UNIVERSIDAD AUTÓNOMA DE GUERRERO

FACULTAD DE INGENIERÍA

TRABAJO DE INVESTIGACIÓN

“SEGMENTACIÓN DE IMÁGENES DEL ENSAYO COMETA”

QUE PRESENTA

C. JUAN GERARDO EPITACIO GÁLVEZ

PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

DIRECTOR DE TRABAJO DE INVESTIGACIÓN

DR. ANTONIO ALARCÓN PAREDES

CHILPANCINGO, GUERRERO, FEBRERO DE 2017.

Contenido

INTRODUCCIÓN

El cáncer es una de las principales causas de mortalidad en el mundo, a nivel mundial se prevé un incremento de muertes por cáncer en un 45% entre 2007 y 2030, (pasaríamos de 7.9 a 11.5 millones de defunciones al año), debido en parte al crecimiento demográfico y al envejecimiento de la población. También se estima que durante el mismo período de tiempo, el número de nuevos casos de cáncer se incrementará de 11.3 millones en 2007 a 15.5 millones para el 2030 (OMS, 2008). Tan sólo en 2012 hubo aproximadamente 14 millones de nuevos casos y 8.2 millones de muertes relacionadas con el cáncer, de acuerdo con el informe mundial sobre el cáncer, en su tercer volumen publicado en 2014.

El cáncer comienza en una célula, mutaciones en la estructura genética de una célula suelen ser los principales responsables en obstaculizar el correcto funcionamiento del ciclo celular, en especial cuando existe daño al genoma que se relaciona al proceso de división celular. Una de las formas para medir daño genético es la electroforesis de células individuales (también llamada ensayo cometa), es uno de los test más utilizados para detectar rupturas en las cadenas de ADN producidos por agentes genotóxicos, la importancia de esta técnica radica en que puede ser llevada a cabo en cualquier célula eucariota, tiene bajo costo, alta sensibilidad y es fácilmente implementada.

El ensayo cometa también ayuda en procesos de biomonitoreo en personas, con esto se pueden detectar estados tempranos de mutaciones en la estructura del ADN y así comenzar un tratamiento para evitar la proliferación del daño y con esto incrementar el riesgo de una persona a sufrir cáncer u otro tipo de enfermedad. En la prueba del ensayo cometa, se identifica el daño genético gracias a la migración de fragmentos de ADN cuando la célula es sometida una corriente eléctrica, lo que permite observar la apariencia de un cometa con una cabeza y cola. La estimación de daño se hace en función de la longitud y cantidad de ADN en la cola del cometa respecto a su núcleo, existen dos maneras de obtener estas estimaciones, una de ellas es, que un operador experimentado las lleve a cabo de manera manual, otra forma de realizarlo es mediante la asistencia de una computadora, que, mediante técnicas de visión computacional, procesamiento de imágenes e inteligencia artificial obtenga las métricas del daño genético que ha sufrido una célula.

En el presente trabajo de investigación se propone e implementa en un programa prototipo un esquema de procesamiento de imágenes que busca segmentar exitosamente las imágenes del ensayo cometa de manera automática buscando mejorar o subsanar deficiencias a los algoritmos que actualmente se están utilizando para este fin. También se proponen y desarrollan ciertas mejoras al algoritmo e interfaz a OpenComet, un complemento del marco de trabajo para procesamiento de imágenes ImageJ que es frecuentemente utilizado para la segmentación de imágenes del ensayo cometa, de código abierto y libre distribución.

JUSTIFICACIÓN

En visión computacional, la segmentación es el proceso en donde se encuentran las regiones de interés en una imagen presentándolas en primer plano y aislarlas de información poco útil para el observador. Contar con técnicas que permitan segmentar correctamente las imágenes generadas en la prueba del ensayo cometa juegan un papel vital al momento de llevar a cabo el desarrollo de un software que obtenga de manera automatizada los datos requeridos en el ensayo, porque como ya se ha mencionado con esto se lograría el correcto monitoreo, oportuno diagnóstico y prevención de diversas enfermedades, entre ellas el cáncer.

Actualmente ya existen alternativas propuestas para esta tarea, pero todas llegan a presentar al menos uno de los problemas que se mencionan a continuación.

- Son muy caros (~$20,000 USD).
- Los resultados arrojan baja sensibilidad y baja especificidad.
- El método de procesamiento no es automático.
- Código fuente no disponible.
- Pobre documentación del código fuente.

En este trabajo de investigación se propone e implementa un sistema de procesamiento de imágenes para lograr una apropiada segmentación de las imágenes del ensayo cometa, mejorando los parámetros de sensibilidad y especificidad, cuidando también la optimización de recursos del sistema y buenas prácticas de desarrollo para asegurar un código mantenible, con esto se solucionan los inconvenientes antes descritos de las alternativas que actualmente existen.

ALCANCES

Proponer e implementar un método de segmentación de imágenes automática que permita identificar las regiones de interés de una imagen típica del ensayo cometa. Con ello, podrá ubicar donde se encuentra el núcleo, halo y cola del cometa, así como el fondo que contiene al cometa en dicha imagen.

OBJETIVO GENERAL

Implementar un método sensible y rápido para la segmentación de imágenes de células de sangre periférica utilizadas en la prueba de ensayo cometa.

OBJETIVOS ESPECÍFICOS

- Realizar operaciones de preprocesamiento a las imágenes resultantes del ensayo.
- Detectar cometas viables a ser analizados.
- Por cada célula resultante, identificar las regiones de núcleo, halo, cola y fondo.

CAPÍTULO I. MARCO TEÓRICO E HIPÓTESIS

1.1 Introducción a la biología celular y molecular

Históricamente el estudio de los seres vivos nos muestra que la evolución produjo una inmensa diversidad de formas vivientes. Existen alrededor de cuatro millones de especies animales, vegetales, protozoarios y bacterias, cuyos comportamientos, morfologías y funciones difieren entre sí. Sin embargo, a nivel molecular y celular estas entidades vivientes presentan un plan maestro de organización único (Robertis & Hib, 2004). La biología celular y molecular es el campo de estudio encargado del análisis de las moléculas y de los componentes celulares con que se conforman todas las formas de vida.

No se puede describir y entender la estructura molecular ni estructuras más complejas de un ser vivo sin antes tener una clara idea de las relaciones que guardan las estructuras moleculares con estructuras celulares, por ello para ahondar en temas de estudio respecto al ADN es necesario conocer la unidad fundamental de todos los organismos vivos, la célula.

1.1.1 La célula

La idea de que todos los organismos están formados por células la propuso por primera vez Robert Hooke, alrededor de 1660. Gracias a los estudios microscópicos de tejido vegetal realizados por Matthias Schleiden y los de tejido animal por Theodor Schwann en 1830 se concluye que, todos los organismos están compuestos por células y que la célula es una unidad estructural de vida. Más tarde, en 1855, Rudolf Virchow demuestra que las células no se forman por generación espontánea si no que emergen únicamente por la división de las células preexistentes.

Los estudios antes mencionados sirvieron para establecer los fundamentos de la teoría celular y nos deja claro que absolutamente todo ser vivo está compuesto por células, desde una diminuta bacteria hasta organismos pluricelulares como el ser humano. Las funciones que una célula debe llevar a cabo en cada organismo dependen del tipo de especialización que tenga. Los organismos más complejos están compuestos por una colección de células que funcionan de manera coordinada, con diferentes células especializadas para desarrollar funciones particulares. En el ser humano, por ejemplo, podemos encontrar hasta 200 clases de células diferentes, cada una de ellas especializadas a cada función distintiva como la memoria, la vista, el movimiento o la digestión (Cooper & Hausman, 2008). En general las células se dividen en dos clases principales eucariotas y procariotas, inicialmente definidas según donde se encuentre el núcleo.

1.1.2 Células Procariotas

Son el tipo de célula más simple que existe en la naturaleza (Figura 1 a), por lo general son bacterias con estructuras que apenas alcanzan unos cuantos micrómetros, a diferencia de una célula eucariota, una célula procariota carece de una envoltura nuclear que englobe su material genético, lo que causa que el ADN se encuentre en contacto directo con el resto de componentes celulares. Al margen de estas diferencias ambos tipos de células comparten propiedades fundamentales que se han conservado a través de la evolución,

por ejemplo, todas las células están rodeadas de una membrana plasmática y utilizan ADN como material genético.

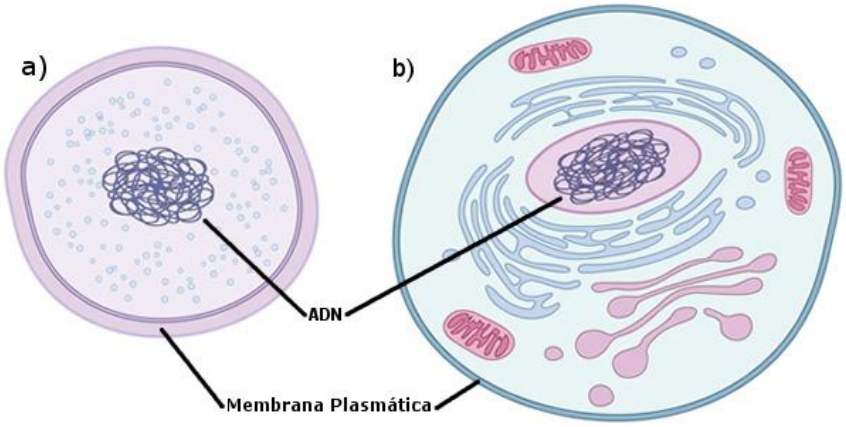

Figura 1 a) Célula procariota b) Célula eucariota, ambas células mantienen una membrana plasmática que engloba todo el contenido de la célula y las separa del mundo no biológico, también contienen ADN como material genético.

1.1.3 Células Eucariotas

Este tipo de células son las que forman los organismos más complejos en la naturaleza, plantas y animales están compuestas por millones de células eucariotas. Además de contener una membrana nuclear, una célula eucariota contiene distintos compartimientos, estructuras internas especializadas llamadas orgánulos.

1.1.4 Orgánulos Celulares

Los orgánulos celulares son los compartimientos de la célula que llevan a cabo funciones específicas, como producir energía, disolver componentes, crear proteínas, entre otras. Algunos orgánulos se encuentran rodeadas de una membrana y otros no, pero todos realizan actividades esenciales en el desarrollo de la célula o durante la división celular. La Figura 2 muestra una típica célula animal y la disposición de sus principales orgánulos. A continuación, se describen brevemente los que se requieren conocer para entender con más claridad las pruebas de genotoxicidad abordados en el apartado 1.2.2.

Membrana Plasmática

Es una estructura trilaminar, formada por una bicapa de lípidos que rodea el contenido de toda la célula, se encuentra tanto en células procariotas, así como en células eucariotas, sus principales funciones son:

- Una barrera de permeabilidad selectiva entre los mundos vivo e inerte, contiene proteínas que son los responsables del tráfico selectivo de las moléculas tanto al interior (endocitosis) como al exterior (exocitosis), por esta razón, también determina la composición del citoplasma.
- Otras proteínas de la membrana plasmática controlan las interacciones entre las células de los organismos multicelulares y actúan como sensores a través de los cuales la célula recibe señales del medio.

- Mediante la formación de pequeñas vesículas transportadoras hacen posible el desplazamiento de sustancias por el citoplasma (Robertis & Hib, 2004).

Citosol

Es la parte del citoplasma que no se encuentra separado por membranas intracelulares. En la mayoría de las células, el citosol es el compartimiento individual más grande. Se encuentra compuesto en su mayoría por agua y otras proteínas, a pesar de su composición, se comporta más como un gel que como un líquido. En el citosol se llevan a cabo los primeros pasos para la descomposición de nutrientes y otras reacciones químicas, por ejemplo, es en el citosol donde los ribosomas llevan a cabo la síntesis de proteínas.

Núcleo

El núcleo es la principal característica que diferencia las células eucariotas de las procariotas, generalmente es el orgánulo más grande y donde se haya confinado el ADN (información genética) de la célula, uno o más nucléolos, así como el nucleoplasma. El núcleo se encuentra limitado por la envoltura nuclear compuesta por dos membranas concéntricas que contiene una red de complejos poros nucleares que definen el tráfico selectivo de componentes desde y hacia el núcleo.

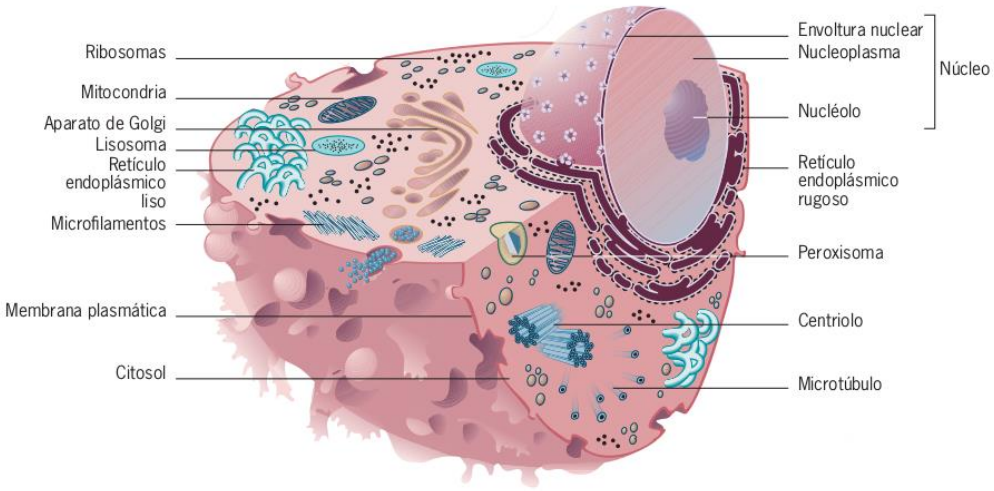

Figura 2 Célula animal y sus principales orgánulos.

1.1.5 ADN

Los miembros de cualquier especie biológica son similares en algunas características, pero diferentes en otras. Por ejemplo, todos los seres humanos comparten un conjunto de características observables, o rasgos, que nos definen como especie. Somos, en finos detalles, un tipo de primate, pero tenemos rasgos que nos apartan de otros primates, como los chimpancés y gorilas. Dentro de la especie humana, sin embargo, hay también mucha variación. Rasgos como el color del cabello, color de ojos, color de piel, altura, peso y características de personalidad son inmensamente variables de una persona a

otra. Algunos de estos rasgos son heredados biológicamente, otros, son heredados culturalmente. El color de ojos resulta de una herencia biológica, el lenguaje nativo que hablamos resulta de una herencia cultural. Muchos rasgos son influenciados por la unión de herencia biología y factores ambientales. La genética se encarga del estudio y análisis de los rasgos biológicamente heredados (Hartl, Jones, & others, 1998).

Los estudios sobre la transmisión de rasgos hereditarios desde progenitores a su progenie comienzan con los experimentos con chicharos llevados a cabo por Mendel en 1860. Sus resultados son vistos hoy en día como leyes que rigen la forma en cómo se transmiten los elementos de herencia de una generación a otra. Posterior a los experimentos de Mendel, Friedrich Miesher descubre un nuevo tipo de sustancia débilmente acidad en los espermas de salmón y en glóbulos blancos sanguíneos. No es hasta a mediados del siglo XX que se descubre que el débil acido de Miesher es, de hecho, el ADN o Acido desoxirribonucleico, la compleja molécula encargada de mover la maquinaria para transmitir los rasgos hereditarios.

1.1.5.1 El ADN como material genético

En 1870 se observa que el núcleo de las células reproductivas masculinas y femeninas se fusionan en el proceso de fertilización, en 1900 se descubre en el núcleo de las células a pequeñas estructuras parecidas a hilos, denominadas más tarde como cromosomas. El número de cromosomas era constante dentro de cada especie, pero difería entre especies. Estas características daban indicios que eran los cromosomas los portadores físicos de los elementos de herencia denominados anteriormente por Mendel como genes.

Los estudios revelaban que el ADN y otras proteínas se encontraban en el cromosoma, sin embargo, debido a que el ADN parecía ser una molécula constante en casi todas las células de cierta especie, se llega a la conclusión errónea que el ADN solo provee un marco de trabajo estructural en los cromosomas. Se denomina entonces a las proteínas como los genes encargados de transmitir la información genética, debido a la gran diversidad en número y tipos que había dentro de las células. Para demostrar que el ADN era realmente el material genético, había que demostrar también que las proteínas no lo eran.

Con los experimentos sobre la bacteria que causa la neumonía (pneumococcus), se define el papel del ADN como material genético. En el experimento se identificaron dos cepas de la bacteria, aquellas con capacidad de infectar al huésped (células S) y otras mutantes que perdieron su capacidad de infección (células R). Las células S eran letales cuando se inyectaban en los ratones, las células R sin embargo eran contrarrestadas por el sistema inmune del ratón. En 1928 se observó que cuando se inyectaban bacterias R más bacterias S (que habían sido inactivadas por calor), los ratones desarrollaban neumonía y morían, y no solo eso, la sangre del ratón parecía contener la forma activa de la bacteria S. Es decir, existía un mecanismo (denominado principio transformador) por la cual las bacterias de tipo S inactivas por calor, trasformaban a las bacterias R a tipo S activas con capacidad de infectar al huésped. En 1944 Oswald Avery, Colin McLeod y Mclyn McCarty establecieron que el principio transformador era el ADN.

A pesar de que estudios demostraban que sin ADN no había infección a otras células, no llevo a la aceptación inmediata del ADN como material genético, fueron ampliados unos pocos años después con experimentos con virus bacterianos. En particular se observó que cuando un virus infectaba a una célula era preciso que el ADN viral penetrara en la célula (y no las proteínas) para que el virus se replicara. Además, a las partículas virales hijas se transmite el ADN del virus progenitor y no sus proteínas. La concurrencia de estos resultados junto con estudios posteriores de la actividad del ADN en la transformación bacteriana llevó a la aceptación de la idea de que el ADN es el material genético (Cooper & Hausman, 2008).

1.1.5.2 Estructura

Incluso con el conocimiento de que los genes son ADN, muchas cuestiones habían permanecido sin respuesta alguna. ¿Cómo se duplica el ADN en el gen cuando una célula se divide?, ¿Cómo controla los rasgos hereditarios el ADN en un gen?, ¿Qué pasa con el ADN cuando una mutación (un cambio en el ADN) toma lugar en el gen? Algunas repuestas a estas preguntas fueron dadas en 1953 con la propuesta de James Watson y Francis Crick de la universidad Cambridge, la estructura fue deslumbrante en elegancia y revolucionaria en sugerir cómo el ADN se duplica a sí mismo, controla los rasgos de herencia y sufre mutaciones (Hartl et al., 1998).

En la época de los trabajos de Watson y Crick se sabía que:

- El ADN era un compuesto de cuatro nucleótidos –dos purinas (Adenina [A], guanina [G]) y dos pirimidinas (Citosina [C] y Timina [T])- unidas a azucares fosforilados.
- Linus Pauling describe las uniones por puentes de hidrogeno, un tipo común de estructura secundaria de las proteínas.
- Existía información experimental sobre la estructura del ADN con los estudios de cristalografía por refracción de rayos X llevados a cabo por Maurice Wilkins y Rosalind Franklin.
- El análisis de los datos obtenidos por la refracción de rayos X reveló que el ADN es una hélice que da un giro cada 3.4 nm, y que la distancia entre las bases es de 0.34 nm por lo que en cada vuelta de la hélice hay diez bases, el diámetro de la hélice era de 2 nm, sugiriendo que estaba compuesto no de una sino de dos cadenas de ADN.

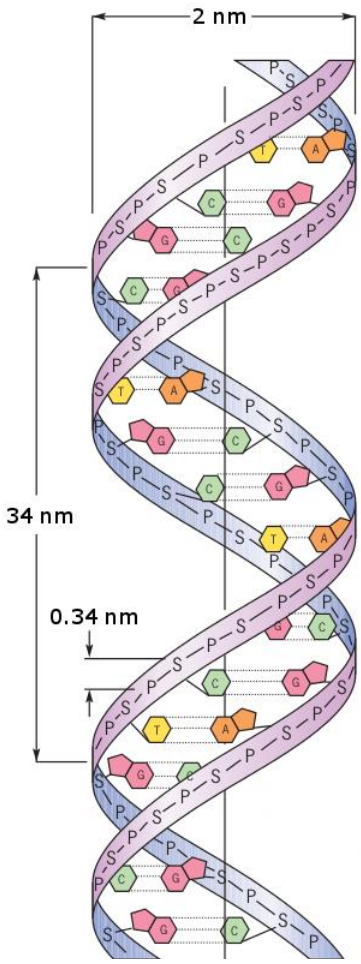

Figura 3 Estructura del ADN propuesta por James Watson y Francis Crick.

A partir de estos datos, Watson y Crick construyeron su modelo de ADN (Figura 3). Del resumen de puntos hechos por Gerald Karp (2011), se toman los que se consideran más importantes y se listan a continuación.

- La molécula se integra con dos cadenas de nucleótidos que se enrollan en espiral una de la otra, formando un par de hélices dextrógiras. Un observador que mire el eje central de la molécula verá que cada cadena sigue una trayectoria en el sentido de las manecillas del reloj a medida que se aleja del observador.
- El esqueleto (azúcar-fosfato-azúcar-fosfato) se localiza en el exterior de la molécula con dos grupos de bases que se proyectan hacia el centro. Los grupos fosfato confieren a la molécula carga negativa.
- Las bases A, T, G y C ocupan planos más o menos perpendiculares al eje longitudinal de la molécula y por lo tanto se colocan una sobre la otra como platos apilados, las vueltas helicoidales y los pares de bases planares en su conjunto confieren a la molécula la semejanza de una escalera de caracol.
- Las dos cadenas se conservan unidas mediante puentes de hidrogeno entre las bases de una cadena y sus correspondientes bases sobre la otra cadena. Debido a la débil unión del puente de hidrogeno posibilita la separación de las cadenas de ADN durante ciertas funciones.
- El ancho de la doble hélice es de 2nm a lo largo de toda su estructura.
- Las restricciones estructurales sugieren que la única purina capaz de unirse a la Timina es la Adenina y que la Guanina es la única purina capaz de unirse con la Citosina, lo que define a los únicos pares posibles en las bases como A-T y G-C. Debido a esta especificidad en el apareamiento de las bases las dos hebras de ADN son complementarias: cada hebra contiene toda la información para especificar la secuencia de bases de la otra.

1.1.5.3 Importancia

Se ha señalado que un gen es un elemento de herencia que es transmitido a las siguientes generaciones, todos los sujetos que integran una especie muestran el mismo grupo de genes, si bien los diferentes individuos poseen de modo invariable ligeras diferencias de muchos de estos genes, es precisamente esta diferencia que permite que cada organismo de la especie muestre un contenido único de información genética. Al conjunto de información genética en un organismo es conocido como genoma. Un genoma define la manera selectiva y regulada, todos los procesos necesarios para “construir” un organismo específico, como nacer, crecer, reproducirse y morir.

Los cambios en el genoma, conocidos también como mutaciones, son procesos necesarios para el desarrollo y evolución de las especies (Karp, 2011). Muchas veces las mutaciones se producen de manera inapropiada o por entidades externas y son adversas a los organismos, estas mutaciones podrían llegar a heredarse a las siguientes generaciones. Cuando ocurren cambios que no son parte del proceso normal de evolución del genoma, se activa en cada organismo una sofisticada maquinaria biológica que intenta revertir este daño o en su defecto, prepara a las células afectadas para entrar a un proceso denominado muerte celular programada. En algunas ocasiones estos procesos de autocontrol fallan o se vuelven insuficientes ya sea por la edad del organismo o que el daño causado al genoma es tan extenso, que ni en las mejores condiciones, los procesos de control celular se pueden llevar a cabo exitosamente.

Las mutaciones descontroladas y no reparadas en un organismo muchas veces terminan en procesos de carcinogénesis o enfermedades degenerativas, por esta razón mantener intacta a las moléculas de ADN en las células de los seres vivos se vuelve una cuestión vital para llevar a cabo un proceso de desarrollo normal.

1.2 Genética toxicológica

La genética toxicológica es la disciplina encargada de estudiar los efectos tóxicos que afectan a las moléculas de ADN (Young, 2002). El daño a la estructura del ADN puede ocurrir por rupturas de cadena simple (SSB), rupturas de cadena doble (DSB), o por eliminación, inserción o reordenamiento de bases o secuencias de ADN en el genoma de un organismo (Hartl et al., 1998). Los responsables del daño al genoma son conocidos como agentes genotóxicos.

1.2.1 Agentes genotóxicos

Se denominan agentes genotóxicos a aquellos agentes capaces de generar mutaciones en las moléculas de ADN, Ximena Abrevaya (2008) clasifica a los agentes genotóxicos según su origen en tres categorías: físicos, químicos y biológicos.

Agentes Físicos

En general se incluyen aquí a las radiaciones en todo su espectro o elementos radioactivos, que van desde los rayos gamma, rayos UV o Rayos X.

Agentes Químicos

Elementos o compuestos químicos, tal como se presentan en estado natural o es producido por el hombre. En este apartado se encuentran distintas sustancias que son conocidos como mutágenos o carcinógenos, por ejemplo, algunos pesticidas, solventes, minerales, metales, gases, vapores, entre otros.

Agentes Biológicos

En agentes biológicos se encuentran algunos parásitos, bacterias, hongos, vegetales o incluso virus que pueden causar daño genético.

1.2.2 Pruebas de genotoxicidad

Tras el descubrimiento de los daños en la estructura del ADN se han venido desarrollando diversas metodologías para cuantificar el daño a nivel celular, a continuación, se listan lo más utilizados.

1.2.2.1 Electroforesis de células individuales

SCGE (por sus siglas en Ingles) es también conocido como el ensayo cometa, es uno de los principales test utilizados para detectar daño al ADN, se identifican dos principales versiones de esta prueba, la versión original, propuesta por Ostling y Johanson realizado en condiciones neutras (Ostling & Johanson, 1984), que solo detectaba roturas de cadena doble o DSB (Double Strand Breaks). Posteriormente Singh y colaboradores en 1988, propusieron una modificación al original que mejoraba la sensibilidad de la prueba, llevándola a cabo en condiciones alcalinas.

Como resultado de las modificaciones, se detectaron tanto roturas de cadena simple o SSB (Simple Strand Breaks), sitios álcali-lábiles (sitios sensibles a la acción de agentes alcalinos), así como DSB, las imágenes de este ensayo se obtienen una vez que las células de manera individual se someten a una electroforesis. Si existe material genético fragmentado, éste tiende a migrar de manera que forman figuras que se asemejan a un cometa. La longitud y la cantidad de ADN en la cola determinan el nivel de daño que existe en la célula (Singh, McCoy, Tice, & Schneider, 1988).

1.2.2.2 Ensayo de micronúcleos

El ensayo de micronúcleos es una técnica ampliamente utilizada que mide el daño genético que induce un agente a una célula, en función de la cantidad de micronúcleos que se forman en la interfase del ciclo celular.

La división celular es un requerimiento esencial para la expresión de micronúcleos, por lo tanto, para mostrar el daño cromosómico después de la exposición a un agente genotóxico, las células suelen someterse a una división celular, la cual es interrumpida permitiendo así la formación de células binucleadas o mononucleadas, con micronúcleos. Los micronúcleos son pequeños fragmentos de ADN que se separan del núcleo principal, que contiene fragmentos centrales o no centrales de un cromosoma, los micronúcleos son el resultado de daño genético causado por un agente genotóxico.

La estimación de daño se realiza en función de la cantidad de micronúcleos presentes en la célula binucleada o mononucleada resultante (Doherty, 2012).

1.2.2.3 Halo

Es un ensayo desarrollado en 1999 por Sestili y Cantoni llamado Alkaline-Halo Assay (AHA), esta técnica es llevada a cabo al igual que el ensayo cometa, basándose en la observación de fragmentos de ADN de células dañadas, preparadas y desproteinizadas de acuerdo al procedimiento de análisis de células a nivel individual (SCG, por sus siglas en ingles).

El ensayo logra la separación de fragmentos dañados de manera radial al ADN sin daño con una simple incubación en una solución desnaturalizada de NaOH (Hidróxido de Sodio). Esto causa que se formen figuras que se asemejan a un halo alrededor del material genético sin daño. La cuantificación de daño se realiza en función del tamaño de los halos formados por cada célula, a mayor halo, mayor daño hay en la estructura de ADN (Sestili & Cantoni, 1999).

1.2.3 Protocolo estándar del Ensayo Cometa

Debido a la versatilidad del ensayo cometa, cuenta con una gran variedad de versiones para detectar distintos tipos de daño en regiones específicas del ADN, sin importar la versión del ensayo, todos cuentan con etapas que se describen brevemente en los siguientes apartados.

1.2.3.1 Microgeles de agarosa

Como primer paso, las células son embebidas en capas de agarosa que permite principalmente asegurar un soporte estable para la manipulación de las células a evaluar

durante todo el proceso, y permitir una fácil y buena visualización con un mínimo ruido en el fondo (Venegas, 2009).

1.2.3.2 Lisis celular

Tras el aseguramiento de las células en los geles de agarosa, las células son sometidas a un proceso de lisis celular en donde las células son tratadas con una alta concentración de detergentes y sales el cual elimina membranas, citoplasma y nucleoplasma. La lisis celular deja como resultado una estructura denominada nucleoide y que consiste en una matriz nuclear compuesto de DNA, RNA y proteínas. Existen dos versiones de la lisis celular, por un lado, la versión neutra propuesta por Ostling y Johanson y otra en condiciones alcalinas por Singh, siendo la segunda la más utilizada.

1.2.3.3 Electroforesis

En el proceso de la electroforesis las células son sometidas a un campo eléctrico, el ADN tiende a migrar hacia el polo positivo de la electroforesis debido a la carga negativa de las cadenas de ADN, la cantidad de migración que tendrán los fragmentos del material genético es inversamente proporcional a su tamaño, los microgeles de agarosa representan en esta etapa una red de poros por el cual el ADN tendrá que traspasar para poder moverse durante la electroforesis.

1.2.3.4 Neutralización y Fijación

La alcalinidad de los detergentes y sales tienen que ser neutralizados después del proceso de electroforesis, para ello se utilizan tampones de neutralización, además de neutralizar la alcalinidad también elimina posibles restos de detergentes que eventualmente podrían inferir en el proceso de tinción y por ende la visualización (Venegas, 2009). La fijación se realiza con alcoholes que permiten almacenar a los objetos durante un tiempo antes de su recuento.

1.2.3.5 Tinción y Visualización

Antes del análisis microscópico las preparaciones deben ser teñidas, generalmente se utiliza el bromuro de etidio para el proceso de tinción. Se debe utilizar un microscopio para la visualización de los cometas, y un sistema de obtención de imágenes para su digitalización.

1.2.4 Imágenes del Ensayo Cometa

Una vez culminado el ensayo cometa se pasa al proceso de observación y cuantificación del daño en el material genético, esta tarea la puede realizar un especialista manualmente o mediante un sistema de software que realice la tarea de manera automática. Las imágenes resultantes del ensayo cometa, varían en gran medida de un laboratorio a otro debido al tipo de tinción, metodología utilizada, así como el tipo de daño que se quiere detectar, pese a que ha habido esfuerzos por proponer una metodología estándar no se han adoptado como tal, incluso en el mismo laboratorio las imágenes cambian.

Las Figuras 4, 5 y 6 muestran diferentes imágenes provenientes del ensayo cometa, las imágenes fueron generadas en los laboratorios de la Unidad Académica de Ciencias Químico-Biológicas de la UAGro.

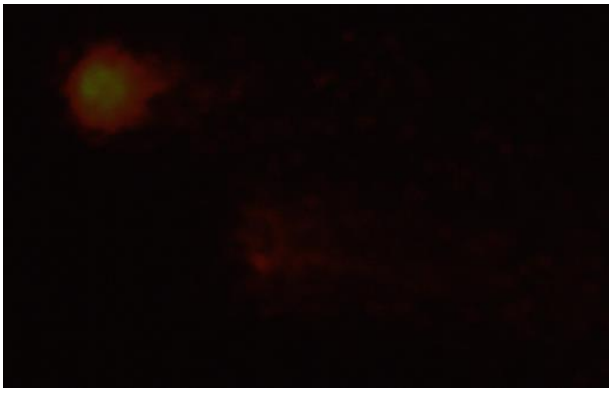

Figura 4 Una imagen del ensayo cometa, a simple vista se observa un solo cometa.

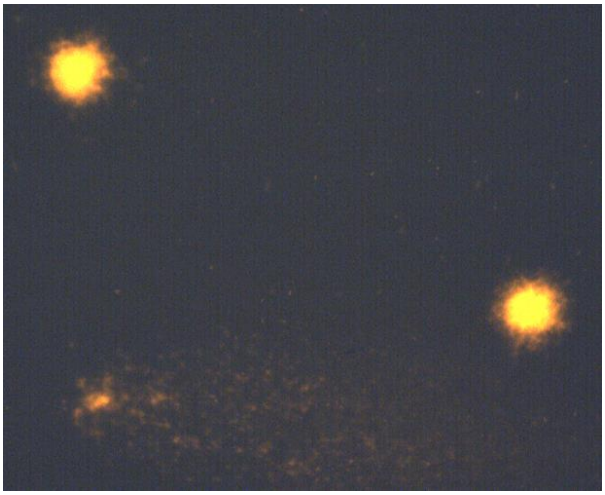

Figura 5 Imagen del ensayo con tres cometas, una con evidente daño y dos sin daño aparente.

Figura 6 Imagen del ensayo cometa, capturada verticalmente.

1.3 Hipótesis

Con técnicas de procesamiento de imágenes, análisis de regiones y algoritmos de aprendizaje no supervisado con un enfoque no difuso es posible segmentar imágenes del ensayo cometa en regiones de Fondo, Cola, Halo y Núcleo de manera automática.

CAPÍTULO II. CONCEPTOS BÁSICOS Y ESTADO DEL ARTE

El apartado de conceptos básicos cubre los temas esenciales de procesamiento de imágenes, se inicia desde los términos más elementales como describir una imagen digital y culmina con descriptores de la imagen. Una vez concluida la teoría básica de procesamiento de imágenes se aborda el tema de reconocimiento de patrones con los conceptos de Patrón, Clase y aprendizaje no supervisado, se describen dos algoritmos de agrupamiento que manejan este enfoque. Este capítulo culmina con la presentación del estado del arte donde se describen las propuestas más relevantes para la segmentación de imágenes del Ensayo Cometa.

2.1 Conceptos básicos

En los conceptos básicos se definen los términos de procesamiento de imágenes, tipos de imágenes, así como los filtros y la convolución, una vez definidos estos conceptos se explica la segmentación de imágenes basada en umbralizado, se abordan también en este apartado algunos descriptores y la forma de calcularlos. Se concluye con la introducción al reconocimiento de patrones y la descripción de los algoritmos K-Means y Fuzzy C-Means.

2.1.1 Procesamiento digital de imágenes

Nos encontramos en un mundo con abundante información visual, el cual se manifiesta a si mismo con una variedad de formas, figuras, colores y texturas. La percepción humana tiene la habilidad de adquirir, integrar e interpretar toda esta abundante información alrededor de nosotros. Es un reto dotar de estas capacidades a una máquina, es importante entender las técnicas de obtención, procesamiento, reconocimiento y finalmente de interpretación de una imagen digital para comenzar este proceso.

Una escena visual o imagen puede ser definida como una función bidimensional $f(x,y)$, donde x e y son coordenadas en el plano y la amplitud de la función f en cualquier par de coordenadas (x, y) es llamado la intensidad o nivel de gris de la imagen en ese punto. Cuando x, y y la amplitud de valores de f son todas finitas llamamos a esta imagen una imagen digital. A cada elemento en la posición (x, y) es llamado pixel (Rafael C Gonzalez & Woods, 2002). Una imagen digital puede representar el nivel de luminosidad de los objetos en una escena (imágenes tomadas desde una cámara ordinaria), la absorción de características del tejido corporal (imágenes con rayos X), el perfil de temperatura de una región (imágenes infrarrojas) o el campo gravitacional en un área (en imágenes geofísicas). En general, cualquier función bidimensional que contenga información puede ser considerada una imagen (Jain, 1989).

No hay un consenso para definir los límites entre el procesamiento de imágenes, análisis de imágenes y visión computacional. Se considera como procesamiento digital de imágenes a los procesos donde las entradas y salidas son imágenes, también incluye procesos que extraen atributos de la imagen, hasta el reconocimiento de objetos individuales (Rafael C Gonzalez & Woods, 2002).

2.1.1.1 Tipos de imágenes

Se requieren de dos procesos para lograr digitalizar una imagen del mundo real, definida previamente como una función continua $f(x,y)$, son conocidos como el muestreo y la cuantificación. El proceso de muestreo es el encargado de definir los límites para los valores x e y en la función $f(x,y)$, o que es lo mismo, define el tamaño de la imagen digital, por otro lado el proceso de cuantificación delimita la amplitud de los valores para todos los componentes (x, y) del muestreo. La Figura 7 muestra un ejemplo del proceso de muestreo y cuantificación, a la izquierda, una imagen continua con valores que tienden a infinito tanto en dimensiones como en niveles de gris, a la derecha una imagen con un límite definido (muestreo) y un rango de valores en pixeles definidos (cuantificación) que van de blancos a negros con valores intermedios de gris.

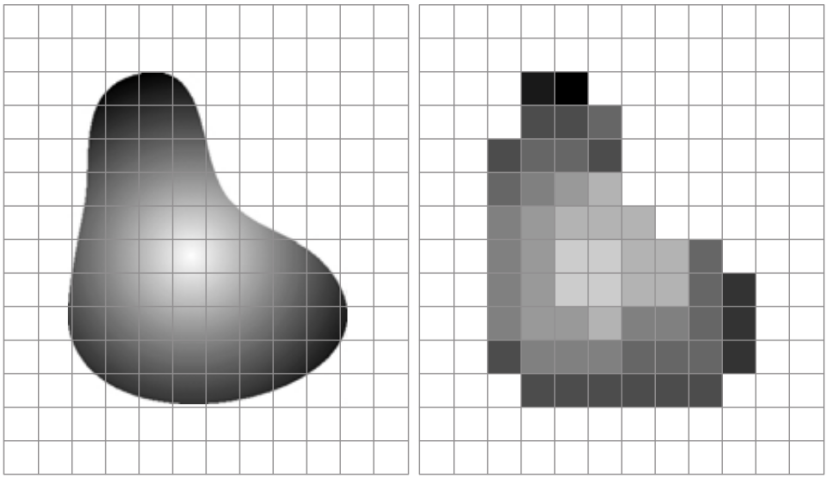

Figura 7 Ejemplo de muestreo y cuantificación.

El resultado del muestreo y la cuantificación es una matriz de números que generalmente son enteros de M filas y N columnas. Los valores de las coordenadas x e y en el origen por convención se toman como $(x,y)=(0,0)$, ubicadas en la esquina superior izquierda. El siguiente valor de coordenada en la primera fila de la imagen está representada como $(x,y) = (0,1)$. La Figura 8 muestra el sistema de coordenadas donde se representa una imagen digital.

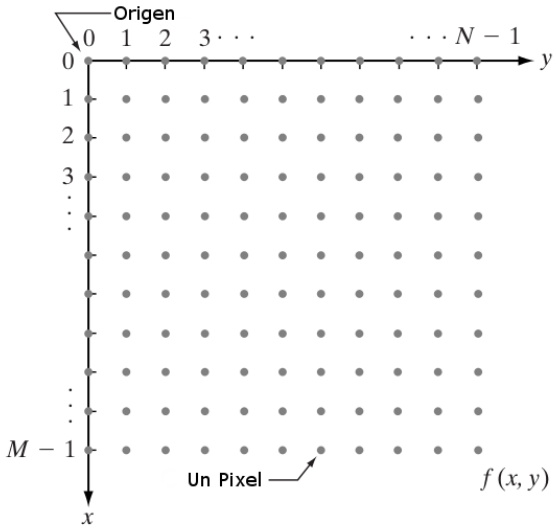

Figura 8 Representación de una imagen en un sistema de coordenadas.

Las notaciones introducidas en párrafos anteriores nos permiten escribir a la imagen digital completa MxN en la siguiente matriz compacta.

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{1}$$

Para el proceso de muestreo no existen requerimientos especiales para determinar los valores de M y N en una imagen digital, más que ser enteros positivos. Sin embargo, debido al procesamiento, almacenamiento, y las consideraciones de hardware para mostrar una imagen digital, se vuelve necesario que el proceso de cuantificación defina el conjunto de valores permitidos para cada pixel a partir de una potencia de dos, esta cuestión es definida en la ecuación (2), donde k es un entero positivo y representa la cantidad de bits necesarios para representar el rango de valores permitidos en un pixel.

$$L = 2^k \tag{2}$$

Se asume que los valores permitidos para cada pixel en una imagen se encuentran igualmente espaciados, que son enteros positivos en un intervalo $[0, L-1]$. Estas consideraciones y la ecuación (3) nos permiten calcular los bits b requeridos para almacenar una imagen digital.

$$b = M * N * k \tag{3}$$

Existen tres principales tipos de imágenes con los que generalmente se trabajan cuando se realiza procesamiento de imágenes, se pueden utilizar cualquiera de estos tres tipos de imágenes para realizar una segmentación, pero generalmente se comienza con una

imagen a color tomada del mundo real, después se obtiene una imagen monocromática (escala de grises) del cual se extrae una imagen binaria.

Imágenes monocromáticas

Estas imágenes son conocidas también como escalas de grises y es la que se ha estado describiendo en la introducción de este apartado, se trata de una función $f(x,y)$ en el plano, que permite visualizar una imagen digital con tonalidades que van de negro a blanco, estos tonos representan la intensidad de luz que fue capturada durante la digitalización, los valores permitidos en los pixeles de una imagen en escala de grises se puede calcular a partir de la ecuación (2) en donde k generalmente vale 8 o 16.

Rango de valores permitidos en un pixel para una imagen en escala de grises según el valor de k.

k = 8

$$L = 2^8$$
$$L = 256$$
$$Niveles\ de\ gris = [0, 256 - 1]$$
$$Niveles\ de\ gris = [0, 255]$$

Por convención y características de hardware utilizados al obtener y mostrar una imagen digital, un pixel con un valor igual a cero es visualizado completamente en negro y un pixel con un valor igual a 255 es visualizado en blanco. Los valores intermedios del rango de valores permitidos en un pixel son conocidos como niveles de grises que se degradan a negro si tienden a cero o a blanco si tienden a 255. La Figura 9 muestra a la izquierda una escena y a la derecha una representación digital de la escena en escala de grises y sus valores en cada pixel, suponiendo a $k = 8$.

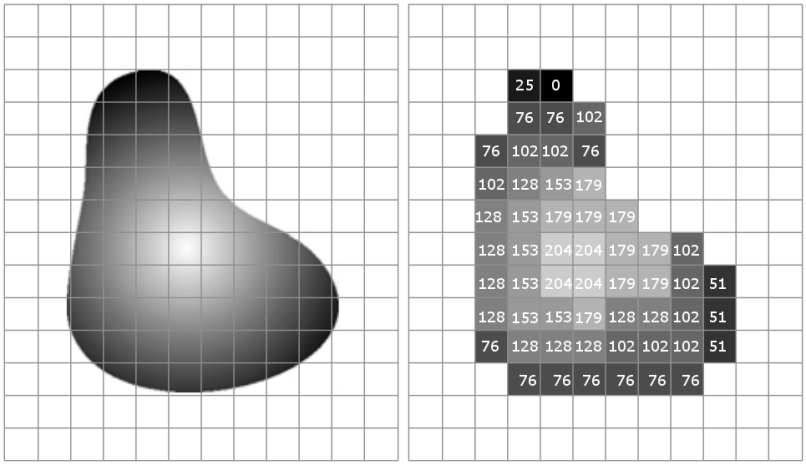

Figura 9 Imagen en escala de grises con una profundidad de 8 bits.

k = 16

$$Niveles\ de\ gris = 0 \rightarrow 65535$$

Cuando $k = 16$, un pixel con valor cero es visualizado como un pixel en color negro, cuando toma el máximo valor del rango permitido (65535) es visualizado en blanco. Mientras más grande el valor de k, mayor es el valor de los niveles de grises permitidos, es importante recordar que el rango de valores permitidos en un pixel se encuentra definido por el proceso de cuantificación y este a la vez lo define el dispositivo que digitaliza una imagen. El ojo humano no suele diferenciar más allá de unas cuantas docenas de escalas de grises en una imagen monocromática, por esta razón un rango de niveles de gris entre 0 y 255 suelen ser suficientes cuando se trabaja con procesamiento de imágenes. Aunque una imagen con mayor profundidad en bits supone mayor nivel de detalle de la escena digitalizada, esto mismo, implica una mayor carga computacional al momento de procesarlas.

Imágenes a color

En el sistema visual humano encontramos dos tipos receptores o “sensores” de luz conocidos como conos y bastones, los bastones son sensibles a bajos niveles de iluminación y los conos son los encargados de percibir colores. La luz reflejada de un objeto es absorbida en distintas cantidades por los conos. La interpretación de estas absorciones de luz en el sistema nervioso es la base de la percepción de color. Por lo tanto, el color es una representación perceptiva de la reflectancia de luz en la superficie de un objeto (Acharya & Ray, 2005). Como lo ilustra la Figura 10, la luz visible la compone una banda relativamente pequeña del espectro electromagnético. Un cuerpo que refleja la luz, el cual se encuentra balanceado en todas las longitudes de ondas visibles (refleja todas las ondas visibles del espectro electromagnético) se visualiza como un color blanco al observador. Sin embargo, un cuerpo que favorece la reflectancia en un limitado rango del espectro visible exhibe algunos tonos de color. Debido a las características de absorción de color en el ojo humano, los colores son vistos como una combinación variable de los denominados colores primarios rojo (R), verde (G), y azul (B).

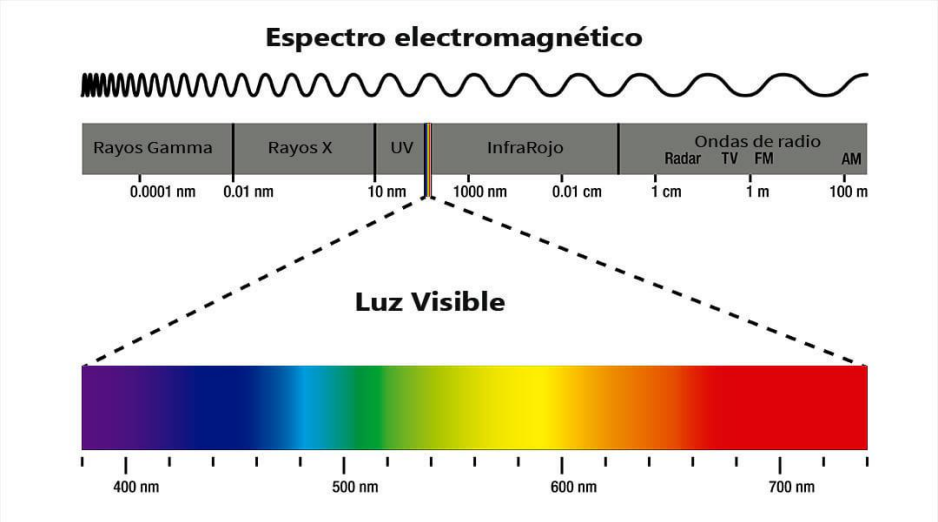

Figura 10 Rango visible del espectro electromagnético.

Los modelos de color (también llamados espacios de color o sistema de color) facilitan las especificaciones de los colores en un estándar, generalmente de una forma aceptada. En esencia un modelo de color es una especificación de un sistema coordenado y un subespacio dentro de ese sistema, donde cada color es representado por un solo punto. La mayoría de modelos de color están orientados hacia hardware (como son monitores e impresoras) o hacia aplicaciones que tienen como objetivo la manipulación de color (como lo es en la creación de gráficos a color para animación). En términos de procesamiento digital de imágenes, el modelo comúnmente utilizado en dispositivos de captura y visualización es el modelo RGB.

El modelo RGB

Para formar una imagen a color en el modelo RGB se requiere la combinación de 3 componentes, el componente R, se define como una función finita $f(x, y)$, en donde el valor de la función en un punto determinado (x, y) está definida por la cantidad de luz roja capturada de la escena, los componentes G (Verde) y B (Azul) comparten con el componente R la misma escena, el mismo tamaño y profundidad de bits, con la diferencia que solo capturan la intensidad de luz verde y azul, respectivamente.

Se ha mencionado anteriormente que el ser humano distingue solo unas cuantas docenas de escala de grises, sin embargo, es capaz de distinguir entre cientos de tonalidades de colores, las actuales capacidades de hardware para almacenar, procesar y mostrar imágenes a color, permiten definir en cada componente del modelo RGB una profundidad de 8, 16 o 32 bits. Como se ha visto anteriormente, con 8 bits se puede definir un rango de valores para cada pixel entre 0 y 255. Un pixel a color [el cual es una tripleta de valores (R, G y B)] se dice que tiene una profundidad de 24 bits, el número total de colores en una imagen RGB de 24 bits es $(2^8)^3$ = 16, 777, 216. La Figura 11 muestra una imagen a color RGB y sus componentes, cada componente muestra la cantidad de color capturado en su respectiva banda, se puede observar que en esta imagen la capa que aporta más información a la imagen es la capa R.

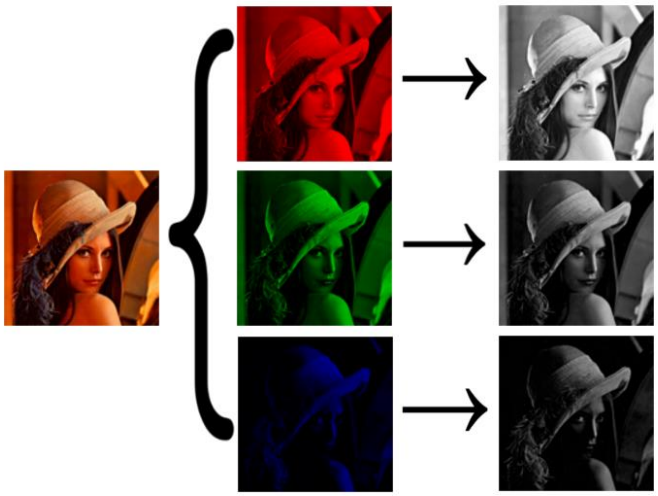

Figura 11 Una imagen a color RGB y sus componentes RGB.

Imágenes binarias

Las imágenes binarias son una reducción de imágenes monocromáticas o imágenes a color, en donde el rango de valores permitidos para cada elemento de la imagen son dos $[0,1]$, son comúnmente utilizadas para marcar regiones en la imagen en donde se tiene un interés, se toma un valor (generalmente 1) para marcar la región de interés y el otro valor para definir las regiones que no representa información útil.

La Figura 12 muestra a la derecha una imagen binaria generada a partir de una imagen a escala de grises (izquierda), para efectos en la impresión, los pixeles de interés se marcaron con el color negro [0] y el fondo en blanco [1], la cuadricula en escala de grises y los números solo son una representación esquemática de la división de los pixeles y el valor del pixel, no se tomaron en cuenta al binarizar la imagen.

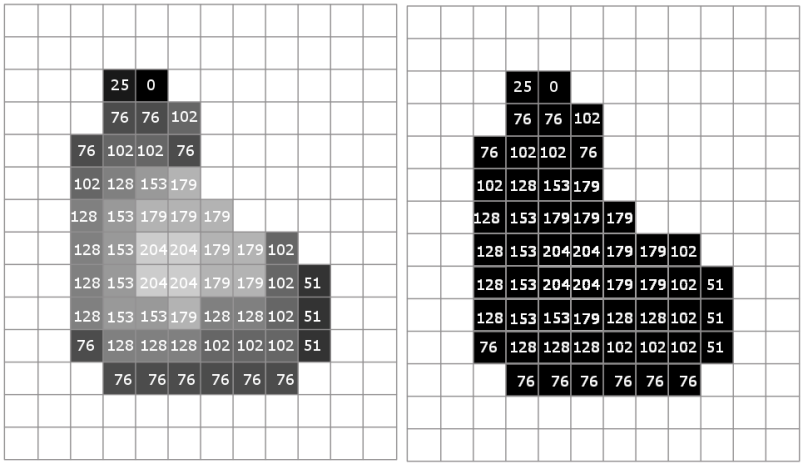

Figura 12 Imagen binaria de una imagen a escala de grises.

2.1.1.2 Formato de imágenes

Existe un número de formatos de archivos donde se pueden almacenar imágenes y recuperarlas. Estos son conocidos como formato de archivos estándar. A continuación, se presentan algunos de interés.

Tagged Image Format (.tif, .tiff): El formato .tif es un formato muy amplio, donde se puede manipular lo que sea, desde bitmaps hasta imágenes con una paleta de colores comprimida. El formato .tiff soporta muchos esquemas de compresión, pero también es utilizado para imágenes sin compresión, este formato es muy popular, relativamente simple y permite color.

Portable Network Graphics (.png): Este formato de archivo extensible provee una pérdida mínima, almacena con buena compresión, imágenes raster. Este simple formato cubre la mayor funcionalidad del formato .tiff. Escala de grises, paleta de colores y full-color son las imágenes soportadas por este formato de archivo. Soporta un canal opcional alfa, y la profundidad de 1 a 16 bits por canal.

JPEG (.jpg): Es el estándar de transmisión de información sobre imágenes más

ampliamente utilizado e incluye una variable perdida en la codificación como parte del standard, establecido por los parámetros de calidad.

2.1.1.3 Filtros y filtrado de una imagen digital

Al digitalizar una imagen pueden surgir diversos problemas que llegan a afectar la captura de la escena, antes de realizar una segmentación de imágenes será necesario minimizar o eliminar estos inconvenientes, existen varios enfoques que nos ayudan a mejorar la apariencia de una imagen, entre los que se utilizan con frecuencia se encuentran las técnicas de filtrado. A continuación, se describen brevemente algunas relaciones importantes entre pixeles, filtros y el proceso de filtrado.

Un pixel p en las coordenadas (x, y) tiene cuatro vecinos horizontales y verticales el cual sus coordenadas están dadas por la ecuación (4).

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1) \tag{4}$$

Este conjunto de pixeles, llamados 4-vecinos de p, esta denotado por $N_4(p)$. Los cuatro vecinos diagonales de p tienen las coordenadas descritas en la ecuación (5).

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1) \tag{5}$$

Y están denotados por $N_D(p)$. Los pixeles $N_D(p)$, junto con los 4-vecinos, son llamados 8-vecinos de p, denotado por $N_8(p)$. La Figura 13 muestra gráficamente los vecinos de un pixel p.

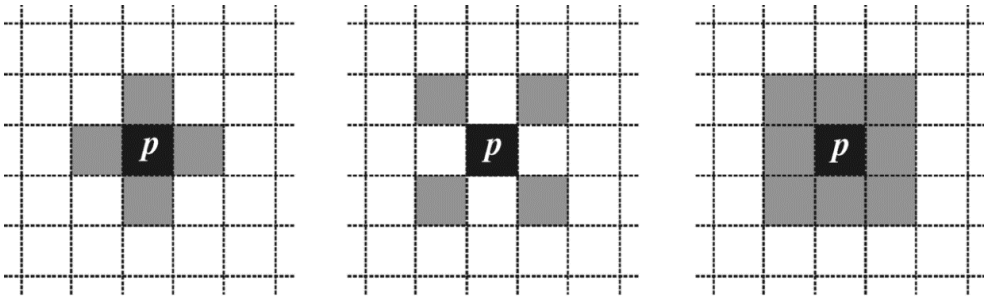

Figura 13 Vecinos $N_4(p), N_D(p)$ y $N_8(p)$ de un pixel p.

Los métodos de procesamiento que operan directamente sobre los pixeles de una imagen son denominados métodos en el dominio espacial. Los procesos en el dominio espacial se denotarán por la ecuación (6).

$$g(x, y) = T[f(x, y)] \tag{6}$$

Donde $f(x, y)$ es la imagen de entrada, $g(x, y)$ es la imagen procesada, y T es un operador sobre f, definido sobre algún vecindario de (x, y). La forma más simple de T es cuando el vecindario es de tamaño 1 x 1 (un solo pixel). En este caso, $g(x, y)$ depende

solo en el valor de f en (x,y), y T se vuelve una función de transformación de nivel de gris, las técnicas en esta categoría son referidas como procesamiento puntual.

Vecindarios más grandes permiten considerablemente más flexibilidad. El enfoque general es usar una función de los valores de f en un vecindario predefinido de (x,y) para determinar el valor de g en (x,y). Uno de los principales enfoques basados en esta formulación está basado en el uso de las denominadas máscaras (también referidas como filtros, kernels o ventanas). Básicamente, una máscara es un pequeño arreglo (digamos, 3x3) en 2-D, en donde el valor de los coeficientes de la máscara determina la naturaleza del proceso, estos procesos son conocidos como procesamiento por máscaras o filtrado. La Figura 14 muestra una imagen $f(x,y)$ con un filtro $w(s,t)$ de 3x3 en un punto (x,y).

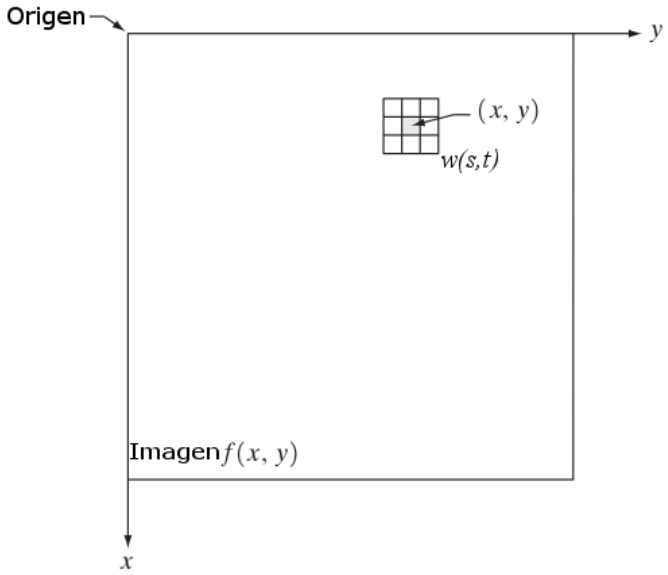

Figura 14 Una máscara o kernel 3x3 en el punto (x,y) en una imagen.

Los mecanismos de filtrado espacial están ilustrados en la Figura 15. El proceso consiste simplemente en ir moviendo la máscara de filtrado de punto en punto en una imagen. En cada punto (x,y), la respuesta del filtro en ese punto es calculado usando una relación predefinida. Para los denominados filtros lineales espaciales, la respuesta está dada por la suma de productos de los coeficientes del filtro y los correspondientes pixeles de la imagen en el área cubierta por la máscara de filtrado. Para la máscara 3x3 mostrada en la Figura 14, el resultado (o respuesta) R, del filtrado lineal con la máscara de filtro en un punto (x,y) en la imagen se representa con la ecuación (7).

$$\begin{aligned} R &= w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots \\ &+ w(0,0)f(x,y) + \ldots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \end{aligned} \tag{7}$$

Note en particular el coeficiente de la máscara $w(0,0)$ coincide con el valor de la imagen

$f(x,y)$, indicando que la máscara está centrada en (x,y) cuando el cálculo de la suma de productos toma lugar. Por la naturaleza del filtrado las máscaras generalmente deben tener un tamaño en pixeles impar.

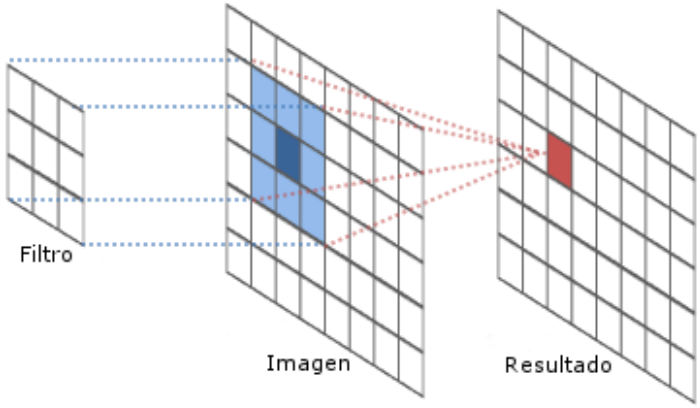

Figura 15 Representación gráfica del filtrado espacial.

En general, el filtrado lineal de una imagen f de tamaño MxN con una máscara de filtrado de tamaño mxn está dada por la ecuación (8).

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{8}$$

Donde, $a = (m-1)/2$ y $b = (n-1)/2$, para generar una imagen filtrada completa esta ecuación deberá ser aplicada para $x = 0,1,2, \ldots, M-1$ y $y = 0,1,2, \ldots, N-1$, de esta manera nos aseguramos que la máscara procese todos los pixeles en la imagen.

Una consideración importante en la implementación de operaciones de vecindad para filtrado espacial es el problema de qué pasa cuando el centro del filtro se enfoca en el borde de la imagen, si el centro de la máscara se mueve sobre un elemento de la imagen en el borde, uno o más filas o columnas de la máscara se localizará fuera del plano de la imagen. La forma más simple de manejar esta situación es restringir el procesamiento a una porción de la imagen donde no se originen conflictos. Esto deja líneas de pixeles sin procesar a través de los bordes de la imagen, iguales a la amplitud del radio de la máscara. La imagen filtrada resultante será más pequeña que la original, pero todos los pixeles en la imagen filtrada habrán sido procesados completamente por la máscara. Si el proceso requiere que la imagen sea del mismo tamaño que la original, el enfoque típicamente empleado es filtrar todos los pixeles solo con la sección de la máscara que esté completamente contenida en la imagen. Con este enfoque, habrá bandas de pixeles cerca del borde que habrán sido procesadas parcialmente con la máscara de filtrado, se pueden implementar otras opciones como extender el borde de la imagen duplicando los pixeles del borde o rellenar el borde con unos o ceros.

La Figura 16 nos muestra un ejemplo esquemático del problema que genera el filtrado espacial.

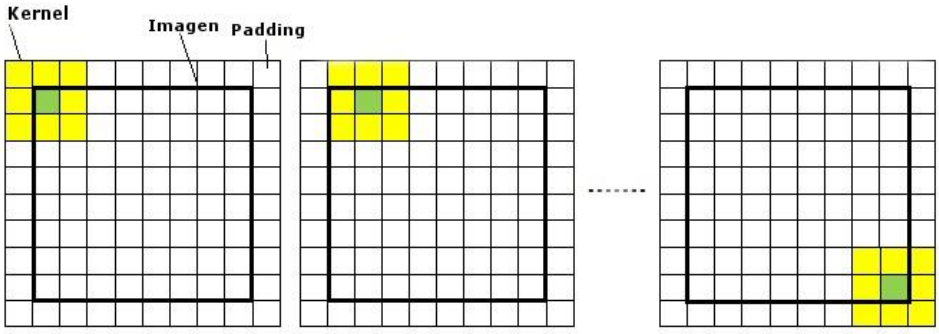

Figura 16 Para poder procesar los pixeles en el borde durante el filtrado se puede aplicar un rellenado de unos o ceros, o limitar el proceso a una porción de la imagen.

2.1.2 Segmentación de imágenes

Los filtros aplicados sobre una imagen no solo minimizan información irrelevante en la imagen, también existen filtros que acentúan algunas características deseables. Todo esto para que en pasos posteriores sea más sencillo el aislamiento y análisis de las características inherentes a los componentes de la imagen. Los procesos que realizan la separación de los componentes de una imagen digital en regiones separadas como objetos independientes, son conocidos como métodos de segmentación. El nivel de separación que van a tener los objetos depende del problema que se está resolviendo, es decir, la segmentación deberá detenerse cuando los objetos de interés en una aplicación se han identificado satisfactoriamente (Rafael C Gonzalez & Woods, 2002).

Se pueden estudiar a los métodos de segmentación en dos principales enfoques, la segmentación de imágenes basada en regiones y la segmentación basada en contornos. En el enfoque basado en regiones, se considera a cada pixel en la imagen y se asigna a una región en particular u objeto basada en la similitud entre pixeles. En el enfoque basado en contornos se particiona una imagen basado en cambios abruptos de niveles de gris, se intenta localizar directamente los contornos que existen entre las regiones o se busca encontrar los pixeles de los bordes y enlazarlos junto con otros pixeles bordes hasta establecer los contornos requeridos. El resultado de ambos enfoques podría no ser el mismo en términos de regiones aislados, pero pueden ser útiles para entender y resolver los problemas de segmentación de imágenes, la combinación de ambos podría significar un mejor desempeño en la tarea de segmentación. La decisión sobre que enfoque utilizar depende generalmente de la naturaleza del problema a resolver.

Sin importar el enfoque utilizado, la mayor parte de algoritmos de segmentación reciben como entrada imágenes en escala de grises y arrojan como resultado una imagen binaria. Como se ha descrito anteriormente, una imagen binaria es muy apropiado para la representación de objetos segmentados, ya que permite asignar un valor de la imagen binaria para representar información de interés (región o contorno) y el otro valor para representar información irrelevante. Además, las imágenes binarias tienen un bajo costo computacional en el procesamiento y almacenamiento.

2.1.2.1 Segmentación de imágenes basada en regiones

Las principales limitaciones de los métodos basados en contornos, es que la información que ofrecen de los objetos de interés es superflua, las técnicas para segmentar bordes muchas veces ignoran la información global de la imagen y el procesamiento se realiza solo en los pixeles de un vecindario predefinido, las imágenes segmentadas suelen requerir, después del proceso de segmentación otro proceso de unión de bordes para encontrar contornos, que tiene un costo computacional elevado, además el proceso de unión de bordes da lugar a discontinuidades y vacíos en la imagen, este error se trata de enmendar con una interpolación arbitraria para a completar los contornos (Acharya & Ray, 2005).

Los métodos de segmentación por regiones y en específico los métodos de segmentación por umbralización son perfectos cuando la escena de una imagen digital tiene un fondo uniforme y existe un aceptable contraste entre el fondo y los objetos a segmentar, como es el caso de las imágenes generadas con un microscopio. Los algoritmos de segmentación por umbralización, además de ser métodos relativamente poco complejos, la mayoría no requieren pesados procesos post-segmentación y no demandan mucho tiempo de procesamiento durante la segmentación como lo requieren otros métodos de segmentación basado en regiones (crecimiento de regiones y división y fusión de regiones).

2.1.2.2 Técnicas de umbralización de imágenes

La umbralización es una técnica esencial basada en regiones, la operación es usada para distinguir entre objetos de interés y el fondo donde se encuentran. El resultado de esta operación es una imagen en donde cada pixel pertenece a una de dos regiones “objeto” o “fondo”. En general, una operación de umbralización en nivel de gris puede ser descrita según la ecuación (9).

$$g(x,y) = \begin{cases} 1 \leftarrow f(x,y) > T \\ 0 \leftarrow f(x,y) \leq T \end{cases} \tag{9}$$

Donde $f(x,y)$ es el nivel de gris en el punto (x,y) de la imagen, T es el umbral y $g(x,y)$ es la imagen umbralizada. Cuando el valor de gris $f(x,y)$ supera el valor umbral, al pixel en la misma posición de la imagen $g(x,y)$ recibe el valor 1 (o cualquier otro conveniente nivel de gris) correspondiente a los objetos, en caso contrario, recibe el valor de 0 (o cualquier otro nivel de gris no asignado a los objetos) correspondiente al fondo.

Histograma

Gran parte de los algoritmos de segmentación por umbralización utilizan el histograma de la imagen en escala de grises para seleccionar un apropiado valor para T. El histograma de una imagen digital con niveles de gris en el rango [0, L-1] es una función discreta $h(r_k) = n_k$, donde r_k es el *k*-ésimo nivel de gris y n_k es el número de pixeles en la imagen que tiene el nivel de gris r_k.

La Figura 17 muestra a la izquierda una imagen en escala de grises y su correspondiente histograma a la derecha.

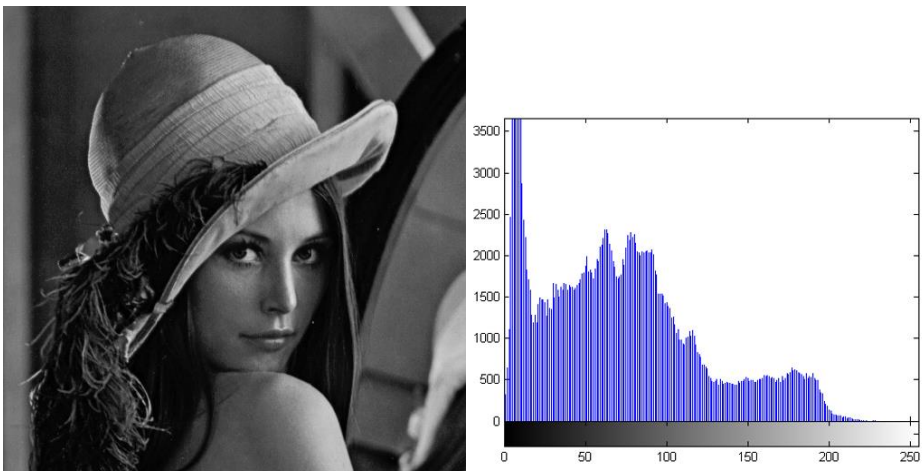

Figura 17 Imagen en escala de grises y su histograma.

Umbralización Global

En la más simple implementación del método de umbralización, el valor del nivel de gris umbral T se mantiene constante a través de la imagen. Si el nivel de gris del fondo es razonablemente uniforme en la imagen y si todos los objetos tienen aproximadamente el mismo contraste sobre el fondo, entonces el histograma de la imagen en escala de grises es bimodal, y un valor umbral constante funciona muy bien cuando se aplica a todos los pixeles de la imagen. En general la elección del valor umbral T, tiene efectos considerables en la posición del contorno y el tamaño de los objetos segmentados. Por esta razón el valor umbral deberá ser determinado cuidadosamente.

Umbralización dinámica o adaptativa

En regiones donde la iluminación de la escena no es lo suficientemente uniforme, el enfoque de umbralizado global no suele tener buenos resultados, ya que mientras en algunas regiones la segmentación divide correctamente el fondo de los objetos, el mismo valor de umbral no funcionaría para otras regiones con niveles de iluminación diferente, por este motivo se busca un valor umbral basado en un predefinido vecindario de un punto (x, y), es decir que el valor de T es calculado por cada pixel en la imagen. A la operación de buscar un valor umbral basado en estadísticas de la imagen $f(x, y)$ como la media o la desviación estándar de valores de grises, así como estadísticas en el vecindario $w(m, n)$ de cada pixel, es llamada umbralización adaptativa.

2.1.2.3 Regiones y sus propiedades geométricas

Una vez finalizada la tarea de segmentación en donde se han separado los objetos de la imagen, del fondo que los contiene, se procede a obtener parámetros que describan a los objetos, estos parámetros son conocidos como descriptores. Los descriptores que se van a obtener de las regiones segmentadas son dependientes de la aplicación. Puede ser tan simple, como proveer una medida de la morfología del objeto o estructura definiendo sus

propiedades en términos de área, perímetro, intensidad, color, forma, entre otros.(Wu, Merchant, & Kenneth, 2008).

Conteo y etiquetado

Es muy común que la segmentación por umbralización termine con una imagen binaria, en donde se asigna el valor de 1 a los pixeles que pertenecen a los objetos y se les asigna 0 a los que no pertenecen a ningún objeto. El proceso de conteo y etiquetado tienen como finalidad identificar la cantidad de componentes conectados y los pixeles que forman parte de cada componente conectado.

Componente conectado: Al conjunto de pixeles que mantienen una relación de conectividad (4 u 8 conectado) se les llama componentes conectados, también denominados objetos o regiones. Dos pixeles p y q forman parte del mismo componente conectado C si existe una secuencia de 1-pixeles $(p_0, p_i, \ldots, p_n)$ de C donde $p_0 = p$, $p_n = q$ y p_i es un vecino de $p_i - 1$ para $i = 1, \ldots, n$.

4-conectado: Dos pixeles p y q se dice que son 4-conectado si $q \in N_4(p)$ Siendo $N_4(p)$ los cuatro vecinos del pixel p.

8-conectado: De igual manera, dos pixeles p y q se dice que son 8-conectado si $q \in N_8(p)$, donde $N_8(p)$ son los vecinos $N_4 p + N_D p$, la Figura 18 muestra esquemáticamente la representación de la conectividad entre pixeles.

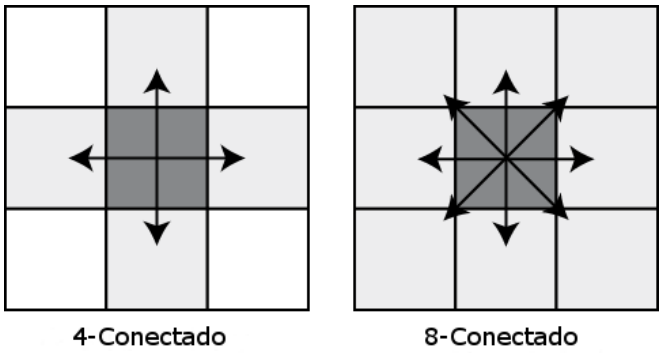

Figura 18 Conectividad 4-conectado y 8-conectado.

Los algoritmos de etiquetado reciben como entrada imágenes binarias y dan como resultado regiones con los pixeles que lo conforman plenamente identificados, esta es una tarea sumamente sencilla cuando se tienen componentes conectados definidos.

Descriptores

Posterior al conteo y etiquetado se analizan individualmente estas regiones para extraer sus descriptores. A continuación de describen algunos descriptores importantes que pueden ser obtenidas en una región de una imagen binaria.

***Área de un objeto*:** El área de un objeto está dado por el número total de pixeles en el componente conectado, esta relación está definida en la ecuación (10).

$$A = \sum_i \sum_j O[i, j] \tag{10}$$

Donde $O[i, j]$ representa los pixeles que conforman la región conectada.

***Perímetro de un objeto*:** Para calcular el perímetro de un objeto, nosotros identificamos los pixeles del borde del objeto cubriendo un área, el perímetro está definido como la suma de los pixeles en el borde del objeto.

La Figura 19 muestra a la izquierda un componente conectado y a la derecha el perímetro del objeto.

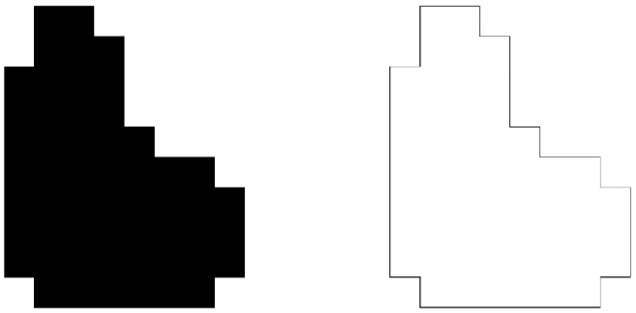

Figura 19 Perímetro de un objeto.

Rectángulo mínimo: El rectángulo mínimo o BoundingBox es el rectángulo más pequeño que puede contener completamente a toda la región, está definido por dos puntos extremos, que son:

1.- El punto límite (x1, y1) que es el mínimo valor de M_i, N_j que contiene a la región en la parte superior izquierda.

2.- El punto límite (x2, y2) que es el máximo valor de M_i' y N_j' que engloba a la región completamente.

La Figura 20 muestra la definición del rectángulo mínimo en dos imágenes, a la izquierda se observa el rectángulo mínimo, donde varios pixeles del borde de la región se alinean sobre un solo punto M_i (x1) y N_j (y1), a la derecha se muestra la misma región, pero con una pendiente, en esta región solo un pixel define el valor para (x1, y1) así como para (x2, y2).

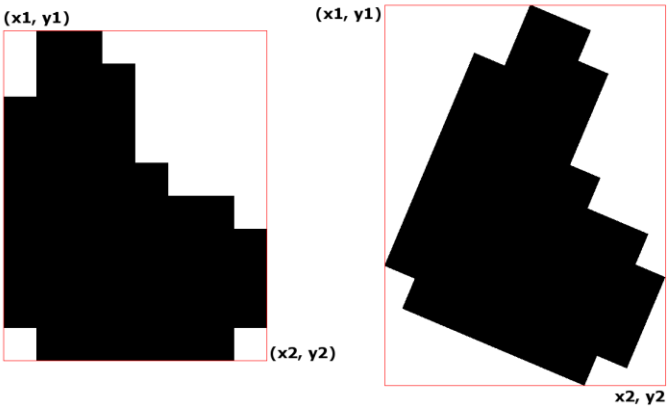

Figura 20 Rectángulo mínimo de una región.

***Centroide*:** El centroide de una región es la localización promedio de los pixeles que pertenecen a la región, se calcula la localización promedio de todos los elementos x de la región y todos los elementos y . Nótese que los resultados para el par de elementos $(\bar{x}, \bar{y})$ generalmente no resultan cantidades enteras por lo que será necesario redondear los elementos cuando sea requerido y poder ser asignados a un par de coordenadas validas en la imagen. La ecuación (11) define el cálculo del centroide para una región R.

$$\bar{x} = \frac{1}{A}\sum (x, y) \in Rx$$
$$\bar{y} = \frac{1}{A}\sum (x, y) \in Ry \qquad (11)$$

Donde A es el área de la región, (x, y) es un pixel que pertenece a la región, Rx y Ry son conjuntos que contienen coordenadas x e y respectivamente de todos los pixeles de la región.

***Proporción de aspecto*:** Es la proporción del alto de la forma dividido entre su longitud, la ecuación (12) define la proporción.

$$HRatio = \frac{AlturaBBox}{LongitudBBox} \qquad (12)$$

Donde AlturaBBox es la altura del rectángulo mínimo de la región y LongitudBBox es la longitud.

Cerco convexo: ConvexHull en inglés, se define como la forma convexa más pequeña que contiene a la región (puede ser visto como la forma de una banda elástica puesta alrededor de la forma original). La deficiencia convexa (área clara dentro de la línea roja de la Figura 21) es definida como la región que debe ser agregada a la región dada para crear el cerco convexo (Davies, 2012).

La Figura 21 Muestra en rojo el cerco convexo de la región.

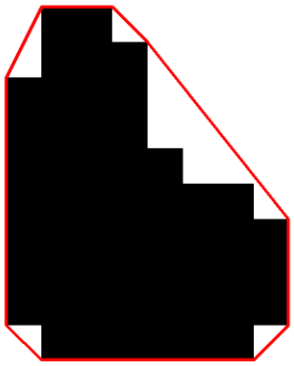

Figura 21 Cerco convexo de una región.

Solidez: *T*ambién conocida como convexidad, se define como la proporción del área de la región entre el área del cerco convexo, se utiliza para medir las intrusiones irregulares a la región con respecto de su cerco convexo, la ecuación (13) define esta relación.

$$Solidez = \frac{A_r}{A_{ConvexHull}} \tag{13}$$

Donde $A_{ConvexHull}$ es el área del cerco convexo y A_r es el área de la región.

Centroide Frontal: Es el valor $\bar{x}$ de la región, pero esta vez no se toman en cuenta todos los pixeles de la región para calcularlo, sino que se toma cierto porcentaje de la imagen para obtener el centroide, ya que el centroide se especifica como frontal, el porcentaje de los pixeles que se utilizan se encuentran más cercanos al origen (0,0) de la imagen y se extienden hacia la derecha de la imagen abarcando todas las filas. La Figura 22, muestra la parte de la región que es tomada para obtener el centroide frontal (color azul, aproximadamente un 30% de ancho de la imagen), la línea roja es el valor $\bar{x}$ para esta imagen que se extiende para cada columna que se ha utilizado para calcularla.

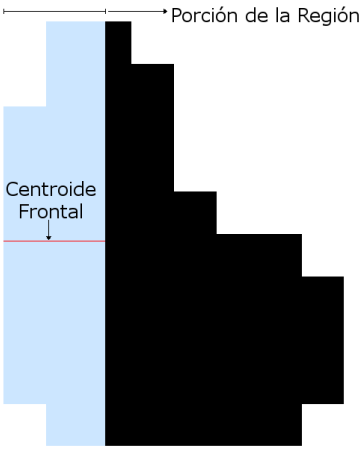

Figura 22 Centroide Frontal Vertical.

Diferencia con la línea central: CLD por sus siglas en inglés, es la diferencia entre el Centroide Frontal y la mitad de la altura del rectángulo mínimo, el resultado de la resta se normaliza dividiendo el valor absoluto de esta diferencia entre la altura total del rectángulo mínimo, este descriptor se define en la ecuación (14).

$$CLD = \frac{\left| CF - (HBoundingBox / 2) \right|}{HBoundingBox} \tag{14}$$

Donde CF es el Centroide Frontal, HBoundingBox es la altura del rectángulo mínimo. La normalización nos permite definir la diferencia entre un valor de 0 a ≈ 0.5 donde 0 indica que CF se encuentra exactamente a la mitad de la altura del BoundingBox y $\approx$ 0.5 indica que el CF se encuentra por debajo o sobre la mitad de la altura del BoudingBox.

Proporción de Asimetría: La proporción de asimetría permite medir la alineación de una región a lo largo de una fila predefinida. Se obtiene promediando los valores de asimetría por cada columna que conforma la región. El valor de asimetría por columna se obtiene de la división del valor absoluto de la diferencia de los elementos sobre la fila umbral y bajo la fila umbral, entre la cantidad de pixeles que pertenecen a la región en esa columna.

Se define a una región completamente simétrica si tiene exactamente la misma cantidad de pixeles sobre y bajo el eje horizontal umbral, cuando una región presenta estas características el valor de proporción de asimetría es cero y tiende a uno si la cantidad de pixeles de la región se encuentran más sobre o bajo el eje horizontal umbral.

La proporción de asimetría se obtiene a partir de la ecuación (15).

$$pAsimetria = \frac{1}{N-1} \sum_{j=0}^{(N-1)} \left(\frac{\left| \sum_{i>CF} p(i,j) - \sum_{i<=CF} p(i,j) \right|}{\sum_{i=0}^{(M-1)} p(i,j)} \right) \tag{15}$$

Donde $p(i,j)$ es el valor del pixel en la posición i, j, se asume que la región es una imagen binaria donde los pixeles que pertenecen a la región tienen un valor de 1 y tienen un valor de 0 en caso contrario, CF es el Centroide Frontal de la región.

2.1.3 Reconocimiento de patrones

Para nosotros los humanos identificar patrones parece ser una tarea trivial, podríamos predecir por ejemplo si un día será lluvioso o no en función de la cantidad de nubes que hay en el cielo, la estación del año en la que nos encontremos y la temperatura ambiental a la que estamos. Es decir, basados en estos parámetros y la experiencia (días previos lluviosos), podemos asociar esos datos a algún estado específico del clima, en otras palabras, hemos aprendido a predecir el clima. Se define al aprendizaje como el conjunto de experiencias vividas previamente y del cual nos podemos basar para tomar decisiones.

El reconocimiento de patrones por maquinas involucra técnicas para asignar patrones a

sus respectivas clases, automáticamente y con la menor intervención humana como sea posible, en (Bezdek, 2013), se define el reconocimiento de patrones como “La búsqueda de una estructura en los datos”. Una descripción general sobre reconocimiento de patrones es que trata sobre análisis de características, agrupamiento o clustering y diseño de clasificadores.

2.1.3.1 Patrón

Un patrón es un arreglo de descriptores o características de algún proceso físico, que en la práctica son vistos como vectores. Los vectores están representados por letras minúsculas en negritas como **x**, **y**, y **z,** tomando la forma de la ecuación (16).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \tag{16}$$

Donde cada componente x_i , representa el *i*-ésimo descriptor y *n* es el número total de esos descriptores asociados con el patrón. Los patrones vector pueden ser representados como columnas en la forma de $n \times 1$ matrices o en filas de forma $1xn$ matrices. Podemos definir a un patrón para nuestro ejemplo de clima como:

$$\mathbf{x} = \begin{bmatrix} 98 \\ 30 \\ 7 \end{bmatrix}$$

Donde:

98 podríamos representar el porcentaje del cielo cubierto por nubes.

30 representa la temperatura ambiental.

7 representa el número del mes (julio).

2.1.3.2 Clase

Es una familia de patrones que comparten algunas propiedades en común. Las clases de los patrones se denotan como $c_1, c_2, \ldots, c_m$ donde m es el número de clases. Para nuestro ejemplo del clima podemos definir dos clases: c_1 = lluvioso, c_2 = no lluvioso.

Para que nuestro método de reconocimiento de patrones sea capaz de predecir si el día de hoy será un día lluvioso o no, se pueden manejar dos enfoques: basados en el aprendizaje supervisado y aprendizaje no supervisado.

En el enfoque supervisado, es necesario que previamente ya hayan existido patrones a los cuales con ciertas características hayan sido etiquetados como días lluviosos o no lluviosos, a este conjunto de patrones se le conoce como conjunto de entrenamiento, el conjunto de entrenamiento define a un método de clasificación como supervisado.

Entonces, si mantenemos este ejemplo, predecir el estado del clima, es muy probable que basado en los datos provistos por el patrón de ejemplo, este día se oriente más a un día lluvioso, ya que las características serán similares o estará más “cerca” de los patrones del conjunto de entrenamiento que ya fueron etiquetados como días lluviosos.

En el enfoque no supervisado no existe un conjunto de entrenamiento, lo que significa que los patrones por si mismos tendrán que encontrar la manera de agrupare a sí mismos en estados de clima previamente definidos (lluvioso o no) o generar nuevos agrupamientos, nuevas etiquetas o clases. Los estudios en el área de reconocimiento de patrones se enfocan en crear algoritmos de aprendizaje que sean capaces de etiquetar lo más exactamente posible un patrón a una determinada clase.

2.1.3.3 Aprendizaje no supervisado o Clustering

(Duda, Hart, & Stork, 2001), define cinco razones básicas por la cual el aprendizaje no supervisado tiene un gran interés.

1.- Coleccionar y etiquetar a un conjunto grande de patrones de muestra (conjunto de entrenamiento) puede ser asombrosamente costoso en tiempo.

2.- Uno podría desear proceder en la dirección inversa: entrenar con un gran conjunto de datos sin etiquetar y solo entonces usar la supervisión para etiquetar a los grupos encontrados. Esto puede ser apropiado en grandes aplicaciones de minería de datos, en donde el contenido de una base de datos grande no se conoce de antemano.

3.- En muchas aplicaciones las características de los patrones pueden cambiar un poco con el tiempo, si estos cambios pueden ser seguidos por un clasificador corriendo en modo sin supervisión, puede lograr un mejor rendimiento.

4.- Podemos utilizar métodos sin supervisión para encontrar características, que serán útiles para la categorización.

5.- En un estado temprano de una investigación podría ser valiosa tener una idea de la naturaleza o estructura de los datos. El descubrimiento de subclases distintas o similitudes entre los patrones o de mayores separaciones entre características de las esperadas podría sugerirnos significantemente cambiar nuestro enfoque al diseñar el clasificador.

K-Means

A pesar de que K-Means fue propuesto hace más de 50 años, se mantiene como uno de los algoritmos más ampliamente usados para clustering. La fácil implementación, simplicidad, eficiencia y el éxito empírico, son las principales razones de su popularidad (Jain, 2010).

El algoritmo K-Means particiona un conjunto de patrones de entrada X en un conjunto de K particiones, donde K es una cantidad conocida de antemano. El método está basado en la identificación de los centroides para cada K clusters (Acharya & Ray, 2005). El método se reduce esencialmente en la búsqueda del mejor conjunto de K centroides de los clusters con el algoritmo que se describe a continuación.

Algoritmo K-Means

Paso 1: Seleccionar los K centros iniciales de los clusters $C1, C2, \ldots, CK$.

Paso 2: Por cada patrón, calcular la distancia entre dicho patrón y cada uno de los K centroides propuestos en el paso anterior.

Paso 3: Cada patrón es agrupado en el cluster $C_i (1 \leq i \leq K)$ cuyo centroide sea el más cercano.

Paso 4: Volver a calcular los centroides $C_j (1 \leq j \leq K)$ en cada cluster basado en el valor promedio de todos los patrones asignados a dicho cluster.

Paso 5: Repetir los pasos 2 a 4 hasta cumplir una condición de terminación.

En este algoritmo es importante remarcar ciertos parámetros que deben ser seleccionados arbitrariamente:

La configuración de los K clusters iniciales se puede seleccionar una de las siguientes opciones:

- Seleccionar los primeros K patrones del conjunto de patrones X como clusters iniciales.
- Selección aleatoria de K patrones del conjunto de patrones X como clusters iniciales.
- El algoritmo k-means++ propuesto por (Arthur & Vassilvitskii, 2007).

Distancia entre patrones y centroides:

- Distancia Euclidiana
- Distancia Manhattan o Cityblock

El método para volver a calcular los clusters

- El promedio de los patrones asignados a cada centroide.

Función objetivo o condición de terminación.

- Que las asignaciones a los clusters no cambien.
- Se alcance un número máximo de iteraciones.

Fuzzy C-Means

Fuzzy C-Means (FCM), es una extensión del algoritmo de clustering K-Means donde cada patrón es miembro de múltiples clusters con cierto valor de pertenencia. El algoritmo de agrupamiento Fuzzy C-Means, produce muy buenos resultados en el agrupamiento de

regiones en imágenes y clasificación entre objetos. Como en el algoritmo puro K-Means, el algoritmo Fuzzy C-Means está basado en la minimización de una función criterio definida en la ecuación (17).

$$J_m = \sum_{i=1}^{D} \sum_{j=1}^{N} \mu_{ij}^{m} \left\| x_i - c_j \right\|^2 \tag{17}$$

Donde:

- D es el número de patrones
- N es el número de clusters
- $m \in [1, \infty]$ es un exponente de factor de peso. No hay una regla establecida para elegir el factor de peso en el exponente. Sin embargo, en muchas aplicaciones es una elección común que m sea igual a 2.
- x_i es el i-ésimo patrón
- c_j es el centro del j-ésimo cluster
- μ_{ij} es el grado de pertenencia de x_i en el j-ésimo cluster. Para dado un patrón x_i, la suma de los grados pertenencia para todos los clusters es uno.

Algoritmo Fuzzy C-means

Paso 1: Inicializa aleatoriamente el cluster de los valores de pertenencia μ_{ij}.

Paso 2: Calcula los centros de los clusters con la ecuación (18):

$$c_j = \frac{\sum_{i=1}^{D} \mu_{ij}^{m} xi}{\sum_{i=1}^{D} \mu_{ij}^{m}} \tag{18}$$

Paso 3: Actualizar μ_{ij} de acuerdo con la ecuación (19):

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{N} \left(\frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_k \right\|} \right)^{\frac{2}{m-1}}} \tag{19}$$

Paso 4: Calcula la función objetivo J_m .

Paso 5: Repetir los pasos 2-4 hasta que el valor de J_m sea menor que un umbral mínimo especificado desde la última iteración o hasta que un máximo número de iteraciones especifica haya sido llevado a cabo.

2.2 Estado del arte

En la revisión de la bibliografía se encontraron con diversas propuestas para la segmentación de imágenes del ensayo cometa, a continuación, se describen brevemente los más relevantes.

2.2.1 Automated measurements of tails in the SCGE assay

Este desarrollo fue hecho por J-F. Rivest y colaboradores en 1996, es uno de los primeros intentos por automatizar la obtención de métricas de las imágenes del ensayo cometa.
Utiliza técnicas de morfología matemática para eliminar componentes remanentes de la electroforesis que puede llegar a causar ruido en las imágenes, también implementa la transformada watershed para localizar los bordes potenciales del cometa (Rivest, Tang, McLean, & Johnson, 1996).
Esta metodología propuesta no es automática, puesto que, para empezar el usuario debe marcar manualmente la localización de cada cometa y el fondo que los contiene para poder diferenciar entre cometa y fondo basado en el contorno del cometa

2.2.2 Segmentation of the Comet Assay Images

Este trabajo de investigación fue presentado en 2004 en la conferencia internacional de análisis y reconocimiento de imágenes (ICIAR por sus siglas en ingles), el autor Bogdan Smolka (Smolka & Lukac, 2004) presenta 3 nuevas formas de segmentar imágenes del ensayo, mismas que son listadas a continuación.

1. *Enfoque probabilístico e iterativo*

Utiliza el modelo de distribución de Gibbs para determinar la vecindad de pixeles, y junto con la realización de campos aleatorios de Markov aplicados iterativamente en la imagen logra una segmentación del núcleo y cola del cometa.

2. *Segmentación basada en regiones*

Se basa en el método de segmentación propuesto por Henryk Palus y Damian Bereska en 1999, con la diferencia que el método de segmentación fue aplicado a imágenes en escala de grises. El método va agregando pixeles a ciertas regiones de la imagen con una vecindad de 4-conectado, si el valor de brillo de los pixeles se encuentra en un rango aceptable dentro del valor de brillo promedio de la región ya existente.

3. *Segmentación por contornos activos*

Se basa en curvas deformables bajo influencia de fuerzas externas e internas.

Aunque los métodos definidos por el autor segmentan apropiadamente, la cabeza y cola de los cometas, carece de un método automático para excluir cometas que no sean aptos para el análisis argumentado por la falta de la estandarización de la técnica del ensayo.

2.2.3 Automated segmentation of the comet assay images using Gaussian filtering and fuzzy clustering

Este es uno de los métodos más novedosos desarrollado por Mario Sansone en 2012 que ya se acerca más a la automatización ya que trata de minimizar la intervención del usuario durante el proceso. El algoritmo se divide en dos apartados: a) la identificación de cometas

por un pre-filtro Gaussiano y operadores morfológicos; b) segmentación de cometas por agrupamiento difuso (Sansone, Zeni, & Esposito, 2012).
El algoritmo utilizado para el agrupamiento de pixeles en regiones del cometa es considerado como uno de los más costosos computacionalmente, aparte, se observa que no se explota del todo la principal virtud de un algoritmo con un enfoque difuso, que son los grados de pertenencia que tiene cada patrón con las clases. Por lo cual, se plantea en el presente trabajo de investigación utilizar un algoritmo diferente para la segmentación de regiones con un costo en tiempo y computacional reducido.

2.2.4 OpenComet: An automated tool for comet assay image analysis Software

Benjamin M. Gyori, en 2014 desarrolló una herramienta de código abierto que realiza un análisis automático de las imágenes del ensayo cometa. Utiliza un nuevo y robusto método para encontrar cometas basado en los atributos geométricos de su forma y segmenta la cabeza a través del análisis del perfil de intensidad de los cometas. OpenComet es propuesta como una herramienta más exacta, rápida y que minimiza los errores humanos. Validado en las versiones neutras y alcalinas del ensayo cometa (Gyori, Venkatachalam, Thiagarajan, Hsu, & Clement, 2014).

Los descriptores de regiones utilizados para realizar el proceso de selección de regiones como cometas viables a ser analizados parecen tener un buen desempeño, sin embargo, se considera que el método utilizado en el preprocesamiento para obtener la imagen en escala de grises, requerida en el proceso de binarizacion, tiene un principal inconveniente, selecciona solo un canal de la imagen a color RGB como una representación de todo el contenido de la imagen, asumen que toda la información requerida para una exitosa segmentación está contenida en el canal que tenga el mayor valor promedio de brillo.

Tras experimentación con imágenes obtenidas de distinta naturaleza y método de tinción en el ensayo, este enfoque parece no ser el correcto, debido a que llega a haber imágenes que contienen información de brillo relevante para la segmentación distribuidas en los tres componentes de la imagen RGB, tomar solo un canal significa una pérdida de información imprescindible para la correcta implementación del método de binarización utilizado y consecuentemente esto afecta los pasos sucesivos del modelo propuesto.

2.2.5 Software Open Source para el análisis de imágenes del ensayo cometa

Además de los trabajos de investigación mencionados, existen diferentes alternativas de software para ser usado libremente, entre las cuales resaltan las siguientes:

CaspLab

CaspLab es un software de libre distribución y de código abierto; el principal inconveniente de este sistema es que no es automático, debido a que el proceso de búsqueda de cometas y el método de segmentación los debe realizar un operador manualmente, por otra parte, el código fuente existente cuenta con escasa documentación (Końca et al., 2003).

CometScore

CometScore fue en un principio un software de libre distribución, no es open source, sin embargo, comparte las mismas características con CaspLab en el proceso de

segmentación, un operador etiqueta y delimita manualmente los cometas antes del proceso de medición. Lamentablemente en 2015 TriTek descontinua el desarrollo del programa por las abusivas faltas a las condiciones de uso del software (Corp, 2003).

OpenComet

OpenComet se plantea como un programa capaz de llevar a cabo la segmentación de cometas en imágenes del ensayo, tiene la capacidad de ser multiplataforma, acepta los principales formatos de imágenes (TIFF, PNG, BMP), también se pueden procesar múltiples imágenes en una sola ejecución del algoritmo. En las pruebas reportadas por el autor se observa una segmentación exitosa de las imágenes de prueba tanto de cometas como de regiones en los cometas segmentados cuyas estadísticas son exportadas a una hoja de cálculo. OpenComet tiene un marco de manejo de errores por cada imagen procesada, lo que permite eliminar a regiones seleccionadas erróneamente como cometas.

Las características antes mencionadas hacen a OpenComet una propuesta interesante que sobresale de las demás. Sin embargo, aparte de la deficiencia mencionada en el apartado 2.2.4, al momento de realizar pruebas con el programa se descubren algunos inconvenientes y se ve a la necesidad de proponer algunas mejoras tanto al algoritmo, como al programa, estas propuestas de mejora son material para la segunda parte del siguiente capítulo.

CAPÍTULO III. DISEÑO DEL MÉTODO DE SEGMENTACIÓN

En el primer apartado de este capítulo se describe el desarrollo del modelo propuesto, el cual se divide en seis etapas. La primera etapa cubre los requerimientos que se deben cumplir para obtener las imágenes digitales del ensayo. La segunda etapa realiza un preprocesamiento, que nos deja una imagen propicia para la segmentación. La tercera etapa trata del proceso de segmentación por umbralizado, cubre los algoritmos utilizados para separar las regiones de interés (ROIs) del fondo. En la cuarta etapa se describe el proceso para obtener los descriptores de los objetos y el proceso utilizado para descartar aquellos que no son considerados cometas. La etapa cuatro garantiza que los procesos siguientes son realizados en regiones que tienen descriptores que permiten identificar a las regiones como cometas, es decir, a partir de la etapa 4, cada ROI restante, es un cometa.

En la quinta etapa, por cada cometa encontrado se obtiene una imagen recortada de la imagen original en escala de grises, esta imagen se denomina subimagen. Se obtienen los patrones de la subimagen y son procesados en dos formas diferentes, descritas en las etapas 6.1 y 6.2. En la etapa 6.1 los patrones de la imagen son procesados por el algoritmo de clustering K-Means y posterior al proceso de clustering se obtienen las regiones encontradas. En la etapa 6.2, los mismos patrones son procesados con el algoritmo de clustering Fuzzy C-Means y se obtienen las regiones encontradas. La Figura 23 muestra el diagrama del método de segmentación propuesto.

En el segundo apartado de este capítulo se describen los cambios propuestos al software de código abierto OpenComet, se considera que los cambios propuestos e implementados mejoran la segmentación de cometas y usabilidad del software.

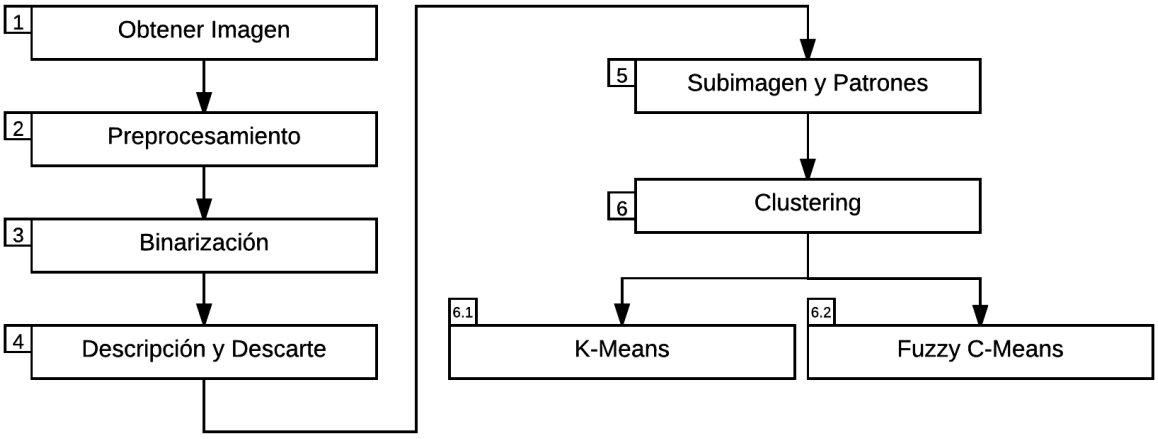

Figura 23 Etapas del algoritmo de segmentación propuesto

3.1 Modelo propuesto

3.1.1 Obtener imágenes

Existen dos formas de manera general de procesar imágenes del ensayo cometa, una es por el análisis de muestras con video en tiempo real y una segunda por el análisis de muestras en imágenes digitalizadas, para este trabajo de investigación se trabaja solo con imágenes digitales, es decir, durante la prueba del ensayo cometa, las imágenes son digitalizadas y guardadas para posteriormente analizarlas con el algoritmo propuesto, aunque claramente el método debería funcionar sin dificultades en un análisis en tiempo real, tomando en cuenta que el video no es más que una secuencia de imágenes.

Se pueden analizar cualquier tipo de imágenes resultado del ensayo, pero éstas deben cumplir ciertos requerimientos para garantizar una segmentación exitosa.

- Los cometas deben estar orientados de izquierda a derecha, (núcleo a la izquierda y cola hacia la derecha).
- Se debe evitar en lo posible durante el ensayo a los cometas sobrepuestos, ya que serán eliminados durante el proceso de selección o descarte.
- Los cometas deberán tener un recorrido horizontal simétrico respecto al núcleo de la célula.

3.1.2 Preprocesamiento

3.1.2.1 Escala de grises

El preprocesamiento se realiza sobre una imagen en escala de grises, el método utilizado para transformar una imagen a color RGB a una imagen en escala de grises es obteniendo el promedio de las tres capas RGB. Generando así una imagen en escala de grises de 8 bits de profundidad.

3.1.2.2 Filtrado de la imagen

El método utilizado para suavizar la imagen es el filtro promediador (average) con un vecindario de 11x11 pixeles, realizando un relleno de ceros en el borde de la imagen para poder procesar los pixeles del borde, el relleno de ceros es eliminado tras culminar el proceso de filtrado. Después de filtrar la imagen, el fondo es más uniforme, ya que los elementos muy obscuros son aclarados tanto como el promedio del valor vecindario y los pixeles con altos valores de brillo son atenuados también al valor promedio del vecindario. Hasta este punto la imagen resultante continúa siendo una imagen en escala de grises de 8 bits.

Fue elegido un vecindario de 11x11 pixeles por experimentación, es importante tomar en cuenta que el vecindario elegido está basado en las características de las imágenes utilizadas para experimentación descritas en el apartado 4.1, para imágenes de distinta naturaleza (lente objetivo y definición de la cámara, entre otros.), el vecindario deberá cubrir un área aproximada a la que cubre el vecindario del presente trabajo.

3.1.3 Binarización

3.1.3.1 Umbralizado Local

Ya que nuestra imagen se encuentra con un fondo medianamente uniforme procedemos a aplicar el método de umbralizado local que nos ayuda a segmentar aquellas regiones que no lograron ser suavizados por el proceso anterior, (R C Gonzalez, Woods, & Eddins, 2009) proponen un método de umbralizado local que se describe en el Algoritmo 1.

Algoritmo 1: Umbralizado local
Entrada: El algoritmo puede ser programado como una función que recibe los siguientes parámetros de entrada (f, nhood, a, b, meantype) Donde: f : Es la imagen en escala de grises de 8 bits. nhood : Es el vecindario que va a formar la ventana del filtro (conformada generalmente de una matriz con unos de tamaño 3X3). a : Es una constante de peso para la matriz de desviaciones estándar. b : Es una constante de peso para la matriz de promedios. meantype : Es el tipo de promedio que se va a utilizar, puede ser 'global' o 'local'.
Salida: $g(x,y)$**:** Es imagen binaria del mismo tamaño que la imagen de entrada, donde los pixeles que pertenecen a objetos tienen 1 como valor y cero en caso contrario.
Proceso: Paso 1: Se calcula la matriz de desviaciones estándar locales de la siguiente manera, por cada pixel de la imagen se calcula la desviación estándar de los elementos que se encuentran bajo la ventana (nhood), el valor resultante se guarda en una nueva matriz (SIG) en la misma posición del elemento central w(0,0) del vecindario. Paso 2: Se calcula la matriz de promedios basados en el parámetro meantype, si el parámetro es igual a 'global' se obtiene el promedio de la imagen basado en el valor de brillo de todos sus pixeles. Si el parámetro recibido es diferente de 'global' se realiza una operación de vecindad a la imagen y se obtiene el promedio para cada elemento de la imagen basado en los elementos bajo la ventana durante la operación de vecindad, que también se guardan en una matriz (MEAN). Paso 3: El resultado de la imagen binarizada se obtiene con la ecuación (20). $$g(x,y) = \left(f(x,y) > a*SIG(x,y)\right) \;\&\; \left(f(x,y) > b*MEAN(x,y)\right) \quad (20)$$

Un pixel de g es igual a 1 si y sólo si el valor de brillo del pixel en la posición $f(x,y)$ es mayor al valor de la desviación estándar local en la posición (x,y) de la matriz de desviaciones estándar SIG(x,y) (multiplicado por el escalar a) y el valor de brillo del pixel en $f(x,y)$ es mayor al valor promedio del mismo pixel en la matriz de promedios MEAN

(x, y) (multiplicado por el escalar b). Si las condiciones antes establecidas no se cumplen el pixel adquiere un valor de 0, el pseudocódigo 1 muestra este proceso.

Pseudocódigo 1: Proceso de umbralizado local

```
1     funcion Respuesta = uLocal(Imagen, Vecindario, a, b, TipoPromedio)
2         mDesvStd = CalcularMatrizDStd(Imagen, Vecindario)
3         si TipoPromedio == "global"
4         mPromedios = CalcularValorProm(Imagen)
5         si no
6         mPromedios = CalcularMatrizProm(Imagen)
7         fin si
8
9         m = ObtenerFilas(Imagen);
10        n = ObtenerColumnas(Imagen);
11        para i = 0 hasta m-1
12            para j = 0 hasta n-1
13                MayorDStd = Imagen(i,j) > a * mDesvStd(i,j)
14                MayorMPromedios = (Imagen(i,j) > b * mPromedios(i,j)
15                g(i,j) = (MayorDStd & MayorMPromedios) ? 1 : 0
16            fin para
17        fin para
18        Respuesta = g
19    fin funcion
```

Tras el proceso de umbralizado se realizan los procesos de conteo y etiquetado para poder definir las regiones encontradas y así, obtener sus descriptores. Recordemos que el proceso de conteo asigna un número consecutivo a una región de tal manera que nos permita saber cuántas regiones fueron encontradas mientras que el proceso de etiquetado marca con el mismo identificador a los elementos (pixeles) que forman parte de una misma región, de esta manera es posible saber que pixeles pertenecen a una región determinada.

3.1.3.2 Eliminación de regiones pequeñas

Se debe calcular el tamaño mínimo que puede tener un objeto considerado como cometa sin daño, para poder calcular la cantidad de pixeles mínimos, se toma en cuenta el objetivo utilizado en el microscopio, así como la definición en megapíxeles de la cámara o dispositivo digitalizador. En las imágenes que se utilizan en el presente trabajo, el número de pixeles mínimos de un cometa es de ≈ 17,500, pero a efectos de prueba este parámetro es dejado en 3000, esto nos permite analizar una mayor cantidad de ROIs y analizar el comportamiento de rechazo de los descriptores (abarcado en el apartado 3.1.4).

3.1.3.3 Eliminación de regiones en el borde

Se eliminan las regiones que tocan el borde de la imagen porque pueden ser cometas que no son aptos para un análisis ya que su nivel de brillo (núcleo, halo o cola) pudo haber sido comprometido al digitalizar la imagen, esto podría arrojar resultados erróneos, el proceso es descrito en el Algoritmo 2, este mismo proceso se describe en el

Pseudocódigo 2.

Algoritmo 2: Eliminación de regiones en el borde
Entrada: Imagen: Imagen binaria original que contiene todas las regiones. Regiones: Un arreglo que contiene a las regiones extraídas de la imagen binaria.
Salida: Arreglo de regiones pero sin las regiones que toquen el borde de la imagen binaria recibida.
Proceso: Paso 1: Iterar sobre el arreglo de regiones y obtener una región para iniciar el análisis. Paso 2: Por cada región iterar sobre cada pixel de la región y obtener la coordenada del pixel. Paso 3: Si al menos un pixel de la región actual toca alguna fila o columna del borde de la imagen, esa región debe ser eliminada del arreglo de regiones.

Pseudocódigo 2: Limpiar regiones que tocan el borde de la imagen

```
1     funcion LimpiarBordes(Imagen, Regiones)
2         m = ObtenerNumeroFilas(Imagen);
3         n = ObtenerNumeroColumnas(Imagen);
4
5         CantidadRegiones = ObtenerCantidadRegiones(Regiones);
6         para i = 0 hasta CantidadRegiones
7         Region = Regiones(i)
8         VectorPixeles = ObtenerPixeles(Region)
9         CantidadPixeles = ObtenerCantidadPixeles(VectorPixeles)
10            para k = 0 hasta CantidadPixeles
11             CoordenadaPixel = ObtenerCoordenadaPixel(VectorPixeles(k))
12              si CoordenadaPixel en rango ([0 (m-1)], 0) ó (0, [0 n-1]
13              ó (m-1, [0 n-1] ó ([0 (m-1)], n-1))
14                    EliminarRegion(Regiones(i))
15                    terminar para k
16                fin si
17            fin para
18        fin para
19    fin funcion
```

3.1.4 Descripción y Descarte

3.1.4.1 Descriptores

Después del proceso de binarización se obtienen regiones que cumplen con las siguientes características:

- Son regiones con pixeles mayores o igual a la cantidad mínima que contiene un cometa sin daño.
- Ninguno de sus pixeles toca el borde de la imagen.

El siguiente paso es obtener los descriptores de las regiones que nos van a servir para eliminar regiones que no cumplen con los parámetros de un cometa, los descriptores utilizados se listan a continuación.

- Solidez
- Asimetría
- Proporción de aspecto (HRatio)
- Diferencia con la línea central (CLD)

Los detalles sobre cómo calcular cada descriptor de pueden consultar en el apartado 2.1.2.3 Regiones y sus propiedades geométricas. Cabe mencionar que, para el cálculo de la asimetría, se utiliza el 30% de las columnas de la región como se describe en el ejemplo del descriptor.

3.1.4.2 Descarte

Las regiones son sometidas a un proceso donde se etiquetan como cometas o se eliminan basado en sus descriptores y ciertos parámetros umbrales establecidos por experimentación. El algoritmo 3 describe el proceso de eliminación de regiones por descriptores.

Algoritmo 3: Eliminación de regiones por sus descriptores
Entrada: Arreglo de regiones: Un arreglo de regiones que no hayan sido eliminados en procesos anteriores. Arreglo de descriptores: Este arreglo deberá contener todos los descriptores definidos previamente por cada región.
Salida: Arreglo de regiones: Cada elemento de este arreglo es una región que es considerada un cometa.
Proceso: Paso 1: Por cada elemento región del arreglo de regiones recibido se deben obtener y comparar los descriptores listados a continuación con los siguientes valores umbrales. Solidez < 0.5 Asimetría > 0.7 HRatio > 1.1 CLD > 0.1 Paso 2: Si al menos una de las comparaciones anteriores es verdadera, esta región debe ser eliminada del arreglo de regiones en caso contrario, esta región es preservada en el arreglo de regiones original.

La forma en cómo se procesarían las regiones es muy parecida a la descrita en el Pseudocódigo 2, solo que se toman los descriptores listados anteriormente y sus valores umbrales para eliminar regiones y no si la región toca el borde de la imagen o no.

3.1.5 Sub-imagen y Patrones

Cuando el proceso de eliminación de regiones termina y existen regiones que cumplen con los requerimientos para ser analizados como cometas, estas regiones son procesadas individualmente llevando a cabo el siguiente proceso en cada una de ellas.

3.1.5.1 Subimagen de la región

Basado en el rectángulo mínimo de cada región restante, debe ser posible localizar y obtener la subimagen del cometa de la imagen en escala de grises que se utilizó para aplicar el filtro promediador.

Hay ciertas cuestiones que se tienen que aclarar sobre la subimagen:

- Es una imagen que encierra a un cometa
- Es una imagen en escala de grises de 8 bits
- El tamaño de la imagen es del tamaño del rectángulo mínimo (ancho y alto)

En el Algoritmo 4 se describe el proceso para obtener la subimagen de una región.

Algoritmo 4: Obtener la subimagen de una región
Entrada: Imagen: Imagen original en escala de grises que se utilizó para generar la imagen filtrada con el filtro promediador. P1: Punto1 con las coordenadas (x1, y1) perteneciente a la esquina superior izquierda del rectángulo contenedor. P2: Punto2 con las coordenadas (x2, y2) que pertenecen a la esquina inferior derecha del rectángulo contenedor.
Salida: Subimagen: Imagen del tamaño del rectángulo mínimo que contiene un cometa segmentado de la imagen en escala de grises.
Proceso: Paso 1: Generar una matriz que va a contener la subimagen. Paso 2: Iterar sobre la imagen original hasta llegar al punto (x1, y1). Paso 3: Copiar los valores de pixeles de la imagen original en escala de grises a la subimagen a partir del punto (x1, y1) hasta el punto (x2, y2). En las coordenadas (0,0) hasta $(M^{'}-1,N^{'}-1)$ de la subimagen.

Donde P1 y P2 hacen referencia a las coordenadas de la imagen original en escala de grises; M’ y N’ es el largo y ancho de la subimagen.

Pseudocódigo 3: Obtener la subimagen de una región

```
funcion Respuesta = ObtenerSubImagen(Imagen, BoundingBox)
    m = ObtenerNumeroFilas(Imagen)
    n = ObtenerNumeroColumnas(Imagen)
    x1,y1 = ObtenerCoordenadasP1(BoundingBox)
    x2,y2 = ObtenerCoordenadasP2(BoundingBox)
    AltoSubimagen = ObtenerAlto(BoundingBox)
    AnchoSubimagen = ObtenerAncho(BoundingBox)
    g = CrearMatrizZeros(AltoSubimagen, AnchoSubimagen);

    para i = 0 hasta m-1
        para j = 0 hasta n-1
        si i,j >= x1,y1 & i,j <= x2,y2
            g(i,j) = Imagen(i,j)
        fin para
    fin para
    Respuesta = g;
fin funcion
```

3.1.5.2 Patrones del cometa

Ya que se tiene segmentada la subimagen, se obtienen los patrones de esa imagen, el proceso es descrito en el Algoritmo 5.

Algoritmo 5: Obtener patrones de la subimagen

Entrada:

Subimagen: Una subimagen que es el cometa segmentado de la imagen en escala de grises, del tamaño del rectángulo contenedor.

Salida:

MatrizPatrones: Una matriz de nx3 que contiene en cada fila los datos de cada patrón (vector fila).

Proceso:

Paso 1: Realizar un relleno a la subimagen con 5 pixeles con valor cero.

Paso 2: Llevar a cabo una operación de vecindad con un vecindario de 11x11.

Paso 3: Para cada paso de la operación de vecindad obtener los siguientes datos.

- Valor de brillo de la imagen en escala de grises en la posición w(0,0) del vecindario.
- Desviación estándar de todos los elementos de la subimagen en escala de grises que son cubiertos por el vecindario 11x11.
- Valor promedio de todos los elementos de la subimagen en escala de grises que son cubiertos por el vecindario.

Paso 4: Agregar esta matriz 1x3 que conforma un patrón a la matriz que contendrá los patrones de toda la subimagen.

Paso 5: Eliminar el padding o relleno agregado a la subimagen.

Al finalizar la operación de vecindad, la matriz de patrones tiene la forma mostrada en la Figura 24. La matriz se presenta con fines ilustrativos y no representa un conjunto de datos en específico.

	ValorBrillo	DesvStd	Promedio
Patron 1	1.111111	2.270856	5.371901
Patron 2	1.388889	2.296222	5.968779
Patron 3	8.333333	2.432208	5.693297
		.	
		.	
		.	
Patron n	5.555556	2.098899	5.578512

Figura 24 Matriz de patrones de la sub imagen de un cometa.

Pseudocódigo 4: Obtener patrones de una subimagen

```
1     funcion Patrones = ObtenerPatrones(SubImagen, Filas, Columnas)
2         SubImagen = TransformarADouble(SubImagen);
3         SubImagen = EstirarHistograma(SubImagen, 0, 1);
4         stdImg = ObtenerDesvStdPorVecindad(SubImagen, 11, 11);
5         meanImg = ObtenerPromedioPorVecindad(SubImagen, 11, 11);
6         Tam = Filas * Columnas;
7         Respuesta = zeros(Tam, 3);
8         k = 0;
9         para i = 0 hasta (Filas - 1),
10            para j = 0 hasta (Columnas - 1) ,
11                Respuesta(k, 1) = SubImagen(i,j);
12                Respuesta(k, 2) = stdImg(i,j);
13                Respuesta(k, 3) = meanImg(i,j);
14                k++;
15            fin para
16        fin para
17        Patrones = Respuesta;
18    fin function
```

3.1.6 Clustering

3.1.6.1 Clustering con K-Means

Una vez se obtiene la matriz de patrones, se lleva a cabo el algoritmo de clustering K-Means descrito en el apartado 2.1.3 con los parámetros siguientes.

Clases: 4, que corresponde a las cuatro regiones que se desean segmentar en el cometa, núcleo, halo, cola y fondo de la imagen.

Centroides Iniciales: Se utiliza el algoritmo k-means++ propuesto por (Arthur & Vassilvitskii, 2007) para obtener los centroides iniciales.

Distancia: Distancia euclidiana, para calcular la distancia de un patrón con los centroides se utiliza la distancia euclidiana, definida en la ecuación (21).

$$d(a,b) = \left[\sum_{i=1}^{n} |ai - bi|^2 \right]^{\frac{1}{2}} \quad (21)$$

Donde a es un patrón y b es un centroide, de n características.

Condición de parada: Las condiciones de parada del algoritmo son las siguientes

- Se alcanzó el número máximo de 100 iteraciones.
- Las asignaciones a los clusters ya no cambian.

El algoritmo de clustering K-Means deberá devolver los siguientes componentes

Matriz de los centroides de clase
Esta matriz es de dimensión Cx3, siendo C la cantidad de clases, y 3 las columnas que indican los valores centrales de las características de Brillo, Desviación estándar y el Promedio para cada clase.

Vector de índices
Es un vector de nx1 donde n es el número de patrones, el valor de cada elemento del vector es igual al índice de la clase a la que pertenece, eso nos dice que el rango de valores que podrían llegar a tener será [1 C].

Regiones
Basados en la matriz de centroides de clase y el vector de índices se pueden obtener las regiones de la imagen, este proceso es definido en el Algoritmo 6.

Algoritmo 6: Obtener las regiones de K-Means
Entrada: MatrizCentroides: Matriz con los centroides que devuelve el algoritmo K-Means VectorIndices: Vector con los índices de clase al que pertenece cada patrón. CantidadFilasSubimagen: Filas de la subimagen de donde se obtienen los patrones. CantidadColumnasSubimagen: Columnas de la subimagen de donde se obtienen los patrones.
Salida: ImagenRegiones: Imagen RGB con las regiones definidas de la siguiente forma • Negro: Color que define el fondo de la imagen. • Verde: Define la cola del cometa. • Azul: Halo del cometa. • Rojo: Núcleo del cometa. CantidadElementos: Un vector que contiene la cantidad pixeles que pertenecen a cada región.
Proceso: Paso 1: De la matriz de centroides de clase obtener un vector que contenga solo los valores centrales referente al brillo (debería estar ubicada en la primera columna de cada fila de la matriz de centroides de clase, según como se define en el apartado 3.1.5.2). Paso 2: Ordenar el vector de valores centrales de brillo de menor a mayor.

Paso 3: Generar la imagen resultado con la cantidad de filas y columnas recibidas.
Paso 4: Iterar sobre el vector de índices y llevar a cabo el siguiente proceso por cada elemento del vector de índices de clase:

1. Obtener el valor de la clase, el cual indica la clase a la que pertenece el patrón.
2. Basado en la clase a la que pertenece el patrón, obtener de la matriz de centroides el valor de brillo central de esa clase, utilizar este valor de brillo para obtener su posición en el vector de brillos centrales ordenado.
3. La posición donde se encuentra el valor de brillo central en el vector ordenado define la región a la que pertenece el patrón y por lo tanto el color del pixel en esa posición.
4. Las regiones son definidas según la posición de brillo central de la clase en el vector de brillos ordenados de menor a mayor, 1 = Fondo, 2 = Cola, 3 = Halo, 4 = Núcleo.
5. Establecer el color a la imagen que se va a devolver según la región asignada al patrón.
6. Sumar 1 al contador de elementos de esa región.

Pseudocódigo 5: Obtener regiones de K-Means

```
funcion [Img, Frec] = ObtenerKRegiones(mCent, vIndices, Filas, Columnas)
    VectorBrillo = ObtenerVectorBrillo(mCent);
    VectorBrilloOrd = OrdenarVector(VectorBrillo);
    Resultado = CrearImagenRGB(Filas, Columnas);
    IndiceActual = 0;
    para i = 0 hasta (Filas-1)
        para j = 0 hasta (Columnas-1)
            Clase = ObtenerClase(vIndices, IndiceActual++);
            Region = ObtenerRegion(Clase, VectorBrillo, VectorBrilloOrd);
            Resultado(i, j) = ObtenerColorRegion(Region);
            ContadorRegion[Region]++;
        fin para
    fin para
    Img = Resultado;
    Frec = ContadorRegion;
fin funcion
```

3.1.6.2 Clustering con Fuzzy C-Means

El segundo método para obtener regiones es con el algoritmo Fuzzy C-Means (FCM), a continuación, se definen los parámetros del algoritmo.

Clases: 4, correspondiente a las regiones del cometa.
Matriz de pertenencia inicial: aleatorio
Centroides iniciales: Basados en el promedio de los valores de pertenencia de los patrones en cada clase.
Condición de parada: Se alcanzó el máximo de 100 iteraciones o la función objetivo se ha mejorado por menos que 1e-5 en las ultimas 2 iteraciones.

El algoritmo FCM devuelve los siguientes componentes
Centroides: Son los patrones centrales de cada clase, una matriz de CX3 columnas.

U o Matriz de pertenencia: Es una matriz de C filas y n columnas, en donde C es el número de clases y n la cantidad de patrones, cada columna de la matriz representa a un patrón y el valor de cada elemento de la columna es el grado de pertenencia de ese patrón con su respectiva clase.

Regiones

Las regiones se definen a partir de los componentes devueltos del algoritmo FCM, la matriz de patrones centrales y la matriz de pertenencia. El algoritmo 7 describe el proceso para obtener las regiones generadas por el algoritmo FCM.

Algoritmo 7: Obtener regiones de Fuzzy C-Means

Entrada:

U: Matriz de pertenencia
Centroides: Matriz que contiene los valores centrales de cada clase.
Filas: Cantidad de filas de la subimagen que generó los patrones.
Columnas: Cantidad de columnas de la subimagen.

Salida:

ImagenRegiones: Imagen RGB con las regiones definidas de la siguiente forma

- Negro: Color que define el fondo de la imagen.
- Verde: Define la cola del cometa.
- Azul: Halo del cometa.
- Rojo: Núcleo del cometa.

CantidadElementos: Un vector que contiene la cantidad pixeles que pertenecen a cada región.

Proceso:

Paso 1: Se obtiene la columna de brillo de la matriz de centroides.

Paso 2: Basado en la columna de brillo de la matriz de centroides se ordenan las filas de la matriz de pertenencia ***U***, dejando al principio la clase (fila) con el valor de brillo central más pequeño (región del fondo) y al final se deja a la fila con el valor de brillo central máximo (región del núcleo).

Paso 3: Se obtiene la clase a la que pertenece cada patrón basado el valor máximo de pertenencia.

Paso 4: La región a la que pertenece el patrón está basada en la posición en la que la fila con mayor grado de pertenencia ha sido reordenada, la posición y sus respectivas regiones se listan a continuación, comenzando de arriba hacia abajo:

- Primera fila: Fondo
- Segunda fila: Cola
- Tercera fila: Halo
- Cuarta Fila: Núcleo

Paso 5: Establecer el color a la imagen que se va a devolver según la región asignada al patrón.

Paso 6: Sumar 1 al contador de elementos de esa región.

Pseudocódigo 6: Obtener regiones FCM

```
funcion [Img, Frec] = ObtenerRegionesFCM(U, mCentroides, Filas, Columnas)
    VectorBrillo = ObtenerVectorBrillo(mCentroides);
    UOrdenada = ReordenarFilasPorVectorBrillo(U, VectorBrillo);
    Resultado = CrearImagenRGB(Filas, Columnas);
    ColumnaActual = 0;
    para i = 0 hasta (Filas-1)
        para j = 0 hasta (Columnas-1)
            Clase = ObtenerClase(U, ColumnaActual++);
            Region = ObtenerRegion(Clase);
            Resultado(i, j) = ObtenerColorRegion(Region);
            ContadorRegion[Region]++;
        fin para
    fin para
    Img = Resultado;
    Frec = ContadorRegion;
fin funcion
```

Del apartado 3.1 se realiza una experimentación en el siguiente capítulo con el fin de realizar una validación y correlación de resultados entre los algoritmos de aprendizaje no supervisado K-Means y FCM para determinar si los resultados de regiones en cada algoritmo tienen diferencias significativas, los resultados de la experimentación y el código fuente de todo el proceso descrito en el apartado 3.1 se puede consultar y descargar desde: https://github.com/gerardoepitacio/CometAssaySegmentation.

3.2 Propuestas de modificación a OpenComet

Tras la revisión del estado del arte se conoce a la solución OpenComet, un pequeño programa que funciona como un complemento al entorno de trabajo de procesamiento de imágenes ImageJ. OpenComet es uno de los programas más completos disponibles para realizar la segmentación de imágenes del ensayo cometa, debido a que la selección de cometas la realiza de manera automática, es rápido, eficiente e implementa un esquema de manejo de errores, además es gratis y de libre acceso al código fuente. Sin embargo, tras pruebas realizadas durante el desarrollo del presente trabajo de investigación, se observaron una serie de inconvenientes, aparte de los ya descritos en el apartado 2.2.4, que se listan a continuación.

- Cuenta con pocas opciones de configuración.
- No mantiene la persistencia de configuraciones previas.
- Se ejecuta como un complemento, por lo que requiere de un software adicional para funcionar.
- Cuando se realizaron pruebas con imágenes de distinta naturaleza (distinto método de tinción, microscopio, cámara, entre otros) la correcta selección de cometas resulta precaria dando como resultado un alto número de falsos positivos y falsos negativos.

En las secciones siguientes se describen una serie de mejoras que son propuestas e implementadas en el código fuente original de OpenComet, estas implementaciones se

consideran necesarias para mejorar el proceso de segmentación, así como la usabilidad del software.

3.2.1 Escala de grises

Se propone que la extracción de la imagen en escala de grises que se utiliza en el preprocesamiento se realice a partir del promedio de las tres capas RGB ya que el enfoque que inicialmente manejaba, no ofrecía buenos resultados. La razón es debido a las diferentes opciones de tinción que se pueden utilizar en la prueba del ensayo no siempre dejan la información útil en una sola capa si no que se distribuye sobre las tres o sobre dos, por lo tanto, optar por sólo procesar una capa RGB implicaba perdida de información que afectaba los resultados de la segmentación.

3.2.2 Opciones de configuración

Las opciones de configuración para OpenComet realmente son pocas, pudiendo elegir solamente si se va a realizar una corrección de fondo y el algoritmo para localizar el núcleo del cometa. Se considera importante poder modificar también los parámetros utilizados para la selección de cometas, por lo cual se agregan los parámetros listados a continuación en las opciones de configuración a OpenComet.

- Área mínima
- Convexidad mínima
- Simetría
- HRatio
- CLD
- Limpiar bordes
- Método de umbralizado

Además, los cambios en la configuración son guardados persistentemente en un archivo de configuración, esto evita la necesidad de volver a configurar los parámetros cada vez que se utilice el software. También se agrega un botón que permite reiniciar todos parámetros a un estado por omisión.

3.2.3 Previsualización

Tras agregar las opciones de configuración se vio la necesidad de obtener una previsualización de la segmentación, así que se agregaron opciones para poder pre visualizar el resultado de la segmentación a partir de las opciones de configuración establecidas, así permite al usuario elegir las opciones para la segmentación adecuada.

3.2.4 Software autónomo

OpenComet funciona como un complemento para ImageJ, esto agrega cierta complejidad y dependencia innecesaria al software, por lo tanto, también se propone una versión autónoma de OpenComet.

Los cambios realizados a OpenComet se encuentran disponibles para consultar y descargar en: https://github.com/gerardoepitacio/OpenCometUAI_.

CAPÍTULO IV. EXPERIMENTACIÓN Y DISCUSIÓN

En el penúltimo capítulo de este trabajo de investigación se realizan una serie de pruebas sobre un conjunto de imágenes utilizadas en un trabajo de investigación titulado “Polimorfismos en el gen pon1 y daño al DNA en población expuesta ocupacionalmente a plaguicidas organofosforados” (Morales, 2010), de la Unidad Academia de Ciencias Químico Biológicas de la UAGro. Se comienza este capítulo describiendo las imágenes que se van a utilizar, así como el software y hardware utilizado y culmina con el desarrollo de la experimentación, validación y discusión de los resultados.

La experimentación se divide en dos apartados, en primer lugar, se comparan los resultados generados entre los algoritmos de clustering K-Means y Fuzzy C-Means, en la segunda parte se realizan pruebas entre la versión original de OpenComet y la versión modificada, los resultados son mostrados a continuación y se debaten en el apartado 4.6.

4.1 Imágenes

El conjunto de imágenes de prueba pertenece al estudio de una tesis de maestría, donde se busca medir el nivel de daño al ADN en agricultores expuestos a plaguicidas, se utilizan células de sangre periférica, con un total de 11 imágenes que en el conteo manual arrojan un total de 20 cometas a segmentar, las imágenes fueron obtenidas bajo las condiciones de hardware y software descritas a continuación.

Microscopio: No especificado

Marca: No especificado

Modelo: No especificado

Objetivo: 40x

Nombre del a cámara (dispositivo digitalizador): No especificado

Tamaño de la imagen: 1280x960

Resolución: 1.2 Megapíxeles

Formato: .bmp

Tipo de imagen: RGB

Profundidad: 24 bits

4.2 Hardware

Las características de hardware de la computadora de pruebas son las siguientes:

Marca: Ensamblada

Modelo: N/A

Procesador: Intel® Core™ i5 CPU 760 @2.80GHz 2.79 GHz

RAM: 8GB Dual-Channel DDR3

Disco Duro: ADATA SP600 119GB (SSD)

Motherboard: GIGABYTE H55M-D2H

4.3 Software

- Se elige Matlab® como entorno de desarrollo, porque cuenta con el Image Processing Toolbox, un conjunto de bibliotecas para el procesamiento de imágenes, esto facilita mucho el prototipo de métodos de segmentación.
- La versión Original de OpenComet es la versión 1.3 y se ejecuta como un complemento para la versión 1.47v de ImageJ.
- Las modificaciones a OpenComet se realizan en Java, el mismo lenguaje de programación donde fue desarrollado originalmente, las pruebas de la versión modificada de OpenComet se realizaron con la versión 1.50b de ImageJ, cabe mencionar que las modificaciones entre las versiones 1.47v y 1.50b son mínimas; por lo tanto, no forma parte de los parámetros que afectan el desempeño de ambas versiones de OpenComet.
- La versión del JDK y JRE utilizada para compilar y testear OpenComet fue la 1.8.
- Ambas pruebas se realizan sobre una misma computadora con el sistema operativo Windows 10 Pro versión 1607 en la versión de 64 bits.

4.4 Experimentación

La finalidad de comparar los algoritmos de clustering K-Means y Fuzzy C-Means es para identificar las regiones que son capaces de segmentar cada uno de los algoritmos comparando las regiones encontradas y el tiempo que tardan en devolver el resultado final, los resultados de esta experimentación serán sometidos a un análisis posteriormente en el apartado de validación. En la experimentación con OpenComet se busca comparar la efectividad con la que segmentan los cometas ambas versiones y comprobar si la modificación realmente representa una mejora al software como se plantea.

4.4.1 Comparativa de regiones K-Means y Fuzzy C-Means

Se procesaron un total de 11 imágenes del ensayo cometa con la configuración considerada óptima, durante el conteo manual se identificaron 20 cometas que son aptos para ser procesados, también se identificaron 2 ROIs evidentes que no son un cometa. El método de segmentación propuesto devolvió los siguientes resultados:

- Identifica correctamente a 19 cometas
- Rechaza correctamente 1 ROI no apto para el análisis
- Rechaza incorrectamente 1 ROI que si era un cometa debido a que la proporción de aspecto sobrepasa el umbral establecido
- Acepta como cometa 1 ROI que no era cometa, cabe aclarar que esta región hubiera sido rechazado si se mantenía el área mínima de un cometa sin daño ($\approx$ 17500 pixeles), como se comenta en el apartado 3.1.3.2.

La Figura 25 muestra distintos casos en los que el método propuesto tuvo que descartar o seleccionar regiones en la imagen.

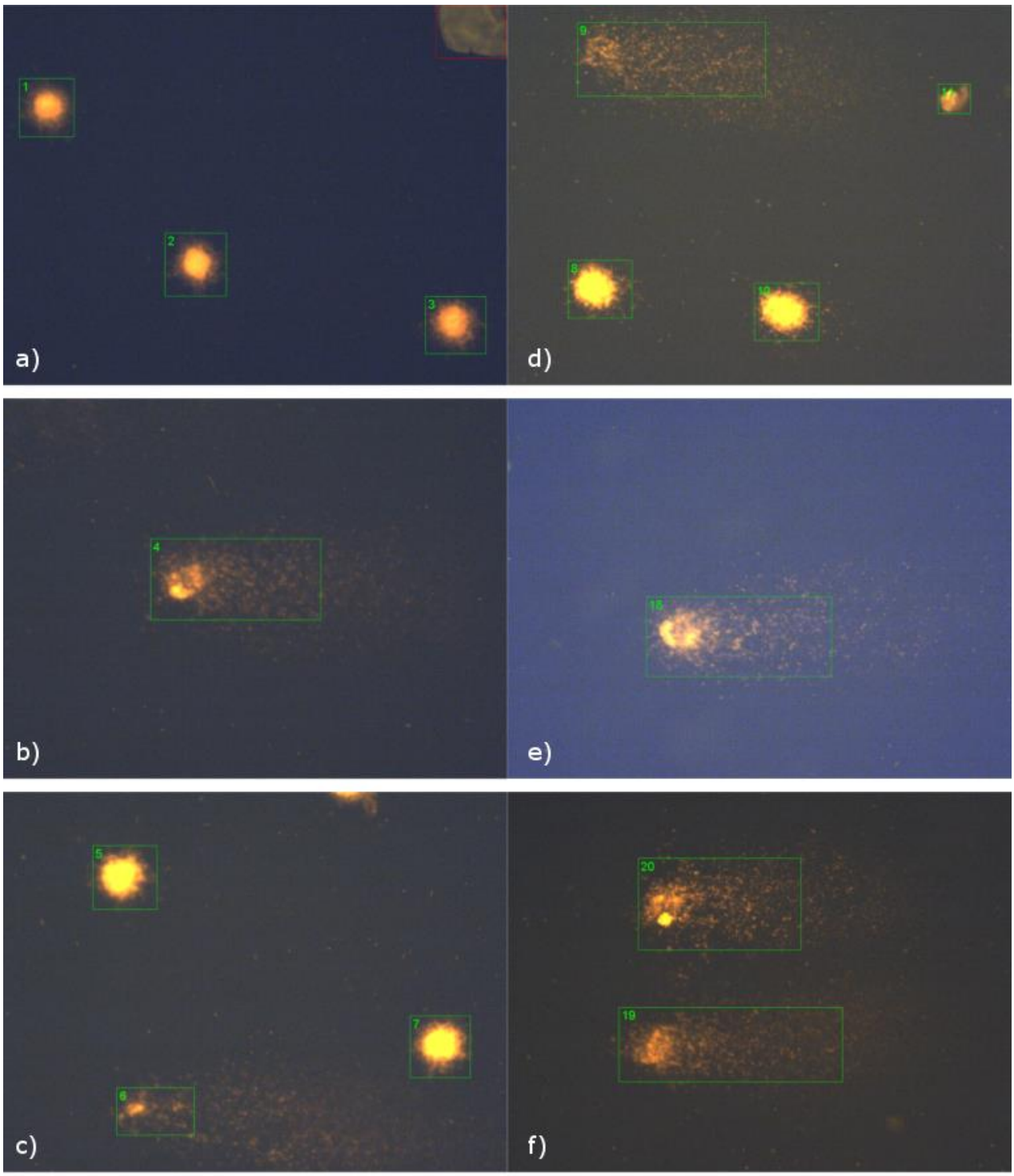

Figura 25, Se muestran diferentes escenarios que se presentaron al momento de realizar las pruebas. ***a)*** *Selección exitosa de 3 cometas sin daño aparente, se rechaza una región que no era un cometa y que tocaba el borde.* ***b) y e)*** *Selección de un solo cometa en la imagen.* ***c)*** *Selección de 3 cometas, una con daño grave.* ***d)*** *Selección errónea de una región como cometa (recuadro con el número 14).* ***f)*** *Selección de 2 cometas con daño medio.*

La Figura 26 muestra la segmentación de regiones en los cometas del conjunto de pruebas, la primera columna muestra los resultados de la segmentación realizada por K-Means y la segunda columna muestra los resultados obtenidos por Fuzzy C-Means. Las imágenes en la parte superior son células sin daño aparente y el daño va incrementando gradualmente a medida que se llega a las células de la parte inferior.

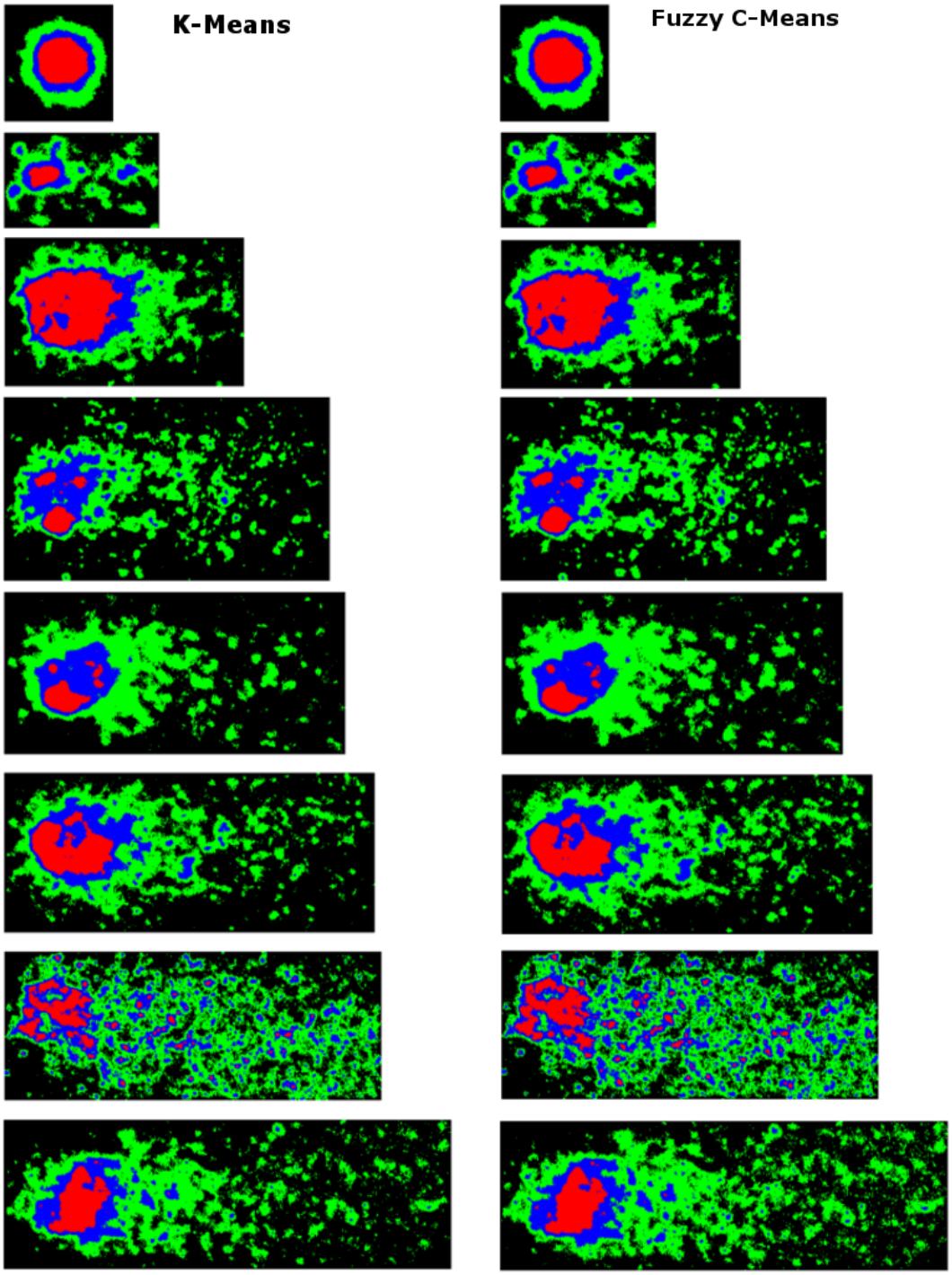

Figura 26 Regiones segmentadas con los algoritmos de clustering para cometas con distinto nivel de daño, a simple vista se puede observar que las regiones segmentadas son muy similares.

Los resultados de sensibilidad y especificidad de las pruebas se muestran en la Tabla 1.

Tabla 1 Parámetros de sensibilidad y especificidad del algoritmo propuesto.

Resultados de las pruebas		
VERDADERO	POSITIVO	19
	NEGATIVO	1
FALSO	POSITIVO	1
	NEGATIVO	1
Tamaño de la muestra		20
Sensibilidad		**95.00%**
Especificidad		**50.00%**

La Tabla 2 lista el área en pixeles de las regiones que se pudieron segmentar en los algoritmos de clustering, donde #Img indica el número de cometa en la imagen y # indica el número con el cual el cometa es etiquetado. La última columna de la Tabla 2 muestra el tiempo de ejecución de cada algoritmo al realizar la segmentación.

Tabla 2 Resultado de áreas entre los algoritmos K-Means y FCM.

K-Means & FCM									
Imagen	#Img	#	Algoritmo	Pixeles	Fondo	Núcleo	Halo	Cola	Tiempo (s)
0309120ctol.bmp	1	1	K-Means	20139	10668	2951	2033	4487	0.27929713
	1	1	FCM	20139	10244	2938	2134	4823	0.5539544
	2	2	K-Means	24486	14363	3299	1957	4867	0.1475969
	2	2	FCM	24486	14149	3218	2008	5111	0.6637547
	3	3	K-Means	22040	11848	3208	2184	4800	0.12267687
	3	3	FCM	22040	11596	3169	2269	5006	0.638383
0309120ctol2.bmp	1	4	K-Means	87516	61838	1743	4773	19162	0.61055185
	1	4	FCM	87516	58886	1639	4936	22055	3.36737414
0309120ctol3.bmp	1	5	K-Means	25921	13509	5237	3007	4168	0.11053758
	1	5	FCM	25921	13199	5161	3129	4432	0.63059576
	2	6	K-Means	23400	14016	746	1971	6667	0.13561122
	2	6	FCM	23400	12901	737	2218	7544	0.493277
	3	7	K-Means	23712	11662	4899	2739	4412	0.11495838
	3	7	FCM	23712	11308	4896	2860	4648	0.44097565
0309120ctol4.bmp	1	8	K-Means	23506	11564	5774	2596	3572	0.06133971
	1	8	FCM	23506	11189	5721	2707	3889	0.5751346
	2	9	K-Means	88638	38849	3720	12484	33585	0.53094853
	2	9	FCM	88638	34148	4862	15531	34097	2.98256658
	3	10	K-Means	23490	11216	5549	3210	3515	0.09235042
	3	10	FCM	23490	10835	5545	3337	3773	0.48907124
	4	11	K-Means	6075	2635	570	999	1871	0.03806651
	4	11	FCM	6075	2596	626	956	1897	0.18005073

0309120ctol5.bmp	1	12	K-Means	23707	10752	6313	2679	3963	0.1253883
	1	12	FCM	23707	10381	6300	2780	4246	0.40893289
	2	13	K-Means	29340	15797	6641	2854	4048	0.1181288
	2	13	FCM	29340	15280	6621	2978	4461	0.3324561
0309120t0 1.bmp	1	14	K-Means	41415	25674	5147	3574	7020	0.26316092
	1	14	FCM	41415	24039	5228	3842	8306	1.29956926
0309120t0 2.bmp	1	15	K-Means	22192	9272	5086	3379	4455	0.07040367
	1	15	FCM	22192	8822	5089	3552	4729	0.43480342
	2	16	K-Means	33128	18361	6015	3579	5173	0.15587896
	2	16	FCM	33128	17610	5995	3806	5717	0.8912431
0309120t0 3.bmp	1	17	K-Means	56287	29552	7782	5192	13761	0.2197531
	1	17	FCM	56287	28354	7645	5415	14873	1.54920628
0309120t0 6.bmp	1	18	K-Means	93666	60581	4699	6775	21611	0.57609937
	1	18	FCM	93666	55664	4728	7556	25718	3.69632592
0309120t60 2.bmp	1	19	K-Means	105844	71110	3581	6010	25143	0.65866244
	1	19	FCM	105844	62546	3738	7297	32263	5.12196335
	2	20	K-Means	94710	68720	1466	5983	18541	0.65511314
	2	20	FCM	94710	64658	1483	6663	21906	4.16240373

Las Figuras del 27 al 30 muestran en una gráfica los resultados de la Tabla 2, en donde se compara la cantidad de pixeles que los algoritmos de clustering asignan a las regiones de Fondo, Núcleo, Halo y Cola respectivamente.

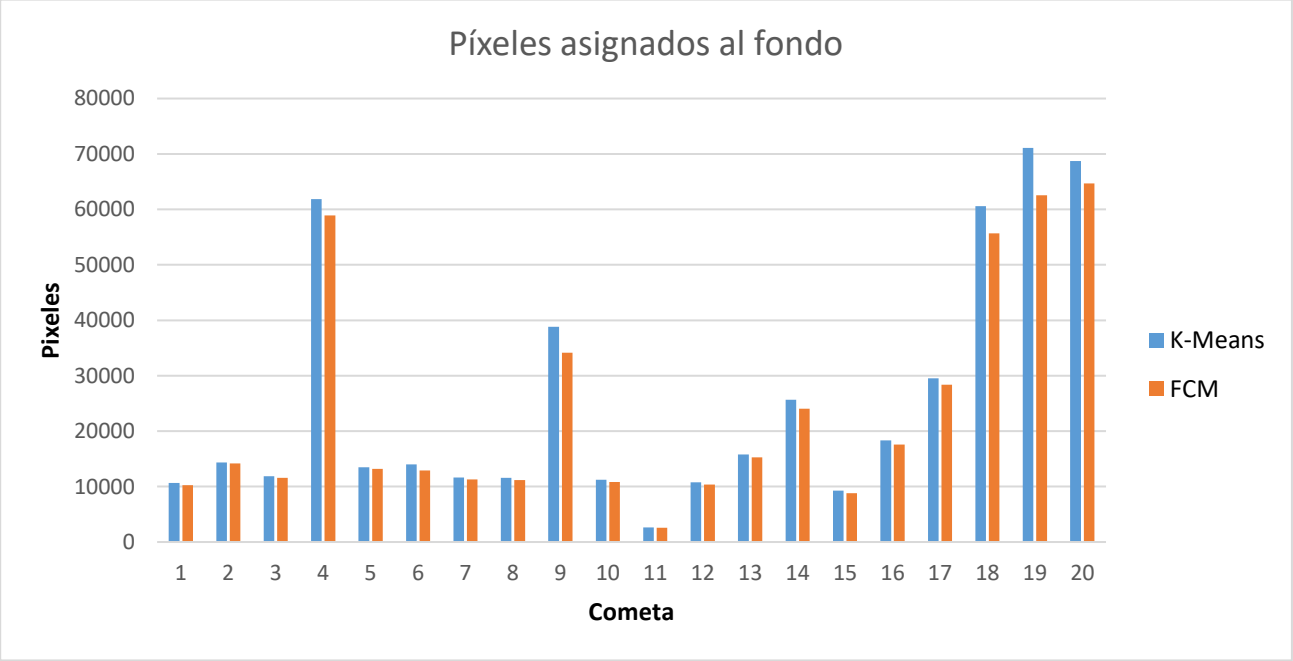

Figura 27 Pixeles que pertenecen a la región del fondo en cada cometa.

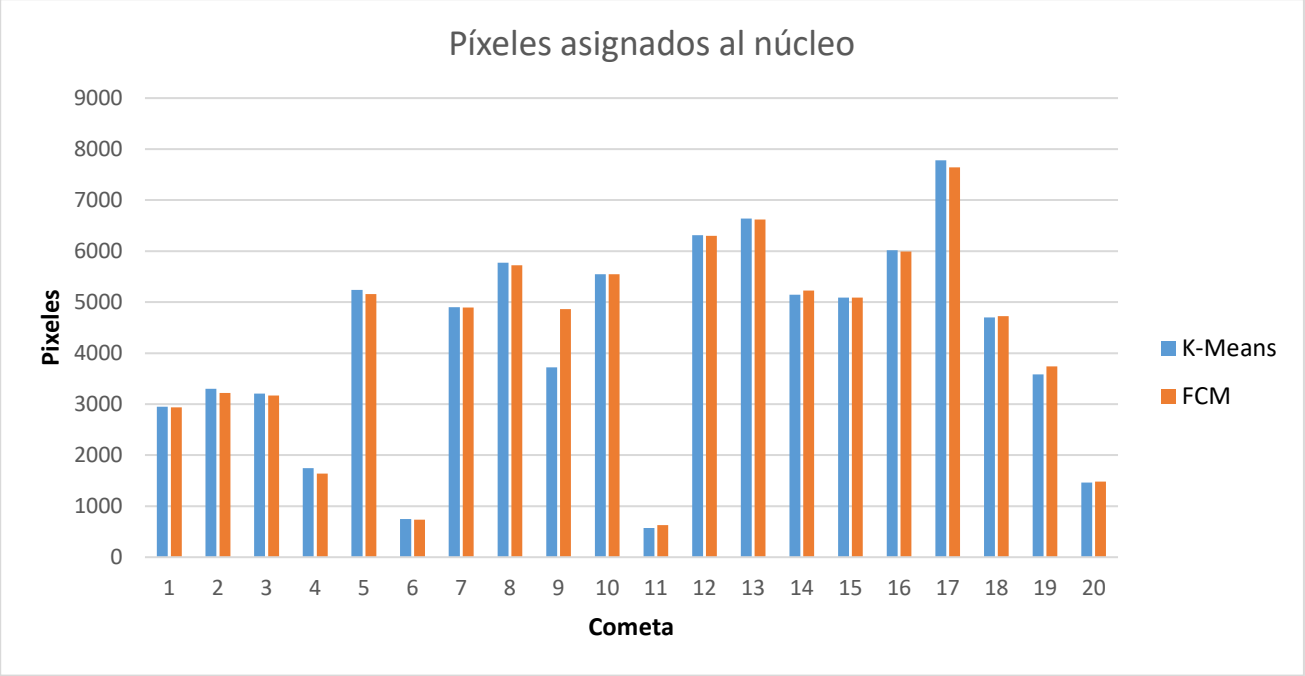

Figura 28 Cantidad de pixeles en el área del núcleo por cometa.

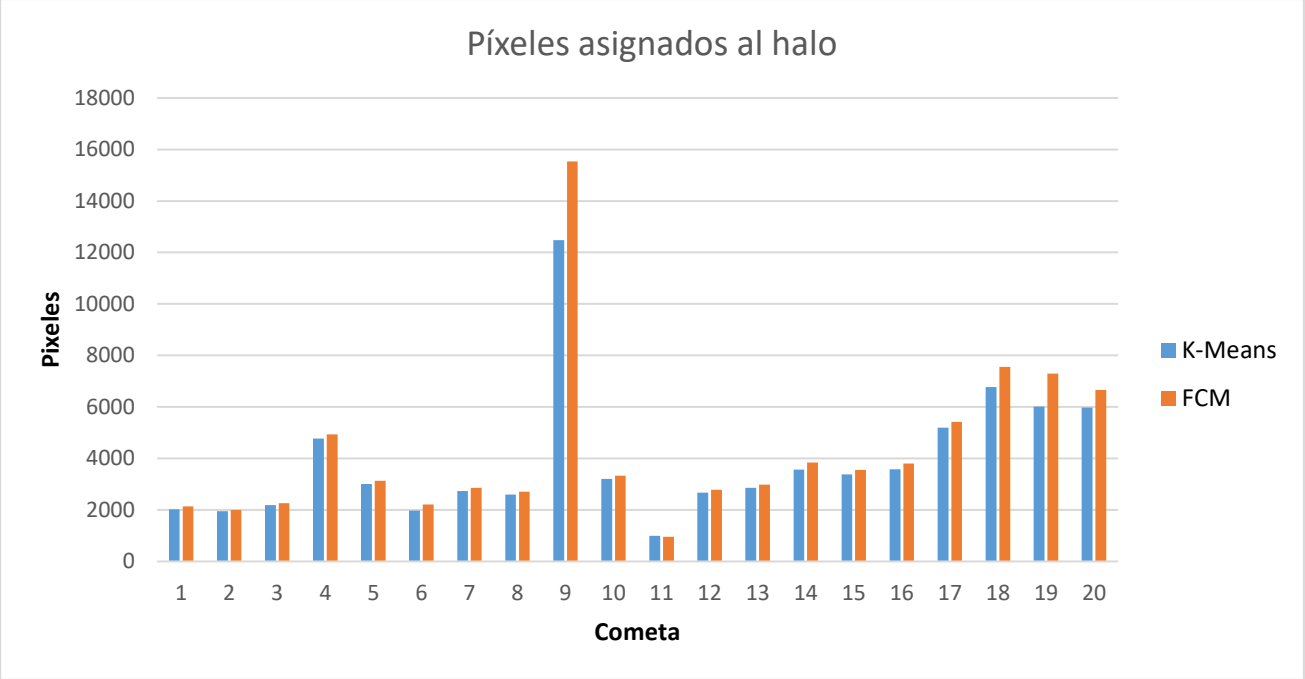

Figura 29 Cantidad de pixeles para el área del Halo.

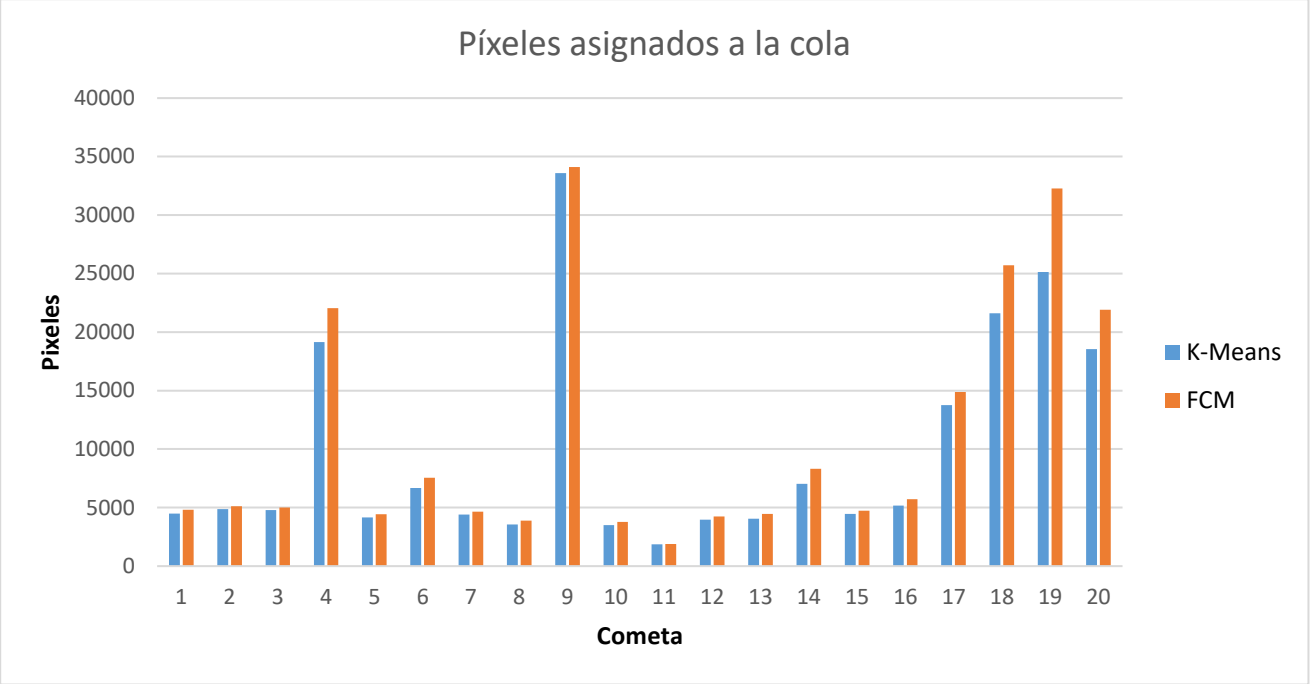

Figura 30 Cantidad de pixeles para el área de la cola del cometa.

Y finalmente, la Figura 31 muestra el tiempo de ejecución en segundos de los algoritmos de clustering. Nuevamente, a simple vista se nota poca diferencia en la asignación de pixeles a las áreas de fondo, núcleo halo y cola de ambos métodos de agrupamiento, mientras que la diferencia en el tiempo de ejecución al segmentar cada cometa es notable.

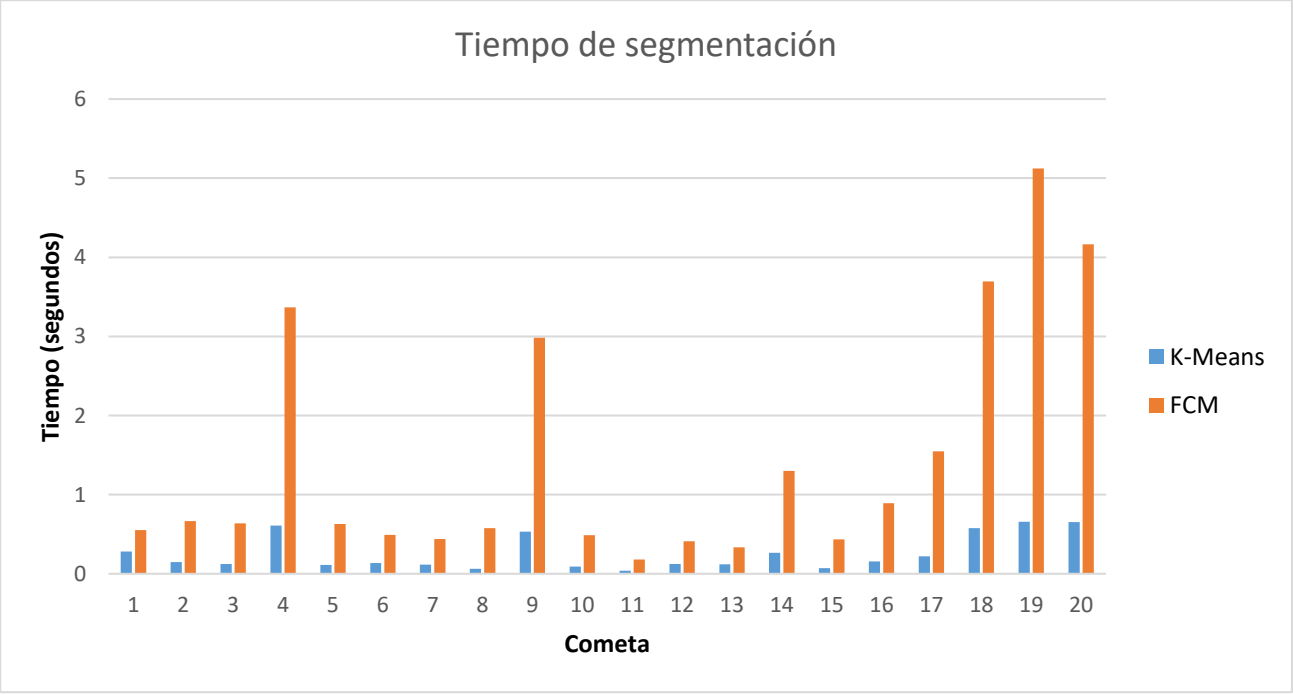

Figura 31 Tiempo en segundos que tarda la segmentación para cada algoritmo.

Los resultados de las pruebas presentadas en el apartado 4.4.1 se encuentran disponibles en: https://github.com/gerardoepitacio/CometAssaySegmentation, siguiendo la siguiente navegación en el árbol de directorios a partir del directorio raíz del repositorio: /img/Conjunto2/bmp/40x/_Resultados2017121.

4.4.2 OpenComet y OpenComet Modificado

Se diseñan las siguientes pruebas para comparar los resultados entre las versiones de OpenComet:

1. Se procesan las 11 imágenes de prueba en ambas versiones del software con los parámetros por default de OpenComet.
2. Se procesan las mismas 11 imágenes para la versión modificada de OpenComet con una configuración que se considera óptima, para demostrar que permitir la modificación de la configuración de parámetros de OpenComet, permite mejorar el proceso de selección de cometas.

Con el fin de comparar también el método de segmentación propuesto, se configuran los parámetros correspondientes y se muestran los resultados generados junto con los resultados de OpenComet.

4.4.2.1 Configuración por default

Los parámetros que se configuran por default en ambas versiones de OpenComet y la propuesta del presente trabajo de investigación se listan en la Tabla 3, obviando que en el método de segmentación propuesto, el *método de* umbralizado es el de umbralizado local y que el parámetro de *segmentación de la cabeza* no aplica.

Tabla 3 Parámetros por defecto de OpenComet Original.

Configuración por default		
Método de umbralizado	=	Huang
Corrección de fondo	=	TRUE
Segmentación de la cabeza	=	Auto
Parámetros de la forma		
Área mínima	<	400
Convexidad mínima	<	0.85
Asimetría Máxima	>	0.5
HRatio Máximo	>	1.05
Diferencia con la línea central		
Máximo para Aislar	>	0.15
Máximo para Grande	>	0.2
Bordes		
Limpiar Bordes	=	TRUE

La Tabla 4 muestra los resultados de sensibilidad y especificidad de las pruebas y se puede verificar que la modificación de OpenComet ya representa una mejora, con sólo

cambiar el método para obtener la imagen en escala de grises que es procesada. Ambas versiones de OpenComet, y el algoritmo de segmentación propuesto tienen dificultades para segmentar imágenes debido a que los valores umbrales para rechazar cometas tienen un rango de error muy pequeño, si estos valores se extendieran daría la posibilidad de rechazar o seleccionar correctamente a mas cometas, esto queda demostrado en los resultados de la experimentación del apartado 4.4.2.2.

Tabla 4 Resultados de sensibilidad y especificidad de los parámetros por default.

Resultados de las pruebas				
		Original	Modificado	Propuesta
VERDADERO	POSITIVO	9	18	10
	NEGATIVO	0	0	0
FALSO	POSITIVO	84	5	12
	NEGATIVO	11	2	10
Tamaño de la muestra		20	20	20
Sensibilidad		**45%**	**90%**	**50%**
Especificidad		**0%**	**0%**	**0%**

4.4.2.2 Configuración en el mejor escenario

Ya que en la versión modificada de OpenComet permite la modificación de los parámetros para la selección de cometas, se realizaron pruebas con distintas configuraciones y se concluye que la configuración mostrada en la Tabla 5 es la configuración que ofrece los mejores resultados para la segmentación de cometas en las imágenes del conjunto de pruebas. Los parámetros que son modificados en la Tabla 5 respecto a la configuración por default son los de Área Mínima y Convexidad. El parámetro de Área Mínima nos sirve para rechazar a ROIs con un área menor a 3000 pixeles y se reduce el valor de Convexidad mínima a 0.5 para poder seleccionar a ROIs que sean menos convexos.

Se realizaron pruebas similares con el método de segmentación propuesto y se concluye que la configuración que ofrece los mejores resultados en la selección de cometas del conjunto de pruebas son las presentadas en la Tabla 6, donde aparte de los parámetros modificados en la Tabla 5, se modifican los parámetros de Asimetria a 0.7, HRatio igual a 1.1 y la diferencia con la línea central a 0.1.

Tabla 5 Parámetros necesarios para obtener un mejor desempeño en la versión modificada de OpenComet.

Alternativa modificado		
Método de umbralizado	=	Huang
Corrección de fondo	=	TRUE
Segmentación de la cabeza	=	Auto
Parámetros de la forma		
Área mínima	<	3000
Convexidad mínima	<	0.5
Asimetría Máxima	>	0.5
HRatio Máximo	>	1.05
Diferencia con la línea central		
Máximo para Aislar	>	0.15
Máximo para Grande	>	0.2
Bordes		
Limpiar Bordes	=	TRUE

Tabla 6 Configuración que ofrece los mejores resultados en el método propuesto.

Configuración Del Método Propuesto		
Método de umbralizado	=	local
Corrección de fondo	=	TRUE
Segmentación de la cabeza	=	N/A
Parámetros de la forma		
Área mínima	<	3000
Convexidad mínima	<	0.5
Asimetría Máxima	>	0.7
HRatio Máximo	>	1.1
Diferencia con la línea central		
Máximo para Aislar	>	NA
Máximo para Grande	>	0.1
Bordes		
Limpiar Bordes	=	TRUE

La Tabla 7 muestra los resultados de las configuraciones presentadas en las tablas 5 y 6, se agregan dos columnas adicionales que muestran los resultados obtenidos por las configuraciones por default propuestas por OpenComet (reportados en la Tabla 4). Se puede apreciar que el proceso de selección y descarte en la versión modificada y la del método propuesto han mejorado respecto a las pruebas realizadas previamente en el apartado 4.4.2.1 ya que han incrementado la selección de verdaderos positivos y se han reducido el número de falsos positivos y falsos negativos.

Tabla 7 Resultados de sensibilidad y especificidad en el mejor escenario.

Resultado de las pruebas					
		OpenComet Modificado		Método Propuesto	
		Default	Óptimo	Default	Óptimo
VERDADERO	POSITIVO	18	19	10	19
	NEGATIVO	0	0	0	1
FALSO	POSITIVO	5	1	12	1
	NEGATIVO	2	1	10	1
Tamaño de la muestra		20	20	20	20
Sensibilidad		**90%**	**95%**	**50%**	**95%**
Especificidad		**0**	**0**	**0**	**50**

Las pruebas y resultados de este apartado se encuentran disponibles en: https://github.com/gerardoepitacio/OpenCometUAI_, en la siguiente navegación del árbol de directorios a partir del directorio raíz del repositorio: /Pruebas/Conjunto2.

4.5 Validación

Se utiliza el test de ANOVA balanceado unidireccional para llevar a cabo la validación, el test de ANOVA prueba la siguiente hipótesis nula: $H0: \mu_1 = \mu_2 = ... = \mu_k$ contra la siguiente hipótesis alternativa: $H1: \mu_1 \neq \mu_2 \neq ... \neq \mu_k$.

Donde μ es igual a la media de un grupo y k el número de grupos.

La validación se realiza sobre los resultados de área y tiempo obtenidos con los algoritmos de agrupamiento K-Means y FCM, estableciendo un nivel de confianza del 95% ($\alpha = 0.05$). Si el resultado la prueba ANOVA es menor al valor de α, si $p < 0.05$, se rechaza la hipótesis nula tomando la hipótesis alternativa como cierta. La hipótesis alternativa nos permite concluir que existe evidencia que la diferencia de las medias entre los grupos es estadísticamente significativa.

La validación se realiza en Matlab con el llamado a la función anova1 que recibe como primer parámetro el conjunto de datos y como segundo parámetro el nombre de los grupos que se están comparando y un tercero para indicar si se desea obtener una gráfica de los resultados. Se compara individualmente el área del fondo, núcleo, halo, cola y tiempo, cada test recibe los datos siguientes.

- Una matriz nx2 donde n representa la cantidad de muestras (cometas), la primera columna contiene los resultados obtenidos del algoritmo K-Means y la segunda columna contiene los resultados obtenidos para el algoritmo FCM para el parámetro que se está comparando.
- El segundo parámetro se determinan los dos grupos para identificar las muestras,

por defecto { ‘K-Means’ y ‘FCM’ } para todas las pruebas realizadas.
- String: ‘on’ u ‘off’ si se desea graficar los resultados, debido a que solo estamos interesados en el valor de p este parámetro se deja en ‘off’.

Los resultados de la prueba y la validación se muestran en la Tabla 8.

Tabla 8 Resultados de la validación a través de ANOVA.

Parámetro	**Valor p**	$p < 0.05$
Fondo	0.8024979	0
Núcleo	0.94299235	0
Halo	0.66426999	0
Cola	0.68857096	0
Tiempo	0.00127238	1

4.6 Discusión

La simplicidad de K-Means y la complejidad de Fuzzy C-Means

Tras la revisión de los resultados plasmados en las Figuras 27 a 30, y la validación a través de ANOVA, se puede verificar que las diferencias en cuanto al área de las regiones segmentadas entre los algoritmos de clustering K-Means y FCM son muy pequeñas, en comparación con el tiempo que toman en ejecutarse. Al promediar el tiempo que tardan ambos algoritmos, resulta que K-Means llega a ejecutarse hasta 5 veces más rápido. Esto nos obliga a plantearnos las siguientes cuestiones.

- ¿Qué tan importante es la exactitud con la que deben segmentarse las regiones del ensayo cometa?
- ¿Vale la pena sacrificar tiempo con el algoritmo FCM solo para sumar ciertos pixeles a una región? Tomemos en cuenta que, a final de cuentas no representan una gran diferencia a las regiones segmentadas por K-Means.
- ¿Es conveniente utilizar un algoritmo difuso cuando no se toman en cuenta los grados de pertenencia de los patrones a las clases?

Para responder a estas cuestiones es necesario adoptar criterios estándar que definan las condiciones sobre los cuales basar la segmentación de las regiones de un cometa. Debido a la similitud de los resultados en segmentar regiones en los algoritmos probados, se considera que, de adoptar una forma para medir la exactitud de la segmentación, ambos algoritmos darían un resultado muy similar, con la ventaja claro en el tiempo de ejecución que provee K-Means respecto a Fuzzy C-Means.

OpenComet modificaciones a un software open source

Las modificaciones propuestas a OpenComet según los resultados de las pruebas en la

sección 4.4.2.2, indican una mejora en la capacidad de segmentación del software, queda a discusión si agregar las opciones de configuración supone más complejidad al sistema o realmente representa una mejora viable. Por una parte, es más complejo, ya que el usuario final, deberá tener conocimientos respecto a los parámetros que se están configurando, por lo tanto, requerirá de una capacitación técnica en el uso de OpenComet. Sin embargo, hay ventajas, ya que ahora para modificar los parámetros para descartar cometas, no requerirán conocimientos ni herramientas de programación.

Imagen en escala de grises

Debido a las múltiples opciones que existen para realizar el proceso de tinción durante la prueba del ensayo cometa, no se puede garantizar que la información útil se localice en una sola capa, como se plantea originalmente en OpenComet, así pues, obtener la imagen en escala de grises basado en el promedio de las tres capas RGB garantiza una mejora en la segmentación, todo esto queda validado en el apartado de experimentación ya que ha sido esta modificación la que ha mejora los resultados en la selección de cometas.

Áreas de cometas y tamaño de las ventanas de filtrado

Llegamos al punto donde existe mayor variación en las imágenes del ensayo cometa: el tamaño de los cometas y la cantidad de pixeles mínimos aproximados que contienen a un cometa, se discute este punto ya que es un parámetro esencial que nos va a evitar obtener un alto número de falsos positivos durante la segmentación.

Muchas veces el equipo microscópico cuenta con las especificaciones técnicas que nos permiten corresponder una longitud con una cantidad de pixeles aproximada, basado en estos parámetros podemos aproximar el área que cubren los pixeles para un cometa con diámetro de aproximadamente 8 micrómetros (tamaño mínimo aproximado de una célula), para el caso del tamaño de las ventanas durante las operaciones de filtrado, deberá calcularse una equivalencia si se manejan imágenes más pequeñas o más grandes respecto a la de 11x11 que se maneja en las pruebas con un objetivo 40x y una definición de la cámara a 1.2 Megapíxeles.

Solidez

Manejar este parámetro como un factor para seleccionar o eliminar cometas no tiene un buen desempeño cuando el núcleo ha sufrido mucho daño ya que las regiones de estos cometas tienen muchas intrusiones en su cerco convexo y por lo tanto tienen una baja solidez y son rechazados para el análisis, se recomienda mantenerlo en un nivel promedio con los cometas con un alto grado de daño o implementar técnicas de morfología matemática como la cerradura para intentar subsanar las intrusiones en el cerco convexo de la región.

Cambiar simetría por Asimetría

Se recomienda cambiar la definición del parámetro de Simetría manejado en OpenComet por Asimetría ya que el valor obtenido cuando es calculado este parámetro parecen describir más el valor asimétrico de una región y no el valor de Simetría.

Desafortunadamente este cambio no se implementó en la versión modificada de OpenComet, queda como propuesta para mejoras a futuro.

OpenComet como un complemento o un software autónomo.

Aparte de las modificaciones mencionadas, en la versión autónoma, se agregan más características como:

- Guardar las configuraciones en un archivo de configuración para que no sea necesario volverlas a establecer cada vez que se utilice el programa.
- Crear nuevas instancias del programa desde el menú, sin necesidad de cerrar el programa para realizar análisis a otro conjunto de imágenes.
- Mejora en la presentación de los datos al llevarse a cabo la depuración (log) del proceso de segmentación, ahora se muestra de manera más amigable al usuario la razón por la cual fue rechazado un ROI, lo que permite junto a la opción de previsualización, recalibrar los parámetros de configuración de manera más rápida.

Las características antes listadas, sumando una ejecución independiente de OpenComet, hacen de las modificaciones a OpenComet obtener una mejor experiencia de usuario en el uso del software.

Procesamiento a imágenes digitalizadas o en tiempo real

Debido a la falta de oportunidad de probar la integración con el análisis en tiempo real con OpenComet en un microscopio, se deja pendiente este apartado para desarrollo en un futuro.

CAPITULO V. CONCLUSIONES Y TRABAJO A FUTURO

Los capítulos de experimentación y discusión nos permiten llegar a las siguientes conclusiones:

1. No es conveniente utilizar un algoritmo difuso para este tipo de segmentación si no se utiliza la correlación que existe entre los patrones respecto a las otras clases.
2. Debido a que no hay un estándar establecido y ampliamente usado para definir los criterios de segmentación de las regiones de las imágenes del ensayo cometa, sólo se puede concluir que no parece haber una diferencia notable respecto a la segmentación de regiones realizada entre los algoritmos K-Means y FCM.
3. En el presente trabajo de investigación, K-Means se presenta como un mejor método de clustering respecto a FCM, ya que, gracias a su simplicidad, el tiempo que toma para ejecutarse llega a ser en algunos casos hasta 5 veces más rápido.
4. Es importante definir o adoptar criterios para la segmentación en las regiones de los cometas o metodologías que nos permitan obtener la cuantificación de daño, ya que esto nos permitiría definir de manera más precisa qué método de clustering utilizar para llevar el prototipo a la construcción de un sistema para el usuario final.
5. Una vez definidos los criterios para la cuantificación de daño en los cometas, valdría la pena realizar pruebas experimentales con el algoritmo FCM involucrando los grados de pertenencia de los patrones sumando una operación de vecindad a fin de obtener resultados que permitan realizar una cuantificación de daño con más exactitud.
6. Debido a los distintos métodos de tinción que existen para la prueba del ensayo cometa, si la imagen es de tipo RGB, no es conveniente utilizar una sola capa para el preprocesamiento y segmentación en las imágenes del ensayo cometa, ya que no se sabe con exactitud si la capa que se está utilizando contenga la suficiente información como para representar todos los componentes de la imagen original.
7. Obtener una imagen en escala de grises simple generada a partir de un promedio de las tres capas RGB garantiza que la mayor parte de la información se mantenga en la imagen, antes de entrar al preprocesamiento.
8. Aunque agregar opciones de configuración a OpenComet parece incrementar la complejidad en el uso del software, se concluye que las propuestas de mejora implementadas en OpenComet permiten al usuario encontrar de manera más sencilla una configuración adecuada para la segmentación de un conjunto de imágenes en específico que hayan resultado de la prueba del ensayo cometa.

Trabajo futuro

A continuación, se proponen algunos experimentos que bien valdrían la pena realizarse en futuros estudios sobre el tema.

- Ampliar el conjunto de pruebas con imágenes de diferentes microscopios, métodos de tinción y tipo de célula.
- Utilizar los resultados obtenidos por los algoritmos de agrupamiento para realizar una cuantificación de daño por alguno de los recomendados en (Venegas, 2009).
- Para el algoritmo de agrupamiento de Fuzzy C-Means utilizar los grados de pertenencia para realizar la segmentación de regiones.
- Implementar un sistema de manejo de errores que permita corregir los errores del algoritmo propuesto al momento de seleccionar falsos positivos o falsos negativos.
- Realizar una validación del algoritmo para encontrar las regiones de un cometa que utiliza OpenComet con los algoritmos utilizados en el presente trabajo de investigación.
- Integrar en la versión autónoma de OpenComet la funcionalidad de procesamiento en tiempo real.

REFERENCIAS

Acharya, T., & Ray, A. K. (2005). *Image Processing: Principles and Applications*. *A Jhon Wiley & Sons, Inc., Pubilcation*. John Wiley & Sons. http://doi.org/10.1117/1.2348895

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027–1035).

Bezdek, J. C. (2013). *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media.

Cooper, G. M., & Hausman, R. E. (2008). *La célula*. Marbán Libros.

Corp, T. (2003). CometScore. T. Corp.

Davies, E. R. (2012). *Computer and machine vision: theory, algorithms, practicalities*. Academic Press.

Doherty, A. T. (2012). The in vitro micronucleus assay. *Genetic Toxicology: Principles and Methods*, 121–141.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification. 2nd*. *Edition. New York*. Wiley-Interscience.

Gonzalez, R. C., & Woods, R. E. (2002). *Digital image processing*. *ed: Prentice Hall Press, ISBN 0-201-18075-8*.

Gonzalez, R. C., Woods, R. E., & Eddins, S. (2009). *Digital Image Processing Using MATLAB*. Gatesmark Publishing.

Gyori, B. M., Venkatachalam, G., Thiagarajan, P. S., Hsu, D., & Clement, M.-V. (2014). OpenComet: an automated tool for comet assay image analysis. *Redox Biology*, *2*, 457–465.

Hartl, D. L., Jones, E. W., & others. (1998). *Genetics: Principles and Analysis 4th ed*. Jones and Bartlett publishers.

Jain, A. K. (1989). *Fundamentals of digital image processing*. Prentice-Hall, Inc.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666.

Karp, G. (2011). *Biologia celular y molecular: conceptos y experimentos (6a)*. McGraw Hill Mexico.

Końca, K., Lankoff, A., Banasik, A., Lisowska, H., Kuszewski, T., Góźdź, S., … Wojcik, A. (2003). A cross-platform public domain PC image-analysis program for the comet assay. *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, *534*(1), 15–20.

Morales, D. L. (2010). *Polimorfismos en el gen pon1 y daño al dna en poblacion expuesta ocupacionalmente a plaguicidas organofosforados*. UAGro.

OMS. (2008). ¿Aumenta o disminuye el número de casos de cáncer en el mundo? Retrieved November 25, 2016, from http://www.who.int/features/qa/15/es/

Ostling, O., & Johanson, K. J. (1984). Microelectrophoretic study of radiation-induced DNA damages in individual mammalian cells. *Biochemical and Biophysical Research Communications*, *123*(1), 291–298.

Rivest, J.-F., Tang, M., McLean, J., & Johnson, F. (1996). Automated measurements of tails in the single cell gel electrophoresis assay. In *Instrumentation and Measurement Technology Conference, 1996. IMTC-96. Conference Proceedings. Quality Measurements: The Indispensable Bridge between Theory and Reality., IEEE* (Vol. 1, pp. 111–114).

Robertis, E. de, & Hib, J. (2004). *Fundamentos de biología celular y molecular de De*

Robertis. Editorial El Ateneo.
Sansone, M., Zeni, O., & Esposito, G. (2012). Automated segmentation of comet assay images using Gaussian filtering and fuzzy clustering. *Medical & Biological Engineering & Computing*, *50*(5), 523–532.
Sestili, P., & Cantoni, O. (1999). Osmotically driven radial diffusion of single-stranded DNA fragments on an agarose bed as a convenient measure of DNA strand scission. *Free Radical Biology and Medicine*, *26*(7–8), 1019–1026. http://doi.org/10.1016/S0891-5849(98)00290-1
Singh, N. P., McCoy, M. T., Tice, R. R., & Schneider, E. L. (1988). A simple technique for quantitation of low levels of DNA damage in individual cells. *Experimental Cell Research*, *175*(1), 184–191.
Smolka, B., & Lukac, R. (2004). Segmentation of the comet assay images. In *International Conference Image Analysis and Recognition* (pp. 124–131).
Venegas, L. a Z. (2009). *Optimizaciones metodológicas del ensayo del cometa y su aplicación en biomonitorización humana*. *Tesis doctoral*. Universidad Autónoma De Barcelona.
Wu, Q., Merchant, F. A., & Kenneth, R. (2008). Castleman, Microscope Image Processing. Academic Press, United States of America.
Ximena C. Abrevaya. (2008). ¿Qué es la genotoxicidad? Retrieved August 30, 2016, from http://www.intramed.net/contenidover.asp?contenidoID=47111
Young, R. R. (2002). Genetic toxicology: web resources. *Toxicology*, *173*(1), 103–121.

LISTA DE TABLAS Y FIGURAS

ANEXOS

ANEXO 1 Código fuente en lenguaje m del método propuesto.

Disponible también en: https://github.com/gerardoepitacio/CometAssaySegmentation

Archivo: EnsayoCometa_k_means_FCM.m

```
% Script principal que lleva a cabo el proceso de segmentacion de imagenes
% del ensayo cometa con los algoritmos K-Means y Fuzzy C-Means.
clear
clc
clear all
close all
% Quitamos advertencia: Image is too big to fit on screen
warning('off', 'Images:initSize:adjustingMag');

global NombresArchivo RutaPrincipal RutaResultados;
global Fila FilaFCM FilaKMeans CometasProcesados ;
global Excel ExcelKMeans ExcelFCM;

[NombresArchivo, RutaPrincipal] = ReadFile();
if RutaPrincipal == 0
    return;
end

InicializarEntorno();

for File = 1 : length(NombresArchivo)
    %% Preprocesamiento
    [Ruta, Archivo] = ObtenerArchivo(File);
    ImagenActual = imread(Ruta);
    [Filtrada, Gris] = PreprocesarImagen(ImagenActual);

    %% Binarizado y remocion.
    tsd = localthresh(Filtrada, ones(3), 1, 1.1, 'global');
    bwW = bwareaopen(tsd, 3000);
    bw = imclearborder(bwW);

    %% Descripcion y descarte
    s = DescartarCometas(ImagenActual, bw, Archivo);

    %% Subimagen y Patrones
    box = cat(1, s.BoundingBox);
    [CantidadCometas, col] = size(box);
    Excel(Fila, 1) = {Archivo};
    ExcelFCM(FilaFCM, 1) = {Archivo};
    ExcelKMeans(FilaKMeans, 1) = {Archivo};
    for p = 1 : CantidadCometas
        CometasProcesados = CometasProcesados + 1;
        Excel(Fila, 2) = {p};
        Excel(Fila, 3) = {CometasProcesados};
        %% Obtener Patrones
        subImagen = imcrop(Gris, (box(p,:)-1));
        [m, n, c] = size(subImagen);
        Patrones = ObtenerPatrones(subImagen, m, n);
        %% K-MEANS
        tic
        [cidx, ctrs] = kmeans(Patrones, 4);
        Tiempo = toc;
        [RegionesKMeans, Areas] = ObtenerKRegiones(ctrs, cidx, m, n);
        %% Guarda Resultados K-MEANS
        Excel(Fila, 4:9) = Areas;
```

```
        Excel(Fila, 10) = {Tiempo};
        ExcelKMeans(FilaKMeans, 2:10) = Excel(Fila, 2:10);
        Fila = Fila + 1;
        FilaKMeans = FilaKMeans + 1;
        %% FUZZY C-MEANS
        opts = [nan;nan;nan;0];
        tic
        [centers, U, obj_fcn] = fcm(Patrones, 4, opts);
        Tiempo = toc;
        [RegionesFCM, Areas]= ObtenerRegiones(U, centers, m, n);
        %% Guarda Resultados FCM
        Excel(Fila, 2) = {p};
        Excel(Fila, 3) = {CometasProcesados};
        Excel(Fila, 4:9) = Areas;
        Excel(Fila, 10) = {Tiempo};
        ExcelFCM(FilaFCM, 2:10) = Excel(Fila, 2:10);
        Fila = Fila + 1;
        FilaFCM = FilaFCM + 1;
        %% Mostrar y guardar resultados
        hold on
        rectangle('Position', box(p,:), 'EdgeColor', 'green');
        text(box(p,1)+7, box(p,2)+20, sprintf('%d', CometasProcesados), ...
            'Color', 'green', 'FontSize', 14);
        hold off
        set(gca,'position',[0 0 1 1],'units','normalized');
        print([RutaResultados Archivo], '-dbmp');

        ExportarImagen(RutaResultados, ['KMeans' ...
            num2str(CometasProcesados)], RegionesKMeans);
        ExportarImagen(RutaResultados, ['FCM' ...
            num2str(CometasProcesados)], RegionesFCM);
        %% exportar patrones del cometa.
        ExportarPatrones(RutaResultados, ['Patrones' ...
            num2str(CometasProcesados)], Patrones);
    end
    if ischar(NombresArchivo) == 1
        break;
    end
end
[pFondo, pNucleo, pHalo, pCola, pTiempo] = AnalisisAnova()
ExportarDatos();
```

Archivo: AnalisisAnova.m

```
function [pFondo, pNucleo, pHalo, pCola, pTiempo] = AnalisisAnova()
% Funcion AnalisisAnova: Realiza el analisis de datos a traves de ANOVA
% unidireccional. Los datos son tomados desde las variables globales
% despues de terminar la tarea de segmentacion.
%
% Donde:
% ExcelKMeans       Es una matriz nx10 que contiene todos los datos
%                   resultantes de la segmentacion de regiones del cometa
%                   con K-Means, donde n es la cantidad de cometas
%                   procesados.
% ExcelFCM          Es la matriz que guarda todos los datos obtenidos por
%                   Fuzzy C-Means.
global ExcelKMeans ExcelFCM CometasProcesados RutaResultados;
FilaFin = CometasProcesados + 2;

DatosFondo = [ExcelKMeans(3:FilaFin, 6) ExcelFCM(3:FilaFin, 6)];
DatosNucleo = [ExcelKMeans(3:FilaFin, 7) ExcelFCM(3:FilaFin, 7)];
```

```
DatosHalo = [ExcelKMeans(3:FilaFin, 8) ExcelFCM(3:FilaFin, 8)];
DatosCola = [ExcelKMeans(3:FilaFin, 9) ExcelFCM(3:FilaFin, 9)];
DatosTiempo = [ExcelKMeans(3:FilaFin, 10) ExcelFCM(3:FilaFin, 10)];

pFondo = anova1(cell2mat(DatosFondo(:, :)), {'K-MEANS', 'FCM' }, 'off');
pNucleo = anova1(cell2mat(DatosNucleo(:, :)), {'K-MEANS', 'FCM' }, 'off');
pHalo = anova1(cell2mat(DatosHalo(:, :)), {'K-MEANS', 'FCM' }, 'off');
pCola = anova1(cell2mat(DatosCola(:, :)), {'K-MEANS', 'FCM' }, 'off');
pTiempo = anova1(cell2mat(DatosTiempo(:, :)), {'K-MEANS', 'FCM' }, 'off');
Destino = [RutaResultados 'ResultadosAnova.xls'];
if exist(Destino, 'file') == 2
    delete(Destino);
end
Validacion(1, 1:10) = {'pFondo', pFondo, 'pNucleo', pNucleo, ...
    'pHalo', pHalo, 'pCola', pCola, 'pTiempo', pTiempo};
Validacion(2, 1:10) = {'K-MEANS', 'FCM', 'K-MEANS', 'FCM', 'K-MEANS', ...
    'FCM', 'K-MEANS', 'FCM', 'K-MEANS', 'FCM'};
Validacion(3:(CometasProcesados+2), 1:10) = ...
    [DatosFondo(:, :) DatosNucleo(:, :) DatosHalo(:, :) ...
    DatosCola(:, :) DatosTiempo(:, :)];
xlswrite(Destino, Validacion);
end
```

Archivo: DescartarCometas.m

```
function Propiedades = DescartarCometas(ImagenRGB, ImgBinaria, Archivo)
% Funcion DescartarCometas: Obtiene los descriptores de la imagen binaria
% y descarta los descriptores que no cumplen con los parametros
% establecidos para ser aceptados como cometas.
%
% Donde:
% ImagenRGB:  Es la imagen original RGB
% ImgBinaria: Es la imagen binaria generada despues del preprocesamiento
% Archivo:    Es el nombre del archivo que actualmente se esta procesando.
    Propiedades = regionprops(ImgBinaria,  'Area', 'BoundingBox', ...
        'Image', 'Solidity');
    figure('NumberTitle', 'off', 'Name', Archivo), imshow(ImagenRGB);
    box = cat(1, Propiedades.BoundingBox);
    ROIs = size(box);
    hold on
    for i = 1 : ROIs
        rectangle('Position', box(i,:), 'EdgeColor', 'red');
    end
    hold off
    fprintf('\n');
    Propiedades = GetComets(Propiedades, Archivo);
end
```

Archivo: ExportarDatos.m

```
function ExportarDatos()
% Funcion ExportarDatos: Exporta los datos generados durante el proceso de
% segmentacion a un archivo excel llamado Resultados.xls que contiene los
% siguientes datos:
%
% 1.- El conjunto de datos donde por cada cometa la primer fila contiene
% los resultados generados por el algorimo de clustering K-Means y la
```

```
% segunda por el algoritmo Fuzzy C-Means.
% 2.- El mismo conjunto de datos anterior, solo que se separan los
% resultados de K-Means y Fuzzy C-Means en dos apartados.
% 3.- El conjunto de datos formateados de tal manera que facilite el
% analisis a traves del test de ANOVA.
global Excel ExcelKMeans ExcelFCM CometasProcesados RutaResultados;
FilaFin = CometasProcesados + 2;
Validacion(1, 1:10) = {'Fondo', '', 'Nucleo', '', 'Halo', '', 'Cola', ...
    '', 'Tiempo', ''};
Validacion(2:FilaFin, 1:10) = [...
    ExcelKMeans(2:FilaFin, 6) ExcelFCM(2:FilaFin, 6) ...
    ExcelKMeans(2:FilaFin, 7) ExcelFCM(2:FilaFin, 7) ...
    ExcelKMeans(2:FilaFin, 8) ExcelFCM(2:FilaFin, 8) ...
    ExcelKMeans(2:FilaFin, 9) ExcelFCM(2:FilaFin, 9) ...
    ExcelKMeans(2:FilaFin, 10) ExcelFCM(2:FilaFin, 10)];
Resultados = [Excel ; ExcelKMeans ; ExcelFCM ; Validacion];

if exist([RutaResultados 'Resultados.xls'], 'file') == 2
  delete([RutaResultados 'Resultados.xls']);
end
xlswrite([RutaResultados 'Resultados'], Resultados);
end
```

Archivo: ExportarImagen.m

```
function ExportarImagen(FolderDestino, NombreArchivo, MatrizRGB)
% Funcion ExportarImagen: Guarda la imagen recibida en la MatrizRGB en el
% subfolder "Cometas" del folder destino. El archivo es guardado en
% formato png con el nombre recibido en el segundo parametro.
FolderDestino = [FolderDestino 'Cometas\'];
if exist(FolderDestino, 'dir') ~= 7
    mkdir(FolderDestino);
end
if exist([FolderDestino NombreArchivo '.png'], 'file') == 2
    delete([FolderDestino NombreArchivo '.png']);
end
imwrite(MatrizRGB, [FolderDestino NombreArchivo '.png']);
end
```

Archivo: ExportarPatrones.m

```
function ExportarPatrones(FolderDestino, NombreArchivo, Patrones)
% Funcion ExportarPatrones: Exporta a un archivo CSV los patrones
% generados por cada cometa con el llamado a la funcion ObtenerPatrones.
FolderDestino = [FolderDestino 'Patrones\'];
if exist(FolderDestino, 'dir') ~= 7
    mkdir(FolderDestino);
end

if exist([FolderDestino NombreArchivo '.csv'], 'file') == 2
    delete([FolderDestino NombreArchivo '.csv']);
end

csvwrite([FolderDestino NombreArchivo '.csv'], Patrones);
end
```

Archivo: GetAsymmetry.m

```
function Asymmetry = GetAsymmetry(Roi)
% Funcion GetAsymmetry: Devuelve el valor asimetrico de la region.
%
% Donde:
% Roi: Es es el arreglo de descriptores generadas por, Roi = Stats(i).
ImgRoi = Roi.Image;
[rows cols] = size(ImgRoi);
absdy = 0.0;
xFrontCentroid = GetXFrontCentroid(ImgRoi);
for i = 1 : cols
    Sum1 = 0.0;
    Sum2 = 0.0;
    cnt1 = 0;
    for j = 1 : rows
        if ImgRoi(j,i) ~= 0
            if j > xFrontCentroid
                Sum1 = Sum1 + 1;
            else
                Sum2 = Sum2 + 1;
            end
            cnt1 = cnt1 + 1;
        end
    end
    absdy = absdy + (abs(Sum2 - Sum1) / cnt1);
end
Asymmetry = absdy / cols;
```

Archivo: GetComets.m

```
function Comets = GetComets(Stats, FileName)
% Funcion GetComets: Devuelve los cometas que son validos para ser
% procesados como cometas.
%
% Donde:
% Stats:    Son las estadisticas generadas por la funcion regionprops
% FileName: Es el nombre del archivo de donde se obtuvieron las
%           estadisticas recibidas.
[Elements, cols] = size(Stats);
Invalid = false(Elements, 1);
disp(['Archivo: ', FileName])
str = sprintf('%5s%10s%10s%10s%10s%10s%10s','#', 'Solidity', ...
    'Asymmetry', 'Hratio', 'CLD', 'Area', 'Valid');
disp(str);
for i = 1 : Elements
    Roi = Stats(i);
    [height, width] = size(Roi.Image);
    Log = sprintf('%5d',i);
    Valid = ' ';
     if Roi.Solidity < 0.5
        Invalid(i) = true;
        Valid = '*';
    end
    Log = [Log sprintf('%9.4f%1s', Roi.Solidity, Valid)];

    Valid = ' ';
```

```
    Asymmetry = GetAsymmetry(Roi);
     if Asymmetry > 0.7
        Invalid(i) = true;
        Valid = '*';
    end
    Log = [Log sprintf('%9.4f%1s', Asymmetry, Valid)];

    Valid = ' ';
    if (height / width) > 1.1    %Hratio
        Invalid(i) = true;
        Valid = '*';
    end
    Log = [Log sprintf('%9.4f%1s', (height / width), Valid)];

    Valid = ' ';
    CLD = abs(GetXFrontCentroid(Roi.Image) - height/2)/height;
    if CLD > 0.1
        Invalid(i) = true;
        Valid = '*';
    end
    Log = [Log sprintf('%9.4f%1s', CLD, Valid)];
    Log = [Log sprintf('%10.2f%', Roi.Area)];
    if Invalid(i) == true
        Log = [Log sprintf('%10s', 'Invalid')];
    else
        Log = [Log sprintf('%10s', 'Valid')];
    end
    disp(Log);
end
Stats(Invalid) = [];
Comets = Stats;
```

Archivo: GetXFrontCentroid.m

```
function xCentroid = GetXFrontCentroid(ImgRoi)
% Funcion GetXFrontCentroid: Devuelve el centroide frontal de la imagen
% del ROI.
%
% Donde:
% ImgRoi: Es la imagen binaria del ROI, ImgRoi = Roi.Image
xFrontCentroid = 0.0;
cnt = 0;
[rows cols] = size(ImgRoi);
for i = 1 : (cols*0.3)
    for j = 1 : rows
        if ImgRoi(j, i) ~= 0
            xFrontCentroid = xFrontCentroid + j;
            cnt = cnt + 1;
        end
    end
end
xCentroid = xFrontCentroid / cnt;
```

Archivo: InicializarEntorno.m

```
function InicializarEntorno()
% Funcion InicializarEntorno: Realiza la inicializacion de variables
% globales para ser utilizadas durante el proceso de segmentacion.
% Crea la carpeta "Resultados" donde se almacenaran los resultados.
global CometasProcesados Fila FilaFCM FilaKMeans;
global Excel ExcelKMeans ExcelFCM;
global RutaPrincipal RutaResultados;
CometasProcesados = 0;
Fila = 2;
FilaFCM = 3;
FilaKMeans = 3;
Excel = cell(0);
ExcelKMeans = cell(0);
ExcelFCM = cell(0);
Excel = {'Imagen','#', '#Img', 'Algoritmo', 'Pixeles', ...
    'Fondo', 'Núcleo', 'Halo', 'Cola', 'Tiempo'};
ExcelKMeans(2, 1:10) = {'RESULTADOS K-Means', '#', '#', 'Algoritmo', ...
     'K-Means', 'K-Means', 'K-Means', 'K-Means', 'K-Means', 'K-Means'};
ExcelFCM(2, 1:10) = {'RESULTADOS FCM','#', '#', 'Algoritmo', 'FCM', ...
     'FCM', 'FCM', 'FCM', 'FCM', 'FCM'};
format shortg
c = clock;
fecha = [num2str(c(1)) num2str(c(2)) num2str(c(3))];
RutaResultados = [RutaPrincipal 'Resultados' fecha '\'];
if exist(RutaResultados, 'dir') ~= 7
    mkdir(RutaResultados);
end
end
```

Archivo: localmean.m

```
function mean = localmean(f, nhood)
% LOCALMEAN Computes an array of local means.
% MEAN = LOCALMEAN(F, NHOOD) computes the mean at the center of
% every neighborhood of F defined by NHOOD, an array of zeros and
% ones where the nonzero elements specify the neighbors used in the
% computation of the local means. The size of NHOOD must be odd in
% each dimension; the default is ones(3). Output MEAN is an array
% the same size as F containing the local mean at each point.
if nargin == 1
    nhood = ones(3) / 9;
else
    nhood = nhood / sum(nhood(:));
end
mean = imfilter(im2double(f), nhood, 'replicate');
```

Archivo: localthresh.m

```
function g = localthresh(f, nhood, a, b, meantype)
% LOCALTHRESH Local thresholding.
% G = LOCALTHRESH(F, NHOOD, A, B, MEANTYPE) thresholds image F by
% computing a local threshold at the center,(x, y), of every
% neighborhood in F. The size of the neighborhoods is defined by
% NHOOD, an array of zeros and ones in which the nonzero elements
% specify the neighbors used in the computation of the local mean
```

```
% and standard deviation. The size of NHOOD must be odd in both
% dimensions.
%
% The segmented image is given by
%
%       1   if (F > A*SIG) AND (F > B*MEAN)
%   G=
%       0   otherwise
%
% where SIG is an array of the same size as F containing the local
% standard deviations. If MEANTYPE = 'local' (the default), then
% MEAN is an array of local means. If MEANTYPE = 'global', then
% MEAN is the global (image) mean, a scalar. Constants A and B
% are nonnegative scalars.

% Intialize.
f = im2double(f) ;
% Compute the local standard deviations.
SIG = stdfilt(f, nhood);
% Compute MEAN.
if nargin == 5 && strcmp(meantype, 'global')
MEAN = mean2(f) ;
else
MEAN = localmean(f, nhood); % This is a custom function.
end
% Obtain the segmented image.
g = (f > a*SIG) & (f > b*MEAN);
```

Archivo: ObtenerArchivo.m

```
function [Ruta, Nombre] = ObtenerArchivo(Indice)
% Funcion ObtenerArchivo: Lee el archivo segun el indice recibido, si
% NombresArchivo no es un arreglo, solo se concatena la ruta base al
% nombre del archvo.
%
% Donde:
% Indice:   De donde se obtiene el archivo indica el indice del archivo
%           del arreglo de archivos leidos.
global NombresArchivo RutaPrincipal;
    if ischar(NombresArchivo) == 0
        Nombre = NombresArchivo{Indice};
        Ruta = [RutaPrincipal Nombre];
    else
        Nombre = NombresArchivo;
        Ruta = [RutaPrincipal NombresArchivo];
    end
end
```

Archivo: ObtenerClase.m

```
function Clase = ObtenerClase(ColumnaPertenencia)
% ObtenerClase: Devuelve la clase a la que pertenece el patron k, basado
% en el grado de pertenencia mas alto que se tenga el patron con todas
% las clases.
%
% Donde:
```

```
% ColumnaPertenencia: Es un vector [1-C] con los grados de pertenencia en
% cada clase y C es el numero de clases.
[~, Clase] =  max(ColumnaPertenencia);
end
```

Archivo: ObtenerKClase.m

```
function Clase = ObtenerKClase(KIdx, VectorBrilloCentral, VBCentralOrdenado)
% Funcion ObtenerKClase: Devuelve el numero de region a la que pertenece
% el KIdx respecto al valor de brillo del centroide de la clase.
%
% Donde:
% KIdx:                Es el indice de la clase a la que pertenece el K
%                      patron.
% VectorBrilloCentral: Vector con la columna de brillos de los centroides
% VBCentralOrdenado:   Vector con la columna de brillos de los centroides
%                      ordenado
ValorCentral = VectorBrilloCentral(KIdx,1);
Clase = find(VBCentralOrdenado == ValorCentral);
end
```

Archivo: ObtenerKRegiones.m

```
function [Img, Areas] = ObtenerKRegiones(ctrs, cidx, m, n)
% ObtenerKRegiones: Devuelve las regiones de una imagen a partir del
% vector de indices generado por el algoritmo de clustering K-Means.
%
% Donde:
% ctrs:     Es la matriz de centroides de cada clase.
% cidx:     Es el vector que contiene los indices de clase a los que
% pertenece cada patron.
% m:        Numero de filas de la subimagen.
% n:        Numero de columnas en la subimagen.
%
% La columna de brillo de cada vector centroide es obtenida y ordenada
% para poder corresponder un indice de clase a un valor de brillo y en
% consecuencia a una region.
    VectorBrillo = ctrs(:, 1);
    VectorBrilloOrd = sortrows(VectorBrillo, 1);
    Img = uint8(zeros(m, n, 3));
    k = 1;

    Total = m*n;
    BG = 0;
    Nucleo = 0;
    Halo = 0;
    Cola = 0;
    for i = 1 : m
        for j = 1 : n
           KIdx = cidx(k,1);
           Clase = ObtenerKClase(KIdx, VectorBrillo, VectorBrilloOrd);
          switch Clase
              case 1
                  Img(i,j,:) = [0 0 0];
                  BG = BG + 1;
              case 2
```

```
                Img(i,j,:) = [0 255 0];
                Cola = Cola + 1;
            case 3
                Img(i,j,:) = [0 0 255];
                Halo = Halo + 1;
            case 4
                Img(i,j,:) = [255 0 0];
                Nucleo = Nucleo + 1;
        end
          k=k+1;
      end
  end
  Areas = {'K-Means', Total, BG, Nucleo, Halo, Cola};
end
```

Archivo: ObtenerPatrones.m

```
function Patrones = ObtenerPatrones(Imagen, h, w)
% Funcion ObtenerPatrones: Devuelve los patrones de la imagen en escala
% de grises.
%
% Donde:
% Imagen    Es una imagen en escala de grises.
% h         Es la altura o la cantidad de filas de la imagen.
% w         Es el ancho de la imagen o la cantidad de columnas.
%
% Patrones  Es una matriz (h*w)x3 de la siguiente forma
%
% [ValorDeBrillo     11x11std     11x11mean]
%                .
%                .
%                .
  Imagen = im2double(Imagen);
  Imagen = imadjust(Imagen, [min(Imagen(:)), max(Imagen(:))], [0 1]);
  stdImg = stdfilt(Imagen, ones(11, 11));
  tam = h*w;
  Img = double(zeros(h, w));
  Respuesta = padarray(Imagen,[5 5], 0);
  [h, w] = size(Respuesta);
  Patrones = zeros(tam, 3);
  k = 1;

  for i = 1 : h-10,
      for j = 1 : w-10,
          SumM = Respuesta(i:i+10,j:j+10);
          Patrones(k, 1) = Respuesta(i+5,j+5);
          Patrones(k, 2) = stdImg(i,j);
          Patrones(k, 3) = mean2(SumM);
          Img(i, j) = Patrones(k, 3);
          k = k+1;
      end
  end
end
```

Archivo: ObtenerRegiones.m

```
function [Img, Areas] = ObtenerRegiones(Clases, Centro,m, n)
% Funcion ObtenerRegiones: Devuelve las regiones de una imagen a partir
% de la matriz de pertenencia y la matriz de centroides devueltos del
% algoritmo de clustering Fuzzy C-Means.
%
% Donde:
% Clases: Es la matriz de grados de pertenencia.
% Centro: Es la matriz que contiene los patrones centroides.
% m:      Numero de filas de la imagen.
% n:      Numero de columnas en la imagen.
%
% La matriz de clases es ordenada respecto al valor de brillo de la
% matriz de centroides, para asegurarse que las clases devueltas se
% obtienen en funcion del brillo, es decir a menor clase, menor brillo
% tiene la imagen.
    Img = uint8(zeros(m,n,3));
    k = 1;
    Clases(:,m*n+1)= Centro(:,1);
    B = sortrows(Clases,m*n+1);
    B(:,m*n+1) = [];

    Total = m*n;
    BG = 0;
    Nucleo = 0;
    Halo = 0;
    Cola = 0;
    for i = 1 : m
        for j = 1 : n
           Clase =  ObtenerClase(B(:,k));
          switch Clase
           case 1
              Img(i,j,:) = [0 0 0];
              BG = BG + 1;
           case 2
            Img(i,j,:) = [0 255 0];
            Cola = Cola + 1;
          case 3
             Img(i,j,:) = [0 0 255];
             Halo = Halo + 1;
          case 4
             Img(i,j,:) = [255 0 0];
             Nucleo = Nucleo + 1;
          end
            k=k+1;
        end
    end
    Areas = {'FCM', Total, BG, Nucleo, Halo, Cola};
end
```

Archivo: PreprocesarImagen.m

```
function [Filtrada, Gris] = PreprocesarImagen(ImagenRGB)
% Funcion PreprocesarImagen: Realiza el proceso de filtrado a la imagen
% RGB recibida.
Gris = rgb2gray(ImagenRGB);
Filtrada = imfilter(Gris,fspecial('average', 11));
end
```

Archivo: ReadFile.m

```
function [FileName, FilePath] = ReadFile()
% Funcion ReadFile: Muestra una ventana para seleccionar la o las imagenes
% que van a ser procesadas.
Filter={'*.jpg;*.jpeg;*.png;*.tif;*.bmp'};
Title='Selecciona una imagen';
[FileName, FilePath] = uigetfile(Filter, Title, 'MultiSelect', 'on');
pause(0.01);
```