UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

Facultad de Ingeniería Eléctrica
División de Estudios de Posgrado

FUSIÓN Y SEGMENTACIÓN EFICIENTE DE IMÁGENES EN BASE A LA CALIDAD DE LOS PIXELES

TESIS

Que para obtener el grado de

DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Opción en Sistemas Computacionales

Presenta

Adan Garnica Carrillo

Dr. Félix Calderón Solorio

Director de Tesis

Morelia Michoacán Septiembre 2019

Quiero dedicar esta tesis a:

** A mi padre † **Aristeo Garnica** por haberme enseñado el valor del trabajo constante, el esfuerzo y la dedicación.*

Deseo expresar mi agradecimiento:

**A Dios, por permitirme lograr siempre mis objetivos.*

** A mis profesores y todos los que creyeron en mi y me apoyaron con sus palabras de aliento.*

** A mi madre y hermanas por el apoyo que siempre me han dado.*

**A la Universidad Michoacana de San Nicolás de Hidalgo y a la Facultad de Ingeniería Eléctrica por los medios que pusieron a mi disposición para la realización de este trabajo.*

**Al CONACYT por el apoyo económico otorgado.*

**A los miembros de la mesa sinodal:*

- Dr. Juan José Flores Romero, presidente del jurado.

- Dr. Jaime Cerda Jacobo, vocal

- Dr. José Antonio Camarena Ibarrola, vocal.

- Dr. Sergio Rogelio Tinoco Martinez, revisor externo.

Por sus valiosas aportaciones a esta tesis.

** De una manera especial a mi asesor el **Dr. Félix Calderón Solorio**, por el tiempo, paciencia y conocimiento invertidos en este proyecto, por soportar las discusiones (muchas veces sin fundamento), pero que, sin duda enriquecieron el trabajo y abrieron nuevas lineas de investigación.*

A las personas más importantes en mi vida: mi esposa **Lizet Gama y mis hijas **Brenda Amayrani** y **Dayana Thaily** que siempre me apoyan e impulsan para no darme por vencido y que sufrieron conmigo los desvelos y la falta de atención y de tiempo para convivir. Las amo y valoro mucho su comprensión y amor, sin su apoyo, esta tesis no habría sido posible.*

Lista de publicaciones

- Contribuciones en revistas indexadas JCR.

    1. Multi-Focus Image Fusion for Multiple Images using Adaptable Sized Windows and Parallel Programming. Signal, Image and Video Processing, **En segunda revisión.** Adan Garnica, Félix Calderon, y Juan Flores.
    2. Multi-focus image fusion by local optimization over sliding windows. Signal, Image and Video Processing, Enero 2018. Adan Garnica, Felix Calderon, y Juan Flores.
    3. Fusión de imágenes multi-foco con ventanas variables. Revista Iberoamericana de Automática e Informática industrial, 2017. Félix Calderón, Adan Garnica y Juan J. Flores.
    4. Fusión de imágenes multi-foco basado en la combinación lineal de imágenes utilizando imágenes incrementales. Revista Iberoamericana de Automática e Informática Industrial RIAI, 13(4):450 -461, 2016. Félix Calderón, Adan Garnica y Juan J. Flores.

- Contribuciones en congresos internacionales.

    1. Development of a mobile service robot for classify objects and place them in containers. International Autumn Meeting on Power, Electronics and Computing (ROPEC), pages 1-6. IEEE. Adan Garnica, Salvador-Daniel Pelayo, Juan de Dios Pelayo y Leonardo Muñoz.
    2. Fuzzy Nearest Neighbor Time Series Forecasting - Computational Complexity. International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, 2016, pp. 490-495. J. J. Flores, F. Calderon, E. Espinosa, R. Cedeno, A. Garnica and G. Flores.
    3. A fast algorithm for binary Segmentation using color information. International Autumn Meeting on Power, Electronics and Computing (ROPEC), 2015, pp. 1-7. Felix Calderon, Juan J. Flores y Adan Garnica.

Resumen

La fusión de imágenes multi-foco consiste en, dadas dos o más imágenes tomadas de la misma escena pero con distinto nivel de enfoque, extraer de ellas las regiones más nítidas. Este proceso se realiza con el fin de formar una nueva imagen en la que aparezcan de forma nítida todos los objetos interesantes de la escena. La propuesta para resolver el problema de fusión de imágenes multi-foco puede ser aplicada para fusionar imágenes con iluminación no uniforme.

En esta tesis se presenta una propuesta para resolver el problema de fusión de imágenes en base a un mapa de decisión, el cual contiene en cada coordenada, el índice de la imagen que tiene el pixel con mayor calidad en esa posición. Para calcular el mapa de decisión se utiliza un sistema de ventanas deslizantes de tamaño adaptable, logrando con ello una buena definición en los bordes de los objetos, sin perder la coherencia espacial en las regiones sin bordes. Adicionalmente, el cálculo del valor de cada coordenada del mapa de decisión se obtiene utilizando imágenes integrales, evitando realizar un proceso de optimización (que generalmente es tardado). El tiempo requerido para calcular un valor del mapa de decisión es constante e independiente del tamaño de la ventana. El tamaño de ventana adecuado se determina de forma eficiente con búsqueda binaria tomando en cuenta los bordes del mapa de decisión.

El algoritmo presentado para la fusión de imágenes multi-foco puede ser aplicado para fusionar imágenes en otros contextos, por ejemplo con iluminación no uniforme, cambiando únicamente la función que mide la calidad de los pixele. Dado que un sub-producto derivado del algoritmo de fusión es el mapa de decisión. Se presenta una modificación del algoritmo de fusión para ser aplicado en la segmentación de imágenes por color. Adicionalmente, se presenta el uso del mapa de decisión obtenido con el algoritmo de fusión para segmentar imágenes en escala de grises con iluminación no uniforme, las cuales son binarizadas utilizando un sistema de ventanas deslizantes de tamaño variable, logrando resultados competitivos con los del estado el arte.

Palabras clave: Fusión, imágenes multi-foco, segmentación, ventanas deslizantes, imágenes integrales, mapa de decisión, binarización, imágenes con iluminación no uniforme.

Abstract

The multifocus image fusion is the process by which two or more partially focused images are combined into a new image in which all interested objects have the maximum sharpness. Proposals to solve the problem of multifocus image fusion can be applied to fuse images with non-uniform illumination.

In this thesis we present a proposal to solve the problem of multi-focus image fusion, based on a decision map, which contains in each coordinate the index of the image with the sharpest pixel in each position. To calculate the decision map is used a sliding windows system with adaptable sizes, thus, achieving a good definition at the edges of the objects with enough spatial coherence in regions where there are no borders. Additionally, calculation of each value in the decision map is obtained using integral images, avoiding an optimization process (which is usually slow). In addition, the size of the applied window does not influence the required time needed to calculate each value in the decision map. This time is constant and does not depend on the window size. The appropriate window size for each position is determined efficiently, using binary search and the information of the edges of the decision map.

The algorithm presented for the multi-focus imagen fusion can be applied to fuse images in other contexts, for example with non-uniform illumination, changing only the function that measures the quality of the pixels in the image. The decision map is a sub-product of the fusion algorithm. So, it can be used for image segmentation. We present a modification of the proposed algorithm, to be applied for image segmentation by color. Additionally, a proposal to use the fusion algorithm, to calculate a decision map that allows to segment grayscale images with non-uniform illumination is presented, and subsequently we binarize them using a sliding window system with adaptable sizes. Results obtained with our proposal are comparable with those reported in the state of art.

Keywords: Fusion, Multi-focus images, Segmentation, Sliding windows, Integral images, Decision map, Binarization , Non-uniform illumination.

Lista de figuras

Lista de tablas

Lista de algoritmos

Lista de símbolos

A	Matriz aumentada en el método Simplex.
$B^{(i)}$	Matriz de bordes del mapa de decisión en la iteración i.
$\mathcal{B}_{r,c}$	Imagen integral de los bordes de P.
b	Diámetro del círculo de luz generado por un punto fuera de foco.
$\mathcal{C}$	Conjunto de entrenamiento para la segmentación.
$\mathcal{C}_j(k)$	j-ésima clase de la k-ésima imagen.
D	Imagen desenfocada.
d_i	Distancia donde se crea la imagen perfecta, según el modelo de la lente.
d_o	Distancia al objeto.
d_s	Distancia de la lente al sensor.
E	Imagen nítida.
$F(k)$	Función de calidad (nitidez/iluminación/verosimilitud) de la imagen k.
$\bar{F}(P)$	Promedio de la función de calidad aplicada la imagen final, calculada con un P dado.
$\bar{\mathcal{F}}_{r,c}$	Promedio de los valores de la función de calidad dentro de una ventana centrada en r, c de tamaño $(2w+1) \times (2w+1)$ en la imagen final, calculada con un submapa de decisión $Q^{r,c}$.
$\mathbf{F}_{r,c}$	Resultados de $F(k)$ para todas las imágenes en la coordenada r, c.
$FWIU$	Frecuencia Pesada de la Intersección sobre la Union (Frequency Weighted Intersection over Union).
f	Distancia focal de la cámara.
g	Gaussiana en una dimensión con media μ y desviación estándar σ.
G	Gaussiana en dos dimensiones.
h	Kernel del Laplaciano discretizado.
$H(k)$	Imagen integral de la nitidez de la k-ésima imagen.
$\mathcal{H}$	Histograma de la k-ésima imagen.
$\mathcal{H}^Q(k)$	Histograma del canal Q de la imagen.
I	Conjunto de imágenes multi-foco.
$I(k)$	k-ésima imagen del conjunto.
$I_{r,c}(k)$	Valor del pixel el pixel r, c de la k-ésima imagen.
J	Imagen fusionada.
$\mathbf{J}_{r,c}$	Conjunto de posibles valores para la coordenada r, c de la imagen final.

L	Laplaciano.
$L_{r,c}^{(s)}$	Imagen escalada a s veces su tamaño original.
LoG	Laplaciano de Gauss.
M	Matriz de restricciones en la optimización con Simplex.
M	Número de restricciones en el proceso de optimización.
$m_j(k)$	Moda de los valores de la j-ésima clase de la k-ésima imagen.
MIU	Promedio de la intersección sobre la unión (Mean Intersection over Union).
MPA	Promedio de precisión a nivel pixel (Mean Pixel Accuracy).
n_r	Número de renglones de la matriz que representa la imagen.
n_c	Número de columnas de la matriz que representa la imagen.
N	Número de imágenes en el conjunto I.
$\mathcal{N}$	Cantidad de variables en la optimización con el método Simplex.
N_{iter}	Iteraciones que se permitirán para refinar el mapa de decisión.
N_h	Número de hilos que participan en el procesamiento en paralelo.
O	Imagen original para el proceso de segmentación.
P	Mapa de decisión
$P^{(i)}$	Mapa de decisión en la iteración i.
$\hat{P}$	Mapa de decisión de referencia utilizado para crear imágenes sintéticas.
PC	Profundidad de campo comprendida, rango comprendido entre PC_0 y PC_1.
PA	Precisión a nivel pixel (Pixel accuracy).
$Q^{r,c}$	Submapa de decisión en una ventana centrada en r, c de $(2w+1) \times (2w+1)$.
T	Tolerancia de pixeles de borde dentro de una ventana.
$\mathcal{T}$	Cantidad de pixeles de ejemplo para la segmentación.
$u_k^\star$	Umbral óptimo que separa dos clases en la k-ésima imagen..
$V_{r,c}$	Número de ventanas en las que abarcan el pixel $p(r, c)$.
w_{max}	Limita el tamaño máximo de ventana $(2w_{max}+1) \times (2w_{max}+1)$.
w	Parámetro usado para establecer el tamaño de las ventanas
$W^{(i)}$	Matriz de tamaños de ventana en la iteración i.
x_h	Variables de holgura en la optimización utilizando Simplex.
z	Ecuación en la forma canónica para el método Simplex.
σ	Desviación estándar de la gaussiana.
Δ	Separación entre los centros de las ventanas (permite controlar el traslape).
μ_k	Media de los valores de la k-ésima clase.
$*$	Símbolo que representa la convolución.

Capítulo 1

Introducción

Dada la gran cantidad de imágenes que existen en la actualidad, se requiere de algoritmos óptimos para procesarlas. El procesamiento de imágenes comprende la fusión, restauración, segmentación, extracción de información, mejoramiento, detección de falsificación o modificación, etc. En la literatura existen muchos trabajos enfocados al procesamiento de versiones digitales de imágenes reales o sintéticas, sin embargo, la mayoría de ellos son aplicables para hacer sólo una de las tareas anteriores. En este trabajo presentamos un algoritmo de fusión de imágenes que puede ser aplicado para realizar segmentación, clasificación y binarización robusta de imágenes.

La fusión de imágenes es de gran utilidad en distintos contextos de la investigación y de la vida cotidiana. Por ejemplo en [Himanshi et˜al., 2015] se presenta una propuesta para realizar la fusión de tomografías computarizadas con imágenes de resonancias magnéticas con el objetivo de combinar información de dos fuentes y apoyar en el diagnóstico médico. En [Chaudhuri y Kotwal, 2013] también se presenta un análisis de técnicas de fusión de imágenes las cuales se aplican para fusionar información de distintas bandas en imágenes satelitales. Otro ejemplo es el presentado en [Kausar et˜al., 2016], donde se fusionan imágenes obtenidas con microscopios con el objetivo de mejorar la calidad de las imágenes y contribuir en el apoyo al diagnóstico médico. En [Agapiou et˜al., 2016] se utiliza la fusión de imágenes para combinar información

contenida en imágenes antiguas que están en escala de grises con imágenes en color tomadas recientemente, donde las imágenes provienen de satélites o de cámaras aerotransportadas; el objetivo de dicha fusión es analizar los cambios en las imágenes, por ejemplo cambios en uso de suelo. En [Nejati et˜al., 2015] se presenta una propuesta para fusionar imágenes multi-foco obtenidas con una cámara Lytro, con el fin de obtener una imagen completamente nítida en base a imágenes multi-foco. En [Li et˜al., 2017] se presenta una compilación de diferentes técnicas de fusión de imágenes y una gráfica del crecimiento de artículos científicos en esta área, demostrando el incremento del interés de los investigadores por resolver dicho problema. Sin embargo, a pesar de la gran cantidad de artículos publicados, existen problemas que no han sido resueltos aún; hay propuestas que ofrecen un alto porcentaje de acierto a costa de un incremento considerable en el tiempo de ejecución y viceversa, otros ofrecen buena coherencia espacial pero sacrifican la correcta segmentación en los bordes. En nuestra propuesta atacamos estos problemas utilizando técnicas que permiten obtener la fusión en décimas de segundo con un alto porcentaje de acierto, además de hacer un procesamiento diferenciado en los bordes para lograr una correcta segmentación sin sacrificar la coherencia espacial. La coherencia espacial consiste en que un pixel y sus vecinos tengan características similares; es decir, que no existan cambios bruscos entre un pixel y sus vecinos.

1.1. Ejemplos de fusión de imágenes.

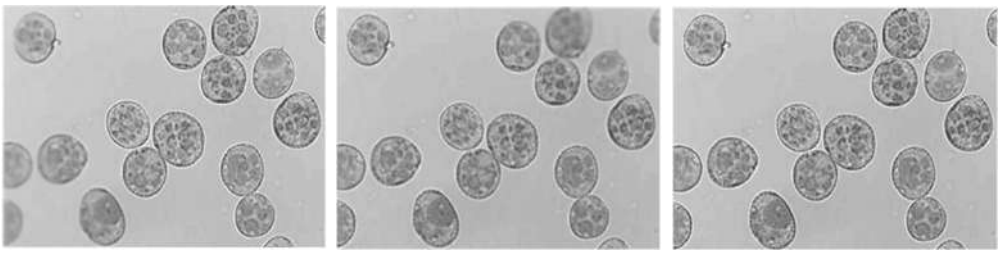
En la Figura 1.1 se muestran dos imágenes tomadas con diferente nivel de enfoque, 1.1(a) y 1.1(b), las cuales contienen regiones borrosas y regiones nítidas. Dichas imágenes son fusionadas para obtener una imagen completamente nítida mostrada en la Figura 1.1(c).

En la Figura 1.2 se muestra un ejemplo de fusión de imágenes tomadas con microscopio, donde la imagen de la Figura 1.2(a) tiene regiones nítidas (esquina superior

(a) **(b)** **(c)**

Figura 1.1: Ejemplo de fusión de imágenes multi-foco. a) y b) son imágenes multi-foco y c) es la imagen fusionada

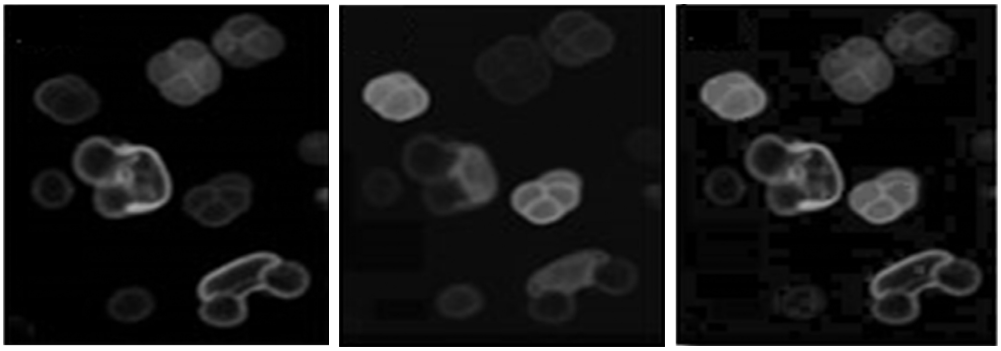
derecha) y regiones borrosas (esquina inferior izquierda) mientras que en la imagen de la Figura 1.2(b) las condiciones son inversas. Mediante un proceso de fusión se permite tener un panorama más completo de la situación tal como se muestra en la Figura 1.2(c) en donde se aprecia una imagen con mayor claridad que en 1.2(a) y 1.2(b).



(a) **(b)** **(c)**

Figura 1.2: Ejemplo de fusión de imágenes tomadas con microscopio. a) y b) Imágenes fuente y c) imagen fusionada.

En la Figura 1.3 se muestra otro ejemplo de fusión de imágenes tomadas con microscopio. En la Figura 1.3(c) se muestra la fusión de la información contenida en 1.3(a) y 1.3(b). Nótese que en la imagen fusionada hay información que no aparece en 1.3(a) pero si en 1.3(b) y viceversa, logrando con la fusión tener en una sola imagen toda la información.



(a) **(b)** **(c)**

Figura 1.3: Ejemplo 2 de fusión de imágenes tomadas con microscopio. a) y b) Imágenes fuente y c) imagen fusionada

El proceso de fusión también se lleva a cabo para mejorar imágenes u obtener nueva información de ellas en base a la iluminación, [Lu et˜al., 2017; Yu et˜al., 2014; Tian y Cohen, 2018]. En la Figura 1.4 se muestra un ejemplo de fusión de imágenes extraído de [Tian y Cohen, 2018] en el que 1.4(a) es la imagen original, 1.4(b) y 1.4(c) se obtienen mejorando el contraste de 1.4(a) de forma global (para toda la imagen) y local (en una vecindad), respectivamente. Finalmente, en 1.4(d) se muestra la fusión de 1.4(b) y 1.4(c), la cual tiene mejor iluminación que el resto de las imágenes.

(a) **(b)** **(c)** **(d)**

Figura 1.4: Fusión de imágenes para mejorar la iluminación. a) imagen original, b) mejoramiento de contraste global, c) mejoramiento de contraste local y d) imagen fusionada.

Otro contexto donde se aplica la fusión combinada con la segmentación es en el mejoramiento de imágenes, cuando se desea incrementar el contraste o la iluminación, pero sólo en las regiones de la imagen donde dichas características no son las deseables. Dado que aplicar el proceso sobre toda la imagen causaría estragos en las regiones que no requieren correcciones, se segmenta la imagen y se aplican las correcciones necesarias sólo en las partes que lo requieren, para finalmente realizar la fusión con el objetivo de volver a tener una sola imagen pero con las correcciones realizadas. En la Figura 1.5 se muestran 3 imágenes, la Figura 1.5(a) es una imagen con iluminación no uniforme a la que se aplica un proceso para mejorar la visibilidad de las regiones obscuras obteniendo la imagen mostrada en la Figura 1.5(b), la cual tiene diferente iluminación que 1.5(a). Finalmente, se realiza un proceso de fusión para obtener la imagen mostrada en la Figura 1.5(c) en la que se puede apreciar que el texto tiene mayor visibilidad que en 1.5(a) y 1.5(b).

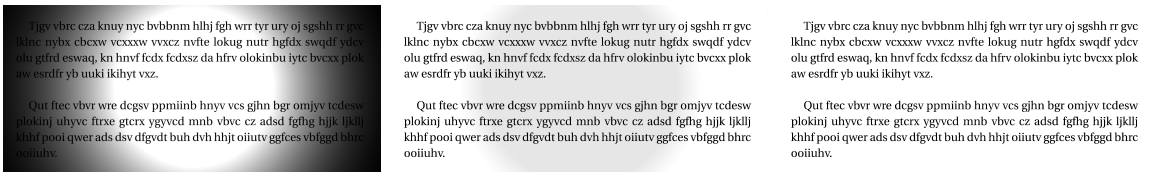
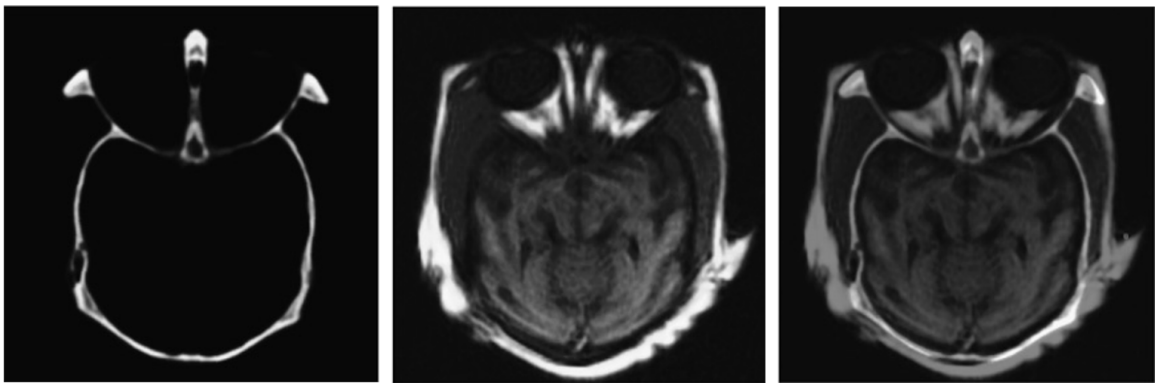


Figura 1.5: Ejemplo de fusión de imágenes con iluminación no uniforme extraída de [Molina et˜al., 2018], donde a) es la imagen original, b) es el resultado de mejorar la iluminación de a) y c) es el resultado de fusionar a) y b).

En el apoyo al diagnóstico médico se utiliza la fusión para extraer información de imágenes que contienen características complementarias para tener un panorama completo de la situación, por ejemplo, de una tomografía, [Wang y Ma, 2008; Bhatnagar et˜al., 2013; Liu et˜al., 2014b; Himanshi et˜al., 2015]. En la Figura 1.6, se muestra un ejemplo de fusión de imágenes médicas extraídas de [Wang y Ma, 2008]. En la Figura 1.6(a) se muestra una imagen de una tomografía computarizada mientras que la Figura 1.6(b) muestra una resonancia magnética. Finalmente, la Figura 1.6(c)

muestra la fusión de la tomografía y la resonancia, con lo cual se logra combinar la información de ambas imágenes una sola. Para la fusión de imágenes médicas se utilizan distintos métodos, por ejemplo, el cálculo de la información mutua (MI por sus siglas en inglés) [Paninski, 2003; Jiang et˜al., 2010]. Para los procesos de fusión de imágenes, es común aplicar un proceso de registro de imágenes, previo a la fusión, con la finalidad de alinear los pixeles de las imágenes.



(a) **(b)** **(c) *Fusionada***

Figura 1.6: Ejemplo de fusión de imágenes de una tomografía y resonancia magnética

En general, siempre que tengamos dos o más imágenes y queramos o necesitemos tomar algunas partes de una imagen y complementarlas con partes del resto de las imágenes se requerirá realizar un proceso fusión. Aunque el proceso se puede llevar a cabo de forma manual, no es práctico hacerlo de esta forma debido a que generalmente se tienen grandes conjuntos de imágenes o se requiere que el proceso se haga de forma rápida, lo cual no siempre es posible si se hace de forma manual, por lo que en la actualidad se trabaja para automatizar dicho proceso.

1.2. Proceso para fusionar imágenes.

Dado un conjunto de imágenes las cuales se pretende fusionar y dada una propiedad que nos interesa maximizar (nitidez, iluminación, color, contraste, etc.), en

general, se pueden seguir los siguientes pasos para llevar a cabo la fusión de las imágenes:

- **Paso I:** Medición de la propiedad seleccionada para discriminar las regiones deseables y las no deseables de cada imagen.

- **Paso II:** Cálculo de un mapa de decisión. Basados en la medición realizada, se requiere determinar las partes que conservaremos de cada imagen, las cuales deben ser complementarias. El mapa de decisión es una matriz del mismo tamaño que las imágenes y en cada coordenada almacena una etiqueta que indica la imagen de la que se deberá extraer el pixel de esa coordenada. Dicho mapa debe garantizar que se conservan las regiones deseables de cada imagen, que la imagen final sea aquélla en la que se maximiza la medida de la propiedad seleccionada y que las regiones extraídas de cada imagen tengan coherencia espacial.

- **Paso III:** Extraer las regiones de interés de cada imagen utilizando el mapa de decisión, conservando de cada imagen k los pixeles donde el mapa de decisión tiene un valor k y desechando el resto.

- **Paso IV:** Fusión o combinación de las partes deseables (según las etiquetas del mapa de decisión) de cada imagen del conjunto, para formar la imagen final, la cual deberá ser más adecuada para la percepción humana o para el procesamiento digital, que cualquiera de las imágenes del conjunto inicial.

Para realizar el proceso de fusión de imágenes de forma automática es necesario determinar primero la propiedad que se desea optimizar y que se utilizará como discriminante, por ejemplo:

- **La iluminación:** para tener imágenes uniformemente iluminadas.

- **El color:** para lograr una imagen con algunas características de color deseadas, por ejemplo: extraer las partes de color verde para medir el crecimiento de las plantas en invernaderos o detectar regiones con mayor o menor densidad de vegetación en imágenes satelitales.

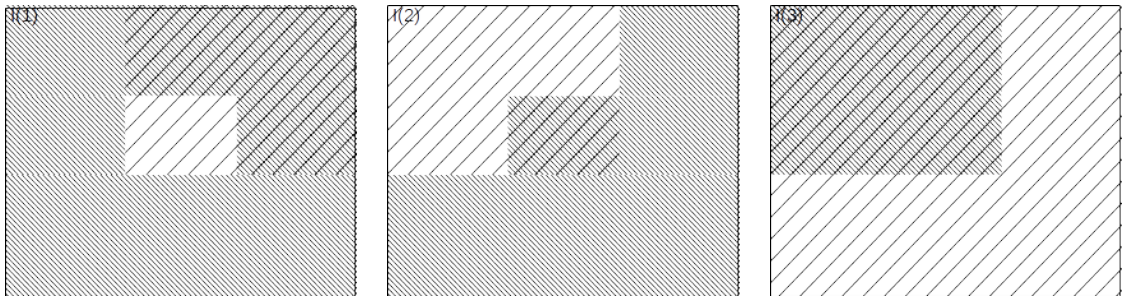
- **El contraste:** para tener imágenes maximizando o minimizando los cambios de contraste, por ejemplo, en imágenes en escala de grises donde sólo hay texto es conveniente tener el máximo contraste entre el texto y el fondo.

- **La nitidez:** para tener imágenes que sean más claras, donde se aprecien fácilmente los detalles, o más suaves (borrosas). Esto último es usado en fotografía o video para resaltar las regiones en las que se busca enfocar la atención, lo cual se logra desenfocando algunas partes de la imagen o fusionando una parte nítida con una borrosa.

Se pueden listar más ejemplos, sin embargo podemos resumir lo anterior en que si se tiene una propiedad que se puede medir y un conjunto de imágenes, entonces podemos realizar un proceso de medición, determinar aquellas partes de las imágenes donde dicha propiedad cumple con lo que se desea y, basados en la medición, determinar un mapa de decisión que permita unir de forma complementaria partes de las imágenes del conjunto para tener una sola imagen que contenga toda la información que nos interesa.

Pongamos un ejemplo en el que se tiene un conjunto de tres imágenes como las mostradas en la Figura 1.7. En cada una de las figuras se pueden observar regiones con dos tipos de achurado uno a 45° y otro a 135° con una densidad mayor, además de regiones donde aparecen ambos tipos. Cada región se usa para representar alguna propiedad general de una región de la imagen. La imagen de la Figura 1.7(a) contiene 3 regiones: una con con el achurado a 45°, otra con achurado a 135° y una con la combinación de ambos achurados. La imagen de la Figura 1.7(b) tiene también tres

regiones pero en diferente posición. La imagen de la Figura 1.7(c) sólo tiene dos regiones: una con el achurado a 45° y otra con la combinación de los dos achurados.



(a) $I(1)$ **(b)** $I(2)$ **(c)** $I(3)$

Figura 1.7: Conjunto con tres imágenes con diferente achurado

Suponiendo que se desea combinar la información de las imágenes del conjunto mostrado en la Figura 1.7, , con la finalidad de obtener una sola imagen que contenga todas las regiones que son de nuestro interés, determinamos primero aquella propiedad a utilizar para discriminar la información o regiones de interés. Supongamos que para este ejemplo la propiedad que hemos elegido para discriminar es la similitud que tiene cada una de las imágenes del conjunto con respecto a la imagen mostrada en la Figura 1.8 (achurado a 45°).

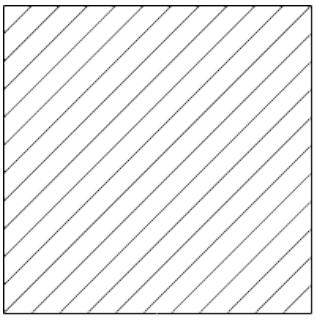


Figura 1.8: Ejemplo de propiedad a maximizar.

Dado que nos interesan aquellas regiones que tienen mayor similitud con un achurado a 45° como el de la Figura 1.8, necesitamos una forma de medir la similitud de

cada pixel de las imágenes. La medida de similitud puede ser obtenida mediante una función o transformación que, aplicada a la imagen en cuestión produzca un valor cuantitativo de la medición de la similitud con la propiedad seleccionada. Al aplicar dicha función o transformación obtenemos un conjunto de matrices del mismo tamaño de las imágenes, al que llamamos $F = \{F(1), F(2), F(3)\}$.

Definimos el problema de fusión como el proceso de extraer las regiones de interés de cada una de las tres imágenes, con el objetivo de formar una nueva imagen J, en la que se maximice la similitud con la imagen de la Figura 1.8.

De acuerdo a los pasos mencionados para llevar a cabo el proceso de fusión, en la Figura 1.9 se muestra el proceso paso a paso para fusionar las imágenes del conjunto mostrado en la Figura 1.7, maximizando la similitud con la imagen de la Figura 1.8. En la columna 1 se muestran las imágenes del conjunto, que son los parámetros de entrada del proceso. En el paso **I** (mostrado en la columna 2) se aplica una transformación a cada imagen para determinar la similitud de cada uno de los pixeles de las imágenes del conjunto con los pixeles de la imagen de muestra. Se ha representado en blanco un mayor valor para la similitud (es muy parecida a la muestra), en gris claro una similitud menor (tiene cierto parecido con la muestra) y en gris obscuro se representa la falta de similitud (no se parece a la muestra). En el paso **II** (ilustrado en la columna 3) se ha generado un mapa de decisión que muestra en color blanco las regiones donde la aptitud de $I(1)$ es mayor que la aptitud de las otras dos imágenes, en gris claro donde la imagen $I(2)$ tiene mayor aptitud que el resto y, finalmente, en gris obscuro las regiones donde el resultado de la función de similitud fue mayor para $I(3)$ que para $I(1)$ e $I(2)$. En el paso **III** (columna 4) se utiliza el mapa de decisión obtenido en el paso **II** para determinar las regiones que se conservarán de cada una de las imágenes del conjunto. Finalmente, en el paso **IV** se combinan las regiones de interés para formar una imagen en la que se maximiza la similitud con la muestra.

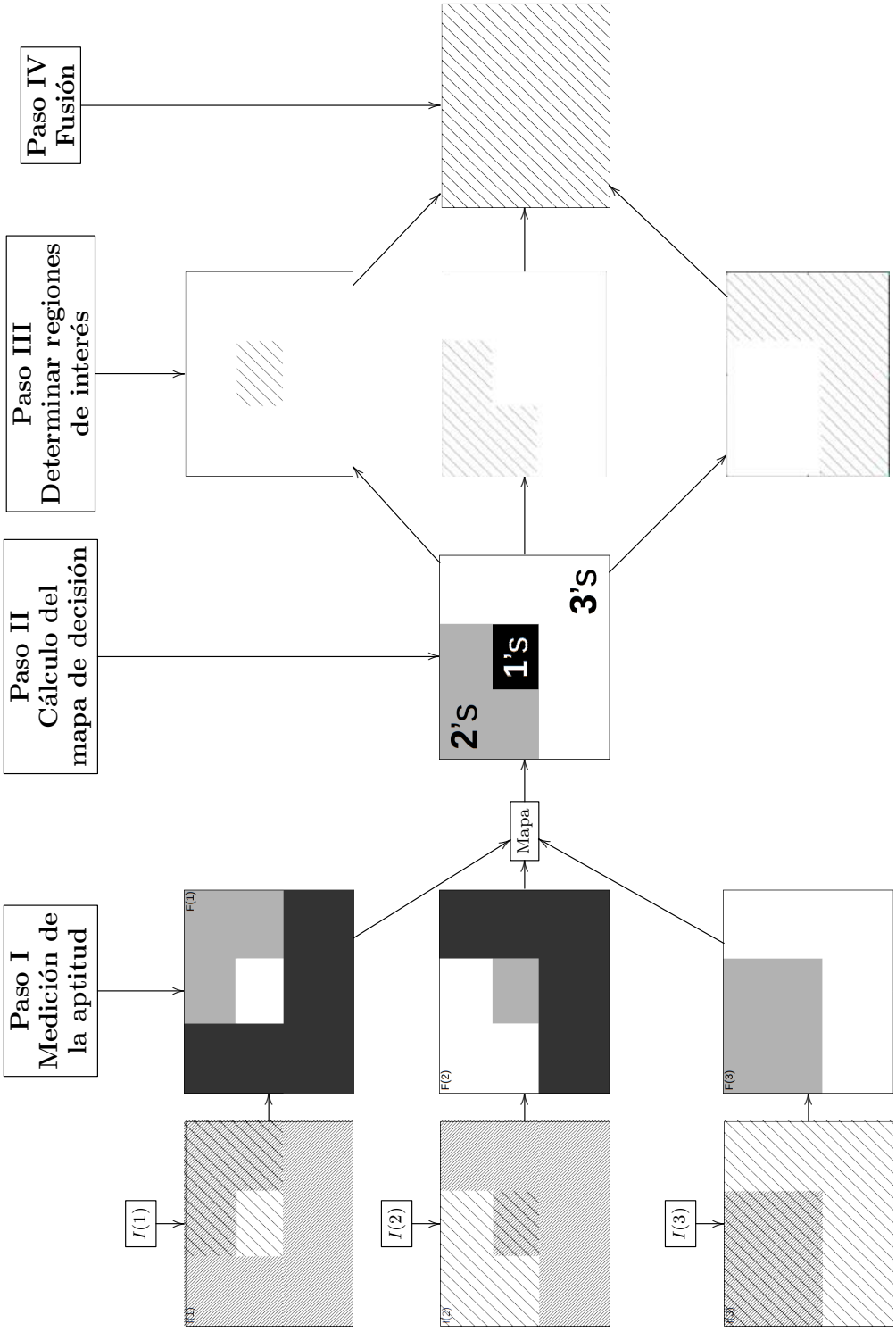


Figura 1.9: Proceso de fusión de tres imágenes multi-foco

Se puede observar que todas las regiones de la imagen final son muy similares a la muestra. Si aplicáramos la función para medir la similitud y sumáramos los valores de la matriz obtenida, el resultado sería un valor mayor que si aplicáramos el mismo procedimiento para cualquiera de las imágenes del conjunto.

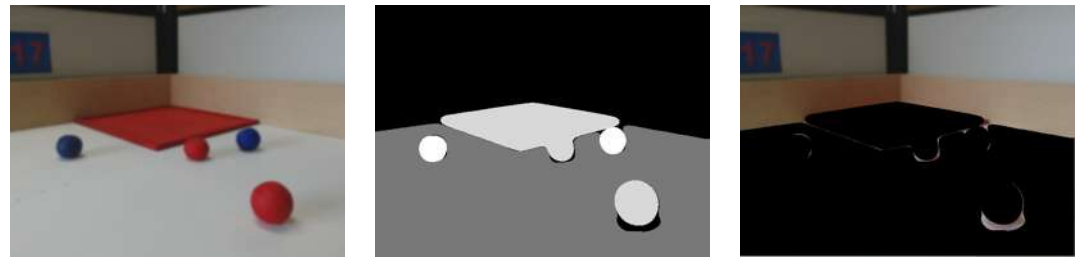
Adicionalmente, dado el mapa de decisión obtenido en el paso **II**, los pasos **III** y **IV** no implican gran complejidad y es posible realizar la fusión combinando las imágenes del conjunto incluso sin realizar el paso **III**. Sin embargo, se ha representado de esta manera para ilustrar que la segmentación es un sub-producto del proceso de fusión.

En la Figura 1.10(a) se muestra una imagen capturada con una cámara de un robot, como el presentado en [Garnica et˜al., 2016], que tiene el objetivo de colocar cada pelota de un tablero plano con fondo blanco en el contenedor de su color. Para realizar dicha tarea se debe clasificar cada pixel para segmentar las pelotas, el fondo y los contenedores, desechando el resto de pixeles. En la Figura 1.10(b) se muestra un mapa de decisión que permite clasificar cada pixel como: sin importancia (negro), fondo (gris obscuro), contenedor y pelotas rojas (gris claro) y pelotas azules (blanco). En las Figuras 1.10(c-f) se muestra la segmentación de cada clase, en base al mapa de decisión.

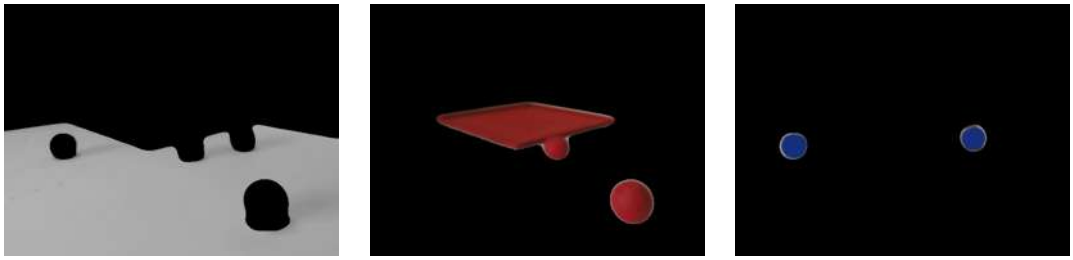
Como se puede observar en el ejemplo mostrado en la Figura 1.10, en base a un mapa de decisión se puede realizar clasificación y segmentación, además de fusión de imágenes. Dicho mapa se puede obtener en base a una función que mida la similitud de cada pixel de la imagen con cada una de las clases que se desean segmentar.

En esta tesis presentamos un algoritmo que permite calcular un mapa de decisión en base a una función que mide la similitud de cada una de las imágenes del conjunto con una muestra de la propiedad que se desea maximizar. Por lo tanto, dicho mapa debe ser aquel que maximice la propiedad deseada en la imagen fusionada, conservando la coherencia espacial y una buena definición en los bordes.

Presentamos evidencia de que el algoritmo propuesto puede ser aplicado a fusión de imágenes en diferentes contextos, por ejemplo, para fusionar imágenes multi-foco



(a) ***Imagen a segmentar*** **(b)** ***Mapa de decisión*** **(c)** ***Sin importancia***



(d) ***Fondo*** **(e)** ***Segmentación del rojo*** **(f)** ***Segmentación del azul***

Figura 1.10: Ejemplo de segmentación de imágenes en base a un mapa de decisión.

o imágenes con iluminación no uniforme. Adicionalmente, es posible aplicar el algoritmo para procesos de segmentación y binarización robusta de imágenes. Dichas aplicaciones del algoritmo se logran cambiando la función que permite medir la propiedad deseada, la cual para imágenes multi-foco mide la nitidez, para imágenes con iluminación no uniforme es una medida de la iluminación y para segmentación mide la similitud de la imagen con cada una de las clases a segmentar. En esencia el algoritmo es el mismo para todas las aplicaciones, cambiando únicamente los parámetros de entrada y la función de aptitud a aplicar en cada proceso.

Para resolver el problema de fusión de imágenes aplicamos técnicas como: la búsqueda binaria, el uso de imágenes integrales y la paralelización para plantear un proceso en el que se obtienen los resultados en décimas de segundo. Por lo tanto, se puede decir que el algoritmo permite su aplicación en tiempo real.

1.3. Motivación

De la revisión del estado del arte sobre procesamiento de imágenes, detectamos la necesidad de un algoritmo eficiente para el cálculo de un mapa de decisión que permita segmentar, fusionar, clasificar y binarizar imágenes de forma robusta.

En la actualidad existen muchos trabajos donde se proponen algoritmos para realizar segmentación, fusión, clasificación, binarización y mejoramiento de imágenes; sin embargo, dichos algoritmos sólo han sido planteados para resolver uno de estos problemas y no conocemos trabajos donde se presente una propuesta que permita resolver los problemas mencionados usando el mismo algoritmo.

La motivación de este trabajo es generar un algoritmo para la fusión de imágenes obtenidas bajo diferentes situaciones, basados en alguna propiedad medible. Dicho algoritmo deberá dar como resultado un mapa de decisión que permita segmentar N imágenes de acuerdo a la medición obtenida de alguna propiedad seleccionada.

1.4. Objetivos de la tesis

El objetivo de esta tesis es presentar un algoritmo de fusión de imágenes aplicable en tiempo real y que pueda ser utilizado para fusión en diferentes condiciones de enfoque o iluminación; permitiendo además, la binarización, la segmentación, la clasificación y el mejoramiento de imágenes, entre otros.

Los objetivos particulares son:

- Elegir la función que permita determinar la aptitud de las imágenes de una forma confiable y rápida; dependiendo de la aplicación en particular.
- Obtener una imagen final con la máxima aptitud posible, con regiones homogéneas y coherentes para una función que permite medir la propiedad establecida.
- Realizar la fusión en tiempo real.

- Generalizar el algoritmo para que pueda aplicarse a cualquier número de imágenes en el conjunto.

- Crear un algoritmo de fusión de imágenes eficiente y general que permita fusionar, binarizar, segmentar y mejorar imágenes e información.

1.5. Descripción de capítulos

Este trabajo consta de 7 capítulos los cuales están distribuidos de la siguiente manera:

- El capítulo 1 describe la importancia del problema, la motivación para abordarlo y los objetivos generales y particulares. La intención de dicho capítulo es coadyuvar a que el lector comprenda de la mejor manera posible el resto de los capítulos.

- El capítulo 2 presenta un análisis del problema de fusión de imágenes multi-foco y un resumen de las técnicas más recientes y prominentes que actualmente se utilizan para resolverlo, mencionando las ventajas y desventajas de cada una de dichas técnicas.

- El capítulo 3 presenta de forma detallada el desarrollo del algoritmo propuesto y su aplicación para la fusión de imágenes multi-foco, se presenta un análisis de complejidad con el que se concluye que el algoritmo tiene una complejidad asintótica lineal con respecto al número de píxeles de la imagen.

- El capítulo 4 presenta la aplicación del algoritmo propuesto para fusionar imágenes con iluminación no uniforme, cambiando únicamente la función de nitidez por una que mide la iluminación de la imagen. Adicionalmente se presenta una modificación del algoritmo de fusión para ser aplicado en la binarización de imágenes con iluminación no uniforme.

- El capítulo 5 presenta una ligera modificación del algoritmo de fusión, para su aplicación en la segmentación de imágenes por color, dadas una imagen original y un conjunto de ejemplos de cada clase. Las modificaciones que sufre el algoritmo de fusión para ser aplicado a la segmentación son mínimas.

- El capítulo 6 muestra evidencia de la eficiencia y exactitud del algoritmo propuesto, para distintos conjuntos de imágenes multi-foco, entre los que se encuentran imágenes sintéticas e imágenes reales. El número de imágenes varía en cada conjunto. Adicionalmente, se presentan ejemplos de resultados obtenidos al aplicar cada uno de los algoritmos para fusionar imágenes multi-foco, fusionar imágenes con iluminación no uniforme, binarizar imágenes con iluminación no uniforme y segmentar imágenes por color.

- El capítulo 7 presenta las conclusiones y las líneas de investigación a seguir en trabajos futuros.

Capítulo 2

Antecedentes de fusión de imágenes multi-foco

El problema de fusión de imágenes consiste en unir información de varias imágenes creando una nueva imagen que está formada con partes de las imágenes fuente. Para realizar la fusión es necesario extraer las regiones de interés de cada una de las imágenes fuente por lo que se deben detectar de forma automática dichas regiones. En el caso particular de la fusión de imágenes multi-foco, la característica que se utiliza para discriminar las partes deseables es la nitidez, definida como la claridad de la imagen o la facilidad con la que se aprecian los detalles de la escena. En este capítulo se presenta información sobre el modelo de la lente, la profundidad de campo como causa de regiones desenfocadas en las imágenes, se discutirá sobre técnicas para medir la nitidez y cómo se pueden aplicar a la fusión, se presenta un método para simular el desenfoque con el fin de crear imágenes sintéticas que nos permitan medir de forma cuantitativa el desempeño del algoritmo de fusión propuesto.

2.1. Modelo óptico de la lente

Las características de las lentes de las cámaras y sistemas de captura de imágenes, impiden obtener una imagen completamente nítida y con todos los detalles de los objetos que se encuentran a diferentes distancias. Al tomar una fotografía se debe decidir en qué objetos enfocaremos la lente para lograr capturar los detalles de dichos objetos con la mayor nitidez; como consecuencia de dicha decisión, el resto de los objetos aparecerán borrosos o desenfocados. La limitación de los sistemas de captura de imágenes basados en lentes está relacionada con la profundidad de campo, la cual es definida como el rango de distancias que la lente de una cámara, con una apertura dada, es capaz de reproducir de manera nítida [Kuthirummal et˜al., 2011]. El modelo matemático de la lente está regido por (2.1), donde f es un parámetro intrínseco de la cámara e indica la distancia focal, d_o es la distancia desde la lente hasta el objeto y d_i es la distancia desde la lente hasta la posición donde se generaría una imagen perfecta, es decir la posición donde la lente concentra la luz reflejada por el objeto. Nótese que si d_o aumenta d_i disminuye y viceversa.

$$\frac{1}{f} = \frac{1}{d_o} + \frac{1}{d_i} \tag{2.1}$$

La Figura 2.1 muestra el modelo de la lente; un punto ρ que se encuentra a una distancia d_o de la lente. La distancia $d_o > PC_1$ lo que significa que dicho punto está fuera de la profundidad de campo (PC) comprendida entre PC_0 y PC_1. Puede verse que en este caso $d_i < d_s$ por lo que los rayos de luz se cruzan antes de alcanzar el sensor y generan un círculo de luz con diámetro b, el cual está en función de la apertura a, la distancia a la imagen d_i y la distancia al sensor d_s de acuerdo con (2.2).

$$b = \frac{a}{d_i}|d_i - d_s| \tag{2.2}$$

De acuerdo con (2.2), cuando $d_i = d_s$ el diámetro del círculo es $b = 0$, generándose un punto en lugar de un círculo. Sin embargo, cuando un objeto se encuentra fuera

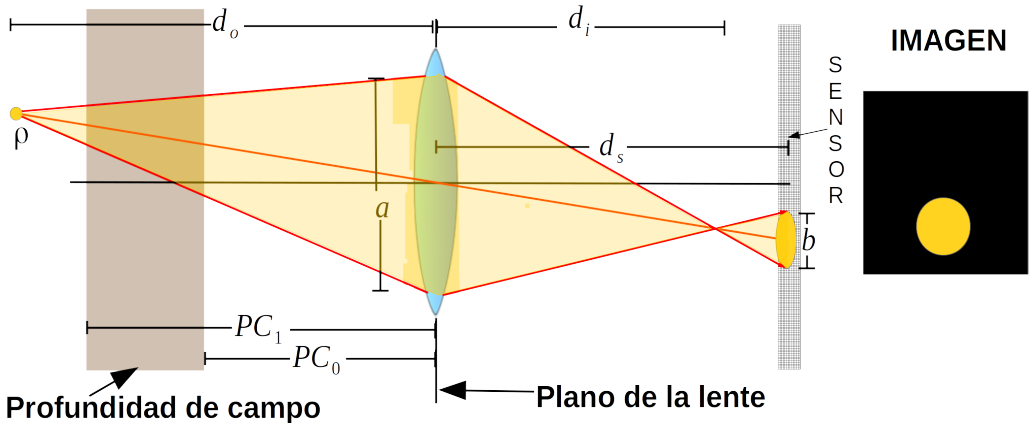


Figura 2.1: Esquema del modelo de la lente

de la profundidad de campo, la diferencia dada por $|d_i - d_s|$ aumenta provocando que se genere un círculo de color en el sensor, cuando lo deseable es que solamente se genere un punto. Los círculos generados cuando $d_i \neq d_s$ son conocidos como círculos de confusión [Sezan et˜al., 1991]. Cuando la diferencia entre d_i y d_s crece, el diámetro del círculo de confusión será mayor mientras que si $d_i = d_s$ significa que el sensor está a la distancia donde se genera la imagen perfecta y no existen los círculos de confusión.

La Figura 2.2 e muestra tres casos correspondientes a un punto de la escena el cual es colocado a diferentes distancias de la lente. En el primer caso, en la Figura 2.2(a), el punto ρ' está a una distancia $d'_o < PC_0$ provocando que no haya una distancia suficiente para que la lente concentre los rayos de luz reflejados por dicho punto en una única posición en el sensor. Esto genera un círculo de color debido a que $d'_i > d_s$. En el segundo caso, en la Figura 2.2(b) el punto ρ'' está a una distancia $PC_0 < d''_o < PC_1$ y $d''_i = d_s$; esto permite que la lente concentre la luz reflejada por el punto ρ'' en una sola posición del sensor produciendo un punto en la imagen generada, es decir el diámetro del círculo de confusión tiende a 0. Finalmente, en el tercer caso, mostrado en la Figura 2.2(c), el punto ρ''' está a una distancia $d'''_o > PC_1$; en consecuencia, $d'''_i < d_s$ y la lente concentra la luz reflejada por el punto antes de

alcanzar el sensor y el cruce de los rayos de luz inciden sobre varias posiciones de éste, generando un círculo similar al primer caso.

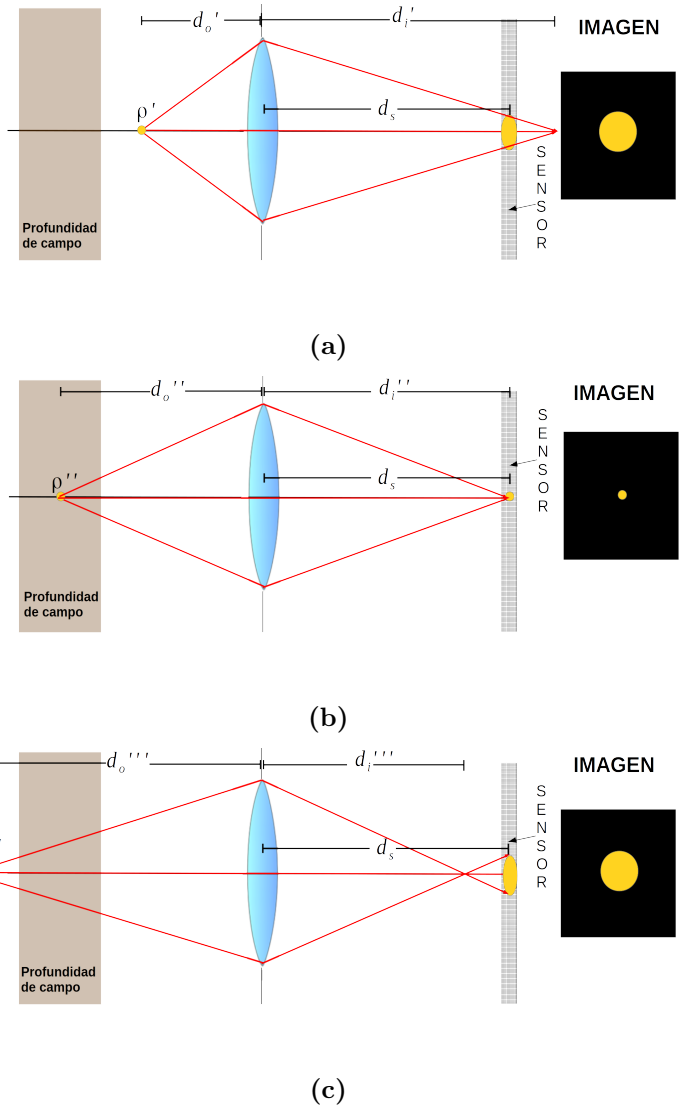
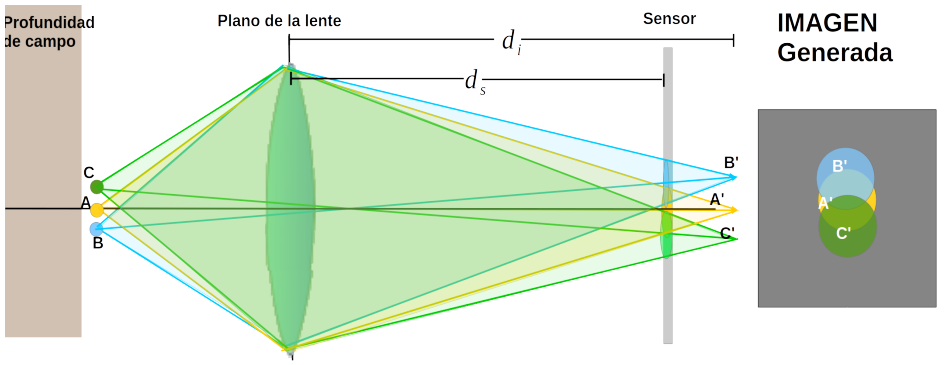


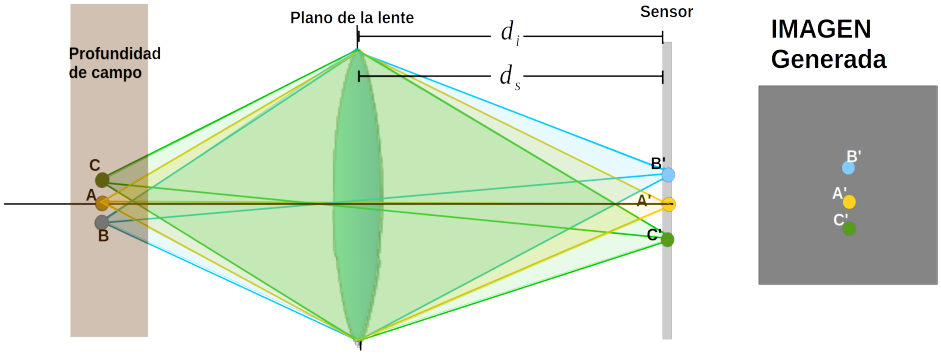
Figura 2.2: Círculos de confusión generados por un punto a diferentes distancias de la lente.

El color de los pixeles de la imagen generada que son afectados por los círculos de confusión es el resultado de una mezcla de los colores que corresponden a varios puntos de la escena, en lugar de ser el resultado del color de un sólo punto. Esta situación provoca que dichos pixeles aparezcan borrosos o desenfocados.

La Figura 2.3 muestra dos casos con tres puntos. En el primer caso, Figura 2.3(a), se ilustra una situación donde los tres puntos están fuera de la profundidad de campo ($d_i > d_s$) provocando que se generen tres círculos de distinto color y el traslape de dichos círculos causa que los colores en la imagen no correspondan a los de la escena, por lo tanto la imagen luce borrosa. En el segundo caso, mostrado en la Figura 2.3(b), se ha modificado la distancia d_s acercando la lente al sensor, por lo que la profundidad de campo se mueve causando que los puntos queden dentro de ésta y, dado que $d_s = d_i$, no hay círculos de confusión en la imagen generada y aparecen los mismos puntos de la escena.



(a)



(b)

Figura 2.3: Círculos de confusión generados por puntos fuera de la profundidad de campo.

Dado que en la mayoría de los casos no es posible capturar de forma nítida todos

los objetos de una escena en una única imagen, será necesario tomar un conjunto de N imágenes $I = \{I(1), I(2), \ldots, I(N-1), I(N)\}$, cada una de ellas enfocada en los objetos que se encuentren a la misma distancia. A este conjunto de imágenes lo denominaremos conjunto de imágenes multi-foco, donde la $k-$ésima imagen $I(k)$ es un arreglo bi-dimensional de tamaño $n_r \times n_c$ y cada coordenada (r, c) almacena un pixel en el renglón r y la columna c.

La Figura 2.4 muestra el ejemplo de un conjunto de tres imágenes multi-foco que corresponden a una escena donde aparece un grupo de lobos marinos. La Figura 2.4(a) muestra de manera nítida el lobo marino que aparece al frente de la escena, la Figura 2.4(b) muestra de forma nítida las regiones donde aparecen el resto de los lobos marinos. Finalmente, la Figura 2.4(c) muestra de manera nítida el fondo de la escena. Identificaremos estas imágenes como $I(1)$, $I(2)$ e $I(3)$, respectivamente. Para obtener dichas imágenes se modificó la posición de la lente haciendo zoom para enfocar en diferentes distancias.

(a) $I(1)$ **(b)** $I(2)$ **(c)** $I(3)$

Figura 2.4: Tres imágenes multi-foco de unos lobos marinos

En el caso de las imágenes multi-foco nos interesa medir la nitidez de cada imagen y, en base a dicha medida, calcular el mapa de decisión. Por lo tanto se requiere definir una función de nitidez que arroje valores altos para aquellos pixeles donde hay mayor nitidez y valores cercanos a cero donde la imagen carece de dicha propiedad.

2.2. Mapa de decisión

Dada una función de nitidez que se aplica a cada una de las imágenes del conjunto, se requiere determinar las regiones o pixeles que se extraerán de cada una ellas para formar la imagen fusionada. Para tal fin, formamos una matriz P, del mismo tamaño de las imágenes, donde cada $P_{r,c}$ contiene etiquetas que hacen referencia a aquella imagen en la que la nitidez es mayor que en cualquiera de las otras imágenes. La matriz P será llamada mapa de decisión y puede ser calculada para una coordenada (r, c) con (2.3) cuando el conjunto tiene 2 imágenes, y de forma general para N imágenes se puede obtener mediante (2.4).

$$P_{r,c} = \begin{cases} 1 & \text{si } F_{r,c}(1) \geq F_{r,c}(2) \\ 2 & \text{En otro caso} \end{cases} \tag{2.3}$$

donde $F_{r,c}(k)$ es una medida de la calidad del pixel en la coordenada r, c de la k-ésima imagen. En este caso particular, $F_{r,c}(k)$ es una medida de nitidez del pixel en la coordenada r, c de la k-ésima imagen.

$$P_{r,c} = \begin{cases} 1 & \text{si } F_{r,c}(1) \geq F_{r,c}(j) \quad \forall j \neq 1 \\ 2 & \text{si } F_{r,c}(2) \geq F_{r,c}(j) \quad \forall j \neq 2 \\ \vdots & \vdots \\ N & \text{si } F_{r,c}(N) \geq F_{r,c}(j) \quad \forall j \neq N \end{cases} \tag{2.4}$$

Adicionalmente, el resultado de la función de nitidez de la imagen fusionada debe ser el mayor valor posible. Entonces, el mapa de decisión debe ser aquel que maximiza la nitidez de la imagen fusionada, considerando situaciones como el ruido, la falta de textura y otras complicaciones inherentes al problema.

2.3. Simulación del desenfoque

De acuerdo a lo planteado por Elder y Zucker, el desenfoque puede modelarse mediante la convolución de la imagen con un kernel Gaussiano dado por (2.5), [Elder y Zucker, 1998; Bae y Durand, 2007; Riaz et˜al., 2008].

$$G_{r,c} = g_r g_c \tag{2.5}$$

donde g_r es la ecuación de una Gaussiana, en una dimensión, con media μ y desviación estándar σ definida en (2.6).

$$g_r \equiv g(r|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(r-\mu)^2}{2\sigma^2}\right) \tag{2.6}$$

Entonces, dada una imagen enfocada E y un kernel Gaussiano podemos calcular una imagen desenfocada D con (2.7).

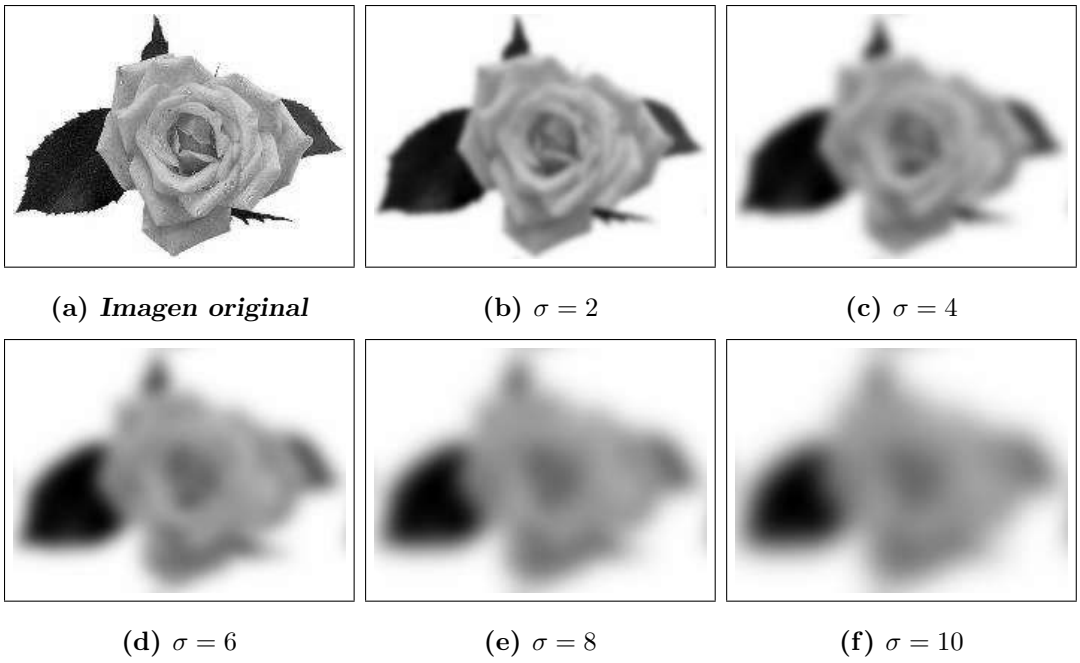
$$D_{r,c} = E_{r,c} * G_{r,c} \tag{2.7}$$

donde el símbolo $*$ representa la convolución.

Según Riaz *et al* aplicando el kernel Gaussiano se atenúan las frecuencias altas mientras que las frecuencias bajas no sufren cambio, [Riaz et˜al., 2008]. El problema de desenfoque o emborronamiento puede plantearse como la incapacidad de la lente de reproducir las frecuencias altas de una región de la escena, por lo que podemos modelar los círculos de confusión producidos por una lente convolucionando la imagen con el kernel obtenido con la ecuación (2.5). El proceso de convolucionar la imagen con una Gaussiana puede interpretarse como el hecho de crear círculos de confusión, cuyo diámetro varía en función de la desviación estándar σ, por lo que cada pixel es recalculado como una combinación de los valores de los pixeles que están dentro del círculo de confusión hipotético y cada pixel resultará de una mezcla del color de los pixeles vecinos.

Para ejemplificar el proceso de desenfoque, convolucionamos la imagen mostrada en la Figura 2.5(a) con un conjunto de Gaussianas con diferente desviación estándar

σ obtenidas usando (2.5). Los resultados de la convolución se muestran en las Figuras 2.5(b)-2.5(f). Nótese que si el valor de σ aumenta la nitidez disminuye y viceversa.



(a) ***Imagen original*** **(b)** $\sigma = 2$ **(c)** $\sigma = 4$

(d) $\sigma = 6$ **(e)** $\sigma = 8$ **(f)** $\sigma = 10$

Figura 2.5: Simulación del desenfoque convolucionando con un kernel Gaussiano.

2.4. Creación de conjuntos de imágenes multi-foco sintéticas

Dadas una imagen original enfocada E y una imagen borrosa D, obtenida con (2.7), se puede crear un conjunto de imágenes multi-foco sintéticas. Las cuales, dado que conocemos la forma en la que fueron creadas, nos permitirán medir el desempeño de la técnicas para la detección de regiones nítidas. Como ejemplo se utilizarán las imágenes E y D que se muestran en las Figuras 2.6(a) y 2.6(b), respectivamente, para crear un par de imágenes multi-foco sintéticas. Cabe señalar, que el desenfoque generalmente ocurre sólo sobre algunas regiones de la imagen. Para lograr tal efecto definimos el mapa de decisión $\hat{P}$ como el mostrado en la Figura 2.6(c), donde se

muestran en blanco los valores $\hat{P}_{r,c} = 2$ y en negro donde $\hat{P}_{r,c} = 1$. Dicho mapa tiene la etiqueta de la imagen que se conservará nítida en esa coordenada de acuerdo con (2.8); es decir, para formar la imagen $I(1)$ se conservarán nítidas las regiones donde $\hat{P}_{r,c} = 1$ y borrosas el resto, mientras que, para formar la imagen $I(2)$ se conservarán nítidas las regiones donde $\hat{P}_{r,c} = 2$ y borrosas el resto. En general, cada imagen $I(k)$ tendrá nítidas las regiones donde $\hat{P}_{r,c} = k$ y borrosas el resto.

(a) ***Imagen de lobos*** *(E)* **(b)** ***Imagen borrosa*** *(D)* **(c)** ***Mapa de decisión*** *($\hat{P}$)*

Figura 2.6: Generación de una imagen borrosa sintética

$$I(k)_{r,c} = \begin{cases} E_{r,c} & \text{Si } \hat{P}_{r,c} = k \\ D_{r,c} & \text{en otro caso.} \end{cases} \tag{2.8}$$

Dados la imagen enfocada E, la imagen borrosa D y el mapa $\hat{P}$, mostrados en las Figuras 2.6(a), 2.6(b) y 2.6(c), respectivamente, aplicando (2.8) con $k = 1, 2$ se obtiene un par de imágenes multi-foco, $I(1)$ e $I(2)$ como las mostradas en la Figura 2.7, las cuales estarán desenfocadas/enfocadas parcialmente de manera complementaria, es decir, las partes nítidas de $I(1)$ aparecerán desenfocadas en $I(2)$ y viceversa. El mapa de decisión mostrado en la Figura 2.6(c) fue establecido tratando de aproximar lo mejor posible los bordes del lobo que aparece al frente en la imagen de la Figura 2.6(a). Lo anterior se realizó con el fin de simular que la Figura 2.7(a) fue capturada enfocando la cámara en el lobo del frente y, en consecuencia, el lobo de atrás y el

fondo aparecen borrosos; mientras que, en la Figura 2.7(b), se simula fue capturada con las condiciones de enfoque inversas. Con estas imágenes se pretende clarificar el procedimiento para generar conjuntos de imágenes multi-foco sintéticas a partir de una original que está enfocada.

(a) *Imagen sintética* $I(1)$ **(b) *Imagen sintética*** $I(2)$

Figura 2.7: Par de imágenes multi-foco sintéticas

Dado lo anterior podríamos pensar que si el desenfoque se puede reproducir mediante convolución, entonces se debería poder corregir con un proceso de deconvolución. Por ejemplo, Yuan *et al* presentan el uso de técnicas de deconvolución para la recuperación de una imagen original desde una imagen que ha sido degradada, [Yuan et˜al., 2008]. Starck y Murtagh plantean el uso de técnicas de deconvolución, acompañadas de técnicas de regularización y procesamiento de las imágenes en diferentes escalas, para mejorar imágenes astronómicas, controlando por un lado, la suavidad y por otro lado el ruido, [Starck y Murtagh, 2006]. Zhou *et al.* presentan la aplicación de la deconvolución con diversas condiciones de contorno, para recuperar una imagen original desde una imagen que ha sufrido emborronamiento y adición de ruido, [Zhou et˜al., 2014]. Entonces, parece que basta con averiguar el valor de la desviación estándar σ y aplicar técnicas de deconvolución sobre las imágenes para mejorar la nitidez, sin embargo, debemos de considerar que el desenfoque es sólo sobre algunas regiones de la imagen y que no tenemos información de los valores de desviación

estándar σ adecuados. Más aún, el desenfoque no es el mismo en todos los pixeles de la imagen, por lo que, aún conociendo un valor de σ, no podríamos resolver el problema de fusión de imágenes multi-foco de esa manera.

2.5. Medición del nivel de enfoque

Para calcular el valor adecuado de $P_{r,c}$ es necesario tener una forma de medir la nitidez o aptitud $F(k)$ de cada imagen, a fin de poder determinar cuál es la imagen que más nos conviene tomar para formar la imagen fusionada, de tal forma que $P_{r,c} = k$ si $F_{r,c}(k) > F_{r,c}(j)\ \forall j \in [1, 2, \ldots, N], j \neq k$.

En el estado del arte se presentan varias propuestas para medir el enfoque, donde se encuentran métodos basados en el análisis de frecuencias para decidir qué áreas extraer de cada una de las imágenes para crear la imagen fusionada. Dichos trabajos basan su investigación en el hecho de que las regiones que presentan mayor nitidez en una imagen, tienen también mayor respuesta a las altas frecuencias que aquellas que aparecen de manera borrosa o desenfocada. En resumen, las diferentes formas de medir la nitidez de una imagen que encontramos se basan ee: la energía del gradiente, la energía del Laplaciano y sus variantes, la varianza, la entropía, el momento central absoluto o la frecuencia espacia. Dichas medidas están basadas en el análisis de qué tan variante es la imagen. Dicha variación se calcula como la diferencia entre el pixel y sus vecinos (primeras o segundas derivadas), como diferencia entre el pixel y la media o midiendo de alguna manera qué tan variante es la imagen. Por otro lado, sabemos que un filtro pasa-altas nos permite medir también (aunque de manera indirecta), las variaciones en la imagen, pues un valor con una magnitud alta indica una imagen muy variante, mientras que, si la magnitud de la respuesta al filtro se acerca a cero será un indicador de que la imagen es poco variante y en consecuencia posee poca información de alta frecuencia.

Fisher y Eskicioglu en [Eskicioglu y Fisher, 1995] y Li *et al.* en [Li et˜al., 2001]

presentan una técnica para medir el nivel de actividad de una imagen o bloque (región de la imagen) mediante el uso de la frecuencia espacial, para lo cual se hace el cálculo de la diferencia entre cada pixel (r, c) y sus dos pixeles vecinos, del renglón anterior y de la columna anterior. Lo cual, es una forma de aproximar las primeras derivadas discretas de una imagen $\mathcal{M}$ en las direcciones x y y que están dadas por (2.9) y (2.10), respectivamente.

$$\frac{\partial \mathcal{M}_{r,c}}{d_x} = \mathcal{M}_{r,c} - \mathcal{M}_{r,c-1} \tag{2.9}$$

$$\frac{\partial \mathcal{M}_{r,c}}{d_y} = \mathcal{M}_{r,c} - \mathcal{M}_{r-1,c} \tag{2.10}$$

Algunos trabajos como [Li et˜al., 2001; Pajares y de˜la Cruz, 2004; Zhou et˜al., 2006; Li y Yang, 2008; Cao et˜al., 2015] proponen el uso de la frecuencia espacial para calcular el nivel de actividad o enfoque de un bloque o pixel.

Riaz *et al* plantean que existen distintas formas de medir el nivel de enfoque, también establecen que los filtros pasa-altas son muy efectivos para este fin y que una técnica para filtrar las frecuencias altas es usar las segundas derivadas, en particular el operador Laplaciano, [Riaz et˜al., 2008]. El Laplaciano se define como la suma de las segundas derivadas en la dirección x y y de una función f en dos dimensiones, de acuerdo con (2.11).

$$L = \frac{\partial^2 f}{\partial x} + \frac{\partial^2 f}{\partial y} \tag{2.11}$$

El uso del Laplaciano completo con kernels de derivadas gaussianas implica un consumo alto de recursos cuando el valor de σ crece, por lo que se han buscado formas de aproximar dicha función. Dentro de estas formas, podemos mencionar: uso de kernels discretizados y la diferencia de Gaussianas (DoG por sus siglas en inglés) que según [Assirati et˜al., 2014] es una muy buena aproximación del Laplaciano.

Al convolucionar la imagen con el kernel del Laplaciano discretizado h, dado por (2.12), se obtiene el mismo resultado que al calcular la suma de las diferencias del pixel con cada uno de sus 8 pixeles vecinos. Dicho kernel puede ser considerado como un medidor del nivel de actividad en las 8 direcciones y es utilizado para medir la

calidad o enfoque de la imagen en [Riaz et˜al., 2008; Calderon y Garnica, 2014; Calderon et˜al., 2016, 2017].

$$h = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.12}$$

Otro filtro que, a pesar de ser un filtro pasa banda, se puede entonar para ser usado como filtro pasa altas es el filtro de Gabor, el cual ya ha sido utilizado en procesamiento de imágenes, por ejemplo, en [Liu et˜al., 2014a] se ha usado como detector de bordes y en [Dunn y Higgins, 1995] se usa para segmentar texturas.

En las Figuras 2.8(a), 2.8(b) y 2.8(c) se muestra el valor absoluto de la respuesta a los filtros de Gabor, a la diferencia de Gaussianas y al Laplaciano discretizado, respectivamente. Estos filtros se utilizan comúnmente para identificar las altas frecuencias. Se utiliza el valor absoluto debido a que nos interesa conocer la magnitud de la respuesta al filtro (sin importar su signo) puesto que un valor con magnitud alta significa que hay una mayor variación en la imagen y, en consecuencia, una mayor nitidez. En dichas imágenes se representan en negro los valores altos y en blanco los valores cercanos a cero.

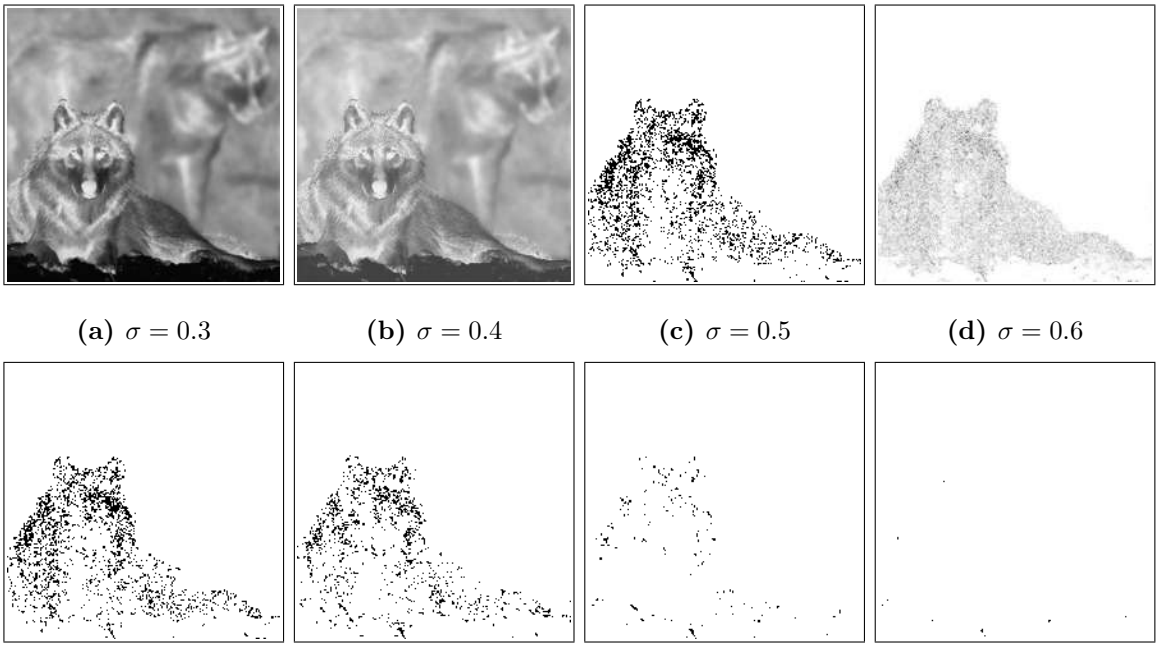
(a) ***Filtro de Gabor*** **(b)** ***Filtro de la DoG*** **(c)** ***Laplaciano discretizado***

Figura 2.8: Respuesta de algunos filtros pasa altas para la imagen de la Figura 2.7(a)

Nótese que en las regiones dónde la fotografía de la Figura 2.7(a) tiene poca nitidez

(el fondo y el lobo que aparece en el fondo de la escena) la respuesta ante cualquiera de los filtros pasa altas 2.8(a), 2.8(b) y 2.8(c) tiene valores cercanos a cero. En cambio en las regiones con mayor detalle o con mayor nitidez, la respuesta a los filtros pasa altas tiene una magnitud mayor, lo que refuerza la teoría de que usando un filtro pasa altas se pueden identificar las regiones con mayor nitidez en una imagen.

Como ejemplo de la medición del nivel de enfoque usando el operador Laplaciano, se ha aplicado dicho operador sobre la imagen de los lobos mostrada en la Figura 2.7(a). Dado que para el cálculo de las derivadas Gaussianas se requiere establecer el valor de la desviación estándar σ, hemos realizado pruebas para establecer el valor de σ que ofrece un mejor resultado, a efectos de detectar las partes nítidas de la imagen. En la Figura 2.9 se muestra el valor absoluto de la respuesta de aplicar el Laplaciano de Gauss sobre la imagen 2.7(a), a medida que se varía el valor de σ usado para el cálculo de las derivadas Gaussianas.



(a) $\sigma = 0.3$ **(b)** $\sigma = 0.4$ **(c)** $\sigma = 0.5$ **(d)** $\sigma = 0.6$

(e) $\sigma = 0.7$ **(f)** $\sigma = 0.8$ **(g)** $\sigma = 0.9$ **(h)** $\sigma = 1.0$

Figura 2.9: Nivel de enfoque utilizando el Laplaciano de Gauss (LoG) al variar σ.

La Figura 2.10 muestra el valor absoluto de la respuesta de aplicar el kernel h mostrado en (2.12) a la imagen mostrada en la Figura 2.7(a).

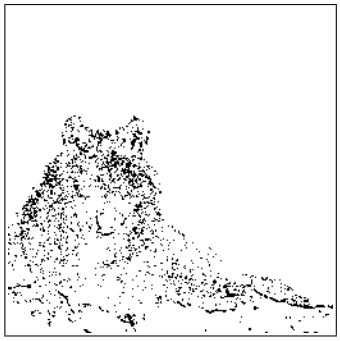
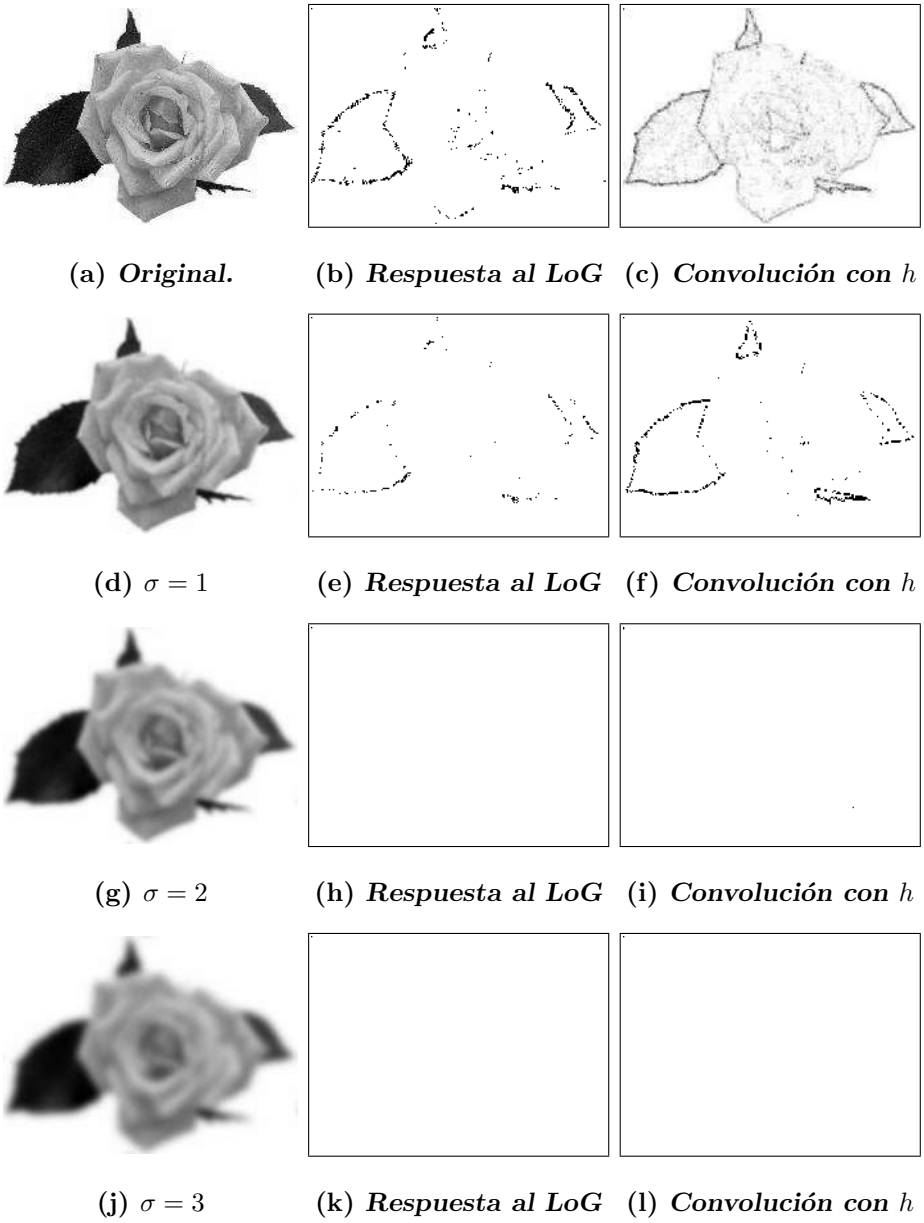


Figura 2.10: Nivel de enfoque utilizando el kernel discretizado del Laplaciano.

La Figura 2.11 muestra tres columnas, en la columna de la izquierda se muestra la imagen de una flor, correspondiente a la mostrada en la Figura 2.5(a), en la que se ha simulado el desenfoque mediante la convolución con una Gaussiana variando el valor de desviación estándar σ. En la segunda columna se muestra el valor absoluto de la respuesta al Laplaciano de Gauss (LoG) aplicado a la imagen de la primera columna, usando $\sigma = 0.8$ para el cálculo de las derivadas Gaussianas. Finalmente, en la tercera columna se muestra el valor absoluto de la respuesta de convolucionar la imagen de la primera columna con el kernel del Laplaciano discretizado h, dado por (2.12). Se representan los valores altos en negro y los valores cercanos a cero con blanco. Se puede observar que a medida que la imagen de la flor es más borrosa, la magnitud de los valores de la respuesta para ambos filtros disminuye, esto debido a que la nitidez se pierde.

Al observar la Figura 2.11, podemos notar que el kernel de convolución del Laplaciano discretizado también permite detectar las regiones nítidas en una imagen, además de que dicho kernel no requiere de establecer parámetro alguno para su aplicación y que, debido a su tamaño, la convolución es muy rápida; mientras que, el tiempo requerido para el cálculo del Laplaciano de Gauss se incrementa a medida que aumenta el valor de σ utilizado en el cálculo de las derivadas Gaussianas.



(a) ***Original.*** **(b)** ***Respuesta al LoG*** **(c)** ***Convolución con*** h

(d) $\sigma = 1$ **(e)** ***Respuesta al LoG*** **(f)** ***Convolución con*** h

(g) $\sigma = 2$ **(h)** ***Respuesta al LoG*** **(i)** ***Convolución con*** h

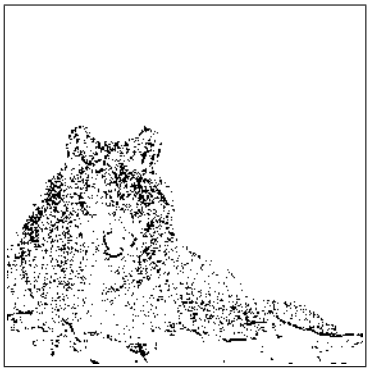
(j) $\sigma = 3$ **(k)** ***Respuesta al LoG*** **(l)** ***Convolución con*** h

Figura 2.11: Respuesta al Laplaciano de Gauss (LoG) y al Laplaciano discreto

La Figura 2.12 muestra el resultado de convolucionar las imágenes mostradas en las Figuras 2.7(a) y 2.7(b) con el kernel Laplaciano h. En dichas imágenes se puede observar que al convolucionar h con $I(1)$ se detectan las regiones enfocadas. Dichas regiones se muestran en negro en la Figura 2.12(b) mientras que en la Figura 2.12(d)

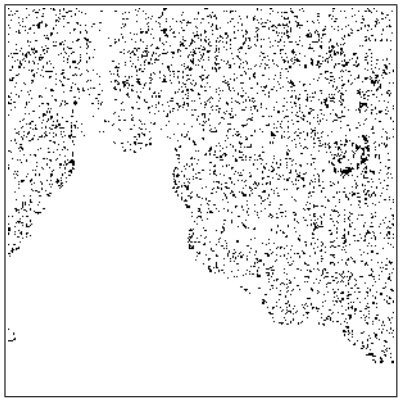
se muestra el resultado aplicar convolucionar h con la imagen $I(2)$.

$$* \quad \begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix} \quad =$$



(a) $I(1)$ **(b)** $F(1)$

$$* \quad \begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix} \quad =$$



(c) $I(2)$ **(d)** $F(2)$

Figura 2.12: Respuesta de aplicar el kernel discretizado del Laplaciano a imágenes multi-foco.

En la Figura 2.13 se muestra el proceso de fusión para el conjunto de imágenes multi-foco de la Figura 2.4. Se pueden distinguir cuatro pasos que consisten en: 1) medir la calidad (nitidez) de cada imagen para obtener $F = \{F(1), F(2) \ldots, F(N)\}$, 2) en base a dicha medida, calcular el mapa de decisión, 3) en base al mapa calculado, extraer las regiones de interés de cada imagen y 4) fusionar las regiones extraídas en una única imagen a la que llamamos imagen fusionada.

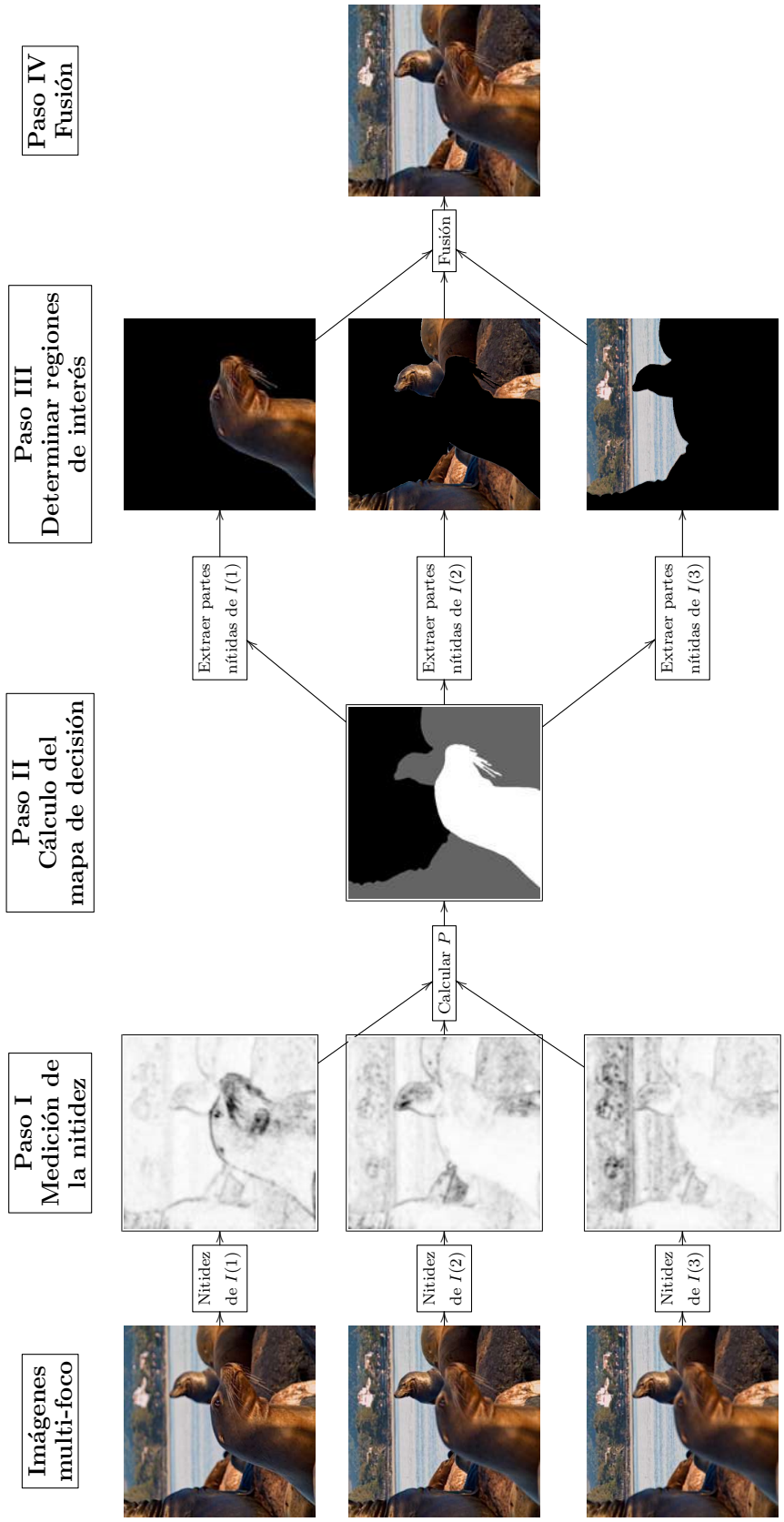


Figura 2.13: Proceso de fusión de tres imágenes multi-foco

2.6. Estado del arte

Existen varias propuestas en la literatura para resolver el problema de fusión de imágenes multi-foco. Algunos de ellos son muy simples, produciendo la imagen fusionada a partir del promedio de las imágenes en el conjunto. Otras propuestas, más elaboradas, usan redes neuronales artificiales [Ma et˜al., 2011; Pagidimarry y Babu, 2011; Yang et˜al., 2017], evolución diferencial [Aslantas y Kurban, 2010], Transformada Discreta Wavelet (WDT por sus siglas en inglés) [Pajares y de˜la Cruz, 2004; Zhang y Blum, 1999; Yang, 2011; Shah et˜al., 2013], Transformada Contourlet [Shah et˜al., 2013; Yang et˜al., 2015], programación paralela [Liu et˜al., 2017; Bejinariu et˜al., 2013], etc.

Otras soluciones asignan un peso a cada pixel de las imágenes en el conjunto y la imagen fusionada se calcula como una combinación lineal ponderada de las imágenes multi-foco. En algunos casos, para pares de imágenes, estos pesos son binarios y se denominan mapas de decisión [Nejati et˜al., 2015].

Los algoritmos de fusión para imágenes multi-foco se pueden dividir en cuatro grupos, de acuerdo con la forma en que se procesan las imágenes. Estos cuatro grupos son: procesamiento a nivel de pixel [Lewis et˜al., 2007; Aslantas y Toprak, 2014; Li et˜al., 2017; Shreyamsha˜Kumar, 2013, 2015], procesado con bloques de tamaño fijo [Aslantas y Kurban, 2010; Zhang et˜al., 2016], utilizando regiones homogéneas [Li y Yang, 2008; Pramanik et˜al., 2013], y utilizando bloques o ventanas de tamaño variable [Bai et˜al., 2015; De y Chanda, 2013]. Cada uno de los cuatro enfoques anteriores tiene sus ventajas y desventajas. El procesamiento a nivel de pixeles requiere un proceso adicional para eliminar inconsistencias en el mapa de decisión. El procesamiento a nivel de bloque tiene la desventaja de que es necesario decidir a priori el tamaño del bloque. En consecuencia, si el tamaño de bloque seleccionado es muy grande, la definición en los bordes se pierde; al disminuir el tamaño del bloque, el resultado obtenido será similar a los resultados obtenidos por el procesamiento a nivel

de pixeles. En trabajos donde el proceso se realiza en regiones homogéneas es necesario un proceso de segmentación previo para encontrar las regiones homogéneas y, por lo general, dicho proceso es complejo. El cálculo del tamaño de bloque de forma adaptable resuelve los problemas inherentes que se encuentran en el procesamiento con bloques de tamaño fijo y el procesamiento a nivel de pixeles; sin embargo, introduce el problema de encontrar un tamaño de bloque apropiado. Los trabajos más destacados y recientes presentados en el estado del arte sobre la fusión de imágenes con multi-foco, se listan a continuación.

Zhang *et al.* [Zhang et˜al., 2016] descomponen imágenes extrayendo información de textura y la utilizan en un sistema de ventanas deslizantes para producir la imagen fusionada. Sin embargo, el tiempo de procesamiento requerido por el algoritmo no permite su aplicación en tiempo real.

Kong y Lei [Kong y Lei, 2017] utilizan un sistema de ventanas deslizantes y diccionarios de descomposición espacial para calcular los mapas de decisión. Su trabajo no informa ningún mapa de decisión y los tiempos de ejecución son superiores a 7 minutos.

Farid *et al.* [Farid et˜al., 2019] desarrolló un algoritmo para la fusión de imágenes multi-foco utilizando Content Bluring Adaptatible (CAB) para detectar las regiones más nítidas. Ellos calculan la imagen fusionada con un mapa de segmentación con valores iniciales obtenidos por medio de la diferencia absoluta entre la imagen original y la imagen borrosa de CAB. El mapa de segmentación se refina utilizando operadores morfológicos y corte de gráfico, luego se utiliza para calcular la imagen fusionada. Farid *et al.* comparan su algoritmo con otros catorce algoritmos del estado del arte, utilizando el conjunto de datos de imagen Lytro. Su implementación utiliza bloques de tamaño fijo para todos los pixeles de la imagen y se aplica a pares de imágenes. En nuestra opinión, el tamaño del bloque no debe ser el mismo para todos los pixeles de la imagen.

Mansour *et al.* [Nejati et˜al., 2015] proponen obtener la imagen fusionada utilizan-

do un diccionario disperso. Ellos calcularon la medida de enfoque para cada imagen extrayendo una característica de enfoque para cada pixel; después de eso, se aplica un proceso de correlación utilizando ese diccionario para obtener el mapa de decisión inicial. Este mapa de decisión se procesa mediante el filtrado guiado y un proceso de optimización MFR, obteniendo un mapa de decisión final, que se utiliza para fusionar las imágenes.

2.7. Conclusiones del capítulo

Aunque existen varias propuestas en el estado del arte, para resolver el problema de fusión de imágenes, no conocemos alguna propuesta que permita realizar la fusión de conjuntos de imágenes multi-foco con más de dos imágenes en tiempo real.

Esta tesis se enfoca en proponer un algoritmo que permita la fusión de conjuntos de imágenes multi-foco y que, además, pueda ser aplicado para la fusión de otros tipos de imágenes, así como para la segmentación robusta, binarización robusta, etc. Esto se puede realizar si se tiene una propiedad de interés medible mediante una función. Con el resultado de dicha función se calcula un mapa de decisión, el cual maximiza la nitidez de la imagen fusionada, sin perder coherencia espacial y garantizando una correcta segmentación en los bordes, todo ello con tiempos de ejecución que permiten la aplicación del algoritmo en tiempo real.

Capítulo 3

Fusión de imágenes multi-foco

Definimos la imagen fusionada como una combinación de las imágenes del conjunto multi-foco y el valor de cada pixel $J_{r,c}$ se calcula mediante la función lineal a pedazos dada por (3.1), donde cada segmento permite combinar información de un par de imágenes y el valor de x es un escalar tal que $x \in \mathbb{R}$.

$$J_{r,c}(x) = \begin{cases} (x-1)I_{r,c}(2) + (2-x)I_{r,c}(1) & \text{si } 1 \leq x < 2 \\ (x-2)I_{r,c}(3) + (3-x)I_{r,c}(2) & \text{si } 2 \leq x < 3 \\ \vdots & \vdots \\ (x-N+1)I_{r,c}(N) + (N-x)I_{r,c}(N-1) & \text{si } N-1 \leq x < N \\ I_{r,c}(N) & \text{si } N \leq x \end{cases} \tag{3.1}$$

$$1 \leq r \leq n_r, 1 \leq c \leq n_c$$

De acuerdo con (3.1) si el valor de x es un valor real entre 0 y N entonces existen posibilidades infinitas para calcular $J_{r,c}(x)$. Una forma de simplificar el problema es hacer que x tome sólo valores enteros tales que $1 \leq x \leq N$. De este modo, cuando x toma un valor k significa que el pixel de la imagen fusionada será extraído de la k-ésima imagen, entonces $J_{r,c}(k) = I_{r,c}(k)$. Por lo tanto habrá un vector de N posibles valores para cada pixel de la imagen fusionada, los cuales corresponden al pixel de

cada una de las imágenes del conjunto en la coordenada r, c, y dicho vector está dado por (3.2).

$$\mathbf{J}_{r,c} = [I_{r,c}(1), I_{r,c}(2), \cdots, I_{r,c}(N)] \tag{3.2}$$

Uno de nuestros objetivos es realizar la fusión de imágenes en el menor tiempo posible por lo que la función de nitidez $F(k)$ para la k-ésima imagen es el valor absoluto de la convolución de la imagen con el kernel del Laplaciano discretizaco en (2.12) y está dada por (3.3).

$$F_{r,c}(k) = |h_{r,c} * I_{r,c}(k)| \tag{3.3}$$

Sabiendo que cada pixel de la imagen $I_{r,c}(k)$ tiene asociada una medida de nitidez $F_{r,c}(k)$, entonces existe un vector de posibles valores de nitidez para cada pixel r, c de la imagen fusionada, el cual está dado por (3.4).

$$\mathbf{F}_{r,c} = [F_{r,c}(1), F_{r,c}(2), \cdots, F_{r,c}(N)] \tag{3.4}$$

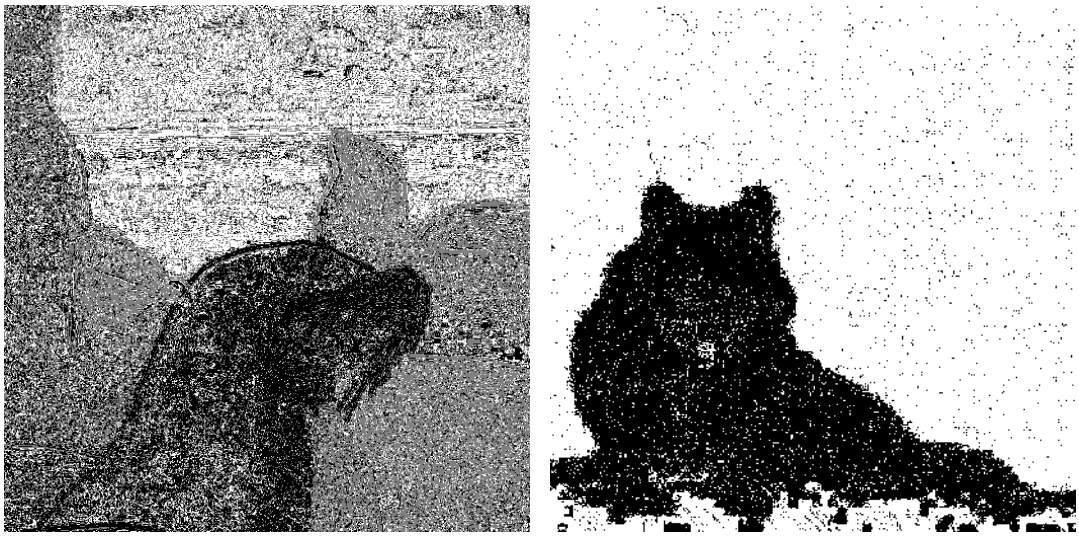
Dadas las consideraciones anteriores, existen $N \times n_r \times n_c$ posibles combinaciones para crear la imagen fusionada y nos interesa aquella combinación que tiene la mayor respuesta a la función de nitidez. Por lo tanto el valor de $P_{r,c}$ debe ser el índice del mayor valor de aptitud del conjunto dado por (3.4) y dicho valor puede ser encontrado de forma simple utilizando (3.5).

$$P^{\star}_{r,c} = \underset{k}{\operatorname{argmax}}\ F_{r,c}(k) \tag{3.5}$$
$$\forall (r,c) \in \{[1,\ldots,n_r] \times [1,\ldots,n_c]\}$$
$$k \in [1,2,\ldots,N]$$

Por lo tanto, la imagen fusionada es obtenida mediante (3.6).

$$J_{r,c} = I_{r,c}(P_{r,c}) \tag{3.6}$$
$$\forall (r,c) \in \{[1,\ldots,n_r] \times [1,\ldots,n_c]\}$$

Para el ejemplo de las imágenes mostradas en la Figuras 2.4(a)-2.4(c) aplicando (3.5), se obtiene el mapa de decisión $P_{r,c}$ mostrado en la Figura 3.1(a), donde podemos distinguir en negro ($P_{r,c} = 1$) aquellas regiones que son más nítidas en $I(1)$ que en $I(2)$ e $I(3)$, en gris ($P_{r,c} = 2$) las regiones de la escena que son más nítidas en $I(2)$ que en $I(1)$ e $I(3)$ y, finalmente, en blanco ($P_{r,c} = 3$) las regiones de la escena que son más nítidas en $I(3)$ que en $I(1)$ e $I(2)$. En la Figura 3.1(a) podemos observar que los valores del mapa de decisión no siempre son iguales en regiones correspondientes a un mismo objeto. Así, por ejemplo, sería deseable que todos los pixeles que pertenecen al lobo marino que se encuentra al frente aparecieran en negro en el mapa de decisión, sin embargo, algunos de los puntos son grises o blancos. Esto significa que los pixeles que forman al lobo marino serán tomados de diferentes imágenes cuando lo esperado es que sean tomados todos de la imagen $I(1)$. Con el resto de los objetos que aparecen en la escena suceden situaciones similares por lo que es necesario garantizar la coherencia espacial en el mapa de decisión para mejorar estas condiciones. Una situación similar sucede con el ejemplo sintético mostrado en la Figura 2.7 para el cual se obtiene el mapa de decisión mostrado en la Figura 3.1(b).



(a) P ***para el ejemplo de la Fig. 2.4*** **(b)** P ***para el ejemplo de la Fig. 2.7***

Figura 3.1: Mapa de decisión para las imágenes multi-foco de las Figuras 2.4 y 2.7

3.1. Coherencia espacial

La coherencia espacial es una propiedad que permite que los pixeles de una imagen que corresponden a una misma región u objeto tengan valores similares de enfoque, color, iluminación, etc. En caso de tener imágenes con ruido o regiones en las que la función de nitidez arroja valores similares o iguales para todas las imágenes del conjunto, esto provocará que pixeles vecinos que pertenecen a la misma región sean tomados de forma indistinta de cualquiera de las imágenes del conjunto, cuando lo deseable es que sean obtenidos de la misma imagen. Lo anterior genera una imagen fusionada con inconsistencias. La coherencia espacial permite garantizar que regiones homogéneas u objetos completos sean tomados de una única imagen y puede ser lograda utilizando restricciones de similitud entre un pixel y sus vecinos, utilizando umbrales de cambio o aplicando medidas estadísticas como la media, la mediana o la moda. En las siguientes secciones presentamos algunas formas de lograr la coherencia espacial en el mapa de decisión y, en consecuencia, en la imagen fusionada.

Calcularemos el promedio de la función de nitidez de todos los pixeles de la imagen fusionada con (3.7).

$$\bar{F}(P) = \frac{\sum_{r=1}^{n_r} \sum_{c=1}^{n_c} F_{r,c}(P_{r,c})}{n_r n_c} \tag{3.7}$$

3.1.1. Métricas de calidad del mapa de decisión

Los mapas de decisión contienen etiquetas que indican qué imagen es más nítida en cada coordenada (r, c). Dichas etiquetas nos permiten fusionar, segmentar y/o clasificar las regiones de las imágenes. Dado lo anterior, el mapa de decisión también puede ser utilizado para segmentar imágenes y, para valorar el desempeño del mapa de decisión obtenido, utilizaremos métricas de segmentación de imágenes.

Long *et al.* presentan 4 métricas que permiten obtener una medida cuantitativa de la calidad de un mapa de segmentación, cuando se conoce un mapa de referencia, [Long et˜al., 2014]. Dichas métricas son: Precisión a Nivel Pixel (Pixel Accuracy,

PA), Media de la Precisión a Nivel Pixel (Mean Pixel Accuracy, MPA), Media de la Intersección sobre la Unión (Mean Intersection over Union, MIU) y Frecuencia Ponderada de la Intersección sobre la Union (Frequency Weighted Intersection over Union, FWIU) dadas por (3.8), (3.9), (3.10) y (3.11), respectivamente.

$$PA = \frac{\sum_{i=1}^{N} n_{i,i}}{\sum_{i=1}^{N} t_i} \tag{3.8}$$

$$MPA = \frac{1}{N} \sum_{i=1}^{N} \frac{n_{i,i}}{t_i} \tag{3.9}$$

$$MIU = \frac{1}{N} \sum_{i=1}^{N} \frac{n_{i,i}}{t_i + \sum_{j=1}^{N} n_{j,i} - n_{i,i}} \tag{3.10}$$

$$FWIU = \left(\sum_{k=1}^{N} t_k \right)^{-1} \sum_{i=1}^{N} \frac{t_i \ n_{i,i}}{t_i + \sum_{j=1}^{N} n_{j,i} - n_{i,i}} \tag{3.11}$$

donde $n_{i,j}$ es el número de pixeles de la clase i que fueron clasificados en la clase j y $t_i = \sum_{j=1}^{N_j} n_{i,j}$ es el total de pixeles de la clase i.

3.2. Fusión utilizando optimización lineal con restricciones

La coherencia espacial en el mapa de decisión P se puede lograr restringiendo a que los valores de pixeles vecinos sean similares, mediante $|P_{r,c} - P_{r-1,c}| < \epsilon$, $|P_{r,c} - P_{r,c-1}| < \epsilon$ y $|P_{r,c} - P_{r-1,c-1}| < \epsilon$ en la función objetivo (3.7). Estas condiciones las podemos plantear en el proceso de maximización de la función de objetivo como (3.12).

$$\begin{gathered} P^{\star} = \underset{P}{\text{argmáx}} \ \bar{F}(P) \\ \text{s.a} \\ P_{r,c} - P_{r-1,c} > \epsilon, \ P_{r-1,c} - P_{r,c} < \epsilon \\ P_{r,c} - P_{r,c-1} > \epsilon, \ P_{r,c-1} - P_{r,c} < \epsilon \\ P_{r,c} - P_{r-1,c-1} > \epsilon, \ P_{r-1,c-1} - P_{r,c} < \epsilon \\ P_{r,c} \geq 1, \ P_{r,c} \leq N \end{gathered} \tag{3.12}$$

El proceso de optimización se puede resolver mediante el método Símplex. Dado

que los valores $P_{r,c}$ deben ser enteros para realizar la fusión, el resultado obtenido en el proceso de optimización debe ser redondeado al entero más próximo.

El proceso dado por (3.12), para un conjunto con dos imágenes, se resume en el Algoritmo 1, llamado Combinación Lineal de Imágenes (CLI), donde se calcula la función de nitidez para cada imagen y, en base a los resultados, se optimiza el sistema lineal con restricciones dado por (3.12) y P es el argumento que maximiza dicho sistema. Con los valores obtenidos para P se calcula la imagen fusionada J con (3.6), la cual es el resultado del proceso.

Algoritmo 1: Combinación Lineal de Imágenes CLI(I)

1 Calcular $F(1)$ con (3.3);
2 Calcular $F(2)$ con (3.3);
3 Optimizar (3.12) dados $F(1)$ y $F(2)$;
4 Obtener P como el argumento que maximiza (3.12);

Resultado: J calculada usando (3.6) con P

Se implementó el proceso de optimización planteado en el Algoritmo 1 utilizando Wolfram Mathematica con dos imágenes multi-foco, logrando resolver el problema para imágenes de hasta 512×512 pixeles, de manera muy eficiente. El código correspondiente en Mathematica se muestra en la Figura 3.2. En dicha figura se muestran tres secciones: en la primera, se calcula la función de nitidez para cada imagen; en la segunda sección, se forma el sistema lineal a maximizar con las restricciones correspondientes; finalmente, en la tercera sección se optimiza el sistema y se obtiene el valor de P que lo maximiza, el cual es redondeado al entero más próximo. Cabe señalar que la selección del máximo valor de $F_{r,c}(k)$ se ha implementado como una diferencia donde: si $F_{r,c}(2) > F_{r,c}(1)$, el valor que maximiza la función es $P_{r,c} = 2$; y en caso contrario el valor óptimo es $P_{r,c} = 1$; que es equivalente a realizar una búsqueda para encontrar el índice de la imagen con mayor valor de la función de nitidez.

Adicionalmente, se ha omitido la división por la cantidad de pixeles, puesto que no afecta ni aporta nada al proceso de optimización.

Cálculo de la función de nitidez

```
LaplacianoDisc[Image_]:=Block[{H},
    H = {{1,1,1},{1,-8,1},{1,1,1}};
    Abs[ImageData[ImageConvolve[Image,H]]]
]
F1=LaplacianoDisc[I1]; (*Calculamos la nitidez de I1*)
F2=LaplacianoDisc[I2]; (*Calculamos la nitidez de I2*)
```

Construcción del sistema

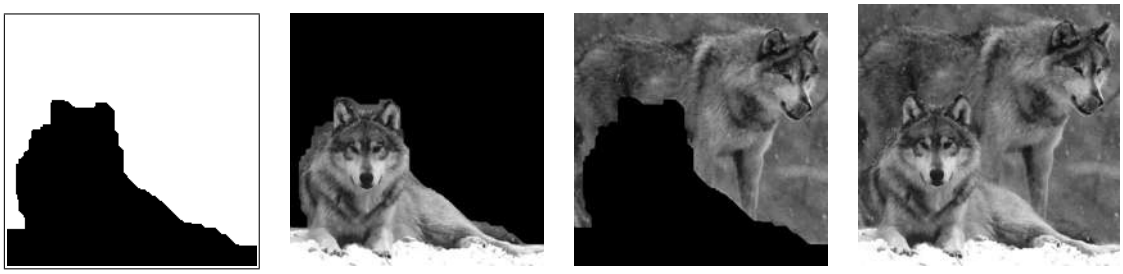
```
RF=∑_{r=1}^{nr} ∑_{c=1}^{nc} P_{r,c}(F2[[r,c]]-F1[[r,c]]); (*Función objetivo*)
(*Variables a optimizar*)
var = Flatten[Table[P_{r,c},{r,1,nr},{c,1,nc}]];
Restricciones = Flatten[{ (*Restricciones para cada valor de P*)
    Table[P_{r,c}≥ 1, {r,1,nr},{c,1,nc}],
    Table[P_{r,c}≤2, {r,1,nr},{c,1,nc}],
    Table[P_{r,c}-P_{r,c-1}≤ ϵ, {r,1,nr},{c,2,nc}],
    Table[P_{r,c-1}-P_{r,c}≤ ϵ, {r,1,nr},{c,2,nc}],
    Table[P_{r,c}-P_{r-1,c-1}≤ ϵ, {r,2,nr},{c,2,nc}],
    Table[P_{r-1,c-1}-P_{r,c}≤ ϵ, {r,2,nr},{c,2,nc}],
    Table[P_{r,c}-P_{r-1,c}≤ϵ, {r,2,nr},{c,1,nc}],
    Table[P_{r-1,c}-P_{r,c}≤ ϵ, {r,2,nr},{c,1,nc}]
}];
```

Solución usando NMaximize

```
(*Solución del sistema usando NMaximize*)
{t, sol} = NMaximize[{RF,Restricciones},var]//Timing;
(* Conversión a enteros de la solución*)
P=Table[Round[P_{r,c}/.Last[sol]],{r,1,nr},{c,1,nc}];
```

Figura 3.2: Implementación del Algoritmo 1 utilizando Wolfram Mathematica

Se aplicó la implementación mostrada en la Figura 3.2 para el ejemplo de las imágenes mostradas en las Figuras 2.7(a) y 2.7(b), con $\epsilon = 0.03$. El mapa de decisión obtenido se muestra en la Figura 3.3(a), la segmentación para cada uno de los lobos aparece en las Figuras 3.3(b) y 3.3(c), mientras que, en la Figura 3.3(d), se muestra la imagen fusionada dada por (3.6) con el mapa de decisión calculado.



(a) P ***con*** $\epsilon = 0.03$ **(b)** ***Lobo 1*** **(c)** ***Lobo 2*** **(d)** ***Imagen fusionada***

Figura 3.3: Fusión de las imágenes sintéticas de los lobos

Dado que conocemos el mapa de decisión original $\hat{P}$ mostrada en la figura 2.6(c) con el que fueron creadas las imágenes multi-foco, podemos calcular el porcentaje de valores que son iguales en $\hat{P}$ y P (Pixel Acuracy, PA) obteniendo como resultado un $PA = .9839$, lo que significa que se acertó en el 98.39 % de los pixeles. Adicionalmente, se puede observar que se ha logrado la coherencia espacial en el mapa de decisión mostrado en la Figura 3.1(b). Sin embargo, se ha perdido definición en los bordes del mapa de decisión, además de que el tiempo para hacer el cálculo de la fusión fue de 94 minutos para la imagen de 512×512 pixeles.

Se puede observar en el ejemplo anterior, que con la implementación propuesta utilizando Wolfram Mathematica, se logra obtener un mapa de decisión con muy buena coherencia espacial. Sin embargo, es importante hacer notar que para esta implementación es necesario determinar el valor de ϵ adecuado, puesto que un valor pequeño dará muy buena coherencia espacial, pero poca definición en los bordes; mientras que, con un valor de ϵ mayor, se perderá la coherencia espacial. Adicionalmente, a pesar de la eficacia de la función NMaximize de Wolfram Mathematica, encontrar la solución

al problema tiene una complejidad alta en tiempo, tal como se muestra en la gráfica de la Figura 3.4, donde se puede notar que el tiempo consumido para encontrar la solución se incrementa rápidamente a medida que crece el número de pixeles de la imagen. En [Calderon y Garnica, 2014], se pueden encontrar más detalles sobre esta implementación y las implicaciones que conlleva el cambio del valor de ϵ.

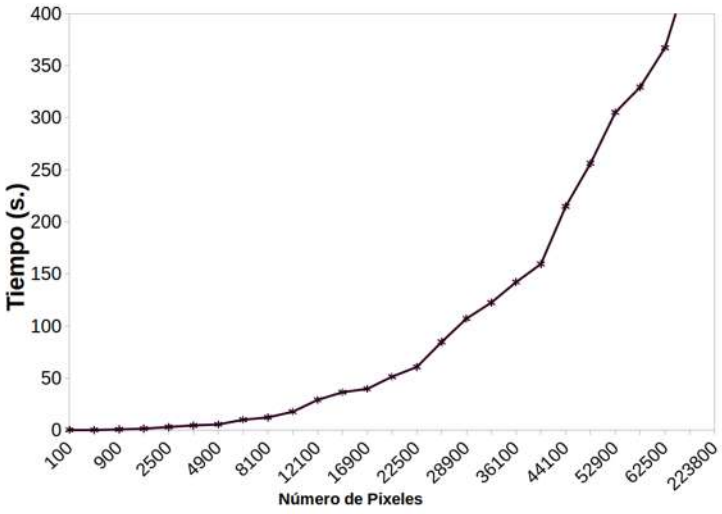


Figura 3.4: Número de pixeles de la imagen vs. tiempo de ejecución con Wolfram Mathematica.

Para solucionar el problema usando el método Símplex, tenemos que su forma canónica está dada por (3.13).

$$\begin{gathered} \text{máx } z = c_1x_1 + c_2x_2 + \cdots + c_{\mathcal{N}}x_{\mathcal{N}} \\ \text{Sujeto a} \\ Ax + \mathcal{I}x_h = b \end{gathered} \tag{3.13}$$

donde $\mathcal{I}$ es una matriz que inicialmente es la identidad, pero que durante las iteraciones del método pierde ésta característica y x_h es el vector de variables de holgura para trasformar el problema de desigualdades en igualdades. Los recursos de memoria necesarios para resolver el problema usando el método Símplex hacen imposible su aplicación en términos prácticos. El total de variables $\mathcal{N}$ será igual al número de renglones (n_r) por el número de columnas (n_c) de las imágenes, entonces $\mathcal{N} = n_r \times n_c$.

La matriz de restricciones o matriz aumentada A tendrá M renglones y $\mathcal{N}$ columnas, donde M es el número de restricciones. Dado que para cada pixel tenemos las 8 restricciones mostradas en (3.12) tendremos $M = 8 \times n_r \times n_c$, así que la matriz de restricciones A será de tamaño $\mathcal{N} \times M = 8n_r^2 n_c^2$. Entonces, la matriz identidad será de tamaño $M \times M = 64n_r^2 n_c^2$. Adicionalmente el método Símplex tiene una complejidad computacional $O(2^M)$, donde M es el número de vértices de la región de factibilidad. Dadas las consideraciones anteriores, si por ejemplo, tenemos una imagen de tamaño 256×256 y suponiendo que los valores sean almacenados en flotantes de 4 bytes, los recursos necesarios para resolver el problema serán de 137.4 GB para la matriz de restricciones y más de un TB para la matriz $\mathcal{I}$. Más aún, las matrices son altamente dispersas, pues inicialmente menos del 0.01 % de los valores en ambas matrices son diferentes de cero. Se podría plantear la propuesta de implementar el problema con ayuda de matrices ralas, sin embargo, el método Símplex tiene la desventaja de que, durante el proceso iterativo, la matriz deja de ser dispersa.

Otras alternativas de solución pueden ser los métodos de punto interior con el manejo de matrices ralas o dispersas, los cuales tienen un mejor desempeño que el método Símplex cuando se trata de este tipo de matrices.

3.3. Sistema de ventanas deslizantes

Otra posibilidad para reducir el costo computacional del proceso de optimización es resolver el problema de forma local, en lugar de resolverlo para toda la imagen. En otras palabras, aplicar el proceso de optimización sobre una región o ventana de la imagen, después mover la ventana y repetir el proceso hasta cubrir la totalidad de los pixeles. Este proceso se logra aplicando un sistema de ventanas deslizantes.

Calcularemos el promedio de la nitidez de la imagen fusionada para una ventana de tamaño $(2w+1) \times (2w+1)$ pixeles, centrada en la posición (r, c) con (3.14). Entonces, el proceso de optimización dado por (3.12) puede ser planteado como en (3.15), con

las restricciones (3.16) y (3.17), para un valor de w dado. Con ello obtenemos un sub-mapa de decisión por cada ventana al que llamamos $Q^{(r,c)}$ del mismo tamaño de la ventana donde l y m son coordenadas dentro del sub-mapa y toman valores desde 1 hasta $2w+1$.

$$\bar{\mathcal{F}}_{r,c}(Q^{(r,c)};w) = \frac{1}{(2w+1)^2}\sum_{l=r-w}^{r+w}\sum_{m=c-w}^{c+w} F_{l,m}\left(Q^{(r,c)}_{l-r+w+1,m-c+w+1}\right) \tag{3.14}$$

$$Q^{(r,c)\star} = \underset{Q^{(r,c)}}{\text{argmáx}}\ \bar{\mathcal{F}}_{r,c}(Q^{(r,c)};w) \tag{3.15}$$

Sujeto a.

$$\begin{aligned}\left|Q^{(r,c)}_{l,m} - Q^{(r,c)}_{l-1,m}\right| &\leq \epsilon \\ \left|Q^{(r,c)}_{l,m} - Q^{(r,c)}_{l,m-1}\right| &\leq \epsilon\end{aligned} \tag{3.16}$$

$$1 \leq Q^{(r,c)}_{l,m} \leq N \tag{3.17}$$

donde (3.16) son las restricciones de coherencia espacial y (3.17) limita los valores de $Q^{(r,c)}$. Se ha eliminado la restricción $|P_{r,c} - P_{r-1,c-1}| < \epsilon$ por considerala redundante y que ofrece poca mejora en la exactitud, a costa un costo considerable en tiempo de ejecución.

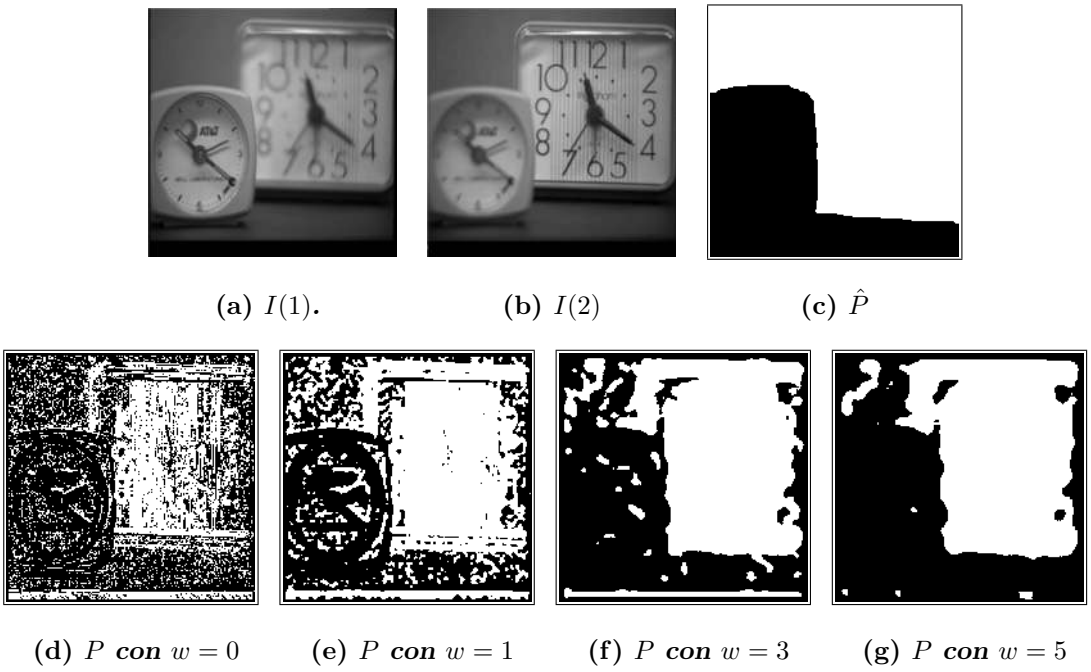
Al resolver el proceso de optimización planteado por (3.15), podemos obtener un sub-mapa de decisión por cada coordenada (r,c) donde la relación que existe entre el mapa de decisión final y cada sub-mapa está dada por (3.18).

$$\begin{gathered}P_{r,c} \leftrightarrow round(Q^{(r+l,c+m)\star}_{w-l+1,w-m+1}) \\ \forall 1 \leq r \leq n_r,\ 1 \leq c \leq n_c \\ -w \leq l \leq w,\ -w \leq m \leq w\end{gathered} \tag{3.18}$$

Dado que existen varias ventanas del sistema de ventanas deslizantes que abarcan la coordenada (r,c), tendremos un posible valor para $P_{r,c}$ por cada una de estas ventanas, por lo que el valor de $P_{r,c}$ será calculado como el promedio de todos ellos dado por (3.19), donde $V_{r,c}$ es el número de ventanas que abarcan la coordenada (r,c).

$$P_{r,c} = round\left(\frac{\sum_{l=-w}^{l=w}\sum_{m=-w}^{m=w} Q^{(r+l,c+m)\star}_{w-l+1,w-m+1}}{V_{r,c}}\right) \tag{3.19}$$

El mapa de decisión P se calcula con (3.19) para cada coordenada r, c, donde $Q^{(r,c)}$ se calcula mediante (3.15). En las Figuras 3.5(a) y 3.5(b) se muestra un par de imágenes multi-foco correspondientes a dos relojes. En la Figura 3.5(c) se muestra el mapa de decisión de referencia para este ejemplo, el cual fue estimado manualmente. En las Figuras 3.5(d-g) se muestran mapas de decisión obtenidos con $w = 0$, $w = 1$, $w = 3$ y $w = 5$, respectivamente.



(a) $I(1)$**.** **(b)** $I(2)$ **(c)** $\hat{P}$

(d) P ***con*** $w = 0$ **(e)** P ***con*** $w = 1$ **(f)** P ***con*** $w = 3$ **(g)** P ***con*** $w = 5$

Figura 3.5: Mapas de decisión para el ejemplo de los relojes variando el valor de w.

En la Figura 3.5 se puede observar que a medida que se incrementa el tamaño de la ventana, se logra una mayor coherencia espacial. Sin embargo, dado que la solución se calcula utilizando el método Simplex, que tiene una complejidad exponencial con respecto al número de variables (cantidad de pixeles en la ventana), a medida que el tamaño de la ventana crece, la cantidad de recursos demandados también crece; tal y como se muestra en la gráfica de la Figura 3.6, donde se muestra el tiempo de ejecución consumido por el algoritmo para fusionar dos imágenes de tamaño 256×256 pixeles cuando se incrementa el valor de w y, en consecuencia, el tamaño de la ventana.

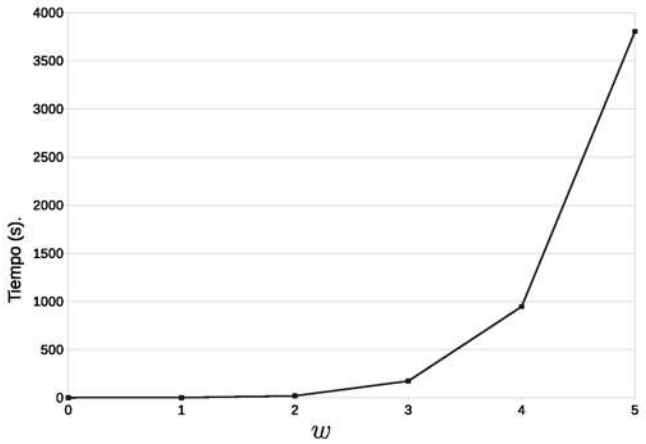


Figura 3.6: Tiempo de ejecución vs. tamaño de ventana.

Aplicando el procedimiento explicado previamente se tiene que las ventanas del sistema tienen un traslape entre sí, de tal suerte que, por ejemplo, para ventanas de tamaño 9×9 existe un traslape del 78 % entre $Q^{(r,c)}$ y $Q^{(r+1,c)}$. Lo mismo sucede con $Q^{(r-1,c)}$, $Q^{(r,c+1)}$ y $Q^{(r,c-1)}$. Más aún, la diferencia entre los valores dentro de un submapa y sus vecinos es mínima o inexistente, por lo que aporta muy poco al cálculo del mapa de decisión. Para reducir el traslape entre ventanas y, en consecuencia, el esfuerzo computacional, se ha introducido una variable Δ que permite controlar el traslape entre las ventanas. Así, para un valor $\Delta = w$, no existirá traslape, cada coordenada (r, c) estará abarcada sólo por una ventana, calculando cada $P_{r,c}$ una sola vez mientras que, con un valor $\Delta = 1$, se tiene el máximo traslape posible, calculando varios posibles valores para cada $P_{r,c}$, por lo que se debe aplicar (3.19).

3.3.1. Escalamiento de imágenes

De los experimentos realizados se observo que un tamaño de ventana que provee un balance entre el tiempo de ejecución y la calidad del resultado es de 9×9 ($w = 4$). Cabe señalar que, es deseable aplicar ventanas de mayor tamaño, sin embargo, al hacerlo se incremente el esfuerzo computacional. Otra manera de simular el incremento en el tamaño de ventana, es reducir el tamaño de la imagen, manteniendo el tamaño de

la ventana se logrará cubrir una mayor parte de la imagen que si se aplicara en el tamaño original, lo cual es similar a aplicar una ventana de mayor tamaño sobre la imagen original. Si se fija el tamaño de la ventana en 9×9, se escalan las imágenes del conjunto multi-foco a un factor de escala s, de acuerdo con (3.20), para cualquier imagen L, se logra un . Lo anterior permite obtener un resultado similar al obtenido aplicando ventanas de mayor tamaño, pero con un consumo mucho menor de recursos.

$$\begin{gathered} L_{r,c}^{(s)} = L_{s\times r, s\times c} \\ 1 \leq r \leq \frac{n_r}{s}, 1 \leq c \leq \frac{n_c}{s} \end{gathered} \tag{3.20}$$

En (3.20), si $s > 1$, la imagen será reducida, mientras que, si $s < 1$, la imagen será expandida y, si $s = 1$, la imagen no sufre cambios. El proceso de escalado lo podemos dividir en dos: la reducción dada por (3.21) y la expansión dada por (3.22). En la reducción se aplica un proceso de suavizado de la imagen y, en la expansión se aplica interpolación bilineal para reducir la pérdida de información.

$$Reduce(L, s) = L^{(s)} \tag{3.21}$$

$$Expande(L, s) = L^{(1/s)} \tag{3.22}$$

3.3.2. Combinación Lineal de Imágenes utilizando Ventanas deslizantes (CLI-V)

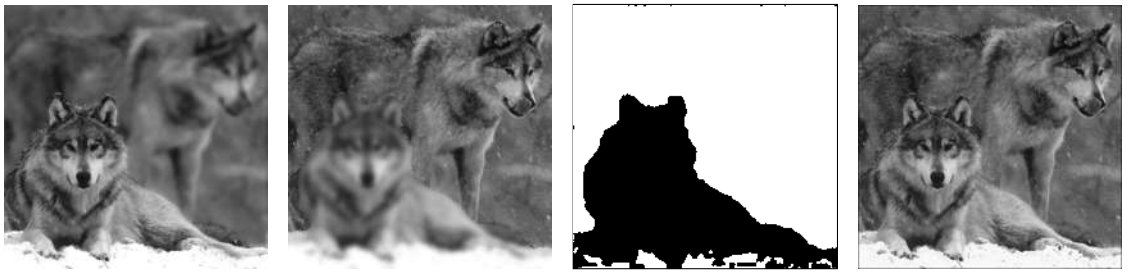
En el Algoritmo 2 (Combinación Lineal de Imágenes por Ventanas, CLI-V) se presenta el proceso para fusionar un conjunto de imágenes multi-foco. En el proceso las imágenes del conjunto se reducen con (3.21), para obtener $I^{(s)}(1)$ e $I^{(s)}(2)$ con un tamaño de $\frac{n_r}{s} \times \frac{n_c}{s}$ pixeles. Con las imágenes reducidas se calcula el mapa de decisión $P^{(s)}$, el cual también es de tamaño $\frac{n_r}{s} \times \frac{n_c}{s}$, por lo que se expande con (3.22) para llevarlo al tamaño de las imágenes originales $n_r \times n_c$ pixeles . Este proceso se repite desde $s = s_{max}$ hasta $s = 1$ y P se calcula como el promedio de los mapas obtenidos en cada iteración. Finalmente, la imagen fusionada es obtenida con (3.6).

Algoritmo 2: Combinación Lineal de Imágenes con Ventanas deslizantes CLI-V(I, ϵ, w, Δ, s_{max})

para $s \leftarrow s_{max}$ **a** 1 **hacer**
  Calcular $I^{(s)}(k) \leftarrow Reduce(I(k), s)\ \forall k \in [1, 2, \ldots, N]$ usando (3.21) ;
  Calcular $F^{(s)}(k)$, $\forall k \in [1, 2, \ldots, N]$ usando (3.3) con $I^{(s)}(k)$;
  Establecer $P_{r,c}^{(s)} \leftarrow 1$ y $N_w(r, c) \leftarrow 0\ \forall 1 \leq r \leq n_r,\ 1 \leq c \leq n_c$;
  para $r \leftarrow 1$ **a** n_r **hacer**
    para $c \leftarrow 1$ **a** n_c **hacer**
      $Q^{(r,c)\star} \leftarrow \underset{Q^{(r,c)}}{\text{argmáx}}\ \bar{\mathcal{F}}_{r,c}(Q^{(r,c)}; w)$ con (3.15);
      para $l \leftarrow -w$ **a** w **hacer**
        para $m \leftarrow -w$ **a** w **hacer**
          si $(1 \leq r + l \leq n_r\ y\ 1 \leq c + m \leq n_c)$ **entonces**
            $P^{(s)}(r + l, c + m) \leftarrow P^{(s)}(r + l, c + m) + Q^{(r,c)\star}_{l+w+1,m+w+1}$;
            $V(r + l, c + m) \leftarrow N_w(r + l, c + m) + 1$;
      $c \leftarrow c + \Delta$;
    $r \leftarrow r + \Delta$;
  $P^{(s)}(r, c) \leftarrow \frac{P^{(s)}(r,c)}{V(r,c)} \qquad \forall \quad 1 \leq r \leq n_r, 1 \leq c \leq n_c$;
  $P \leftarrow P + Expande(P^{(s)}, s)$;
$P_{r,c} \leftarrow round\left(\frac{P_{r,c}}{s_{max}}\right)$;
$J_{r,c} \leftarrow I_{r,c}(P_{r,c})$ usando (3.6);
Resultado: J

La Figura 3.7 muestra los resultados de aplicar el algoritmo CLI-V para las imágenes multi-foco mostradas en las Figuras 3.7(a) y 3.7(b). La Figura 3.7(c) muestra el mapa de decisión obtenido con el algoritmo CLI-V con $w = 4, \Delta = 3$ y $\epsilon = 0.01$. Finalmente, la Figura 3.7(d) muestra la imagen fusionada. Para este conjunto uti-

lizando Wolfram Mathematica, se obtuvo un porcentaje de acierto de 98 % en 94 minutos, mientras que con CLI-V se logra el mismo porcentaje en 2.75 minutos; lo cual representa una reducción de 35 veces el tiempo de ejecución.



(a) $I(1)$ **(b)** $I(2)$ **(c)** P **(d)** J

Figura 3.7: Resultados de aplicar CLI-V al conjunto de imágenes sintetico de los lobos.

En la gráfica mostrada en la Figura 3.8 se muestra que la curva de crecimiento del tiempo de ejecución consumido por el algoritmo CLI es mucho más pronunciada que la del algoritmo CLI-V, por lo que CLI-V es preferible a medida que el tamaño de las imágenes se incrementa. Adicionalmente, ambos algoritmos tienen un porcentaje de acierto similar.

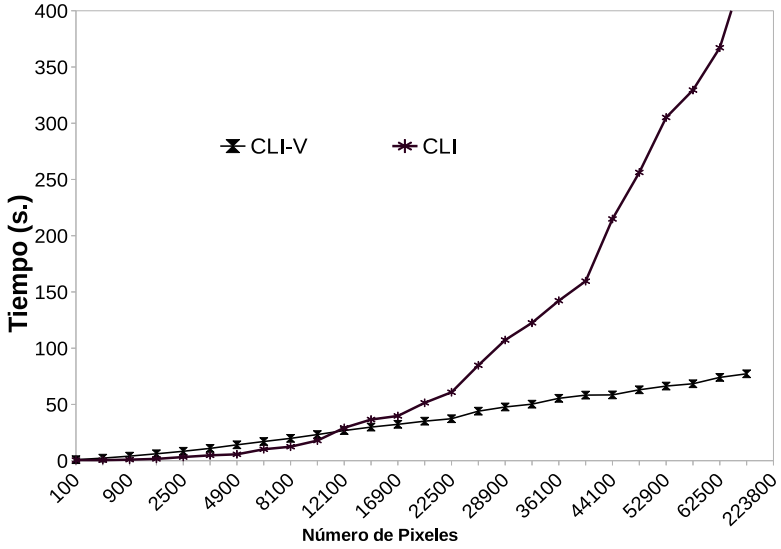


Figura 3.8: Cantidad de pixeles vs. tiempo de ejecución para CLI y CLI-V.

En [Garnica et˜al., 2018] se presentan más detalles sobre el algoritmo CLI-V, el cual permite la fusión de imágenes con un alto porcentaje de acierto, sin embargo, el tiempo de ejecución consumido no permite aún su aplicación en tiempo real, por lo que es necesario el uso de técnicas que nos permitan reducir el tiempo consumido en la optimización.

3.4. Combinación Lineal de Imágenes Simple (CLI-S) con imágenes integrales

Para las restricciones de coherencia espacial dadas por (3.15), si consideramos que el valor de ϵ tiende a cero los valores dentro de los sub-mapas de decisión $Q^{(r,c)}$ tenderán a un valor constante al que denominamos $\mathcal{Q}_{r,c}$. Bajo esta consideración podemos replantear las restricciones de coherencia espacial como (3.23).

$$\begin{aligned} \mathcal{Q}_{r,c} \equiv Q_{l,m}^{(r,c)} &\approx Q_{l-1,m}^{(r,c)} \\ \mathcal{Q}_{r,c} \equiv Q_{l,m}^{(r,c)} &\approx Q_{l,m-1}^{(r,c)} \\ \mathcal{Q}_{r,c} \equiv Q_{l,m}^{(r,c)} &\approx Q_{l-1,m-1}^{(r,c)} \end{aligned} \tag{3.23}$$

Asumiendo que todos los valores $Q_{l,m}^{(r,c)}$ son constantes dentro de una ventana, sólo es necesario buscar un valor $\mathcal{Q}_{r,c}$ por cada coordenada (r, c), en lugar de $(2w + 1)^2$ valores, para ello maximizamos (3.24),

$$\bar{\mathcal{F}}_{r,c}(P_{r,c}; w) = \tfrac{1}{(2w+1)^2} \sum_{l=r-w}^{r+w} \sum_{m=c-w}^{c+w} F_{l,m}(P_{r,c}) \tag{3.24}$$

Además, si $\mathcal{Q}_{r,c} \in \mathbb{N}$, entonces $P_{r,c} \equiv \mathcal{Q}_{r,c}$ y el mapa de decisión se calcula como (3.25).

$$\begin{gathered} P_{r,c}^{\star} = \underset{P_{r,c}}{\text{argmáx}}\ \bar{\mathcal{F}}_{r,c}(P_{r,c}; w) \\ \text{s.a} \\ P_{r,c} \in [1, 2, \ldots, N] \end{gathered} \tag{3.25}$$

Dado que $P_{r,c} \in [1, 2, \ldots, N]$ entonces sólo existen N posibles valores para cada $P_{r,c}$ por lo que se puede encontrar $P^{\star}_{r,c}$ de una forma simple mediante un proceso iterativo. En función de la simpleza para encontrar el $P^{\star}_{r,c}$, vemos que el cálculo más costoso es la sumatoria que aparece en (3.24).

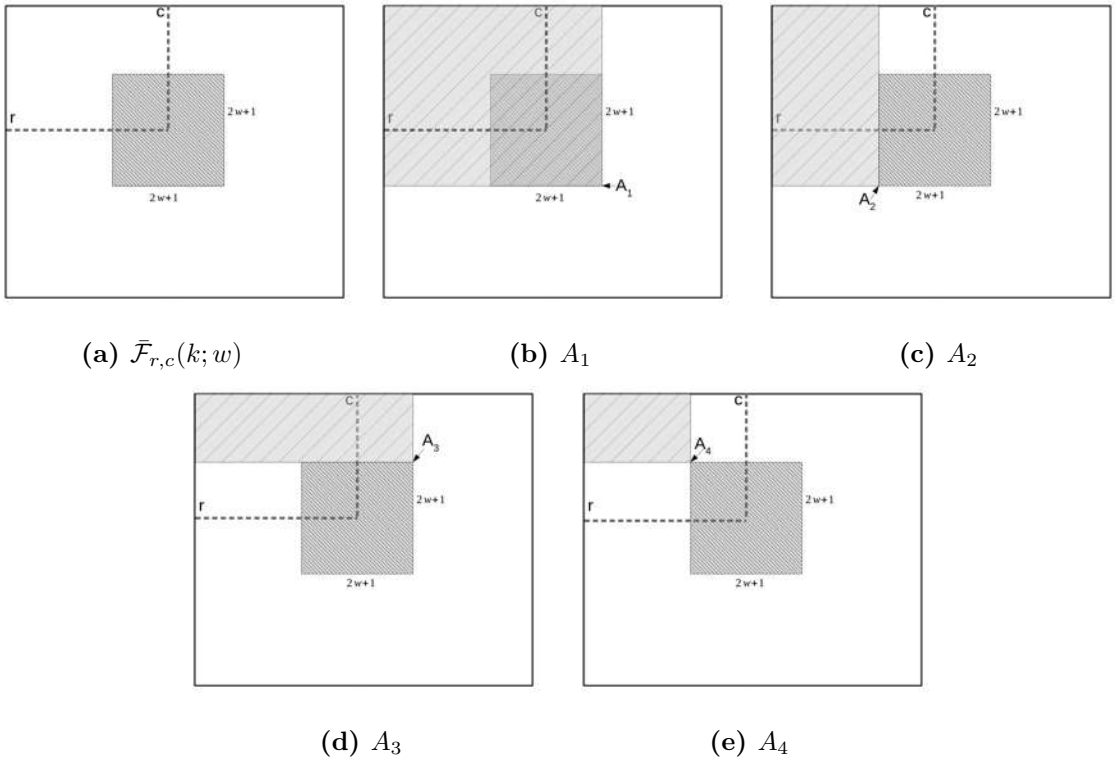
3.4.1. Imágenes integrales

La parte derecha de (3.24), es el promedio de los valores de la función de aptitud de la imagen dentro de una ventana de tamaño $(2w+1) \times (2w+1)$ centrada en las coordenadas (r, c), el cual se puede obtener de forma eficiente utilizando imágenes integrales, procedimiento propuesto en [Viola y Jones, 2001]. El procedimiento permite calcular la suma de los valores dentro de una ventana en un tiempo constante e independiente del tamaño de la ventana y consiste en calcular primero una imagen integral $H(k)$ utilizando (3.26).

$$H_{r,c}(k) = \sum_{l=1}^{r} \sum_{m=1}^{c} F_{l,m}(k) = F_{r,c}(k) + H_{r,c-1}(k) + H_{r-1,c}(k) - H_{r-1,c-1}(k) \qquad (3.26)$$

Para calcular la suma de los valores dentro de una ventana se deberán obtener cuatro valores de la imagen integral. Estos valores son las sumatorias de los valores de las áreas: $A_1 = H_{r+w,c+w}(k)$, $A_2 = H_{r+w,c-w-1}(k)$, $A_3 = H_{r-w-1,c+w}(k)$ y $A_4 = H_{r-w-1,c-w-1}(k)$, las cuales se muestran en las Figuras 3.9(b), 3.9(c), 3.9(d) y 3.9(e) respectivamente. Analizando estos valores y vistos como áreas sobre la imagen podemos decir que el valor de la suma dentro de una ventana se puede calcular como $A_1 - A_2 - A_3 + A_4$, lo que significa que la complejidad es independiente del tamaño de la ventana.

Dada la imagen integral $H(k)$ y un valor de w, podemos definir el promedio de los valores de la función de aptitud de una imagen k, dentro de una ventana que abarca desde $r-w, c-w$ hasta $r+w, c+w$ como (3.27).



(a) $\bar{\mathcal{F}}_{r,c}(k;w)$ **(b)** A_1 **(c)** A_2

(d) A_3 **(e)** A_4

Figura 3.9: Cálculo de la sumatoria dentro de una ventana usando imágenes integrales.

$$\bar{\mathcal{F}}_{r,c}(k;w) = \frac{H_{r+w,c+w}(k) - H_{r+w,c-w-1}(k) - H_{r-w-1,c+w}(k) + H_{r-w-1,c-w-1}(k)}{(2w+1)^2} \tag{3.27}$$

Con la aplicación de los conceptos anteriores podemos encontrar el mapa de decisión utilizando un sistema de ventanas de tamaño $(2w+1)\times(2w+1)$ y, si queremos que los mapas de decisión tengan mayor coherencia espacial debemos usar valores superiores a 0 sabiendo que, a medida que el valor de w crece, se obtiene una mayor coherencia espacial; pero con la ventaja de que el incrementar o disminuir el valor de w no incrementará el tiempo de ejecución requerido para encontrar la solución.

El Algoritmo 3, al que denominamos Combinación Lineal de Imágenes Simple (CLI-S) muestra el procedimiento para obtener el mapa de decisión dado por (3.25) y la imagen fusionada dada por (3.6). En este algoritmo la imagen integral se calcula

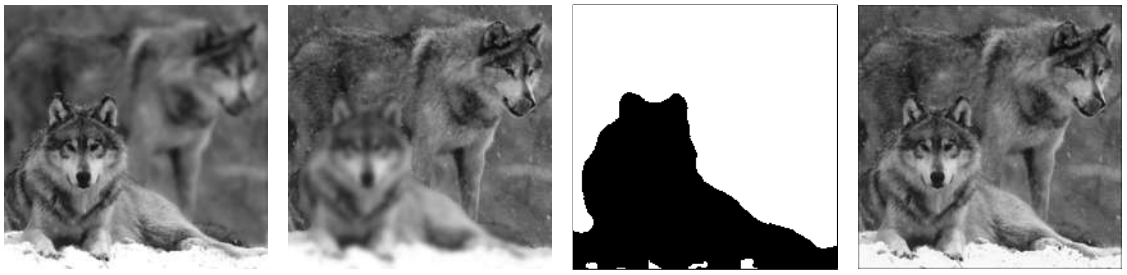
una sola vez y los valores por ventana corresponden a una simple suma y una división, en lugar de un proceso de optimización.

Algoritmo 3: Combinación Lineal de Imágenes Simple CLI-S(I, w)

```
1  Calcular F(k) ←Nitidez de I(k), ∀k ∈ [1, 2, ..., N] usando (3.3);
2  Calcular H(k) ∀k ∈ [1, 2, ..., N] usando (3.26);
3  para r ← 1 a n_r hacer
4      para c ← 1 a n_c hacer
5          Definir P_{r,c} ← 1 y f_max ← F̄_{r,c}(1; w);
6          para k ← 2 a N hacer
7              f_k ← F̄_{r,c}(k; w);
8              si f_k > f_max entonces
9                  f_max ← f_k;
10                 P_{r,c} ← k;
```

Resultado: J calculada usando (3.6) con P

Después de aplicar el algoritmo CLI-S con las imágenes sintéticas de los lobos presentadas en las Figuras 3.10(a) y 3.10(b), se obtuvo el mapa de decisión mostrado en la Figura 3.10(c) utilizando un valor $w = 31$.



(a) $I(1)$ **(b)** $I(2)$ **(c)** P **(d)** J

Figura 3.10: Resultados de aplicar CLI-S al conjunto de imágenes sintetico de los lobos.

En la Tabla 3.1 se muestran los resultados de aplicar los tres algoritmos a las

imágenes de los lobos. La primera columna corresponde al nombre del algoritmo, la segunda a los parámetros utilizados en la implementación, la tercera al tiempo en segundos y, la cuarta, al porcentaje de acierto entre el mapa de decisión utilizado para crear las imágenes y cada uno de los mapas calculados. Podemos notar que el porcentaje de aciertos es muy similar para los tres algoritmos, sin embargo, el Algoritmo CLI-S es miles de veces más rápido al realizar la fusión en décimas de segundo mientras que, CLI-V tarda minutos y, CLI tarda horas. En [Calderon et˜al., 2016] se presentan más detalles sobre el algoritmo CLI-S.

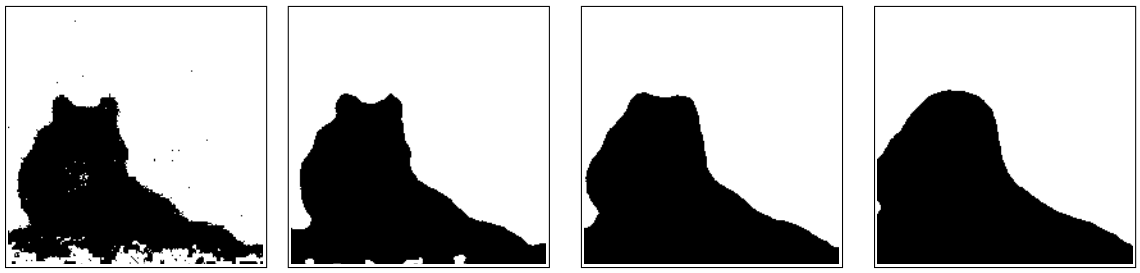
Tabla 3.1: Comparación de los algoritmos CLI, CLI-V y CLI-S con las imágenes multi-foco de los lobos.

Alg.	Parámetros	tiempo (seg).	% de aciertos
CLI	$\epsilon = 0.03$	5641.36	98.38
CLI-V	$\epsilon = 0.01$ $w = 4$ $\Delta = 3$	160.99	98.3
CLI- S	$w = 31$	0.16	98.39

3.5. Combinación Lineal de Imágenes con Ventanas Adaptables (CLI-VA)

En la Figura 3.11 se muestran los mapas de decisión para el ejemplo de las imágenes sintéticas mostradas en la Fig. 2.7 calculados al aplicar el algoritmo CLI-S con $w = 0$ (Fig. 3.11(a)), $w = 15$ (Fig. 3.11(b)), $w = 30$ (Fig. 3.11(c)) y $w = 45$ (Fig. 3.11(d)). En la imagen de la Figura 3.11(a) se observa que con un tamaño de ventana pequeño se logra una buena definición en los bordes de segmentación, pero se tiene poca coherencia espacial. En la Figura 3.11(b) se tiene suficiente coherencia espacial con bordes que se aproximan a los bordes reales de las imágenes multi-foco mientras que, con tamaño de ventana grande (Figs. 3.11(c) y 3.11(d)) se tiene cohe-

rencia espacial, pero se deslocalizan los bordes del mapa de decisión. Entonces debe existir un valor w tal que minimice las diferencias entre el mapa de decisión calculado P y el mapa de decisión original $\hat{P}$, con el que se generaron las imágenes sintéticas.



(a) $w = 0$ **(b)** $w = 15$ **(c)** $w = 30$ **(d)** $w = 45$

Figura 3.11: Mapas de decisión dados por el algoritmo CLI-S variando w.

En la Figura 3.12 se muestra una gráfica del comportamiento del porcentaje de aciertos obtenidos con el algoritmo CLI-S (Alg. 3) para el ejemplo de las imágenes de los lobos. Note que el valor que maximiza dicho porcentaje es $w = 15$. Sin embargo, este valor w óptimo sólo puede ser obtenido cuando se tiene un mapa de decisión de referencia $\hat{P}$, el cual en conjuntos de imágenes multi-foco reales obviamente no existe. Además de que el valor óptimo para w puede cambiar para cada par de imágenes multi-foco.

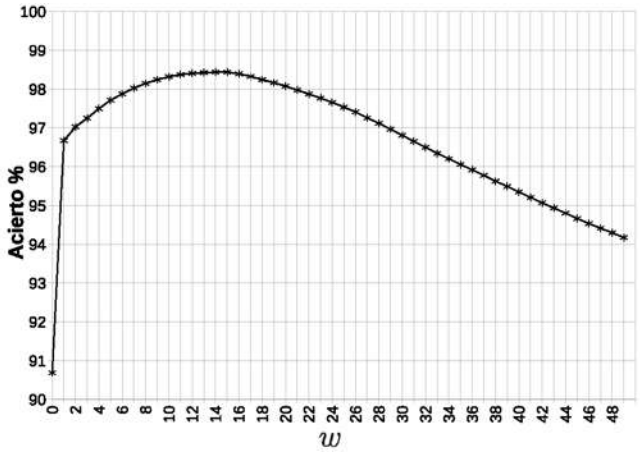


Figura 3.12: Gráfica del porcentaje de acierto del Algoritmo CLI-S al variar w.

Dado que con un valor grande de w se logra muy buena coherencia espacial pero se pierde definición en los bordes y que, con un valor de w cercano a cero se tiene buena definición en los bordes pero se pierde la coherencia espacial. Entonces, se debería aplicar un valor de w pequeño cerca de los bordes y un valor de w grande en las regiones donde no hay bordes. Más aún, lo ideal es calcular automáticamente el tamaño de las ventanas en cada uno de los pixeles. Este tamaño debe estar en función de la distancia del pixel a los bordes del mapa de decisión, los cuales deberían coincidir con las fronteras entre las regiones nítidas y borrosas de la imagen.

3.5.1. Cálculo eficiente del tamaño de la ventana

Definimos una ventana como el conjunto de las coordenadas (r, c) asociadas a cada pixel dentro de una región cuadrada. Esta ventana estará centrada en las coordenadas (r, c) con área $A(W_{r,c}) = (2W_{r,c} + 1)^2$ donde $W_{r,c} \in \mathbb{N}$.

Definimos la matriz de bordes B del mapa de decisión, donde el valor de cada $B_{r,c}$ será igual a 1 en caso de existir un borde y 0 si no existe; así, dado P podemos calcular B mediante (3.28). Inicialmente asumimos que la imagen 1 es la más nítida y el mapa de decisión es $P_{r,c} = 1$ en todas las coordenadas. Por lo tanto, la matriz de bordes tendrá ceros en cada una de las coordenadas de la imagen, $B_{r,c} = 0$.

$$B_{r,c} = \begin{cases} 1 & P_{r,c} \neq P_{r-1,c} \text{ ó } P_{r,c} \neq P_{r,c-1} \\ 0 & \text{Si no.} \end{cases} \tag{3.28}$$

El objetivo es crear la ventana más grande posible donde no haya bordes y tan pequeña como sea necesaria cerca de ellos. Dada una matriz de bordes B, maximizamos el área de la ventana A, asociada a cada coordenada (r, c), limitando a que la cantidad de puntos del borde dentro de la ventana sea menor a una tolerancia T y que el área de la ventana sea menor o igual a $(2w_{max} + 1)^2$. Esto lo podemos plantear como un proceso de optimización dado por (3.29). Permitir que el valor mínimo de $W_{r,c}$ sea 0 en (3.29) hace posible la creación de ventanas de 1×1 en las regiones

cercanas a los bordes, lo cual es equivalente a hacer un procesamiento a nivel pixel, y limitar $W_{r,c} \leq w_{max}$ evita que las ventanas crezcan indefinidamente debido a la falta de bordes inicialmente.

$$W^{\star}_{r,c} = \underset{W_{r,c}}{\text{argmáx}} \; A(W_{r,c}) \tag{3.29}$$

sujeto a.

$$0 \leq W_{r,c} \leq w_{max} \tag{3.30}$$

$$\sum_{l=r-w}^{r+w} \sum_{m=c-w}^{c+w} B_{l,m} < T \tag{3.31}$$

En la Figura 3.13 se muestran dos ventanas centradas en las coordenadas (r, c), con áreas $(2w_{max} + 1)^2$ y $(2W_{r,c} + 1)^2$. Se puede observar que la ventana mayor cumple con la restricción dada por (3.30), pero abarca gran cantidad de puntos del borde violando la restricción dada por (3.31) mientras que, la ventana interna, es la ventana más grande que se puede crear sin violar ninguna de las restricciones.

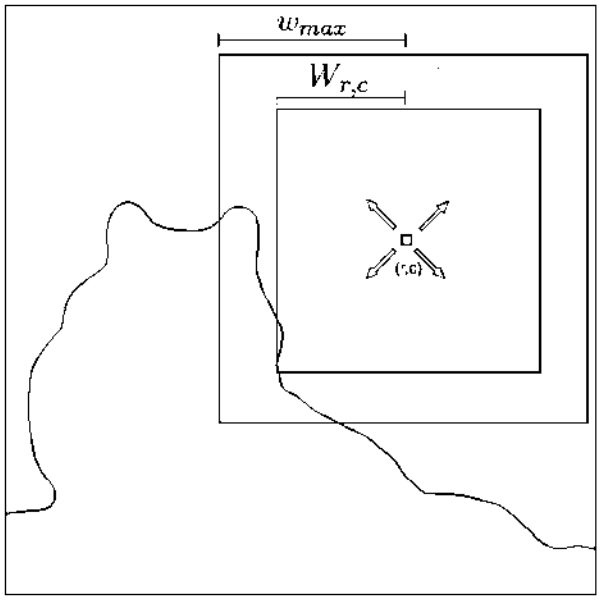


Figura 3.13: Ejemplo del proceso para obtener $W_{r,c}$ óptimo para una coordenada (r, c).

Para hacer eficiente la búsqueda del valor $W_{r,c}$ que optimiza (3.29) proponemos utilizar búsqueda binaria y calcular la sumatoria de la restricción dada por (3.31)

utilizando imágenes integrales [Viola y Jones, 2001] de acuerdo con (3.32) y (3.33).

$$\mathcal{B}_{r,c} = \sum_{l=1}^{r}\sum_{m=1}^{c} B_{l,m} = B_{r,c} + \mathcal{B}_{r,c-1} + \mathcal{B}_{r-1,c} - \mathcal{B}_{r-1,c-1} \tag{3.32}$$

$$SB(r,c,w) = \mathcal{B}_{r+w,c+w} - \mathcal{B}_{r+w,c-w-1} - \mathcal{B}_{r-w-1,c+w} + \mathcal{B}_{r-w-1,c-w-1} \tag{3.33}$$

donde $\mathcal{B}_{r,c}$ es la suma de todos los puntos de la matriz de bordes desde $(1,1)$ hasta (r,c) y $SB(r,c,w)$ es la suma de los bordes desde las coordenadas $(r-w, c-w)$ hasta $(r+w, c+w)$.

En el Algoritmo 4 llamado Ventana Óptima (VO) se resume lo planteado en los párrafos anteriores. El Algoritmo VO recibirá como parámetros: el valor máximo para w_{max}, la tolerancia T y un mapa de decisión P y regresará una matriz W con los valores que optimizan (3.29) para todas las coordenadas $(r,c) \in \{1,\ldots,n_r\} \times \{1,\ldots,n_c\}$.

Algoritmo 4: Algoritmo de Ventana Optima VO(w_{max}, T, P)

```
Calcular B_{r,c}, ∀(r,c) ∈ {1,...,n_r} × {1,...,n_c} con (3.28);
Calcular B con (3.32);
para r ← 1 a n_r hacer
    para c ← 1 a n_c hacer
        Hacer a ← 0 y b ← w_max;
        mientras b − a > 1 hacer
            w ← ⌊(a+b)/2⌋;
            si SB(r,c,w) ≤ T entonces
                a ← w;
            en otro caso
                b ← w;
        W_{r,c} ← w;
```

Resultado: W

En la Figura 3.14 se muestran ejemplos de ventanas calculadas con el Algoritmo VO centradas en sus pixeles correspondientes sobre un fondo negro, para el par multi-foco de la Figura 2.7. Note que se obtuvieron ventanas de tamaño adaptable que son grandes en regiones homogéneas y pequeñas en regiones cercanas a los bordes. Las ventanas se traslapan ya que dos coordenadas adyacentes pueden tener ventanas de tamaño diferente.

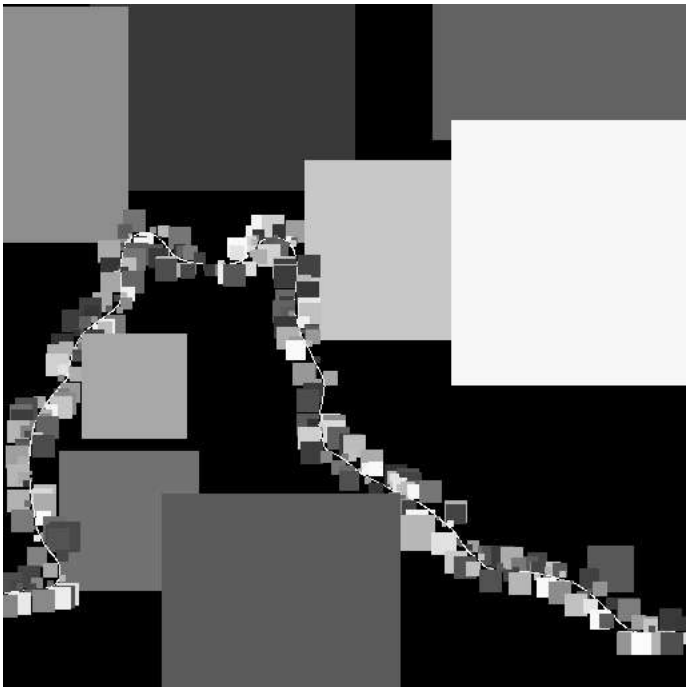


Figura 3.14: Ejemplos de tamaños de ventana obtenidos con el Algoritmo VO (Alg. 4).

Al inicio no se tiene información sobre el mapa de decisión P, por lo que definimos un mapa de decisión inicial $P^{(0)}$ con 1 en todas las coordenadas. Se aplica el algoritmo VO, usando $P^{(0)}$ para obtener $W^{(1)}$, el cual dará como resultado ventanas con área $(2w_{max}+1)^2$ (debido a que no existen bordes). Con esta información podemos calcular un nuevo mapa de decisión $P^{(1)}$, usando $W^{(1)}$. Sin embargo, dicho mapa tendrá poca exactitud en los bordes. Para afinar nuestro mapa proponemos aplicar nuevamente el algoritmo VO, usando $P^{(1)}$ para obtener $W^{(2)}$, que contiene los nuevos tamaños de ventana adecuados y que nos permite calcular un nuevo mapa de decisión $P^{(2)}$, el cual

deberá tener una mayor precisión en los bordes de segmentación. En general podemos repetir el proceso para calcular $W^{(iter)}$ en función de los bordes de $P^{(iter-1)}$ y $P^{(iter)}$ en función de $W^{(iter)}$, hasta que la diferencia entre $P^{(iter)}$ y $P^{(iter-1)}$ sea menor a un umbral o se supere un número máximo de iteraciones.

El Algoritmo 5 al que denominamos Combinación Lineal de Imágenes con Ventanas de tamaño Adaptable (CLI-VA) es una modificación del Algoritmo 3 (CLI-S). Los principales cambios que se han hecho son: dar un valor inicial a P (línea 2) e iterar el proceso para refinar el mapa de decisión.

Algoritmo 5: Combinación Lineal de Imágenes con Ventanas Adaptables CLI-VA (I, w_{max}, T, N_{iter})

```
1  Calcular F(k) ←Nitidez de I(k), ∀k ∈ [1, 2, ..., N] usando (3.3);
2  Calcular H(k) ∀k ∈ [1, 2, ..., N] usando (3.26);
3  P_{r,c}^{(0)} ← 1 ∀(r, c) ∈ {1, ..., n_r} × {1, ..., n_c};
4  para iter ← 1 a N_iter hacer
5      W^{(iter)} ← VO(w_max, T, P^{(iter-1)}) (Alg. 4);
6      para r ← 1 a n_r hacer
7          para c ← 1 a n_c hacer
8              f_max ← F̄_{r,c}(1; W_{r,c}^{(iter)});
9              para k ← 2 a N hacer
10                 f_k ← F̄_{r,c}(k; W_{r,c}^{iter});
11                 si f_k > f_max entonces
12                     f_max ← f_k;
13                     P_{r,c}^{(iter)} ← k;
```

Resultado: J calculada usando (3.6) con $P^{(iter)}$

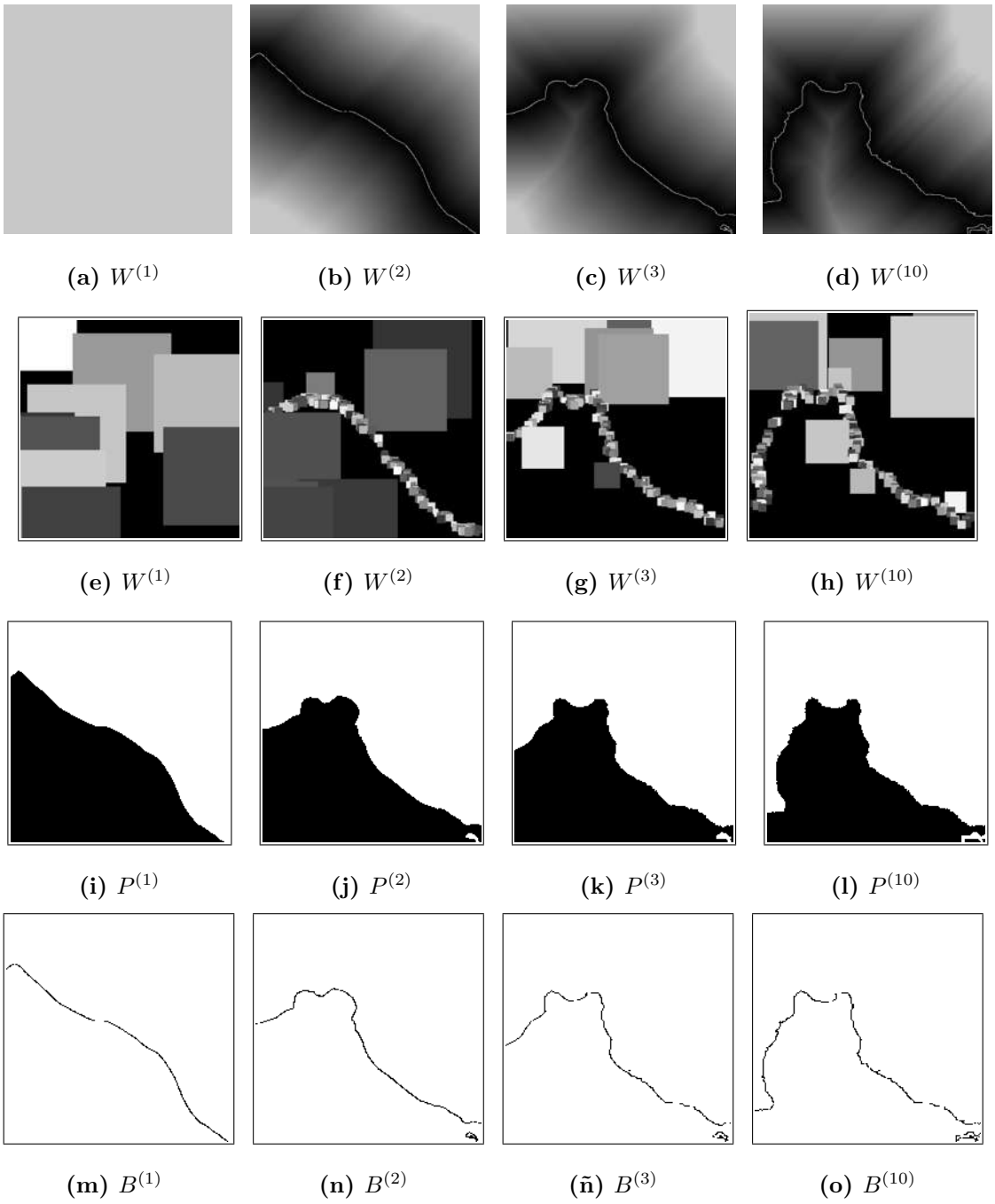
El algoritmo CLI-VA(I, w_{max}, T, N_{iter}) requiere de cuatro parámetros, donde w_{max} debe ser un valor suficientemente grande para generar ventanas que abarquen entre

un 25 y un 50 % del tamaño total de la imagen, el valor de T indica cuantos puntos del borde se toleran dentro de una ventana y N_{iter} es el número de iteraciones que se harán para mejorar P. Cabe mencionar que se utilizan imágenes integrales en el Algoritmo 5 (línea 10), lo cual permite que el tamaño de la ventana no influya en el tiempo de ejecución.

3.5.2. Resultados del algoritmo CLI-VA

En las imágenes de la Figura 3.15 se muestran algunas de las iteraciones del Algoritmo CLI-VA (Alg. 5) aplicado a las imágenes sintéticas de la Figura 2.7. Los parámetros utilizados fueron $w_{max} = 128$, $T = 1$ y $N_{iter} = 10$. En las Figuras 3.15(a-d) se muestran los valores de W para cada iteración, ilustrados con tonos de gris, donde el valor máximo corresponde a color blanco y el mínimo al color negro (se han marcado los bordes de P para que sirvan como referencia al lector). Las Figuras 3.15(e-h) muestran ejemplos de ventanas para los valores de W en cada iteración. Las Figuras 3.15(i-l) muestran la evolución del mapa de decisión P. Finalmente, las Figuras 3.15(m-o) muestran los bordes de los mapas de decisión correspondientes. Cada columna de la Figura 3.15 corresponde a las iteraciones 1, 2, 3 y 10, respectivamente. Dado que inicialmente no se tiene información de los bordes de segmentación, los tamaños de la ventana tomarán un valor constante w_{max} en todas las coordenadas de la imagen. Se puede notar que, en la primera iteración, el mapa de decisión será equivalente a la solución del Algoritmo 3 (CLI-S) con un valor $w = w_{max}$. A medida que se avanza en las iteraciones, $W_{r,c}$ toma valores pequeños cerca de los bordes permitiendo mejorar la segmentación y valores cercanos a w_{max} en las regiones homogéneas (sin bordes), los cuales maximizan la coherencia espacial del mapa de decisión y de la imagen fusionada. Esta situación se ilustra en las Figuras 3.15(e-h) donde se puede observar que el tamaño de ventana se adapta a la existencia de bordes y que el cambio más drástico sucede de la iteración 1 (sin bordes) a la iteración 2.

Para ilustrar el desempeño del algoritmo CLI-VA utilizamos cuatro pares de



(a) $W^{(1)}$ **(b)** $W^{(2)}$ **(c)** $W^{(3)}$ **(d)** $W^{(10)}$

(e) $W^{(1)}$ **(f)** $W^{(2)}$ **(g)** $W^{(3)}$ **(h)** $W^{(10)}$

(i) $P^{(1)}$ **(j)** $P^{(2)}$ **(k)** $P^{(3)}$ **(l)** $P^{(10)}$

(m) $B^{(1)}$ **(n)** $B^{(2)}$ **(ñ)** $B^{(3)}$ **(o)** $B^{(10)}$

Figura 3.15: Evolución de los bordes, tamaños de ventana y del mapa de decisión para las imágenes sintéticas de los lobos.

imágenes multi-foco sintéticas obtenidas a partir de las imágenes de los lobos y Lena mostradas en las Figuras 3.16(a) y 3.16(b), respectivamente.

(a) *Imagen de lobos enfocada* **(b) *Imagen de Lena enfocada***

Figura 3.16: Imágenes utilizadas para crear los conjuntos multi-foco sintéticos

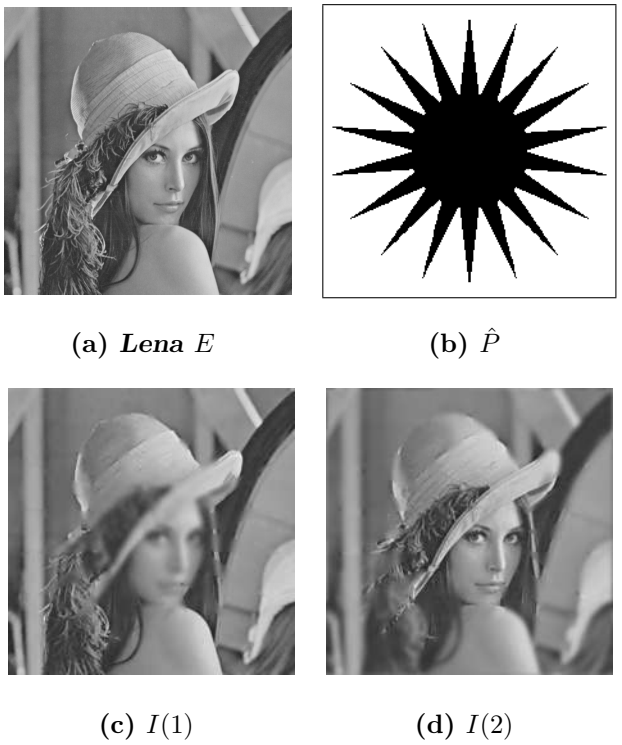
Para crear los pares de imágenes sintéticas utilizamos tres diferentes mapas de decisión, el primero coincide con el contorno de uno de los dos lobos (Fig. 3.17(a)), el segundo con el contorno de Lena (Fig. 3.17(b)) y el tercero es una estrella de 18 picos (Fig 3.17(c)). El mapa de decisión de la estrella se utilizó tanto en la imagen de los lobos como en la de Lena para que el lector pueda comparar el desempeño de los algoritmos con un mapa de decisión que no coincide con los bordes de los objetos en las imágenes, además de analizar la capacidad del algoritmo para segmentar correctamente regiones puntiagudas en los mapas de decisión.

(a) *Lobos* **(b) *Lena*** **(c) *Estrella***

Figura 3.17: Mapas de decisión usados para generar los conjuntos multi-foco sintéticos.

En la Figura 3.18, se presenta un segundo par de imágenes sintéticas generado

a partir de la imagen mostrada en la Figura 3.18(a), utilizando (2.8) y el mapa $\hat{P}$ mostrado en 3.18(b).



(a) ***Lena*** E **(b)** $\hat{P}$

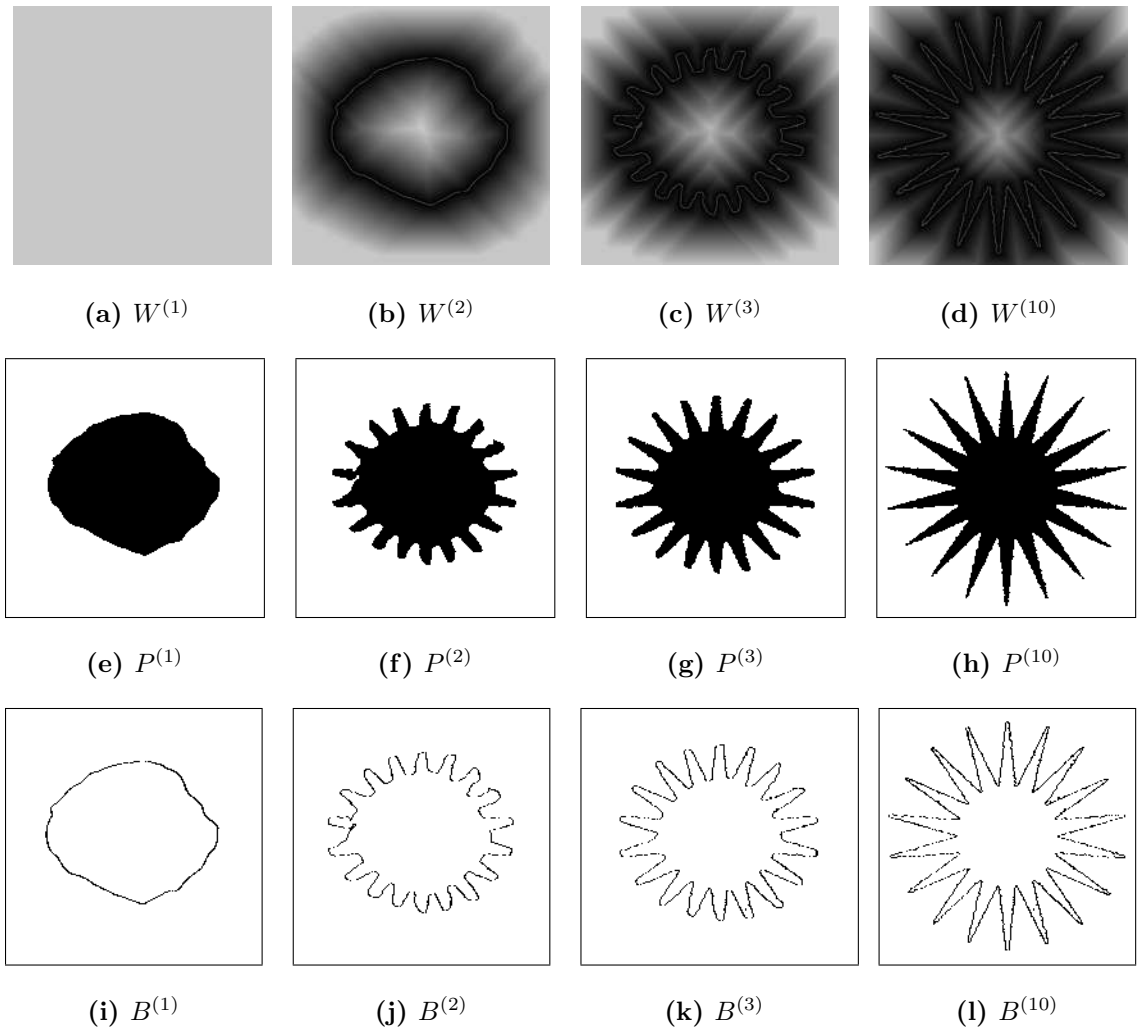
(c) $I(1)$ **(d)** $I(2)$

Figura 3.18: Simulación de perdida de nitidez utilizando la imagen de Lena y un mapa de decisión con forma de estrella

Se ha seleccionado un mapa de decisión P_0 con forma de estrella con 18 picos con el objetivo de mostrar el desempeño del Algoritmo CLI-VA con mapas de decisión con regiones puntiagudas.

En la Figura 3.19 se muestran los resultados de algunas iteraciones del Algoritmo CLI-VA para las imágenes multi-foco sintéticas de Lena creadas con el mapa de decisión mostrado en la Figura 3.18(b). Para la Fig 3.19 en el primer renglón (Figs. 3.19 (a-d)) contiene imágenes con los valores de W en las iteraciones 1, 2, 3 y 10, el segundo renglón (Figs. 3.19 (e-h)) muestra al mapa de decisión calculado y, el último renglón (Figs. 3.19 (i-l)) se muestra la evolución de los bordes. Cabe hacer notar que en cada iteración el mapa de decisión es mejorado hasta llegar al valor de porcentaje

de acierto de 99.37 %.



(a) $W^{(1)}$ **(b)** $W^{(2)}$ **(c)** $W^{(3)}$ **(d)** $W^{(10)}$

(e) $P^{(1)}$ **(f)** $P^{(2)}$ **(g)** $P^{(3)}$ **(h)** $P^{(10)}$

(i) $B^{(1)}$ **(j)** $B^{(2)}$ **(k)** $B^{(3)}$ **(l)** $B^{(10)}$

Figura 3.19: Evolución de los tamaños de ventana W, el mapa de decisión P y los bordes B para la imagen de Lena con un mapa de decisión en forma de estrella.

En la Figura 3.20 se muestran 4 columnas de imágenes, la primera columna tiene la imagen E y la segunda columna el mapa $\hat{P}$ utilizados para generar el conjunto multi-foco sintético. La tercera columna muestra el error del algoritmo CLI-S y finalmente la cuarta columna muestra el error obtenido con el algoritmo CLI-VA. En las imágenes de la Figura 3.20, negro significa que hay diferencia y blanco que no la hay. Se puede notar que el algoritmo CLI-VA presenta menores diferencias que el algoritmo CLI-S.

Para esta prueba se utilizó $w = 15$ como parámetro para CLI-S, el cual es el óptimo de acuerdo con la gráfica de la Figura 3.12. Para el caso del algoritmo CLI-VA se utilizaron como parámetros $w_{max} = 58$, $T = 5$ y $N_{iter} = 10$. Los parámetros se mantuvieron fijos para analizar el comportamiento de los algoritmos bajo diferentes condiciones de imágenes y mapas de decisión.

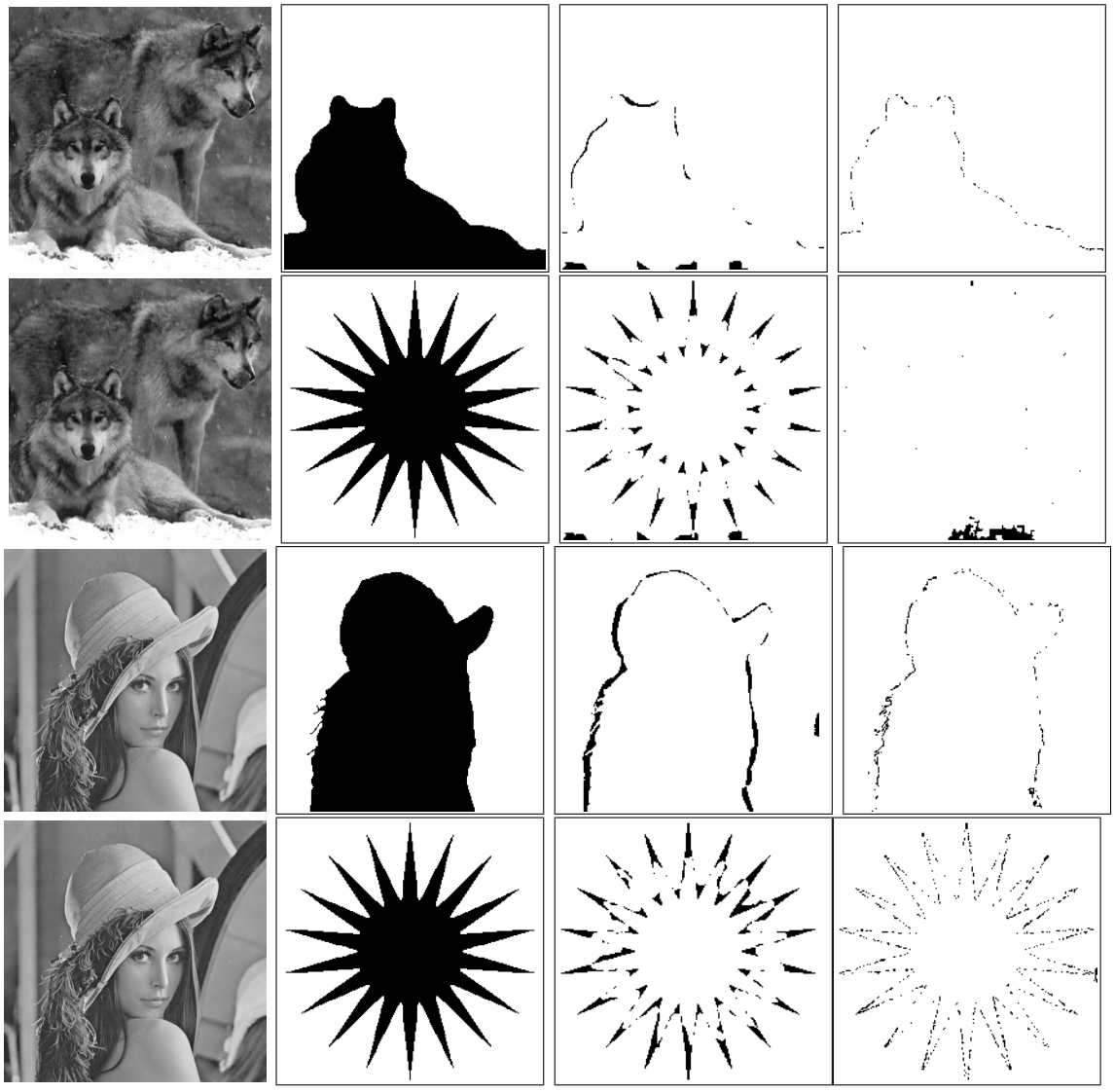


Figura 3.20: Diferencias entre el mapa $\hat{P}$ y el mapa calculado con CLI-S y CLI-VA.

En la Tabla 3.2 se muestra en la primera columna el nombre de la imagen utilizada para crear el conjunto multi-foco sintético, en la segunda columna el nombre del mapa de decisión utilizado, en las columnas 3 y 5 se muestra el pixel accuracy (PA) para el algoritmo CLI-S y CLI-VA, respectivamente. Finalmente, en las columnas 4 y 6 se muestra el tiempo consumido por el algoritmo CLI-S y CLI-VA, respectivamente. En todos los casos el Algoritmo CLI-VA supera al Algoritmo CLI-S, aunque el tiempo de ejecución consumido por CLI-VA es mayor que el requerido por CLI-S.

Tabla 3.2: Precisión a Nivel Pixel (PA) y tiempos obtenidos con los algoritmos CLI-S y CLI-VA.

Imagen	**Mapa**	**CLI-S**		**CLI-VA**	
		PA	Tiempo en seg.	PA	Tiempo en seg.
Lobos	Fig. 3.17(a)	98.4	0.16	**99.8**	0.256
	Fig. 3.17(c)	93.4	0.16	**98.1**	0.255
Lena	Fig. 3.17(b)	96.4	0.16	**99.3**	0.244
	Fig. 3.17(c)	93.0	0.16	**98.7**	0.239

La Figura 3.21 muestra los promedios de tiempo de ejecución en segundos del Algoritmo CLI-VA requerido para fusionar pares de imágenes con tamaños $n_p = n_r \times n_c$ pixeles. El Algoritmo CLI-VA se ejecutó 50 veces para cada par y se calculó el promedio de tiempo. Los tiempos promedio se graficaron en la Fig. 3.21, como puntos sobre la línea de mínimos cuadrados para resaltar el comportamiento lineal del Algoritmo CLI-VA.

3.6. Paralelización

Con varios núcleos integrados en un procesador es posible distribuir el flujo de trabajo de manera paralela en hilos, lo cual disminuye el tiempo de ejecución de

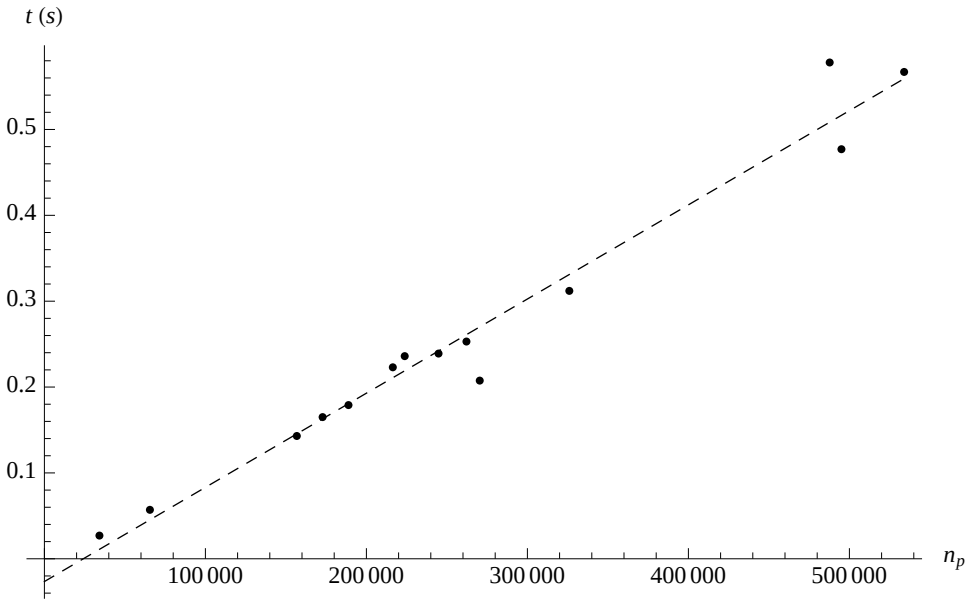


Figura 3.21: Tiempo de ejecución en s de CLI-VA vs. el número de pixeles n_p.

un algoritmo. Sin embargo, la paralelización de los programas no se lleva a cabo de forma automática y es responsabilidad del programador adaptar sus algoritmos para que puedan ejecutarse de forma paralela en los hilos del procesador.

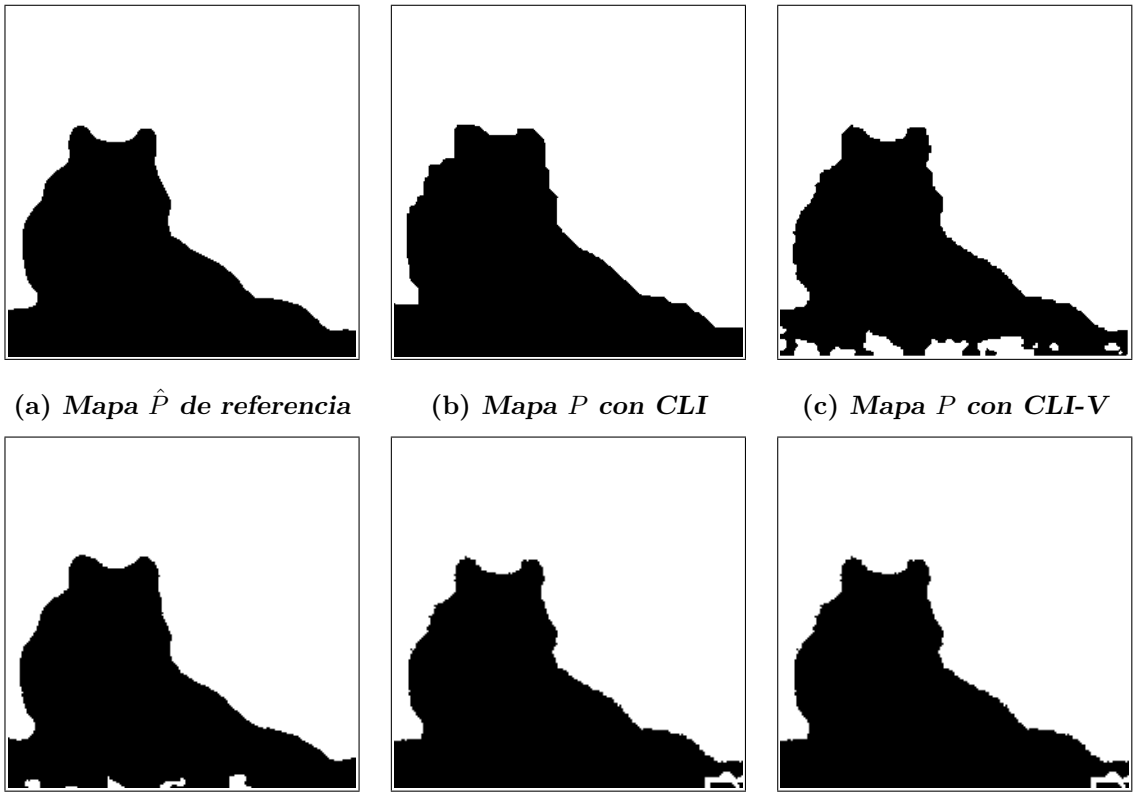
Considerando que $F(k)$ se calcula a priori para todas las imágenes, así como las imágenes integrales $H(k)$ correspondientes, podemos notar que el cálculo de un valor $P_{r,c}$, calculado con (3.25) en las coordenadas (r, c) es totalmente independiente de cualquier otro valor en coordenadas diferentes. Esta propiedad permite implementar el algoritmo en paralelo, donde cada uno de los hilos llevará a cabo el cálculo de los valores del mapa de decisión sobre una determinada región. Para determinar la región sobre la cual actuará cada hilo dividimos los renglones de la imagen entre el número de hilos N_h, de tal suerte que cada hilo calculará el mapa de decisión en RP renglones. Esta idea se refleja en el Algoritmo 6, Fusión de Imágenes con Ventanas Adaptables (FI-VA), en las líneas 8 a la 15. En particular en la línea 8 hay un ciclo que procesa en paralelo regiones de tamaño $RP \times n_c$, una región por cada hilo que participa en el proceso.

Algoritmo 6: Fusión de Imágenes con Ventanas Adaptables FI-VA(I, w_{max}, T, N_h)

$RP \leftarrow \lceil \frac{n_r}{N_h} \rceil$;
Calcular $F(k) \leftarrow$Nitidez de $I(k)$, $\forall k \in [1, 2, \ldots, N]$ usando (3.3).;
Calcular $H(k)$ $\forall k \in [1, 2, \ldots, N]$ usando (3.26);
$P_{r,c}^{(0)} \leftarrow 1$ $\forall (r, c) \in \{1, \ldots, n_r\} \times \{1, \ldots, n_c\}$;
para $iter \leftarrow 1$ **a** N_{iter} **hacer**
    $W^{(iter)} \leftarrow VO(w_{max}, T, P^{(iter-1)})$ (Alg. 4).;
    para $m \leftarrow 1$ **a** N_h *(en paralelo)* **hacer**
        para $r \leftarrow (m-1)RP$ **a** $(m)RP - 1$ & $r < n_r$ **hacer**
            para $c \leftarrow 1$ **a** n_c **hacer**
                $f_{max} \leftarrow \bar{\mathcal{F}}_{r,c}(1; W_{r,c}^{(iter)})$;
                para $k \leftarrow 2$ **a** N **hacer**
                    $f_{act} \leftarrow \bar{\mathcal{F}}_{r,c}(k; W_{r,c}^{(iter)})$;
                    si $f_{act} > f_{max}$ **entonces**
                        $f_{max} \leftarrow f_{act}$;
                        $P_{r,c}^{(iter)} \leftarrow k$;
Construir J utilizando (3.1);
Resultado: J

Se han aplicado todos los algoritmos presentados en éste capítulo para el ejemplo del conjunto de imágenes multi-foco sintético de los lobos, con el fin de mostrar una comparación cualitativa de los mapas de decisión obtenidos con cada algoritmo. La Figura 3.22(a) presenta el mapa de referencia con el que fueron generadas las imágenes sintéticas, en las Figuras 3.22(a-f) se muestran los mapas de decisión obtenidos con los algoritmos CLI, CLI-V, CLI-S, CLI-VA y FI-VA, respectivamente. En dichas imágenes se puede observar que los mapas de decisión que más se aproximan al mapa de referencia son los obtenidos con CLI-VA y FI-VA. Adicionalmente, se debe decir

que el mapa resultante de los algoritmos CLI-VA y FI-VA es el mismo puesto que FI-VA es una versión paralelizada de CLI-VA.



(a) ***Mapa*** $\hat{P}$ ***de referencia*** **(b)** ***Mapa*** P ***con CLI*** **(c)** ***Mapa*** P ***con CLI-V***

(d) ***Mapa*** P ***con CLI-S*** **(e)** ***Mapa*** P ***con CLI-VA*** **(f)** ***Mapa*** P ***con FI-VA***

Figura 3.22: Mapas de decisión dados por CLI, CLI-V, CLI-S, CLI-VA y FI-VA para las imágenes de los lobos.

En la Tabla 3.3 se muestran los el porcentaje de acierto y el tiempo de ejecución calculados al aplicar los algoritmos CLI, CLI-V, CLI-S, CLI-VA y FI-VA para fusionar las imágenes sintéticas de los lobos. La primera columna corresponde al nombre del algoritmo, la segunda a los parámetros utilizados en la implementación, la tercera al tiempo en segundos y la cuarta al porcentaje de acierto entre el mapa de decisión utilizado para crear las imágenes y cada uno de los mapas calculados. Se puede notar que el porcentaje de aciertos es muy similar para todos los algoritmos, sin embargo, el Algoritmo FI-VA es el más rápido y con el mayor porcentaje de acierto.

Tabla 3.3: Comparación de los algoritmos CLI, CLI-V, CLI-S, CLI-VA y FI-VA con las imágenes multi-foco de los lobos.

Alg.	Parámetros	tiempo (seg).	% de aciertos
CLI	$\epsilon = 0.03$	5641.36	98.38
CLI-V	$\epsilon = 0.01$ $w = 4$ $\Delta = 3$	160.99	98.3
CLI-S	$w = 31$	0.16	98.39
CLI-VA	$w_{max} = 128$, $T = 1$	0.255	99.19
FI-VA	$w_{max} = 128$, $T = 1$, $N_h = 8$	**0.135**	**99.19**

3.7. Análisis de complejidad

El Algoritmo 6 (FI-VA) recibe un conjunto de imágenes, el tamaño de ventana máximo a aplicar, la tolerancia de bordes por ventana y el número de hilos a utilizar y regresa la imagen fusionada.

Dado $n_p = n_r \times n_c$ en el Algoritmo 6, la línea 1 tiene complejidad constante $O(1)$, las líneas 2 y 3 tienen complejidad $O(N\ n_p)$, dado que barren matrices del mismo tamaño que las imágenes originales y se aplica para las N imágenes. La línea 4 tiene complejidad $O(n_p)$. El ciclo de la línea 5 se ejecuta N_{iter} veces por lo que tiene una complejidad $O(N_{iter})$. La línea 6 implementa la búsqueda binaria en el algoritmo VO, que tiene una complejidad $O(n_p\ log\ w_{max})$. El cuerpo del ciclo de las líneas 8 a 15 tiene una complejidad $O(n_p)$. Finalmente, el ciclo de la línea 11 es de complejidad lineal para la cantidad de imágenes $O(N)$. Entonces la complejidad total del algoritmo es $O\left(1 + n_p\left(2N + 1 + N_{iter}\left(log\ w_{max} + N\right)\right)\right)$. Dado que $n_p \gg N$, $n_p \gg N_{iter}$ y $n_p \gg log\ w_{max}$ y la constante es despreciable para este caso, la complejidad del algoritmo 6 es lineal con el número de píxeles $O(n_p)$, es decir con el tamaño del problema, lo cual corrobora lo que se muestra en la gráfica de la Figura 3.21.

3.8. Conclusiones del capítulo

Se ha presentado el algoritmo de Fusión de Imágenes con Ventanas Adaptables (FI-VA Alg.6), el cual surge como una mejora de los algoritmos CLI, CLI-S, CLI-V y CLI-VA; y permite fusionar un conjunto N de imágenes, donde $N \geq 2$, y las imágenes pertenecen a la misma escena, pero fueron capturadas bajo diferentes condiciones. FI-VA utiliza una función de aptitud F que permite identificar las regiones de interés de cada una de las imágenes del conjunto y, en base a dicha medición, calcula un mapa de decisión utilizando un sistema de ventanas deslizantes, donde el tamaño de la ventana aplicada en cada píxel es de tamaño adaptable según la distancia del píxel a los bordes. El mapa de decisión es refinado en un proceso iterativo, donde se calculan los bordes del mapa de decisión y los nuevos tamaños de ventana para obtener el nuevo mapa de decisión. Los tamaños de ventana se obtienen mediante búsqueda binaria y se utilizan imágenes incrementales para calcular la suma dentro de una ventana, con un tiempo constante e independiente del tamaño de la misma. En base a esto la imagen fusionada se obtiene en décimas de segundo. Se ha presentado también un análisis de la complejidad asintótica del algoritmo FI-VA, concluyendo que es de complejidad lineal con respecto a la cantidad de píxeles n_p, a diferencia de otras propuestas que tienen complejidad exponencial. Adicionalmente, se ha mostrado la independencia en el cálculo del valor de cada posición r, c del mapa de decisión, por lo que se ha presentado la paralelización del algoritmo, permitiendo con ello reducir aún más el tiempo de ejecución requerido.

Capítulo 4

Imágenes con iluminación no uniforme

La iluminación es un factor muy importante al momento de tomar fotografías, sin embargo, no siempre es posible controlar este aspecto en la escena, por lo que en muchas ocasiones las imágenes capturadas tienen regiones bien iluminadas y regiones que no lo están y se vuelve necesario procesarlas para corregir dicho problema. Este capítulo presenta la aplicación del algoritmo planteado en el capítulo anterior para fusionar y binarizar imágenes con iluminación no uniforme.

4.1. Fusión de imágenes con iluminación no uniforme

La fusión no es exclusiva de las imágenes multi-foco existen otros contextos donde se requiere fusionar información de dos o más imágenes que corresponden a la misma escena, pero obtenidas bajo diferentes condiciones. Para las imágenes multi-foco, la condición que cambia es el nivel de enfoque; sin embargo, existen otras condiciones que pueden cambiar al tomar una fotografía, por ejemplo, la iluminación o la exposición de

la lente a la luz. Algunas cámaras permiten controlar dicha exposición, sin embargo, si se reduce demasiado la exposición, las regiones con poca iluminación aparecerán obscuras en la imagen, tal y como se muestra en la Figura 4.1(a); mientras que, si se aumenta demasiado la exposición a la luz, las regiones con mayor iluminación se saturan y aparecen como una región completamente blanca en la imagen, tal y como se muestra en la Figura 4.1(b).

(a) ***Imagen sub-expuesta*** **(b)** ***Imagen sobre-expuesta,***

Figura 4.1: Ejemplo de imágenes con diferente exposición a la luz.

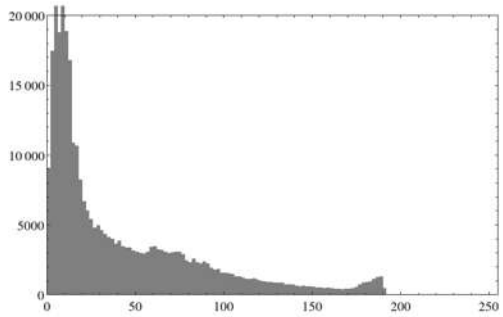
Las fotografías capturadas con muy poca iluminación son conocidas como bajo-expuestas o sub-expuestas y las capturadas con exceso de iluminación son llamadas sobre-expuestas. En las imágenes sub-expuestas no es posible ver los detalles de las zonas con falta de iluminación y, en las imágenes sobre-expuestas, no es posible observar los detalles de las regiones saturadas. Si se desea capturar los detalles de la escena completa será necesario cambiar la exposición del sensor, tomando dos o más fotografías como las mostradas en las Figuras 4.1(a) y 4.1(b), y fusionarlas para generar una imagen nueva con las regiones bien iluminadas de cada una de ellas.

El Algoritmo 6 (FI-VA) permite la fusión de imágenes con base en una función que permite medir la nitidez. En el caso de la fusión de imágenes con iluminación no uniforme, el objetivo es conservar las regiones con iluminación "adecuada"; es decir, que no están sobre-expuestas o sub-expuestas. La función utilizada debe medir la iluminación de cada pixel arrojando un valor mayor en regiones adecuadamente iluminadas y valores cercanos a cero para las regiones sobre-expuestas o sub-expuestas.

Para analizar la iluminación de una imagen podemos procesarla en escala de grises, donde el valor de cada pixel está directamente relacionado con la iluminación, donde 255 corresponde al color blanco obtenido con la mayor iluminación, mientras que 0 corresponde a negro y significa ausencia de iluminación. La iluminación en estas imágenes puede representarse con el histograma $\mathcal{H}$ donde $\mathcal{H}_j$ es el conteo de los pixeles de la imagen cuyo tono de gris es j.

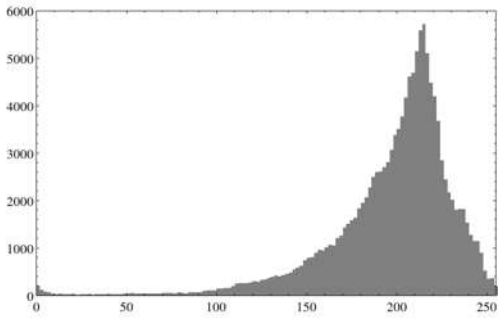
La Figura 4.2(a) muestra una imagen donde predominan los pixeles con valores cercanos a 0. En la Figura 4.2(b) se muestra el histograma correspondiente, el cual tiene conteos altos para valores cercanos a 0 y bajos para valores cercanos a 255. Mientras tanto, para imágenes con tonos de gris claros, como la mostrada en la Figura 4.2(c), el histograma tendrá forma similar al mostrado en la Figura 4.2(d), con una situación inversa a lo sucedido con la imagen obscura.

(a) ***Imagen con colores obscuros***



(b) ***Histograma de la Fig. 4.2(a)***

(c) ***Imagen con colores claros***

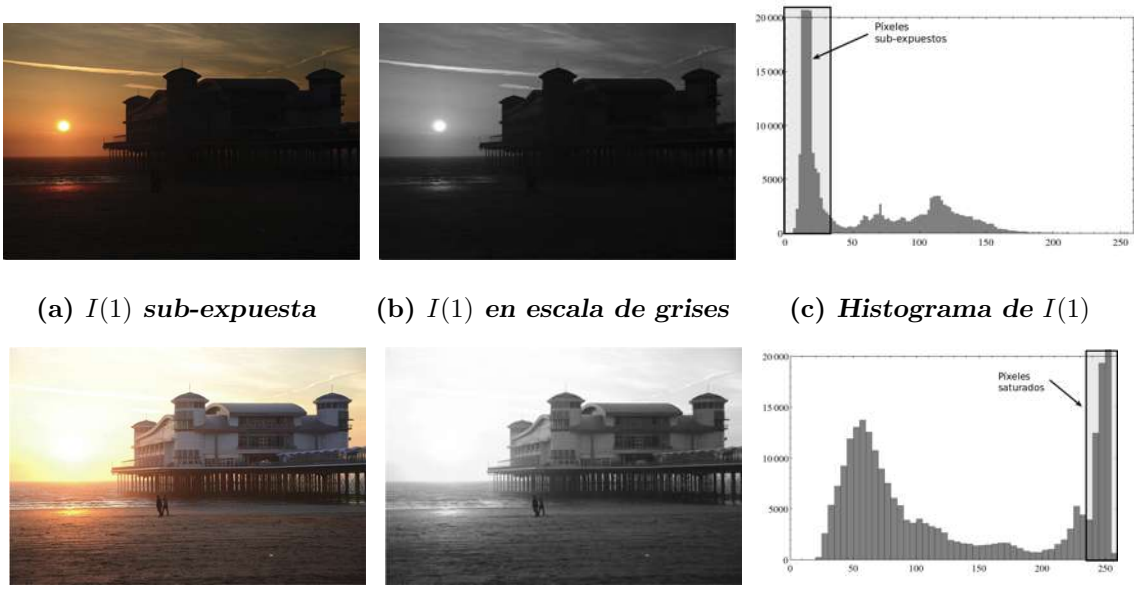


(d) ***Histograma de la Fig. 4.2(c)***

Figura 4.2: Imagen con colores obscuros y claros y sus histogramas.

Para nuestros propósitos supondremos que las imágenes sobre-expuestas o sub-expuestas tienen solamente dos clases de pixeles: los que están correctamente iluminados y los mal iluminados (saturados o sub-expuestos). Con el objetivo de separar dichas clases, para conservar los pixeles de la clase correctamente iluminada de cada imagen, transformamos las imágenes en color a imágenes en tonos de gris y calculamos el histograma de cada imagen en tono de gris, estableciendo $I(1)$ como la imagen con regiones sub-expuestas (muy obscuras) e $I(2)$ como la imagen con regiones saturadas.

En la Figura 4.3(a) se muestra una imagen sub-expuesta $I(1)$ en color, en la Figura 4.3(b) se muestra su representación en tonos de gris y, la Figura 4.3(c) muestra su histograma, donde se han marcado los valores de los pixeles que pertenecen a regiones sub-expuestas. En la Figura 4.3(d) se muestra una imagen sobre-expuesta $I(2)$ en color, la Figura 4.3(e) muestra la misma imagen en escala de grises y, en la Figura 4.3(f) aparece el histograma correspondiente, donde se han marcado los valores de los pixeles que pertenecen a regiones saturadas de la imagen.



(a) $I(1)$ ***sub-expuesta*** **(b)** $I(1)$ ***en escala de grises*** **(c)** ***Histograma de*** $I(1)$

(d) $I(2)$ ***sobre-expuesta*** **(e)** $I(2)$ ***en escala de grises*** **(f)** ***Histograma de*** $I(2)$

Figura 4.3: Ejemplo de histogramas de imágenes con diferente exposición a la luz.

Los valores los umbrales mostrados en las Figuras 4.3(c) y 4.3(f), que separan los pixeles saturados (regiones donde el exceso de iluminación no permite observar los detalles de la escena) o sub-expuestos (regiones donde falta de iluminación no permite observar los detalles de la escena), se han establecido manualmente, tratando de encontrar el umbral con que separa los pixeles mal iluminados. Sin embargo, este procedimiento no es práctico. Por tal motivo, es necesario determinar de forma automática la separación entre la clase que está correctamente iluminada y la que no.

Una forma de separar dos clases de pixeles de una imagen k es estableciendo un valor que permita separarlas mediante el histograma, a este valor se le llama umbral, donde los pixeles que tienen un valor de tono de gris inferior al umbral son clasificados como clase $\mathcal{C}_1(k)$ y el resto son de la clase $\mathcal{C}_2(k)$.

En términos ideales, el histograma de una imagen con dos clases de pixeles es como el mostrado en la Figura 4.4, donde se puede apreciar que existen dos cúmulos de valores y que el umbral que separa las clases se puede determinar fácilmente. Sin embargo, para imágenes reales la separación de las clases no es evidente, tal como se puede apreciar en los histogramas mostrados en las Figuras 4.3(c) y 4.3(f).

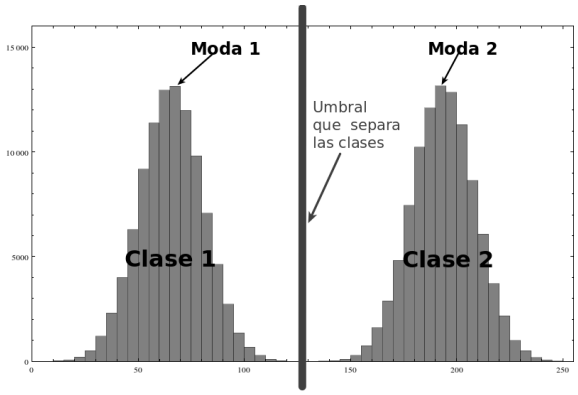


Figura 4.4: Ejemplo de histograma de imagen en escala de grises con dos clases

Una alternativa para separar las clases es utilizar técnicas de clustering como k-medias. El problema es que estos métodos generalmente son lentos y no garantizan

una correcta separación de las clases cuando existe ruido. *Otsu* propuso que el umbral óptimo que separa dos clases para imágenes en escala de grises es el que minimiza la varianza intra-clase y maximiza la varianza inter-clase, [Otsu, 1979]. Además, demostró que minimizar la varianza intra-clase es equivalente a maximizar la varianza inter-clase. La varianza está dada por (4.1).

$$\sigma_B^2(u) = \frac{[\mu_T \mathcal{W}(u) - \mu(u)]^2}{\mathcal{W}(u)[1 - \mathcal{W}(u)]} \tag{4.1}$$

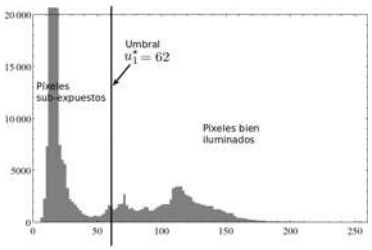
donde μ_T es la media de todos los pixeles de la imagen, $\mathcal{W}(u)$ y $\mu(u)$ son los momentos acumulativos de orden 0 y 1 del histograma respectivamente, los cuales se obtienen como $\mathcal{W}(u) = \sum_{i=1}^{u} p_i$ y $\mu(u) = \sum_{i=1}^{u} ip_i$ y se pueden calcular de forma simple con el histograma. Así, el umbral óptimo u_k que separa las dos clases de una imagen k está dado por (4.2).

$$u_k^\star = \underset{1 \leq u_k \leq 255}{\text{argmáx}}\ \sigma_B^2(u_k) \tag{4.2}$$

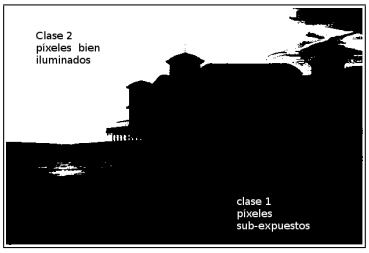
Los umbrales que separan las clases de las imágenes mostradas en las Figuras 4.5(a) y 4.5(d), de acuerdo con (4.2), son $u_1^\star = 62$ y $u_2^\star = 147$, respectivamente. En las Figuras 4.5(b) y 4.5(e) se muestran los histogramas de las imágenes 4.5(a) y 4.5(d), respectivamente, donde se señala el umbral que separa las clases de cada imagen. En la Figura 4.5(c) se muestra la clasificación de cada pixel de la imagen 4.5(a), donde negro significa que es de la clase $\mathcal{C}_1(1)$ y está sub-expuesto, mientras que blanco significa que es de la clase $\mathcal{C}_2(1)$ y está bien iluminado. En la Figura 4.5(f) se muestra la clasificación de cada pixel de la imagen 4.5(d), donde negro significa que es de la clase $\mathcal{C}_1(2)$ y está bien iluminado mientras que blanco significa que es de la clase $\mathcal{C}_2(2)$ y está saturado.

En la Figura 4.6 se muestran en negro las coordenadas que están bien iluminadas en 4.5(a) y no lo están en 4.5(d), en blanco las que están bien iluminadas en 4.5(d) y mal iluminadas en 4.5(a) y en gris las regiones que se clasificaron igual en ambas imágenes (bien iluminadas o mal iluminadas). En dicha imagen podemos observar que a pesar de haber separado las clases de cada imagen, aún no es posible realizar

(a) $I(1)$ ***en escala de grises***

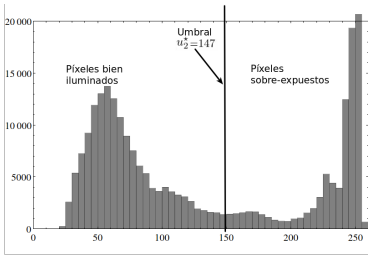


(b) ***Umbral en*** $\mathcal{H}(1)$

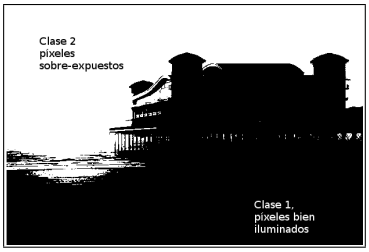


(c) ***Separación de clases de*** $I(1)$

(d) $I(2)$ ***en escala de grises***



(e) ***Umbral en*** $\mathcal{H}(2)$



(f) ***Separación de clases de*** $I(2)$

Figura 4.5: Ejemplo de imágenes con diferente exposición a la luz.

la fusión debido a que existen regiones que no es posible determinar en qué imagen están mejor iluminadas.

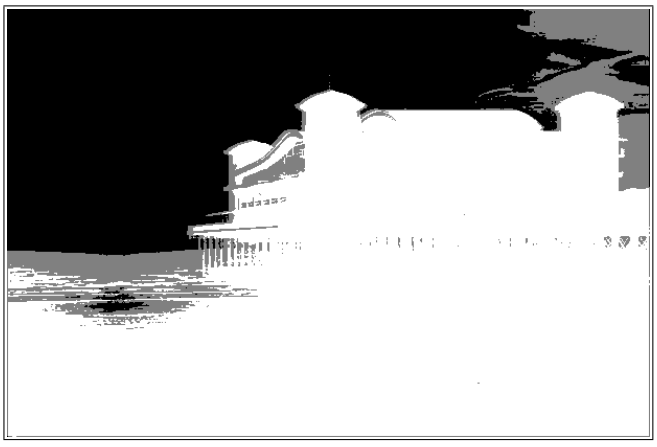


Figura 4.6: Mapa de decisión usando la separación por umbral.

Una vez determinados los umbrales podemos separar las clases y estimar la moda de cada una de ellas, con el fin de obtener el valor más representativo. La moda $m_j(k)$ de los pixeles pertenecientes a la clase $\mathcal{C}_j(k)$ puede ser estimada de forma simple en base al histograma usando (4.3).

$$m_j(k) = i^\star = \underset{i \in R_j}{\text{argmáx}}\ \mathcal{H}_i(k) \tag{4.3}$$

donde $R_1 \in [1, 2, \ldots, u_k^\star]$, $R_2 \in [u_k^\star + 1, u_k^\star + 2, \ldots, 255]$ y $j \in [1, 2]$.

El objetivo de fusionar imágenes con iluminación no uniforme es conservar los pixeles mejor iluminados; es decir, los que no están en la clase $\mathcal{C}_1(1)$ (sub-expuestos) de la primera imagen imagen ni en la clase $\mathcal{C}_2(2)$ (saturados) de la segunda imagen. Por lo tanto, mediremos la "adecuada" iluminación de la imagen, como el valor absoluto de la diferencia entre cada pixel $I_{r,c}(k)$ y la moda $m_k(k)$, la cual se puede calcular con (4.4), la cual arroja valores altos para pixeles bien iluminados y valores cercanos a cero para pixeles de regiones saturadas o sub-expuestas según el caso.

$$F_{r,c}(k) = |m_k(k) - I_{r,c}(k)| \tag{4.4}$$
$$k \in [1, 2]$$

En las Figuras 4.7(b) y 4.7(d) se muestra el resultado de calcular $F(k)$ utilizando (4.4) para las imágenes 4.7(a) y 4.7(c). En estas figuras los valores mayores se muestran en blanco y en negro los valores cercanos a cero.

Aplicando el algoritmo FI-VA para fusionar las imágenes mostradas en las Figuras 4.7(a) y Fig.4.7(c) se obtiene el mapa de decisión mostrado en la Figura 4.8(a), con el cual se extraen las regiones mejor iluminadas de cada imagen (Figuras 4.8(b) y 4.8(c)) y se combinan para obtener la imagen fusionada, mostrada en la Figura 4.8(d), en la que se pueden observar de forma clara los detalles de la escena completa. Los bordes que se aprecian en la imagen fusionada se pueden corregir mediante suavizados, técnicas de ecualización o emparejamiento de histogramas. Estos bordes se han conservado para que el lector pueda observar las regiones que provienen de las imágenes originales.

(a) ***Imagen*** $I(1)$ **(b)** $F(1)$ ***para 4.7(a)***

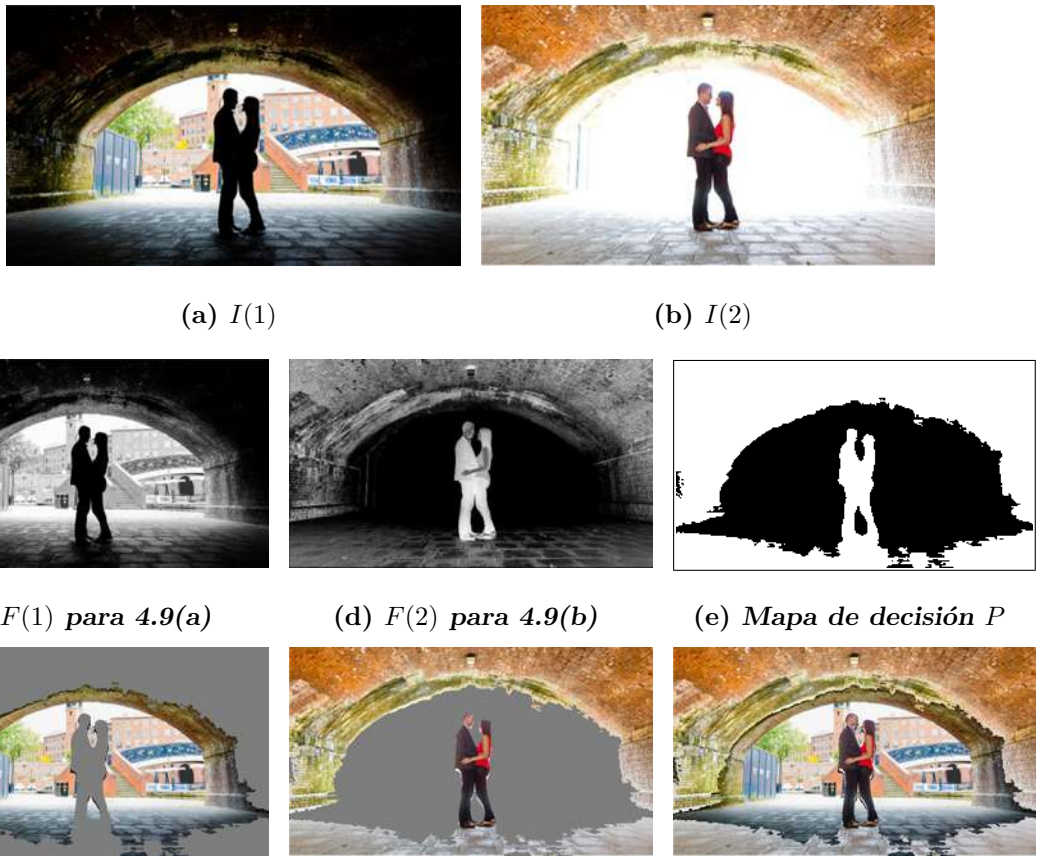
(c) ***Imagen*** $I(2)$ **(d)** $F(2)$ ***para 4.7(c)***

Figura 4.7: Medición de la iluminación usando la ecuación (4.4).

(a) **(b)** **(c)** **(d)**

Figura 4.8: Ejemplo de fusión de imágenes con iluminación no uniforme. a) mapa de decisión, b) y c) regiones de interés de 4.7(a) y 4.7(c), respectivamente, y d) imagen fusionada.

En la Figura 4.9 se muestra el resultado de aplicar el proceso de fusión con el algoritmo FI-VA usando (4.4) para las imágenes mostradas en las Figuras 4.9(a) y 4.9(b). En las Figuras 4.9(c) y 4.9(d) se muestra $F(1)$ y $F(2)$, respectivamente. La Figura 4.9(e) muestra el mapa de decisión obtenido. En las Figuras 4.9(f) y 4.9(g) se muestran las regiones bien iluminadas de las imágenes 4.9(a) y 4.9(b), respectivamente. Finalmente, en la Figura 4.9(h) se muestra la imagen fusionada, donde son visibles todos los detalles de la escena y no existen regiones saturadas ni sub-expuestas.



(a) $I(1)$ **(b)** $I(2)$

(c) $F(1)$ ***para 4.9(a)*** **(d)** $F(2)$ ***para 4.9(b)*** **(e)** ***Mapa de decisión*** P

(f) ***Regiones bien iluminadas de 4.9(a)*** **(g)** ***Regiones bien iluminadas de 4.9(b)*** **(h)** ***Imagen fusionada***

Figura 4.9: Ejemplo de fusión de imágenes con iluminación no uniforme para imágenes de dos personas en un túnel, las cuales fueron extraídas de Internet.

4.2. Binarización de imágenes con iluminación no uniforme

La binarización es una técnica que permite resaltar la información utilizando únicamente dos posibles valores para cada pixel. Un caso particular es la escritura sobre fondo blanco, donde 1 representa fondo y 0 representa texto.

En la Figura 4.10 se muestra un ejemplo de la binarización de una imagen en color que contiene texto escrito a mano Fig. 4.10(a). Después del proceso de binarización,

la imagen de la Fig. 4.10(a) luce como la Figura 4.10(b). El objetivo de la binarización es eliminar el fondo de la imagen correspondiente al color del papel, para contrastar los valores de los pixeles y hacer más legible la información.

(a) ***Imagen en color*** **(b)** ***Imagen binarizada***

Figura 4.10: Ejemplo de binarización de imágenes de texto.

El problema de binarización puede ser visto como un problema de segmentación, donde se separa el fondo de la información; también puede ser interpretado como clasificación, donde se etiquetan los pixeles. En la literatura se reportan distintas propuestas para resolver el problema, entre las más comunes están las que utilizan un umbral para separar las clases. Algunas de estas técnicas utilizan un umbral global, como en [Otsu, 1979], donde se propone encontrar el umbral maximizando la varianza inter-clase, haciendo 0 los pixeles que tienen valor menor o igual al umbral y estableciendo en 255 los que su valor supera el umbral. El método propuesto por *Otsu* y, en general, los que utilizan un umbral global, fallan cuando las imágenes tienen cambios de iluminación o texturas que impiden la separación clara de las clases de pixeles. Por lo anterior, han surgido propuestas que utilizan un sistema de ventanas deslizantes y un umbral para cada ventana, dentro de estas propuestas podemos mencionar [Bernsen, 1986; Sauvola y Pietikäinen, 2000; Bradley y Roth, 2007; Khurshid et˜al., 2009; Singh et˜al., 2012]. En particular, el método propuesto por *Bradley y Roth* es muy popular por su velocidad y fácil implementación. Los autores proponen calcular el umbral con la media en una ventana de tamaño 15×15 pixeles, la cual se obtiene utilizando imágenes integrales, logrando un algoritmo que permite su implementación en tiempo real. El umbral dentro de cada ventana en el método propuesto por *Bradley*

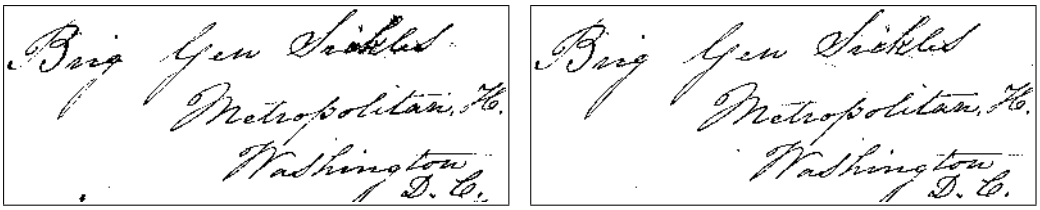
y Roth se calcula con (4.5).

$$T_{r,c} = \mu_{r,c}\frac{100-\tau}{100} \tag{4.5}$$

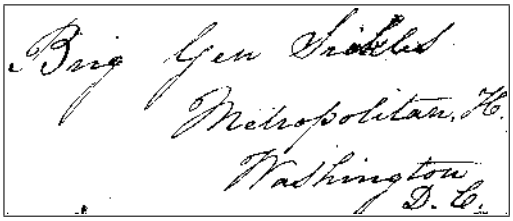
donde $\mu_{r,c}$ es la media de los valores en una ventana y τ es una constante que establece el porcentaje de separación entre el valor del pixel y la media en la ventana. *Bradley y Roth* proponen utilizar $\tau = 15$.

Otra posibilidad para binarizar imágenes consiste en separar las dos clases mediante técnicas de clustering como k-medias.

En las Figuras 4.11(a), 4.11(b) y 4.11(c) se muestra la binarización obtenida utilizando los métodos propuestos por *Otsu*, por *Bradley y Roth* y k-medias, respectivamente. En esta figura podemos observar que los tres métodos probados logran una buena binarización separando el texto y el fondo de manera adecuada.



(a) ***Método de Otsu*** **(b)** ***Método de Bradley y Roth.***



(c) ***k-medias***

Figura 4.11: Ejemplo de binarización de imágenes usando métodos del estado del arte.

4.2.1. Imágenes con iluminación no uniforme

La iluminación es un factor trascendental al capturar una imagen, puesto que una iluminación inadecuada produce imágenes con regiones no visibles. En la Figura 4.12

se presentan ejemplos de imágenes capturadas con iluminación no uniforme, las cuales tienen regiones muy obscuras en las que no es posible observar claramente los datos, e incluso, los algoritmos de binarización tienen problemas con estas imágenes.

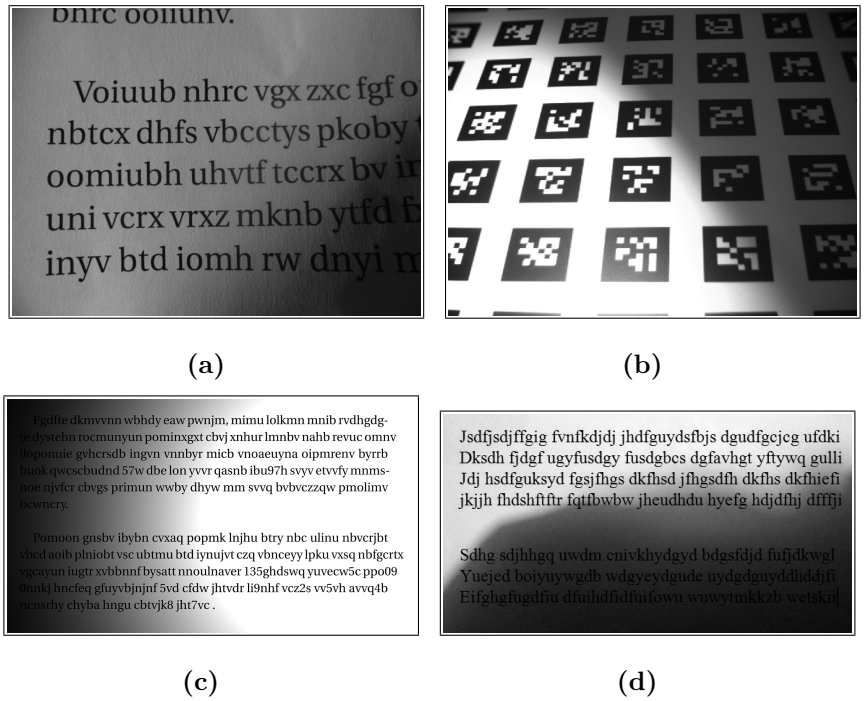
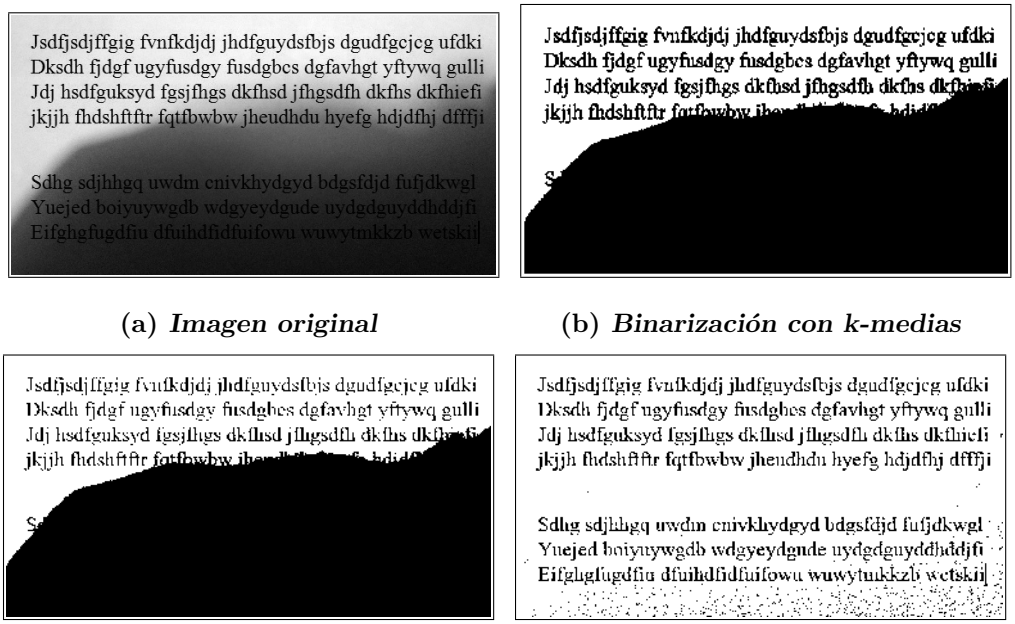


Figura 4.12: Ejemplo de imágenes con iluminación no uniforme.

En la Figura 4.13(a) se muestra un ejemplo de una imagen con iluminación no uniforme. En las Figuras 4.13(b), 4.13(c) y 4.13(d) se muestran los resultados de aplicar k-medias, y los métodos propuestos por *Otsu* y *Bradley y Roth*, respectivamente. En este caso k-medias y el método propuesto por *Otsu* no logran una binarización adecuada, esto debido a que procesan la imagen a nivel global y el cambio de iluminación afecta demasiado el proceso. El método propuesto por *Bradley y Roth* logra un mejor resultado, sin embargo, también falla al clasificar algunos pixeles debido a que el tamaño de ventana es fijo y no se adapta a las condiciones de la imagen.

Idealmente, el histograma de imágenes binarias con fondo blanco tiene una forma como el de la Figura 4.14, donde se distingue un grupo de pixeles con valores cercanos a 0, otro grupo con valores cercanos a 255 que corresponden al fondo y no existen



(a) ***Imagen original*** **(b)** ***Binarización con k-medias***

(c) ***Método de Otsu*** **(d)** ***Método de Bradley y Roth.***

Figura 4.13: Ejemplo de binarización de imágenes usando métodos del estado del arte.

valores entre 75 y 180. En este caso es sencillo notar la separación de clases y la binarización resulta simple, solamente es necesario buscar un umbral (entre 75 y 180) y mapear los valores por debajo de éste a 0 y los superiores a 255.

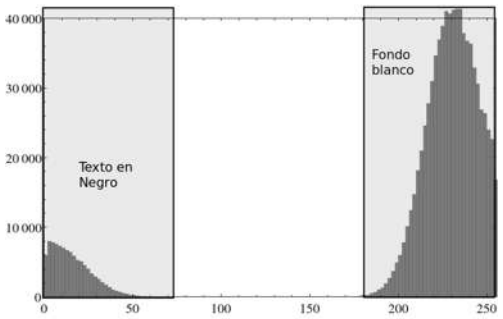
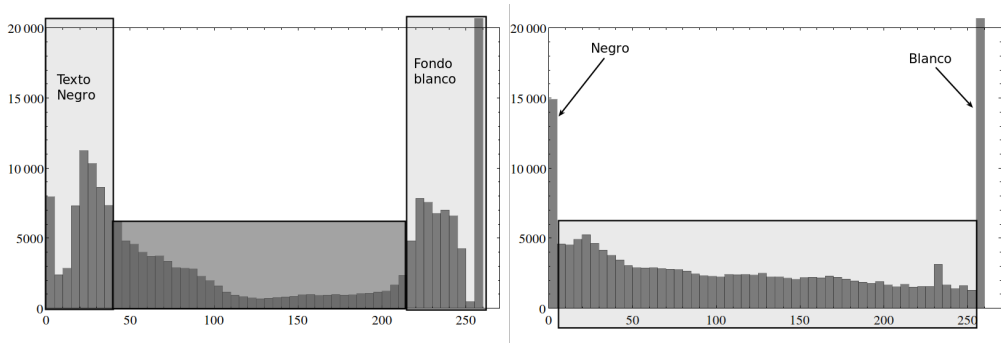


Figura 4.14: Ejemplo de histograma de imagen binaria con fondo blanco.

Sin embargo, en los histogramas de imágenes con iluminación no uniforme, la separación entre las clases no es evidente y resulta muy complejo encontrar el umbral adecuado para binarizar. Esta es la razón por la que los métodos de umbralización

global, como el planteado por *Otsu*, fallan en dichos casos. En las Figuras 4.15(a) y 4.15(b) se muestran dos ejemplos de histogramas de imágenes con iluminación no uniforme. En estos histogramas podemos notar que, aunque es fácil identificar el blanco y el negro absolutos, un alto porcentaje de los pixeles con valores entre 40 y 210 para el primer histograma, y entre 5 y 250 para el segundo, están distribuidos y no es trivial clasificarlos como texto o como fondo.

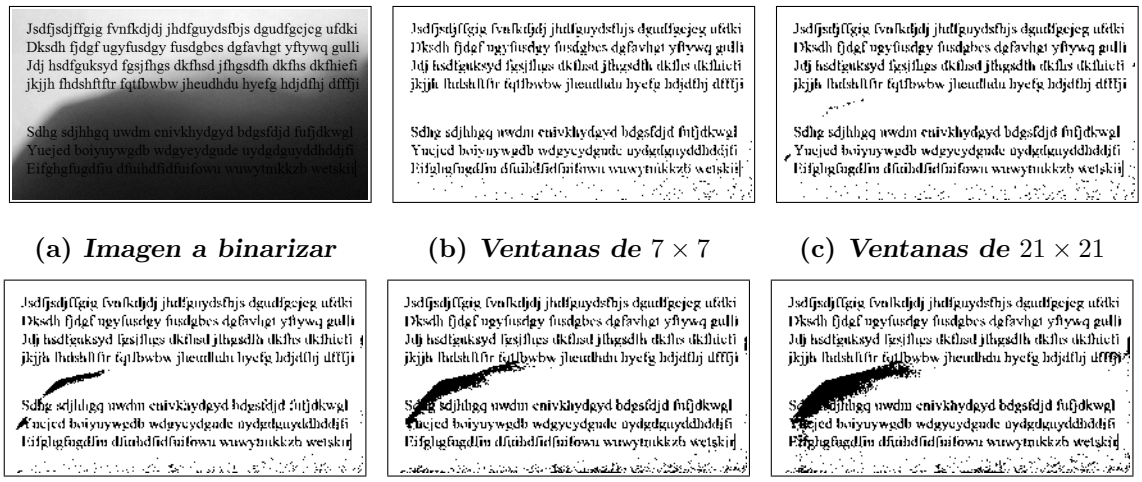


(a) **(b)**

Figura 4.15: Ejemplo de binarización de imágenes usando métodos del estado del arte.

Muchos métodos de binarización basan su funcionamiento en el cálculo de un umbral. Las propuestas que usan un único umbral global se ven severamente afectadas por la iluminación no uniforme. Por otro lado, los métodos donde se calculan umbrales locales dentro de una ventana de tamaño $(2w+1) \times (2w+1)$ tienen la desventaja de que requieren establecer un tamaño de ventana a priori y que, un tamaño de ventana inadecuado puede causar una binarización incorrecta. En la Figura 4.16 se muestra un ejemplo de la aplicación del método propuesto por *Bradley y Roth* variando el tamaño de ventana para binarizar la imagen mostrada en la Figura 4.16(a). Las Figuras 4.16(b-f) muestran los resultados de aplicar la binarización para $w = 7$, $w = 10$, $w = 20$, $w = 35$ y $w = 45$, respectivamente. Para este ejemplo se puede notar que es preferible utilizar ventanas de tamaño pequeño en la binarización.

Las Figuras 4.17(b-f) muestran los resultados de binarizar la imagen mostrada en la Figura 4.17(a) con $w = 7$, $w = 10$, $w = 20$, $w = 35$ y $w = 45$, respectivamente. Para

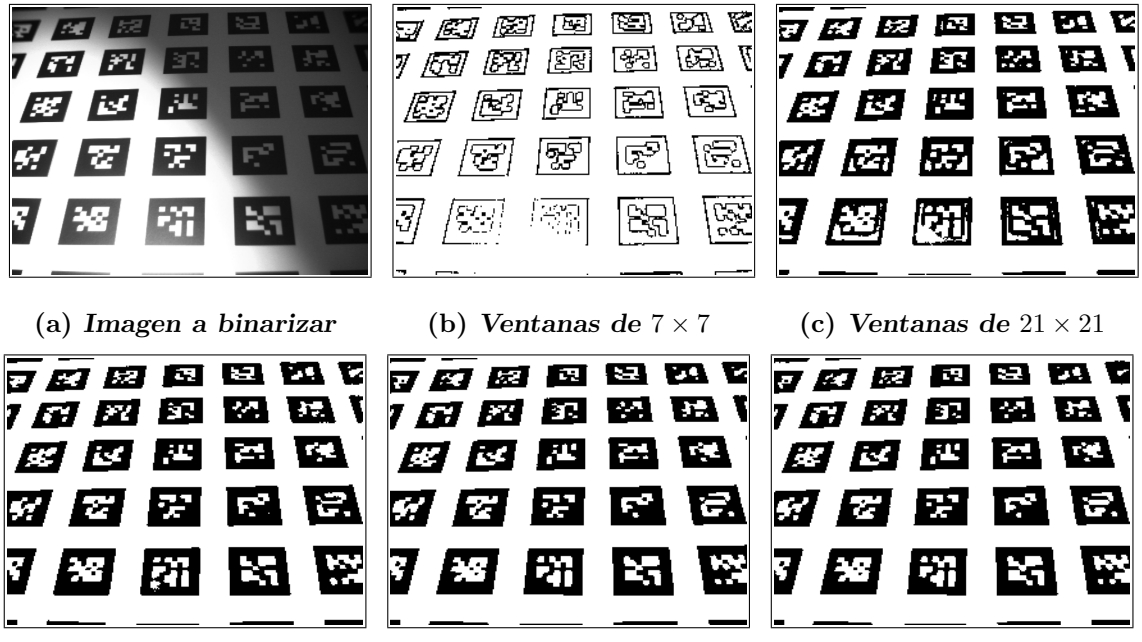


(a) ***Imagen a binarizar*** **(b)** ***Ventanas de*** 7×7 **(c)** ***Ventanas de*** 21×21

(d) ***Ventanas de*** 41×41 **(e)** ***Ventanas de*** 71×71 **(f)** ***Ventanas de*** 91×91

Figura 4.16: Binarización con el método de *Bradley* variando el tamaño de ventana.

este ejemplo, los mejores resultados se obtienen con tamaños de ventana grandes; por ejemplo, con $w = 7$ podemos ver que el relleno de los cuadros de la imagen se pierde



(a) ***Imagen a binarizar*** **(b)** ***Ventanas de*** 7×7 **(c)** ***Ventanas de*** 21×21

(d) ***Ventanas de*** 41×41 **(e)** ***Ventanas de*** 71×71 **(f)** ***Ventanas de*** 91×91

Figura 4.17: Binarización con el método de *Bradley y Roth* variando w.

4.2.2. Binarización de imágenes con ventanas adaptables

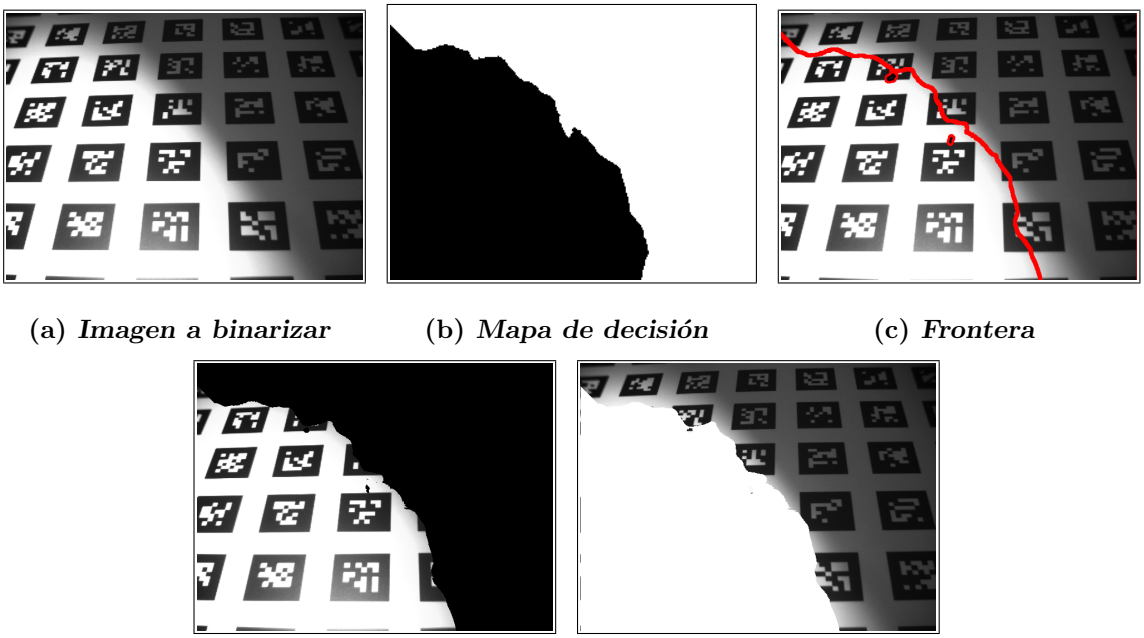
En la Sección 4.1 se presentó el uso del Algoritmo 6 (FI-VA) para la fusión de imágenes con iluminación no uniforme. Dichas imágenes presentan un problema similar al de las imágenes mostradas en la Figura 4.12, por lo que se puede aplicar una función similar para medir la iluminación. Dado que en binarización se tiene únicamente una imagen calcularemos $F(1)$ y $F(2)$ con (4.6), para estimar una medida de la iluminación de la imagen permitiendo con ello, encontrar un mapa de decisión que permite separar las zonas iluminadas de la imagen de las que no lo están.

$$F_{r,c}(k) = |m_k(1) - I_{r,c}(1)| \tag{4.6}$$

En la Figura 4.18 se muestra el resultado de aplicar el Alg. 6 utilizando (4.6). En la Figura 4.18(a) se muestra la imagen original, la Figura 4.18(b) muestra el mapa de decisión, en la Figura 4.18(c) se muestran los bordes resaltados sobre la imagen original y, finalmente, en las Figuras 4.18(d) y 4.18(e) se muestra la separación de las regiones clara y obscura, respectivamente.

A pesar de que la separación de las regiones clara y obscura, mostrada en las Figuras 4.18(d) y 4.18(e) no es la solución al problema de binarización, esta separación representa un avance significativo, puesto que así es posible aplicar técnicas de binarización a cada región por separado, con lo cual se reduce el problema generado por el cambio brusco de iluminación.

Un subproducto del algoritmo (FI-VA) es el mapa de decisión, el cual usaremos en este caso para identificar las regiones que tienen buena iluminación y las que no. Adicionalmente, durante el proceso para obtener el mapa de decisión se calculan los tamaños de ventana adecuados para cada pixel de la imagen. Se utiliza esta información para aplicar una variante del método propuesto por *Bradley y Roth* pero, en lugar de utilizar ventanas de tamaño fijo, se aplican ventanas de tamaño variable delimitado por los bordes del cambio de iluminación en la imagen original, reduciendo con ello la problemática del cambio de iluminación.



(a) ***Imagen a binarizar*** **(b)** ***Mapa de decisión*** **(c)** ***Frontera***

(d) ***Región clara*** **(e)** ***Región obscura***

Figura 4.18: Ejemplo de aplicación del algoritmo FI-VA para identificar la iluminación.

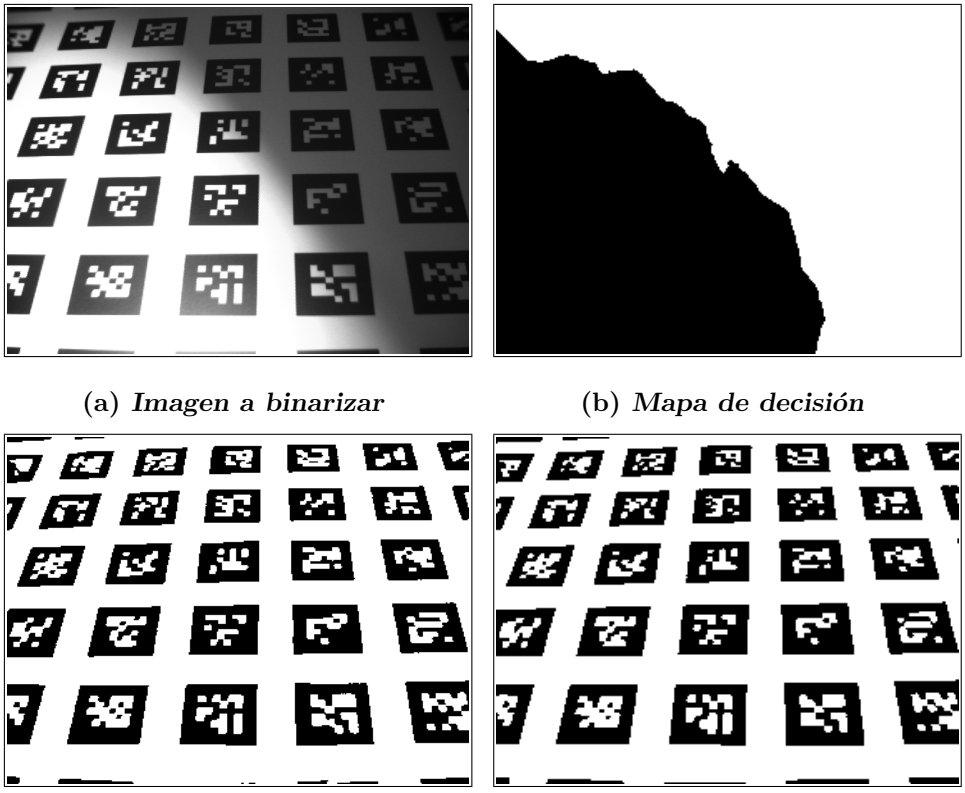
El Algoritmo 7, al que llamamos Binarización de Imágenes con Ventanas Adaptables (BI-VA), resulta de una modificación del Algoritmo 6 (FI-VA) y puede ser utilizado para la binarización de imágenes.Cabe hacer notar que el Algoritmo 7 no sufrió cambios en su operación esencial, puesto que hasta la línea 15 la única variante es la función utilizada para calcular $F(k)$, que es una medida de la iluminación. Desde la línea 16 hasta la línea 24 se realiza en paralelo el cálculo de la imagen binarizada, utilizando los tamaños de ventana previamente calculados. El objetivo es aplicar ventanas de tamaño mayor en regiones donde no hay cambios de iluminación y ventanas de tamaño más pequeño en las regiones cercanas a los bordes formados por el cambio de iluminación. Esta estrategia reduce el efecto que causa el cambio de iluminación en el proceso de binarización. Adicionalmente, se puede utilizar un valor diferente para τ en cada región (clara/obscura), puesto que ambas regiones tienen características de iluminación diferentes y, por ende, deben ser tratadas de forma distinta, lo cual sólo es posible gracias a la separación previa de dichas regiones.

Algoritmo 7: Binarización de Imágenes con Ventanas Adaptables BI-VA(I, w_{max}, T, N_h)

$RP \leftarrow \lceil \frac{n_r}{N_h} \rceil$;
Calcular $F(k)$ y $H(k)$ $\forall k \in \{1, 2\}$ con (4.6) y (3.26), respectivamente;
$P_{r,c}^{(0)} \leftarrow 1$ $\forall (r, c) \in \{1, \ldots, n_r\} \times \{1, \ldots, n_c\}$;
para $iter \leftarrow 1$ **a** N_{iter} **hacer**
    $W^{(iter)} \leftarrow VO(w_{max}, T, P^{(iter-1)})$ (Alg. 4);
    para $m \leftarrow 1$ **a** N_h *(en paralelo)* **hacer**
        para $r \leftarrow (m-1)RP$ **a** $(m)RP - 1$ & $r < n_r$ **hacer**
            para $c \leftarrow 1$ **a** n_c **hacer**
                $f_{max} \leftarrow \bar{\mathcal{F}}_{r,c}(1; W_{r,c}^{(iter)})$;
                para $k \leftarrow 2$ **a** N **hacer**
                    $f_{act} \leftarrow \bar{\mathcal{F}}_{r,c}(k; W_{r,c}^{iter})$;
                    si $f_{act} > f_{max}$ **entonces**
                        $f_{max} \leftarrow f_{act}$;
                        $P_{r,c}^{(iter)} \leftarrow k$;
$M^{(inc)} \leftarrow$ Imagen incremental de $I(1)$;
para $m \leftarrow 1$ **a** N_h *(en paralelo)* **hacer**
    para $r \leftarrow (m-1)RP$ **a** $(m)RP - 1$ & $r < n_r$ **hacer**
        para $c \leftarrow 1$ **a** n_c **hacer**
            $\mu_{r,c} \leftarrow \frac{M_{r+w,c+w}^{(inc)}(k) - M_{r+w,c-w-1}^{(inc)}(k) - M_{r-w-1,c+w}^{(inc)}(k) + M_{r-w-1,c-w-1}^{(inc)}(k)}{(2W_{r,c}^{(iter)}+1)^2}$;
            si $I_{r,c}(1) < \frac{100-\tau}{100}\mu_{r,c}$ **entonces**
                $IB_{r,c} \leftarrow 0$;
            en otro caso
                $IB_{r,c} \leftarrow 255$;
Resultado: IB

4.2.3. Aplicación del algoritmo BI-VA a imágenes con iluminación no uniforme

En la Figura 4.19 se muestra el resultado de aplicar el algoritmo BI-VA (Figura 4.19(c)) y el método propuesto por *Bradley y Roth* (Figura 4.20(d)), para la binarización de la imagen presentada en la Figura 4.19(a). Se puede apreciar que ambos métodos logran una correcta binarización teniendo un desempeño muy similar para este caso. En la Figura 4.19(b) se muestra el mapa de decisión que permite la separación de las regiones claras y obscuras y cuyos bordes son utilizados para el cálculo del tamaño de las ventanas a aplicar en la binarización.



(a) ***Imagen a binarizar*** **(b)** ***Mapa de decisión***

(c) ***Resultado (BI-VA)*** **(d)** ***Resultado Bradley y Roth***

Figura 4.19: Resultados de aplicar el algoritmo BI-VA y el método propuesto por *Bradley y Roth* con la imagen de la Figura 4.19(a).

En la Figura 4.20 se muestra otro ejemplo de la aplicación del algoritmo BI-VA y

el método de *Bradley y Roth*, pero ahora sobre una imagen cuyo contenido es texto. Puede apreciarse que ambos métodos logran una binarización aceptable, sin embargo, el método propuesto por *Bradley y Roth* presenta ruido en la imagen binarizada, lo cual se debe al tamaño fijo de ventana, mientras que el algoritmo BI-VA arroja letras más delgadas en el borde de iluminación, lo cual se puede deber a que la tolerancia aplicada para el cálculo de W no es la indicada.

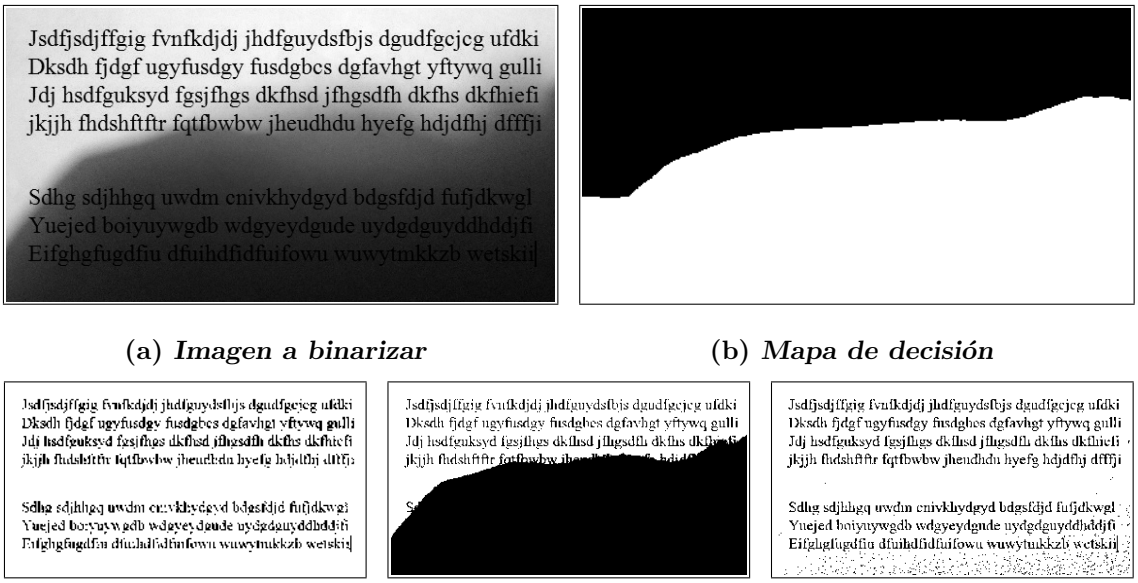
(a) ***Imagen a binarizar*** **(b)** ***Mapa de decisión***

(c) ***Resultado (BI-VA)*** **(d)** ***Resultado Bradley y Roth***

Figura 4.20: Resultados de aplicar el algoritmo BI-VA y el método de *Bradley y Roth* para binarizar una imagen con texto.

En la Figura 4.21(a) se muestra una imagen en escala de grises con texto sobre un fondo blanco y en las Figuras 4.21(c) y 4.21(e) se muestra la binarización lograda con

el método de *Bradley y Roth* y la obtenida con el algoritmo BI-VA respectivamente. En este caso podemos ver que el algoritmo BI-VA tiene un mejor desempeño al evitar en gran medida la presencia de ruido en la imagen binarizada, lo cual se logra gracias a la aplicación de tamaños de ventana adaptable, los cuales están en función de los bordes del mapa de decisión mostrado en la Figura 4.21(b).



(a) ***Imagen a binarizar*** **(b)** ***Mapa de decisión***

(c) ***Resultado (BI-VA)*** **(d)** ***Método de Otsu*** **(e)** ***Resultado Bradley y Roth***

Figura 4.21: Resultados de aplicar el algoritmo BI-VA y los métodos de *Otsu* y de *Bradley y Roth* para binarizar la imagen de la Figura 4.21(a)

4.3. Conclusiones del capítulo

En éste capítulo se presentó la aplicación del Algoritmo 6 (FI-VA) para la fusión de imágenes con iluminación no uniforme con la finalidad de obtener una imagen correctamente iluminada, a partir de dos imágenes: una con regiones bajo-expuestas y otra con regiones sobre-expuestas. Adicionalmente se presenta el Algoritmo 7 (BI-VA) que permite la binarización de imágenes con iluminación no uniforme, el cual

proviene de una modificación del Algoritmo 6 (FI-VA). La adecuación consiste en aplicar los tamaños de ventana calculados en la última iteración del cálculo del mapa de decisión, a fin de binarizar la imagen reduciendo el impacto que produce el cambio de iluminación al calcular el umbral.

Se presentan algunos ejemplos de la aplicación del algoritmo BI-VA, con resultados que cualitativamente son competitivos con el método propuesto por *Bradley y Roth* y que superan el desempeño de la propuesta de *Otsu* para imágenes con iluminación no uniforme con las que se probó.

La binarización de imágenes con iluminación no uniforme se presenta como una aplicación del algoritmo propuesto en este trabajo de tesis, con el objetivo de ejemplificar que se puede aplicar en distintos ámbitos del procesamiento de imágenes, para fusionar, segmentar, binarizar, clasificar, etc., cambiando la función a maximizar por una que se adapte al problema que se desea resolver.

Capítulo 5

Segmentación de imágenes por color

La segmentación es utilizada en el procesamiento de imágenes para extraer información de interés o que cumple con alguna característica. La segmentación también es utilizada para clasificar o identificar objetos en una imagen. El término clasificación, en procesamiento de imágenes, consiste en la actividad de ordenar o separar un conjunto de objetos (pixeles) en grupos.

En el caso de la fusión de imágenes, un mapa de decisión contiene el índice de la imagen que tiene mayor respuesta a la función de nitidez, para cada pixel. Dicho mapa de decisión también puede ser utilizado para segmentación o clasificación asumiendo que el valor de cada coordenada es una etiqueta de la clase a la que pertenece el pixel. Para la aplicación del algoritmo de fusión presentado en este trabajo a la segmentación, debemos hacer notar que existen dos diferencias principales entre estos procesos:

1. La segmentación se aplica a una sola imagen original O, mientras que la fusión se aplica sobre u conjunto de imágenes.

2. En la fusión se utiliza el mapa de decisión para formar la imagen fusionada,

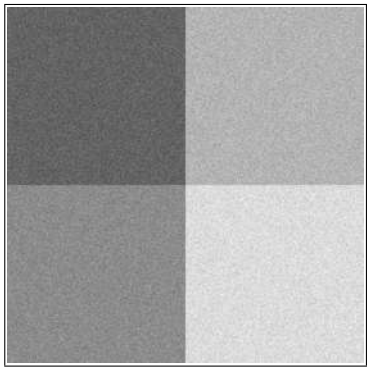
mientras que en la segmentación el mapa de decisión es utilizado para crear un conjunto de N imágenes, donde cada imagen k contenga sólo los pixeles de la k-ésima clase.

Dadas las diferencias y similitudes entre la fusión y segmentación, el algoritmo FI-VA puede ser aplicado para segmentar una imagen, realizando pequeños cambios:

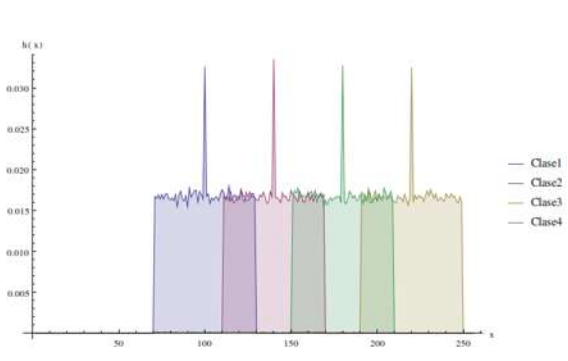
- Cambiar los parámetros, para recibir sólo una imagen O de entrada y un conjunto de ejemplos de cada clase, en lugar de un conjunto de imágenes I.
- Cambiar la función de nitidez por una función de verosimilitud que mida la semejanza de cada pixel de la imagen $O_{r,c}$ con cada una de las clases. Así, $F_{r,c}(1)$ tiene una medida de similitud entre $O_{r,c}$ y la clase 1, $F_{r,c}(2)$ es la medida de similitud entre $O_{r,c}$ y la clase 2 y así sucesivamente.
- Regresa el mapa de decisión o N imágenes, cada una con los pixeles de cada k-ésima clase.

5.1. Problema de segmentación de imágenes

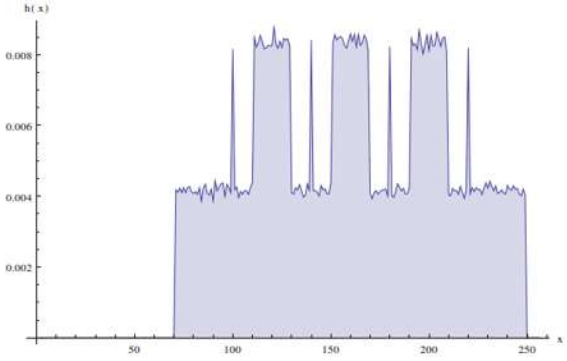
En la Figura 5.1(a) se muestra un ejemplo con una imagen O en escala de grises (con valores de 0 a 255 en cada pixel), donde se aprecian 4 clases, cada una con valores aleatorios entre $\mu_k - 30$ y $\mu_k + 30$ y una distribución uniforme. Se utiliza $\mu_1 = 100$, $\mu_2 = 140$, $\mu_3 = 180$ y $\mu_4 = 220$ para las clases de 1 a 4, respectivamente. Lo anterior con el objetivo de simular la existencia de ruido en la imagen y permitir el traslape entre las distribuciones de las clases, tal como se muestra en la Figura 5.1(b), donde se muestran los histogramas de cada una de las clases por separado. En la Figura 5.1(c) se muestra el histograma de la imagen 5.1(a) y en ella se puede apreciar que no existe una separación clara entre las clases. El objetivo de la segmentación en este caso, es separar cada una de las clases existentes en la Figura 5.1(a).



(a) *Imagen Original* O



(b) *Histogramas de cada clase*



(c) *Histograma de la imagen* O

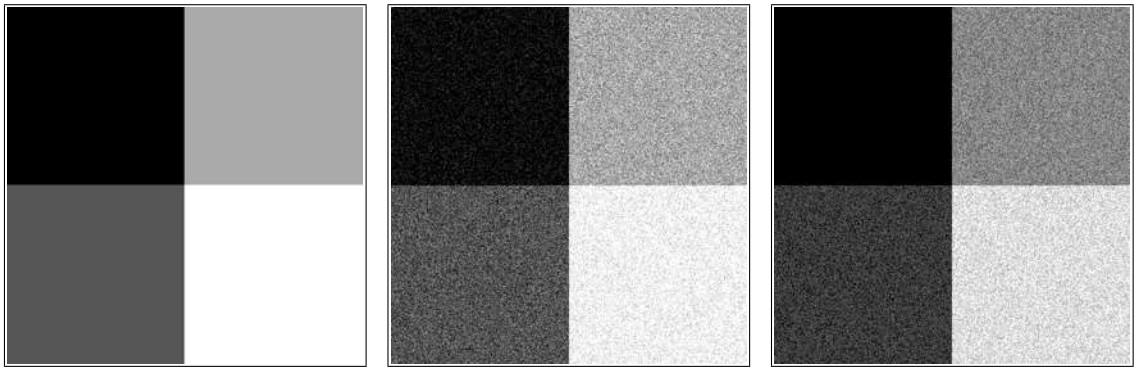
Figura 5.1: Ejemplo de segmentación de imágenes con 4 clases.

Para segmentar cada clase de la imagen mostrada en la Figura 5.1(a) asignamos a cada posición $P_{r,c}$ la etiqueta k que minimiza la distancia entre μ_k y el valor de la imagen O en dicha coordenada. Entonces, $P_{r,c}$ se puede calcular con (5.1).

$$\begin{aligned} P_{r,c} = k^{\star} = \underset{k}{\operatorname{argmin}} \; |O_{r,c} - \mu_k| \\ \text{s. a.} \\ k \in [1, 2, 3, 4] \end{aligned} \tag{5.1}$$

En la Figura 5.2(a) se muestra el mapa de decisión de referencia para segmentar la imagen mostrada en la Figura 5.1(a). La Figura 5.2(b) corresponde al mapa de decisión obtenido al aplicar (5.1) y, la Figura 5.2(c), muestra la segmentación obtenida al utilizar k-medias. Nótese el error en la clasificación obtenida con estos dos

clasificadores puesto que, debido al traslape entre las clases y al ruido en la imagen, no fueron capaces de arrojar una segmentación correcta.



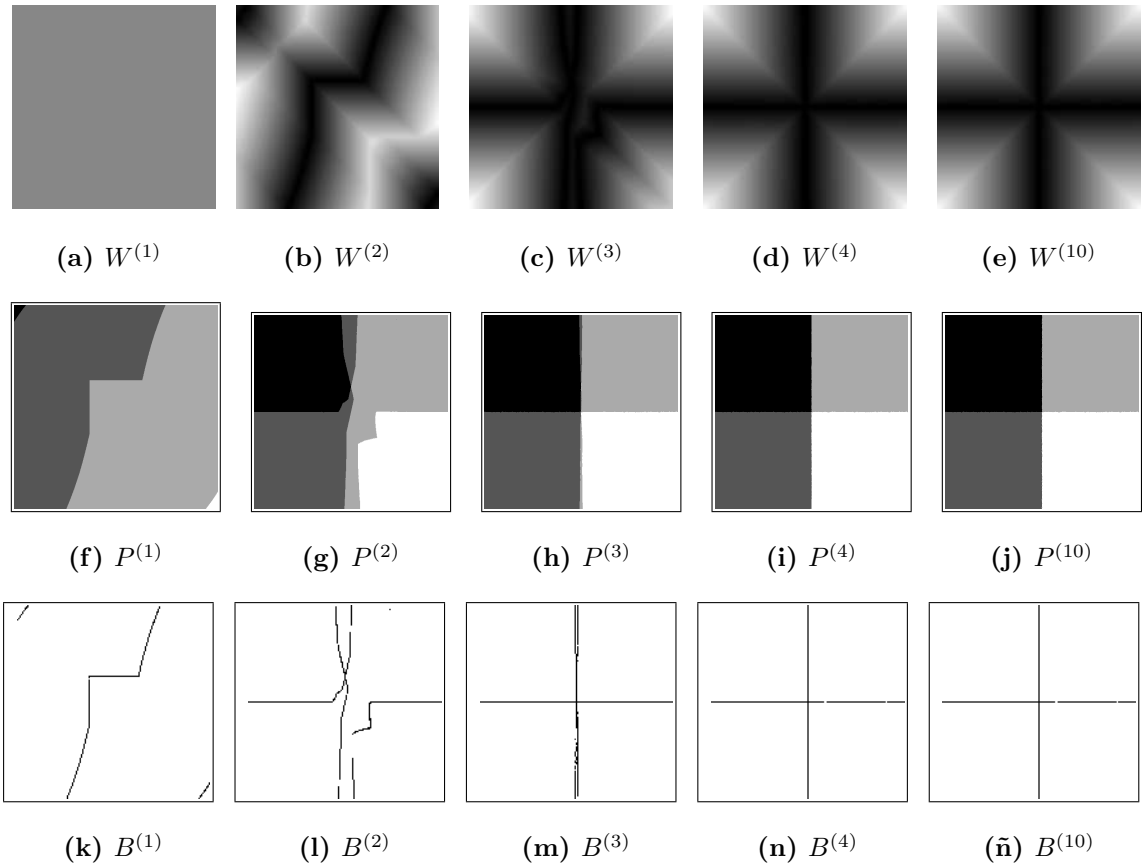
(a) ***Segmentación esperada*** **(b)** ***Segmentación con*** (5.1) **(c)** ***Segmentación K-medias***

Figura 5.2: Segmentación obtenida al aplicar (5.1).

5.2. Uso de FI-VA en segmentación de imágenes

Para aplicar el algoritmo FI-VA en la segmentación de la imagen mostrada en la Figura 5.1(a), sólo es necesario cambiar la función de nitidez por una función de verosimilitud $F_{r,c}(k) = -|O_{r,c} - \mu_k|$ para $k \in [1, 2, 3, 4]$. En las Figuras 5.3(a-e) se muestra la representación de la matriz W para las iteraciones 1, 2, 3, 4 y 10, respectivamente. En dichas imágenes negro representa 0 y blanco representa un valor alto. Las Figuras 5.3(f-j) muestran el mapa de decisión obtenido en las iteraciones mencionadas. Finalmente, en las Figuras 5.3(k-ñ) se muestran los bordes del mapa de decisión correspondiente. Se puede notar que, en únicamente 3 iteraciones se logra un mapa de decisión aceptable, a pesar de haber iniciado con una muy mala aproximación al mapa de decisión esperado. Cabe mencionar que se aplicó un valor $w_{max} = 350$, que es un valor muy alto para el tamaño de la imagen, esto con el objetivo de probar el comportamiento del algoritmo cuando se seleccionan parámetros inadecuados.

El procedimiento anterior puede ser aplicado para imágenes en escala de grises,



(a) $W^{(1)}$ **(b)** $W^{(2)}$ **(c)** $W^{(3)}$ **(d)** $W^{(4)}$ **(e)** $W^{(10)}$

(f) $P^{(1)}$ **(g)** $P^{(2)}$ **(h)** $P^{(3)}$ **(i)** $P^{(4)}$ **(j)** $P^{(10)}$

(k) $B^{(1)}$ **(l)** $B^{(2)}$ **(m)** $B^{(3)}$ **(n)** $B^{(4)}$ **(ñ)** $B^{(10)}$

Figura 5.3: Evolución de los tamaños de ventana, mapa de decisión y bordes; para la segmentación de la imagen de la Figura 5.1(a).

segmentando por la similitud de cada pixel de la imagen $O_{r,c}$ con el valor medio de cada clase k. Sin embargo, este procedimiento no es lo suficientemente robusto para ser aplicado con imágenes en otros espacios de color como RGB. Nuestra propuesta para aplicar el algoritmo FI-VA a la segmentación de imágenes es cambiar la función de nitidez por una que mida la probabilidad de que un pixel $x = [x_r, x_g, x_b]$ pertenezca a la clase $\mathcal{C}_k$ de acuerdo con (5.2).

$$p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_{i=1}^{C} p(x|\mathcal{C}_i)p(\mathcal{C}_i)} \tag{5.2}$$

La probabilidad empírica o a priori de que una tripleta de valores $[R, G, B]$ pertenezca a la clase k puede ser estimada mediante el histograma de la clase, el cual se calcula en base al conjunto de pixeles que pertenecen a la clase $\mathcal{C}_k$ con N_k pixeles

de ejemplo de dicha clase [Calderon et˜al., 2015]. En las imágenes en RGB hay tres canales y cada canal tiene valores enteros de 0 a 255, por lo tanto, el histograma de la clase k es un arreglo $\mathcal{H}(k)$ de tamaño $256 \times 256 \times 256$; y cada coordenada $\mathcal{H}_{r,g,b}(k)$ contiene un conteo normalizado de los pixeles x de la clase k que cumplen con $x_r = r$, $x_g = g$ y $x_b = b$, de acuerdo con (5.3).

$$\mathcal{H}_{r,g,b}(k) = \frac{1}{N_k} \sum_{x \in \mathcal{C}_k} U(x, r, g, b) \tag{5.3}$$

$$U(x, r, g, b) = \begin{cases} 1 & \text{Si } x_r = r,\ x_g = g \text{ y } x_b = b \\ 0 & \text{En otro caso} \end{cases} \tag{5.4}$$

$$\forall r \in [0, 1, \ldots, 255],\ \ \forall g \in [0, 1, \ldots, 255],\ \ \forall b \in [0, 1, \ldots, 255]$$
$$\forall k \in [1, 2, \ldots, N]$$

La probabilidad a priori de que un pixel x esté en la clase k es estimada con (5.5) y la probabilidad a priori de la clase $\mathcal{C}_k$ puede ser establecida a un valor constante $p(\mathcal{C}_k) = \frac{1}{N}$ si, dadas N clases, suponemos que todas son equiprobables. Si la cantidad de ejemplos de cada clase es proporcional al área ocupada por los pixeles de esa clase en la imagen, entonces la probabilidad a priori de una clase puede ser estimada con (5.6).

$$p(x|\mathcal{C}_k) = \mathcal{H}_{x_r,x_g,x_b}(k) \tag{5.5}$$

$$p(\mathcal{C}_k) = \frac{N_k}{\mathcal{T}} \tag{5.6}$$

donde $\mathcal{T}$ es el total de pixeles proporcionados como ejemplos y se calcula como la suma de la cantidad de pixeles de cada una de las clases $\mathcal{T} = \sum_{k=1}^{N} |\mathcal{C}_k|$.

La función de verosimilitud $F(k)$ para el proceso de segmentación debe arrojar un valor alto, si el pixel $O_{l,m}$ es similar a los pixeles de la clase k y un valor cercano a 0 en caso contrario. Adicionalmente, el denominador de (5.2) es solamente un factor de normalización. Por lo tanto, calculamos $F_{l,m}(k)$ con (5.7).

$$F_{l,m}(k) = p(\mathcal{C}_k|O_{l,m}) = p(O_{l,m}|\mathcal{C}_k)p(\mathcal{C}_k) = \frac{N_k}{\mathcal{T}} \mathcal{H}_{x_r,x_g,x_b}(k) \tag{5.7}$$

5.3. Segmentación de Imágenes con Ventanas Adaptables (SI-VA)

El Algoritmo 8, llamado Segmentación de Imágenes con Ventanas Adaptables (SI-VA) es una modificación del Algoritmo 6 (FI-VA), para aplicarlo a segmentación de imágenes por color, a partir de una imagen original O y un conjunto de ejemplos $\mathcal{C}$, donde cada subconjunto $\mathcal{C}_k$ tiene ejemplos de la clase k. Cabe resaltar que los cambios

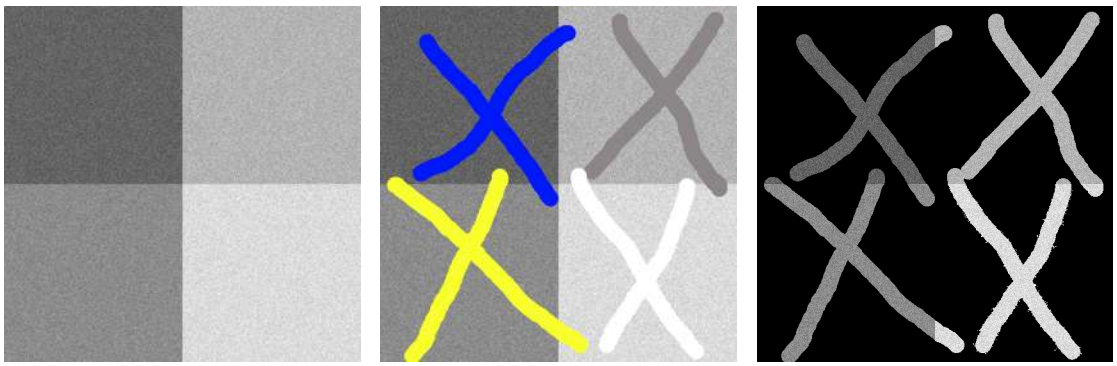
Algoritmo 8: Segmentación de Imágenes con Ventanas Adaptables SI-VA(O, $\mathcal{C}$, w_{max}, T, N_h)

```
1  RP ← ⌈n_r / N_h⌉;
2  Calcular F(k) ∀k ∈ {1, 2, ..., N} con (5.7);
3  Calcular H(k) ∀k ∈ {1, 2, ..., N} con (3.26);
4  P^(0)_{r,c} ← 1 ∀(r, c) ∈ {1, ..., n_r} × {1, ..., n_c};
5  para iter ← 1 a N_iter hacer
6      W^(iter) ← VO(w_max, T, P^(iter−1)) (Alg. 4);
7      para m ← 1 a N_h (en paralelo) hacer
8          para r ← (m − 1)RP a (m)RP − 1 & r < n_r hacer
9              para c ← 1 a n_c hacer
10                 f_max ← F̄_{r,c}(1; W^(iter)_{r,c});
11                 para k ← 2 a N hacer
12                     f_act ← F̄_{r,c}(k; W^iter_{r,c});
13                     si f_act > f_max entonces
14                         f_max ← f_act;
15                         P^(iter)_{r,c} ← k;
   Resultado: P
```

de SI-VA con respecto a FI-VA son sólo 3: recibe una imagen original y un conjunto de ejemplos de cada clase, en lugar de un conjunto de imágenes como en FI-VA; se

cambia la función de nitidez por la función de verosimilitud (5.7); y regresa el mapa de decisión, en lugar de una imagen fusionada. También es posible utilizar el mapa de decisión para calcular N imágenes, cada una con los pixeles de cada clase segmentada y regresar dichas imágenes si así se desea.

En la Figura 5.4(a) se muestra una imagen O, con 4 clases de pixeles. En la Figura 5.4(b) se marcan los pixeles de la imagen que servirán como ejemplos de cada clase; la Figura 5.4(c) muestra los pixeles que forman el conjunto de entrenamiento $\mathcal{C}$, donde los pixeles no marcados (aparecen en negro) forman el conjunto de validación. Los ejemplos están marcados: en azul la clase 1, en gris la clase 2, en amarillo la clase 3 y en blanco la clase 4; $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ y $\mathcal{C}_4$ respectivamente.

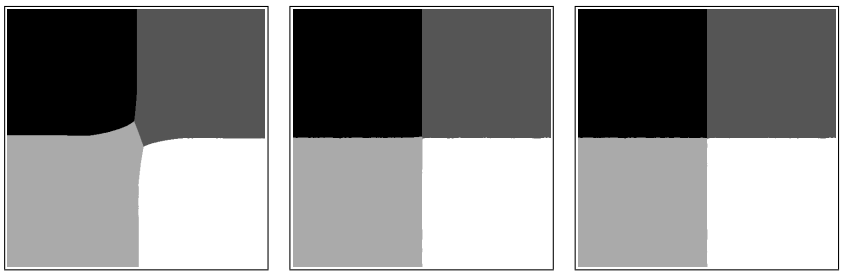


(a) ***Imagen*** O **(b)** ***Ejemplos*** **(c)** $\mathcal{C}$.

Figura 5.4: Ejemplo de segmentación de imágenes con 4 clases y presencia de ruido.

Utilizando la imagen O, mostrada en la Figura 5.4(a), y el conjunto de pixeles $\mathcal{C}$, que aparece en la Figura 5.4(b), como parámetros de entrada para el algoritmo SI-VA se obtiene como resultado el mapa de decisión mostrado en la Figura 5.5(c). Las Figuras 5.5(a-c) muestran los mapas de decisión obtenidos en las iteraciones 1 a 3, respectivamente. Se puede notar en estas figuras que, en la segunda iteración se logra el mapa de decisión esperado.

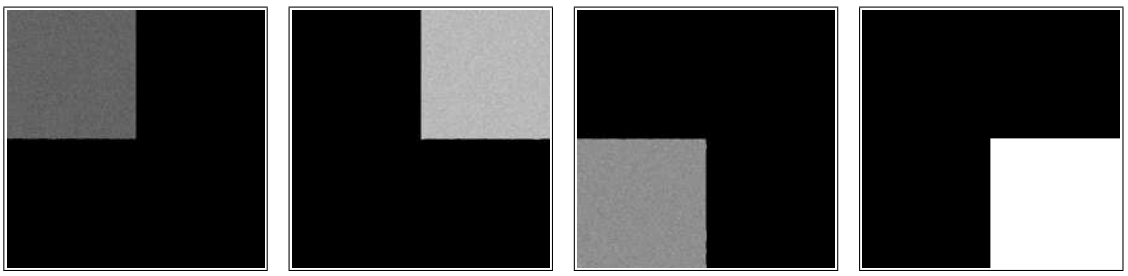
En las Figuras 5.6(a-d) se muestra la segmentación de cada una de las clases de la imagen mostrada en la Figura 5.4(a) utilizando el mapa de decisión obtenido



(a) $P^{(1)}$ **(b)** $P^{(2)}$ **(c)** $P^{(3)}$

Figura 5.5: Ejemplo de segmentación de imágenes con 4 clases con presencia de ruido.

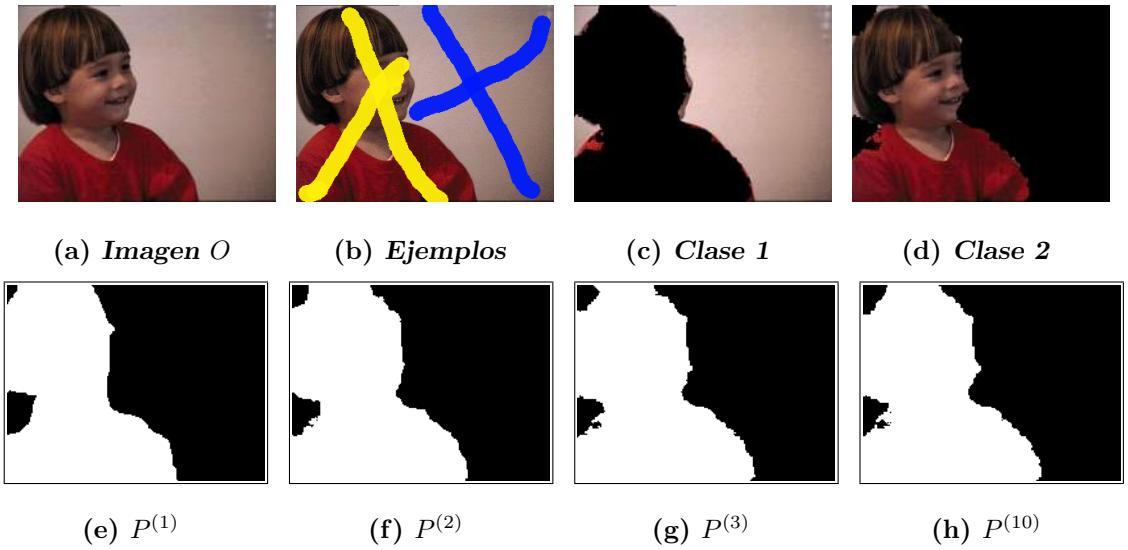
con el algoritmo SI-VA (Figura 5.5(c)). Nótese que, a pesar de que se dieron ejemplos de pixeles mal clasificados (extremos de algunas líneas) el algoritmo logra una segmentación adecuada.



(a) ***Clase 1*** **(b)** ***Clase 2*** **(c)** ***Clase 3*** **(d)** ***Clase 4***

Figura 5.6: Ejemplo de segmentación de imágenes con 4 clases con presencia de ruido.

En la Figura 5.7(a) se muestra una imagen de un niño y un fondo plano, sobre la cual se han marcado ejemplos de dos clases: el fondo (clase 1) y pixeles que forman parte de la imagen del niño (clase 2), tal como se muestra en la Figura 5.7(b). Las Figuras 5.7(e)-5.7(h) muestran la evolución del mapa de decisión en las iteraciones 1, 2, 3 y 10, respectivamente. Las Figuras 5.7(c) y 5.7(d) muestran la segmentación de cada clase obtenida con el mapa de decisión arrojado por el algoritmo SI-VA. Para este ejemplo la segmentación es adecuada a pesar de la similitud que existe entre el color del fondo y el color de la cara del niño.



(a) ***Imagen*** O **(b)** ***Ejemplos*** **(c)** ***Clase 1*** **(d)** ***Clase 2***

(e) $P^{(1)}$ **(f)** $P^{(2)}$ **(g)** $P^{(3)}$ **(h)** $P^{(10)}$

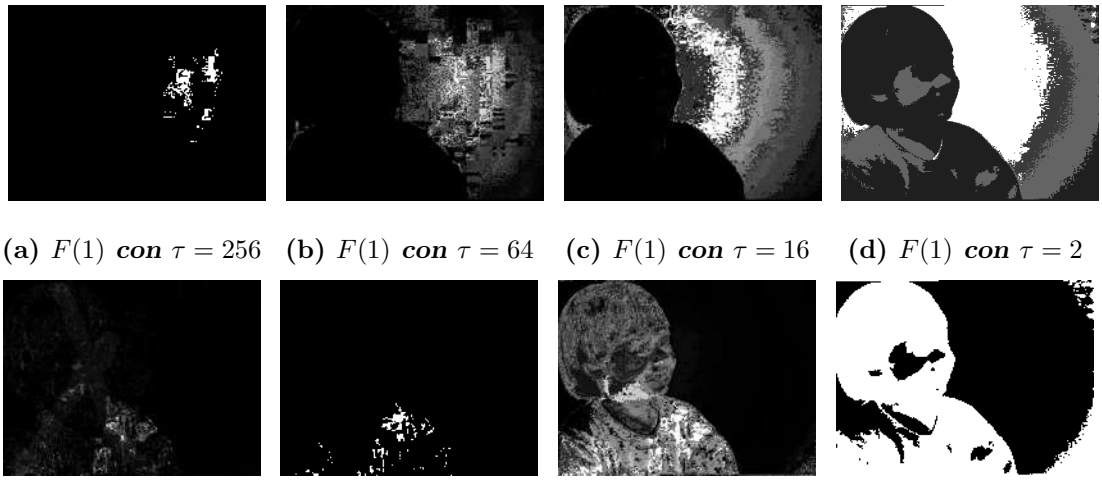
Figura 5.7: Ejemplo de segmentación de 2 clases en una fotografía de un niño.

5.4. Reducción de la resolución del histograma

Según lo mostrado en los ejemplos anteriores, el algoritmo SI-VA logra una muy buena segmentación. Sin embargo, dado que generalmente la cantidad de pixeles de ejemplo de cada clase es relativamente pequeña y el tamaño de $\mathcal{H}(k)$ es $256\times256\times256$, un alto porcentaje de coordenadas de $\mathcal{H}(k)$ tendrán un valor 0. Por ejemplo, para una muestra de 256×256, a lo más $\frac{1}{256}$ del arreglo $H(k)$ tendrá valores diferentes de 0. Lo anterior provoca un desperdicio de memoria, además de que una gran cantidad de colores $[R, G, B]$ tendrán probabilidad a priori 0, debido a que no aparecen en el conjunto de entrenamiento. Una alternativa para solucionar este problema es reducir la resolución del histograma, partiendo el rango de valores de cada canal en τ intervalos de tamaño $\frac{256}{\tau}$ y construyendo un histograma de menor tamaño $(\tau \times \tau \times \tau)$.

En la Figura 5.8 se muestra la medida de verosimilitud para el ejemplo de la Figura 5.7(a), con los ejemplos de cada clase marcados en 5.7(b), utilizando (5.7) y variando la resolución del histograma $\mathcal{H}(k)$. En esta figura, el primer renglón muestra imágenes del valor de verosimilitud de cada pixel para la clase 1 (fondo) y, el segundo renglón muestra imágenes de la medida de verosimilitud de cada pixel con la clase

2 (niño). Los valores altos se representan con blanco y, valores cercanos a 0, con negro. Se puede observar en las imágenes de la Figura 5.8 que, cuando τ disminuye, la función de verosimilitud arroja valores más altos para los pixeles que sí pertenecen a la clase y valores cercanos a 0 para los que no. Sin embargo, no es recomendable utilizar valores $\tau < 16$ debido a que al reducir demasiado la resolución del histograma, se corre el riesgo de no distinguir entre las clases y perder la definición en los bordes.



(a) $F(1)$ ***con*** $\tau = 256$ **(b)** $F(1)$ ***con*** $\tau = 64$ **(c)** $F(1)$ ***con*** $\tau = 16$ **(d)** $F(1)$ ***con*** $\tau = 2$

(e) $F(2)$ ***con*** $\tau = 256$ **(f)** $F(2)$ ***con*** $\tau = 64$ **(g)** $F(2)$ ***con*** $\tau = 16$ **(h)** $F(2)$ ***con*** $\tau = 2$

Figura 5.8: Función de verosimilitud disminuyendo la resolución de $\mathcal{H}(k)$.

Otra forma de evitar el problema del gran porcentaje de valores que son 0 en $\mathcal{H}_{r,g,b}$ es suponiendo independencia entre las probabilidades en cada uno de los canales de la imagen; es decir, $p(x_r = r, x_g = g, x_b = b|C_k) = p(x_r = r|C_k)p(x_g = g|C_k)p(x_b = b|C_k)$. De este modo, habrá un histograma para cada canal de la imagen $\mathcal{H}^r(k)$, $\mathcal{H}^g(k)$ y $\mathcal{H}^b(k)$ y cada uno de ellos es de tamaño 256. Adicionalmente, se puede aplicar un suavizado a los histogramas, con el fin de garantizar que colores cercanos sean clasificados como parte de la misma clase. Dicho suavizado se logra aplicando un promedio móvil, donde cada valor en la posición q del histograma es recalculado como la media de los valores desde $q-\alpha$ hasta $q+\alpha$. Entonces, el histograma $\mathcal{H}^Q(k)$ del canal Q para la clase k está dado por 5.8. Por lo tanto, es posible medir la verosimilitud de

un pixel con cada clase utilizando (5.9).

$$\mathcal{H}_q^Q(k) = \frac{1}{2\alpha+1} \sum_{j=q-\alpha}^{q+\alpha} \frac{1}{N_k} \sum_{x \in C_k} U(x,j) \tag{5.8}$$

$$U(x,q) = \begin{cases} 1 & \text{Si } x_q = q \\ 0 & \text{En otro caso} \end{cases}$$

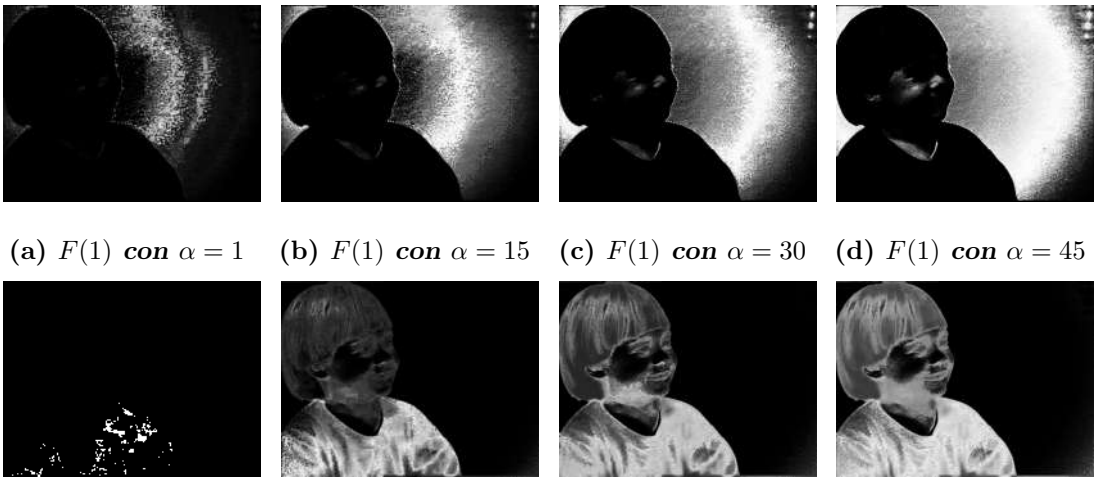
$$0 \leq q \leq 255 \quad \forall k \in [1,2,\ldots,N]$$

$$F_{l,m}(k) = \frac{N_k}{\mathcal{T}} \mathcal{H}_{x_r}^r(k) \mathcal{H}_{x_g}^g(k) \mathcal{H}_{x_b}^b(k) \tag{5.9}$$

En la Figura 5.9 se muestran imágenes del resultado de aplicar la función de verosimilitud dada por (5.9) para la imagen de la Figura 5.7(a), con los ejemplos de cada clase marcados en 5.7(b), al variar el valor de α. En dicha figura el primer renglón muestra la verosimilitud de cada pixel de la imagen con la clase 1 (fondo) y, el segundo renglón, muestra la verosimilitud de cada pixel con la clase 2 (niño). De acuerdo con las imágenes de la Figura 5.9, con histogramas independientes por cada canal también se logra obtener una medida de verosimilitud adecuada para cada clase, de forma similar a lo obtenido con el histograma completo.

En la Figura 5.10 se muestra el mismo ejemplo que en la Figura 5.7, con la diferencia de que en este caso se utiliza la función de verosimilitud dada por (5.9). En las Figuras 5.10(e)-5.10(h) se muestra la evolución del mapa de decisión y, en las Figuras 5.10(i)-5.10(l), se reporta la evolución de los bordes. En las Figuras 5.10(c) y 5.10(d) se muestran los pixeles de la clase 1 y 2 respectivamente. Nótese que, la suposición de independencia entre los canales no afecta el desempeño del algoritmo.

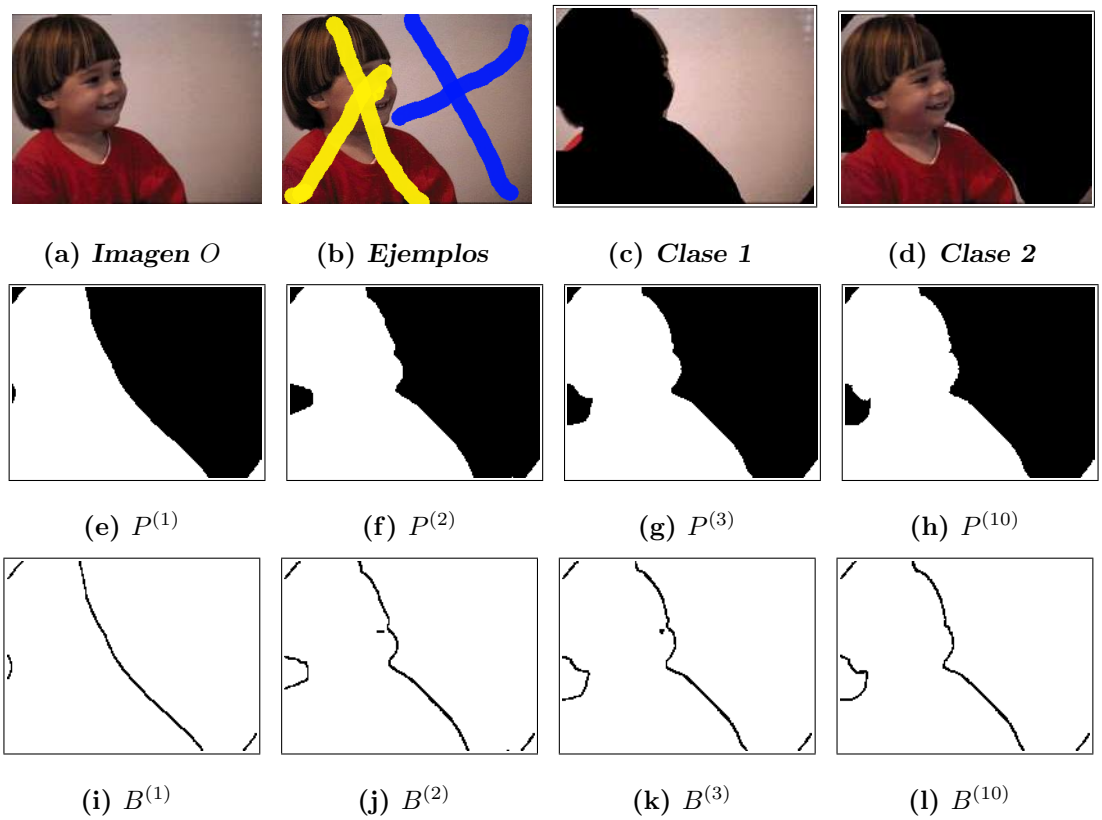
En la Figura 5.11(a) se muestra una fotografía tomada con la cámara a bordo de un robot de servicio como el reportado en [Garnica et˜al., 2016], el cual tiene como objetivo encontrar todas las pelotas que hay en el tablero y colocar cada una de ellas en el contenedor de su color. El primer paso para lograr el objetivo del robot es encontrar las pelotas y los contenedores y, para ello. se debe realizar la



(a) $F(1)$ ***con*** $\alpha = 1$ **(b)** $F(1)$ ***con*** $\alpha = 15$ **(c)** $F(1)$ ***con*** $\alpha = 30$ **(d)** $F(1)$ ***con*** $\alpha = 45$

(e) $F(2)$ ***con*** $\alpha = 1$ **(f)** $F(2)$ ***con*** $\alpha = 15$ **(g)** $F(2)$ ***con*** $\alpha = 30$ **(h)** $F(2)$ ***con*** $\alpha = 45$

Figura 5.9: Resultado de la función de aptitud usando (5.9) y variando α.



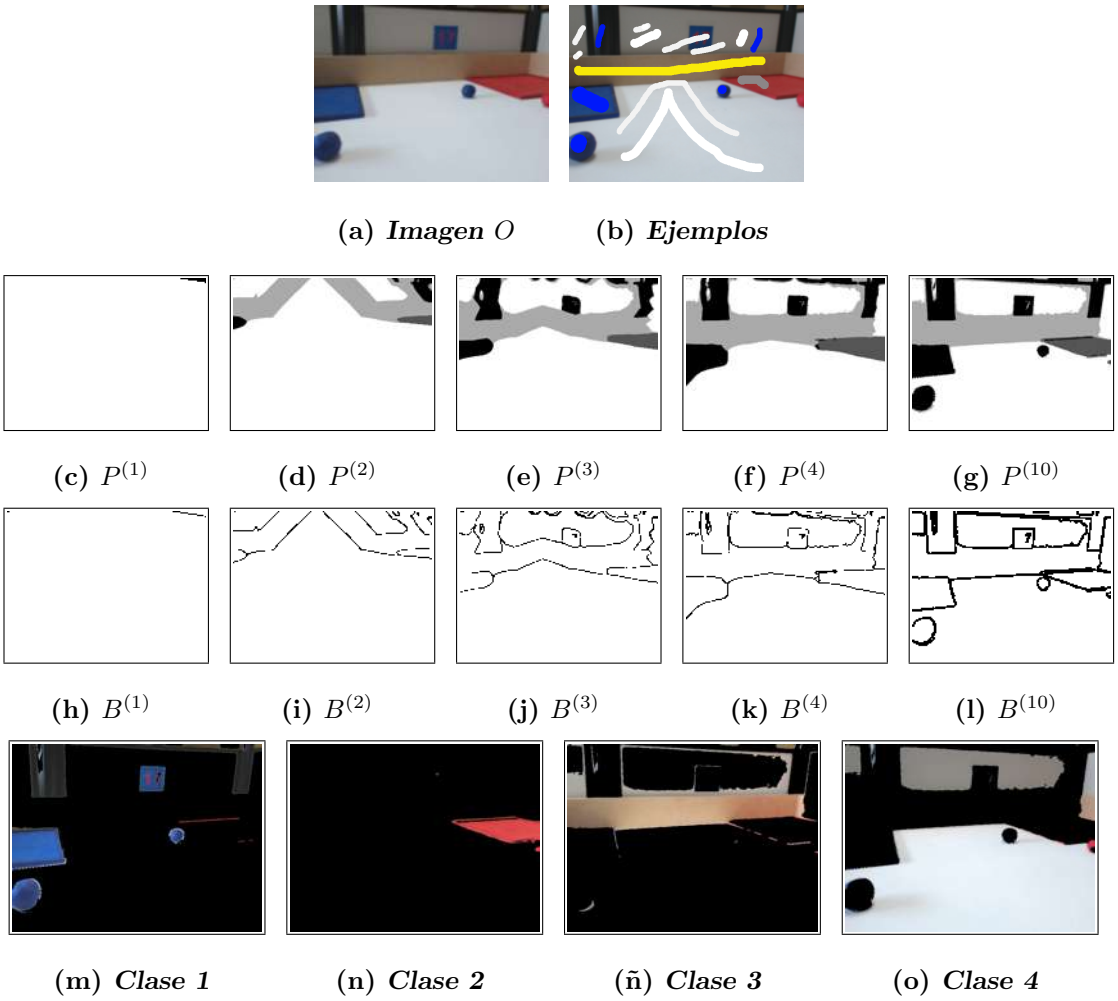
(a) ***Imagen*** O **(b)** ***Ejemplos*** **(c)** ***Clase 1*** **(d)** ***Clase 2***

(e) $P^{(1)}$ **(f)** $P^{(2)}$ **(g)** $P^{(3)}$ **(h)** $P^{(10)}$

(i) $B^{(1)}$ **(j)** $B^{(2)}$ **(k)** $B^{(3)}$ **(l)** $B^{(10)}$

Figura 5.10: Ejemplo de segmentación usando el algoritmo SI-VA y (5.9).

segmentación. En la Figura 5.11 se muestra el resultado de aplicar el algoritmo SI-VA con la imagen O (Fig. 5.11(a)) y los ejemplos mostrados en la Fig.5.11(b). Las Figuras 5.11(c)-5.11(g) y 5.11(h)-5.11(l) muestran la evolución del mapa de decisión y los bordes, respectivamente. La segmentación de cada clase 1 a 4 se muestra en las Figuras 5.11(m)-5.11(o), respectivamente.



(a) ***Imagen*** O **(b)** ***Ejemplos*** **(c)** $P^{(1)}$ **(d)** $P^{(2)}$ **(e)** $P^{(3)}$ **(f)** $P^{(4)}$ **(g)** $P^{(10)}$ **(h)** $B^{(1)}$ **(i)** $B^{(2)}$ **(j)** $B^{(3)}$ **(k)** $B^{(4)}$ **(l)** $B^{(10)}$ **(m)** ***Clase 1*** **(n)** ***Clase 2*** **(ñ)** ***Clase 3*** **(o)** ***Clase 4***

Figura 5.11: Ejemplo de segmentación de imágenes con 4 clases.

En la Figura 5.12 se muestra un esquema del proceso para segmentar una imagen, en base a color, utilizando el algoritmo SI-VA (Alg. 8). La imagen original O y un conjunto de ejemplos de las clases $\mathcal{C}$ se muestran en la primera columna. Con ellos se calculan las N funciones de verosimilitud, que se muestran en la segunda columna, y

que son utilizadas para el cálculo del mapa de decisión P, que se puede observar en la columna 3. Finalmente, en base a éste mapa se segmenta la imagen O para obtener N imágenes que se muestran en la última columna de la figura, donde cada imagen k contiene los pixeles de la k-ésima clase.

5.5. Conclusiones del capítulo

Se ha presentado el Algoritmo 8 (SI-VA) que surge como una modificación del Algoritmo 6 (FI-VA), para ser aplicado a la segmentación de imágenes por color. Se han presentado ejemplos de que el algoritmo propuesto permite la segmentación de imágenes con alta eficiencia. Los cambios realizados con respecto al algoritmo FI-VA son básicamente los parámetros que recibe, el cambio de la función de nitidez por una de verosimilitud y cambiar el resultado retornado.

El objetivo de presentar esta aplicación del algoritmo propuesto en este capítulo es mostrar que así como FI-VA puede ser aplicado a la fusión de cualquier tipo de imágenes al cambiar la función utilizada para medir la nitidez, también puede ser aplicado a la segmentación de imágenes en base a color, nitidez, iluminación etc. Siempre y cuando se seleccione una función adecuada para el problema que se desea resolver.

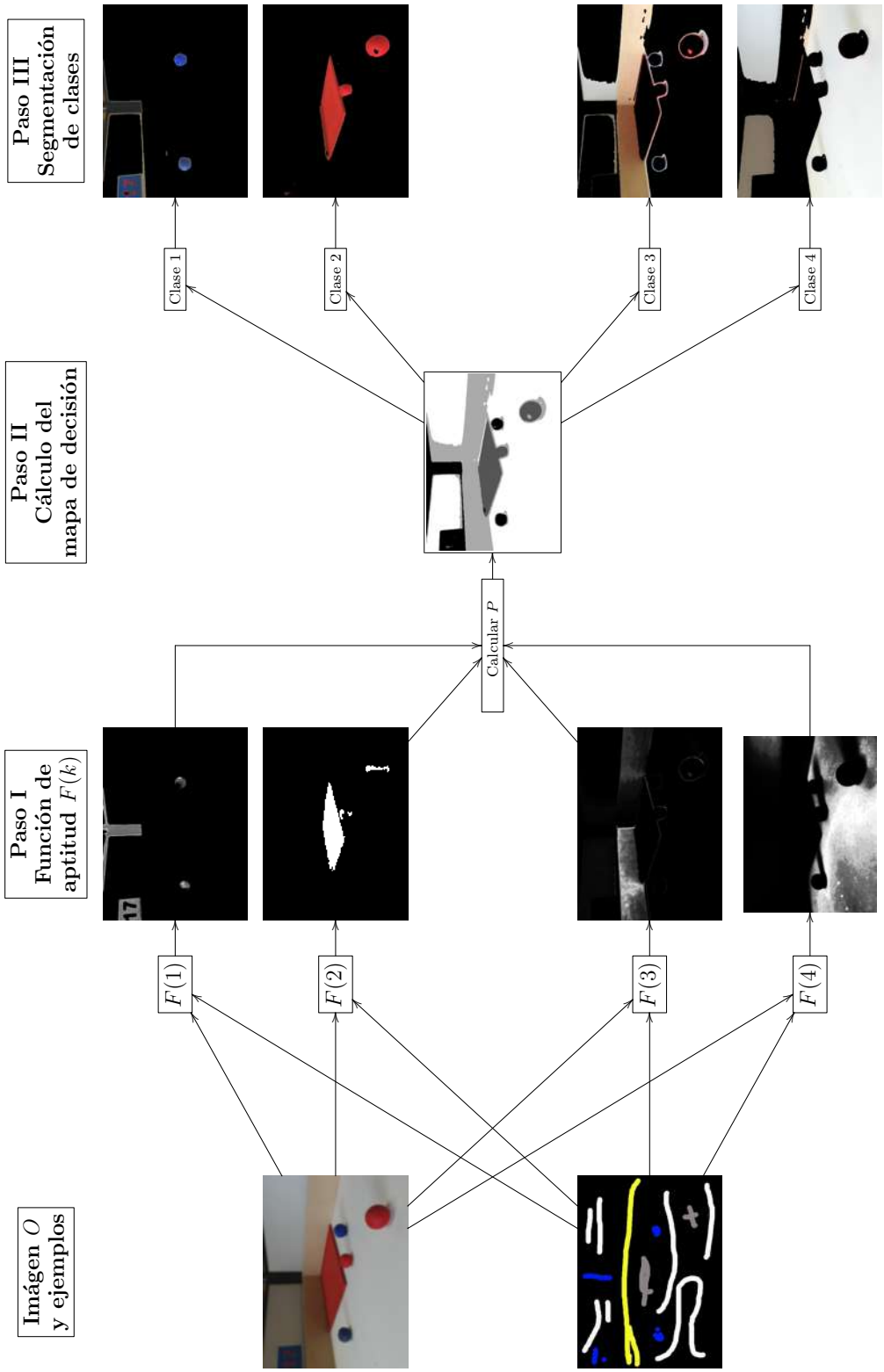


Figura 5.12: Proceso de segmentación por color usando el algoritmo SI-VA.

Capítulo 6

Resultados

En este capítulo se presentan los resultados obtenidos al aplicar el algoritmo propuesto para la fusión de imágenes multi-foco, fusión de imágenes con iluminación no uniforme, binarización de imágenes con iluminación no uniforme y segmentación de imágenes por color. Adicionalmente se presenta información sobre el tiempo de ejecución consumido por el algoritmo. Los experimentos mostrados en este capítulo, se realizaron en una computadora con procesador Intel Core i7-7700HQ de 64 bits con 4 núcleos, 8 hilos a una frecuencia 2.8 GHz y con 8Gb de memoria RAM.

6.1. Fusión de imágenes multi-foco

Se ha aplicado el Algoritmo 6 (FI-VA) a diferentes conjuntos de imágenes multi-foco. Los parámetros utilizados para los experimentos (salvo en los que se indique lo contrario), fueron $w_{max} = 0.5\sqrt{n_r * n_c * 0.05 - 1}$ y $T = 5$.

Para mostrar las ventajas que ofrece la paralelización en el Algoritmo 6, realizamos un experimento variando el número de hilos que realizan la fusión multi-foco considerando un par de imágenes de tamaño 520×520 pixeles. En la Figura 6.1 se muestra una gráfica con el tiempo de ejecución en milisegundos (ms). contra el número de hilos de procesador utilizados. Nótese que el tiempo de ejecución disminuye de

$331ms$ a $143ms$, para este experimento.

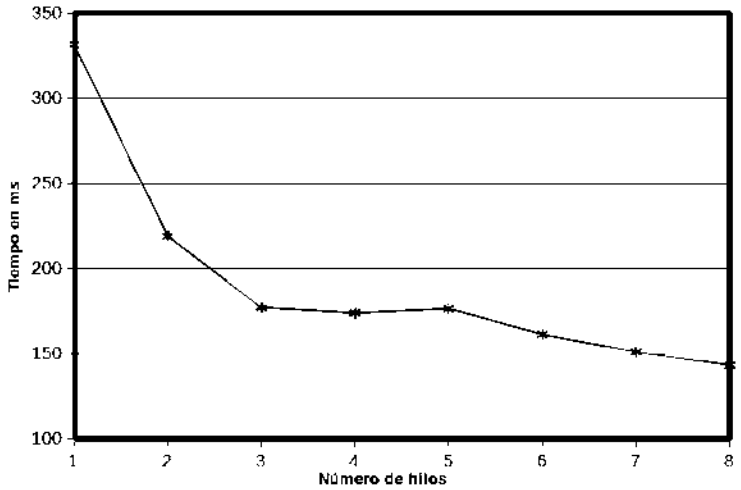


Figura 6.1: Tiempo de ejecución variando la cantidad de hilos para el algoritmo FI-VA.

Dado que se demostró que el algoritmo FI-VA tiene complejidad asintótica lineal en tiempo de ejecución con respecto al número de pixeles de la imagen, se realizó la fusión de distintos pares de imágenes multi-foco, con diferentes tamaños de imagen, variando la cantidad de hilos que participan en el proceso, para mostrar la condición de linealidad. En la Figura 6.2 se muestra la gráfica del tiempo de ejecución contra el número de pixeles de las imágenes. Se puede observar un conjunto de líneas que muestra el comportamiento del algoritmo FI-VA con varios hilos y se puede notar la reducción en tiempo que ofrece su uso. Además, también es fácil notar que el tiempo crece de forma lineal al incrementarse el tamaño de la imagen.

En la Tabla 6.1 se muestran los tiempos de ejecución consumidos por el algoritmo FI-VA para fusionar conjuntos de imágenes de 375×500 pixeles. En esta tabla podemos ver, en la primera columna, el número de imágenes N de cada conjunto, el resto de las columnas muestran el tiempo requerido para la fusión usando desde 1 hasta 8 hilos y se puede resaltar que, para pares de imágenes, se logra la fusión en menos de una décima de segundo; y, para septetas de imágenes, la fusión consume menos de dos décimas de segundo.

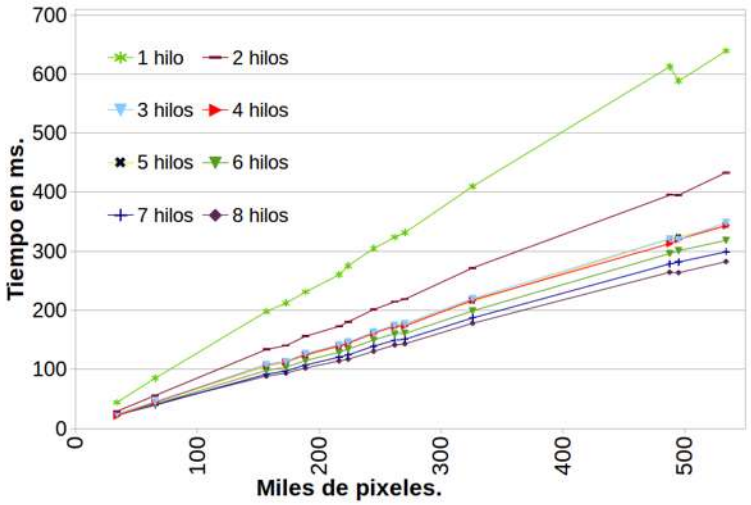


Figura 6.2: Tiempo de ejecución consumido por el algoritmo FI-VA variando la cantidad de hilos.

Tabla 6.1: Comparación en tiempo para conjuntos de imágenes de 375×500 pixeles

N	**Tiempo de ejecución** ($ms.$) **FI-VA con diferente número de hilos.**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
2	219	145	120	121	120	110	103	**99**
3	255	167	137	140	139	127	121	**119**
4	264	183	155	155	154	142	134	**130**
5	320	216	181	179	183	169	162	**157**
6	326	225	191	190	190	177	170	**166**
7	343	245	209	209	210	196	190	**185**

En la Tabla 6.2 se muestra el PA para 75 conjuntos de imágenes multi-foco de diferentes tamaños y con diferente cantidad N de imágenes. Los conjuntos de imágenes utilizados son 55 pares, 10 triadas, 4 cuartetos, 3 quintetos, 1 sexteto y 2 septetos. La tabla contiene el promedio del PA para los conjuntos de imágenes. La columna 1 muestra la cantidad de imágenes en el conjunto. La columna 2 muestra la cantidad de conjuntos. La columna 3 muestra el promedio del PA usando parámetros fijos para

todos los conjuntos de imágenes. Finalmente, la columna 4 muestra el promedio de PA obtenido aplicando los mejores parámetros (w_{max} y T) para cada caso. De los valores de esta tabla podemos ver que en todos los casos se tiene un porcentaje de acierto muy bueno, obteniendo un promedio para todos los casos de 96 % de acierto.

Tabla 6.2: Precisión del algoritmo FI-VA

		Pixel Accuracy FI-VA	
N	Conjuntos	Parámetros fijos.	Parámetros óptimos.
2	55	.961	.971
3	10	.909	.922
4	4	.978	.986
5	3	.813	.820
6	1	.932	.954
7	2	.957	.981

En [De y Chanda, 2013] y [Bai et˜al., 2015] se presenta el uso de quad-trees para realizar la fusión de imágenes, dicha técnica es similar a nuestra propuesta en el aspecto de que se usan bloques o ventanas de tamaño variable para el cálculo de la imagen fusionada. Sin embargo, nuestra propuesta tiene un consumo de tiempo mucho menor que estas propuestas, tal y como se puede observar en la Tabla 6.3, donde se presenta el tiempo de ejecución para cada propuesta. Se puede notar que, el algoritmo FI-VA tiene tiempos de ejecución muy por debajo de los tiempos reportados en estos trabajos, para un par de imágenes sintéticas de tamaño 512×512 pixeles, incluso, con un sólo hilo.

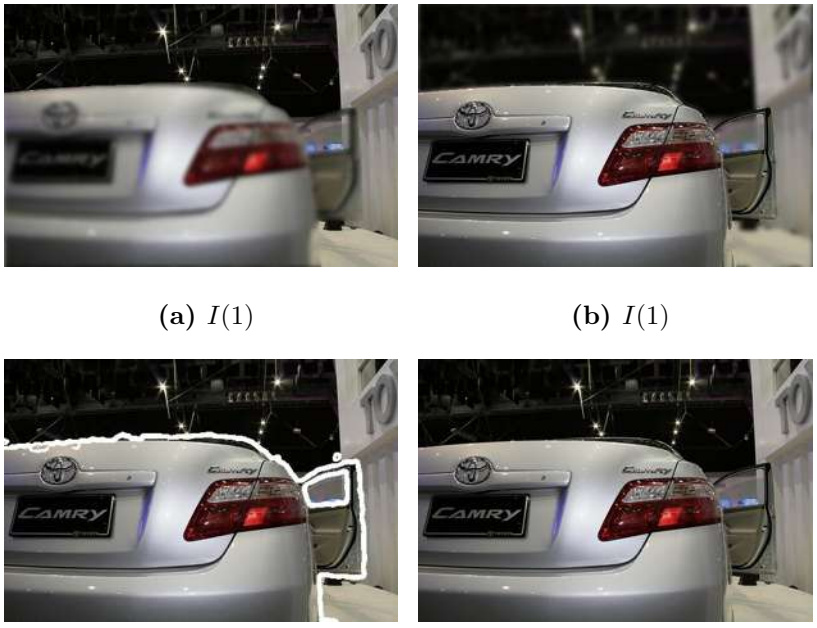
6.1.1. Pares de imágenes multi-foco sintéticas

Para evaluar del algoritmo FI-VA se creó un conjunto de 21 pares de imágenes sintéticas, de las cuales 4 fueron creadas usando mapas de decisión generados por

Tabla 6.3: Comparación de tiempo de ejecución en propuestas similares.

Tiempo (s.)			
[De y Chanda, 2013]	[Bai et˜al., 2015]	FI-VA 1 hilo	FI-VA 8 hilos
29	1.029	0.253	**.141**

nosotros y 17 fueron tomadas de la página http://host.robots.ox.ac.uk/pascal/VOC/, donde existen imágenes de referencia y mapas de segmentación que, en este caso, hemos usado para crear las imágenes sintéticas. En las Figuras 6.3 y 6.4 se muestran dos ejemplos de un par de de imágenes multi-foco. Se han marcado los bordes en la imagen fusionada con el fin de que el lector tenga una referencia de qué regiones se extrajeron de cada imagen. Note en la Figura 6.3 la capacidad del algoritmo para segmentar el vidrio de la ventana del auto.

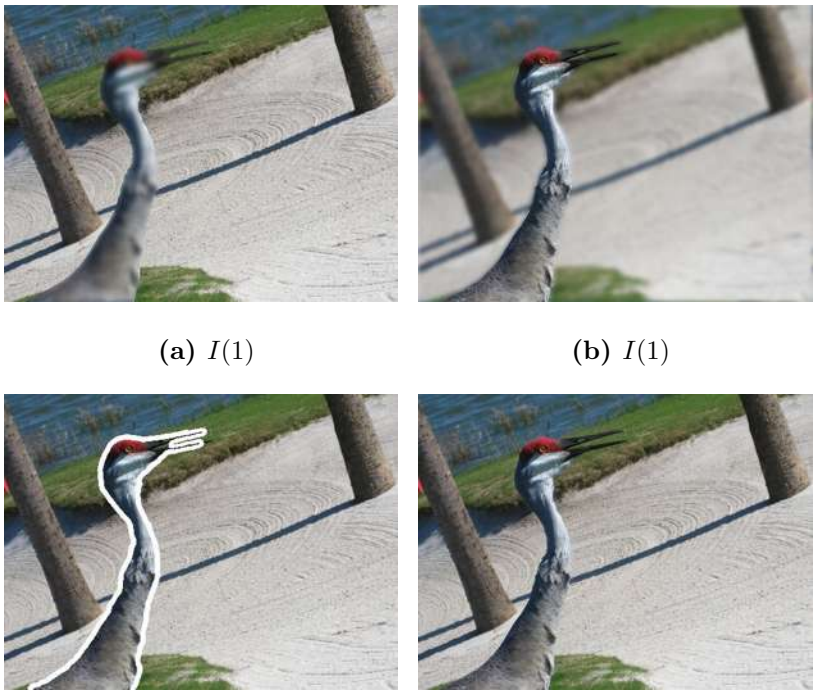


(a) $I(1)$ **(b)** $I(1)$

(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.3: Ejemplo de aplicación el algoritmo FI-VA a imágenes ulti-foco reales.

En la Figura 6.4 se puede notar la capacidad del algoritmo para segmentar regiones puntiagudas como el pico del ave mostrada en la figura.



(a) $I(1)$ **(b)** $I(1)$

(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.4: Ejemplo 2 de aplicación el algoritmo FI-VA a imágenes ulti-foco reales.

En la Tabla 6.4 se muestran los promedios de las métricas dadas por (3.8), (3.9), (3.10) y (3.11); para el mapa de decisión calculado mediante los algoritmos CLI-S y FI-VA para los 21 pares de imágenes sintéticas. Nótese que, en promedio, se tiene una precisión a nivel pixel (PA) del 99.19 % con el algoritmo FI-VA, contra 95.7 % del Algoritmo CLI-S.

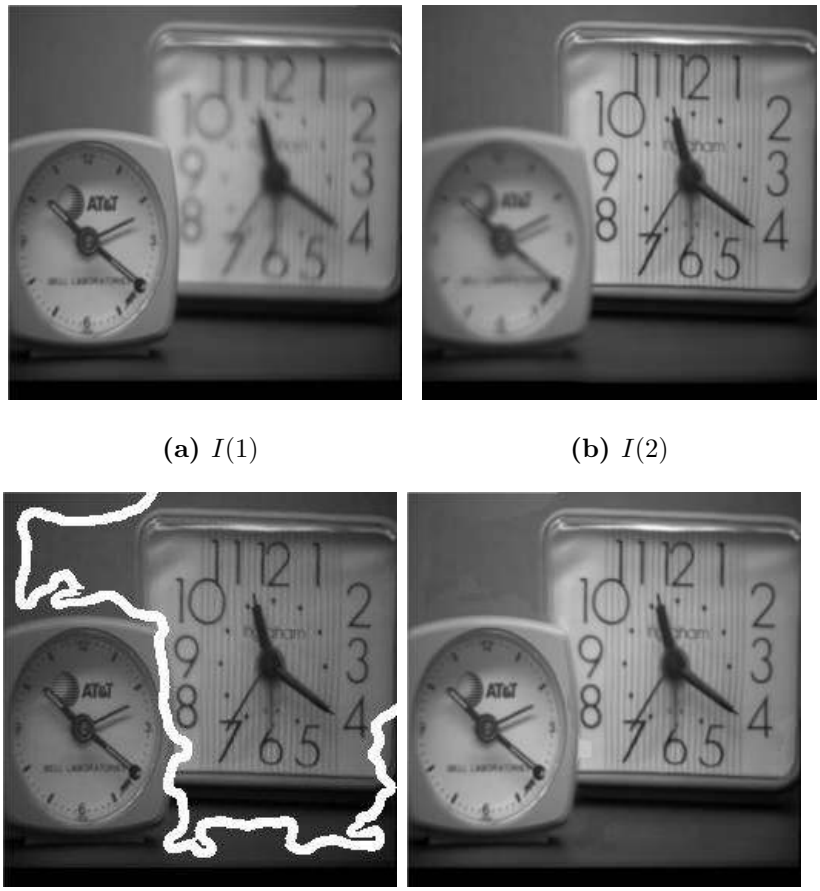
Tabla 6.4: Índices promedio calculados para 21 pares de imágenes multi-foco sintéticas utilizando los algoritmos CLI-S, CLI-VA y FI-VA

Algoritmo	**PA**	**MPA**	**MIU**	**FWIU**
CLI-S	95.70	93.91	88.73	91.94
CLI-VA	**99.19**	**98.90**	**97.65**	**98.39**
FI-VA	**99.19**	**98.90**	**97.65**	**98.39**

6.1.2. Pares de imágenes multi-foco reales

En las Figuras 6.5 a 6.8 se muestran los resultados de aplicar el algoritmo FI-VA a pares de imágenes multi-foco reales, obtenidas de Internet y de algunos artículos de fusión de imágenes, todas ellas de tamaño diferente. En estas figuras se muestran los pares de imágenes utilizadas, así como la imagen fusionada con y sin los bordes del mapa de decisión.

La Figura 6.5 muestra un par de imágenes que son muy utilizadas en la literatura para probar el desempeño de algoritmos de fusión de imágenes multi-foco. Es posible notar que la imagen fusionada contiene de forma nítida los dos relojes.



(a) $I(1)$ **(b)** $I(2)$

(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.5: Resultado de aplicar el algoritmo FI-VA con imágenes de relojes.

El segundo ejemplo, mostrado en la Figura 6.6 muestra una escena con una escultura y en el fondo un edificio, se ha mostrado este ejemplo con el fin de ilustrar el desempeño del algoritmo para sementar regiones pequeñas contenidas dentro de otras más grandes, como los espacios que hay entre los pies de los personajes de la escultura.

(a) $I(1)$

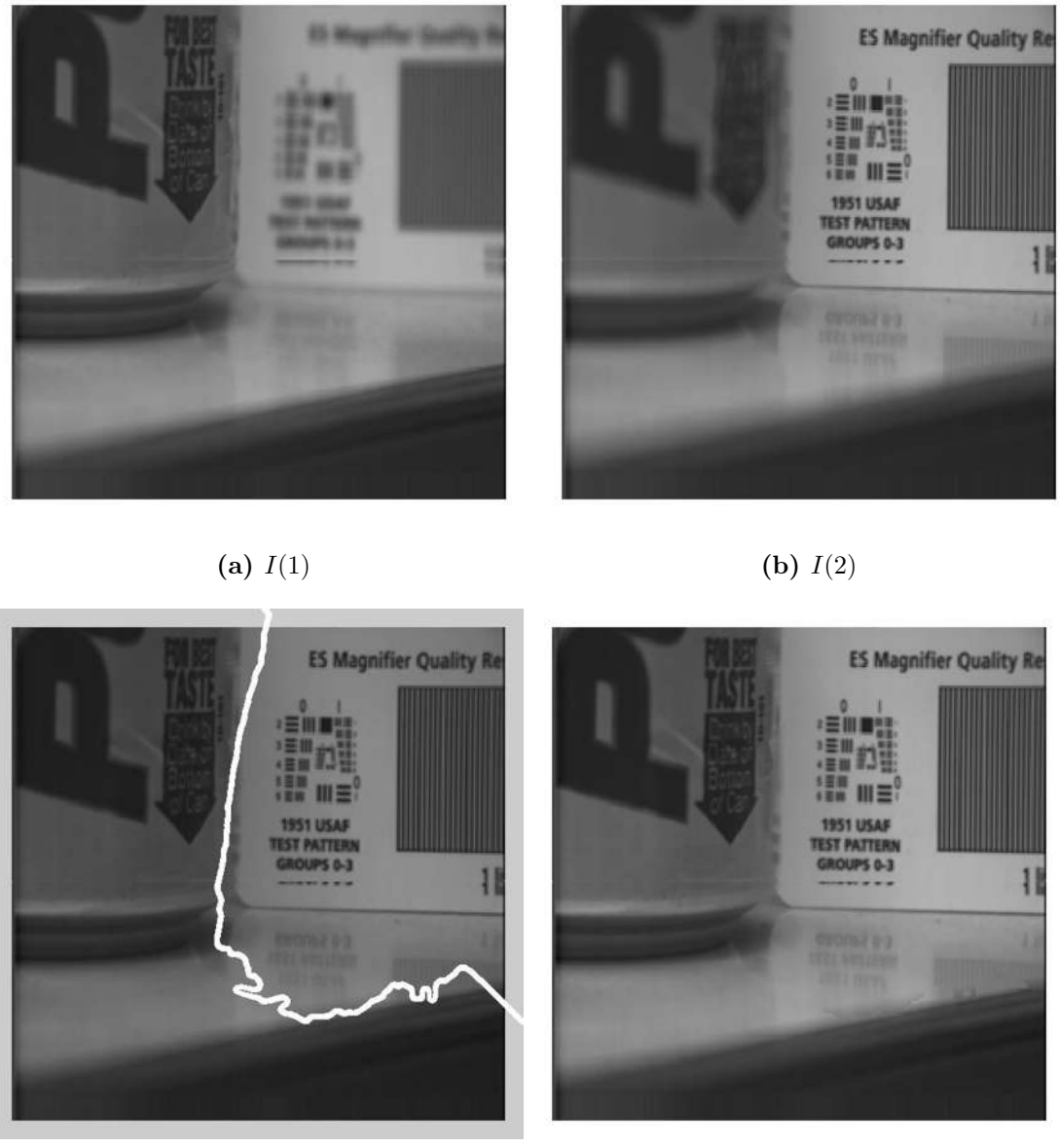
(b) $I(2)$

(c) ***Fusionada con bordes***

(d) ***Imagen fusionada***

Figura 6.6: Resultado de aplicar el algoritmo FI-VA con imágenes de una escultura y en el fondo un edificio.

En la Figura 6.7 se muestra un ejemplo muy utilizado en la literatura, sobre este ejemplo, es importante hacer notar la capacidad del algoritmo para segmentar de manera correcta las regiones nítidas de cada imagen, incluso, se ha segmentado de forma adecuada el reflejo en la superficie plana del código de barras que aparece en la caja.



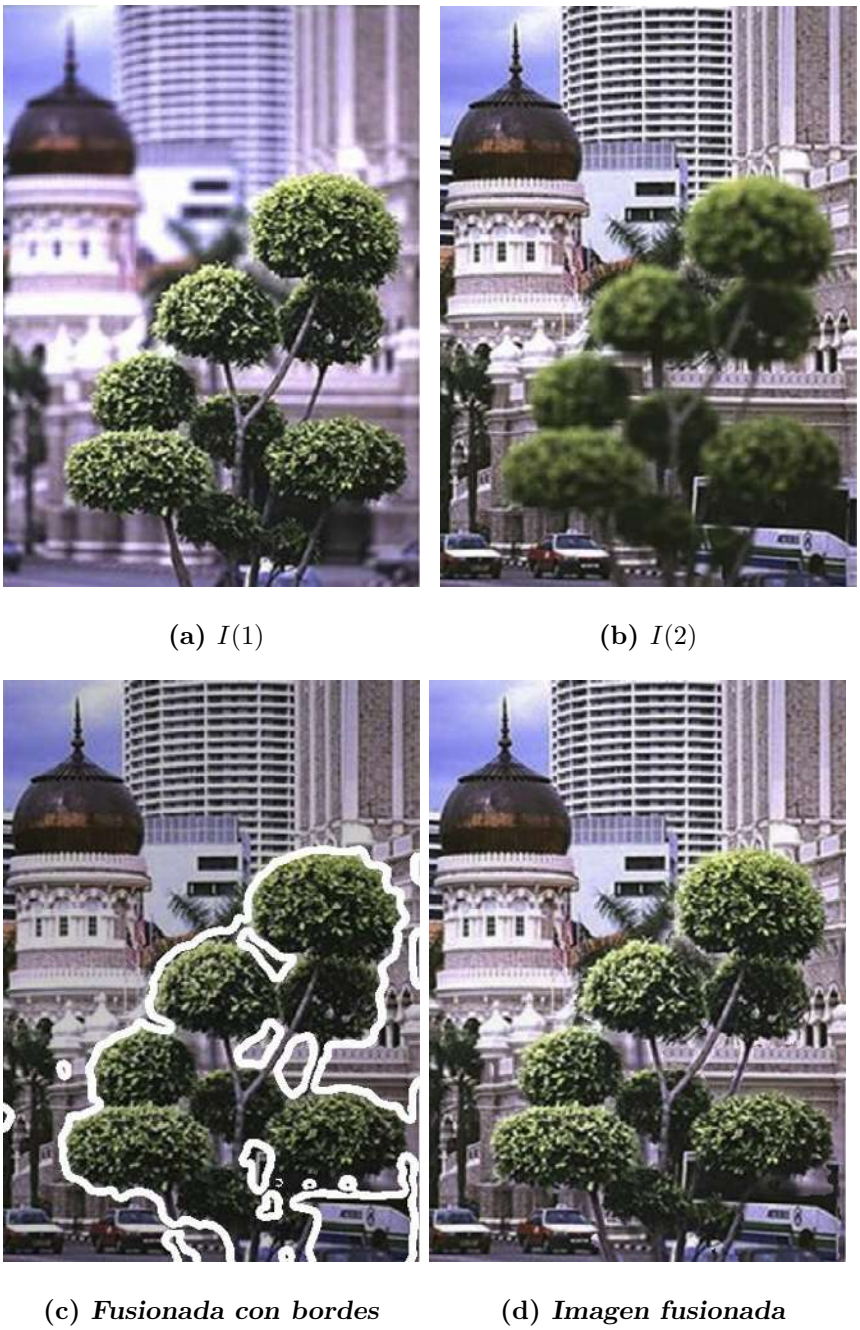
(a) $I(1)$ **(b)** $I(2)$

(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.7: Algoritmo FI-VA aplicado a imágenes de lata de refresco y caja con el código de barras.

El ejemplo mostrado en la Figura 6.8 muestra una escena con una planta u en el fondo algunos edificios, Sobre esta escena es importante hacer notar la capacidad del algoritmo FI-VA para recortar de forma adecuada la planta, a pesar de que existen

regiones muy delgadas, como el tallo de la planta.



(a) $I(1)$ **(b)** $I(2)$

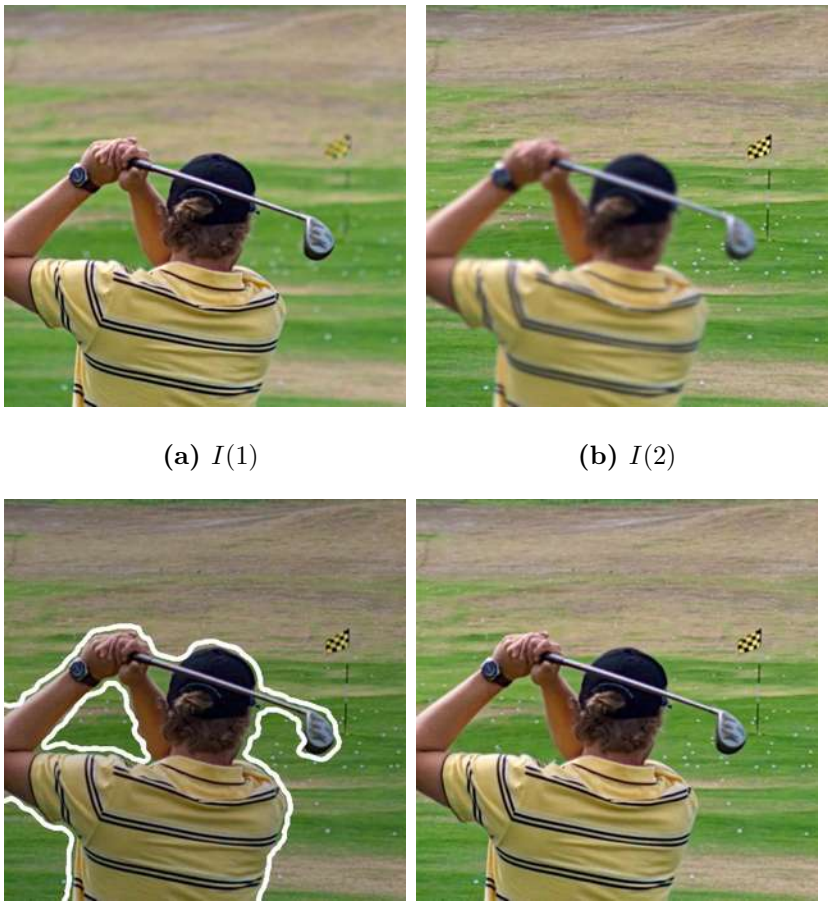
(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.8: Resultado de aplicar el algoritmo FI-VA con imágenes de planta y edificios en el fondo.

En las siguientes Figuras 6.9 a 6.11 se muestran los resultados obtenidos con el

algoritmo FI-VA para tres de los veinte pares de imágenes multi-foco presentados en [Nejati et˜al., 2015] y que fueron obtenidas con una cámara Lytro. En cada figura, se muestran el par de imágenes multi-foco y la imagen fusionada con y sin los bordes del mapa de decisión. Los 20 pares de imágenes son de tamaño 512×512 pixeles y los parámetros para el algoritmo FI-VA fueron $T = 20$, $w_{max} = 52$ y $N_{iter} = 10$.

En la Figura 6.9 se muestra un ejemplo donde aparece una persona y en el fondo un campo de golf, puede verse que el algoritmo logró segmentar de forma adecuada a la persona y el fondo, para lograr una imagen fusionada que es nítida en todas las regiones.

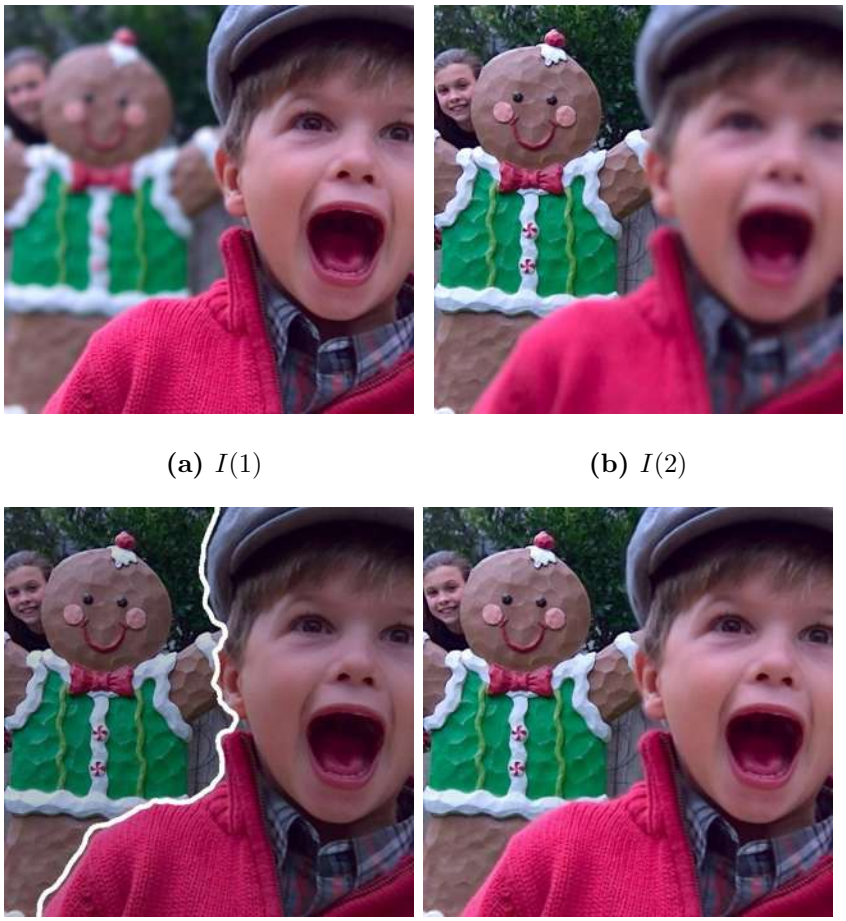


(a) $I(1)$ **(b)** $I(2)$

(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.9: Ejemplo de fusión de imágenes multi-foco reales para la escena del campo de golf.

En la Figura 6.10 se muestra un ejemplo de una escena donde aparece un niño, en este caso puede apreciarse la precisión del algoritmo al segmentar de forma adecuada la imagen del niño del fondo de la escena, logrando una imagen completamente nítida como resultado de la fusión.

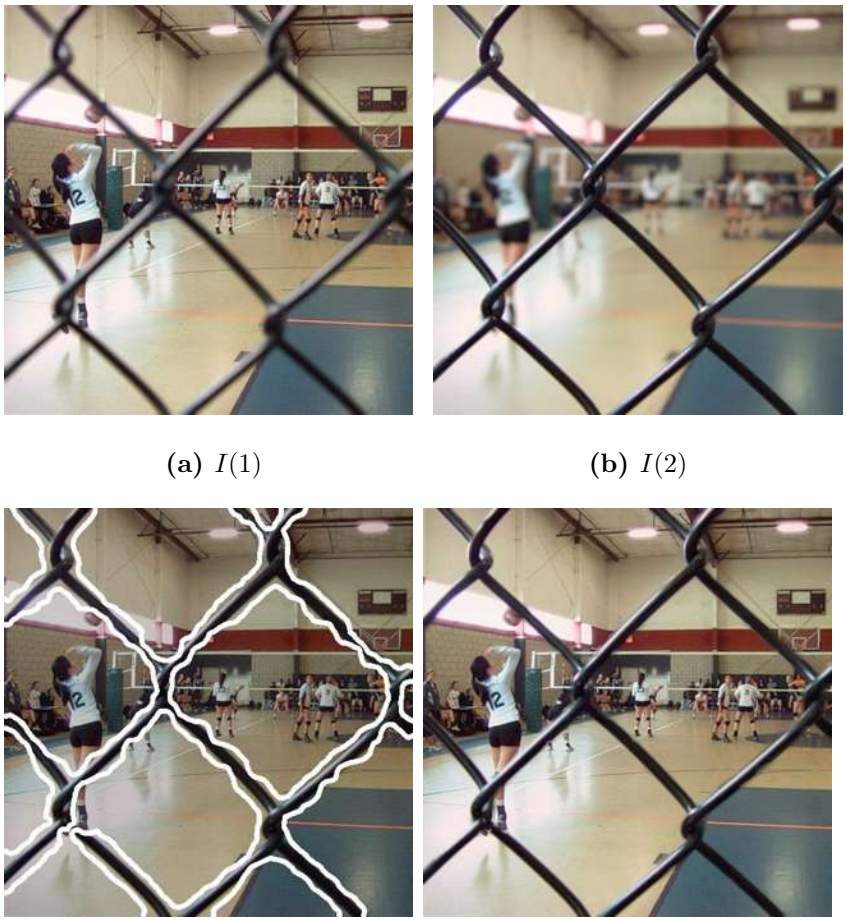


(a) $I(1)$ **(b)** $I(2)$

(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.10: Ejemplo de fusión de imágenes multi-foco reales para la escena del niño.

En la Figura 6.11 se muestra un ejemplo de una escena de una fotografía tomada con una maya de por medio, se ha seleccionado este ejemplo debido a la dificultad que implica el segmentar de forma adecuada la maya. A pesar de la dificultad que esto implica, el algoritmo FI-VA logra una imagen fusionada completamente nítida.



(a) $I(1)$ **(b)** $I(2)$

(c) ***Fusionada con bordes*** **(d)** ***Imagen fusionada***

Figura 6.11: Ejemplo de fusión de imágenes multi-foco reales para la escena con una maya de por medio.

En [Nejati et˜al., 2015] se presentan resultados para el conjunto de imágenes Lytro. La Tabla 6.5 presenta una comparación numérica entre los resultados de FI-VA y los resultados reportados por Nejati *et al.*, usando la Información Mutua Normalizada (NMI por sus siglas en inglés). De acuerdo con los valores presentados en dicha tabla, el algoritmo FI-VA es competitivo con respecto a la propuesta de Nejati *et al.*. Nejati *et al.* no presentan datos de tiempos de ejecución, sin embargo, el método propuesto presenta un proceso más simple por lo que creeemos que el tiempo consumido por el algoritmo de Nejati *et al.* es superior al consumido por FI-VA.

Tabla 6.5: Comparación de el algoritmo FI-VA con la propuesta de [Nejati et˜al., 2015]

Image	**Información mutua normalizada (NMI)**	
	Nejati *et al.* [Nejati et˜al., 2015]	FI-VA
Lab	**1.26**	1.16
Pepsi	1.25	**1.32**
Clock	**1.25**	1.20
Disk	1.15	**1.19**
Lytro dataset	1.19	**1.21**

La tabla 6.6 muestra la información mutua normalizada (NMI) y el tiempo de ejecución, para los 7 algoritmos más rápidos de los presentados en [Farid et˜al., 2019], así como para FI-VA aplicados al conjunto de imágenes multi-foco Lytro (con imágenes de tamaño 520×520 pixeles). Nótese que FI-VA comparte el mejor NMI con la propuesta de Farid *et al.* y con el algoritmo PCNN. Adicionalmente, FI-VA tiene el segundo mejor tiempo de ejecución. De acuerdo con lo anterior, podemos decir que FI-VA logra el mejor equilibrio entre el tiempo de ejecución y la precisión del resultado.

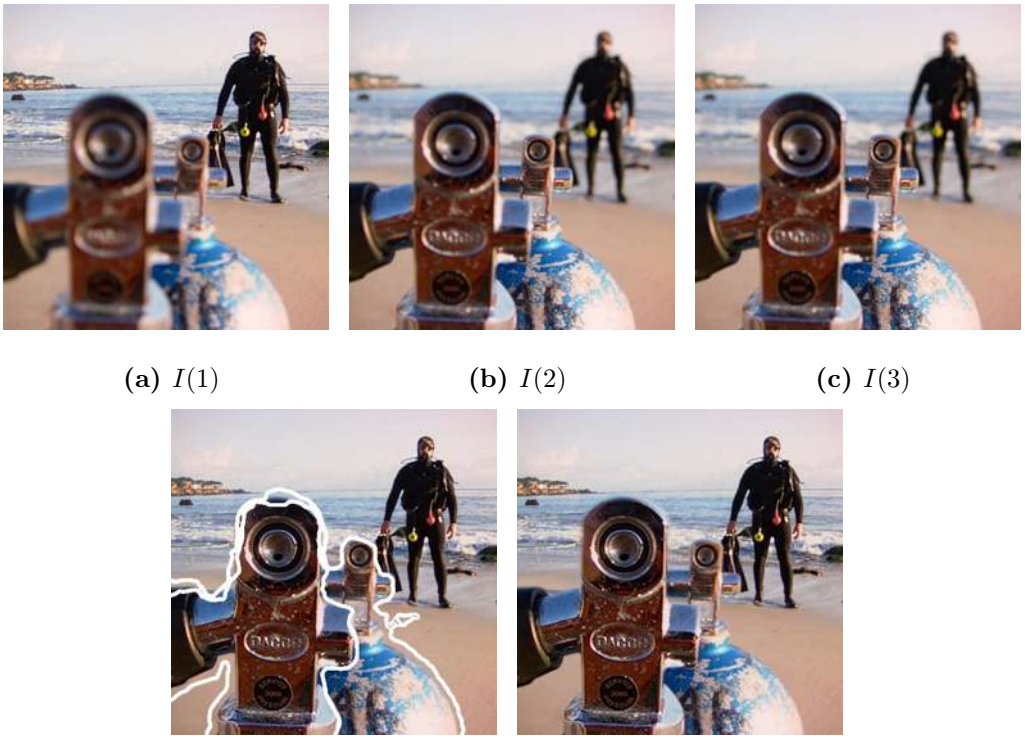
Tabla 6.6: Comparación entre el algoritmo FI-VA y algunas de las propuestas más prominentes del estado del arte.

	MWGF	*Farid et al.*	*DCTV*	*PCNN*	*IFGD*	*CLI-VV*	*DCTLP*	*GIF*	FI-VA	*PCA*
NMI	1.15	**1.21**	1.19	**1.21**	1.05	**1.21**	0.83	1.19	**1.21**	0.89
Tiempo (s)	4.07	1.87	1.71	1.45	0.84	0.35	0.30	0.25	0.14	**0.03**

6.1.3. Conjuntos con más de dos imágenes multi-foco

Los ejemplos que se muestran a continuación corresponden a 4 conjuntos con 3 imágenes multi-foco, obtenidas de http://mansournejati.ece.iut.ac.ir. Las escenas corresponden a un buzo en la playa, unos lobos marinos, un teclado de computadora y unas baterías en fila. Se han fijado los valores de los parámetros en $T = 30$ y $w_{max} = 104$ (20 % de las dimensiones de las imágenes). En las Figuras 6.12 a 6.14 se muestran ejemplos de fusión de conjuntos de 3 imágenes multi-foco. Cada figura muestra el conjunto de imágenes multi-foco, la imagen fusionada con los bordes del mapa de decisión y la imagen fuionada sin los bordes.

La Figura 6.12 se muestra una escena donde aparece en primer y segundo plano partes del un tanque de oxígeno y en el fondo aparece el buzo el mar. Después de la fusión se logra una imagen en la que toda la escena luce nítida.

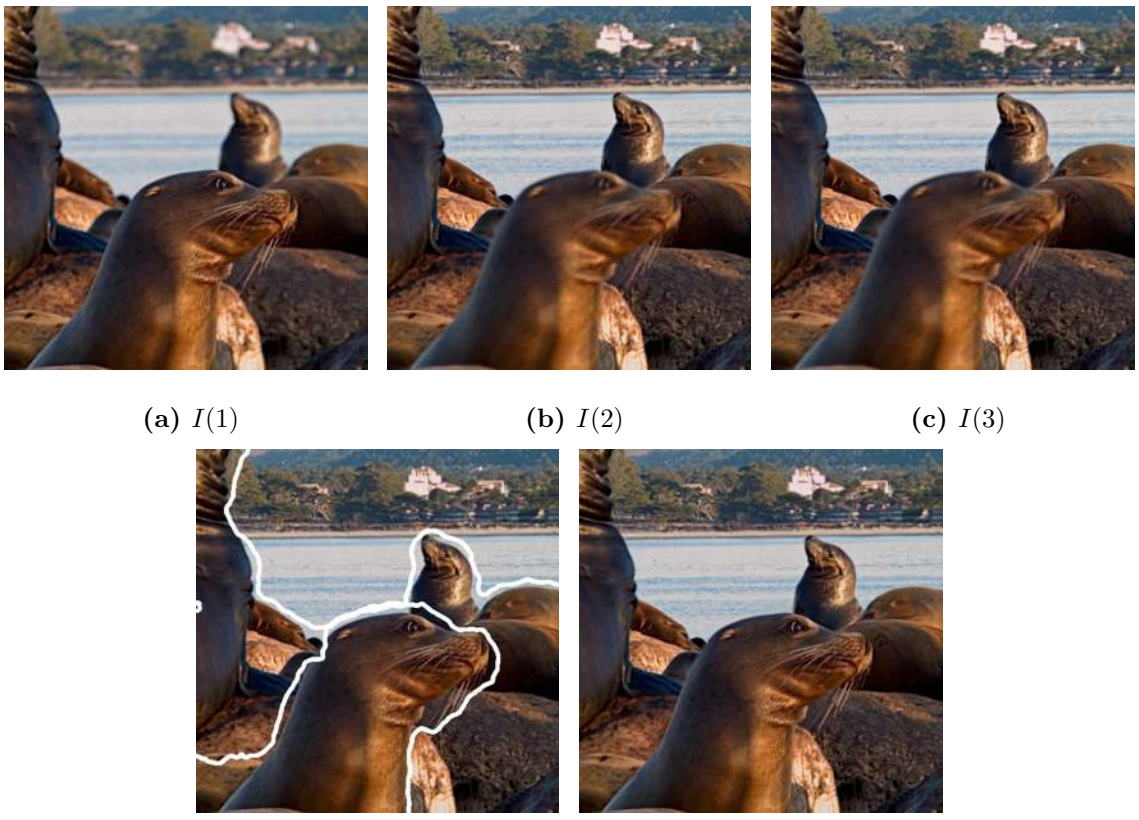


(a) $I(1)$ **(b)** $I(2)$ **(c)** $I(3)$

(d) ***Fusionada con bordes*** **(e)** ***Imagen fusionada***

Figura 6.12: Resultado de aplicar el algoritmo FI-VA conjunto de imágenes del buzo.

En la Figura 6.13 se muestra una escena donde aparece un grupo de lobos marinos y en el fondo el mar y vegetación. El cálculo del mapa de decisión, utilizando FI-VA permite segmentar de forma adecuada el lobo de enfrente que aparece nítido en la primera imagen, el resto de los lobos, que aparecen nítidos en la segunda imagen y el fondo que aparece nítido en la tercera imagen. Finalmente, la fusión nos permite ver una imagen completamente nítida.



(a) $I(1)$ **(b)** $I(2)$ **(c)** $I(3)$

(d) ***Fusionada con bordes*** **(e)** ***Imagen fusionada***

Figura 6.13: Resultado de aplicar el algoritmo FI-VA conjunto de imágenes de lobos marinos.

La Figura 6.14 muestra tres fotografías de un teclado de computadora, tomadas cada una de ellas, cambiando el nivel de enfoque, cada una de las tres imágenes del conjunto multi-foco muestra algunas regiones nítidas y otras que no lo están. Es sencillo notar en este ejemplo que la imagen fusionada es más nítida que cualquiera

de las tres imágenes del conjunto, permitiendo con ello, observar todos los detalles de la escena.

(a) $I(1)$ **(b)** $I(2)$ **(c)** $I(3)$

(d) ***Fusionada con bordes*** **(e)** ***Imagen fusionada***

Figura 6.14: Resultado de aplicar el algoritmo FI-VA conjunto de imágenes del teclado.

Puede observarse en las Figuras 6.12, 6.13 y 6.14 que los bordes de los mapas de decisión obtenidos se ajustan a los bordes reales de las regiones nítidas de cada una de las imágenes y que además se logra la coherencia espacial en las zonas sin bordes. En la Tabla 6.7 se muestran 4 métricas de la calidad del mapa de decisión calculado con FI-VA con respecto al mapa de decisión de referencia. El mapa de decisión de referencia fue establecido de forma manual (utilizando el programa GIMP). El promedio de PA para los tres conjuntos de imágenes mostradas en las figuras anteriores fue .9319 .

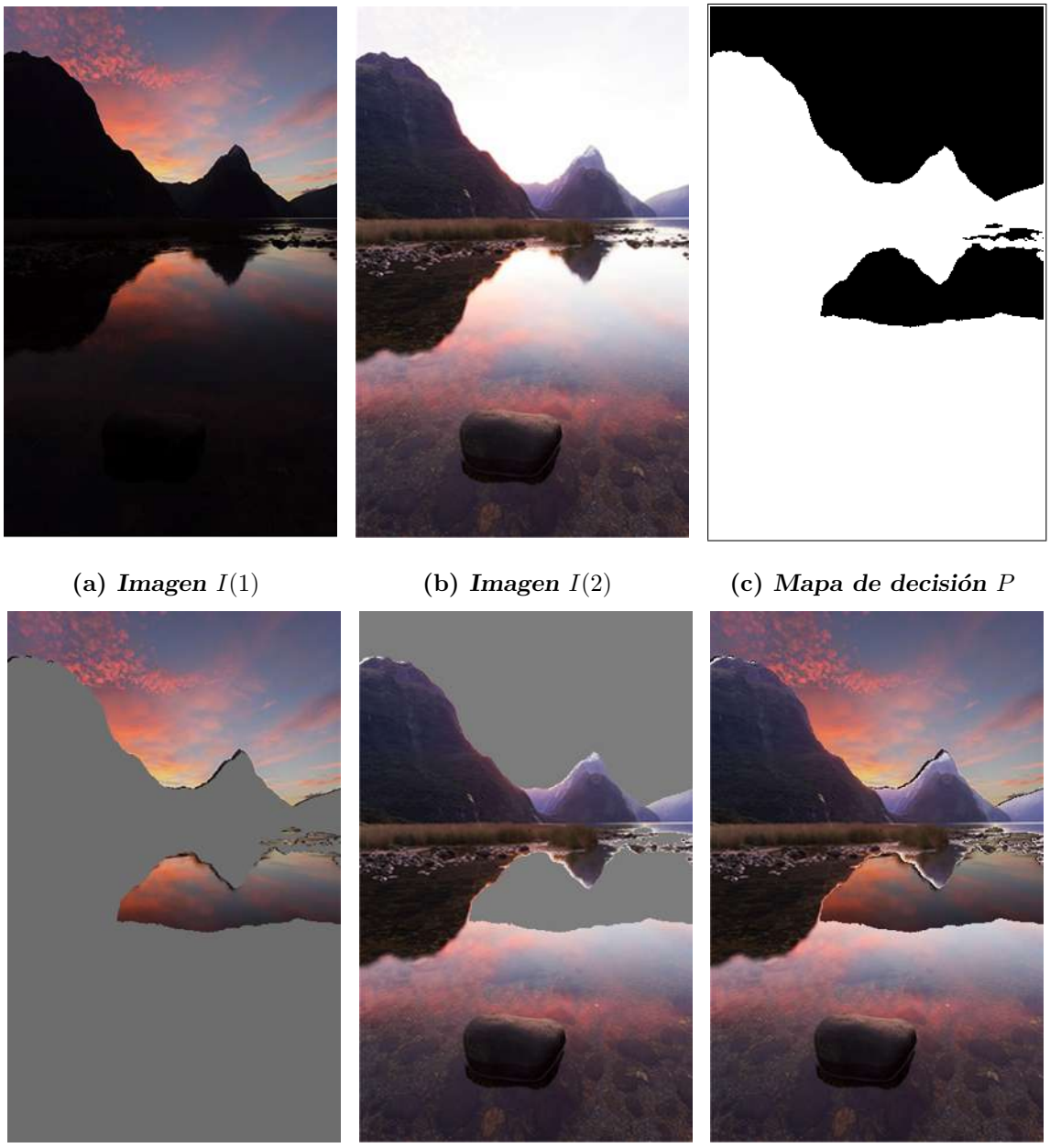
Tabla 6.7: Métricas calculadas para los 4 conjuntos de 3 imágenes multi-foco utilizando los mismos parámetros.

Desc.	**PA**	**MPA**	**MIU**	**FWIU**
Buzo	0.9263	0.9393	0.8228	0.8726
Lobos Marinos	0.9201	0.9319	0.8496	0.8551
Teclado	0.9781	0.9755	0.9540	0.9571
Promedio	0.9319	0.9364	0.8612	0.8776

6.2. Fusión de imágenes con iluminación no uniforme

El algoritmo FI-VA permite la fusión de imágenes en base a una función que permite medir la propiedad de interés de las imágenes. En el caso de las imágenes con iluminación no uniforme, la función debe medir la correcta iluminación. En las Figuras 6.15(a) y 6.15(b) se muestra un ejemplo de dos imágenes (una sub-expuesta y otra sobre-expuesta), correspondientes a la escena de una puesta de sol. En la Figura 6.15(c) se muestra el mapa de decisión obtenido con el algoritmo FI-VA. En las Figuras 6.15(d) y 6.15(e) se muestra las regiones bien iluminadas de las Figuras 6.15(a) y 6.15(b) respectivamente. Finalmente en la Figura 6.15(f) se muestra la fusión de las imágenes de las Figuras 6.15(a) y 6.15(b), en esta última figura se puede observar la escena completa sin regiones saturadas ni sub-expuestas.

En la Figura 6.16(d) se muestra la fusión de las imágenes con iluminación no uniforme mostradas en las Figuras 6.16(a) y 6.16(b), las cuales se fusionan en base al mapa de decisión mostrado en 6.16(c) obtenido con el algoritmo FI-VA. Las Figuras **??** y **??** se muestran las regiones bien iluminadas de 6.16(a) y 6.16(b) respectivamente. En la Figura 6.16(d) se pueden observar todas las regiones de la escena sin que existan zonas saturadas ni sub-expuestas.



(a) ***Imagen*** $I(1)$ **(b)** ***Imagen*** $I(2)$ **(c)** ***Mapa de decisión*** P

(d) ***Regiones bien ilumindas de 6.15(a)*** **(e)** ***Regiones bien ilumindas de 6.15(b)*** **(f)** ***Imagen fusionada***

Figura 6.15: Ejemplo de fusión de imágenes con diferente iluminación de una escena de puesta del sol.

Podemos notar en los ejemplos de fusión de imágenes con iluminación no uniforme mostrados anteriormente, que el algoritmo FI-VA logra la fusión de las imágenes,

(a) ***Imagen*** $I(1)$ **(b)** ***Imagen*** $I(2)$ **(c)** ***Mapa de decisión*** **(d)** ***Fusionada***

Figura 6.16: Ejemplo de fusión de imágenes con iluminación no uniforme de una escultura.

eliminando las regiones sub-expuestas. Por lo tanto, el algoritmo de fusión puede ser aplicado a la fusión de imágenes bajo diferentes contextos, siempre y cuando se seleccione la función adecuada para medir la propiedad de interés.

6.3. Binarización de imágenes con iluminación no uniforme

En la Figura 6.17 se muestra un conjunto de ejemplos de binarización de imágenes en escala de grises, con iluminación no uniforme. Para cada renglón de dicha figura, en la primera columna se muestra la imagen a binarizar, en la segunda columna se muestra el mapa de decisión obtenido por el algoritmo BI-VA. La tercera y cuarta columna muestran el resultado de la binarización lograda con el algoritmo BI-VA y el método de *Bradley y Roth*, respectivamente. Comparando las imágenes de la tercera y cuarta columna, podemos ver que el algoritmo BI-VA obtiene imágenes binarizadas muy similares a las obtenidas por el método de *Bradley y Roth* y en dichas imágenes no existen zonas mal iluminadas. En algunos casos el algoritmo BI-VA obtiene una binarización con menos ruido que el método de *Bradley* lo cual se logra gracias al uso

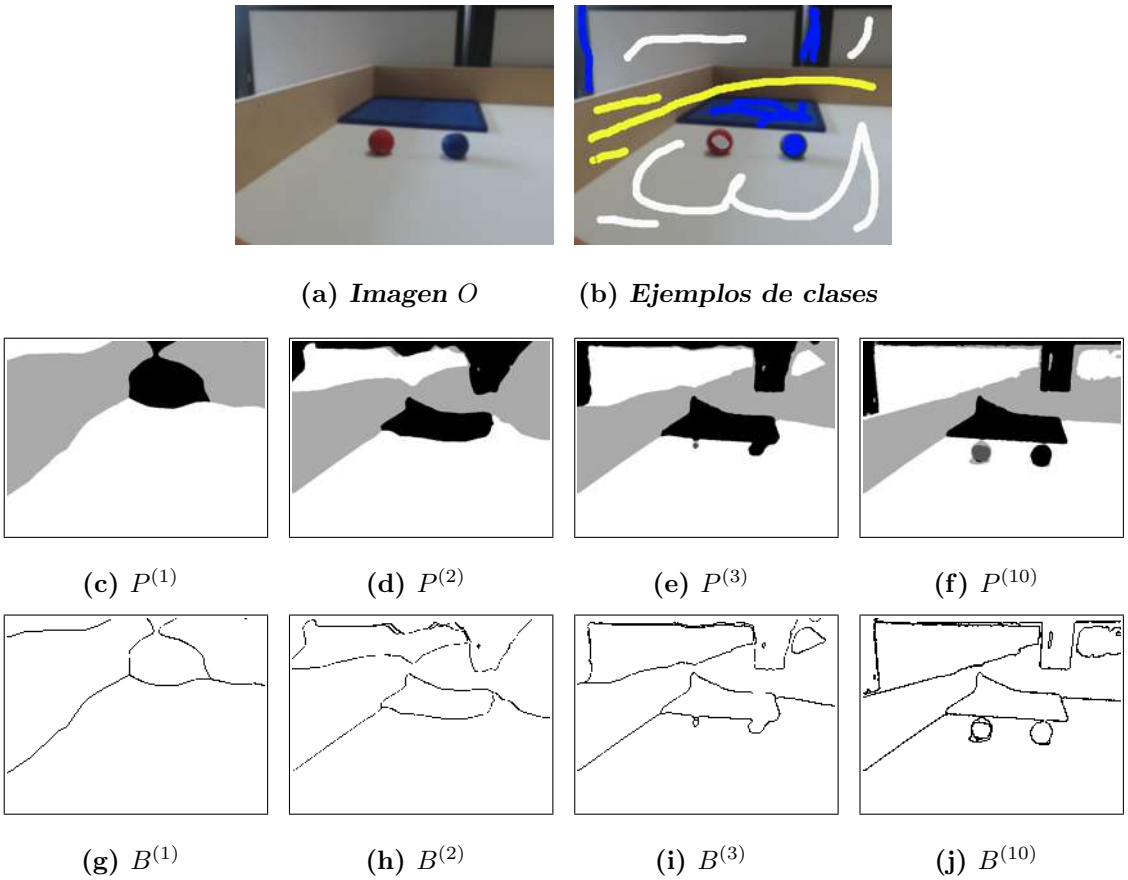
de ventanas de tamaño adaptable y a la separación de las regiones claras y obscuras de la imagen original.

Figura 6.17: Ejemplos de binarización de imágenes con iluminación no uniforme utilizando el método de *Bradley y Roth* y el algoritmo BI-VA.

6.4. Segmentación de imágenes por color

En el Capítulo 5 se presenta la aplicación del algoritmo SI-VA para la segmentación de imágenes. En la Figura 6.18 se muestra un ejemplo de la aplicación de dicho algoritmo para segmentar la Figura 6.18(a) con los ejemplos marcados en la Figura 6.18(b). Las Figuras 6.18(c-f) muestran los mapas de decisión en las iteraciones 1, 2, 3 y 10, respectivamente. Las Figuras 6.18(g-j) muestran los bordes del mapa de decisión respectivo.

En las Figuras 6.19(a-d) se muestra la segmentación de las clases 1 a 4 de la



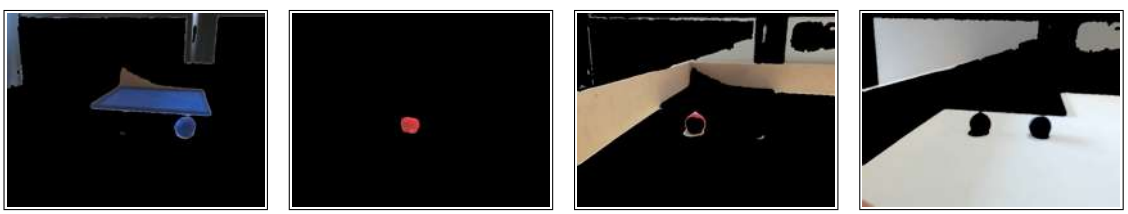
(a) ***Imagen*** O **(b)** ***Ejemplos de clases***

(c) $P^{(1)}$ **(d)** $P^{(2)}$ **(e)** $P^{(3)}$ **(f)** $P^{(10)}$

(g) $B^{(1)}$ **(h)** $B^{(2)}$ **(i)** $B^{(3)}$ **(j)** $B^{(10)}$

Figura 6.18: Evolución del mapa de decisión para la segmentación de la imagen de la Figura 6.18(a).

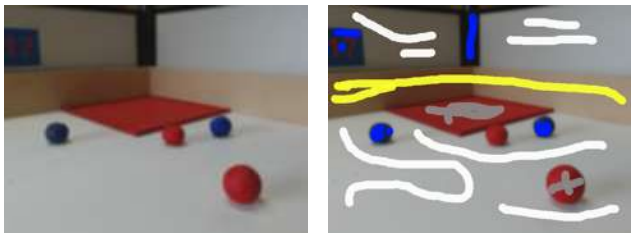
imagen mostrada en la Figura 6.18(a), utilizando el mapa de decisión obtenido con el algoritmo SI-VA que se muestra en la Figura 6.18(f). Podemos observar que la separación de las clases lograda en este caso es adecuada y que se han segmentado de forma adecuada cada uno de los objetos de la escena.



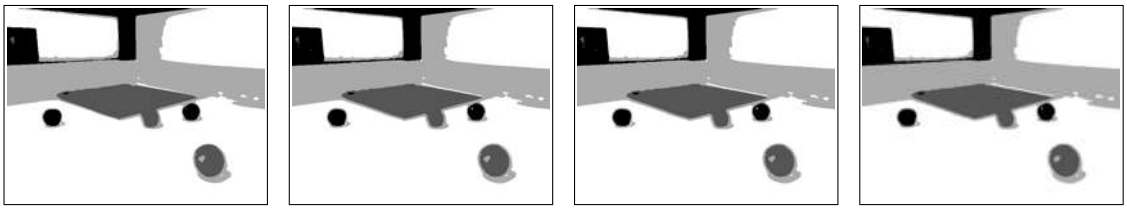
(a) ***Clase 1*** **(b)** ***Clase 2*** **(c)** ***Clase 3*** **(d)** ***Clase 4***

Figura 6.19: Segmentación de las 4 clases de la imagen mostrada en la Figura 6.18(a).

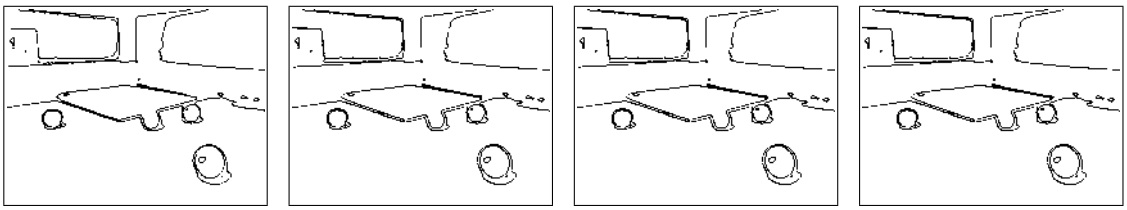
En la Figura 6.20(a) se muestra otra imagen que se segmenta en base a los ejemplos de las clases marcadas en la Figura 6.20(b). El segundo renglón de la Figura 6.20 se muestra el mapa de decisión en as iteraciones 1, 2, 3 y 10. Las imágenes mostradas en el tercer renglón de la Figura 6.20 muestran los bordes de los mapas de decisión respectivos. Nótese que los bordes en la iteración 10 se ajustan muy bien a los bordes de los objetos de la escena y que el mapa de la última iteración permite identificar de forma clara cada objeto de la escena.



(a) ***Imagen*** O **(b)** ***Ejemplos de clases***



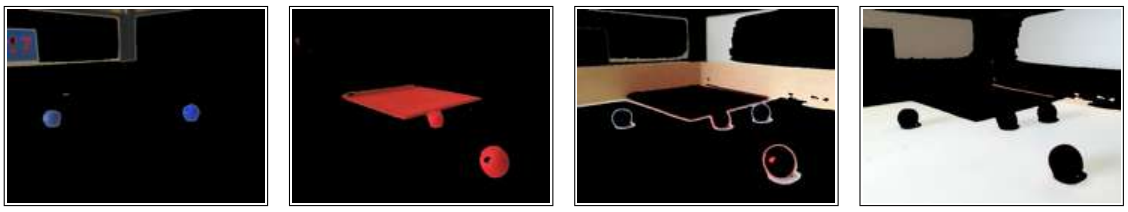
(c) $P^{(1)}$ **(d)** $P^{(2)}$ **(e)** $P^{(3)}$ **(f)** $P^{(10)}$



(g) $B^{(1)}$ **(h)** $B^{(2)}$ **(i)** $B^{(3)}$ **(j)** $B^{(10)}$

Figura 6.20: Ejemplo de segmentación de imágenes con 4 clases.

En la Figura 6.21(a-d) se muestra la separación de las 4 clases de la imagen de la Figura 6.20(a) en base al mapa de decisión mostrado en la Figura 6.20(f). Podemos notar en las imágenes de la Figura 6.21, que se ha logrado separar de forma correcta cada una de las clases de la escena marcadas con los ejemplos.



(a) ***Clase 1*** **(b)** ***Clase 2*** **(c)** ***Clase 3*** **(d)** ***Clase 4***

Figura 6.21: Ejemplo 2 de segmentación de imágenes con 4 clases.

Finalmente, mostramos un ejemplo de la segmentación de la fotografía mostrada en la Figura 6.22(a) que corresponde a una escena donde aparece una persona y en el fondo de la escena se aprecia vegetación y agua. Para la segmentación de la imagen se seleccionan ejemplos de dos clases los cuales se marcaron como se muestra en la Figura 6.22(b). Las Figuras 6.22(e)-6.22(h) muestran los mapas de decisión obtenidos en las iteraciones 1, 2, 3 y 10 respectivamente. Es posible notar que se obtiene un mapa de decisión que permite segmentar las clases de forma adecuada, adicionalmente, podemos ver que desde la primera iteración se logra una buena aproximación al mapa de decisión final. La segmentación de cada clase se muestra en las Figuras 6.22(c) y 6.22(d), donde se puede notar que se ha logrado separar las clases con muy buena precisión.

6.5. Conclusiones

En este capítulo se ha presentado evidencia del funcionamiento del algoritmo FI-VA (presentado en le Capítulo 3) para la fusión de imágenes multi-foco y de imágenes con iluminación no uniforme. También se han presentado los tiempos de ejecución consumidos por el algoritmo para el proceso de fusión y se puede notar en los datos mostrados, que el algoritmo logra la fusión en décimas de segundo, lo que permite la aplicación del algoritmo para procesos en tiempo real.

También presentamos evidencia de la eficacia del algoritmo BI-VA (presentado

(a) ***Imagen*** O (b) ***Ejemplos*** (c) ***Clase 1*** (d) ***Clase 2***

(e) $P^{(1)}$ (f) $P^{(2)}$ (g) $P^{(3)}$ (h) $P^{(10)}$

Figura 6.22: Ejemplo de segmentación de imágenes con 2 clases.

en el Capítulo 4) para la binarización de imágenes con iluminación no uniforme, obteniendo resultados que son competitivos con las propuestas del estado del arte. Finalmente presentamos ejemplos de la aplicación del algoritmo SI-VA (presentado en el Capítulo 5) aplicado a la segmentación de imágenes por color, logrando una buena segmentación de las clases para los ejemplos dados.

Adicionalmente presentamos evidencia el algoritmo FI-VA para la fusión de imágenes multi-foco el cual tiene tiempos de ejecución de décimas de segundo. Este algoritmo FI-VA es similar al algoritmo CLI-VA en el caso de pares de imágenes multi-foco y presenta mejores resultados en tiempo y porcentaje de acierto que el algoritmo CLI-VA de acuerdo con los resultados presentados.

Capítulo 7

Conclusiones

7.1. Conclusiones

En este trabajo presentamos los Algoritmos FI-VA (para fusión de imágenes), BI-VA (para binarización de imágenes con iluminación no uniforme) y SI-VA (para la segmentación de imágenes por color). Dichos algoritmos son el mismo en esencia, con pequeñas modificaciones que permiten la aplicación al problema en cuestión. La diferencia más significativa entre los algoritmos listados es la función que permite medir la propiedad de interés para el problema especifico, por lo que podemos decir que el algoritmo FI-VA puede ser aplicado a otros contextos cambiando dicha función y adecuando los parámetros de entrada y el resultado en función del problema al que se aplicará.

Se ha mostrado evidencia de que el algoritmo tiene complejidad asintótica lineal con respecto al número de pixeles de la imagen, a diferencia de las propuestas en las que se presentan soluciones que involucran procesos de optimización, los cuales generalmente tienen complejidad exponencial. Adicionalmente, hemos explicado que el calculo del valor de cada coordenada del mapa de decisión es independiente de los valores de cualquier otra coordenada, por lo que hemos presentado la paralelización del algoritmo como una opción para reducir el tiempo necesario para obtener la solución.

El mapa de decisión calculado en los algoritmos presentados en este trabajo, se obtiene procesando las imágenes con un sistema de ventanas deslizantes, lo que permite resolver el problema de forma local en lugar de resolverlo para toda la imagen, con lo que se logra una reducción significativa en el consumo de recursos. Adicionalmente, se ha evitado la necesidad de un proceso de optimización para encontrar los valores del mapa de decisión, con lo que se reduce el tiempo necesario para encontrar la solución.

El cálculo del mapa de decisión utilizando ventanas tiene la desventaja de que tamaños de ventana muy grandes hacen que se pierda la definición en los bordes y tamaños de ventana muy pequeños provocan la existencia de ruido y falta de coherencia espacial. En nuestra propuesta, presentamos el uso de ventanas de tamaño adaptable, usando ventanas de tamaño pequeño cerca de los bordes y ventanas de tamaño mayor en las regiones donde no hay bordes. Lo anterior permite aprovechar las ventajas de las ventanas de tamaño mayor para lograr la coherencia espacial y las ventajas de menor tamaño para lograr una buena definición en los bordes. Se ha presentado el uso de imágenes integrales, con lo que se logra que el tamaño de la ventana no influya en el tiempo requerido para encontrar el mapa de decisión, puesto que el tiempo requerido para el cálculo del valor de una coordenada del mapa de decisión es constante y no depende del tamaño de la ventana. Adicionalmente, para obtener el tamaño de ventana adecuado en cada coordenada, se aplica búsqueda binaria, lo que permite encontrar dicho valor de forma eficiente.

Se presentan ejemplos de la aplicación de cada uno de los algoritmos y en el caso del algoritmo FI-VA se muestran métricas cuantitativas que nos permiten concluir que nuestra propuesta es competitiva con las propuestas del estado del arte, logrando porcentajes de acierto promedio de 99.19 % para 21 imágenes sintéticas y resultados que cualitativa mente son muy buenos para los conjuntos de imágenes reales.

Nuestra propuesta ofrece la ventaja de que puede ser aplicado, además de la fusión multi-foco, en otros contextos del procesamiento de imágenes en este trabajo presentamos la aplicación para binarización, segmentación y fusión de imágenes con

iluminación no uniforme.

La simpleza del algoritmo propuesto permite que los resultados se obtengan en décimas de segundo, por lo que podría ser utilizado en software de cámaras o de equipos de captura de imágenes para mejorar las imágenes obtenidas con dichos equipos.

Finalmente, debemos hacer notar que el algoritmo FI-VA permite la fusión de conjuntos de N imágenes con $N \geq 2$ cuando la mayoría de los algoritmos propuestos en el estado del arte sólo atacan el problema para pares de imágenes.

Por lo expuesto en los párrafos anteriores, podemos concluir que se han cumplido los objetivos planteados en el Capítulo 1 de este trabajo, presentando un algoritmo que permite resolver varios problemas relacionados con el procesamiento de imágenes con muy buena precisión y con un consumo de tiempo que podría permitir su aplicación en tiempo real.

7.2. Trabajos futuros

En el área de la fusión de imágenes multi-foco y en general en el procesamiento de imágenes, existe aún mucho camino por recorrer. Dada la gran cantidad de imágenes que existen hoy en día, se requieren algoritmos que permitan realizar la fusión, binarización, segmentación, extracción de información, corrección de las imágenes de manera más rápida y con un desempeño mejor. A continuación se presentan algunas lineas de investigación que podrían ser continuación de este trabajo de tesis o de una nueva investigación.

1. Una tarea trascendental para la solución del problema de fusión de imágenes multi-foco es encontrar una medida del nivel de enfoque, robusta e independiente de la imagen en cuestión. La mayoría de los métodos presentados en la literatura para medir el nivel de enfoque, son muy sensibles al ruido, por lo que un trabajo a futuro es buscar una medida de nitidez robusta, que permita lograr un mejor desempeño al medir el nivel de enfoque o desenfoque en una imagen. Por ejemplo

el uso de filtros pasa-banda que permitan rechazar el ruido de alta frecuencia y las regiones borrosas o con poca textura de las imágenes.

2. Aunque en este trabajo se presenta el uso del algoritmo presentado para el procesamiento de imágenes en otros contextos diferentes a fusión, es necesario establecer funciones que permitan medir de mejor manera la correcta ilumina-ción de un pixel en una imagen, con lo que s lograría un mejor desempeño en la fusión y en la binarización de imágenes con iluminación no uniforme.

3. Del mismo modo, para la segmentación de imágenes se requiere de funciones de verosimilitud más robustas que permitan segmentar con mayor precisión cada una de las clases de una imagen.

Referencias

Agapiou, A., Alexakis, D. D., Sarris, A., y Hadjimitsis, D. G. (2016). Colour to greyscale pixels: Re-seeing greyscale archived aerial photographs and declassified satellite corona images based on image fusion techniques. *Archaeological Prospection*, 23(4):231–241.

Aslantas, V. y Kurban, R. (2010). Fusion of multi-focus images using differential evolution algorithm. *Expert Systems with Applications*, 37(12):8861 – 8870.

Aslantas, V. y Toprak, A. N. (2014). A pixel based multi-focus image fusion method. *Optics Communications*, 332:350 – 358.

Assirati, L., Silva, N. R., Berton, L., Lopes, A. A., y Bruno, O. M. (2014). Performing edge detection by difference of gaussians using q-gaussian kernels. *Journal of Physics: Conference Series*, 490(1):012020.

Bae, S. y Durand, F. (2007). Defocus magnification. *Computer Graphics Forum*, 26(3):571–579.

Bai, X., Zhang, Y., Zhou, F., y Xue, B. (2015). Quadtree-based multi-focus image fusion using a weighted focus-measure. *Information Fusion*, 22:105 – 118.

Bejinariu, S., Rotaru, F., Nita, C., y Luca, R. (2013). Parallel approach for multi-focus image fusion. In *Signals, Circuits and Systems (ISSCS), 2013 International Symposium on*, pages 1–4.

Bernsen, J. (1986). Dynamic thresholding of gray-level images. In *Proc. Eighth Int'l conf. Pattern Recognition, Paris, 1986*.

Bhatnagar, G., Wu, Q. J., y Liu, Z. (2013). Human visual system inspired multi-modal medical image fusion framework. *Expert Systems with Applications*, 40(5):1708 – 1720.

Bradley, D. y Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of graphics tools*, 12(2):13–21.

Calderon, F., Flores, J., y Garnica, A. (2015). A fast algorithm for binary segmentation using color information. In *2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–7.

Calderon, F. y Garnica, A. (2014). Multi focus image fusion based on linear combination of images. In *Power, Electronics and Computing (ROPEC), 2014 IEEE International Autumn Meeting on*, pages 1–7. IEEE.

Calderon, F., Garnica, A., y Flores, J. J. (2016). Fusión de imágenes multi foco basado en la combinación lineal de imágenes utilizando imágenes incrementales. *Revista Iberoamericana de Automática e Informática Industrial {RIAI}*, 13(4):450 – 461.

Calderon, F., Garnica, A., y Flores, J. J. (2017). Fusión de imágenes multi-foco con ventanas variables. *Revista Iberoamericana de Automática e Informática industrial*, 0(0).

Cao, L., Jin, L., Tao, H., Li, G., Zhuang, Z., y Zhang, Y. (2015). Multi-focus image fusion based on spatial frequency in discrete cosine transform domain. *Signal Processing Letters, IEEE*, 22(2):220–224.

Chaudhuri, S. y Kotwal, K. (2013). *Hyperspectral image fusion*. Springer.

De, I. y Chanda, B. (2013). Multi-focus image fusion using a morphology-based focus measure in a quad-tree structure. *Information Fusion*, 14(2):136 – 146.

Dunn, D. y Higgins, W. (1995). Optimal gabor filters for texture segmentation. *Image Processing, IEEE Transactions on*, 4(7):947–964.

Elder, J. y Zucker, S. (1998). Local scale control for edge detection and blur estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(7):699–716.

Eskicioglu, A. y Fisher, P. (1995). Image quality measures and their performance. *Communications, IEEE Transactions on*, 43(12):2959–2965.

Farid, M. S., Mahmood, A., y Al-Maadeed, S. A. (2019). Multi-focus image fusion using content adaptive blurring. *Information Fusion*, 45:96 – 112.

Garnica, A., Calderon, F., y Flores, J. (2018). Multi-focus image fusion by local optimization over sliding windows. *Signal, Image and Video Processing*.

Garnica, A., Gómez, S. D. P., Gómez, J. d. D. P., y Muñoz, L. R. (2016). Development of a mobile service robot for classify objects and place them in containers. In *2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–6. IEEE.

Himanshi, B. V., Krishn, A., y Sahu, A. (2015). Medical image fusion in curvelet domain employing pca and maximum selection rule. In *Proc.(Springer) 2nd International Conference on Computers and Communication Technologies (IC3T-2015), Hyderabad, India*, volume 1, pages 1–9.

Jiang, A.-H., Huang, X.-C., Zhang, Z.-H., Li, J., Zhang, Z.-Y., y Hua, H.-X. (2010). Mutual information algorithms. *Mechanical Systems and Signal Processing*, 24(8):2947 – 2960.

Kausar, N., Majid, A., y Javed, S. G. (2016). Developing learning based intelligent fusion for deblurring confocal microscopic images. *Engineering Applications of Artificial Intelligence*, 55:339 – 352.

Khurshid, K., Siddiqi, I., Faure, C., y Vincent, N. (2009). Comparison of niblack inspired binarization methods for ancient documents. In *Document Recognition and Retrieval XVI*, volume 7247, page 72470U. International Society for Optics and Photonics.

Kong, W. y Lei, Y. (2017). Multi-focus image fusion using biochemical ion exchange model. *Applied Soft Computing*, 51:314 – 327.

Kuthirummal, S., Nagahara, H., Zhou, C., y Nayar, S. (2011). Flexible depth of field photography. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):58–71.

Lewis, J. J., Callaghan, R. J. O., Nikolov, S. G., Bull, D. R., y Canagarajah, N. (2007). Pixel- and region-based image fusion with complex wavelets. *Information Fusion*, 8(2):119 – 130. Special Issue on Image Fusion: Advances in the State of the Art.

Li, S., Kang, X., Fang, L., Hu, J., y Yin, H. (2017). Pixel-level image fusion: A survey of the state of the art. *Information Fusion*, 33:100 – 112.

Li, S., Kwok, J. T., y Wang, Y. (2001). Combination of images with diverse focuses using the spatial frequency. *Information Fusion*, 2(3):169 – 176.

Li, S. y Yang, B. (2008). Multifocus image fusion using region segmentation and spatial frequency. *Image and Vision Computing*, 26(7):971 – 979.

Liu, S. L., Niu, Z. D., Sun, G., y Chen, Z. P. (2014a). Gabor filter-based edge detection: A note. *Optik - International Journal for Light and Electron Optics*, 125(15):4120 – 4123.

Liu, Y., Chen, X., Peng, H., y Wang, Z. (2017). Multi-focus image fusion with a deep convolutional neural network. *Information Fusion*, 36:191 – 207.

Liu, Z., Yin, H., Chai, Y., y Yang, S. X. (2014b). A novel approach for multimodal medical image fusion. *Expert Systems with Applications*, 41(16):7425 – 7435.

Long, J., Shelhamer, E., y Darrell, T. (2014). Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038.

Lu, L., Zhang, X., Xu, X., y Shang, D. (2017). Multispectral image fusion for illumination-invariant palmprint recognition. *PLOS ONE*, 12(5):1–22.

Ma, Y., Zhan, K., Wang, Z., y service), S. O. (2011). Applications of pulse-coupled neural networks.

Molina, E., Diaz, J., Hidalgo-Silva, H., y Chávez, E. (2018). Algoritmos de binarizacion robusta de imágenes con iluminación no uniforme. *Revista Iberoamericana de Automática e Informática industrial*, 15(3):252–261.

Nejati, M., Samavi, S., y Shirani, S. (2015). Multi-focus image fusion using dictionary-based sparse representation. *Information Fusion*, 25:72 – 84.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

Pagidimarry, M. y Babu, K. A. (2011). An all approach for multi-focus image fusion using neural network. *Artificial Intelligent Systems and Machine Learning*, 3(12):732–739.

Pajares, G. y de la Cruz, J. M. (2004). A wavelet-based image fusion tutorial. *Pattern Recognition*, 37(9):1855 – 1872.

Paninski, L. (2003). Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191.

Pramanik, S., Prusty, S., Bhattacharjee, D., y Bhunre, P. K. (2013). A region-to-pixel based multi-sensor image fusion. *Procedia Technology*, 10:654 – 662.

Riaz, M., Park, S., Ahmad, M., Rasheed, W., y Park, J. (2008). Generalized laplacian as focus measure. In Bubak, M., van Albada, G., Dongarra, J., y Sloot, P., editors, *Computational Science ICCS 2008*, volume 5101 of *Lecture Notes in Computer Science*, pages 1013–1021. Springer Berlin Heidelberg.

Sauvola, J. y Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern recognition*, 33(2):225–236.

Sezan, M., Pavlovic, G., Tekalp, A., y Erdem, A. (1991). On modeling the focus blur in image restoration. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 2485–2488 vol.4.

Shah, P., Merchant, S. N., y Desai, U. B. (2013). Multifocus and multispectral image fusion based on pixel significance using multiresolution decomposition. *Signal, Image and Video Processing*, 7(1):95–109.

Shreyamsha Kumar, B. K. (2013). Multifocus and multispectral image fusion based on pixel significance using discrete cosine harmonic wavelet transform. *Signal, Image and Video Processing*, 7(6):1125–1143.

Shreyamsha Kumar, B. K. (2015). Image fusion based on pixel significance using cross bilateral filter. *Signal, Image and Video Processing*, 9(5):1193–1204.

Singh, T. R., Roy, S., Singh, O. I., Sinam, T., Singh, K., et al. (2012). A new local adaptive thresholding technique in binarization. *arXiv preprint arXiv:1201.5227*.

Starck, J.-L. y Murtagh, F. (2006). Deconvolution. In *Astronomical Image and Data Analysis*, Astronomy and Astrophysics Library, pages 71–110. Springer Berlin Heidelberg.

Tian, Q.-C. y Cohen, L. D. (2018). A variational-based fusion model for non-uniform illumination image enhancement via contrast optimization and color correction. *Signal Processing*, 153:210 – 220.

Viola, P. y Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1.

Wang, Z. y Ma, Y. (2008). Medical image fusion using m-pcnn. *Information Fusion*, 9(2):176 – 185.

Yang, Y. (2011). A novel {DWT} based multi-focus image fusion method. *Procedia Engineering*, 24(0):177 – 181. International Conference on Advances in Engineering 2011.

Yang, Y., Que, Y., Huang, S.-Y., y Lin, P. (2017). Technique for multi-focus image fusion based on fuzzy-adaptive pulse-coupled neural network. *Signal, Image and Video Processing*, 11(3):439–446.

Yang, Y., Tong, S., Huang, S., y Lin, P. (2015). Multifocus image fusion based on nsct and focused area detection. *IEEE Sensors Journal*, 15(5):2824–2838.

Yu, X., Ren, J., Chen, Q., y Sui, X. (2014). A false color image fusion method based on multi-resolution color transfer in normalization ycbcr space. *Optik*, 125(20):6010 – 6016.

Yuan, L., Sun, J., Quan, L., y Shum, H.-Y. (2008). Progressive inter-scale and intra-scale non-blind image deconvolution. *ACM Transactions on Graphics*, 27(3):74.

Zhang, Y., Chen, L., Zhao, Z., y Jia, J. (2016). Multi-focus image fusion based on cartoon-texture image decomposition. *Optik - International Journal for Light and Electron Optics*, 127(3):1291 – 1296.

Zhang, Z. y Blum, R. (1999). A categorization of multiscale-decomposition-based image fusion schemes with a performance study for a digital camera application. *Proceedings of the IEEE*, 87(8):1315–1326.

Zhou, L., Ji, G., Shi, C., Feng, C., y Nian, R. (2006). *A Multi-focus Image Fusion Method Based on Image Information Features and the Artificial Neural Networks*, volume 344, pages 747–752. Springer Berlin Heidelberg, Berlin, Heidelberg.

Zhou, X., Zhou, F., Bai, X., y Xue, B. (2014). A boundary condition based deconvolution framework for image deblurring. *Journal of Computational and Applied Mathematics*, 261(0):14 – 29.

Glosario

BI-VA Binarización de imágenes con ventanas adaptables. 95

CLI Combinación lineal de imágenes. 44

CLI-VA Combinación lineal de imágenes con ventanas adaptables. 59

CLI-S Combinación lineal de imágenes simple. 55

CLI-V Combinación lineal de imágenes utilizando ventanas deslizantes. 52

DoG Diferenchia de Gaussianas . 29

FI-VA Fusión de imágenes con ventanas adaptables. 73

FWIU Frecuencia Pesada de la Intersección sobre la Union (Frequency Weighted Intersection over Union). 42

LoG Laplaciano de Gauss . 32

MIU Promedio de la intersección sobre la unión (Mean Intersection over Union). 42

MPA Promedio de precisión a nivel pixel (Mean Pixel Accuracy). 42

PA Precisión a nivel pixel. 42

PC Profundidad de campo . 18

SI-VA Segmentación de imágenes con ventanas adaptables. 108