0.77	0.47	0.27	0.52	0.77	0.93	0.89	0.66
Producto	0.53	0.53	0.67	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.66
Sugeno	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.92	0.89	0.68
Choquet	0.85	<b>0.90</b>	<b>0.98</b>	0.80	0.92	0.30	0.68	<b>0.81</b>	0.93	<b>0.90</b>	0.81
Lukasiewicz	0.53	0.56	0.67	0.77	0.54	0.28	0.52	0.78	0.93	0.89	0.66
Drastic	0.57	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.68
Nilpotent	0.82	0.89	<b>0.98</b>	0.78	0.77	0.29	0.68	0.79	0.93	0.89	0.78
Hamacher	0.53	0.53	0.68	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.66
Bounded sum	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.68
Drastic t-conorm	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.68
Nilpotent maximum	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.68
Einstein sum	0.53	0.53	0.68	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.66
CF12	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.68
F-Sugeno	0.56	0.58	0.93	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.68
CF	0.82	0.89	<b>0.98</b>	0.78	0.76	0.29	0.68	0.79	0.93	0.89	0.78
Media geométrica	0.89	<b>0.90</b>	<b>0.98</b>	<b>0.83</b>	<b>0.95</b>	0.31	0.70	<b>0.81</b>	<b>0.94</b>	<b>0.90</b>	<b>0.82</b>
Media armónica	0.85	<b>0.90</b>	<b>0.98</b>	0.80	0.92	0.30	0.68	0.80	0.93	<b>0.90</b>	0.81
Overlap seno	0.55	0.56	0.78	0.77	0.55	0.28	0.53	0.77	0.93	0.89	0.66
OWA1	0.74	0.89	0.50	0.77	0.93	0.25	0.42	0.74	0.93	0.89	0.71
OWA2	0.82	0.89	<b>0.98</b>	0.78	0.76	0.29	0.68	0.79	0.93	0.89	0.78
OWA3	0.84	<b>0.90</b>	<b>0.98</b>	0.79	0.87	0.29	0.68	0.80	0.93	<b>0.90</b>	0.80

Figura 4.7: Resultados F1 score del algoritmo Adaptive binarization based on fuzzy integrals con toma de decisión. Las funciones obtenidas de la elección al azar son: media geométrica, Media y Lukasiewicz. Parametros: ventana 3x3. Umbral 0,5

Si comparamos la tabla 4.7 con la tabla 4.5 (Adaptive binarization based on fuzzy integrals, parametros: ventana  $3 \times 3$ , umbral 0,5) se puede observar que por lo general los resultados obtenidos han sido mejores. Esto lo hemos conseguido aplicando Adaptive binarization based on fuzzy integrals con cada una de las funciones seleccionada al azar y una vez que tenemos las tres imágenes volvemos a recorrerlas aplicándoles cada función de agregación de la sección 2.1. De esta forma, obtenemos la imagen final que comparamos con su imagen solución y obtenemos el F1 Score. Observamos que por lo general la precisión media no ha mejorado notablemente y de hecho en algunos casos como son las funciones máximo y OWA1 ha disminuido pero sin embargo, en otros casos la precisión ha mejorado considerablemente como es el caso de la media geométrica, la media armónica y la Choquet.

## 4.6. Resultados con umbral dinámico

Como la importancia del umbral es muy grande, puesto que vemos que el resultado se modifica mucho según la selección de este, realizamos el cálculo con un umbral adaptativo. En este caso, obtenemos dos vectores 'X' de tamaño  $[n \times m, 9]$  e 'y' un vector de dimensiones  $[n \times m]$ . 'X' contiene en cada casilla las ventanas  $3 \times 3$  de la imagen umbralizada e 'y' los valores de la imagen solución.

A estos datos le aplicamos una clasificación con LogisticRgression de sklearn. Los separamos en un 80 % para el train y un 20 % para el test, los entrenamos y obtenemos el vector de predicciones de los datos de test. Una vez que tenemos este vector lo comparamos, con un F1 score, con los datos del test del vector de soluciones.

En la tabla 4.8 mostramos los resultados obtenidos, al calcular el F1 Score de las nueve imágenes, tras aplicarles la regresión logística.

Imagen \ Función	0	1	2	3	4	5	6	7	8	9	media
Media	<b>0.98</b>	0.89	0.98	<b>0.99</b>	<b>0.96</b>	<b>0.91</b>	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	<b>0.95</b>
Mediana	<b>0.98</b>	0.89	0.98	<b>0.99</b>	<b>0.96</b>	0.90	0.96	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	<b>0.95</b>
Mínimo	0.97	<b>0.90</b>	0.96	0.98	0.95	0.90	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	0.94
Máximo	<b>0.98</b>	0.86	0.98	<b>0.99</b>	<b>0.96</b>	0.89	<b>0.97</b>	0.84	<b>0.93</b>	<b>0.89</b>	0.93
Producto	0.66	0.73	0.67	0.77	0.70	0.29	0.53	0.84	<b>0.93</b>	<b>0.89</b>	0.70
Sugeno	0.53	0.52	0.67	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
Choquet	<b>0.98</b>	0.88	0.97	0.98	<b>0.96</b>	0.90	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	<b>0.95</b>
Lukasiewicz	<b>0.98</b>	<b>0.90</b>	<b>0.99</b>	<b>0.99</b>	<b>0.96</b>	0.37	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	0.90
Drastic	0.53	0.52	0.67	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
Nilpotent	0.97	<b>0.90</b>	0.96	0.98	0.95	0.90	0.96	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	<b>0.95</b>
Hamacher	0.53	0.52	0.61	0.77	0.54	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
Bounded sum	0.53	0.52	0.67	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
Drastic t-conorm	0.53	0.52	0.67	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
Nilpotent maximum	0.53	0.52	0.67	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
Einstein sum	0.53	0.52	0.65	0.77	0.54	0.28	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
CF12	0.53	0.52	0.67	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
F-Sugeno	0.53	0.52	0.67	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
CF	0.97	0.87	0.96	0.98	0.95	0.28	0.96	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	0.87
Media geométrica	<b>0.98</b>	0.88	0.98	<b>0.99</b>	<b>0.96</b>	<b>0.91</b>	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	<b>0.95</b>
Media armónica	<b>0.98</b>	0.89	0.97	<b>0.99</b>	<b>0.96</b>	<b>0.91</b>	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	<b>0.95</b>
Overlap seno	0.53	0.52	0.61	0.77	0.55	0.29	0.53	0.77	<b>0.93</b>	<b>0.89</b>	0.66
OWA1	0.97	0.83	0.98	<b>0.99</b>	0.95	0.89	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	0.94
OWA2	0.97	0.86	0.96	0.98	0.94	0.69	0.96	0.94	<b>0.93</b>	<b>0.89</b>	0.92
OWA3	<b>0.98</b>	0.89	0.98	0.98	<b>0.96</b>	<b>0.91</b>	<b>0.97</b>	<b>0.95</b>	<b>0.93</b>	<b>0.89</b>	<b>0.95</b>

Figura 4.8: Resultados F1 score del algoritmo Adaptive binarization based on fuzzy integrals con regresión logística.

Viendo los resultados de la tabla 4.8 se puede comprobar que hay una clara mejoría en la umbralización. Como era de esperar gracias a la regresión logística la precisión ha aumentado, ya que como hemos mencionado en la sección 4.3, el umbral juega un papel clave en este algoritmo y gracias a utilizar un umbral dinámico casi la mitad de las funciones de agregación obtienen una precisión superior al 90%.

## 4.7. Comparación con otros métodos

Para poder tener más algoritmos con los que comparar los resultados utilizamos los métodos no adaptativos de OTSU y Sauvola. En la tabla 4.9 mostramos los resultados de estos algoritmos comparados con los mejores del resto de algoritmos comentados durante el resto de la memoria.

Imagen \ Algoritmo	0	1	2	3	4	5	6	7	8	9	media
OTSU	0.91	0.88	<b>0.98</b>	0.97	0.95	0.87	0.94	0.91	0.80	<b>0.89</b>	0.91
Sauvola	0.67	0.87	<b>0.98</b>	0.79	0.65	0.29	0.59	0.78	0.93	<b>0.89</b>	0.75
ATUII	0.83	0.89	<b>0.98</b>	0.78	0.77	0.29	0.69	0.79	0.93	<b>0.89</b>	0.78
FLAT original	0.83	0.89	<b>0.98</b>	0.78	0.77	0.29	0.68	0.79	0.93	<b>0.89</b>	0.78
ELAB funciones de agregación	0.82	0.89	<b>0.98</b>	0.78	0.76	0.29	0.68	0.79	0.93	<b>0.89</b>	0.78
ELAB toma de decisión	0.89	<b>0.90</b>	<b>0.98</b>	0.83	0.95	0.31	0.70	0.81	<b>0.94</b>	<b>0.89</b>	0.82
Dynamic ELAB	<b>0.98</b>	0.88	<b>0.98</b>	<b>0.99</b>	<b>0.96</b>	<b>0.91</b>	<b>0.97</b>	<b>0.95</b>	0.93	<b>0.89</b>	<b>0.95</b>

Figura 4.9: Comparación de los métodos analizados.

Comparando todos los métodos utilizados durante este trabajo vemos que el uso del umbral dinámico es crucial puesto que, como se puede observar en la tabla 4.9, la precisión media es muy superior al resto de algoritmos de umbralización utilizados. Asimismo, vemos que el único que supera el 90% de precisión del resto es el algoritmo de OTSU y esto recalca la importancia del umbral ya que, como explicamos en la sección 2.2, el método de OTSU es un método donde se evalúa la precisión de varios umbrales para seleccionar el umbral óptimo.



# CAPÍTULO 5

---

## Resultados

---

### 5.1. Conclusiones y líneas futuras

En este trabajo hemos estudiado el Fuzzy Adaptive. Lo que queríamos demostrar era que podíamos mejorar la precisión de FLAT con experimentos aun inexplorados como la utilización de distintas funciones de agregación, entre las que se encuentran integrales difusas modernas como CF1,2, T-normas y T-conormas, los overlaps y OWAs. Además, también hemos estudiado la toma de decisión o el umbral adaptativo.

Las evidencias mostradas en el apartado de experimentación muestran una clara mejora en el primer algoritmo tras aplicarle las distintas modificaciones. Con el algoritmo original obtuvimos unos resultados parecidos o peores según la función aplicada. Las pruebas realizadas de modificación de ventanas nos sirvieron para comprobar cuales eran las mejores ventanas a utilizar. También realizamos una combinación de todos los umbrales con todas las ventanas y para obtener la mejor combinación que nos maximizara la precisión media. La toma de decisión la realizamos para demostrar que, si le aplicamos distintas funciones y agregamos los resultados de estas con cada una de las funciones de agregación, siempre íbamos a obtener un mejor resultado que si la estudiamos con una sola función. Por último, utilizamos el umbral adaptativo y los resultados mejoraron claramente obteniendo una precisión que superaba la media del 90 % con casi la mitad de las funciones de agregación. Vemos una clara mejoría con respecto al algoritmo original y obviamente con respecto al algoritmo de Adaptive Thresholding Using Integral Image.

De cara al futuro se estudiará la utilización de las d-integrales como funciones de agregación o la aplicación de métodos intervalares como OWA intervalares e integrales Choquet intervalares sustituyendo la función integral con la que se calcula la imagen integral por estos operadores de agregación.



---

## Bibliografía

---

1. P. A. Cheremkhin y E. A. Kurbatova, Comparative appraisal of global and local thresholding methods for binarisation of off-axis digital holograms. *Optics and Lasers in Engineering* **115**, 119-130, ISSN: 0143-8166 (2019).
2. T. Kalaiselvi, P. Nagaraja y V. Indhu, A comparative study on thresholding techniques for gray image binarization. *Int. J. Adv. Res. Comput. Sci* **8**, 1168-1172 (2017).
3. P. Roy, S. Dutta, N. Dey, G. Dey, S. Chakraborty y R. Ray, presentado en 2014 International conference on control, Instrumentation, communication and Computational Technologies (ICCICCT), págs. 1182-1186.
4. X. Yan, L. G. Jeub, A. Flammini, F. Radicchi y S. Fortunato, Weight thresholding on complex networks. *Physical Review E* **98**, 042304 (2018).
5. T. J. Gross, M. Bessani, W. D. Junior, R. B. Araújo, F. A. C. Vale y C. D. Maciel, An analytical threshold for combining Bayesian Networks. *Knowledge-Based Systems* **175**, 36-49 (2019).
6. F. Bardozzo, P. Lió y R. Tagliaferri, A study on multi-omic oscillations in Escherichia coli metabolic networks. *BMC bioinformatics* **19**, 194 (2018).
7. J. Sauvola y M. Pietikäinen, Adaptive document image binarization. *Pattern recognition* **33**, 225-236 (2000).
8. D. Bradley y G. Roth, Adaptive Thresholding using the Integral Image. *Journal of Graphics Tools* **12**, 13-21 (2007).
9. J. Debayle y J.-C. Pinoli, General adaptive neighborhood Choquet image filtering. *Journal of Mathematical Imaging and Vision* **35**, 173-185 (2009).
10. F. Bardozzo, B. De La Osa, Ľ. Horanská, J. Fumanal-Idocin, M. d. Priscoli, L. Troiano, R. Tagliaferri, J. Fernandez y H. Bustince, Sugeno integral generalization applied to improve adaptive image binarization. *Information Fusion* **68**, 37-45, ISSN: 1566-2535 (abr. de 2021).
11. F. El Baf, T. Bouwmans y B. Vachon, presentado en 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), págs. 1729-1736.
12. G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri y F. Herrera, Deep learning in video multi-object tracking: A survey. *Neurocomputing* **381**, 61-88 (2020).
13. P. D. Wellner, Adaptive thresholding for the DigitalDesk. *Xerox, EPC1993-110*, 1-19 (1993).

14. W. Niblack, presentado en G. Leedham, C. Yan, K. Takru, J. Tan and L. Mian. Comparison of Some Thresholding Algorithms for Text/Background Segmentation in Difficult Document Images. In Proceedings of the Seventh International Conference on Document Analysis and Recognition, págs. 859-864.
15. J. Wu, F. Da, C. Wang y S. Gai, presentado en International Conference on Image and Graphics, págs. 347-360.
16. A. Kasagi, K. Nakano e Y. Ito, presentado en 2014 43rd International Conference on Parallel Processing, págs. 251-260.
17. G. Beliakov, H. B. Sola y T. C. Sánchez, *A practical guide to averaging functions* (Springer, 2016).
18. M. Xia, Z. Xu y B. Zhu, Some issues on intuitionistic fuzzy aggregation operators based on Archimedean t-conorm and t-norm. *Knowledge-Based Systems* **31**, 78-88 (2012).
19. G. Arenas-Díaz y E. R. R. Lamus, Medidas difusas e integrales difusas. *Universitas scientiarum* **18**, 7-32 (2013).
20. L. Ko, Y. Lu, H. Bustince, Y. Chang, Y. Chang, J. Fernandez, Y. Wang, J. A. Sanz, G. Pereira Dimuro y C. Lin, Multimodal Fuzzy Fusion for Enhancing the Motor-Imagery-Based Brain Computer Interface. *IEEE Computational Intelligence Magazine* **14**, 96-106 (2019).
21. H. Bustince, J. Fernandez, R. Mesiar, J. Montero y R. Orduna, Overlap functions. *Nonlinear Analysis: Theory, Methods & Applications* **72**, 1488-1499 (2010).
22. R. R. Yager, Families of OWA operators. *Fuzzy sets and systems* **59**, 125-148 (1993).
23. L. A. Zadeh, en *Computational linguistics* (Elsevier, 1983), págs. 149-184.
24. X. Xu, S. Xu, L. Jin y E. Song, Characteristic analysis of Otsu threshold and its applications. *Pattern recognition letters* **32**, 956-961 (2011).