UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

FACULTAD DE INGENIERÍAS

CARRERA DE INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del Título de

Ingeniería de Sistemas

TÍTULO:

"RECUPERACIÓN DE IMÁGENES MEDIANTE EXTRACCIÓN DE BLOBS APLICANDO EL OPERADOR LAPLACIANO DE GAUSS Y EL KERNEL GAUSSIANO Y DESARROLLO DE UN PROTOTIPO."

AUTORES:

Alba Raquel Paguay Paguay
Pedro Remigio Urgilés Ortíz

DIRECTOR:

Ing. Vladimir Robles.

Cuenca, Febrero del 2012

DECLARACIÓN DE RESPONSABILIDAD

Los conceptos desarrollados en este trabajo, así como todo el estudio e implementación de este Proyecto, son de exclusiva responsabilidad de los Autores. Los textos de otros autores llevan su correspondiente cita bibliográfica.

Cuenca, 9 de febrero del 2012

Alba Raquel Paguay Paguay.

Pedro Remigio Urgilés Ortíz.

Ing. Vladimir Robles

CERTIFICA

Haber dirigido y revisado prolijamente cada uno de los capítulos del informe de tesis, realizado por la Srta. Alba Raquel Paguay Paguay y el Sr. Pedro Remigio Urgilés Ortíz, y por cumplir los requisitos autorizo su presentación.

Cuenca, febrero del 2012

Ing. Vladimir Robles B.
DIRECTOR DE TESIS

Dedicatoria

Dedico esta tesis a mi madre Carmen Paguay que siempre me ha protegido y cuidado aunque no la he podido ver físicamente he sentido su presencia junto a mí.

De manera especial a mi tía Elvia Paguay que ha sido mi apoyo incondicional, mi guía, que más que una tía ha sido como una segunda madre aquí en la tierra.

A mis tíos Rosario y Benjamín, que han confiado en mí y me han brindado su apoyo para culminar mi carrera.

Gracias a todos ellos por brindarme todo lo que han podido y así permitirme lograr una meta más en mi vida.

Alba Raquel Paguay Paguay

Dedicatoria

Este trabajo de grado lo dedico con todo mi cariño a mis queridos padrinos: Ingeniero Marcelo Espín y Doctora Milyn Andrade, porque gracias a sus sabios consejos e incondicional apoyo, he logrado encaminarme en busca de una profesión académica.

A mi querida madre María Ortíz, que siempre estuvo pendiente en los momentos de tristeza y alegría, llenándome de fortaleza para que pueda continuar en el sendero de la vida.

Pedro Remigio Urgilés Ortíz

Agradecimientos

Quiero agradecer primeramente a Dios por darme la sabiduría y la fortaleza para seguir adelante y no desistir jamás.

A toda mi familia, que de una u otra forma siempre me ha brindado su apoyo, gracias por sus consejos y confianza.

A mis amigos con los que viví momentos inolvidables que siempre recordare, con ellos compartí alegrías, tristezas, aventuras y experiencias que han sido parte de mi vida y me han permitido crecer y ser mejor, gracias a TODOS por compartir sus experiencias y hacerme parte de ellas.

A mis compañeros del centro de investigación Tania y Pedro con quienes trabajé en conjunto estos últimos meses.

A mi director de tesis Ingeniero Vladimir Robles que siempre me brindó su apoyo mediante sus sabios consejos, siendo un amigo incondicional.

Alba Raquel Paguay Paguay

Agradecimiento

A Dios por llenarme de fortaleza y sabiduría para conseguir una de las metas más importantes de mi vida.

Un particular y ferviente agradecimiento al Ingeniero Marcelo Espín y su esposa Doctora Milyn Andrade, por ser los pilares fundamentales sobre los cuales se edificó este proyecto universitario.

A toda mi familia, quienes me han apoyado de diversas maneras en esta travesía, específicamente a mi querida madre María Ortíz, por estar a mi lado en todo momento.

A nuestro director de tesis Ingeniero Vladimir Robles, por sus valiosas aportaciones brindadas en el desarrollo de este proyecto.

A mis amigas y compañeras: Alba Paguay y Tania Patiño, con quiénes he compartido el lugar de trabajo donde desarrollamos la tesis.

Pedro Remigio Urgilés Ortíz

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO 1

INTRODUCCIÓN

1. INTRODUCCIÓN

1.1. ANTECEDENTES

El punto de partida del análisis de imágenes digitales nace del estudio de la visión por computadora, siendo esta considerada una ciencia por grandes científicos, debido a que engloba el estudio desde perspectivas diferentes de un sinnúmero de problemas, perspectivas que están ligadas a principios y leyes de diversas áreas del conocimiento. Esta ciencia tiene como objetivo fundamental lograr que los robots puedan interactuar eficientemente en ambientes complejos, para esto la visión por computadora buscará proveerle del sentido de la vista. Este objetivo conlleva a que el sentido de la vista sea percibido por medio de cámaras digitales en busca de modelar el mundo real mediante la aplicación de los principios geométricos. Los fundamentos de esta disciplina según lo indicado en [7] son:

- El dominio de la relación espacial.
- La medición del espacio tridimensional.
- La propagación de la luz a través de un lente y,
- El modelo matemático de formas y tamaños de objetos.

Mediante dichos fundamentos se logrará tener una comprensión espacial del mundo percibido por un robot.

Una vez dentro de la temática “visión por computadora”, aparece el estudio de los modelos de percepción visual, que permite tener aproximaciones computacionales. La presencia tanto de los mecanismos de procesamiento así como también la representación de la información visual datan desde la época de los griegos; a lo largo de los tiempos hasta la presente fecha se han planteado teorías que intentan demostrar la manera cómo percibe el hombre su realidad exterior y de qué forma utiliza y representa la información captada por los sentidos. Entonces aparecen estudios realizados por Kepler acerca del análisis geométrico de la formación de la

imagen en el ojo, de acuerdo a lo anotado en [6], asimismo Newton realiza estudios sobre la visión en color, mientras que en 1910 Helmhotz estudia la óptica fisiológica, y en 1912 Wertheimer realiza estudios acerca del movimiento aparente de agrupaciones de puntos o campos. Estos estudios se han convertido en los pilares fundamentales para las teorías actuales de la percepción visual. Dichos estudios tomaron fuerza en los años sesenta cuando empezaron aparecer las primeras computadoras, que se requerían para el tratamiento de información espacial donde involucraba el análisis de propiedades del sistema de visión humana.

La determinación de los saltos o discontinuidad presentes en una imagen fue el primer problema que se intentó solucionar, pero por la complejidad del sistema de visión humana y por la lentitud del avance de los algoritmos y teorías que explican su funcionamiento y propiedades, se procedió a resolver este tipo de problemas a partir de tres enfoques diferentes indicados en [7]:

- En primera instancia Azriel Rosenfeld, emprendió el desarrollo de técnicas empíricas que se basan en ciertos criterios de tipo matemático que aproximaban estos procesos a través de la estimación de bordes y líneas usando diferentes criterios.

- En segunda instancia se realizó el estudio a un mundo de bloques blancos mate iluminados sobre fondo negro, estos bloques obtenían muchas formas diferentes, pero siempre y cuando las superficies totales sean planas y sus bordes rectos. Este método permitió que se solucionara la interpretación de los dibujos lineales que se calculaban a partir de imágenes de prismas sólidos.

- Finalmente en tercera instancia Horn en 1986 consigue establecer algunos modelos de cálculos que expresan la formación de la imagen mediante ecuaciones diferenciales relacionando los valores de intensidad de esta con la superficie.

Entre 1973 y 1980, David Marr profundiza el estudio de la percepción permitiendo avanzar más en la Visión Artificial, por lo tanto se focalizó en la descripción de mecanismos que sean similares en lo posible a los del hombre, buscando abordar actividades preceptuales tales como: la percepción del movimiento, textura, distancia, entre otras. Consiguiendo obtener el modelo llamado Paradigma de Marr, que consistía en aproximar la percepción visual y cómo estudiar los sistemas de conocimiento. Sin embargo tanto Gibson en 1979, así como Aloimonos en 1993, aseguran que la teoría de Marr no se puede determinar como una teoría de percepción visual debido a que no consideró que todos los sistemas visuales existentes en la naturaleza son sistemas activos, por lo tanto llevan el control del proceso de adquisición de las imágenes, implicando grandes restricciones para obtener información sobre escenas en tres dimensiones.

Los investigadores Aloimonos, Weiss y Bandopadhayn, en 1988 realizaron las primeras pruebas de desarrollo de visión activa, al definir el observador activo intentaron controlar ciertos parámetros geométricos del sistema sensorial, de acuerdo a lo que dice en [7].

Cabe indicar que los observadores activos son mucho más eficientes que los observadores pasivos de David Marr.

En la era computacional nace el procesamiento de imágenes digitales, que comienza en los años 1950 – 1960, impulsado por la reciente aparición del computador y los programas espaciales de los Estados Unidos, para lo cual se cuenta con una matriz que contiene los valores correspondientes a la imagen que posteriormente mediante el uso de un computador, serían procesados con el propósito de mejorar la calidad visual de las imágenes.

Para conseguir el propósito primordial del procesamiento digital de imágenes es necesario eliminar el ruido tanto en la captación como en la transmisión de

imágenes. Para eso se utilizan técnicas de filtrado tales como: filtro pasa-bajo, filtro pasa-alto, filtro pasa-banda, filtrado no lineal, los mismos que trabajan en el dominio de la frecuencia.

Asimismo existe la técnica de Gauch para eliminar el ruido de imágenes a color. Cabe indicar que a más de estas técnicas de eliminación de ruido existe métodos para la restauración de imágenes que se basan en: filtros inversos, filtros Wiener, filtros Kalman, [7]. Mientras que los científicos: Grenander en 1976, 1978 y 1981, los hermanos Geman en 1984 y Besag en 1986, publican en los respectivos años los estudios que realizaron sobre la restauración y análisis de imágenes digitales mediante la aplicación de los modelos Bayesianos.

Por el año de 1948 el científico Shanon proporciona la base teórica fundamental para realizar la compresión eficiente de las imágenes, [7].

Como ya se mencionó, el procesamiento de imágenes se realiza en el dominio espectral y en el dominio de la frecuencia, en el primer caso debido a que se puede operar sobre la misma imagen, mientras que en el otro caso porque la imagen se puede transformar a un dominio alternativo, permitiendo que posteriormente sea factible aplicar la transformada inversa. “Surgiendo así el uso de las transformadas de Larhunen-Loeve, Fourier, Walsh, Hadamard, Coseno discreto, Wavelets, entre otras”, de acuerdo a lo expuesto en [7].

Una vez que la imagen haya sido procesada por los métodos existentes para el efecto, se puede proceder al análisis de la misma para aplicarla en diferentes áreas, [7]:

- **Reconocimiento de caracteres.-** Al inicio se buscaba la forma de evitar introducir por teclado texto que ya se disponía en forma escrita.

- **Microscopía.-** Las imágenes médicas son examinadas por expertos médicos con la finalidad de clasificar cromosomas, contar células, detectar patologías, entre otros.

- **Radiología.-** A mediados de los años sesenta inician los trabajos que buscan la automatización del análisis de imágenes radiológicas.

- **Teledetección.-** Desde los años sesenta se obtienen grandes cantidades de imágenes satelitales, para las cuales se aplican múltiples técnicas de extracción y análisis de su información.

Adicional a las áreas descritas, el análisis de imágenes también es aplicado en otros campos muy extensos tales como: reconocimiento de huellas dactilares, reconocimiento de caras, búsqueda de imágenes en bases de datos, entre otros.

Las metodologías para la resolución de problemas de análisis de las imágenes han venido evolucionando a lo largo de los años. Para realizar el análisis de una imagen se requiere en primera instancia extraer partes de las cuales está formada la misma, para el efecto se empieza con la utilización de técnicas de segmentación, es así que se da inicio a una primera aproximación a la extracción de partes mediante la aplicación de técnicas de umbralización que utilizan los histogramas, estas técnicas consisten en comparar los niveles de gris con un valor de referencia y asignarle a cada uno una clase en función del rango en el que se encuentre. De esta manera empiezan aparecer técnicas de segmentación tales como: segmentación basada en la moda, los cuartiles, la transformación del histograma, búsqueda de la umbralización óptima, umbralización multiespectral y las basadas en estructuras jerárquicas, esto está especificado en [7].

Posterior a la umbralización aparece otra alternativa para lograr la segmentación de las imágenes, esta alternativa consiste en metodologías de detección de bordes que no son más que saltos abruptos en los niveles de gris o color en las fronteras de los

objetos que constituyen una imagen, esto implica la búsqueda del gradiente o Laplaciano. En 1955 Kavasznay y Joseph dieron paso a la detección de bordes usando el gradiente, mientras que en 1965 Roberts realizó el primer trabajo de detección de bordes con la aplicación de diferencias.

Los experimentos neurofisiológicos realizados a partir del método de Marr dieron como conclusión que las fronteras de los objetos eran los elementos más importantes para unir una imagen de intensidad a su interpretación. En 1982 Marr y Hildreth crearon el filtro “Diferencia de Gaussianas” para la detección de fronteras. Sin embargo previo a obtener este filtro se logra realizar el suavizado para eliminar el ruido de la imagen aplicando una Gaussiana de la cual se obtenía como varianza un parámetro de escala, que sería posteriormente sometido a una Laplaciana con la cual se conseguiría detectar los bordes como cruces por cero de la imagen resultante. En 1986 aparece el detector de Canny, que se formula como la maximización del producto de la relación señal ruido y la localización del borde sujeto a una restricción sobre respuesta múltiple, tal como se indica en [7].

Posterior al estudio de la detección de bordes se procede a estudiar las técnicas que permitan obtener el contorno de una región de la imagen o en su efecto que permitan extraer regiones, en vista de que los métodos de detección de bordes no garantizan su delimitación, aparecen así los métodos de relajación de bordes.

Para la resolución de problemas de obtención del contorno de una región, aparece la transformada de Hough, con este método se logra detectar en principio líneas rectas, posteriormente detecta formas definidas analíticamente, y finalmente se realiza la detección de cualquier tipo de forma.

También es posible realizar la segmentación a partir de las regiones de una imagen, estas regiones en principio pueden ser obtenidas a partir de la umbralización del histograma o pueden ser los píxeles de la imagen, de tal forma que posteriormente se

pueda realizar los procesos de unión, separación o ambos a la vez sobre las regiones obtenidas inicialmente, para conseguir en definitiva las regiones de interés de la imagen, [7].

Finalmente cuando ya haya sido realizado el proceso de detección de regiones o fronteras, se procede a identificarlas o etiquetarlas, concluyendo con su descripción respectiva, dando paso de esta manera al reconocimiento de objetos físicos identificados en la imagen.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Análisis y estudio de las técnicas de extracción de Blobs para el desarrollo de un prototipo de recuperación de imágenes.

1.2.2. OBJETIVOS ESPECÍFICOS

- Analizar los parámetros básicos para el tratamiento de bordes de un blob.
- Analizar la técnica de extracción de blobs basada en funciones del operador Laplaciano de Gauss y el *Kernel* Gaussiano.
- Describir las técnicas de localización de las regiones de interés de una imagen.
- Analizar y seleccionar las herramientas de extracción de blobs de una imagen.
- Seleccionar y preparar un corpus de imágenes que servirán para realizar las pruebas de análisis.
- Desarrollar el prototipo funcional basado en las funciones del operador Laplaciano de Gauss y el *Kernel* Gaussiano.

1.3. ALCANCE

El enfoque de este proyecto se centra en la extracción de blobs de una imagen para el cual se realizará el análisis de varios métodos para el efecto, profundizando en un método específico: Operador Laplaciano y el *Kernel* Gaussiano, como parte del proceso se realizará la segmentación basada en umbralización, se analizarán técnicas basadas en regiones, y como resultado la extracción de los blobs de una imagen.

Se verificará las herramientas disponibles que permitan la extracción de blobs, para este proceso se utilizará un corpus, mismo que permitirá hacer los análisis y pruebas necesarias con cada herramienta.

Posteriormente al análisis del método Operador Laplaciano y el *Kernel* Gaussiano, se procederá al desarrollo de un prototipo funcional que permitirá extraer las características de las imágenes con las cuales se construirá un modelo de conocimiento q permita la recuperación de imágenes basada en blobs, finalmente se ejecutarán entrenamientos para verificar la efectividad del reconocimiento de blobs, y se medirá la precisión, comparando esta información con el estado del arte.

CAPITULO 2

TÉCNICAS DE DETECCIÓN DE BLOBS BASADAS EN FUNCIONES GAUSSIANAS

2. TECNICAS DE DETECCIÓN DE BLOBS BASADAS EN FUNCIONES GAUSSIANAS

2.1. INTRODUCCIÓN

La detección de blobs consiste en identificar y diferenciar cada objeto que se encuentre dentro de una imagen digital, siendo este objeto la región o la unión de regiones que serán extraídas de la imagen mediante métodos de segmentación.

Previo a la revisión de los métodos de segmentación existentes, es necesario realizar una breve descripción de la forma como está estructurada una imagen digital que será procesada mediante las técnicas mencionadas, así como también se indicarán las posibles operaciones a la que una imagen podrá ser sometida. Esto facilitará el manejo de conceptos posteriores en los procesos que permitirán la extracción de blobs.

2.1.1. DESCRIPCIÓN DE UNA IMAGEN DIGITAL

Una imagen al ser digitalizada trae consigo algunos componentes y conceptos que son utilizados al ser analizados y procesados. A continuación se los describe:

- **Píxel.-** Un píxel p es la representación de intensidad de un punto (x, y) en una imagen.
- **Adyacencia.-** Dos píxeles son adyacentes si y solo si, tienen en común una de sus fronteras o esquinas.
- **Píxel Vecino.-** Dos píxeles son vecinos, si los mismos cumplen con la definición de adyacencia. Si los píxeles comparten una de sus fronteras son llamados “vecinos directos”. Si comparten alguna de sus esquinas son nombrados “vecinos indirectos”.

- **Vecindad de un Píxel.-** Una vecindad de un píxel p denotada como V_p es una submatriz M_{ji} de un tamaño JxJ, con J e I enteros impares, contenida en la matriz imagen I_{MN}.

$$V_p = \left\{p: p \in M_{ji}; M_{ji} \subset I_{ji}; J = I = \{3, 5, 9\}\right\} \quad \textbf{(2.1)}$$

- **Imagen Digital.-** En términos digitales una imagen es una función de intensidad bidimensional $F(x, y)$, donde "x" e "y" son las coordenadas espaciales y el valor de F es la intensidad o nivel de gris en cualquier punto (x, y), tal como se puede observar en la figura 2.1.

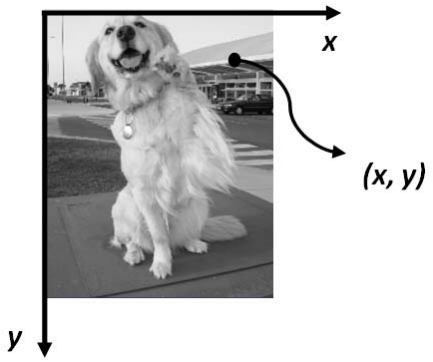


Figura 2.1: Representación de una imagen digital.

Una imagen digital en escala de grises puede ser representada mediante una matriz de MxN, donde cada índice de fila y columna contiene un elemento numérico que representa a un píxel de la imagen digital, estos valores numéricos están dentro del rango (0 - 255), donde el "0" representa al color negro y el "255" al color blanco, y cada valor numérico es la intensidad luminosa o nivel de gris correspondiente a un determinado píxel.

En la figura 2.2 se observa que a partir de una imagen digital en escala de grises se extrae una porción de ésta, donde a través de un zoom se puede identificar la intensidad luminosa que posee cada píxel ubicado en su respectiva posición (i, j),

posteriormente se arma la matriz con los valores numéricos que corresponden a los niveles de gris de los píxeles y con la identificación de su posición matricial.

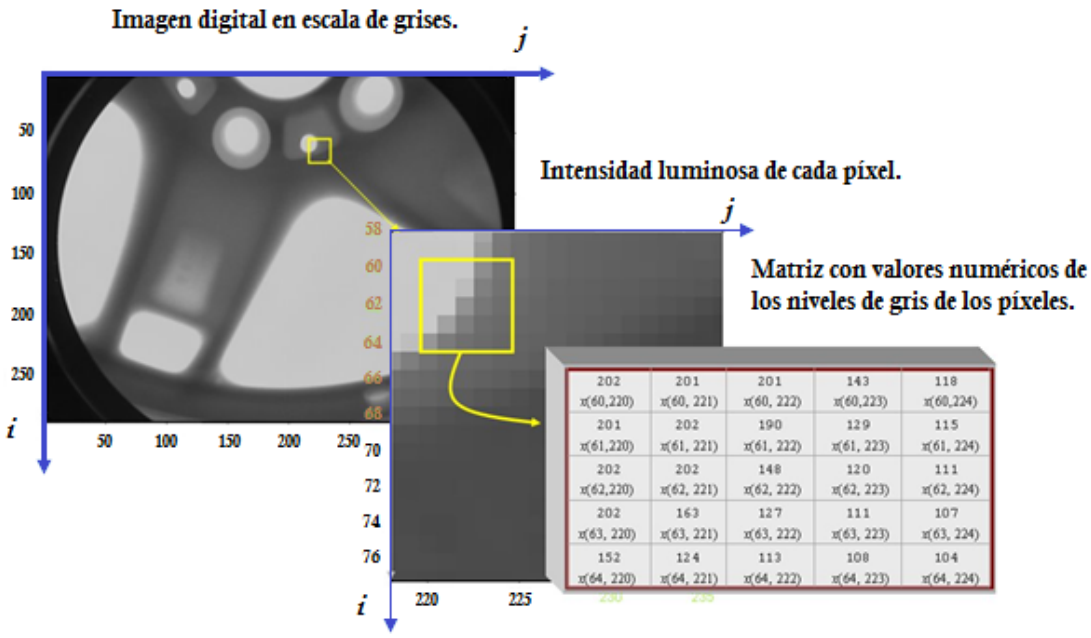


Figura 2.2: Representación matricial de una imagen digital [27].

2.1.2. ETAPAS DEL PROCESAMIENTO DIGITAL DE UNA IMAGEN

En los antecedentes del capítulo 1 se habló brevemente sobre la historia del procesamiento digital de imágenes, sin embargo en este apartado se habla sobre las etapas o fases de este procesamiento.

Cabe recalcar que el procesamiento digital de imágenes busca mejorar la calidad visual de las imágenes para permitir la interpretación humana, así como también cumplir con la función de extraer información de las imágenes en un formato entendible por el ordenador.

En la figura 2.3 se puede verificar las etapas del procesamiento digital de imágenes que se ejecutan consecutivamente para obtener a partir de una imagen original otra imagen como resultado de dicho procesamiento.

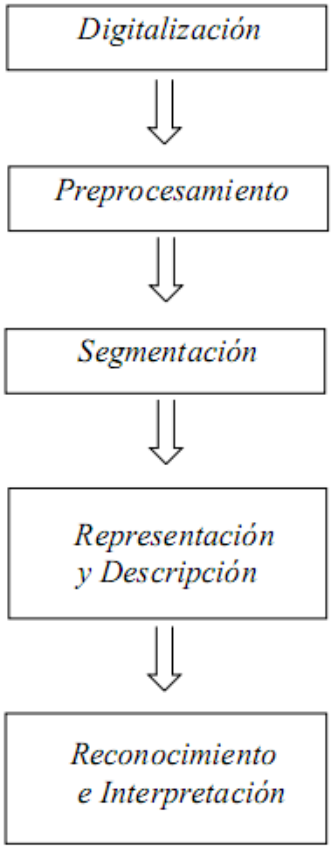


Figura 2.3: Fases del procesamiento digital de imágenes [2].

A continuación se realiza una breve descripción de cada fase del procesamiento digital de imágenes:

Digitalización.- En esta fase se realiza la captura de una imagen del mundo real a través de un dispositivo para el efecto, tal como: cámara de video, escáner, etc. Posteriormente se realizará la conversión de la imagen capturada a un formato digital utilizando un conversor analógico-digital, tarea que facilitará que un computador pueda leer e interpretar la imagen.

En la fase de digitalización existen varios factores que hacen que se produzca una degradación de la imagen digitalizada con respecto a la original, siendo esta degradación producto del ruido y la pérdida de definición de la imagen, es aquí donde entra en operación la segunda fase del procesamiento digital de imágenes que se describe a continuación.

Pre-procesamiento.- Esta fase tiene la función de contrarrestar la degradación producida en la fase de digitalización, para lo cual realiza la supresión del ruido y el realce del contraste. Al aplicar esta fase, se hace posible que los resultados del posterior tratamiento de la imagen sean más precisos.

Segmentación.- Ya en la etapa de segmentación se pretende extraer la información contenida en la imagen, el proceso de extracción se realiza mediante la descomposición de la imagen en partes homogéneas con respecto a una o más características que están relacionadas con objetos del mundo real, obteniendo una imagen binarizada.

Representación y Descripción.- En esta fase se encuentran dos etapas: en la etapa de la representación, como su nombre lo indica, se representa ya sea el contorno de los objetos encontrados o la región que abarca cada objeto, esto como producto de la salida del proceso de segmentación. Mientras que en la etapa de la descripción se procede a extraer información de la representación que haya sido elegida, obteniéndose las características o los descriptores que harán posible la posterior clasificación de los objetos encontrados. Estos descriptores pueden ser: el perímetro del contorno, el área, el número de huecos, los puntos más significativos o relevantes, entre otros.

Reconocimiento e Interpretación.- Finalmente en esta fase también se encuentran dos etapas: en la etapa de reconocimiento se etiquetan o clasifican los diferentes objetos de la imagen utilizando sus descriptores, si existe el caso

donde los objetos detectados presenten unos descriptores semejantes, entonces se procederá a agruparlos en una misma clase. Mientras que en la etapa de la interpretación se dará un significado a los grupos de objetos reconocidos.

2.1.3. OPERACIONES EN UNA IMAGEN DIGITAL

En este apartado se analizarán las formas existentes para procesar imágenes, de tal manera que a través de la modificación de una imagen original se obtendrá una nueva imagen resultante.

Una imagen se forma como una sucesión de píxeles, por consiguiente el procesamiento de imágenes se realiza mediante la alteración de dichos píxeles. Esta alteración se lleva a cabo aplicando operaciones de filtrado con la finalidad de optimizar la imagen, conseguir un efecto especial en la misma, ó enfatizar cierta información. Obteniendo como resultado el suavizado de la imagen, la eliminación del ruido, el realce de bordes, o la detección de bordes, entre otros.

Existen dos formas de alterar los píxeles de una imagen [4]:

- Alteraciones píxel a píxel (individual).
- Alteración en múltiples píxeles (vecindad).

En la figura 2.4 se puede evidenciar gráficamente las dos alternativas de operar sobre las imágenes digitales. Las operaciones mencionadas se utilizan para eliminar el ruido que existe en las imágenes y también para la segmentación de imágenes.

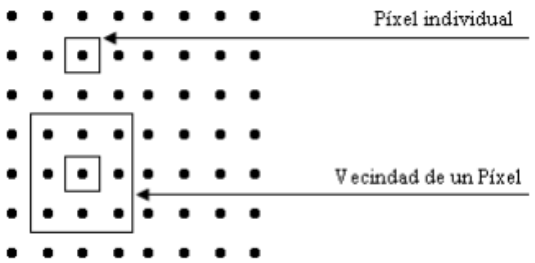


Figura 2.4: Píxel individual y vecindad del píxel [4].

Operación de píxel.- Esta operación se realiza a cada píxel de manera individual, teniendo como resultado una nueva imagen producto de la modificación impuesta al valor del píxel mediante un operador f, este operador se aplica a cada uno de los píxeles de la imagen para tener una salida que dependerá únicamente de la magnitud correspondiente al píxel de entrada. El operador transforma el valor de gris de cada píxel y el valor resultante se obtiene mediante la siguiente ecuación [4]:

$$y(i,j) = f\big(p(i,j)\big) \qquad \textbf{(2.2)}$$

Donde $y(i,j)$ es el valor transformado del píxel, $p(i,j)$ el píxel de entrada, y f el operador de transformación.

Operaciones de vecindad.- En primera instancia se conoce que una vecindad es una matriz bidimensional, de filas y columnas impares, y que contiene valores de píxeles extraídos de una imagen digital, donde el píxel que será procesado se encuentra siempre en el centro de la matriz vecindad, esto lo podemos verificar en la imagen presentada en la figura 2.5. El procedimiento es igual al que se aplica a un píxel individual, sin embargo existe una excepción que consiste en que el valor del píxel para la imagen de salida dependerá de la combinación de los valores de los píxeles en la vecindad, de tal manera que estos píxeles serán sometidos a la convolución con una matriz llamada máscara.

La figura 2.5 muestra el procedimiento de aplicar una operación de vecindad a partir de una imagen de entrada y devolviendo una imagen de salida como producto de la modificación a la que se someten sus píxeles.

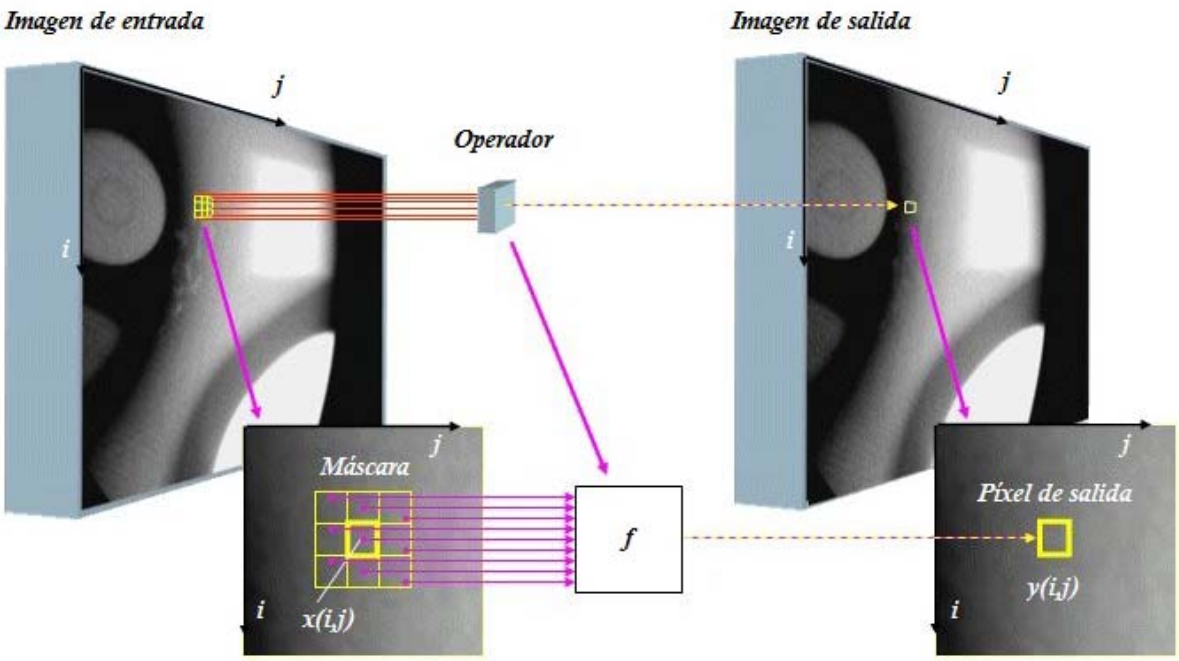


Figura 2.5: Procedimiento de la operación de vecindad [27].

Aplicación de la máscara de convolución.- La máscara de convolución, también conocida como filtro, se aplica para modificar una imagen con el propósito de conseguir un determinado fin, siendo este fin la detección de bordes, la modificación del aspecto o también para la eliminación de ruido.

Por lo tanto la convolución discreta aplicada a una imagen digital consiste en realizar la suma de productos entre los píxeles de una imagen y otra subimagen definida como máscara, esta máscara es convolucionada alrededor de un píxel de la imagen. La ecuación de esta convolución discreta se define de la siguiente manera:

$$y[i,j] = x[i,j] * h[i,j] = \sum_{k=1}^{n} \sum_{l=1}^{m} x(i-k, j-l)h(k,l) \qquad \textbf{(2.3)}$$

Donde $x[i,j]$ es el espacio bidimensional de la imagen de entrada, y el filtro o máscara de convolución con respuesta impulsiva $h(k,l)$, en donde k y l definen la vecindad a considerar de acuerdo al tamaño de la máscara de convolución $h(k,l)$, obteniendo como resultado de la convolución una imagen modificada de salida $y[i,j]$.

El procedimiento para obtener un nuevo píxel producto de la convolución, se muestra en la figura 2.6. En primera instancia tenemos la imagen de entrada representada por una matriz de píxeles, en donde está marcada la ventana de convolución sobre la que interactuará la máscara de convolución que en este caso es de $3x3$ píxeles. Este procedimiento consiste en centrar la máscara sobre cada píxel a ser procesado, en este caso el píxel central que es procesado es el P_{E4}. Posteriormente se multiplica cada valor de la ventana de convolución, es decir el píxel central y sus píxeles vecinos por el valor correspondiente de la máscara de convolución. Finalmente se suman los resultados de la multiplicación para obtener un nuevo píxel de salida P_{S4}, con coordenadas iguales a las coordenadas del píxel central de la ventana de convolución, tal como se observa en el gráfico de la figura 2.6, donde la ventana de la imagen de salida está ubicada en la misma posición que se encuentra la ventana de convolución de la imagen de entrada.

Este procedimiento se aplica a todos los píxeles de la imagen de entrada para obtener la nueva imagen resultante producto de su convolución integral.

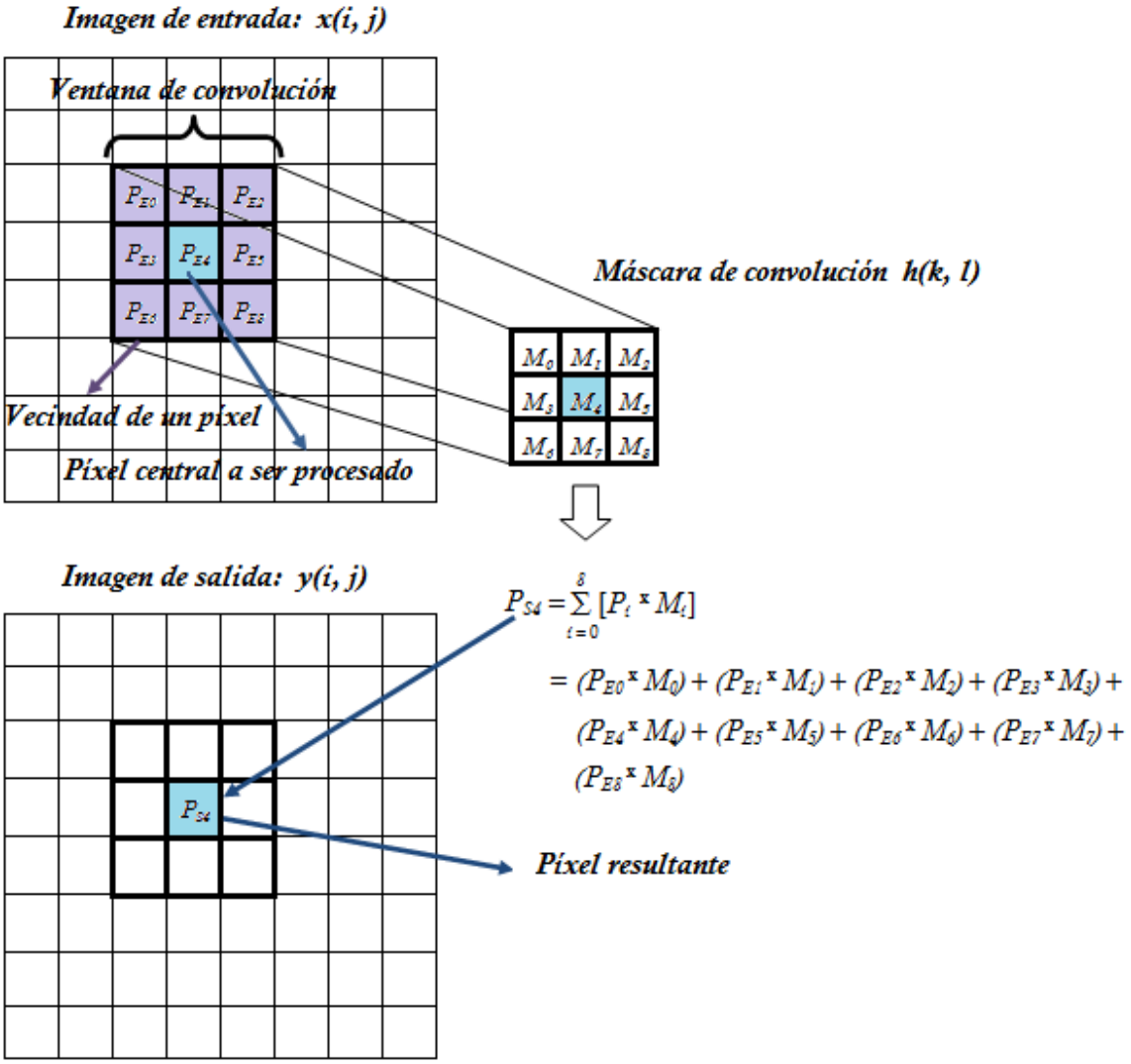


Figura 2.6: Píxel resultante aplicando la máscara de convolución.

2.1.4. PROBLEMAS DE CONVOLUCIÓN DE LOS BORDES DE LA IMAGEN

Uno de los problemas que se presentan en el proceso de convolución consiste en que no se puede centrar la máscara de convolución sobre los píxeles que se encuentran en el borde de la imagen, ya que estos píxeles no tienen todos los vecinos, tal como se observa en la figura 2.7, por lo tanto estos vecinos que no existen se los considera por defecto como nulos. Sin embargo existen varias

alternativas que permiten que estos píxeles en dilema puedan ser operados por la máscara de convolución.

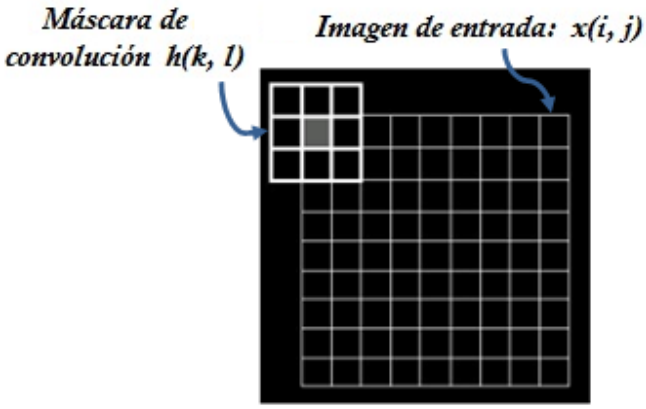


Figura 2.7: Problema de convolucionar los píxeles del borde [28].

Para no tener problemas con la convolución de los píxeles de los bordes de la imagen, lo que se puede hacer es convolucionar toda la imagen menos los píxeles de los bordes, en este caso la imagen de salida será más pequeña que la imagen de entrada. Es decir si la imagen de entrada es de 500x300, entonces al omitir la convolución de los píxeles de los bordes, se tendría como salida una imagen de tamaño 498x298. Esto se lo puede verificar en la figura 2.8.

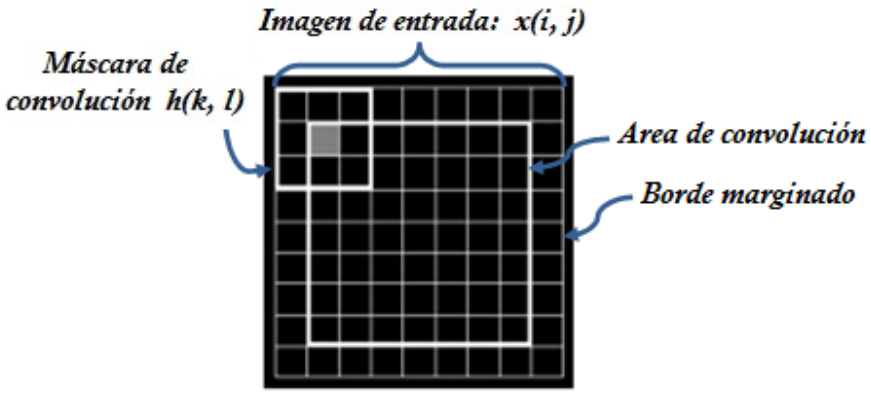


Figura 2.8: Convolución de la imagen marginando sus bordes [28].

En caso de que sea necesario analizar los bordes de la imagen, no es recomendable aplicar la marginación de los bordes en el proceso de convolución. Por lo tanto existen alternativas que permiten tomar en consideración dentro de la convolución los bordes de la imagen. A continuación se presentan algunas alternativas para combatir el problema del tratamiento de los bordes de la imagen [28]:

1. Se considera que los píxeles vecinos no existentes en la ventana de convolución son tratados como ceros, llamado también *zero-padding*, tal como se observa en la figura 2.9, este método no es recomendable aplicarlo en caso de que los bordes son importantes dentro del análisis posterior. Además se puede asignar un valor arbitrariamente, o también se toma un valor intermedio de los píxeles existentes.

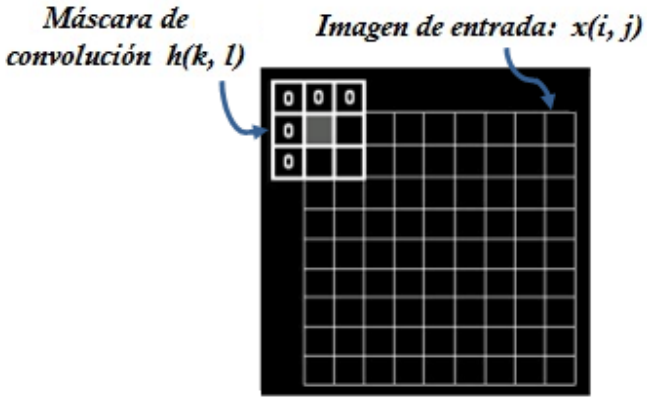


Figura 2.9: Método *zero-padding* [28].

2. Otra alternativa consiste en duplicar los bordes de la imagen antes de ejecutar el proceso de convolución. En la figura 2.10 los bordes de color tomate son los duplicados del borde original, de esta manera la máscara de convolución no tendría ningún problema en convolucionar los píxeles del borde original de la imagen de entrada.

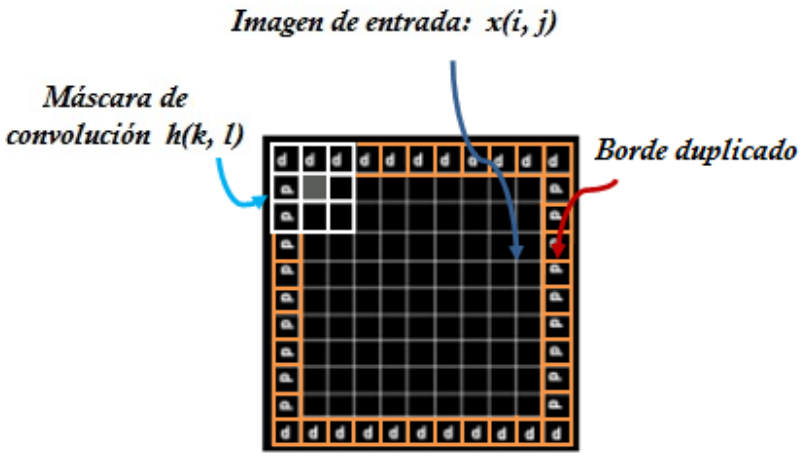


Figura 2.10: Borde duplicado de la imagen [28].

3. Dentro de las alternativas también se puede envolver la imagen. Esto consiste en considerar como píxel contiguo al del borde izquierdo, el píxel del borde derecho y viceversa, así como con los del borde superior e inferior. En la figura 2.11 se puede observar la forma como opera esta alternativa.

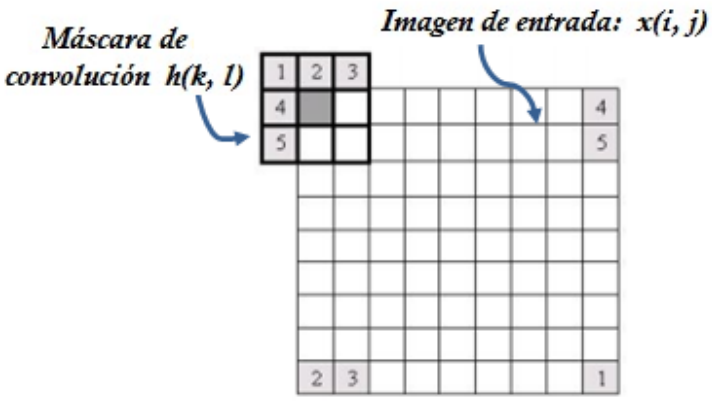


Figura 2.11: Método de envoltura de la imagen [28].

2.1.5. SUAVIZADO BINARIO DE LA IMAGEN

Muchos métodos de detección de bordes realizan también el suavizado de la imagen previo a la detección de dichos bordes. El suavizado se aplica para eliminar el ruido existente en la imagen para que no exista ninguna distorsión en la detección de los bordes, ya que el ruido ocasiona irregularidades en los contornos, produce pequeños huecos, permite la presencia de puntos aislados, entre otros.

A continuación se describe los efectos que produce la aplicación del suavizado sobre imágenes binarias de acuerdo a lo especificado en [27]:

- Permite eliminar los puntos blancos aislados.
- Permite rellenar los pequeños huecos de un píxel (píxeles blancos) en zonas oscuras.
- Permite eliminar las pequeñas protuberancias a lo largo de segmentos de lados rectos.
- Permite rellenar cortes y muescas en segmentos de lados rectos.
- Permite reponer los puntos perdidos de las esquinas.

Claramente se puede verificar en la figura 2.12, la diferencia entre una imagen distorsionada por los efectos del ruido, y la imagen de salida luego de aplicar algunos métodos matemáticos de suavizado binario.

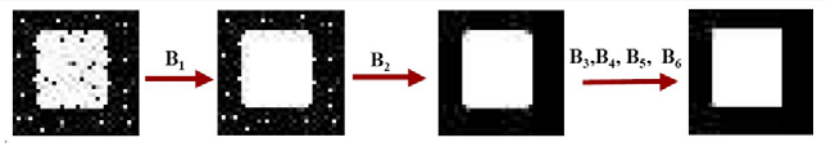


Figura 2.12: Supresión de ruido mediante algoritmos de suavizado binario [27].

En un apartado posterior se profundizará el estudio del algoritmo *Kernel* Gaussiano, debido a que este algoritmo de suavizado se aplicará en la parte práctica de la tesis junto con el operador Laplaciano para la detección de bordes.

2.1.6. MÉTODOS DE SEGMENTACIÓN DE UNA IMAGEN

Los métodos de segmentación realizan la partición de una imagen en blobs o regiones homogéneas con respecto a características definidas tales como: la forma, el color, el brillo, la textura, el movimiento, entre otras.

Las técnicas de segmentación están basadas fundamentalmente en dos propiedades de los valores de intensidad, estos son:

- **Discontinuidad.-** Mediante esta propiedad se buscan cambios abruptos en la intensidad, para el efecto se utilizan técnicas de segmentación basadas en la extracción de bordes.

- **Similitud.-** Mientras que esta propiedad permite buscar regiones similares considerando criterios que determinarán el método a aplicar. Los métodos de segmentación que se utilizan en esta propiedad son los basados en la umbralización del histograma, extracción de regiones, movimiento y uso de color.

Considerando las propiedades antes expuestas se describe brevemente las técnicas que permiten segmentar una imagen y que se agrupan en varios tipos como son: segmentación basada en umbralización, segmentación basada en la detección de bordes, segmentación basada en regiones, entre otros:

Segmentación basada en umbralización.-

Esta técnica define un umbral, de tal manera que separe los objetos de interés respecto del fondo. Para el efecto, debe existir claramente una diferenciación entre los objetos y el fondo de la escena, tal como se observa en la figura 2.13.

Figura 2.13: Diferencia entre los objetos y el fondo [8].

Este método permite convertir una imagen original $I(i,j)$ que está en niveles de gris, en una imagen binaria $B(i,j)$ como salida, mediante la aplicación de un umbral de separación U seleccionado previamente, teniendo:

$$B(i,j) = 1, \quad \text{si} \quad I(i,j) \geq U \tag{2.4}$$

$$B(i,j) = 0, \quad \text{si} \quad I(i,j) < U \tag{2.5}$$

El umbral que se obtiene se basa en el histograma de la imagen, mediante la búsqueda y análisis de sus mínimos locales. El histograma de una imagen considera solamente la distribución de grises en la imagen, dejando de lado la información espacial. De esta forma se asignará 0 a los píxeles de los objetos y 1 a los píxeles de fondo y viceversa. La salida de este procedimiento se observa en la figura 2.14.

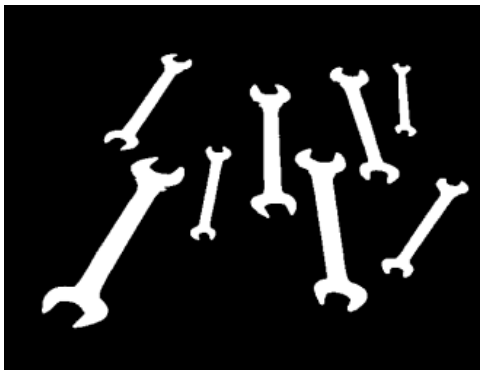


Figura 2.14: Imagen segmentada mediante umbralización [8].

Este tipo de segmentación es rápido, sencillo e intuitivo, y se puede realizar en tiempo real y con un coste computacional bajo, tal como se indica en [8].

Segmentación basada en técnicas de detección de fronteras.-

La segmentación de una imagen es posible realizarla también a través de técnicas de detección de fronteras, debido a que estas técnicas proporcionan información muy valiosa sobre las fronteras de los objetos, lo que consecuentemente permite el reconocimiento y extracción de estos objetos, conocidos también como blobs, mismos que están contenidos en la imagen en tratamiento.

La detección de bordes o fronteras se caracteriza porque existen límites estrechamente relacionados entre las regiones y los bordes, esto debido a que hay un fuerte ajuste en la intensidad en los límites de las regiones, encontrándose discontinuidades en los niveles de gris.

En la figura 2.15 se puede apreciar la detección de bordes a través de varios métodos para el efecto, una vez detectados los bordes se debe obtener los contornos de los objetos específicos que se desea extraer e identificar mediante una etiqueta. Por ejemplo se podría extraer el objeto sombrero.

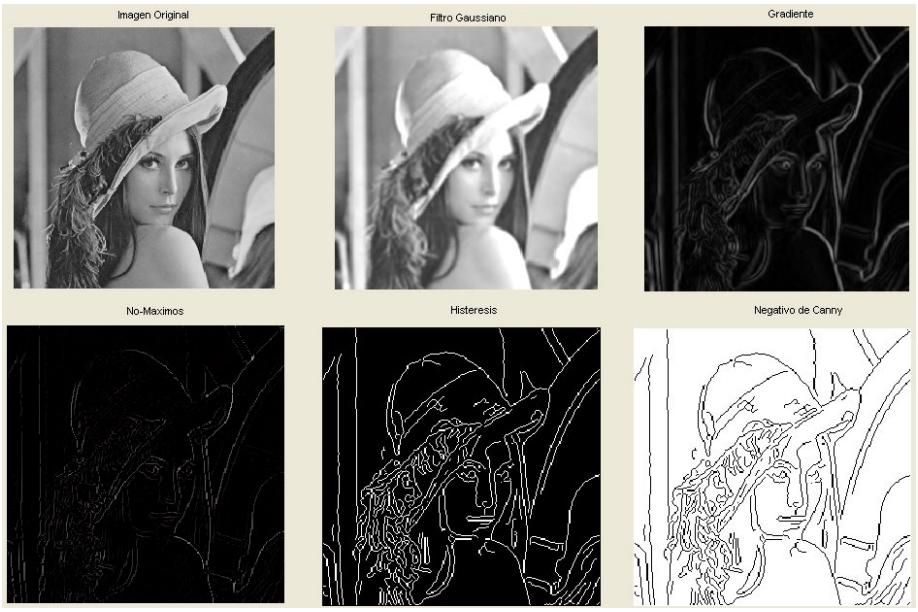


Figura 2.15: Detección de bordes previo a la segmentación [9].

Segmentación mediante técnicas basadas en regiones.-

Este tipo de segmentación tiene como objetivo dividir una imagen en regiones, de tal manera que se construyen las regiones directamente usando información de la conectividad para crearlas.

La formulación básica de esta técnica de segmentación está definida de la siguiente manera según se indica en [8]:

Sea U la imagen entera. La segmentación es un proceso que divide R en n subregiones R_1, R_2,..., R_n, tal que se deben cumplir las siguientes condiciones:

a) $\bigcup_{i=1}^{n} R_i = R$. La segmentación debe ser completa, donde cada píxel debe estar en una región.

b) R_i es la región conectada, $R_i = 1, 2, \dots n$. Los puntos de una región deben estar conectados.

c) $R_i \cap R_j =$ para todo i y j, $i \neq j$. Las regiones deben ser disjuntas

d) $P(R_i) = TRUE$ para $i = 1, 2, \ldots, n$. Los píxeles de una región segmentada deben cumplir una propiedad, por ejemplo $P(R_i) = TRUE$ si todos los píxeles de R_i tienen el mismo nivel de gris.

e) $P(R_i \cup R_j) = FALSE$ para $i \neq j$. Las regiones R_i y R_j son diferentes en el sentido del predicado P.

$P(R_i)$ es un predicado definido en R_i, y es el conjunto vacío. El predicado es una condición que se verifica en todos los puntos pertenecientes a una región y que es falsa en las demás.

La segmentación basada en regiones se divide en dos grupos de métodos que se describen brevemente a continuación:

Métodos de crecimiento de regiones.- Realiza un procedimiento iterativo que consiste en agrupar píxeles o regiones adyacentes que cumplan criterios tales como: nivel de intensidad, varianza de los niveles de la región, y textura de la región, para lo cual se establecen puntos semilla, que son una colección de píxeles iniciales en las regiones que crecen. Es decir, a estos píxeles semillas se les agregan otros píxeles adyacentes y con alguna propiedad similar a la semilla, esto quiere decir que pasarán a pertenecer a la misma región y a tener los mismos valores que los puntos semilla. Por lo general el criterio de agregación que se utiliza consiste en que un píxel adyacente a una región se agregará a ésta si su intensidad es similar a la de los píxeles de la región. El algoritmo finaliza cuando dejan de haber regiones adyacentes que cumplan los criterios.

Procedimientos de separación-unión.- Este procedimiento utiliza un algoritmo que tiene como entrada una imagen y como salida devuelve una imagen formada por regiones homogéneas. Esta técnica se divide en dos fases: la primera consiste en dividir la imagen en sub-imágenes o regiones disjuntas, y la segunda fase consiste en unificar las sub-imágenes similares. Para este proceso es necesario utilizar una medida de similitud de regiones, para lo cual se puede utilizar los valores de la media y la desviación típica de los niveles de gris de los píxeles de las regiones.

Segmentación Morfológica (Watersheds).-

Esta técnica de segmentación consiste en procedimientos basados en morfología matemática, con el objetivo de extraer las fronteras de las regiones de una imagen. Los procedimientos realizan la partición de la imagen en niveles de gris para obtener como resultado otra imagen formada por regiones significativas, proporcionando resultados robustos de regiones y de fronteras. Esta técnica maneja restricciones que se basan en el conocimiento de las regiones a segmentar, además se aplican conceptos de otras técnicas de segmentación tales como bordes, umbralización y crecimiento por regiones. La aplicación de la técnica es muy útil para extraer objetos uniformes de la imagen.

El sobre-segmentación es un problema que está presente dentro de esta técnica, esto debido a la alteración que sufre el gradiente por varias irregularidades locales, entre las que se encuentra el factor ruido.

Segmentación basada en el color.-

La segmentación de una imagen basada en el color es posible llevarlo a cabo mediante la utilización de umbrales. Esta segmentación es más conocida como segmentación en RGB (Red, Green, Blue), esto se debe a que el color es representado como la unión de tres planos, donde cada plano contendrá

información de la intensidad de cada punto respecto a cada una de las componentes de una base de color, siendo las bases de color el rojo, verde y azul.

Las técnicas usadas en imágenes en escala de grises son las que se utilizan para la segmentación de imágenes basadas en el color. Consiste en aplicar estas metodologías sobre cada uno de los planos de color que son tratados por separado.

El proceso de la segmentación basado en el color se lleva a cabo segmentando individualmente cada una de las tres componentes de color de las que consta la imagen. En primera instancia se debe aplicar el suavizado en cada uno de los tres planos de color, de tal manera que se obtengan mejoras en el histograma, para posteriormente aplicar el método de umbralización multinivel de Otsu a cada una de las tres componentes de color, y obtener los umbrales con un mayor éxito. Se obtendrán por lo tanto tres conjuntos de umbrales tal como se puede observar en la figura 2.16.

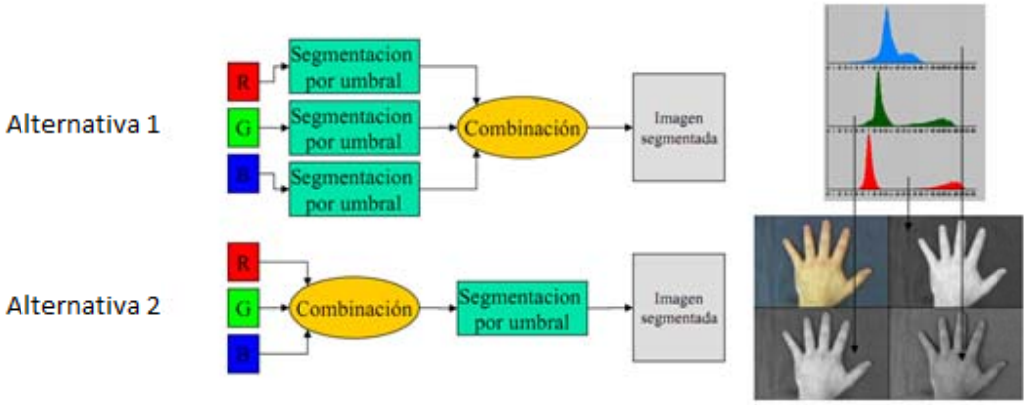


Figura 2.16: Alternativa 1: Combinación por segmentaciones. Alternativa 2: Combinación lineal [5].

Finalmente los tres conjuntos de umbrales, o las tres segmentaciones por umbral obtenidas de cada componente de color serán combinados mediante la aplicación

de ciertos criterios para el efecto, teniendo como resultado la imagen de color segmentada tal como se puede apreciar en la figura 2.17.

En la figura 2.16 se puede ver las dos alternativas que se tiene para segmentar imágenes de color. En la primera alternativa la combinación de los resultados de la segmentación de cada componente por separado suele ser menos eficiente, porque limita severamente la partición del espacio de color. Mientras que la combinación lineal presenta mejores resultados, esto se debe a que se realiza la combinación linealmente de las componentes de color, creando así una única característica a partir de la cual se procede a segmentar aplicando la umbralización.

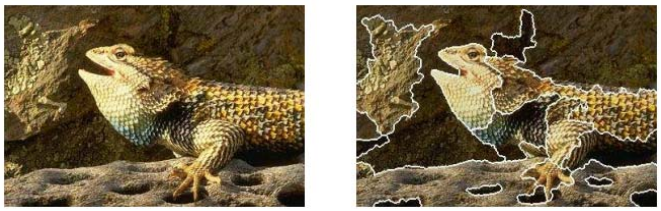


Figura 2.17: Segmentación basada en el color [8].

Segmentación a partir del movimiento.-

Este tipo de segmentación se utiliza para segmentar cada frame que representa la escena cambiante en un determinado instante de tiempo t, tal como se puede observar en la figura 2.18. La secuencia de frames interrelacionados son capturados de escenas dinámicas.

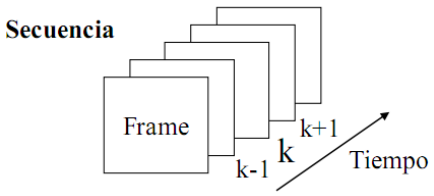


Figura 2.18: Secuencia de *Frames* [8].

Existen técnicas que hacen posible la detección de movimiento y que permiten segmentar objetos que se mueven, técnicas que se describen en [8]:

Imagen diferencia.- Método de detección de cambios mediante la comparación entre dos imágenes, implementada como una recta en valor absoluto, se puede umbralizar la resta obtenida previamente para conseguir así una "imagen diferencia binaria".

Los problemas que presenta esta técnica se ven reflejados en la atenuación del ruido, así como también hay una gran dependencia del umbral, y los movimientos lentos pueden perderse, lo que significa la pérdida de información para un análisis exitoso.

Imagen de diferencias acumuladas.- Mediante esta técnica se puede evitar el problema de la pérdida de los movimientos lentos que se presenta en el método de "Imagen Diferencia". Esto es posible debido a que este método analiza mínimo tres *frames* para que sea posible conseguir información más fiable, de tal manera que se evita también el ruido casual y permite la detección de movimientos más lentos. La imagen resultante se forma comparando todos los *frames* de la secuencia con el de referencia [8].

En el siguiente apartado se describirán exhaustivamente las técnicas basadas en detección de fronteras, específicamente las técnicas que se enfocan en funciones Gaussianas. Las técnicas mencionadas serán utilizadas para la construcción de un prototipo mediante el cual se llevará a cabo la implementación y el posterior entrenamiento utilizando un corpus y herramientas específicas para el efecto.

2.2. DETECCIÓN DE BORDES

Los bordes de una imagen se encuentran dados por el cambio significativo de niveles de gris entre dos o más píxeles vecinos, o en su efecto se puede definir como transiciones entre dos regiones de niveles de gris significativamente distintos, así como también se puede conceptualizar como la discontinuidad en los atributos de la imagen.

Dentro del procesamiento digital de imágenes existen múltiples tareas que se realizan previo a la detección de los bordes, tareas tales como: realce de imágenes, segmentación de imágenes, compresión de imágenes, reconocimiento de objetos, reconstrucción 3D, entre otras. Esto se hace con el fin de obtener resultados favorables en cada tarea, por ello es esencial que la detección de los bordes sea realizada con mucha precisión, ya que contienen información esencial de la imagen que sirve como punto de partida para posteriormente realizar las tareas del procesamiento digital.

De acuerdo a lo que se especifica en [2], la mayoría de los algoritmos de detección de bordes están basados en las derivadas de primer orden (el gradiente), en las derivadas de segundo orden (el Laplaciano) y en criterios de optimización. Así como también han ido apareciendo nuevos algoritmos de detección de bordes mediante la utilización de otros modelos matemáticos, estos son: ajuste paramétrico, morfología matemática, análisis de textura, teoría de conjuntos borrosos, relajación, redes neuronales, y algoritmos genéticos. Cabe indicar que además existen algoritmos diseñados para la detección de líneas, uniones y esquinas.

Para el proceso de detección de bordes tenemos como entrada una imagen digitalizada y su resultado es un mapa de bordes, tal como se puede ver en la figura 2.19. Dependiendo del detector de bordes que se utiliza, el mapa de bordes va a poseer cierta información explícita sobre la posición y la fuerza de los bordes, así como también su orientación y escala.

(a) **(b)**

Figura 2.19: (a) Imagen Original **(b)** Resultado del detector de bordes [2].

Para la detección de un borde se toma en cuenta los atributos de la imagen, mismos que pueden estar dados por la iluminación, reflejos, sombras, geometría de los objetos, profundidad de los objetos en escena, ocultaciones parciales de los objetos, la textura de los objetos, el color, cambios materiales, etc., poniendo énfasis en el color, ya que una imagen a color brinda mayor información. En la figura 2.20 se puede identificar los atributos de una imagen.

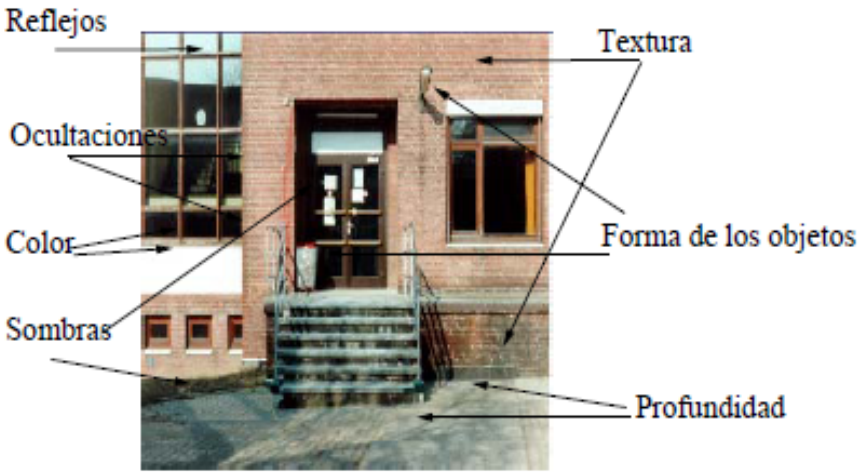


Figura 2.20: Atributos de una imagen [2].

2.2.1. TIPO DE BORDES

Para lograr un buen diseño de un detector de bordes, es importante estar familiarizado con los tipos de bordes que existen tanto en una imagen monocromática como en una imagen en color.

A continuación se describen los tipos de bordes que se encuentran de acuerdo a la clasificación de las imágenes mencionadas [2]:

2.2.1.1. BORDES EN IMÁGENES MONOCROMÁTICAS

En este tipo de imágenes los bordes tienen una correspondencia con los cambios o discontinuidades de la función de intensidad de los niveles de gris. Por consiguiente la forma de la función de intensidad a lo largo de una línea de la imagen que sea perpendicular a un borde permite diferenciar los siguientes tipos de bordes:

- Escalón o salto (*step edge*).
- Rampa (*ramp edge*): puede ser con pendiente cóncava (*concave slope*), con pendiente convexa (*convex slope*) o con ambas.
- Escalera (*staircase edge*).
- Pico (*peak edge*), cresta (*ridge edge*) o pulso (*pulse edge*).
- Valle (*valley edge*).
- Tejado (*roof edge*).

Los perfiles de la función de intensidad se deben extraer tomando aquellos valores de la función de intensidad a lo largo de una línea de la imagen que fuera perpendicular al borde, en la figura 2.21 se puede apreciar los perfiles de la función de intensidad que corresponden a cada tipo de borde.

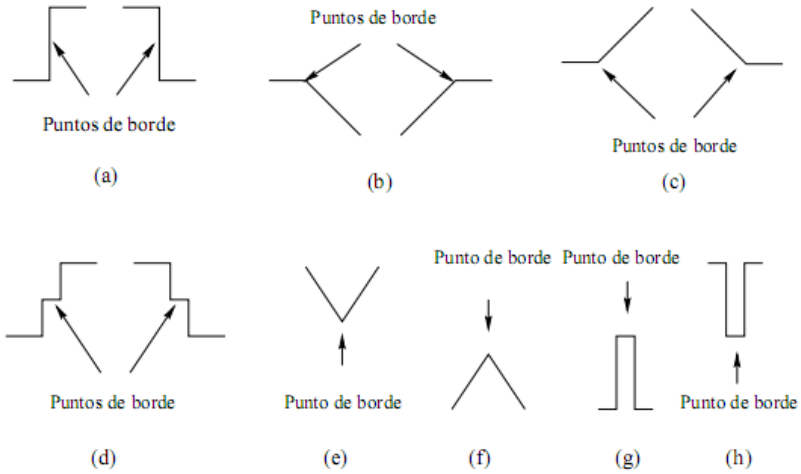


Figura 2.21: Tipos de bordes: (a) escalón o salto, (b) rampa convexa, (c) rampa cóncava, (d) escalera, (e) valle, (f) tejado, (g) y (h) pico o pulso [2].

De todos los métodos de detección de bordes, la mayoría fueron diseñados para detectar bordes del tipo escalón. El tipo de borde escalón es caracterizado por el ruido, el contraste, la pendiente y su anchura. Este tipo de borde aparece en la coincidencia de dos regiones homogéneas que tienen niveles de gris diferentes entre sí. El borde se ubica en el punto donde se produce la discontinuidad de los niveles de gris, y se produce cuando aparece una sombra sobre un objeto o cuando un objeto oculta parcialmente a otro. En la figura 2.22 se evidencia un modelo del borde de tipo escalón o salto.

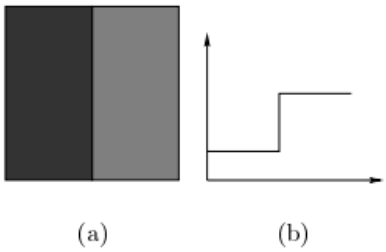


Figura 2.22: Borde de tipo escalón o salto: (a) imagen y (b) perfil de la función de intensidad a lo largo de una línea horizontal de la imagen [2].

Mientras que los bordes de tipo pico, cresta o pulso se producen por aquellas líneas que atraviesan la imagen, estas líneas se pueden producir por la presencia de objetos que estén juntos o por objetos delgados que se sitúan sobre un fondo. Se puede apreciar en la figura 2.23 los bordes provocados por dos líneas.

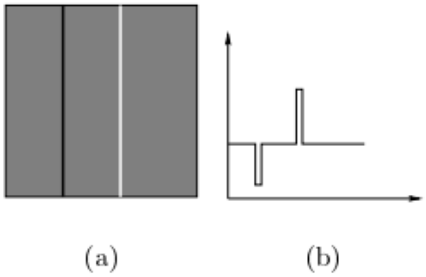


Figura 2.23: Bordes provocados por líneas: (a) imagen y (b) perfil de la función de intensidad a lo largo de una línea horizontal de la imagen [2].

Es importante también analizar la forma geométrica que tienen los bordes, para ello se debe conocer su clasificación:

- Líneas rectas que pueden ser horizontales, verticales, diagonales o con cualquier otra orientación.
- Líneas curvas.
- Uniones (*junctions*).
- Esquinas (*corners*).

En el momento que dos bordes físicos se encuentran, entonces la unión y esquina serán formadas. Además factores como los efectos de la iluminación o la presencia de oclusiones pueden provocar esquinas en la imagen. Existen algunos modelos de esquinas o uniones, estos son: T, L, Y, X. Se puede ver

un ejemplo gráfico de un borde del tipo T-unión en la figura 2.24, producido por la concurrencia de bordes de tipo escalón.

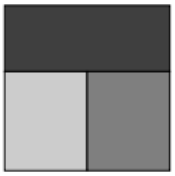


Figura 2.24: Imagen con un borde del tipo T-unión [2].

Clasificación de los algoritmos de detección de bordes en imágenes monocromáticas.-

Esta clasificación se realiza considerando los detectores de bordes como un elemento integrado en un sistema de visión de acuerdo a lo establecido en [2]:

- **Detectores autónomos:** Este tipo de algoritmos no utilizan ningún conocimiento a priori, esto quiere decir que no se encuentran influenciados por otros componentes del sistema de visión ni por información contextual. Este grupo de detectores realizan un procesamiento local para la extracción de los bordes, de tal manera que los bordes son detectados a través del análisis de los puntos de su entorno. Los detectores autónomos no se limitan tan solo a imágenes específicas sino también son apropiadas para sistemas de visión de propósito general, por lo tanto son muy flexibles en su aplicación. Los detectores autónomos fueron clasificados en dos clases de técnicas de detección de bordes, estos son:

    - **Técnicas secuenciales:** En este tipo de técnicas, los resultados obtenidos por el detector en algunos puntos examinados

previamente, son considerados el punto de partida para definir si un punto de la imagen pertenece o no a un borde.

- **Técnicas paralelas:** En estas técnicas el punto de partida que define si un punto pertenece o no a un borde depende del mismo punto y de su entorno, de tal manera que se hace posible que los operadores puedan ser aplicados de manera simultánea a todos los puntos de la imagen.

- **Detectores contextuales:** A diferencia de los algoritmos del grupo de los detectores autónomos, los detectores contextuales hacen uso del conocimiento a priori del borde o la escena en que van a ser procesados. Es decir, previo a aplicar estos métodos necesariamente deben existir resultados de otros componentes del sistema de visión del cual dependen. Esto implica la existencia de limitaciones en su funcionamiento a un contexto preciso donde las imágenes procesadas siempre incluyen los mismos objetos.

Es de indicar que tanto los detectores autónomos como los detectores contextuales se diferencian en [2]:

- Los filtros de suavización.
- Los operadores de diferenciación.
- Los procesos de identificación.
- Los objetivos.
- La complejidad computacional y los modelos matemáticos usados en su diseño.

Sin embargo ambos grupos de algoritmos utilizan las tres fases de detección de bordes que se describirán en un apartado más adelante.

2.2.1.2. BORDES EN IMÁGENES EN COLOR

En este apartado no se profundizará sobre las técnicas o algoritmos para la detección de bordes en imágenes en color, esto debido a que no serán aplicadas en el proceso de desarrollo de la tesis, sobretodo en la parte práctica, ya que para esto se utilizarán técnicas y métodos basadas en detección de bordes en imágenes monocromáticas.

Un borde de una imagen en color está definido como una discontinuidad en el espacio tridimensional de color, donde las discontinuidades son producidas por cambios en una o en varias de las componentes del espacio de color.

Existe una clasificación de bordes en color y que a continuación se describirá brevemente:

Clasificación de los bordes según su origen.-

La clasificación de los bordes según su origen es la siguiente:

- Bordes provocados por formas geométricas.
- Bordes provocados por reflejos.
- Bordes provocados por sombras.
- Bordes provocados por cambios materiales.

Existen algunos métodos que analizan la sensibilidad o invarianza de los módulos de los gradientes calculados en tres espacios de color diferentes, de tal manera que mediante estos métodos se pueda determinar el origen de un borde en color.

Clasificación de los bordes según los cambios en un campo vectorial.-

De acuerdo a esta clasificación se dice que si una imagen en color es tomada en cuenta como un espacio vectorial tridimensional, entonces un borde de

color se define como una discontinuidad significativa en el campo vectorial que representa la función de la imagen en color. Por consiguiente un borde de tipo escalón se caracterizaría por un cambio abrupto en el campo vectorial, mientras que un cambio gradual se correspondería con un borde de rampa.

Clasificación de los bordes según la información de color de su alrededor.-

Esta clasificación se caracteriza porque los bordes de color han sido determinados en función de la información de color existente a su alrededor. Por lo tanto se encarga de identificar el perfil de cada borde según las características de color en el espacio RGB que poseen los dos lados contiguos al borde de acuerdo a lo especificado en [2].

En la figura 2.25 se pueden ver los bordes provocados por fenómenos físicos.

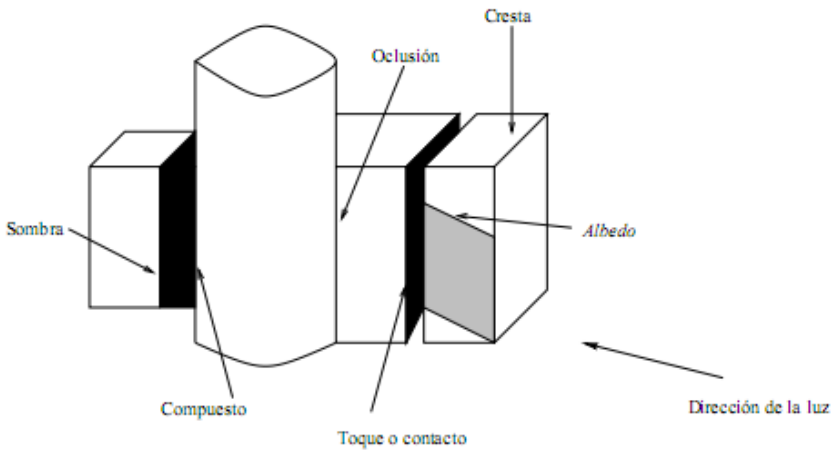


Figura 2.25: Bordes provocados por fenómenos físicos [2].

A continuación se procede a describir aquellos fenómenos físicos considerados para obtener los bordes en imágenes en color de acuerdo a la descripción establecida en [2]:

- **Albedo:** Se produce un cambio a través del borde pero no existe ninguna discontinuidad sustancial de profundidad.

- **Oclusión:** Dos regiones que pertenecen a objetos diferentes se encuentran en el borde. No existe ninguna sombra significativa entre los dos objetos.

- **Sombra:** Dos regiones pertenecen a una región homogénea de un objeto. Una de las regiones se corresponden con la sombra proyectada por otro objeto y la otra región está directamente iluminada por una fuente de luz normal.

- **Cresta:** Dos regiones se juntan en una "cresta" o "valle" de un objeto.

- **Compuesto:** Dos objetos se solapan, proyectando uno de ellos su sombra sobre el otro.

- **Contacto:** Dos regiones pertenecen a dos objetos que se tocan o están muy próximos. La iluminación es atenuada en el hueco que separa a los objetos.

Cabe indicar que los fenómenos físicos descritos no siempre serán excluyentes, ya que se podrán combinar entre uno y otro como por ejemplo los fenómenos provocados por un "contacto" y una "composición" pueden ser combinados.

2.2.2. FASES EN LA DETECCIÓN DE BORDES EN IMÁGENES MONOGROMÁTICAS

En el apartado 2.2.1 donde se habló sobre los tipos de bordes que existen, se mencionó que la mayoría de los algoritmos de detección de bordes en imágenes monocromáticas han sido diseñados para bordes de tipo salto o escalón, por lo

tanto, para llevar a cabo la detección de este tipo de bordes se deben seguir las tres fases que se observan en la figura 2.26.

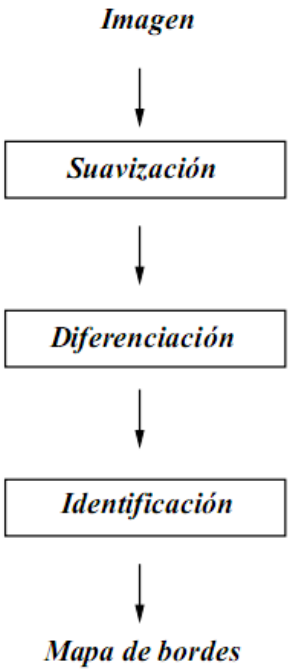


Figura 2.26: Fases de la detección de bordes [2].

Las tres fases para lograr la detección de bordes buscan que la identificación de los bordes sea correcta en lo posible. Estas fases se encuentran ligadas entre sí, ya que la fase de la suavización busca regularizar la siguiente fase que es la diferenciación, y finalmente la identificación de los bordes dependerá de los resultados arrojados por las dos fases anteriores, es decir, si la operación en la etapa del suavizado consigue reducir el ruido sin pérdida de información, entonces la supresión de los bordes falsos se puede lograr con mayor facilidad.

De acuerdo a lo anotado en [2]; se dice que las especificaciones de un detector de bordes basadas en estas tres fases es incompleta, esto debido a que no está incluida información precisa del contexto en el cual puede ser utilizada de mejor

manera, ni tampoco se incluye información acerca de la escala de computación, y también porque existen otros tipos de bordes tales como líneas, uniones y esquinas que necesitan de métodos diferentes para su extracción. Por consiguiente se ve la necesidad de determinar un método en particular para hacer explícito el cómo se va a elegir la escala y cómo se va a seleccionar un detector de bordes para una aplicación concreta.

En seguida se describe cada una de las tres fases que permiten realizar la detección de bordes:

Suavización.-

En la etapa del suavizado de la imagen se busca eliminar el ruido de la imagen, de tal manera que la imagen se encuentre regularizada, permitiendo que la fase de diferenciación se logre plantear adecuadamente para garantizar que la detección de los bordes se haga de manera correcta.

La suavización de la imagen tiene ciertas características que son las siguientes [2]:

- **Efectos positivos:** Reduce el ruido presente en la imagen y asegura una robusta detección de los bordes.

- **Efectos negativos:** Pérdida de información.

Considerando las características descritas anteriormente que no son más que los efectos positivos y negativos que acarrea la suavización de la imagen, se pretende conseguir como objetivo final la detección de bordes que asegure un compromiso favorable entre la reducción de ruido y la conservación del borde.

Para realizar la operación de la suavización de una imagen se requiere de la aplicación o intervención de métodos específicos, dentro de los cuales se encuentran los siguientes [2]:

- Filtros espaciales lineales, dentro de los cuales los más utilizados son los filtros de pasa bajo y el filtro de la Gaussiana.
- Filtros de suavización mediante aproximaciones.

Diferenciación.-

Posterior al suavizado de la imagen se procede a la etapa de la diferenciación de la imagen, donde se pretende evaluar las derivadas de la imagen con el fin de resaltar las características de los bordes. Ya se había mencionado que los bordes están presentes donde existen discontinuidades o cambios bruscos en el nivel de intensidad, estos cambios de intensidad pueden ser acentuados a través de operaciones de diferenciación, obteniendo como resultado una representación de la imagen, de la cual será más factible la extracción de las propiedades de sus bordes.

La diferenciación se realiza mediante métodos basados en la primera y segunda derivada, por lo tanto se utilizan dos aproximaciones para detectar los cambios bruscos de intensidad en una imagen [2]:

- Localización de los extremos (máximos o mínimos) de la primera derivada de la función de intensidad.
- Localización de los cruces por cero (*zero crossings*) o transiciones de valores negativos a positivos, o viceversa, de la segunda derivada de la función de intensidad.

En la figura 2.18 se tienen dos imágenes con bordes de tipo escalón o salto, el perfil de la función de intensidad y sus derivadas de primer y segundo orden, en la imagen de la izquierda se visualiza una franja blanca sobre un fondo oscuro, mientras que en la imagen de la derecha está una franja oscura sobre un fondo blanco, de las dos imágenes se extrae el nivel de intensidad de los puntos ubicados a lo largo de una línea horizontal.

En seguida se hace una breve explicación del comportamiento de las derivadas ante la presencia de un borde en una imagen de acuerdo a lo especificado en [2]:

- **Primera derivada.-** El comportamiento consiste en que la primera derivada toma valores positivos cuando hay una transición desde una zona a otra con una intensidad mayor o más clara, mientras que cuando la transición es hacia zonas con menor intensidad, entonces la derivada toma valores negativos, finalmente cuando se recorre una zona con niveles de intensidad constante, la derivada toma un valor nulo. Por consiguiente la primera derivada siempre presenta ya sea un extremo máximo o un extremo mínimo justamente en los puntos situados en un borde de la imagen procesada. Cabe indicar que la primera derivada en un punto de la imagen se consigue utilizando la magnitud del gradiente.

- **Segunda derivada.-** La segunda derivada se obtiene usando el operador Laplaciano, de tal forma que cuando se produce una transición entre zonas con diferentes niveles de intensidad, toma valores positivos en la zona de la transición asociada a la intensidad menor, y negativos en la zona asociada a la intensidad mayor. Por lo tanto, cuando la intensidad permanece constante la segunda derivada toma un valor nulo, mientras que para determinar si un punto pertenece a la parte oscura o clara de un borde, dependerá del signo de la segunda derivada. Finalmente, cuando un punto se encuentra ubicado exactamente en un borde, se produce un

cruce por cero, dicho de otra manera, la función corta el eje de abscisas en dicho punto, pasando de un valor positivo a otro negativo, o viceversa.

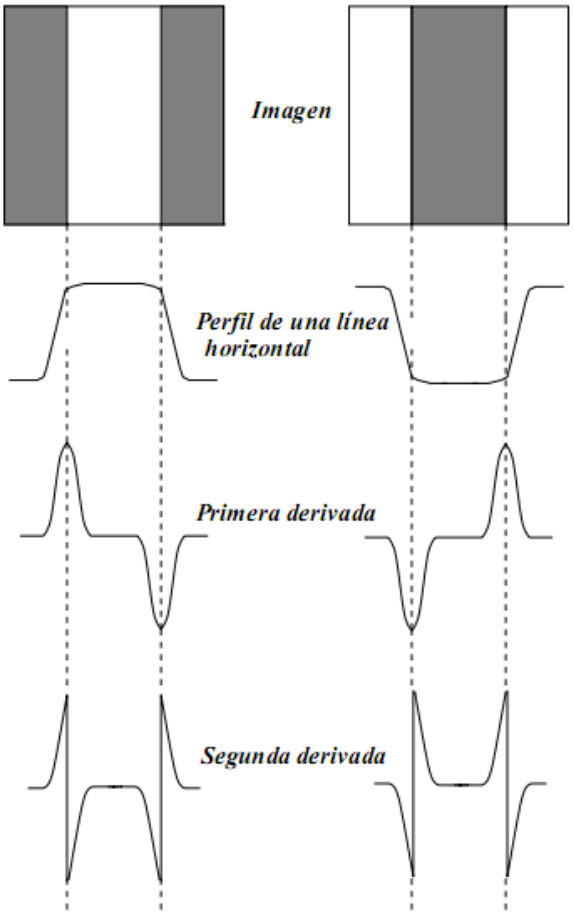


Figura 2.27: Comportamiento de las derivadas de primer y segundo orden ante la presencia de un borde [2].

El operador Laplaciano será revisado con mayor profundidad en un apartado posterior al igual que la función Gaussiana, ya que ambos serán utilizados en conjunto en la parte práctica del desarrollo de esta tesis.

Identificación de los bordes.-

Una vez procesada la imagen en la fase 1 y 2, se concluye con la etapa de la identificación de los bordes, donde se procede a localizar los bordes reales y suprimir aquellos bordes falsos. La fase de identificación de los bordes está enfocada a conseguir los siguientes objetivos:

- **La localización de los bordes.-** La identificación o localización de los bordes dependerá de los operadores que hayan sido aplicados en las etapas tanto de la suavización así como de la diferenciación, consecutivamente. Estos operadores son los siguientes tal como se indica en [2]:

    - **Métodos basados en el gradiente:** Mediante este tipo de métodos se localizan los bordes en los puntos donde la magnitud del gradiente alcanza un máximo local.

    - **Métodos basados en las derivadas de segundo orden:** A través de estos métodos basados en la segunda derivada, la localización de los bordes se realiza en los puntos donde se produce un cruce por cero de la segunda derivada.

- **La supresión de los bordes falsos.-** Es de trascendental importancia realizar la supresión de los bordes falsos detectados en la imagen previamente procesada por los operadores de la suavización y diferenciación, ya que se logra incrementar la proporción de la señal frente al ruido.

En un punto posterior se profundizará sobre los métodos basados en las derivadas de segundo orden, específicamente sobre el método basado en el operador Laplaciano.

2.2.3. TÉCNICAS DE DETECCIÓN DE BORDES BASADA EN LA PRIMERA DERIVADA

Las técnicas de detección de bordes basadas en la primera derivada se llevan a cabo a través del uso del operador gradiente, por consiguiente el gradiente de una imagen viene a ser la variación de intensidad que puede ocurrir de un punto "x" a un punto "y". Por lo tanto el objetivo primordial de este operador consiste en detectar los cambios de niveles de gris que se producen en zonas o regiones reducidas, de tal manera que se logre determinar la existencia de un borde si hay una discontinuidad en la función de intensidad de la imagen, o lo que es lo mismo, si la derivada de los valores de intensidad de la imagen da como resultado un máximo.

El gradiente de una función escalar bidimensional $f(x,y)$, en las coordenadas (x,y), se define como un vector de dos dimensiones, donde su dirección coincide con la dirección de la variación máxima de la función en el punto en cuestión, y su magnitud indica el valor de la variación por unidad de distancia y en la dirección de la máxima variación. El vector gradiente bidimensional se expresa de la siguiente manera:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial_x} f(x,y) \\ \frac{\partial}{\partial_y} f(x,y) \end{bmatrix} = \frac{\partial}{\partial_x}\overrightarrow{\imath} + \frac{\partial}{\partial_y}\overrightarrow{J} \qquad \textbf{(2.6)}$$

Donde G_x representa el gradiente calculado con respecto a "x", mientras que G_y constituye el gradiente obtenido con respecto a "y", siendo estos, producto del cálculo de las derivadas parciales con respecto a "x" y "y" sucesivamente.

Dentro de la imagen, sus puntos pertenecientes a los bordes son aquellos que dan un valor máximo en la magnitud del gradiente. El cálculo de la magnitud se realiza a través de la siguiente fórmula:

$$\left|G_{mag}\right| = \sqrt{G_x^2 + G_y^2} \quad \textbf{(2.7)}$$

Existen dos alternativas más que permiten calcular la magnitud del gradiente, mostrando una ventaja en cuanto a la eficiencia en su cálculo computacional, sin embargo los resultados obtenidos mediante la fórmula 2.9 son mucho más precisos, sobretodo en la detección de los bordes de tipo escalón o salto. Las dos alternativas se calculan mediante las siguientes fórmulas:

$$\left|G_{mag}\right| \approx |G_x| + \left|G_y\right| \quad \textbf{(2.8)}$$

$$\left|G_{mag}\right| \approx max\left(|G_x|, \left|G_y\right|\right) \quad \textbf{(2.9)}$$

Es necesario también el cálculo de la dirección del gradiente, esta dirección es perpendicular a la orientación del borde y se utiliza para localizar los bordes. Para obtener dicha dirección se procede al cálculo del ángulo con respecto al eje "x" haciendo uso de las derivadas parciales obtenidas previamente. A continuación se encuentra la fórmula para calcular la dirección del gradiente:

$$G_{dir} = \emptyset(x, y) = \tan^{-1}\frac{G_x}{G_y} \quad \textbf{(2.10)}$$

En la figura 2.28 se presenta el sistema para la detección de bordes bidireccionalmente basado en técnicas del gradiente, donde tenemos como entrada una función escalar bidimensional $f(x, y)$, de esta función se obtiene en primera instancia los gradientes parciales G_x y G_y, y seguidamente su magnitud $\left|G_{mag}\right|$, denominada gradiente de la imagen. Posteriormente se efectúa la comparación de la magnitud obtenida con un umbral previamente establecido, en esta comparación los puntos para los cuales la magnitud es más grande que el umbral serán candidatos a ser parte del contorno.

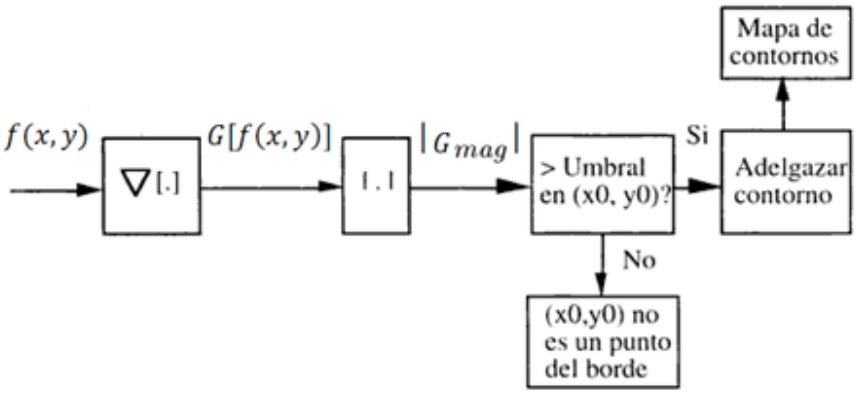


Figura 2.28: Sistema detector de bordes basado en el gradiente [26].

En el tratamiento de imágenes digitales se manejan funciones con valores discretos, esto implica que los gradientes G_x y G_y se obtienen mediante la convolución de la imagen de entrada con una máscara de convolución. En virtud de ello tenemos las siguientes fórmulas:

$$G_x = \frac{\partial}{\partial_x} = I(x,y) * h_1(x,y) \quad \textbf{(2.11)}$$

$$G_y = \frac{\partial}{\partial_y} = I(x,y) * h_2(x,y) \quad \textbf{(2.12)}$$

Donde las máscaras de convolución h_1 y h_2 se obtienen por medio de la aproximación por la diferencia de píxeles separados en filas y columnas respectivamente. Estas máscaras de convolución son representadas a través de las siguientes fórmulas:

$$h_1(x,y) = \frac{df}{dx} = f(x+1,y) - f(x,y) \quad \textbf{(2.13)}$$

$$h_2(x,y) = \frac{df}{dy} = f(x,y+1) - f(x,y) \quad \textbf{(2.14)}$$

En la figura 2.29 se puede verificar el comportamiento de los desplazamientos de la posición desde un píxel hacia otro. Donde el gradiente de fila G_x puede aproximarse por la diferencia de píxeles adyacentes de la misma fila, mientras que el gradiente de columna G_y se aproxima por la diferencia de píxeles adyacentes de la misma columna.

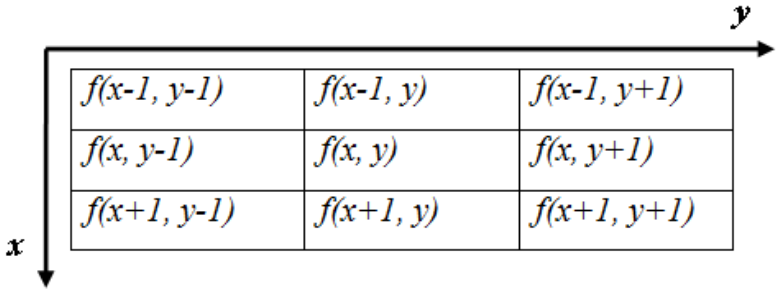


Figura 2.29: Desplazamiento desde un píxel hacia otro.

Las máscaras de convolución h_1 y h_2 se traducen en matrices de 3x3, como se indica en la figura 2.30.

$$h1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix} \qquad h2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

(a) **(b)**

Figura 2.30: Máscaras para obtener G_x y G_y respectivamente.

Las máscaras de la figura 2.30 no se utilizan con gran frecuencia, esto debido a que son sensibles al ruido. Por lo tanto a partir de las fórmulas 2.11 y 2.12 se pueden idear nuevas máscaras que apliquen el gradiente a cualquier dirección. Es así que han surgido algunas máscaras de convolución con el propósito de detectar los bordes de las imágenes de manera mucho más eficiente.

Los resultados de aplicar la detección de bordes basada en el gradiente se presentan en la figura 2.31. En este ejemplo se aplica el operador Sobel; en primera instancia se tiene la imagen en escala de grises de entrada, a esa imagen

se le aplica las máscaras de convolución vertical y horizontal para obtener como salida las imágenes *y* en las dos direcciones respectivamente, posteriormente se realiza la suma de estas imágenes resultantes. Posteriormente a esta suma se aplican dos umbralizaciones, el primero con una tonalidad de 80 y la segunda con una tonalidad de 130, estas tonalidades permiten obtener dos imágenes diferentes con sus bordes respectivos.

Cabe indicar que es muy importante seleccionar adecuadamente el valor del umbral para que los bordes sean detectados satisfactoriamente, como ya se ha indicado anteriormente, si el valor del gradiente es mayor al umbral, entonces, quiere decir que en ese punto existe un borde.

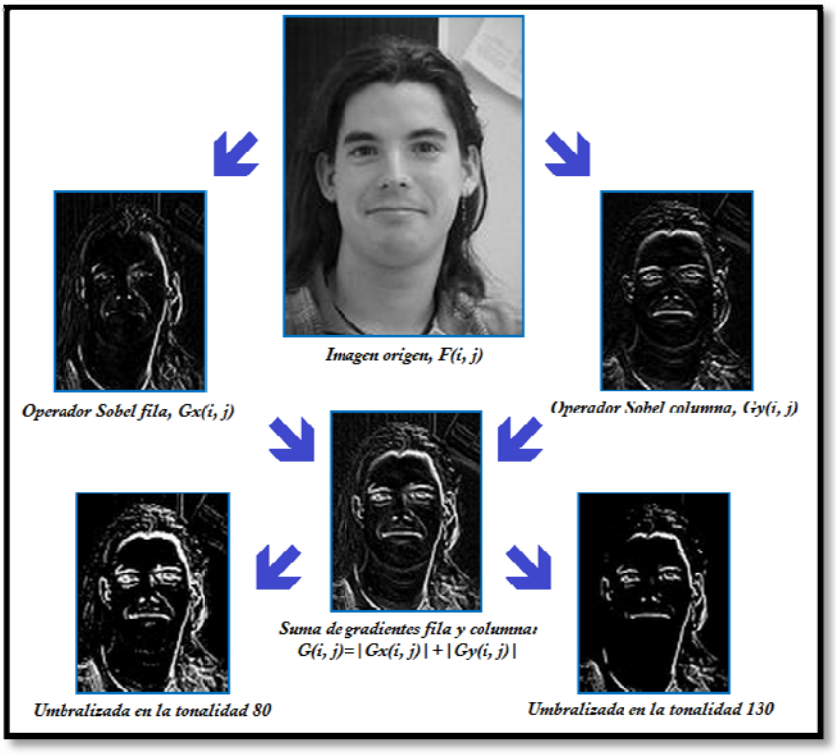


Figura 2.31: Resultados de la detección de bordes basado en el gradiente [29].

Existen algunas desventajas que se presentan en la detección de bordes basada en la primera derivada, de acuerdo a lo que se especifica en [1]:

Desventajas.-

- La fijación realizada por el usuario tanto de los umbrales así como también el tamaño de la máscara, afectará a la posición del borde.

- El gradiente presenta una excesiva dependencia con respecto a la dirección de barrido, por lo expuesto, las aristas cuyas pendientes están próximas a la dirección de barrido no se detectan con facilidad.

- La debilidad del gradiente en los puntos esquina provocará la pérdida de puntos relevantes y marcado de junturas.

La mayoría de los métodos de detección de bordes basados en la primera derivada, utilizan máscaras de convolución para implementar en forma digital las derivadas G_x y G_y, que corresponden al gradiente horizontal y vertical respectivamente.

A continuación se procede a realizar una breve descripción de algunos de estos métodos con sus peculiaridades y ventajas:

Operador de Roberts.-

Este operador está indicado para la detección de bordes diagonales, las máscaras que utiliza este método para la detección de bordes son las que se presentan en la figura 2.32.

0	1
-1	0

Diagonal 1

1	0
0	-1

Diagonal 2

Figura 2.32: Operadores de Roberts.

Como ventajas se puede mencionar que este operador presenta una respuesta muy buena en la detección de bordes diagonales, también realiza una buena

localización de los bordes y es simple y rápido el cálculo computacional. Sin embargo presenta algunas desventajas tales como ser extremadamente sensible al ruido, lo que hace que este operador tenga cualidades pobres de detección de los bordes en relación con otros operadores, además solamente marca los puntos del borde sin proporcionar información acerca de la orientación del borde y la anchura del borde es de varios píxeles.

Operador Prewitt.-

Este operador brinda la ventaja de que suaviza la imagen previo a la detección de bordes, de esta forma elimina el ruido y hace posible que la detección sea mucho más efectiva. Esto es factible por que emplea máscaras de 3x3 píxeles, a diferencia del operador de Roberts que utiliza máscaras de 2x2, y por lo tanto no tiene incorporado este suavizado.

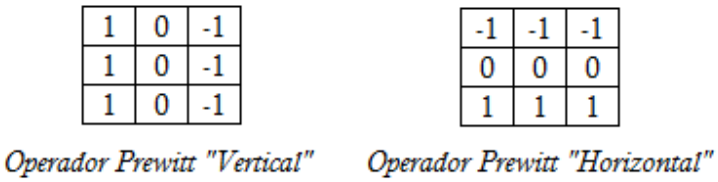


Figura 2.32: Operadores de Prewitt.

El Operador de Prewitt brinda grandes respuestas en bordes tanto horizontales como verticales, además proporciona la magnitud y la dirección del borde. Sin embargo como desventajas se puede mencionar que es lento en su cálculo, también la respuesta en bordes diagonales es deficiente, y la anchura de los bordes es de varios píxeles.

Operador Sobel.-

El operador Sobel es una variante del operador de Prewitt, esto se debe a que Sobel utiliza $[1 \quad 2 \quad 1]$ en reemplazo de $[1 \quad 1 \quad 1]$ que utiliza Prewitt en su máscara de convolución, esta variación hace que el operador de Sobel realice un

mejor suavizado que Prewitt, sin embargo no hay mucha diferencia en cuanto a resultados. En la figura 2.34 están las máscaras que utiliza este operador.

1	0	-1
2	0	-2
1	0	-1

Operador Sobel "Vertical"

-1	-2	-1
0	0	0
1	2	1

Operador Sobel "Horizontal"

Figura 2.33: Operadores de Sobel.

Así como el operador de Prewitt, el operador Sobel brinda buenas respuestas en bordes horizontales y verticales. Sin embargo tiene la desventaja de ser más sensible a los bordes diagonales que el de Prewitt.

Una comparación de los resultados que arrojan los tres métodos revisados previamente se puede verificar en la figura 2.35.

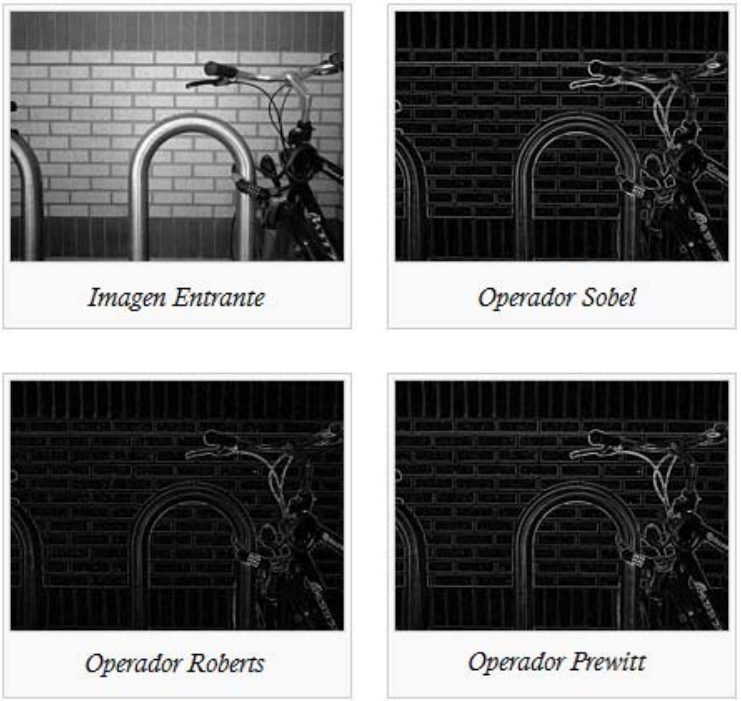


Figura 2.34: Detección de bordes con varios operadores [30].

2.3. KERNEL GAUSSIANO

Para conseguir una comprensión profunda sobre la aplicación del *Kernel* Gaussiano para el suavizado de imágenes, es necesario primero realizar una descripción generalizada de este operador.

2.3.1. DESCRIPCIÓN GENERAL DE LA FUNCIÓN GAUSSIANA

El matemático alemán, Carl Friedrich Gauss quien vivió desde 1777 hasta 1855, es el inventor del *Kernel* Gaussiano que lleva su nombre en su honor.

Esta función está definida por la siguiente expresión:

$$f(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad \textbf{(2.15)}$$

Donde sus parámetros deben cumplir las siguientes condiciones:

- Dominio: $x \in R$
- La media: $\mu \in R$
- Desviación estándar: $\sigma > 0$
- Varianza: σ^2
- Altura de la campana de Gauss: $\frac{1}{\sqrt{2\pi}\,\sigma} > 0$

En la figura 2.36 se detalla el comportamiento de la función por medio del cambio de sus valores de la media μ y la desviación estándar σ, donde se puede verificar su simetría con respecto a la media, que le da una forma de campana, motivo por el cual se le conoce como campana de Gauss.

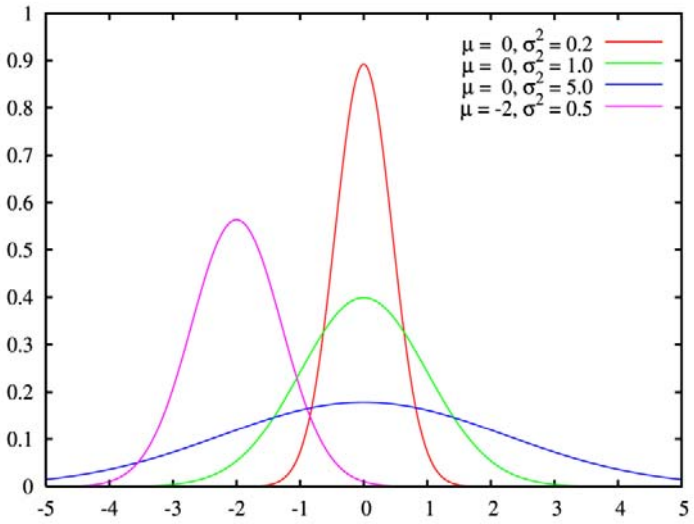


Figura 2.35: Curvas de la función Gaussiana con cambios de valor de sus parámetros [40].

Una vez observado el comportamiento de la función, queda evidenciado que la forma de la campana de Gauss es dependiente tanto de la media μ, y de la desviación estándar σ.

Asimismo se puede ver que la media indica la posición de la campana, de tal forma que al asignarle diferentes valores de μ, la función se desplaza a lo largo del eje horizontal. Mientras que al ser de mayor valor la desviación estándar más dispersos se encontrarán los datos con relación a la media, lo que conlleva a que la curva de la función se vaya aplanando. Por el contrario, si el valor de la media es pequeño, existirá una gran probabilidad de que los datos que se obtengan estén cercanos al valor medio de la distribución.

Finalmente es importante indicar que el área calculada bajo la función Gaussiana será siempre igual a la unidad, esto debido a su parámetro de normalización.

2.3.2. APLICACIÓN DEL FILTRO GAUSSIANO PARA EL SUAVIZADO DE IMÁGENES

Adentrándonos específicamente en el tema de la aplicabilidad de la función Gaussiana para el tratamiento de las imágenes, se debe indicar que este filtro se lo utiliza con la finalidad de suavizar la imagen. Esto consiste en disminuir la sensibilidad ante el ruido y filtrar las frecuencias altas.

Desde la perspectiva de la frecuencia, la varianza determina el ancho de banda del filtro pasa bajo, de tal forma que si la varianza es mayor, entonces el ancho de banda será menor. Mientras que el lóbulo del módulo de la respuesta de la frecuencia, depende también de la varianza. En este dominio no existe la presencia de lóbulos secundarios dentro del módulo. Hay que indicar que el filtro Gaussiano es de fase lineal.

La función Gaussiana también conocida como campana de Gauss, se define en múltiples dimensiones, sin embargo el tratamiento de las imágenes se llevará a cabo en dos dimensiones. A continuación se presentan las fórmulas de la función Gaussiana en el dominio de la frecuencia:

Una sola dimensión (1-D):

$$G_{1D}\,(x;\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma}\; e^{-\frac{x^2}{2\sigma^2}} \qquad \textbf{(2.16)}$$

Dos dimensiones (2-D):

$$G_{2D}\,(x,y;\sigma) = \frac{1}{2\pi\sigma^2}\; e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \textbf{(2.17)}$$

N dimensiones (N-D):

$$G_{ND}\,(\vec{x};\sigma) = \frac{1}{\left(\sqrt{2\pi}\,\sigma\right)^N}\; e^{-\frac{|\vec{x}|^2}{2\sigma^2}} \qquad \textbf{(2.18)}$$

Para operar este filtro sobre imágenes se considera que la media tiene que ser cero y la varianza determinada a conveniencia.

La aplicación del filtro Gaussiano sobre la imagen se realiza a través de la convolución, es decir, si se considera que la función Gaussiana es $g(x)$, y la imagen una señal $f(x)$, entonces la convolución de la función con la señal da como resultado una nueva señal suavizada $h(x)$, donde el valor de cada uno de los puntos obtenidos viene a ser el resultado de promediar con distintos pesos los valores vecinos a ambos lados del punto central.

La importancia de la desviación estándar es determinante en el momento de definir el grado de suavizado de la imagen. Esto es, cuando mayor sea σ, se tiene mayor consideración a los píxeles lejanos, y como consecuencia se consigue un suavizado mucho mayor. Mientras que cuando σ disminuya, los valores de los píxeles vecinos irán recibiendo menor ponderación, lo que conlleva a que la reducción del ruido sea menor.

La función Gaussiana bidimensional se utiliza para realizar el cálculo de cada uno de los coeficientes de una máscara, como una aproximación discreta. Esta máscara es el *Kernel* con el que se convoluciona una imagen para producir el efecto de desenfoque Gaussiano.

Se presenta a continuación un ejemplo, donde se realiza el suavizado de una imagen. Para el efecto, se utiliza una máscara de tamaño $17x17$ y $\sigma = 2$. En la figura 2.37 se encuentra la gráfica de la función resultante y en la figura 2.38 se visualiza la imagen original y la imagen desenfocada.

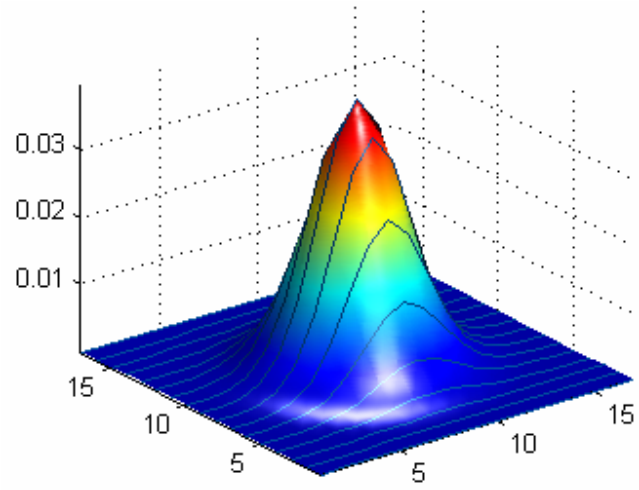


Figura 2.36: Representación del filtro Gaussiano bidimensional, con una máscara de $17x17$ y $\sigma = 2$ [43].

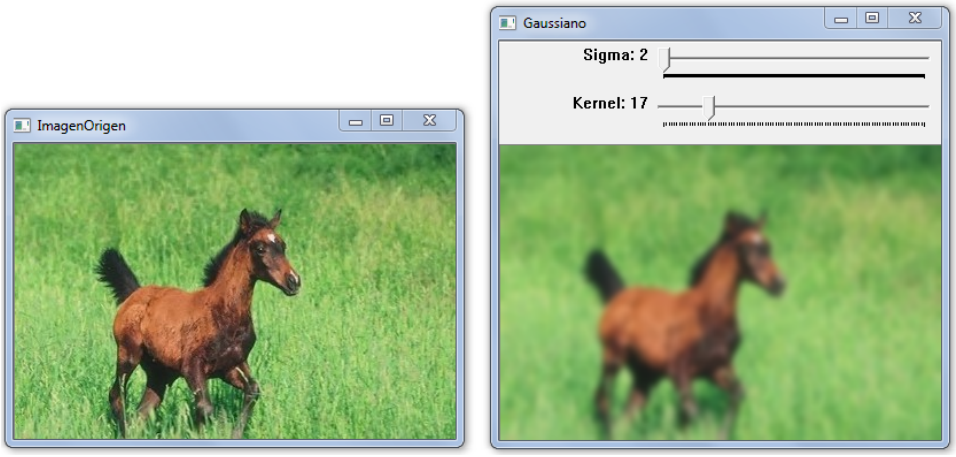


Figura 2.37: Imagen suavizada con el filtro Gaussiano, con una máscara de $17x17$ *y* $\sigma = 2$.

Propiedades principales.-

Resumiendo lo antes expuesto, se podría decir que las principales propiedades de este filtro son [44]:

- El grado de filtrado es controlado por σ.

- Simetría Rotacional.- Produce el mismo efecto en todas las direcciones.

- Tiene un único lóbulo (pico).- Este lóbulo se encuentra ubicado sobre la media de la campana de Gauss, donde el peso de los píxeles decrece con la distancia al centro de la campana, de tal manera que, mientras más alejado está un píxel, menos significativo es.

- Preserva las bajas frecuencias y tiende a eliminar las altas, por lo tanto pertenece a la clase de filtros lineales, pasa bajo.

- Tiene una relación sencilla e intuitiva entre el tamaño de sigma y el suavizado.

- La función Gaussiana bidimensional es separable en dos funciones unidimensionales.

Normalización.-

La fórmula unidimensional de la función Gaussiana contiene el término $\frac{1}{\sqrt{2\pi}\,\sigma}$, que es el valor de la altura de la campana de Gauss, pero a su vez se convierte en la constante de normalización que determina que el área de la campana sea siempre la unidad.

Con una explicación exhaustiva; la integral de la función exponencial devuelve como resultado un valor diferente de la unidad, que corresponde a su área:

$$\int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}}\,\partial x \;=\; \sqrt{2\pi}\,\sigma \qquad \textbf{(2.19)}$$

Mientras que con la constante de normalización se obtiene un *Kernel* normalizado, esto quiere decir que el cálculo de su integral sobre su dominio para todo σ, da como resultado siempre la unidad:

(2.20)

Por lo tanto, al incrementar el valor de , se reduce sustancialmente la amplitud de la función. Un ejemplo sobre el comportamiento de la función normalizada para diferentes valores de la desviación estándar, se puede ver en la figura 2.39.

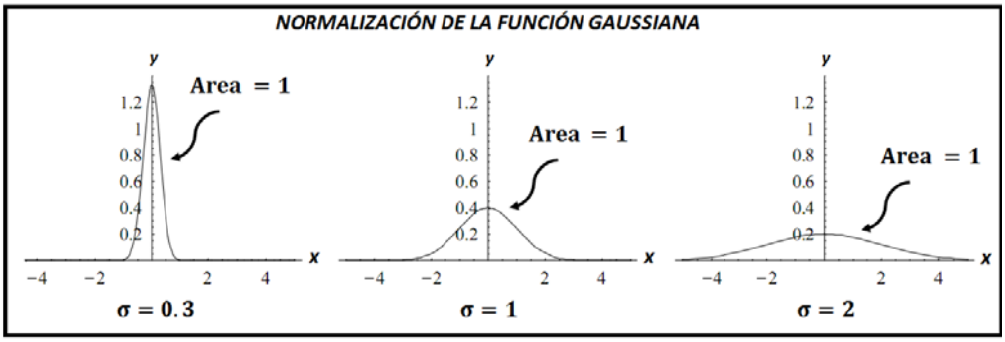


Figura 2.38: Función Gaussiana a escalas . El kernel está normalizado, por lo tanto el área bajo la curva es siempre la unidad [42].

La importancia de la normalización de Gauss en el tratamiento de imágenes radica en que esta asegura que el promedio del nivel de gris de la imagen siga siendo el mismo cuando se realice el proceso de suavizado o desenfoque. En otras palabras lo que se consigue es la invarianza promedio de los niveles de gris.

Separabilidad del filtro Gaussiano.-

El filtro Gaussiano por la propiedad de la simetría rotacional, es el único operador que puede descomponerse. Es así, que el filtro bidimensional se puede separar en dos filtros unidimensionales, tanto en filas como en columnas, correspondientes a sus componentes e . La descomposición del filtro se realiza de la siguiente manera:

(2.21)

Donde:

$$g_1(x;\sigma) = \left(\frac{1}{\sqrt{2\pi}\,\sigma}\; e^{-\frac{x^2}{2\sigma^2}}\right) \quad \textbf{(2.22)}$$

$$g_2(y;\sigma) = \left(\frac{1}{\sqrt{2\pi}\,\sigma}\; e^{-\frac{y^2}{2\sigma^2}}\right) \quad \textbf{(2.23)}$$

Siendo $g_1(x;\sigma)$ y $g_2(y;\sigma)$los filtros unidimensionales pertenecientes a "x" e "y" respectivamente.

Ahora, considerando la propiedad asociativa y conmutativa de la convolución, se pueden llevar a cabo las siguientes relaciones operacionales entre los dos filtros unidimensionales y la imagen I:

$$I * G_{2D} = (I * g_1) * g_2 \quad \textbf{(2.24)}$$

$$I * G_{2D} = (I * g_2) * g_1 \quad \textbf{(2.25)}$$

En la ecuación 2.24 se realiza una convolución en filas de la imagen con $g_1(x;\sigma)$, y posteriormente al resultado obtenido se le aplica la convolución por columnas con $g_2(y;\sigma)$. Mientras que en la ecuación 2.25 se convoluciona a la imagen por columnas y al resultado se le ejecuta una convolución por filas.

El estudio de esta propiedad es importante, porque permite reducir el número de operaciones que se llevan a cabo en el proceso del suavizado de cada píxel de la imagen. Esto debido a que la convolución con el filtro unidimensional necesita k operaciones para obtener el píxel de salida, mientras que, con el filtro bidimensional se requieren k^2 operaciones. Esto indica que existe una ventaja computacional de la convolución separable sobre la no-separable de $k^2/2k$.

Analizando en la forma discreta, la equivalencia de la separabilidad se lleva a cabo análogamente a la fórmula 2.21:

$$G_{2D}\ (i,j;\sigma) = g_1(i;\sigma) * g_2(j;\sigma) \qquad \textbf{(2.26)}$$

Donde g_1 se convierte en un vector de filas y g_2 en un vector de columnas.

A continuación se realiza una demostración de la aplicación de la propiedad de separabilidad del *Kernel* Gaussiano:

En primera instancia, en la figura 2.40 se demuestra que a partir de la convolución de dos máscaras Gaussianas unidimensionales, se obtiene una nueva máscara bidimensional. Con esta nueva máscara se convoluciona la ventana de la imagen de entrada, y se obtienen los píxeles de salida.

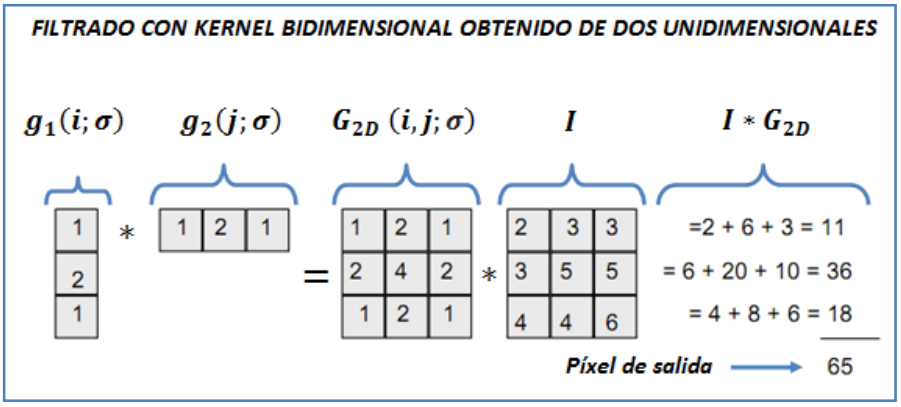


Figura 2.39: *Kernel* bidimensional obtenido a partir de dos máscaras unidimensionales, y aplicación del filtro bidimensional [45].

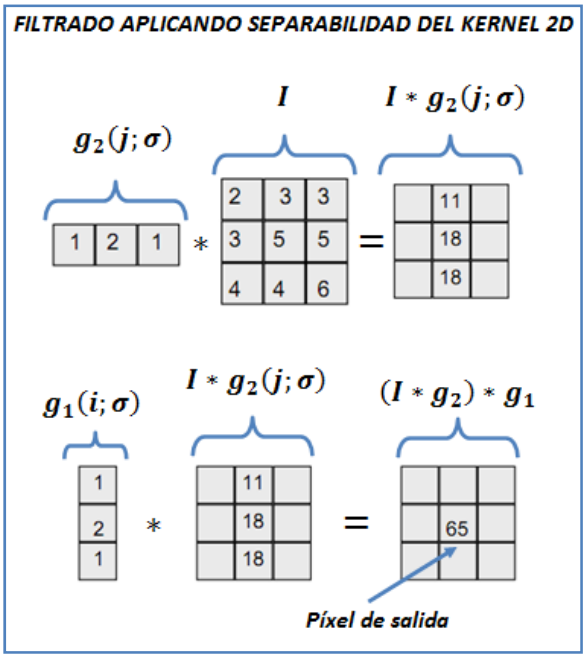


Figura 2.40: Aplicación de la propiedad de separabilidad de la función Gaussiana [45].

En segunda instancia se procede a demostrar en la figura 2.41, la aplicación de la propiedad de separabilidad del *Kernel* Gaussiano. Se puede ver que se convoluciona la primera máscara unidimensional con la ventana de la imagen de entrada, la máscara resultante se convoluciona ahora con la segunda máscara unidimensional, teniendo como salida el mismo resultado que se presenta aplicando la máscara bidimensional en la figura 2.40.

Filtro Gaussiano en el dominio espacial.-

Dado que la imagen se almacena como un conjunto de píxeles discretos, entonces es necesario que se realice una aproximación discreta a la función Gaussiana, de tal manera que se pueda realizar el tratamiento en el dominio espacial.

Para el efecto se aplica la convolución discreta que viene dada por la siguiente ecuación:

$$G_{2D}\,(i,j;\sigma) = \frac{1}{2\pi\sigma^2}\sum_k\sum_l e^{-\frac{x^2+y^2}{2\sigma^2}} f(i-k, j-l) \qquad \textbf{(2.27)}$$

Es importante puntualizar que en la práctica los sumatorios de la fórmula 2.27, están limitados a un entorno determinado de vecindad del píxel central, que es objeto de análisis. Es decir, se realizará una convolución de la imagen con una máscara cuadrada.

A continuación se presentan los problemas ocasionados al momento de discretizar la máscara:

Problemas en el proceso de discretización de la máscara.- De acuerdo a lo apuntado en [25], se dice que al discretizar la máscara Gaussiana aparecen varios coeficientes finitos a emular a la función Gaussiana continua, es decir la aparición de lóbulos secundarios.

Con el fin de evitar la aparición de lóbulos secundarios en la respuesta frecuencial de la máscara, se han planteado varias relaciones entre sigma y el tamaño de la máscara, y una de las relaciones más aceptadas es la siguiente:

$$w \geq 3c \qquad \textbf{(2.28)}$$

Donde

$$c = 2\sqrt{2\sigma^2} \qquad \textbf{(2.29)}$$

Siendo *w* el tamaño de la máscara, y que depende del valor de sigma.

En la etapa de la experimentación se va a considerar la relación descrita entre sigma y el tamaño del *Kernel.*

Desventajas del filtro Gaussiano.-

Finalmente se debe mencionar que este filtro presenta desventajas. Estas se ven reflejadas en la disminución de la nitidez, el aumento de borrosidad y la pérdida de detalles.

En la figura 2.42 se evidencian las desventajas de este filtro.

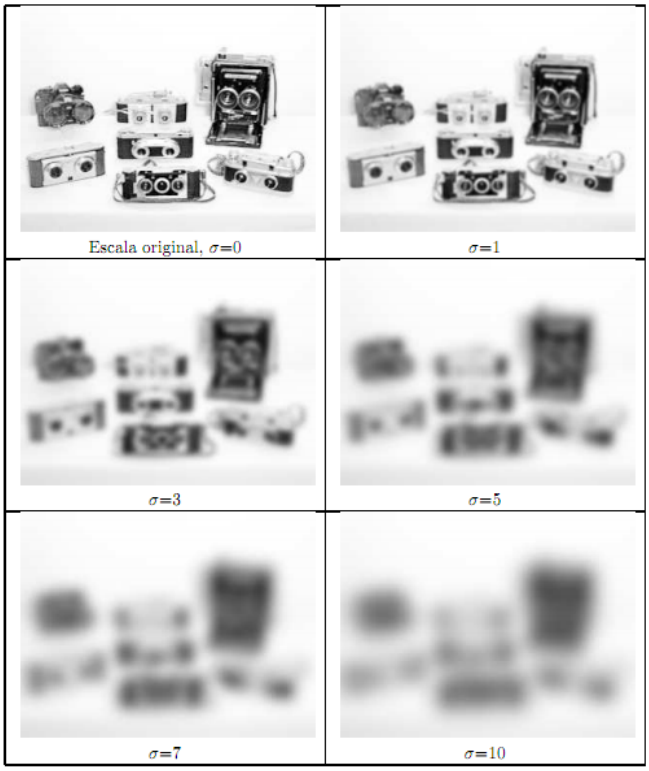


Figura 2.41: Aplicación del filtro Gaussiano con variación de σ[47].

Está claro que para evitar la pérdida de información en los bordes de los objetos de la imagen, se debe tener cuidado al momento de definir el valor de sigma.

2.4. OPERADOR LAPLACIANO

En esta tesis se realizará la segmentación basada en la detección de contornos, esto consiste en delimitar el borde de un objeto y segmentar los píxeles dentro del contorno.

Para llevar a cabo este tipo de segmentación, se usará el operador Laplaciano, el mismo que cumple la función de detectar los bordes. Este operador es la suma de las segundas derivadas calculadas en las direcciones$x$ ey, tal como se observa en la ecuación 2.30:

$$\nabla^2 f(x,y) = \nabla^2 x + \nabla^2 y = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \qquad \textbf{(2.30)}$$

El cálculo de la segunda derivada permite tener una respuesta mucho más fuerte ante detalles como puntos aislados o líneas, por consiguiente se aplica el operador Laplaciano para la detección de estas discontinuidades.

La forma como encuentra los bordes de una imagen este operador, consiste en detectar los cruces por cero de la segunda derivada. Estos cruces por cero se dan en los puntos donde el valor de píxeles adyacentes cambia de signo.

Por el hecho de que las imágenes son tratadas digitalmente, la ecuación anterior se puede implementar tomando las segundas derivadas en filas y columnas. Esto consiste en realizar las diferencias de los gradientes como se indica a continuación:

$$\begin{aligned} \nabla^2 x &= [f(x+1,y) - f(x,y)] - [f(x,y) - f(x-1,y)] \\ &= [f(x+1,y) - 2f(x,y) + f(x-1,y)] \end{aligned} \qquad \textbf{(2.31)}$$

$$\begin{aligned} \nabla^2 y &= [f(x,y+1) - f(x,y)] - [f(x,y) - f(x,y-1)] \\ &= [f(x,y+1) - 2f(x,y) + f(x,y-1)] \end{aligned} \qquad \textbf{(2.32)}$$

Ahora se procede a sumar los componentes de la ecuación ∇^2x con los componentes de la ecuación ∇^2y:

$$\begin{aligned}\nabla^2 x + \nabla^2 y &= [\, f(x+1,y) - 2f(x,y) + f(x-1,y) + \\ &\quad + f(x,y+1) - 2f(x,y) + f(x,y-1)] \\ &= [\, f(x+1,y) + f(x-1,y) + f(x,y+1) + \\ &\quad + f(x,y-1) - 4f(x,y)]\end{aligned} \tag{2.33}$$

De las ecuaciones 2.31 y 2.32 se obtienen; el vector filas y el vector columnas respetivamente:

$$\nabla^2 x = [1 \quad -2 \quad 1] \tag{2.34}$$

$$\nabla^2 y = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \tag{2.35}$$

Finalmente al sumar los vectores; fila y columna se obtiene la máscara Laplaciana:

$$\nabla^2 f(x,y) = \nabla^2 x + \nabla^2 y = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.36}$$

Claramente se observa que el píxel central toma el valor negativo de la suma de todos sus vecinos, de tal forma que garantiza que la suma aritmética de todos los píxeles sea cero. Por lo tanto, el requisito básico para determinar la máscara Laplaciana de cualquier dimensión consiste en garantizar que la suma algebraica de todos los coeficientes de la máscara de como resultado cero.

Al ser un operador isotrópico1, tiene la gran ventaja sobre métodos de primera derivada de detectar los bordes en todas las direcciones espaciales, lo que significa que es invariante bajo rotación; es decir, si primero el operador Laplaciano es aplicado a una imagen y luego la imagen es rotada, daría el mismo resultado si se realiza al revés.

Mientras que en la figura 2.43 se pueden ver los resultados luego de aplicar diferentes máscaras Laplacianas:

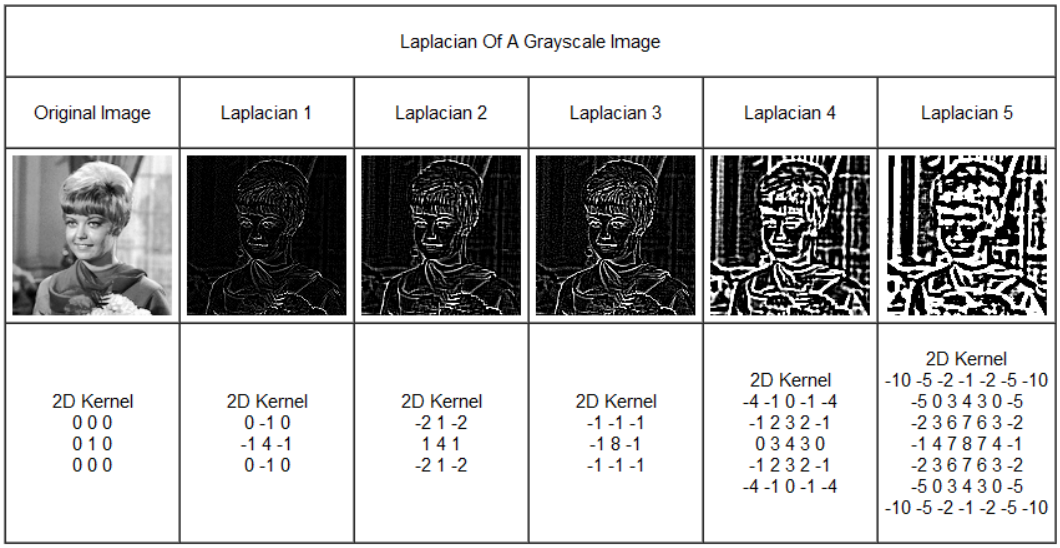


Figura 2.42: Detección de bordes mediante el filtro Laplaciano en una imagen en escala de grises [50].

Este operador presenta algunas desventajas que se mencionan a continuación:

- Es extremadamente sensible al ruido.
- No brinda información sobre la dirección del borde.
- Produce bordes dobles.

[1] isotrópico: Se refiere al hecho de que ciertas magnitudes vectoriales conmensurables, dan resultados idénticos con independencia de la dirección escogida para dicha medida.

En la figura 2.44 se puede apreciar como la función Laplaciana detecta los bordes mediante el cruce por cero:

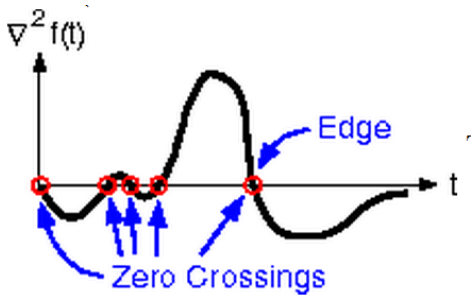


Figura 2.43: Gráfica de la segunda derivada que detecta los bordes mediante el cruce por cero [52].

En la figura 2.44 claramente podemos observar que a más de detectar el borde que corresponde a la imagen, también está detectando bordes falsos; esto debido al ruido que existe en la imagen, y puesto que se hace la detección de cruces por cero, cualquier variación por más pequeña que sea, está propensa a generar un punto de borde. Para minimizar este problema, se han planteado algunas alternativas, sin embargo vamos a mencionar una en concreto. Se trata de la aplicación del filtro Gaussiano sobre la imagen, para eliminar el ruido y por ende minimizar la detección de bordes falsos.

En el siguiente punto se da a conocer la combinación entre el filtro Gaussiano y el operador Laplaciano, que da como origen al operador Laplaciano del Gaussiano.

2.4.1. OPERADOR LAPLACIANO DEL GAUSSIANO

Para minimizar el excesivo ruido de la imagen que ocasiona una detección deficiente de bordes por medio del operador Laplaciano, es recomendable primero filtrar a la imagen con una Gaussiana y posterior a este proceso ejecutar el operador Laplaciano para detectar los bordes. Sin embargo, existe la posibilidad de realizar una combinación de los dos filtros para obtener un solo

operador que realice el suavizado y la detección de bordes en un solo procedimiento de convolución.

A través del cálculo de la segunda derivada de la función Gaussiana se consigue el operador Laplaciano del Gaussiano, mediante el cual se podrá conseguir los valores de los coeficientes de la máscara correspondiente a este operador para el tratamiento de una imagen digital.

A continuación se presenta la fórmula de este operador:

$$LoG = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \textbf{(2.37)}$$

El detector de bordes Laplaciano del Gaussiano, aprovecha la propiedad de suavizado del filtro Gaussiano y la característica de cruce por cero del operador Laplaciano para mejorar la detección de bordes.

Este método consiste en realizar tres pasos:

- **Filtrado.-** El filtro aplicado para suavizar la imagen es el Gaussiano.
- **Realce.-** El paso de realce es la segunda derivada.
- **Detección.-** El criterio de detección es la presencia de un cruce por cero en la segunda derivada con el correspondiente pico en la primera derivada.

Gracias al cambio que puede recibir el valor de la desviación estándar del filtro Gaussiano, es posible trabajar a diferentes escalas. De esta forma, cuando mayor es la desviación estándar existirá un menor número de pasos por cero.

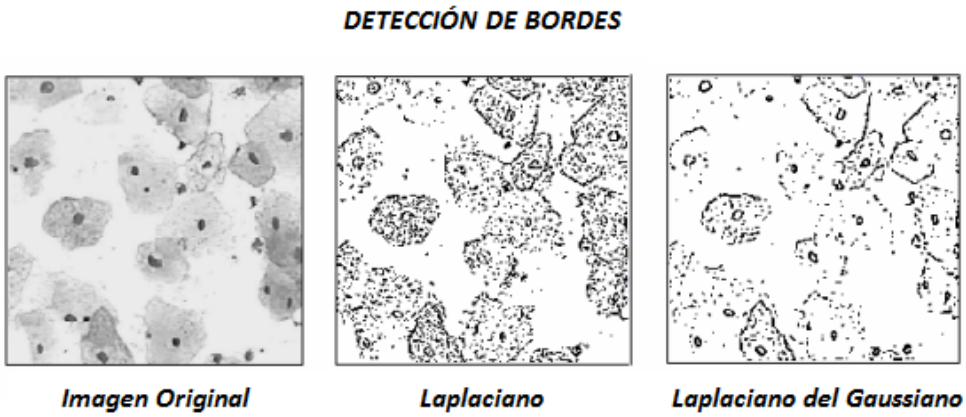


Figura 2.44: Detección de bordes mediante el Laplaciano y el Laplaciano del Gaussiano [47].

Se puede verificar la diferencia que existe entre la detección de bordes con el operador Laplaciano y el operador Laplaciano del Gaussiano. Con el segundo operador se reduce sustancialmente el ruido que está presente en la detección con el primer operador.

Se concluye que este operador es muy costoso computacionalmente, ya que requiere de mayores recursos para su ejecución y procesamiento.

CAPITULO 3

OTRAS TÉCNICAS DE DETECCIÓN

3. OTRAS TÉCNICAS DE DETECCIÓN

3.1. DIFERENCIA DE GAUSSIANAS

Es un algoritmo muy usado en el campo de la visión artificial para la detección de bordes, se basa en la diferencia de dos funciones Gaussianas y trabaja con imágenes en escala de grises.

La diferencia de Gaussianas aplica dos funciones Gaussianas[2] a la misma imagen pero su desviación estándar cambia para cada una, es recomendable que la segunda sea mayor que la primera para obtener mejores resultados[3]. Al aplicar la función Gaussiana se elimina el ruido de la imagen original; esto se logra aplicando la ecuación (3.1):

$$g_i(x,y) = \frac{1}{2\pi\sigma_i^2} exp^{-\frac{x^2+y^2}{2\pi\sigma_i^2}} \quad \textbf{(3.1)}$$

Donde x es la distancia desde el origen en eje horizontal, y es la distancia desde el origen en el eje vertical y sigma es la desviación estándar de la distribución de Gauss.

El resultado obtenido al aplicar la distribución de Gauss se utiliza para la convolución que se aplica a la imagen original. Es importante mencionar que la función Gaussiana se puede aplicar a una imagen de dos dimensiones, como dos cálculos independientes unidimensionales, por lo que se denominan linealmente separables. Las funciones Gaussianas son normalmente aplicadas cuando se reduce el tamaño de una imagen.

[2] Mayor detalle de la función Gaussiana se puede encontrar en el capítulo 2.3.

[3] Se recomienda que se cumpla la desigualdad $\sigma_1 < \sigma_2$ puesto que al incrementar el valor del σ_1 se obtienen bordes más gruesos y al reducir el valor del σ_2 se incrementa el umbral para reconocer los bordes, porque si se usa los variables con el mismo valor el resultado al aplicar la diferencia Gaussiana será una imagen en blanco.

Cuando existe una disminución de la resolución de la imagen es común aplicar un filtro paso bajo a la imagen antes del remuestreo. Se hace esto para asegurar que la información falsa de altas frecuencias no aparezca en la imagen.

Si se suaviza la imagen se suprime la información espacial de alta frecuencia y al restar una imagen de la otra se conserva la información espacial que se encuentra entre el rango de frecuencias de las dos imágenes restadas.

La diferencia de Gaussianas es muy utilizada para la detección de bordes, como habíamos citado anteriormente este algoritmo consta de dos funciones Gaussianas [12], su ecuación está definida por (3.2):

$$g(x,y) = g_1(x,y) - g_2(x,y) \qquad \textbf{(3.2)}$$

En la figura 3.1, podemos observar los resultados al aplicar el algoritmo DoG[4] en el cual la extracción de los bordes es más precisa, dejando de lado las partes irrelevantes de la imagen y enfocándose en las regiones de interés. En la figura 3.1 b, se observa la presencia de orillas dobles, es por esta razón que se aplica la eliminación de los mismos. En la figura 3.1 c, se puede observar la imagen a la que se a aplicado el filtro DoG junto con la suspensión de bordes.

[4] DoG: Diferencia de Gaussianas

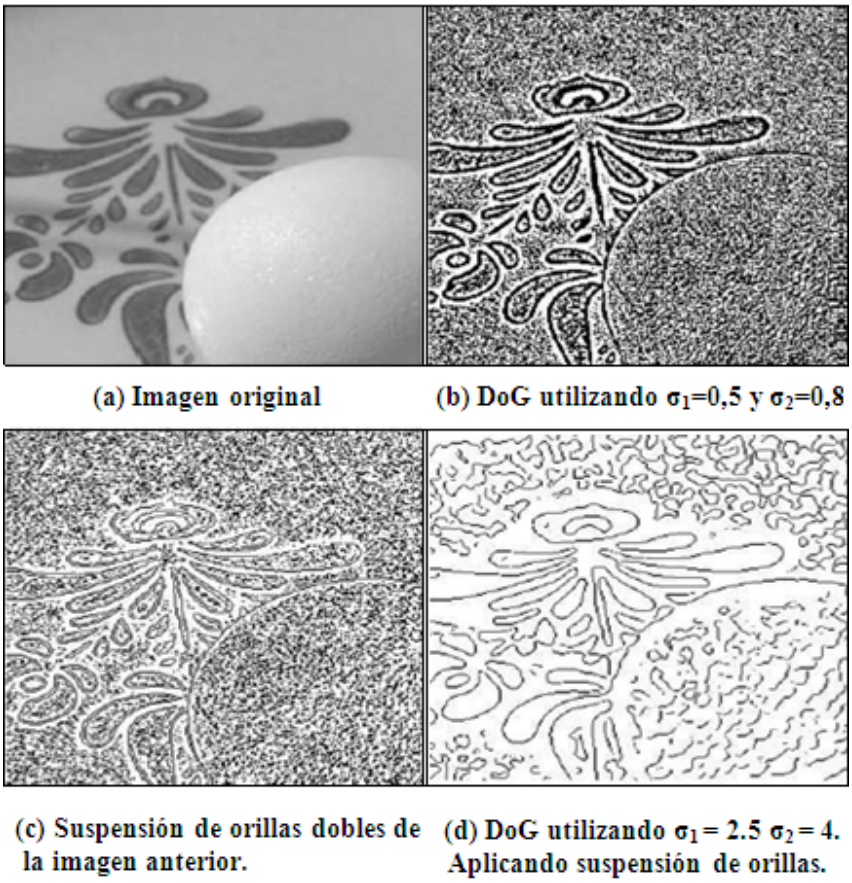


Figura 3.1: Diferencia de Gaussianas [24].

3.2. DETERMINANTE HESSIANO

El discriminante o matriz hessiana fue creada en el año 1844 por el matemático alemán Ludwig Otto Hess [13]. El objetivo de la matriz hessiana es analizar los puntos críticos, los valores máximos, mínimos y valores silla para realizar la extracción de las esquinas de una imagen. El tamaño de la matriz depende de las variables de la ecuación, dicha matriz está formada por los valores que se detallan a continuación [13]:

- f_{xx} Se deriva la función original por primera vez con respecto a x y luego ese resultado se deriva por segunda vez con respecto a x nuevamente.

- f_{xy} Significa que se deriva la función original por primera vez con respecto a x y luego ese resultado se deriva por segunda vez pero ahora con respecto a y.

- f_{xz} Significa que se deriva la función original por primera vez con respecto a x y luego ese resultado se deriva por segunda vez pero ahora con respecto a z.

- f_{yz} Significa que se deriva la función original por primera vez con respecto a y, luego ese resultado se deriva por segunda vez pero ahora con respecto a x.

- f_{yy} Significa que se deriva la función original por primera vez con respecto a y, luego ese resultado se deriva por segunda vez con respecto a y nuevamente.

- f_{yz} Significa que se deriva la función original por primera vez con respecto a y, luego ese resultado se deriva por segunda vez pero ahora con respecto a z.

- $f_{z\,x}$ Significa que se deriva la función original por primera vez con respecto a z y luego ese resultado se deriva por segunda vez pero ahora con respecto a x.

- f_{zy} Significa que se deriva la función original por primera vez con respecto a z y luego ese resultado se deriva por segunda vez pero ahora con respecto a y.

- f_{zz} Significa que se deriva la función original por primera vez con respecto a z y luego ese resultado se deriva por segunda vez con respecto a z nuevamente.

Con la descripción de cada uno de los valores, la matriz hessiana quedaría formada como lo indica la ecuación (3.3):

$$\begin{bmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} \end{bmatrix} \quad \textbf{(3.3)}$$

El análisis de la matriz hessiana se lo hace encontrando los determinantes, valores que están dados por:

$$\Delta < 0 \; Míninos$$
$$\Delta > 0 \; Máximo$$
$$\Delta <> 0 \; Silla$$

El procedimiento que se realiza para determinar los valores de la matriz consiste en resolver el primer determinante con respecto a x, luego respeto a y dependiendo de las variables que posea la ecuación se encontrarán los resultados de la matriz, el resultado se igualará a cero para poder determinar los puntos críticos. De igual forma como se obtuvieron los valores del primer determinante se obtendrán los valores del segundo determinante y con los resultados se procede a llenar la matriz hessiana.

Los determinantes y puntos críticos que se obtengan dependerán de las variables de cada ecuación. Con los valores obtenidos se procede a llenar la matriz para determinar los puntos máximos y mínimos, este proceso se lo realiza con una multiplicación en x y el resultado se resta de derecha a izquierda.

El determinante Hessiano se calcula aplicando la siguiente ecuación (3.4):

$$detHL(x, y; t) = t^2 \left(f_{xx} f_{yy} - f_{xy}^2\right) \quad \textbf{(3.4)}$$

Donde t es la escala y HL es la matriz hessiana de f de la cual se obtiene el detector diferencial de regiones**:**

$$(\hat{x}, \hat{y}; \hat{t}) = argmaxlocal_{(x,y;t)}\big(detHL(x, y; t)\big) \quad \textbf{(3.5)}$$

Donde argmaxlocal es el valor máximo de la función en ese punto de análisis. Y los valores que se obtienen de $(\hat{x}, \hat{y})$ y escala $\hat{t}$ son valores que se extraen de la operación geométrica diferencial que nos permiten obtener descriptores de regiones invariante al traslado, rotado y reescalado de la imagen.

3.3. MAXIMALLY STABLE EXTREMAL REGIONS

El algoritmo de MSER[5] fue propuesto por J. Matas en el 2002 y tiene como objetivo encontrar correspondencia entre las imágenes [11].

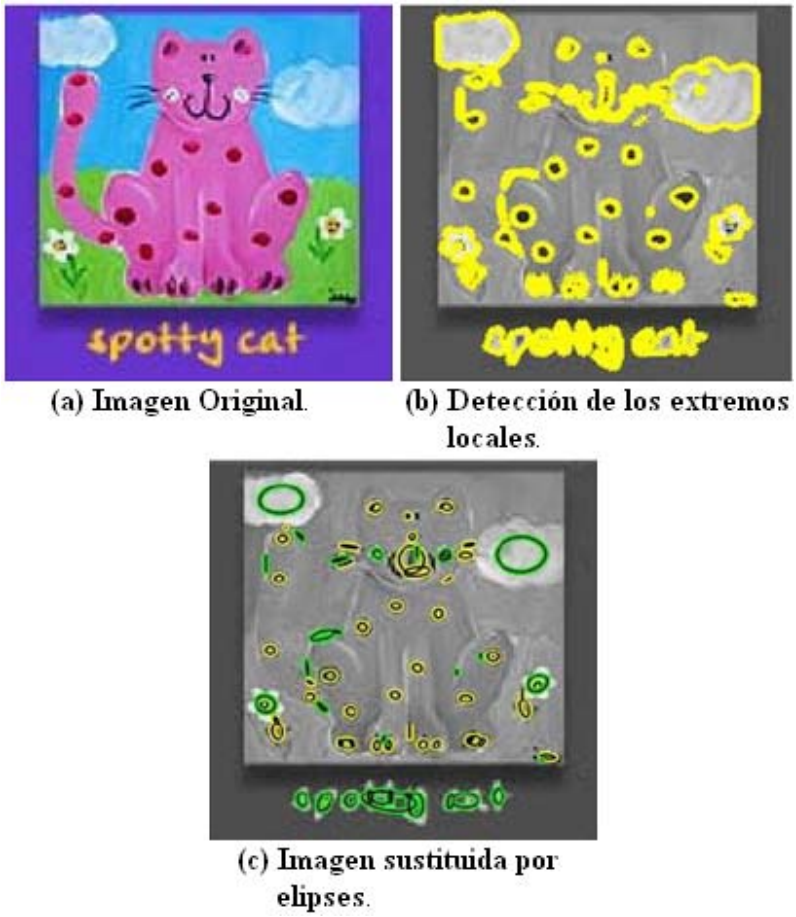
Su nombre "Máxima estabilidad de regiones extremas" se basa dos aspectos: las regiones extremas que son definidas por un conjunto de elementos de una imagen que poseen correspondencia y la máxima estabilidad permite analizar sin inconvenientes imágenes con variaciones en la iluminación, escalabilidad y posición. Por tanto el MSER es un algoritmo de gran utilidad para el análisis de pares de imágenes.

Este algoritmo trabaja con otros algoritmos de procesamiento de imágenes como SIFT que le permite el reconocimiento de imágenes que posean correlación.

El algoritmo MSER permite encontrar la similitud entre dos imágenes con características similares. Su funcionamiento es por medio de los umbrales, extrae regiones de pixeles que posee un nivel similar de grises, es decir, los pixeles que forman este conjunto de regiones están por debajo de los valores del umbral

[5] MSER: MAXIMALLY STABLE EXTREMAL REGIONS (Máxima estabilidad de Regiones Extremas).

utilizado. Como paso posterior las regiones se sustituyen por elipses, para luego aplicar otros algoritmos como SURF o SIFT. A continuación se procede a calcular los descriptores de las elipses detectadas para comparar ambas imágenes [11].



(a) Imagen Original. (b) Detección de los extremos locales. (c) Imagen sustituida por elipses.

Figura 3.2: Ejemplo de aplicar el algoritmo MSER a una imagen [11].

A continuación se explica el proceso del algoritmo para la extracción de blob como explica [11].

- El funcionamiento básico del algoritmo MSER es agrupar los pixeles por su nivel de intensidad, para posteriormente formar un árbol de valores que deberá poseer ciertas características:

- Todos los niveles de intensidad que sean iguales a un determinado pixel del árbol de intensidad serán un subconjunto que forman una región.

- Las regiones que se encuentren son descendientes de los pixeles del árbol de intensidad.

- La extracción de las regiones es de acuerdo a los valores del árbol de intensidad y los umbrales definidos.

- Si existen regiones duplicadas u otras regiones definidas incorrectamente serán eliminadas.

- Se procede a calcular las elipses que poseen las regiones obtenidas.

Con la explicación expuesta sobre el procesamiento de la imagen del algoritmo MSER se tiene una idea general del cómo trabaja y analiza las imágenes para extraer los blob comparando dos imágenes.

En la figura 3.3 se puede observar dos imágenes que poseen correspondencia.

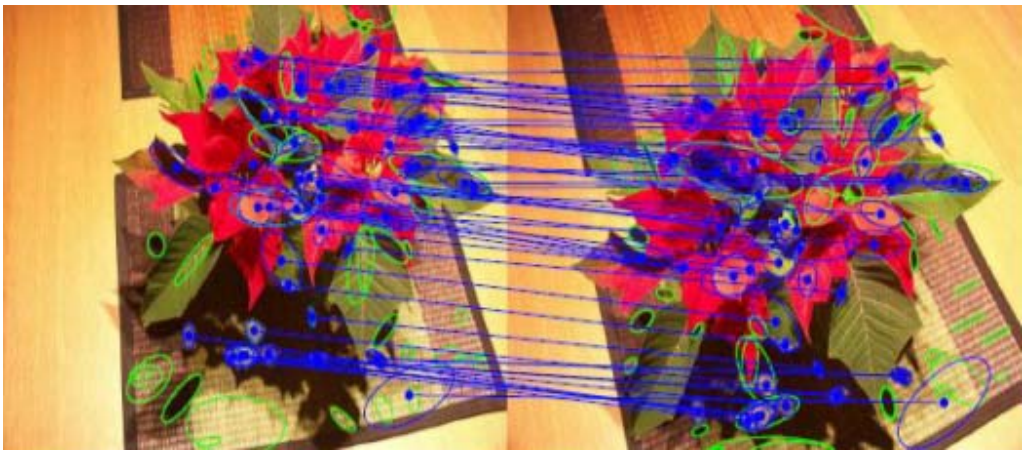


Figura 3.3: Imágenes a las que se aplican el algoritmo MSER [11].

3.4. GREY – LEVEL BLOBS

Surge con el propósito de solucionar el problema de sensibilidad al ruido que presenta los métodos basados en extremos locales. El algoritmo *Grey-level blobs* fue creado por Tony Lindeberg en el año de 1993-1994 con el objetivo de analizar los máximos locales a diferentes escalas [22].

Este algoritmo está desarrollado en base a la clasificación previa de pixeles, es decir, realiza un análisis a las regiones que poseen similar intensidad de color.

Por tanto *Grey-level blob* analiza los pixeles vecinos, usualmente se le conoce con la notación "Vecino mayor" que quiere decir "el pixel vecino que posee un valor alto de nivel de gris". A continuación se detalla la clasificación en la cual basa su análisis:

- Si por lo menos posee un vecino mayor, este puede ser el fondo, por lo tanto no puede pertenecer a ningún blob.
- En caso contrario, si posee más de un vecino mayor y si los vecinos mayores son las partes de diferentes blobs entonces no puede ser parte de un blob, por lo que deben ser pixeles que pertenecen al fondo.
- En caso contrario si tiene uno o más vecinos con valores mayores de gris que formen parte de un mismo blob, entonces pertenecerá al blob.

En la figura 3.4 se muestra un ejemplo aplicando *Grey-Level blobs* a diferentes escalas $t = 0,1,2,4,8,16,32,64,128,256,512\ y\ 1024$ respectivamente. Se puede ver una escala fina principalmente en objetos pequeños debido al ruido y la textura de la superficie detectada [21].

Cuando el parámetro de escala es aumentado, el ruido que presentan los objetos desaparece poco a poco, aunque pueden variar de acuerdo a las regiones que posean cambios representativos de su textura.

Cuando poseemos escalas de mayor tamaño los bloques que se encuentran adyacentes se agrupan formando objetos más grandes.

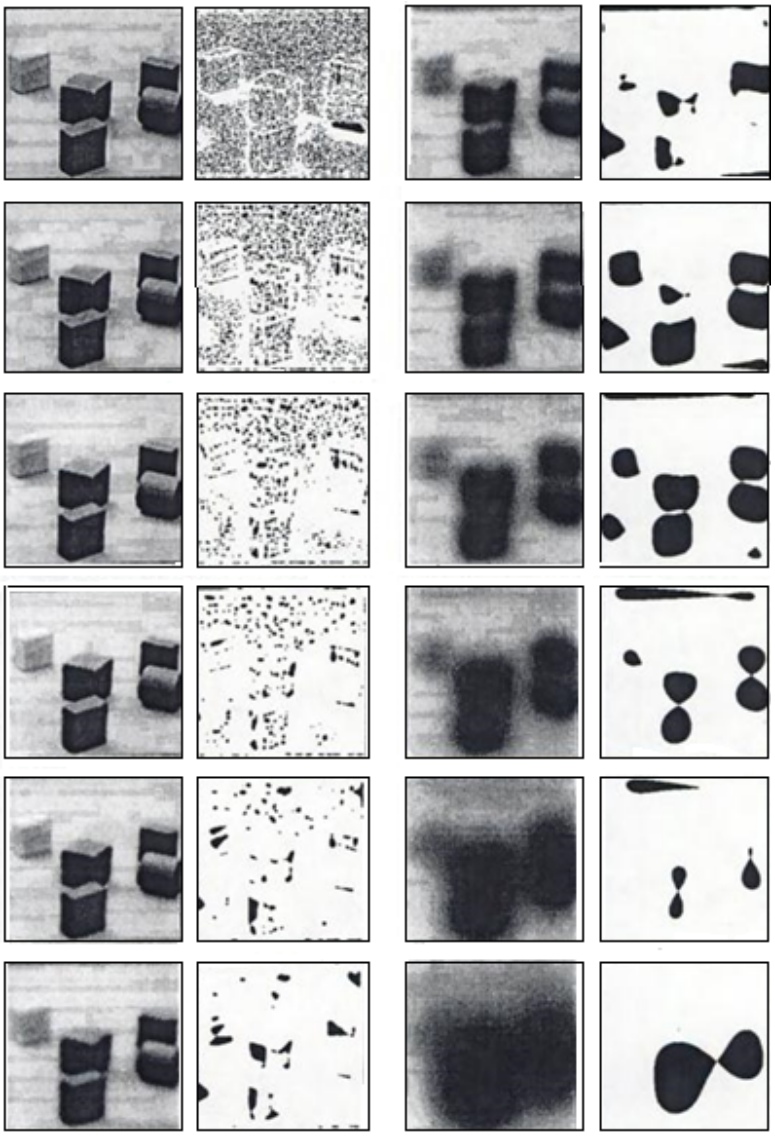


Figura 3.4: Imágenes aplicando el algoritmo Grey level blob a diferente escala [21].

En otro ejemplo se puede observar una similar secuencia de imágenes aplicando *Grey-level blob* al teléfono y la calculadora. También en este caso se obtienen principalmente pequeños objetos debido al ruido y textura de la superficie [21].

Por otra parte, un comportamiento jerárquico entre las manchas a diferentes escalas aparece de nuevo. Los botones en el teclado se manifiestan como manchas después de aplicar un ligero suavizado. Cuando el nivel de escala es mayor se fusionan en una sola unidad (el teclado). También se puede observar que algunos detalles oscuros en la imagen: la calculadora, el cable y el receptor se ven como manchas al aplicar un nivel de escala mayor [21].

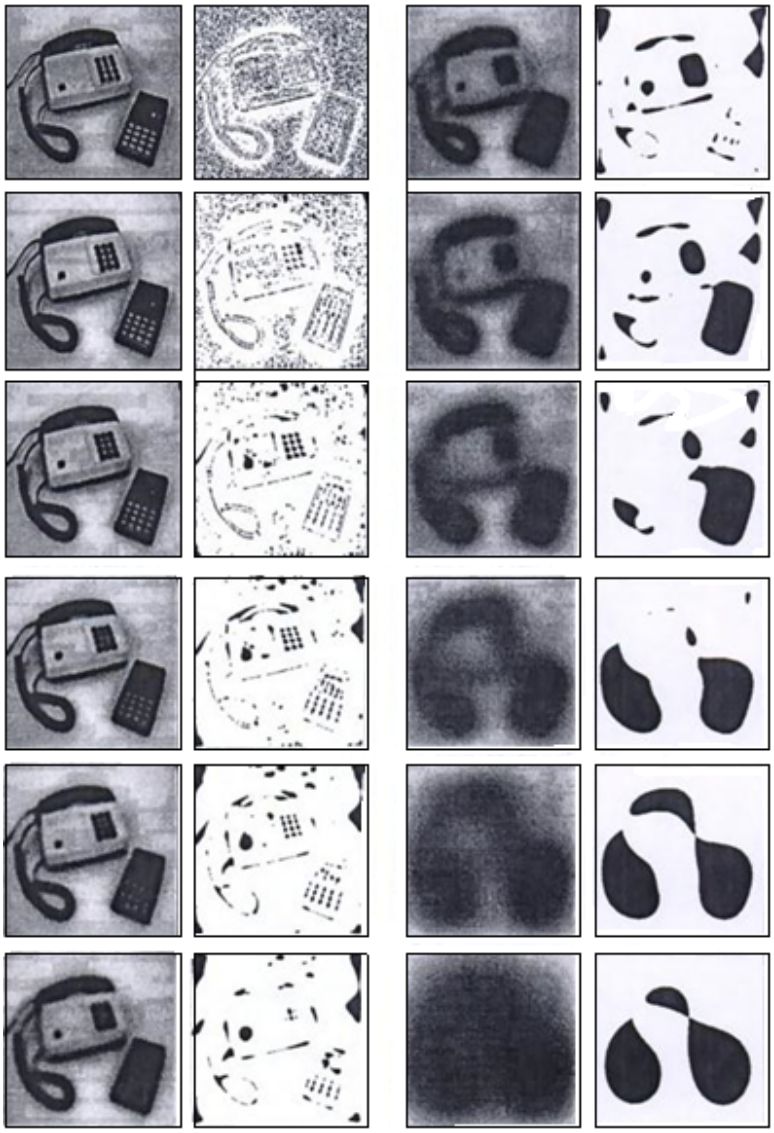


Figura 3.5: Imágenes aplicando Grey Level Blob a diferente nivel de escala *t=0,1,2,4,8,16,32,64,128,256,512, y 1024* respectivamente [21].

3.5. ESQUINA DE HARRIS

El objetivo del detector de esquinas de Harris es la extracción de los puntos más relevantes de una imagen, es muy utilizado por su alta invariancia frente a la escala, la rotación, los cambios de iluminación y el ruido en la imagen [16].

El detector de Esquinas de Harris basa su análisis en la matriz de correlación la cual se calcula aplicando derivadas de primer orden a cada pixel, como se puede apreciar en la ecuación 3.6.

$$C = \begin{bmatrix} \left(\frac{dI}{dx}\right)^2 & \frac{dI}{dx}\frac{dI}{dy} \\ \frac{dI}{dx}\frac{dI}{dy} & \left(\frac{dI}{dy}\right)^2 \end{bmatrix} \quad \textbf{(3.6)}$$

En donde *I(x,y)* es la intensidad del nivel de grises.

Posteriormente se realiza el cálculo del gradiente para ello se utilizan máscaras de convolución 3x3. El detector de Harris realiza el suavizado de la matriz correlación aplicando un filtro Gaussiano.

Al analizar los autovalores de la matriz se define si un pixel es una esquina, un borde o ninguno de los dos. Si la matriz C es simétrica y semidefinida positiva[6], sus dos autovalores $\left(\left(\frac{dI}{dx}\right)^2 y \left(\frac{dI}{dy}\right)^2\right)$ son positivos.

Al evaluar los pixeles de la imagen se deben tener en cuenta ciertos criterios para poder determinar si el pixel pertenece a una esquina, a un borde o a una región plana [39].

- Si los dos autovalores de la matriz C son grandes entonces estamos ante a una esquina.
- Caso contrario, si uno de los autovalores es grande y el otro pequeño el pixel será parte de un borde

[6] Para que la matriz sea simétrica y semidefinida debe cumplir que: la matriz original sea igual a su transpuesta y los valores de la diagonal principal deben ser positivos.

- Finalmente, si los dos autovalores de la matriz C son pequeños entonces el pixel formará parte de una región plana.

El parámetro de Harris esta dado por:

$$R = \det(C) - k(traza(C))^2 \quad \textbf{(3.7)}$$

En donde det(C) es el determinante de la matriz C, k es el parámetro de sensibilidad ajustable y se establece con el valor de 0,04 (sugerido por Harris). La función traza(C) calcula la suma de los elementos de la diagonal principal de la matriz C [22].

3.6. ESCALA-ESPACIO

Según Lindeberg (1993, 1994) la representación Escala-Espacio es un modelo de análisis de multi-escala que posee un parámetro continuo de escala y conserva el mismo muestreo espacial para todas las escalas. Otra de sus características es introducir la señal original en una familia de señales de un solo parámetro generada por la convolución con señales Gaussianas de sigma creciente. A medida que va creciendo el valor de sigma la imagen se ve más borrosa [18] [23].

Las propiedades deseables para el procesamiento multi-escala son [23]:

- Invarianza a los desplazamientos: Isotropía[7] espacial, es decir, todas las posiciones espaciales son tratadas en forma equitativa.
- Invarianza a la escala: Homogeneidad espacial, todas las escalas espaciales son tratadas por igual.

[7] Isotropía: viene del griego *isos* que quiere decir equitativo igual, y *tropos* que es medio espacio lugar dirección.

- Causalidad:

    - No deben crearse nuevas curvas de nivel en los espacios-escala.
    - No deben crearse nuevos extremos locales.
    - No deben acentuarse los extremos locales, es decir, ningún extremo en una cierta escala se vuelve mayor en las escalas superior e inferior.

La función de Escala-Espacio está definida por la convolución de la imagen inicial *I(x,y)* con el *Kernel* Gaussiano (ecuación 3. 10) [21]:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad \textbf{(3.8)}$$

Al aplicar la representación Escala-Espacio se agrega una dimensión a la imagen como se puede observar en la figura 3.6.

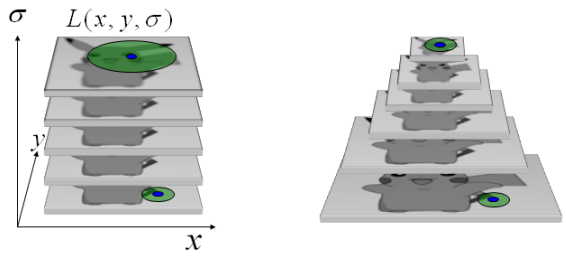


Figura 3.6: Imagen usando la representación Escala-Espacio [18].

3.7. OTRAS TÉCNICAS DE DETECCIÓN

En el proceso de la investigación se han podido estudiar varias técnicas que son utilizadas para el procesamiento de imágenes, entre ellas mencionaremos las siguientes:

- Algoritmo de Canny
- Algoritmo de SUSAN

3.7.1. ALGORITMO DE CANNY

El algoritmo Canny fue creado en el año de 1986 por John Canny, su objetivo es la detección de contornos mediante el empleo de máscaras de convolución. Este filtro detecta una ruptura f que suaviza el ruido localizando los contornos de la imagen [17].

El Algoritmo Canny está basado en tres criterios los cuales se detallan a continuación tal como lo indica [17]:

- Criterio de detección: evita la eliminación de bordes importantes, para no presentar bordes falsos.
- El criterio de localización: permite establecer una distancia entre la posición real y la del borde localizado para minimizarla.
- El criterio de una respuesta: permite integrar las múltiples respuestas correspondientes a un solo borde.

3.7.1.1. PROCESO PARA LA DETECCIÓN DE BORDES

El detector de bordes Canny está basado en la primera derivada, y su proceso consiste en tres pasos:

- Obtención del gradiente (magnitud y orientación del vector gradiente en cada píxel).
- Supresión no máxima o adelgazamiento del ancho de los bordes obtenidos con el gradiente hasta lograr bordes de un píxel de ancho.

- Histéresis[8] de umbral al resultado de la supresión no máxima, con este proceso se pretenden reducir la posibilidad de aparición de falsos contornos.

Obtención del gradiente.-

Para la obtención de la gradiente primero debemos aplicar un filtro Gaussiano, el cual suaviza la imagen eliminando el ruido que afecta a la imagen. El suavizado que se aplique debe ser el apropiado para no perder características importantes de la imagen que pueden dificultar un correcto análisis en el apartado 2.3 se explica a detalle sobre el filtro Gaussiano.

Algoritmo: Obtención del gradiente.-

El proceso que se usa para la obtención del gradiente se encuentra detallado en el apartado 2.2.3, allí podemos observar fórmulas que aplica para este proceso.

Supresión no máxima o adelgazamiento del ancho de los bordes.-

Del paso anterior se obtienen dos imágenes que servirán para generar una imagen con los bordes adelgazados, para lo cual se considerarán cuatro direcciones identificadas por las orientaciones de 0º, 45º, 90º y 135º con respecto al eje horizontal. A cada pixel se le aplica una dirección que mejor se aproxime al ángulo del gradiente.

[8] Histéresis: Es la tendencia a conservar las propiedades originales de la imagen. Como bordes, textura, color, sombras, forma de los objetos y profundidad.

Algoritmo: Suspensión no máxima [19].-

De entrada tenemos una imagen E_m de la magnitud del gradiente y E_o imagen de la orientación del gradiente. Como salida obtendremos la imagen I_n.

Es importante tener en cuenta cuatro direcciones *1d, 2d, 3d y 4d* identificadas por las direcciones de 0º, 45º, 90º y 135º con respecto al eje horizontal.

Para cada píxel *(i, j)***:**

- Se debe localizar la dirección dk que mejor se aproxima a la dirección $E_o\ (i, j)$, que viene a ser la perpendicular al borde.
- Si $E_m\ (i, j)$ es más pequeño que al menos uno de sus dos vecinos en la dirección dk, al píxel *(i, j)* de In se le asigna el valor 0, $I_n\ (i , j) = 0$ (supresión), de otro modo $I_n\ (i, j) = E_m\ (i, j).$
- Devolver I_n.

Histéresis de umbral a la supresión no máxima.-

La imagen que se obtuvo del proceso anterior suele tener máximos locales creados por el ruido y para eliminar el ruido generado se debe aplicar la histéresis del umbral.

Para su proceso se obtendrá la orientación de los puntos de borde de la imagen y se procede a tomar dos umbrales; teniendo en consideración que el primero debe ser más pequeño que el segundo.

A cada pixel de la imagen se debe localizar un siguiente pixel del borde el cual debe ser mayor al segundo umbral a partir de este pixel se continua con la cadena de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde, siempre que sean mayores al primer umbral. Continuando con el mismo proceso de marcar los pixel explorados, se almacena una lista de todos los pixeles que se encuentran conectados al contorno de modo que se logran eliminar las uniones en forma de Y de los segmentos que coincidan en un pixel; el algoritmo que permite calcular la histéresis de umbral a la suspensión no máxima se detalla a continuación.

Algoritmo: Histéresis de umbral a la supresión no máxima [19].-

Como resultado de los pasos anteriormente descritos se obtiene una imagen *G* con los bordes conectados de contornos.

Para todos los puntos de I_n y explorando I_n en orden fijo:

- Localizar el siguiente punto de borde no explorado previamente, $I_n\,(i, j)$, tal que $I_n(i, j) > t_2$
- Comenzar a partir de $I_n\,(i, j)$, seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde, siempre que $I_n > t_1$.
- Marcar todos los puntos explorados y, salvar la lista de todos los puntos en el entorno conectado encontrado.

Devolver la imagen *G* formada por el conjunto de bordes conectados de contornos de la imagen, así como la magnitud y orientación, describiendo las propiedades de los puntos de borde.

Resultados.-

A continuación se indica los resultados de aplicar el detector de bordes Canny:

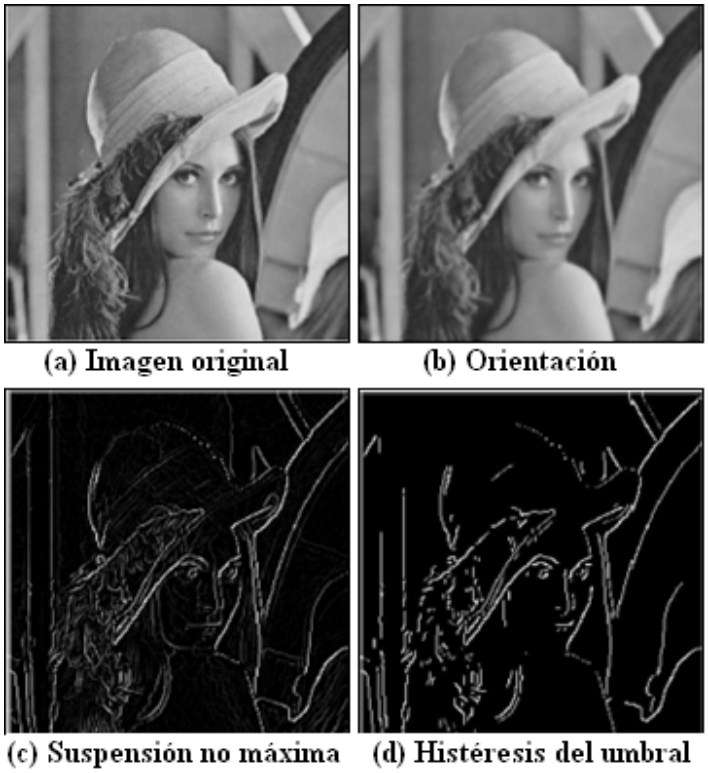


Figura 3.7: Resultados de aplicar el detector de bordes Canny [19].

3.7.2. ALGORITMO DE SUSAN

El algoritmo de SUSAN, que de acuerdo a sus siglas en inglés significa "Smallest Univalue Segment Assimilating Nucleus" (Segmento más pequeño del núcleo de asimilación), es una técnica publicada por Stephen M. Smith y J. Michael Brady en 1997 y su objetivo es detectar las esquinas y bordes principales de los objetos.

El área de SUSAN es de mucha utilidad para detectar la presencia de bordes en figuras de dos dimensiones.

Algo muy importante de este algoritmo es que no utiliza derivadas para detectar bordes, por tanto no es muy susceptible al ruido, de igual forma analiza diferentes regiones por separado para detectar las fronteras en base los cambios de valores mínimos de SUSAN.

Este método utiliza una máscara circular, la cual está formada por un centro llamado núcleo y un radio que puede ser variable. La máscara es sobrepuesta en las zonas de interés de la imagen, si la intensidad de los pixeles que estén debajo de la máscara tienen el mismo valor que el núcleo de la máscara dichos pixeles forman el área llamada SUSAN.

Los valores de los pixeles que se encuentren por debajo de la máscara son los utilizados para determinar las esquinas principales y bordes, por lo tanto generan tres posibles resultados que son explicados a continuación [20]:

1. Se denomina un valor máximo cuando el núcleo está en una región de la imagen que posee el mismo brillo.

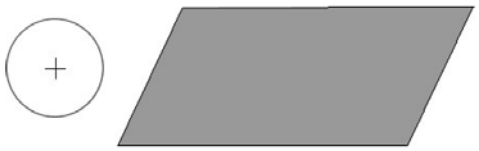


Figura 3.8: Valor máximo de un punto de interés [20].

2. Un valor medio cuando el núcleo está muy cerca de un borde.

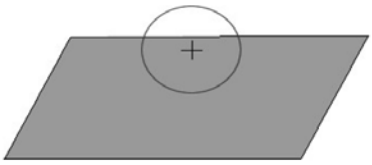


Figura 3.9: Valor medio de un punto de interés [20].

3. Un valor mínimo se encuentra cuando el núcleo está sobre una esquina.

Figura 3.10: Valor mínimo del punto de interés [20].

El procedimiento a seguir para la detección de las esquinas principales es el siguiente:

Para proceder a calcular la máscara circular que va a posicionarse sobre los puntos de interés existen varias formas de hacerlo, sin embargo el método más apropiado es aplicando la función de distribución Gaussiana que es:

$$g[i,j] = K * e^{\frac{i^2+j^2}{2\sigma^2}} \quad \textbf{(3.9)}$$

En donde:

- K es la constante de normalización.
- σ es el parámetro de dispersión.
- $i\ y\ j$ son dos dimensiones de la función.

Como siguiente paso se calcula la similitud de los valores con respecto al núcleo y la máscara, esto se lo realiza con la siguiente función:

$$C(r, r_0) = exp - \left[\frac{I(r) - I(r_0)}{t}\right]^6 \quad \textbf{(3.10)}$$

En donde:

- r_0 es la posición del núcleo
- r es la posición cualquier punto bajo la máscara.
- $I(r)$ es el brillo del pixel.
- t es el contraste mínimo entre detalles que van a detectarse y el ruido a eliminar.

Ya obtenidos los valores bajo la máscara se suman y el resultado se asigna a la posición del pixel central, el cual fue aproximado por la ecuación 3.12:

$$n(r_0) = \sum_r C(r, r_0) \quad \textbf{(3.11)}$$

Utilizando la ecuación:

$$R(r_0) = \begin{cases} g - n(r_0) \Leftrightarrow \ \ n(r_0) < g \\ 0 \Leftrightarrow n(r_0) \geq g \end{cases} \quad \textbf{(3.12)}$$

Donde: *g* es el umbral establecido. Y el valor de g se calcula de a través de la relación*:* $g = n_{max}/2$, donde n_{max} es el número máximo de los valores obtenidos de la máscara.

Como paso final se buscan todos los puntos falsos dentro del conjunto de esquinas definidas, esto se produce por la existencia de esquinas borrosas. Esto puede ser solucionado por medio del cálculo del centro de gravedad del área de

USAN, la mediana de las coordenadas de x e y, permiten calcular la distancia entre el centro de gravedad y el núcleo de la máscara, si la distancia calculada es grande nos encontramos frente a esquinas falsas [20].

3.8. COMPARATIVAS DE LAS TÉCNICAS DE DETECCIÓN.

Después del análisis de cada una de las técnicas de extracción de blob se realiza una evaluación de los mismos para una visión más clara sobre su precisión para segmentar imágenes. Se procede de esta forma, puesto que cada técnica posee su procedimiento propio para el análisis de las imágenes.

El análisis comparativo de las técnicas se basará en un análisis teórico, y no en un análisis funcional puesto que las técnicas han sido analizadas teóricamente y no se ha podido observar el tiempo de respuesta de cada algoritmo.

Al aplicar la técnica DoG en una imagen la detección de regiones de interés es más amplia, es muy importante mencionar que este algoritmo posee mayor precisión por que usa dos filtros Laplaciano y su resultado es la resta de los mismos, por tanto el porcentaje de puntos detectados es mayor. Esto se constata en los estudios realizados en [23]: su porcentaje de detección de puntos de interés es de 38%.

El método MSER compara dos imágenes para encontrar la correspondencia entre ellas, hace un reconocimiento 3D, su extracción se basa en varios algoritmos como SIFT y SURF, los cuales colaboran para la exactitud en la extracción de blob para posteriormente realizar la comparación entre dos imágenes y encontrar la correspondencia entre ella; su porcentaje de extracción de detección de puntos de interés es de 60%.

Grey-Level Blob su análisis está enfocado en los extremos locales, es importante citar que su precisión en la detección de los objetos de interés depende de la escala a la cual está siendo analizada la imagen. Si el valor *t* es menor, la extracción de

objetos de interés será mayor; su porcentaje de extracción de objetos de interés es de 35%.

El algoritmo Esquinas de Harrris es muy utilizado puesto que al analizar una imagen sus cambios de: rotación, iluminación y ruido de la imagen son controlados por este algoritmo, por lo tanto al extraer blobs de una imagen su porcentaje de acierto es de 16% [23].

El algoritmo Canny es un método muy utilizado para detectar bordes existentes de una imagen, este algoritmo es considerado como uno de los mejores métodos de detección de contornos mediante el empleo de máscaras de convolución y la primera derivada. Canny está basado en tres criterios que han sido detallados en capitulo 3.7.1 Según la investigación realizada, Canny posee un porcentaje de extracción de 40% lo cual nos da una visión de su precisión al detectar bordes de una imagen.

El algoritmo de SUSAN tiene un porcentaje de extracción de bordes y esquinas de un 30%. Es importante destacar que este algoritmo busca introducir un aspecto clave como es la rapidez.

Nota: Los resultados de la comparación realizada con los métodos que son usados para el tratamiento de imágenes son valorados por los desarrolladores de la tesis basados en imágenes obtenidas de investigaciones realizadas por otros autores. Estos algoritmos son: MSER, Grey-Level, Canny, SUSAN.

CAPITULO 4

SELECCIÓN DE HERRAMIENTAS

4. SELECCIÓN DE HERRAMIENTAS

4.1. INTRODUCCIÓN

En la actualidad existen varias herramientas orientadas al procesamiento de imágenes, muchas de estas analizan desde imágenes sencillas hasta imágenes con un grado muy elevado de complejidad.

Las herramientas realizan procesos específicos o combinatorios como: suavizado, detección de bordes, detección de puntos de interés, segmentación de imágenes, etc.

Las herramientas deben ser seleccionadas de acuerdo al criterio del tratamiento de imágenes que se quiera aplicar, es decir, considerando los operadores y la complejidad de la imagen a ser tratada. La consideración particular de los operadores se debe a que existen muchos de estos que realizan la misma función, sin embargo difieren en la exactitud de los resultados.

En este caso se busca una herramienta que utilice el *Kernel* Gaussiano y el operador Laplaciano, tanto para el suavizado como para la detección de bordes, respectivamente. Es importante mencionar que el *Kernel* Gaussiano y el operador Laplaciano se efectúan previamente a la realización de la segmentación. Una vez que se ha priorizado en estas dos operaciones, se procederá a considerar otros factores clave para la selección de la herramienta con la que se trabajará a lo largo de los siguientes capítulos.

Por consiguiente en este punto se realizará una breve descripción de cada herramienta con sus prestaciones, para posteriormente efectuar una comparación y selección.

4.2. DETECTOR SIFT9 KEYPOINT

SIFT Keypoint está desarrollado en el lenguaje C++ y funciona bajo las plataformas GNU/Linux y Microsoft Windows, y es capaz de trabajar con imágenes en formato PGM[10].

El objetivo de la herramienta es la extracción de puntos o características de interés invariantes a la rotación, traslación, escala y cambios de iluminación, para luego poder realizar la comparación con otras imágenes. Hace uso del algoritmo SIFT (Transformada de características invariable en escala) el mismo que detecta los puntos característicos estables de un objeto en una determinada posición. La herramienta al usar el algoritmo SIFT detecta los puntos de interés pero posee también la característica de identificar un objeto cuando forma parte de otra imagen, como se puede observar en la figura 4.3.

La figura 4.1 servirá para determinar los puntos de interés para una posterior identificación de la imagen. En la figura 4.2 se puede observar un conjunto de objetos, entre ellos la figura 4.1 de la cual se obtuvo los puntos de interés. En base a estas dos imágenes la herramienta procederá a la identificación del objeto que posee los puntos referencia que previamente se obtuvo de la figura 4.1.

Figura 4.1: Imagen Original utilizada para determinar los puntos de interés. [31]

[9] SIFT son las siglas de Scale Invariant Feature Transform (Transformada de características invariable en Escala)
[10] PGM: Portable graymap.

Figura 4.2: Imagen utilizada para la comprobación de la efectividad de SIFT [31]

La herramienta empleada (ver figura 4.2) permite extraer las características o puntos de interés de las imágenes y posterior a la extracción se dibujan las líneas entre las características que poseen los puntos más cercanos, logrando definir e identificar la imagen.

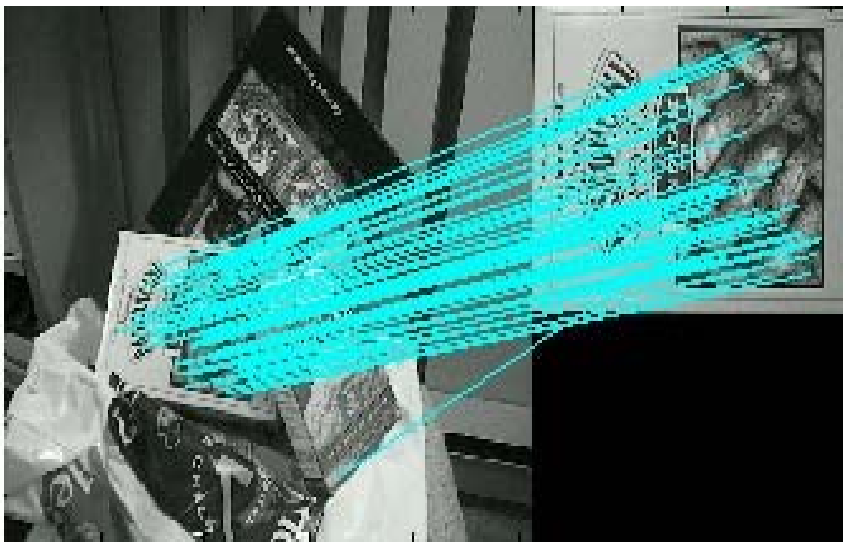


Figura 4.3: Resultados que presenta la Herramienta SIFT Keypoint [31].

Es muy importante tener ya establecidos los puntos característicos de las imágenes puesto que es el proceso previo que realiza la herramienta para el análisis de las imágenes.

4.3. MICROSOFT PHOTOSYNTH

Esta herramienta funciona bajo la plataforma Microsoft Windows y por ser una aplicación compilada no se puede tener acceso al código.

La herramienta Microsoft Photosynth posee un poderoso conjunto de utilitarios que permiten capturar y ver las imágenes en 3D. Asimismo crea escenarios 3D a partir de imágenes 2D.

Para el procesamiento de imágenes en Microsoft Photosynth se realiza un análisis de varias fotografías tomadas en una misma zona. A cada una de las fotografías se le aplica la detección de los puntos de interés, este proceso se realiza con el algoritmo desarrollado por Microsoft Research, y que trabaja de manera similar a SIFT. Con este proceso reconocen las características específicas de las imágenes.

Esta herramienta permite identificar objetos de una imagen que forman parte de otras imágenes o escenas. De igual forma la herramienta permite realizar un reconocimiento de acuerdo a la posición de la fotografía, es decir, puede identificar si una fotografía es parte de otras. Otro aspecto importante, es que Photosynth también proporciona un reconocimiento de la posición en 3D de cada característica, así como la posición y ángulo en el que se obtuvo la fotografía (figura 4.4).

Figura 4.3: Imagen resultante de aplicar Microsoft Photosynth.

4.4. OTRAS HERRAMIENTAS

En el proceso de investigación se han podido analizar varias herramientas que permiten realizar el tratamiento de imágenes, entre las que permiten efectuar este proceso consideramos las siguientes:

- Blobdemo
- OpenCV 2.3.1
- Segment

4.4.1. BLOBDEMO

Está basada en el lenguaje de programación Java, lo cual facilita su uso en todas plataformas que soporten una máquina virtual de Java. Trabaja con imágenes en formato PNG[11].

Blobdemo segmenta cada uno los objetos pertenecientes a una imagen, en este proceso no define ningún operador para realizar la segmentación. En tal virtud, la segmentación es realizada mediante comparaciones entre colores, tomando en consideración las variaciones que sean considerables.

[11] PNG: Portable Network Graphics

Por consiguiente Blobdemo trabaja de manera eficiente con imágenes donde no se tiene texturas y las áreas de las imágenes están separadas por colores. Un ejemplo de los resultados que genera esta herramienta se puede apreciar en la figura 4.5.

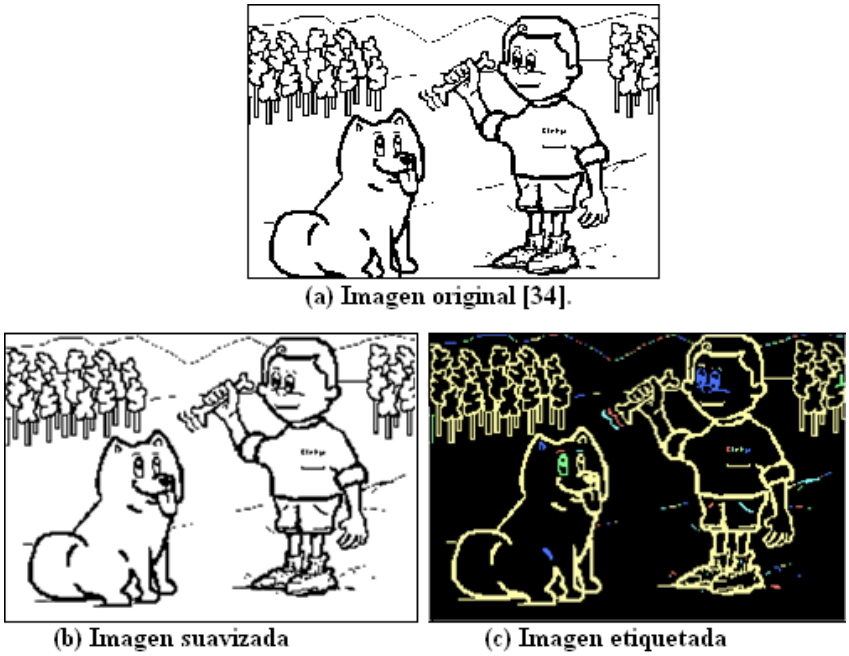


Figura 4.4: Resultado de la segmentación

En imágenes más complejas, es decir, con texturas y zonas solapas, la segmentación no es óptima y se pierden ciertas características e incluso objetos de la imagen, esto es debido a que antes del proceso de segmentación no aplica ningún algoritmo que detecte bordes para definir cada área de los objetos que posea la imagen. Cada objeto identificado es etiquetado con diferentes colores como se puede observar en la figura 4.6.

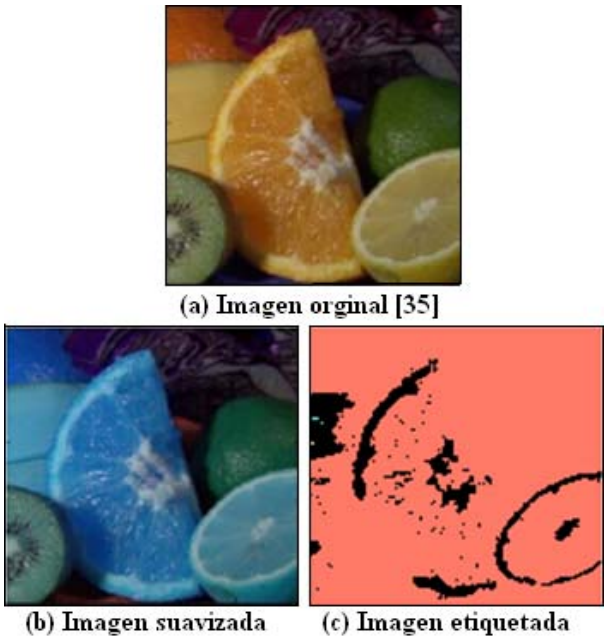


Figura 4.5: Imágenes Segmentada

4.4.2. OPENCV

OpenCV (Open Source Computer Vision) es una herramienta de libre distribución, orientada a los procesos de visión artificial.

La herramienta se ejecuta en las plataformas Microsoft Windows, GNU/Linux, Mac y Android. La misma está desarrollada en C, C++ y Python.

La herramienta posee más de 2000 algoritmos estandarizados para operaciones sobre imágenes que la convierten en uno de los utilitarios más completos para el tratamiento de imágenes. A su vez ofrece una variedad de ejemplos que ayudan a comprender su funcionamiento [33].

La estructura general de OpenCV está basada en 5 librerías que se detallan a continuación [36]:

- **CXCORE:** Es la librería que posee las estructuras y algoritmos básicos que usan las demás funciones.
- **CV:** Implementa las funciones principales de procesamiento.
- **HighGUI:** Sirve para realizar la interfaz gráfica e interacción con archivos de imágenes y video.
- **ML:** Posee algoritmos de aprendizaje y clasificadores.
- **CVAux:** Admiten algoritmos experimentales BG/FG.

OpenCV proporciona el manejo de estructuras básicas, procesado de imágenes, análisis de movimiento, reconocimiento del modelo, reconstrucción 3D, calibración de la cámara e interfaz gráfica [32]. A su vez trabaja con varios formatos de imágenes como: BMP, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF.

Para apreciar sus funcionalidad se ha realizado una prueba mediante la utilización del filtro LoG[12] los resultados se pueden apreciar en la figura 4.8 en donde claramente se visualiza cada uno de los procesos que va realizando para la detección de bordes.

[12] LoG: Operador Laplaciano de Gaussiano. Mayor detalle se encontrara en el Capitulo 2.4.

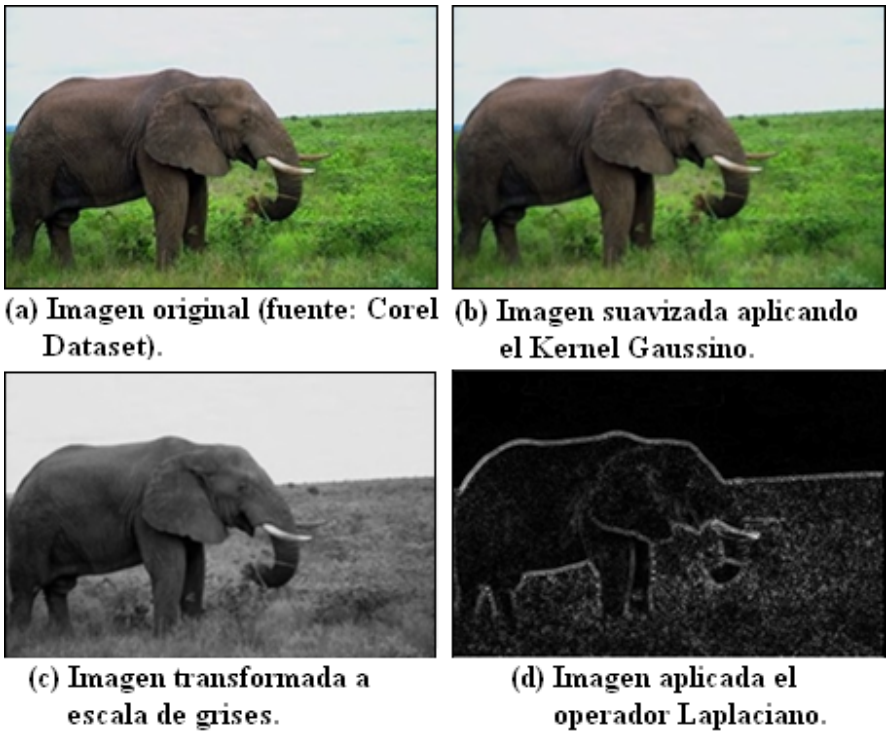


Figura 4.6: Aplicación de la herramienta OpenCV.

4.4.3. SEGMENT

Segment es un paquete generado por la universidad de Texas en conjunto con la empresa Austin Robot Technology.

Este paquete realiza la segmentación utilizando el lenguaje C++ puro, no utiliza librerías predefinidas para el procesamiento de imágenes, realiza un suavizado basado en la función Gaussiana y la detección de bordes con el operador Laplaciano para la posterior segmentación.

El resultado de la segmentación a través de esta herramienta se puede apreciar en el figura 4.9:

(a) Imagen original (fuente: Corel Dataset). (b) Imagen segmentada

Figura 4.7: Imagen aplicada la segmentación con la herramienta Segment.

4.5. COMPARATIVA Y SELECCIÓN

Se ha usado una tabla que recopila características y funcionalidades para realizar un análisis comparativo de las herramientas. En la tabla 4.1 se puede observar los parámetros considerados para determinar la herramienta adecuada para el tratamiento de imágenes.

	SIFT Keypoint	Microsoft Photosynth	Blobdemo	OpenCV	Segment
Tipo de imágenes que recibe	PGM	JPG	PNG	BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF.	PPM
Algoritmos Utilizados	SIFT	SIFT	*Kernel* Gaussiano y Operador Laplaciano	Soporta más de 500 funciones para el tratamiento de imágenes dentro de las cuales muchos de estos aplican varias funciones y operadores que conllevan a la segmentación de la imagen.	*Kernel* Gaussiano y Operador Laplaciano
Acceso al Código	Código abierto	Privativo	Código abierto	Código abierto	Código abierto
Portabilidad	Microsoft Windows y GNU/Linux	Microsoft Windows	Microsoft Windows y GNU/Linux	Microsoft Windows, GNU/Linux y Mac	Microsoft Windows y GNU/Linux
Librerías para tratamiento de imágenes	No soportado	No soportado	No soportado	Soportado	No soportado

Tabla 4.1: Comparativa de herramientas.

Al analizar cada uno de los beneficios que presentan las herramientas se ha optado por la herramienta OpenCV debido a las prestaciones que ofrece la misma, para desarrollar el prototipo aplicaremos el filtro Gaussiano y el operador Laplaciano y como siguiente fase se procede a la segmentación.

OpenCV ofrece múltiples beneficios debido a que la herramienta está enfocada principalmente en el tratamiento de imágenes permitiendo agilitar los procesos y presentar resultados más precisos.

También es importante considerar que sus librerías son código abierto lo cual permite la utilización y modificación para acoplar a los diversos tratamientos que se puede dar a una imagen.

Las herramientas **SIFT Keypoint** y Photosynth realizan tratamiento de imágenes basados en algoritmos como SIFT con la cual se procede a la detección de puntos de interés y no permite efectuar la segmentación, de igual forma **Blobdemo** segmenta imágenes poco complejas en las cuales existe una diferencia entre los objetos y el fondo de la imagen lo que facilita la segmentación. Mientras que la herramienta **Segment** no posee librerías para el tratamiento de imágenes debido a que utiliza código de C++ puro.

CAPITULO 5

DISEÑO DE UN PROTOTIPO

5. DISEÑO DE UN PROTOTIPO

5.1. SELECCIÓN Y PREPARACIÓN DEL CORPUS

La selección del corpus se hace en base a ciertos atributos de la imagen como es el color, textura y complejidad. Por esta razón se ha escogido las imágenes del Corel proporcionado por [37]. Este corpus se Corel DataSet posee mil imágenes clasificadas en carpetas del 0 al 9 dependiendo de la categoría. Todas las imágenes se encuentran en formato PPM[13].

La complejidad de las imágenes del *Corel DataSet* es diferente en cada una de las categorías, puesto que se encuentran imágenes donde el número de objetos varían. Asimismo, existen categorías donde los objetos se solapan y las texturas tienen tramas más complicadas. Haciendo un análisis general de las imágenes se ha calculado que en promedio poseen 4 objetos en cada imagen del *Corel DataSet*.

Las imágenes poseen 16 bits para representar el color en modo RGB[14], tomando en cuenta que mientras mayor sea la profundidad de color, mayor será la cantidad de tonos que puedan ser representados aumentando la gama de colores disponibles. Esto hará que sea más precisa la representación del color en la imagen digital.

La textura es otro factor importante para la segmentación ya que las imágenes del *Corel DataSet* poseen varias texturas que van variando de acuerdo a la categoría e imagen. La textura de una imagen está dada por el orden espacial y sus colores o intensidades [38]. En las imágenes se puede definir la textura analizando ciertas propiedades como: granularidad, dirección, repetitividad, aleatoriedad y linealidad, entre otras.

[13] PPM: Portable PixMap (Mapa de pixeles portables).
[14] RGB: Red Green Blue.

CARACTERISTICAS				
Textura	**Color**	**Complejidad**	**Formato**	**# de objetos**
Granularidad Repetitividad Linealidad	16 bits modo RGB	Varios objetos y sub objetos que no se encuentran claramente definidos.	PPM	Un Promedio de 4 objetos dependiendo de la imagen.

Tabla 5.1: Características de las imágenes.

En la figura 5.1 se puede observar las distintas categorías de imagen que posee el *Corel DataSet*.

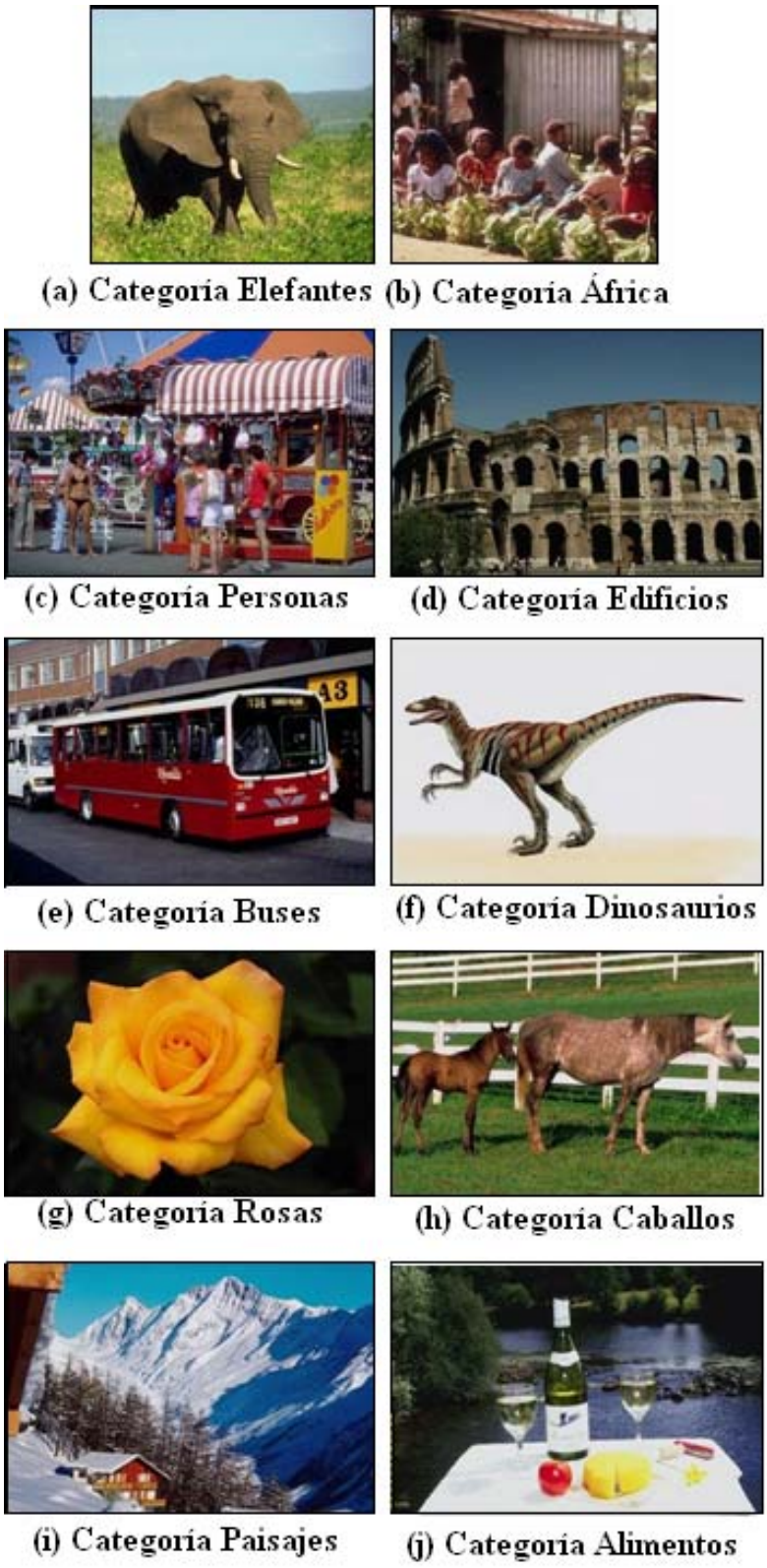


Figura 5.1: Imágenes del Corel dataset (fuente: *Corel DataSet*).

Para la preparación del corpus se han valorado ciertas características de las imágenes, como habíamos mencionado anteriormente, cada imagen es diferente por lo tanto su grado de complejidad varía, ya sea por el número de objetos que posea o por la textura que presente.

Para nuestro proceso de experimentación se ha elegido 2 categorías, puesto que el proceso de segmentación no solo se realizará automáticamente sino manualmente para lo cual se requiere más recursos humanos, materiales y el tiempo no permite hacer un análisis a todas las 10 categorías del corpus. Por lo tanto, las categorías seleccionadas para la posterior experimentación son: edificaciones, la cual se encuentran dentro de la carpeta número dos y dinosaurios que están dentro de la capeta cuatro.

En el caso de los edificios, estos poseen diferentes texturas para cada imagen e igualmente presentan diferente luminosidad en los objetos y sub objetos que los componen. También es importante mencionar, que los edificios poseen sombras en algunas imágenes y cada característica va variando de acuerdo al ángulo en el cual fue tomada la imagen.

En las categorías “dinosaurios” y “monumentos” se puede observar sus cambios de textura, pero en el aspecto de luminosidad y presencia de sombras su aparición no es considerable, por lo tanto su segmentación es mejor.

5.2. DISEÑO DEL PLAN DE EXPERIMENTACIÓN

Objetivo del experimento

La experimentación se llevará a cabo con el fin de que en la función Gaussiana, en el operador Laplaciano y en los métodos de dilatación y segmentación se puedan encontrar los valores apropiados de sus variables, para que de este modo se pueda medir su impacto en la operación de segmentación. Es importante considerar las

capacidades y prestaciones brindadas por cada una de las funciones, métodos y operadores en cuestión, mismos que son aplicados en secuencia hasta la extracción de los objetos de la imagen.

Identificación de las fuentes de variación

Para obtener el resultado de la extracción de objetos de la imagen, se debe pasar por diferentes fases de tratamiento de la imagen en cuestión. Existen factores de tratamiento que son determinantes en la calidad del resultado esperado, pero que se encuentran distribuidas en algunas fases.

A continuación se describen los correspondientes factores de tratamiento:

- **Factores de tratamiento en la etapa del suavizado de la imagen.-** En esta etapa los factores de tratamiento son las variables: sigma y *Kernel*. La primera corresponde a la desviación estándar, mientras que la segunda es el tamaño de la máscara Gaussiana. Las condiciones a las que están sujetas estas variables son:

    - **Sigma:**

        - La desviación estándar tomará valores mayores que cero.

    - ***Kernel* del filtro Gaussiano:**

        - Para determinar el tamaño de la máscara de convolución, el tamaño de la variable *Kernel* tomará valores positivos impares mayores que uno.

    - **Relación entre sigma y *Kernel*:**

- Se utilizará la fórmula $w \geq 3\left(2\sqrt{2\sigma^2}\right)$, para obtener el tamaño de la máscara de convolución en función del tamaño de la desviación estándar.

- **Factores de tratamiento en la etapa de detección de bordes.-** El factor de tratamiento en esta fase, es la variable que determina el tamaño de la máscara de convolución del operador Laplaciano. A continuación se describen las condiciones establecidas para esta variable:

  - ***Kernel* del operador Laplaciano:**

    - El valor que tomará esta variable se ha establecido que sea 3, definiendo así, una máscara de 3x3 para el proceso de la detección de los bordes.

    - Además, en caso de cambiar el valor de esta variable se debe tomar en cuenta que sea entera, positiva e impar.

- **Factores de tratamiento en la etapa de dilatación.-** En la dilatación el factor más importante es el valor del tamaño que será asignado al elemento de estructuración. Este elemento de estructuración puede ser rectangular, elíptico o en forma de cruz, de tal manera que los bordes detectados por el operador Laplaciano serán dilatados de acuerdo al tamaño asignado al elemento elegido. Los condicionamientos aplicados a estas variables son:

  - **Elemento de estructuración:**

    - Este elemento será de la forma rectangular, ya que así se mantendrá sin distorsionar la forma de los bordes resultantes de la fase de detección de bordes.

- **Tamaño del elemento de estructuración:**

    - El tamaño deberá ser positivo, mayor o igual a cero, de tal manera que en caso de asignarle un valor de cero, significa que se realizará una dilatación sin que surta efecto en los bordes detectados. Esto se realizará en caso que los resultados de la extracción de objetos sea óptima sin la necesidad de aplicar el proceso de dilatación, o con una dilatación cero.

Unidades experimentales

Las 200 imágenes previamente seleccionadas del *Corel Data Set*, son las unidades experimentales donde será evaluada la extracción de objetos, y a las que se les aplicarán los factores de tratamiento. Estas imágenes se encuentran divididas en dos grupos de 100 cada una, el primero corresponde a fotografías de dinosaurios, mientras que el segundo grupo está conformado por imágenes de edificaciones.

Respuesta de interés a analizar

Las respuestas experimentales que serán objeto de análisis están enfocadas en la calidad de la extracción de objetos, para lo cual se consideran los siguientes aspectos:

- **Suavizado de la imagen.-** El suavizado de la imagen debe ser efectivo, esto es, difuminar la imagen sin que se pierdan los detalles ni los contornos de los objetos, ya que esto afectaría en la detección de bordes y por ende se obtendría un resultado negativo en la obtención de los objetos contenidos en la imagen.

- **Detección de bordes.-** De la misma manera que el suavizado, se deben ajustar los parámetros del operador Laplaciano, con la finalidad de conseguir que los bordes de los objetos sean detectados

correctamente, en búsqueda de marcar los contornos de cada objeto de la imagen. Este paso es fundamental para garantizar la calidad de la extracción de objetos.

- **Proceso de dilatación.**- En el caso de que el operador Laplaciano deje ciertos contornos sin cerrarlos por completo, es la dilatación el proceso que se encarga de corregirlo y de aclarar los bordes de los objetos detectados previamente. Al ser este el paso previo a la segmentación, amerita a que su aplicación sea efectiva, para lo cual se recurrirá al ajuste de sus variables.

- **Segmentación.-** Esta es la etapa final donde se procesa a la imagen dilatada, con el fin de detectar los contornos completamente encerrados y proceder a extraer sus objetos mediante un procedimiento de etiquetado. Esta tarea consiste en darle un color aleatorio y diferente a cada objeto encontrado. Asimismo, se requiere del ajuste apropiado de ciertos parámetros.

Procedimientos de la experimentación

La experimentación se llevará a cabo siguiendo ciertos pasos y procedimientos, desde la selección de las imágenes hasta el almacenamiento de los resultados, mismos que son producto de la extracción de los objetos detectados dentro de cada imagen.

La estructuración de los procedimientos que se seguirán en la experimentación, se puede ver en la figura 5.2.

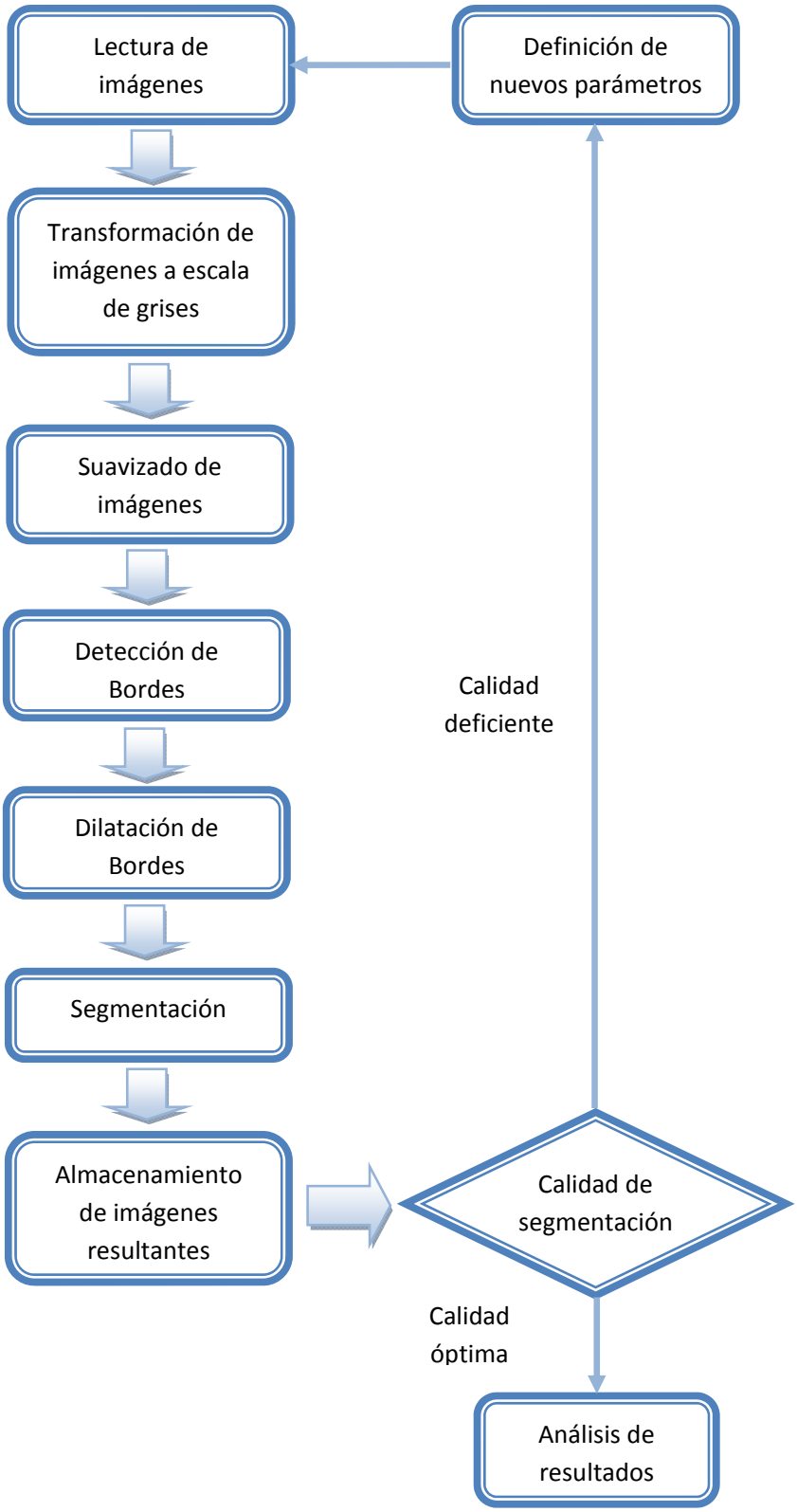


Figura 5.2: Esquema del proceso de experimentación.

A continuación se detallan los procedimientos que forman parte de la experimentación:

- **Selección de imágenes.-** Las imágenes fueron seleccionadas aleatoriamente de un conjunto de mil imágenes a color, mismas que están divididas en diez categorías compuestas por cien imágenes cada una. Se tomaron dos categorías, la primera pertenece a la categoría de imágenes de dinosaurios, mientras que la segunda corresponde a la categoría de edificaciones. Una descripción detallada sobre el origen del conjunto de imágenes y la selección se encuentra en el apartado 5.1.

- **Lectura de imágenes.-** En este punto se procederá a capturar la imagen del conjunto de imágenes previamente seleccionado.

- **Transformación de imágenes a escala de grises.-** El tratamiento de imágenes se debe realizar aplicando técnicas para imágenes monocromáticas. Por consiguiente, es necesario realizar la conversión de la imagen a escala de grises.

- **Suavizado de imágenes.-** Una vez que se obtiene la imagen en escala de grises, se debe difuminar a la imagen con el fin de regularizarla mediante la reducción del ruido. Para realizar esta tarea se utilizará el filtro Gaussiano. Este filtro requiere que se establezcan los valores de la desviación estándar y del tamaño del *Kernel* Gaussiano que definirán el nivel de suavizado, a fin de garantizar la robustez de la detección de los bordes en la siguiente etapa.

- **Detección de bordes.-** Una vez obtenida la imagen regularizada por el filtro Gaussiano, se procede a la etapa en donde se utilizará al operador Laplaciano para conseguir la detección de los bordes. Aquí

se trabajará con las condiciones establecidas en los factores de tratamiento.

- **Dilatación de bordes.-** En caso de ser necesario se aplicará la dilatación de la imagen para conseguir que los bordes sean resaltados y que los contornos logren cerrarse. Esto garantizará que los objetos detectados por el operador Laplaciano puedan ser extraídos en la fase de segmentación.

- **Segmentación.-** La imagen procesada por la dilatación, será tratada en esta fase de segmentación. En esta fase se detectarán los contornos cerrados, identificándoles como objetos, y se etiquetarán asignándoles un color diferente a cada uno de ellos.

- **Almacenamiento de imágenes resultantes.-** Finalmente, la imagen resultante se guardará en un directorio específico, como un archivo con extensión PPM, para el análisis posterior a la que estará sujeta.

- **Repetición del procedimiento.-** Los procedimientos, desde el suavizado de la imagen hasta el almacenamiento de las imágenes resultantes, se deben repetir ajustando los factores de tratamiento, hasta conseguir una segmentación y extracción de objetos de la manera más óptima posible.

Es de indicar que se automatizará el entrenamiento, de tal manera que con tan solo una ejecución del mismo, todas las 200 imágenes sean procesadas con los mismos parámetros definidos, y pasen por todas las fases indicadas anteriormente. Es decir, cada proceso arrojará una imagen resultante que a su vez será llamada por el siguiente proceso hasta llegar a la etapa final que es el almacenamiento.

El entrenamiento se ejecutará hasta encontrar la segmentación más óptima posible. Una vez determinados los mejores parámetros, se procederá a realizar el análisis respectivo de ese resultado en un apartado posterior.

5.3. PRUEBAS CON LAS HERRAMIENTAS SELECCIONADAS

En el capítulo 4 se realizó la selección de un conjunto de herramientas para el tratamiento de imágenes. Estas fueron comparadas en el apartado 4.5, y se decidió que la herramienta que mejores prestaciones tiene para llevar a cabo el proceso de extracción de objetos de las imágenes es la librería OpenCV.

Por consiguiente, en este punto se realizarán las pruebas de la herramienta seleccionada, con el fin de verificar que en la práctica las prestaciones de esta sean efectivas.

A continuación se presentan los resultados de la prueba ejecutada:

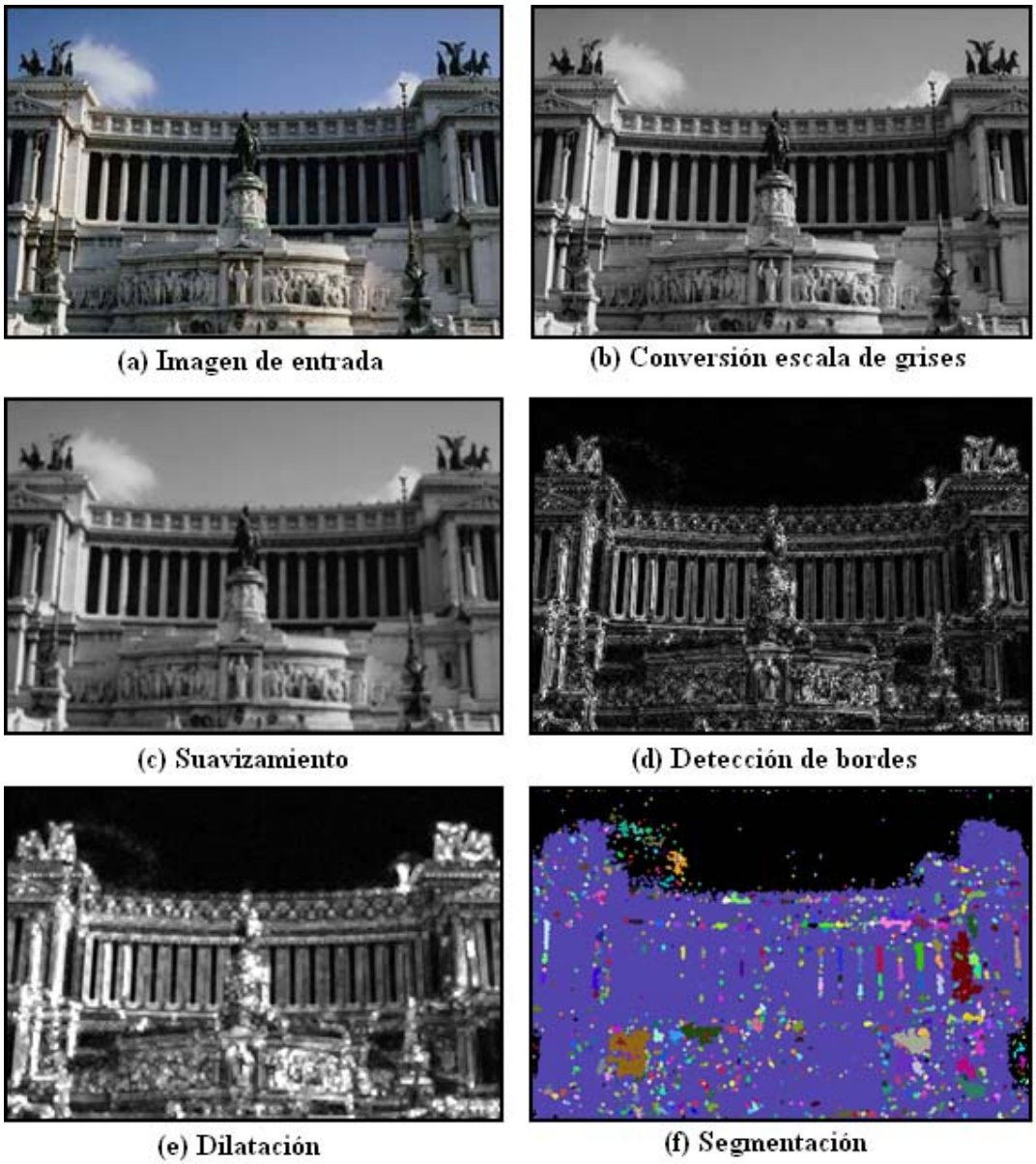


Figura 5.3: Resultados de las pruebas con OpenCV.

La prueba se ejecutó sobre una imagen tomada aleatoriamente del conjunto de imágenes seleccionadas para el entrenamiento. En la figura 5.3 (a) está la imagen a la que se le aplicaron todos los procedimientos que se requieren para la extracción de objetos. Con la función *imread*[15] se hizo la lectura de la imagen.

[15] Las funciones *imread*, *ctvColor*, *GaussianBlur*, *Laplacian*, *dilate*, *findContours*, *drawContours* y *circle* forman parte de la librería OpenCV.

Para detectar los bordes el operador Laplaciano trabaja sobre imágenes monocromáticas. Por consiguiente, se convirtió la imagen de entrada a escala de grises mediante la función *cvtColor*. El resultado se visualiza en la figura 5.3 (b).

El siguiente paso que se dio fue la eliminación del ruido de la imagen mediante el filtro Gaussiano. Para el efecto, se utilizó la función *GaussianBlur*. La figura 5.3 (c) presenta el resultado de aplicar este operador sobre la imagen en escala de grises.

En la detección de bordes se recibió la imagen previamente suavizada, y se le aplicó el operador Laplaciano mediante la función *Laplacian*. En la figura 5.3 (d) se observa la imagen saliente.

Posteriormente, la imagen con los bordes detectados, pasa al proceso de dilatación mediante la función dilate. La salida de este proceso se ve en la figura 5.3 (e).

Finalmente se procedió a segmentar la imagen mediante la búsqueda de contornos cerrados con la función *findContours*, y para extraer los objetos detectados se procedió a etiquetarlos asignándole un color diferente a cada uno. Esto se logró con la función *drawContours* y *circle*.

En conclusión, los resultados de la extracción de objetos son alentadores, y la herramienta es flexible, ya que permite mejorar la segmentación a través del cambio de parámetros tales como: la desviación estándar, el tamaño de la máscara para el proceso de suavizado y el tamaño del elemento estructural de la dilatación, para asegurar que los contornos estén completamente cerrados.

5.4. DISEÑO DEL PROTOTIPO

En este apartado se presenta el diseño del prototipo para la segmentación, y se describe su funcionamiento.

El prototipo de segmentación se compone de dos módulos: el módulo de pruebas de parámetros y el módulo de segmentación automatizada, tal como se puede ver en la figura 5.4.

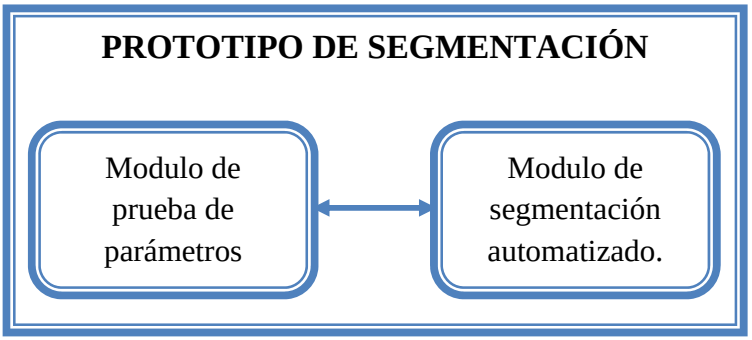


Figura 5.4: Resultados de las pruebas con OpenCV.

5.4.1. MÓDULO DE PRUEBAS DE PARÁMETROS

El diseño de este módulo se puede ver en la figura 5.5, y está diseñado para segmentar una imagen con el fin de regular los parámetros hasta obtener la segmentación más óptima posible.

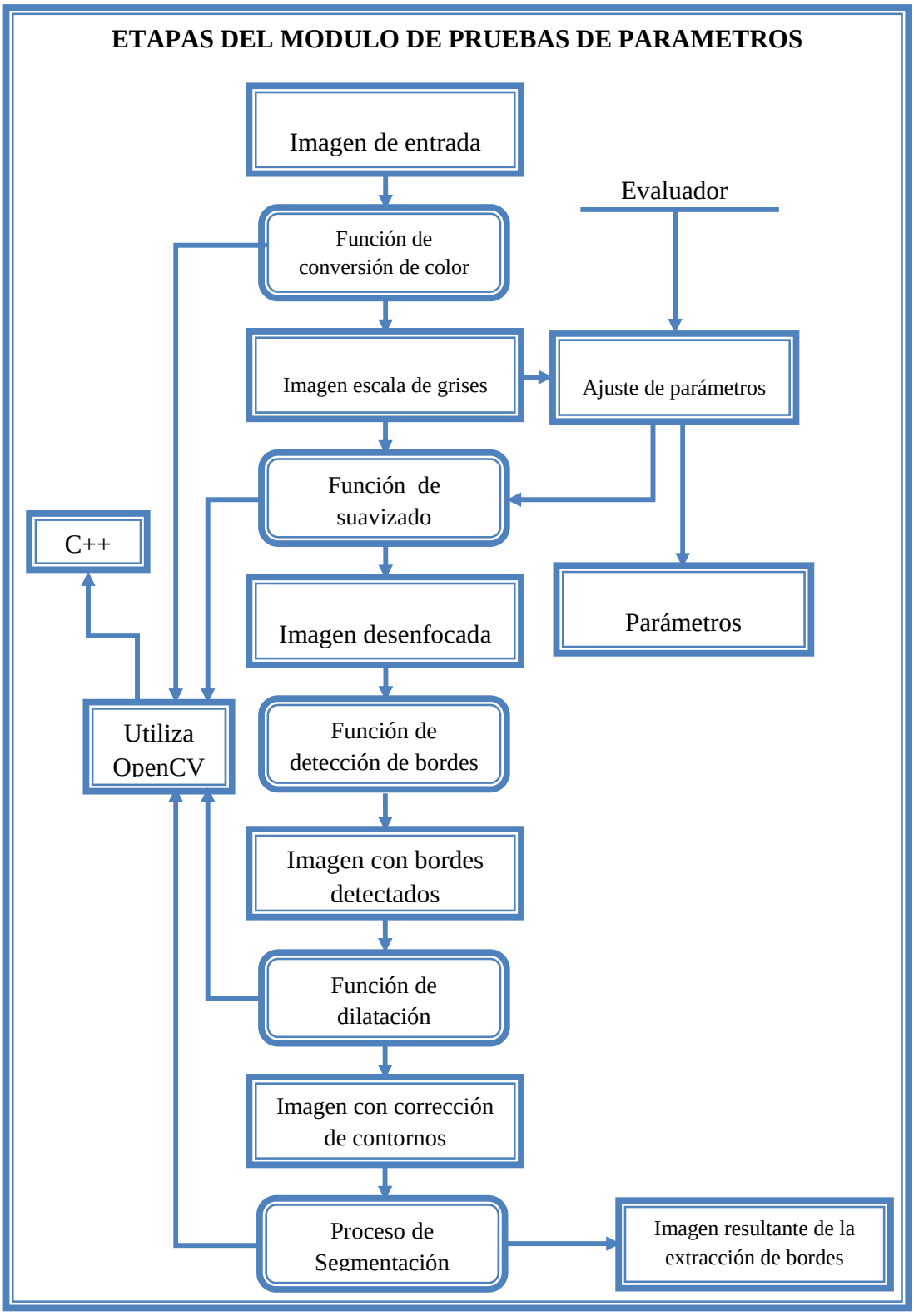


Figura 5.5: Módulo de prueba de parámetros.

En el diseño se puede evidenciar que la implementación se realiza en C++ y utilizando la librería de OpenCV.

Este módulo está compuesto por etapas, donde se especifican las funciones y sus parámetros relevantes:

Imagen de entrada.- La imagen de ingreso debe ser de color, y la función que realiza la lectura es la siguiente:

- imread(nombre_imagen)

Función de conversión de color.- Una vez que se haya hecho la lectura de la imagen original, se procede a cambiar su formato de color a escala de grises. Esta transformación se lleva a cabo por medio de la función:

- cvtColor(src, dst, *code*)

  Parámetros:

  - src.- Imagen obtenida del proceso de lectura.
  - dst.- Imagen resultante una vez que ha sido transformada a escala de grises.
  - *code*: CV_RGB2GRAY.- Parámetro que especifica que la transformación sea realizada a escala de grises.

Función de suavizado.- A la imagen en escala de grises se le aplicará el desenfoque Gaussiano, mismo que se llevará a cabo por esta función:

- GaussianBlur(src, dst, Size (ksize, ksize), sigmaX, sigmaY=0, *BORDER_DEFAULT*)

Parámetros:

- src.- Imagen en escala de grises.
- dst.- Imagen suavizada una vez aplicado el filtro Gaussiano.
- Size(ksize, ksize).- Es el tamaño de la máscara. La máscara deberá ser cuadrada, positiva e impar.
- SigmaX, SigmaY.- Valor de la desviación estándar del *Kernel* Gaussiano. Estas dos variables toman el mismo valor de sigma.
- *BORDER_DEFAULT*.- Es el método que define la extrapolación de pixeles.

Función de detección de bordes.- Se realiza la detección de bordes sobre la imagen que fue suavizada. Para ello, se utiliza el operador Laplaciano mediante la siguiente función:

- Laplacian(src, dst, *ddepth*, *ksize*, *scale*, *delta*, *BORDER_DEFAULT*)

Parámetros:

- src.- Imagen previamente suavizada.
- dst.- Imagen con el resultado de la detección de bordes.
- *ddepth*.- La profundidad deseada de la imagen de destino.
- *ksize* – Define el tamaño de la máscara laplaciana. Debe ser positivo e impar.
- *scale* – El factor de escala opcional para los valores laplacianos calculados ().

- delta – El valor delta es opcional, sumado a los resultados antes de almacenarla en dst.
- *BORDER_DEFAULT* – El método extrapolación de pixeles.

Función de dilatación.- Esta operación se ejecuta para encerrar los contornos de los bordes detectados por el operador Laplaciano, empleando la función:

- dilate(src, dst, *element*)

  Parámetros:
  - src.- Imagen producto de la detección de bordes.
  - dst.- Imagen resultante de la dilatación.
  - *element*.- Este parámetro es el elemento estructural utilizado para la dilatación. Este elemento es calculado a través de la función *getStructuringElement( ),* mismo que recibe como entrada dos variables, la primera indica el tipo de elemento como puede ser: rectangular, elipsoide y en forma de cruz. Mientras que la segunda variable determina el tamaño del tipo de elemento estructural.

Proceso de segmentación.- Al aplicar este procedimiento se pretende obtener la segmentación y extracción de los objetos por medio de la identificación y etiquetado de contornos. Para este proceso se aplicarán las siguientes funciones:

- **findContours**.- A través de esta función se consigue almacenar los contornos encontrados en la imagen.

findContours(image, *contours, mode, method, Point()*)

Parámetros:

- image.- Esta es la imagen que está lista para ser segmentada y etiquetada, ya que ha pasado por los procesos de: cambio a escala de grises, suavizado, detección de bordes y dilatación.

- *contours*.- Es el vector donde se guardan los contornos encontrados en la imagen. Los contornos son almacenados como un vector de puntos.

- *mode*.- Es el modo de recuperación del contorno, en este caso se utiliza el modo CV_RETR_TREE, ya que con este se recuperan todos los contornos.

- *Method*.- Es el método de aproximación de contornos.

- *Point*.- Es el desfase opcional, por e cual cada punto del contorno se desplaza

- **drawContours.-** Esta función recibe el vector de contornos para proceder a dibujarlos como objetos independientes dentro de la imagen.

  drawContours(image, *contours, contourIdx,* color, *thickness, lineType, hierarchy).*

  Parámetros:

  - image.- Es la imagen destino, donde se dibujan los contornos.

- *contours*.- Vector con los contornos detectados previamente.
- *contourIdx*.- Indica el contorno a dibujar. Si es negativo, todos los contornos se dibujan.
- color.- Es el color asignado a cada contorno.
- *Thickness*.- Valor del grosor de las líneas que dibujan al contorno. Si es negativo, es decir, thickness = CV_FILLED, entonces el interior del contorno se dibuja.
- *lineType*.- Es el tipo de línea para conectar los píxeles.
- *hierarchy*.- La información opcional sobre la jerarquía. Solo es necesario si se requiere dibujar solo algunos de los contornos

Ajuste de parámetros.- Los parámetros serán evaluados manualmente, de tal manera que se podrán ajustar independientemente los valores de las funciones. Los valores que se modificarán son el valor de la desviación estándar y el tamaño del *Kernel* en la función Gaussiana, de igual forma en el procedimiento de dilatación se podrá variar los valores del tipo y tamaño del elemento estructural. Esto se hace considerando que los parámetros indicados los más importantes para conseguir una adecuada segmentación y en su efecto la eficiente extracción de objetos.

Parámetros adecuados.- Los parámetros adecuados serán aquellos que al aplicarlos en la extracción de objetos, brinden el mejor resultado y servirán de entrada para el módulo de segmentación automatizada.

5.4.2. MÓDULO DE SEGMENTACIÓN AUTOMATIZADO

El módulo de segmentación automatizado fue diseñado para ejecutar la segmentación del corpus, es decir, para procesar las 200 imágenes a través del script de segmentación. Para este procedimiento se toman como entrada los parámetros adecuados que son definidos en el módulo de pruebas de parámetros. En la figura 5.6 se observa el diseño de este módulo.

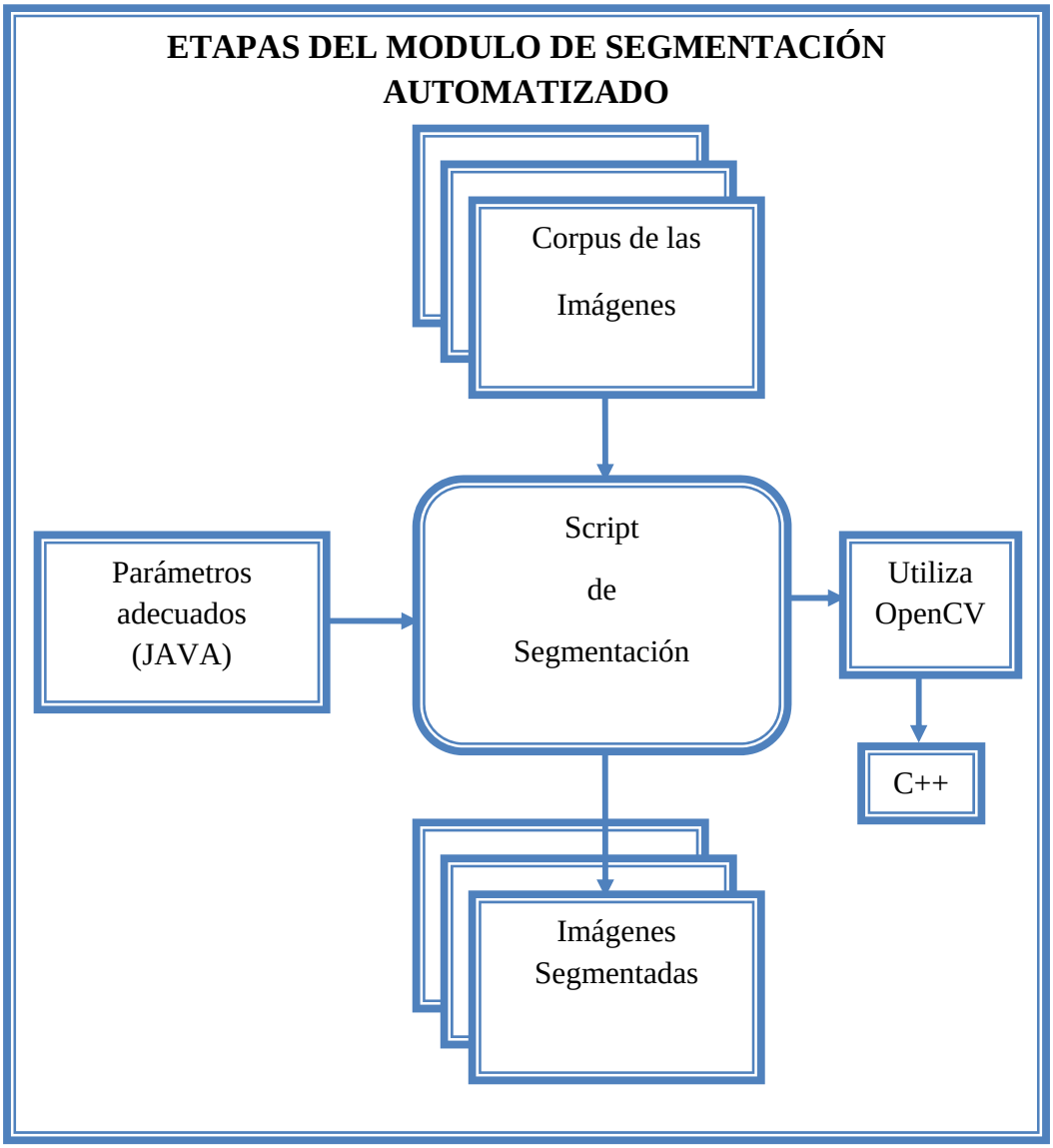


Figura 5.6: Módulo de segmentación automatizado.

De forma detallada, a continuación se describe la ejecución de este módulo:

Interfaz de ejecución del script.- La ejecución se realizará a través de una interfaz gráfica, esta será implementada en el lenguaje de programación Java, a través del editor NetBeans 6.8. Por medio de esta se ingresarán los siguientes parámetros:

- **Función Gaussiana**
  - Desviación estándar
  - Tamaño del Kernel Gaussiano

- **Proceso de dilatación**
  - Tamaño del elemento de estructuración
  - Tipo del elemento de estructuración

Una vez que se trabaje con la interfaz, se realizará la lectura automática de las 200 imágenes que conforman el corpus que estarán ubicadas en un directorio previamente definido. A continuación desde Java se llamará al script de segmentación que implementado en el lenguaje C++ con la utilización de la librería OpenCV. Este script ejecuta secuencialmente las funciones descritas en el módulo de prueba de parámetros, hasta obtener la segmentación de cada imagen que se almacenará en un directorio por medio de la función *imwrite(nombre_imagen_segmentada, imagen_segmentada).*

A continuación en la figura 5.7 se puede observar el diseño de la interfaz gráfica:

Figura 5.7: Interfaz gráfica para la ejecución automática de la segmentación.

CAPITULO 6

RESULTADO Y DISCUSION

6. RESULTADOS Y DISCUSIÓN

6.1. EJECUCIÓN DEL PLAN DE EXPERIMENTACIÓN

El plan de experimentación se ejecuta en dos etapas, en la primera etapa se pone en marcha el módulo de prueba de parámetros, y una vez definidos los valores adecuados de los parámetros que dan un resultado óptimo, se procede a inicializar el módulo de segmentación automatizado con estos parámetros. Los resultados de estas ejecuciones se detallan a continuación.

6.1.1. EJECUCIÓN DEL MÓDULO DE PRUEBA DE PARÁMETROS

Esta ejecución se llevó a cabo desde la consola de Windows. Para el efecto se realizó lo siguiente:

1. Se ingresó al directorio donde se encuentra el archivo ejecutable generado en C++ y la imagen a ser procesada.

2. En esta ubicación se enviaron los parámetros que recibe la aplicación y se enumeran a continuación, de acuerdo al orden de envío:

    - Parámetro 1: Archivo ejecutable.
    - Parámetro 2: Imagen de entrada.
    - Parámetro 3: Sigma.
    - Parámetro 4: *Kernel*.
    - Parámetro 5: Tamaño del elemento de estructuración.
    - Parámetro 6: Tipo del elemento de estructuración

El resultado de ejecutar esta orden se puede observar en la figura 6.1.

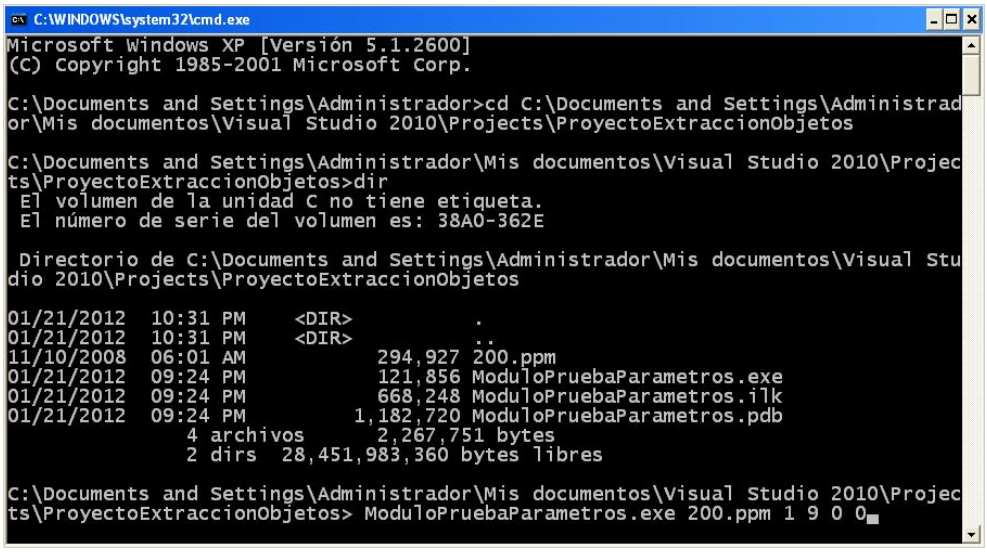


Figura 6.1: Ejecución de la aplicación de pruebas.

Se realizaron 35 pruebas, de las cuales se definió el mejor resultado, sin embargo, consideramos importante presentar los tres resultados más aceptables:

Prueba 1:

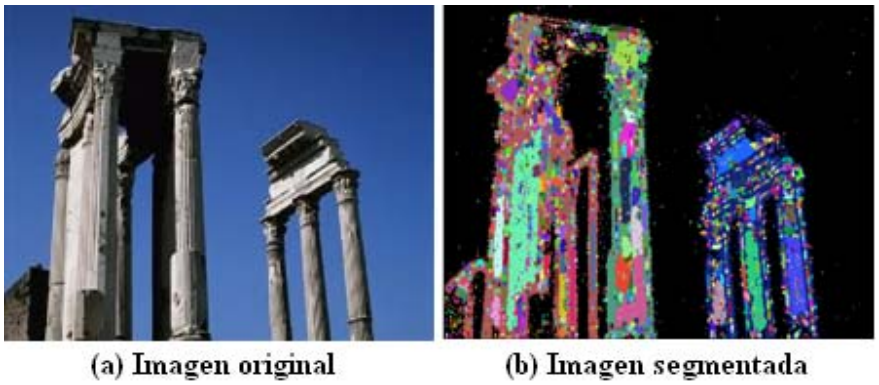


Figura 6.2: Segmentación con sigma = 1.85, *Kernel* = 9, sin dilatación.

Prueba 2:

Figura 6.3: Segmentación con sigma = 0.01, *Kernel* = 7, sin dilatación.

Prueba 3:

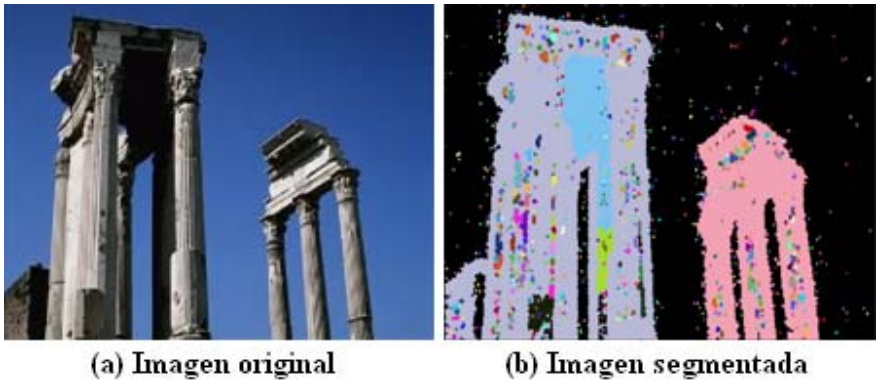


Figura 6.4: Segmentación con sigma = 0.01, *Kernel* = 7, tipo de elemento = 1 (rectangular), tamaño del elemento estructural = 1.

De las tres pruebas se eligió la prueba 2, esto debido a que el resultado presenta mayor homogeneidad en el color asignado a cada objeto. Mientras que la prueba 1 es descartada porque a un mismo objeto a través de la asignación del color le divide en dos partes. Finalmente, en la prueba 3 al aplicar la dilatación presenta buenos resultados, con mayor disminución de ruido, sin embargo debido a la dilatación se pierden detalles en imágenes más complejas. Por ejemplo, en caso

de que los objetos se encuentren juntos o sobrepuestos, estos se fusionan en uno solo, impidiendo que los podamos diferenciar y por lo tanto se descarta esta prueba.

6.1.2. EJECUCIÓN DEL MÓDULO DE SEGMENTACIÓN AUTOMATIZADO.

Los valores de los parámetros utilizados en la prueba 2 son aplicados para la segmentación automatizada. Este procedimiento es realizado mediante una interfaz gráfica, como se muestra en la figura 6.5, donde se ingresarán los valores correspondientes.

Figura 6.5: Ejecución de la segmentación automatizada.

Una vez ingresados los valores se procede a ejecutar la aplicación. Este proceso tardó 3 minutos con 20 segundos en segmentar y guardar las 200 imágenes del corpus. Los resultados son analizados en el siguiente apartado.

6.2. ANÁLISIS DE RESULTADOS

En este punto se procede al análisis de los resultados de la segmentación. Este análisis se realiza a través de la comparación de la segmentación del sistema con la segmentación manual. Esto consiste en determinar el porcentaje de precisión en la extracción de objetos se ha conseguido, para esto se tomó una muestra aleatoria de 50 de las 200 imágenes.

De las 50 imágenes segmentadas se decidió que 32 imágenes aprueban positivamente, ya que los objetos extraídos son identificables. Mientras que las 18 restantes son rechazadas porque no se puede distinguir a cada uno de los objetos, tal como se lo hace en la segmentación manual.

Para una explicación más objetiva se presentan 4 resultados representativos para el análisis:

En la figura 6.6 tenemos los resultados de la segmentación automatizada versus la segmentación manual; en la segmentación (a), observamos que el evaluador humano es preciso al segmentar el objeto, mientras que la automatizada presenta una segmentación con cierto grado de imprecisión, debido a que un fragmento ajeno al objeto es reconocido como parte del mismo.

En la segmentación (b) en cambio, tenemos una gran precisión en la segmentación automatizada con respecto a la segmentación manual.

Mientras que en la segmentación (c), en la automatizada frente a la manual, se puede ver que se pierden los detalles de cada objeto, obteniendo como resultado una sola masa distorsionada e irreconocible.

Finalmente en la segmentación (d), se observa una excelente precisión en la extracción de los dos objetos que conforman la imagen. Claro está que existe un poco de ruido en la parte externa de los objetos, sin embargo los objetos son correctamente identificables como en la segmentación manual.

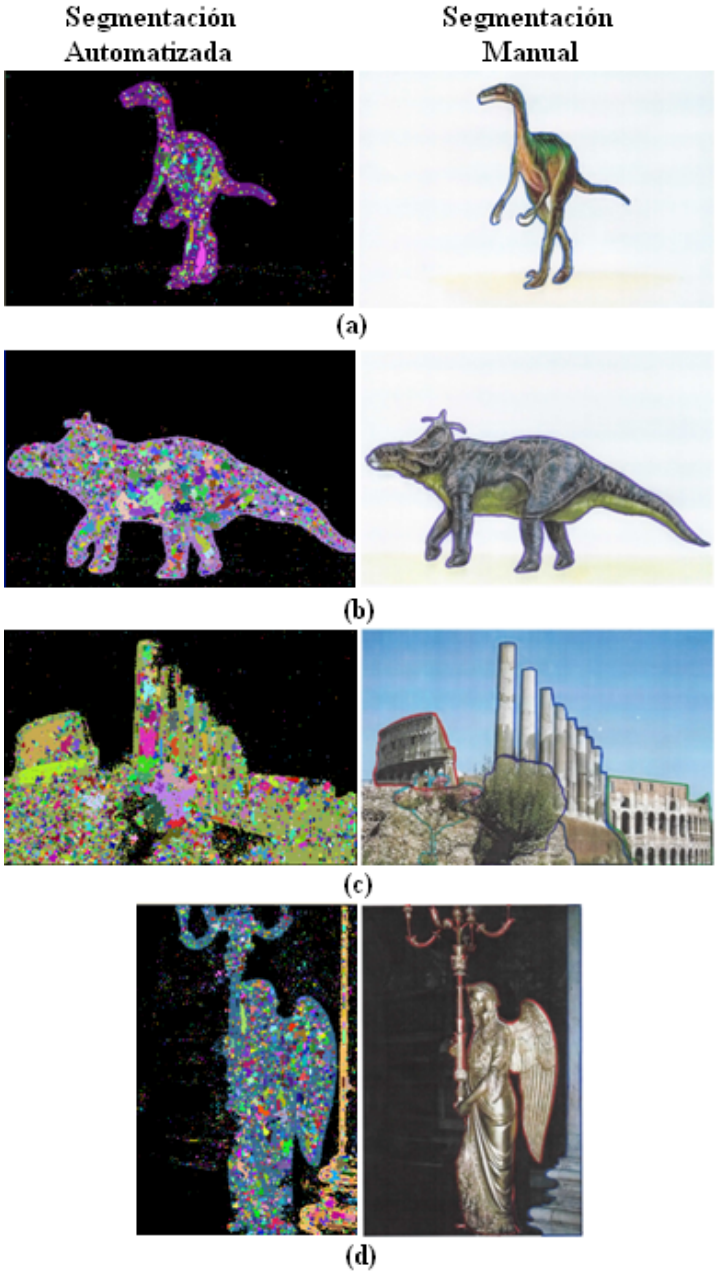


Figura 6.6: Comparativa de la segmentación automatizada con la segmentación hecha por un evaluador humano.

Cabe indicar que los fragmentos de color dentro de cada objeto, representan ya sea a sub-objetos o, son alteraciones de intensidad que lleva dentro. Por lo tanto no son concluyentes dentro del análisis.

En el siguiente punto se calcula y analiza la precisión de estos resultados.

6.2.1. PRECISIÓN

Partimos indicando que la precisión es la fracción de las imágenes segmentadas que presentan resultados relevantes y alentadores.

Este cálculo se realiza en base a la proporción entre las imágenes con resultados relevantes y el total de las imágenes obtenidas de la segmentación.

$$\text{precisión} = \frac{|\text{imágenes relevantes} \cap \text{imágenes obtenidas}|}{|\text{imágenes obtenidas}|} \quad \textbf{(6.1)}$$

Una vez explicada la precisión, se procede al cálculo de la misma:

$$\text{precisión} = \frac{|32 \cap 50|}{|50|} = 0.64 \quad \textbf{(6.2)}$$

En la figura 6.7 se puede apreciar el resultado de la precisión mediante una representación estadística:

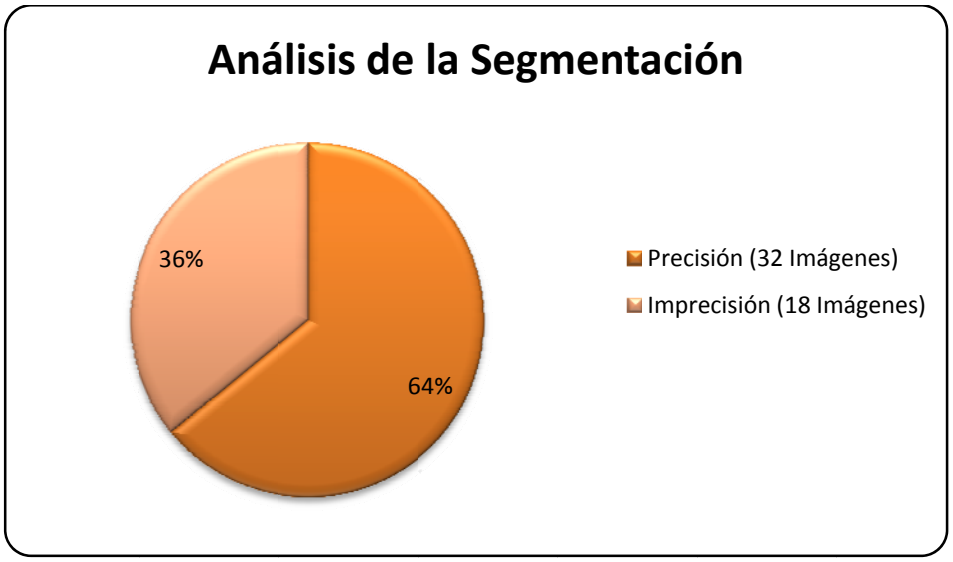


Figura 6.7: Resultado de la precisión.

Concluimos que el 64% de precisión de la segmentación automatizada resulta ser satisfactorio, esto, considerando que la segmentación es un tema abierto porque aún no se consigue obtener una técnica que realice la segmentación íntegra con ninguna metodología. Por consiguiente, el análisis es realizado con cierto grado de flexibilidad sobre los resultados.

6.3. COMPARATIVA CON EL ESTADO DEL ARTE

De acuerdo a la segmentación que se realizó aplicando el operador Laplaciano de Gaussiano, se obtuvieron resultados variantes de acuerdo a la complejidad de cada imagen, es decir, si la imagen se compone de varios objetos, y estos están unidos o solapados, entonces la segmentación tiende a ser deficiente. Esto se produce porque a veces sucede que por la intensidad de los pixeles no se detectan adecuadamente los bordes de cada objeto, o porque los objetos tienen un color homogéneo, y esto hace que al detectar los contornos, uno de estos este formado por dos o más objetos, lo que conlleva a que en la segmentación los objetos sean fusionados, por lo tanto no se los puede distinguir una vez que son etiquetados.

Sin embargo, la segmentación en imágenes que están formadas por objetos que difieren en la intensidad de su color, y también al estar separados el uno del otro, presenta resultados buenos en la segmentación. Eso sí, considerando que dentro de cada objeto detectado también son encontradas y resaltadas áreas pequeñas por causa de la variación de la intensidad y que son también etiquetadas con la asignación de un color.

De acuerdo al estudio realizado por la Universidad Carlos III de Madrid, *Normalized Cut* es considerado como un algoritmo de agrupamiento particional, es decir, se establece el número de regiones en que debe ser dividida la imagen y busca la similitud y disimilitud total de los objetos (basándose en la división de grafos).

Para operar con este algoritmo se debe ingresar el número de regiones en que se dividirá la imagen, a diferencia de nuestro prototipo, el cual no limita la cantidad de regiones y arroja como resultados todos lo que encuentre (ya sea objetos y sub-objetos).

Habiendo analizado de manera general el funcionamiento del algoritmo *Normalized Cut* y el prototipo basado en el algoritmo Laplaciano de Gaussiano definimos que no se puede realizar una comparación directa de precisión, cobertura[16] y F-measure[17] puesto que NCut[18] divide en ***n*** segmentos establecidos por el usuario y nuestro prototipo busca automatizar el proceso de selección de segmentos. Por ello, la medición de precisión la hemos realizado comparando los resultados con la segmentación manual realizada por un humano.

[16] Cobertura: Proporción que abarca un determinado proceso o método.
[17] F-Measure: Es un balance entre los resultados de precisión y cobertura.
[18] NCut: Normalized Cut

6.4. ESPECIFICACIÓN DE MEJORAS PLANTEADAS A LA EXPERIMENTACIÓN

Para aquellos lectores que intenten mejorar los resultados que se obtuvieron en esta tesis, se recomienda el método de segmentación *Mean Shift*, ya que este método por sus características, probablemente logre extraer los objetos de mejor manera. Esto es gracias a que es un algoritmo de agrupamiento y no necesita que se le aplique ningún algoritmo de suavizado previo, porque el mismo aplica el suavizado Gaussiano. Sin embargo, no difumina los bordes, sino más bien los aclara, de tal forma que ayuda a diferenciar entre un objeto y otro.

En términos generales, este tipo de segmentación consiste en encontrar los picos que representan los centros o valores ejemplares de cada tipo de segmento, y determinar cuántos son y agrupar o etiquetar los píxeles según el centro al que se parezcan más, para este procedimiento se utiliza también una máscara que recorrerá la imagen calculando para cada píxel el valor de la moda[19] en base a sus píxeles vecinos.

Además este algoritmo se lo puede encontrar implementado en OpenCV, lo que implicaría que únicamente se deba modificar el código fuente generado en la actual aplicación de segmentación que utiliza OpenCV. Es de indicar que este tipo de segmentación es posible de aplicarla en imágenes en escala de grises así como en imágenes a color, lo que probablemente podría mejorar los resultados según al tipo de imágenes a las que se le aplique.

Finalmente recordar que este algoritmo es costoso computacionalmente, ya que requiere de grandes recursos para que su ejecución sea eficiente.

[19] Moda: es la observación que ocurre con mayor frecuencia en la muestra.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Los resultados de la segmentación basada en la detección de bordes a través del kernel Gaussiano y el operador Laplaciano son altamente dependientes de la desviación estándar elegida para el suavizado, por tanto, debemos ser cuidadosos con el valor que demos a este parámetro.

Por otro lado, se comprobó que al calcular el tamaño de la máscara con dependencia del valor de la desviación estándar, la segmentación arroja resultados óptimos; es así que, mientras mayor sea el valor de sigma, el tamaño de la máscara va tomando grandes dimensiones, hasta el punto que si el valor de sigma es la unidad, entonces el tamaño mínimo de la máscara será de 9x9. Esto conlleva a que el coste computacional sea muy elevado, porque con máscaras muy grandes tarda demasiado en calcular solamente un píxel. Por este motivo, OpenCV realiza el suavizado de la imagen mediante la descomposición de su función bidimensional en dos funciones unidimensionales, con lo que se logra equilibrar entre la aplicación de grandes máscaras de convolución y la reducción sustancial del coste computacional. Sin embargo, esto no quiere decir que al utilizar máscaras de tamaños independientes al valor de sigma no se puedan encontrar resultados óptimos.

En cuanto a la efectividad de la segmentación de las imágenes del corpus, podemos decir que los resultados difieren en cada imagen; en el caso de las imágenes de la categoría de los edificios, se obtuvieron resultados incrementalmente deficientes de acuerdo al grado de complejidad que presentaron, claro que existieron ciertas excepciones. Sin embargo es de anotar que en las imágenes de la categoría de los dinosaurios los resultados fueron muy buenos, por cuanto su grado de complejidad es bajo, ya que en una imagen existe un objeto en concreto a ser detectado, y esto facilitó la segmentación. Dicha complejidad complicó la detección de bordes, es decir, no se cerraron los contornos correspondientes a cada objeto, sino que en algunos casos se fusionaron dos o más objetos dentro de un contorno, que a la larga fue detectado como si fuese un solo objeto, teniendo como producto final, una segmentación deficiente.

Finalmente, debemos apuntar que existen técnicas de segmentación que podrían mejorar sustancialmente los resultados de esta tesis, sin embargo, la técnica que hemos implementado ha dado resultados satisfactorios, considerando las limitaciones que tiene, pero podría ser más eficiente al aplicar en imágenes más sencillas que las que hemos utilizado.

RECOMENDACIONES

Se recomienda en esta técnica usar imágenes que contengan objetos bien resaltados, es decir; que los objetos tengan colores más intensos que el fondo de la imagen, para que la detección de bordes sea más efectiva y por ende la segmentación sea mucho mejor.

Se recomienda buscar una alternativa de segmentación basada en la detección de contornos, que excluya los sub-contornos encontrados en un objeto, ya que al lograrlo, se podrían tener mejores resultados.

Si se quiere mejorar la detección de bordes, se podría utilizar el método de Canny en sustitución del operador Laplaciano.

Para mejorar el suavizado de la imagen se podría reemplazar el filtro Gaussiano por el filtro de media que es más efectivo.

BIBLIOGRAFÍA

[1] GONZÁLEZ AGUILAR, Diego, "Procesamiento de imágenes", Departamento de Ingeniería Cartográfica y del Terreno, Universidad de Salamanca, España. Recuperado 18-07-2011.
Web: http://212.128.130.23/eduCommons/ensenanzas-tecnicas/procesamiento-avanzado-de-imagenes-digitales/contenidos/Tema2.pdf

[2] FERNÁNDEZ GARCÍA, Nicolás Luis, "Contribución al Reconocimiento de Objetos 2D mediante Detección de Bordes en Imágenes en Color", Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Córdoba - España, mayo 2002. Recuperado 18-07-2011.
Web: http://oa.upm.es/557/1/NICOLAS_FERNANDEZ_GARCIA.pdf

[3] SÁNCHEZ, Omar, "Los Filtros espaciales y Frecuenciales", Universidad de Huelva, Huelva-España. Recuperado 19-07-2011.
Web: http://omarsanchez.net/filtroespa.aspx

[4] ZAPOTECAS MARTÍNEZ, Saíl, "Sistema de seguimiento utilizando lógica no clásica como medida de decisión n, implementado en un sistema operativo de tiempo real", Benemérita Universidad de Puebla, Puebla – México, Mayo 2005. Recuperado 20-07-2011.
Web: http://perseo.cs.buap.mx/bellatrix/tesis/TES826.pdf

[5] ALBA, José Luis, "Nombre del tema", Universidad de Vigo Jesús Cid, Universidad Carlos III, Inmaculada Mora, Universidad Rey Juan Carlos, Madrid – España. Recuperado 21-07-2011.
Web: http://www.gts.tsc.uvigo.es/pi/Analisis%20de%20imagenes.pdf

[6] RESTREPO, Alejandro, y otros, "Método Semiautomático para la Caracterización del Hormigón empleando procesamiento Digital de Imágenes", Escuela de Sistemas,

Facultad de Minas, Universidad Nacional de Colombia, Facultad de ingeniería, Universidad de Antioquia. Colombia 2001. Recuperado 22-07-2011.
Web: http://www2.unalmed.edu.co/dyna2005/133/metodo.pdf

[7] BRANCH, John William y OLAGUE, Gustavo, "La Visión por Computador una Aproximación al Estado de Arte", Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia, Medellín, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California - México. Recuperado 25-07-2011.
Web: http://www2.unalmed.edu.co/dyna2005/133/vision.pdf

[8] "Visión Artificial", Universidad Rey Juan Carlos, Madrid – España. Recuperado 26-06-2011.
Web: http://www.gavab.etsii.urjc.es/wiki/download/va/temario/tema3.pdf

[9] FERNÁNDEZ ASUNCIÓN, Oscar, "nombre del tema" Universidad Nacional de Trujillo. Perú, 13 de abril de 2009. Recuperado 28-07-2011.
Web: http://oefa.blogspot.com/2009/04/deteccion-de-bordes-algoritmo-de-canny.html

[10] GIL, P. y otros, "Ingeniería de Sistemas y Teoría de la Señal", Grupo de Automática, Robótica y Visión Artificial, Universidad de Alicante. Recuperado 29-07-2011.
Web: http://www.cea-ifac.es/actividades/jornadas/XXV/documentos/19-abtsufilva.pdf

[11] VALDIVIESO MIRANDA, Félix, "Reconocimiento y Registro 3d de Objetos Conocidos en una Escena". Universidad de Zaragoza. Septiembre 2010. Recuperado 1-08-2011.

[12] MESTRE, Gelys, y otros, " Estudio Comparativo del Desempeño de un Correlador Digital de Imágenes de Huellas Dactilares Pre – procesadas", Laboratorio de Óptica e Informática, Universidad Popular del Cesar, Revista Colombiana de Física, VOL. 35, No. 1, 2003. Recuperado 1-08-2011.

Web: http://calima.univalle.edu.co/revista/vol35_1/articulos/pdf/3501156.pdf

[13] MONTOYA, Jaime, “Discriminante o Hessiano (Matriz Hessiana)”, Santa Ana, El Salvador, 4 de julio de 2008. Recuperado 2-08-2011.
Web: http://www.jaimemontoya.com/discriminanteohessiano1.php

[14] ING. SÁNCHEZ RANGEL, Francisco Javier, “Medición y Análisis de las Variaciones en el Nivel de un Modelo Físico Empleando Imágenes”, Universidad Autónoma Metropolitana, 2010. Recuperado 2-08-2011.

[15] ARTIEDA TRIGUEROS, José, “SLAM Monocular”, 2007. Recuperado 3-08-2011.
Web: http://www.diphernet.com/web/cvision/dea_j/deaweb.html

[16] BALLESTA, Mónica, y otros, “Evaluación de Detectores de Puntos de Interés para Slam Visual”, Universidad Miguel Hernández, 2007. Recuperado 3-08-2011.
Web:http://www.tsplabs.net/actividades/jornadas/XXVIII/documentos/1262-JA-2007%20MarcasVisuales.pdf

[17] CASTELLO MARTÍNEZ, Vicente, “Localización y Decodificación de Códigos de Barras en Imágenes Digitales”, Universitat Jaume I, Junio-2005. Recuperado 4-08-2011.
Web: http://www3.uji.es/~vtraver/e80/E80_Vicente_Castello.pdf

[18] “Reconocimiento de objetos mediante descriptores locales”, 2009 y 2011. Recuperado 5-08-2011.

[19] VALVERDE REBAZA, Jorge, “Detección de bordes mediante el algoritmo de Canny”, Universidad Nacional de Trujillo, 2002. Recuperado 8-08-2011.

[20] “Capitulo 2: Marco Teórico”. Recuperado 8-08-2011.
Web: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/garcia_a_jl/capitulo2.pdf

[21] LINDEBERG, Tony, “SCALE-SPACE THEORY IN COMPUTER”, Royal Institute of Technology Stockholm, Sweden, 1994. Recuperado 9-08-2011.
Web:
http://books.google.com.ec/books?hl=es&lr=&id=u21OX41EbVkC&oi=fnd&pg=PA1&dq=GREY+%E2%80%93+LEVEL+BLOBS+tony+lindeberg&ots=cD9nhxrROg&sig=qFBLcJS_TNxP_QdoYorxyJZcT-w#v=onepage&q&f=false

[22] “Blob detection”. Recuperado 10-08-2011.
Web: http://en.wikipedia.org/wiki/Blob_detection

[23] UNAMDSP, “Espacio-Escala en Imágenes”, Procesamiento de Señales, marzo-28-2011. Recuperado 10-08-2011.
http://fierdetregauche.wordpress.com/2011/03/28/espacio-escala-en-imagenes/

[24] SUCAR, L. Enrique, “Visión por Computacional”, Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla México. Recuperado 11-08-2011.
Web: http://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf

[25] RODRIGUEZ POVEDANO, Juan. Recuperado 27-10-2011.
Web: http://e-archivo.uc3m.es/bitstream/10016/11180/1/PDF_Juan_Rodriguez_Povedano.pdf

[26] PASSARIELLO, Gianfranco y MORA, Fernando, “Imágenes Médicas: Adquisición, Análisis”, Procesamiento e Interpretación” Universidad Simón Bolívar. Venezuela 1995. Recuperado 12-10-2011.
Web:http://books.google.com.ec/books?id=Ah-qmV3wrs0C&pg=PA84&lpg=PA84&dq=procedimiento+m%C3%A1scara+convolucion+horizontal+vertical&source=bl&ots=MO7pNh0Ojq&sig=8EYV83lunlgqbHEJB0q68TNLu3c&hl=es&ei=tud3Tp-

jEeTr0gHF5PnmCw&sa=X&oi=book_result&ct=result&resnum=3&ved=0CCEQ6AEwAjgK#v=onepage&q&f=false

[27] MACHARE GALINDO, Juan Antonio. “Proyecto de investigación PDS – UNMSM”. Recuperado 15-10-2011.
Web: http://machare03190163.blogspot.com/

[28] GONZÁLEZ, Yolanda, “Operadores de Imagen”, Universidad de les Illes Balears, Islas Baleares - España. Recuperado 15-08-2011.
Web:
http://dmi.uib.es/~ygonzalez/VI/Material_del_Curso/Teoria/Tema3_Operadores.pdf

[29] “Detección de Bordes”. Recuperado 16-08-2011.
Web: http://members.fortunecity.es/davidweb2/visart/bordes.htm

[30] “Sobel operator”. Recuperado 17-08-2011.
Web: http://en.wikipedia.org/wiki/Sobel_operator

[31] LOWE, David, “Sift KeyPoint Detector”, University of British Columbia Vancouver, B.C., V6T 1Z4, Canada, Version 4, July 2005. Recuperado 10-11-2011.
Web: http://www.cs.ubc.ca/~lowe/keypoints/.

[32] DE LA LLANA ALAMAR, Sergio, y otros, “Introducción a Librerías OpenCV”. Recuperado 18-11-2011.
Web: http://web-sisop.disca.upv.es/imd/cursosAnteriors/2k3-2k4/copiaTreballs/serdelal/trabajoIMD.xml

[33] RODRÍGUEZ PÉREZ, Sergio, “II31 Sistemas Informáticos”, Castellón, Curso 2008-2009. Recuperado 12-12-2011.
Web: http://www.vision.uji.es/~montoliu/docs/pfc/SergioRodriguez.pdf

[34] MOLINA, Juan José Mateo “Dibujos para Colorear”. Recuperado 16-12-2011.
Web: http://www.indicedepaginas.com/dibujos.html

[35] RODRÍGUEZ ARAUJO, Jorge, “Introducción a los Sistemas De Visión Artificial”, 11 de agosto de 2011. Recuperado 15-11-2011.
Web: http://es.scribd.com/doc/8343510/Vision-Artificial

[36] ING. CALDERÓN, Francisco Carlos, “OpenCV”, Pontificia Universidad Javeriana. Recuperado 16-12-2011.
Web: http://opencvjaveriana.wikispaces.com/

[37] DUYGULU, P., BARNARD, K., DE FREITAS, N., y FORSYTH, D. Object Recognition as machine translation: Learning a lexicon for afixed image vocabulary”, In ECCV’02, pages 97-112, 2002. Recuperado 5-01-2012.

[38] SHAPIRO, Linda y STOCKMAN, George, “Computer Vision”, 2003. Recuperado 12-01-2012.

[39] VILÁ UBIETO, Karen “Reconstrucción 3d de Modelos utilizando Técnicas de Visión Artificial”, Escuela Técnica Superior de Ingeniería (icai). Recuperado 24-01-2012.

[40] “Parámetro estadístico”. Recuperado 21-12-2012.
Web: http://www.esacademic.com/dic.nsf/eswiki/900126

[41] PLATERO, Carlos, “Técnicas de Prepocesado”, Dpto. Electrónica, Automática e Informática Industrial. Recuperado 16-01-2012.
Web:
http://www.elai.upm.es:8009/spain/Asignaturas/Robotica/ApuntesVA/cap4Procesadov1.pdf

[42] “The Gaussian kernel” 03Gaussiankernel.nb. Recuperado 15-01-2012.
Web:
http://www.stat.wisc.edu/~mchung/teaching/MIA/reading/diffusion.gaussian.kernel.pdf

[43] “Procesamiento de imágenes”. Recuperado 18-01-2012.
Web: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/ramos_r_m/capitulo3.pdf

[44] “Filtrado de Imágenes”, Tema5. Recuperado 19-01-2012.
Web: http://dmi.uib.es/~ygonzalez/VI/Material_del_Curso/Teoria/Tema5_Filtrado.pdf

[45] ELGAMMAL, Ahmed “Filters” Dept. of Computer Science, Rutgers University, Spring 2008. Recuperado 20-9-2012.
Web: http://www.cs.rutgers.edu/~elgammal/classes/cs334/Filters.pdf

[46] MONLLOR MONTOYO, Fernando M. “Análisis, diseño e implementación de un dispositivo gestual”, Departamento de la Ciencia de la Computación e Inteligencia Artificial, Junio 2004. Recuperado 21-9-2012.
Web: http://www.alu.ua.es/f/fmmm/down/documento_dispositivo_gestual.pdf

[47] MUÑOZ, Alejandro, “Detección de Patología Celulares Empleando Visión Artificial”, Grupo de Percepción y Control Inteligente, Universidad Nacional de Colombia, Manizales, septiembre 2001. Recuperado 26-01-2012.
http://www2.unalmed.edu.co/dyna2005/133/deteccion.pdf

[48] “Detección de Bordes”, Universidad Miguel Hernández. Recuperado 30-10-2012.
Web: http://isa.umh.es/asignaturas/rvc/Tema%205.%20Deteccion%20de%20bordes.pdf.

[49] “Procesamiento de Imágenes”, CAPITULO 6 (ILWIS USER’S GUIDE). Recuperado 12-9-2012.

http://www.itc.nl/external/unesco-rapca/Casos%20de%20estudios%20SIG/02%20Sensores%20remotos/sensores%20remotos.pdf

[50] “Fred's ImageMagick Scripts”. Recuperado 12-9-2012. http://www.fmwconcepts.com/imagemagick/laplacian/index.php

[51] ING. JUÁRES SANDOVAL, Oswaldo, “Análisis de Filtros Laplaciano, Gradiente Sobel, Gradiente Roberts y Suavización de Imágenes para Imágenes en escala de Grises”, Col. San Francisco Culhuacán C.P. 04430. Recuperado 20-10-2012. Web: http://whueya.files.wordpress.com/2011/10/anc3a1lisis-de-filtros-laplaciano.pdf

[52] PASSARIELLO, Gianfranco, “Imágenes médicas Adquisición, Análisis”, Venezuela, marzo de 1995. Recuperado 29-10-2012. Web: http://books.google.com.ec/books?id=Ah-qmV3wrs0C&pg=PA91&lpg=PA91&dq=an%C3%A1lisis+del+operador+laplaciano&source=bl&ots=MO8pOf7Onr&sig=gF6fLWB7-PddZyyBywkqCBhCNrs&hl=es&sa=X&ei=N9IRT5uPHtOUtwf6xYiZAg&ved=0CEIQ6AEwBQ#v=onepage&q&f=false.

ANEXOS

Anexo 1: Instalación de OpenCV 2.3.1 para Microsoft Visual Studio 2010

Instalación OpenCV

La instalación consta de varios pasos que se detallarán a lo largo de este anexo, los mismos que permitirán el correcto funcionamiento de OpenCV en la plataforma de desarrollo Microsoft Visual Studio 2010.

El instalador de OpenCV lo podemos encontrar en el sitio en la siguiente dirección web: http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.3.1/.

Pasos:

1. Se ejecuta el instalador OpenCV-2.3.2-GPU-demo-pack-win32.exe, se presenta una pantalla en la cual debemos colocar la dirección en la cual deseamos descargar los archivos que contiene el instalador.

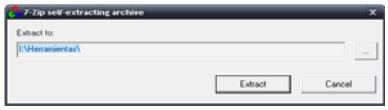


Figura A1.1: Diálogo que permite agregar la dirección de almacenamiento de OpenCV.

2. Se elige la carpeta o unidad en la cual se van a extraer los archivos de OpenCV, para facilitar la manipulación de los archivos se recomienda realizar la extracción de OpenCV en la unidad C:\.

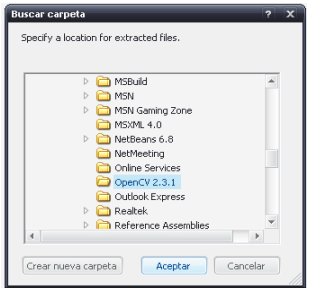


Figura A1.2: Buscador de carpetas o directorios.

3. Una vez elegido el destino en el cual se va a guardar los archivos de OpenCV damos clic en la opción “Extract” y los archivos comenzarán a extraerse, como podemos ver en la figura A1.3.

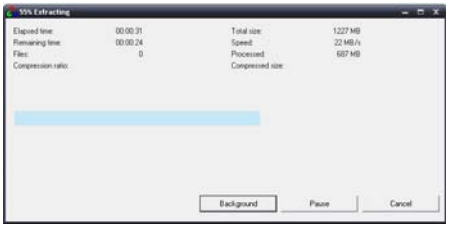


Figura A1.3: Proceso de Extracción de los archivos OpenCV.

4. Los archivos extraídos se visualizaran como se muestra en la figura A1.4.

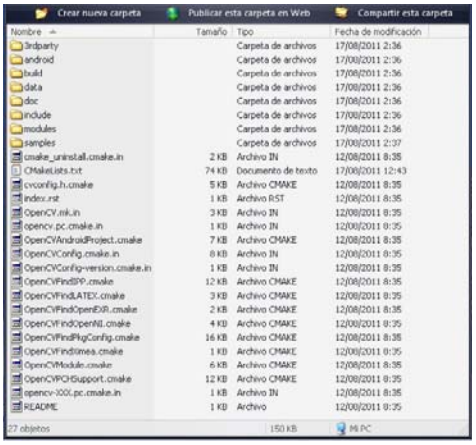


Figura A1.4: Archivos que posee la herramienta OpenCV.

Vinculación de Visual Studio con OpenCV 2.3.1

Ya con los archivos extraídos de OpenCV debemos vincularlos con Microsoft Visual Studio 2010, para ello realizaremos los siguientes pasos:

1. Ingresamos a la aplicación de Microsoft Visual Studio 2010

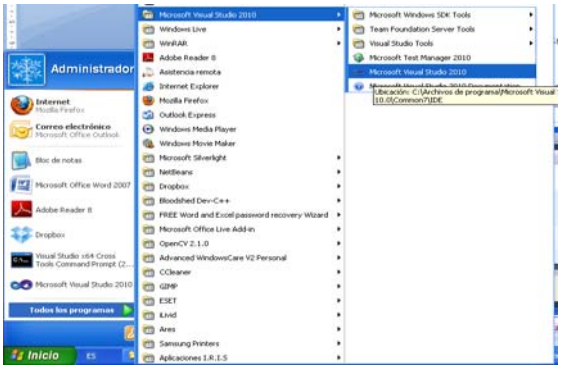


Figura A1.5: Proceso para ingresar a la aplicación Microsoft Visual Studio 2010.

2. Una vez ingresado a Microsoft Visual Studio 2010 es necesario crear un proyecto para vincular con OpenCV, el proyecto a crear debe ser creado en el lenguaje C++ para ello nos ubicamos en el menú “File” opción “New” → “Proyect”.

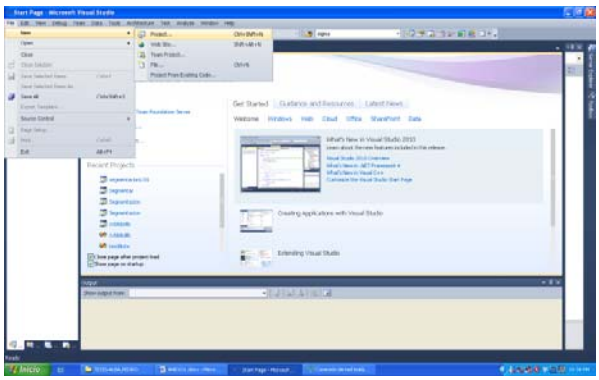


Figura A1.6: Pasos para crear un proyecto en Microsoft Visual Studio 2010.

3. Posteriormente visualizaremos un cuadro de dialogo en el cual elegimos el tipo de proyecto, en nuestro caso Win32 Console Application y el lenguaje de programación a utilizar será C++. En el mismo cuadro de dialogo se ingresa el nombre del proyecto y la ubicación en la que se guardara el mismo.

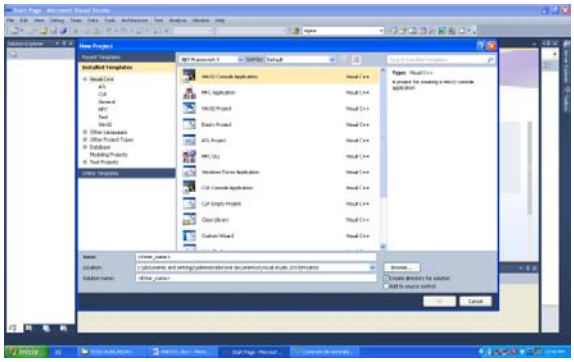


Figura A1.7: Pantalla de configuraciones para crear el proyecto.

4. Una vez realizado los pasos anteriormente descritos se visualiza el proyecto creado en Microsoft Visual Studio 2010 como lo podemos observar en la figura A1.8:

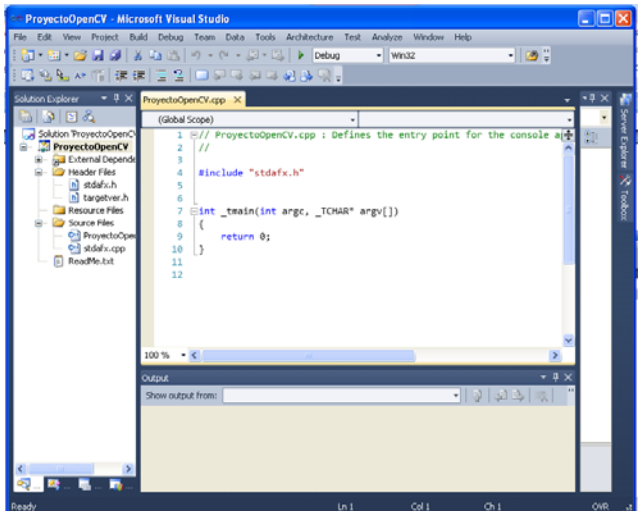


Figura A1.8: Aplicación Microsoft Visual Studio 2010.

Una vez creado el proyecto se procede a configurar OpenCV, tarea que se realiza de la siguiente manera:

a) Ingresamos a "Proyect" → "Properties" o Alt+F7.

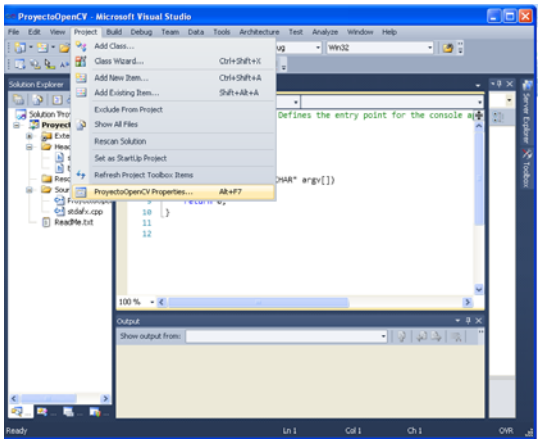


Figura A1.9: Pasos para ingresar a la propiedades del proyecto.

b) Al ingresar al "Properties" se presenta una ventana que permitirá realizar las respectivas configuraciones, para ello escogemos "Configuration Properties"

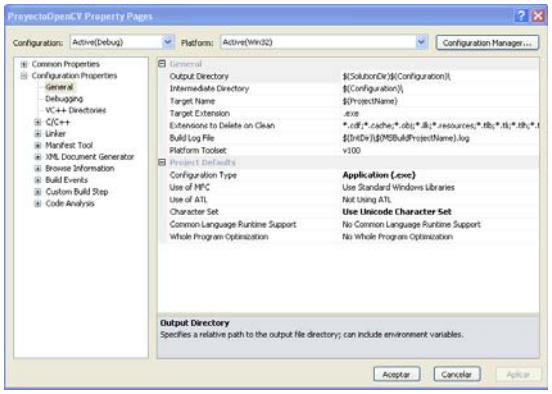


Figura A1.8: Pantalla de configuraciones.

c) Como siguiente paso se elige “C/C++” → “General” → “Additional Include Directories”.

En “Additional Include Directories” se agregan los directorios de OpenCV que se presentan a continuación:

C:\opencv\build\include\opencv
C:\opencv\build\include\opencv2
C:\opencv\modules\imgproc\src

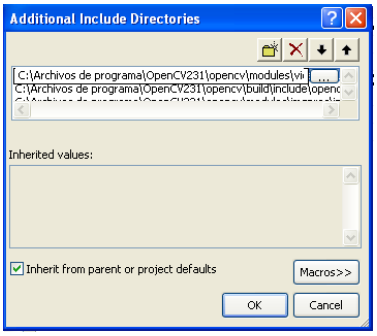


Figura A1.9: Ventana para agregar los directorios

d) De igual forma, se agregan las direcciones de las librerías. Esto se realiza ingresando a: “Linker” → “General” → “Additional Library Directories.”

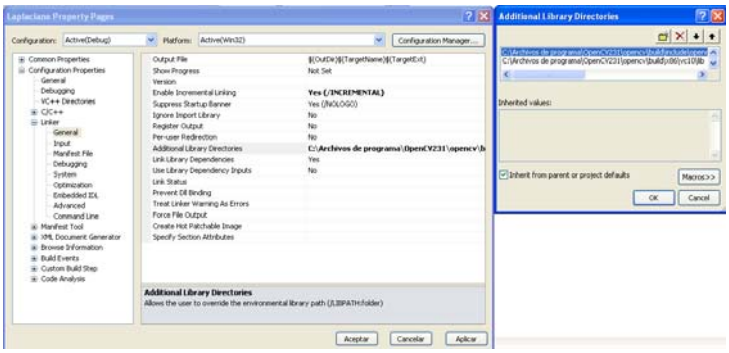


Figura A1.10: Ventana que permite agregar los directorios de las librerías.

e) En la misma opción “Linker” se deben agregar las librerías, para ello elegimos “Input” → “Additional Dependencies”.

En “Additional Dependencies” se agregan las siguientes librerías:

opencv_calib3d231d.lib
opencv_contrib231d.lib
opencv_core231d.lib
opencv_features2d231d.lib
opencv_flann231d.lib
opencv_gpu231d.lib
opencv_haartraining_engined.lib
opencv_highgui231d.lib
opencv_imgproc231d.lib
opencv_legacy231d.lib
opencv_ml231d.lib
opencv_objdetect231d.lib
opencv_ts231d.lib
opencv_video231d.lib

Estas librerías se encuentran ubicadas en el siguiente directorio:

C:\opencv\build\x86\vc10\lib

Una vez ubicadas las librerías se procede a agregarlas como se puede observar en la figura A1.10.

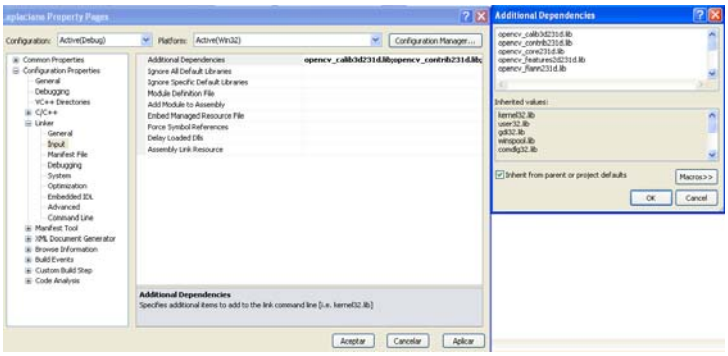


Figura A1.11: Ventanas para agregar las librerías de OpenCV.

Actualización de la librería TBB y crear la variable de entorno

La librería TBB (Threading Building Blocks)[20] sirve para realizar el balanceo de forma homogénea en todos los núcleos del procesador

En la herramienta OpenCV viene incluida la librería TBB, sin embargo es necesaria su actualización puesto que la misma no posee el directorio "bin", ocasionando problemas en la ejecución de OpenCV. Actualizando la librería TBB garantizamos el correcto funcionamiento de OpenCV al momento usarlo en el proyecto.

La librería actualizada la podemos descargar de la página Web: http://threadingbuildingblocks.org/ver.php?fid=171.

La actualización se logra mediante el siguiente proceso:

1. La librería TBB que viene por defecto en OpenCV se encuentra en el siguiente directorio: C:\opencv\build\common. Es allí en donde se realiza la actualización del mismo.

[20] TBB: Creación de bloques a través de hilos.

Una vez ubicada e identificada la librería procedemos a cambiar de nombre de TBB a TBB_Old, en el mismo directorio agregamos la librería actualizada que nos descargamos.

La librería TBB actualizada posee los siguientes directorios:

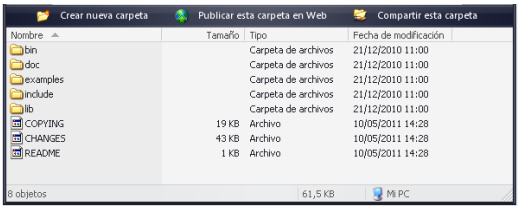


Figura A1.12: TBB nuevo y completo.

Después de realizar la actualización de la librería TBB se procede a agregarla en la aplicación Microsoft Visual Studio 2010.

Esto se realiza seleccionando del menú “Proyect” la opción “Properties”, y de la ventana que se visualiza seleccionamos la opción “Additional Library Directorio”, a través del menu: Linker → General → Additional Library Directories.

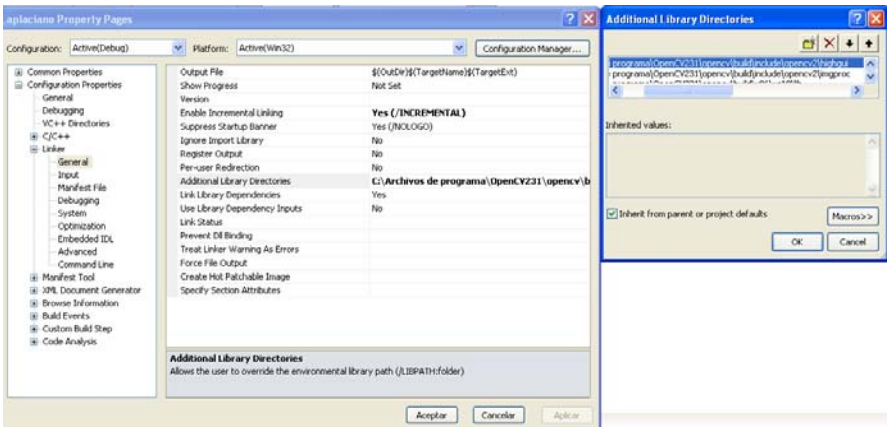


Figura A1.13: Captura de la pantalla para agregar la librería TBB.

Creando variable de entorno TBB y OpenCV.

Para utilizar las librerías TBB y OpenCV se procede a crear una variable de entorno, para lo cual realizamos los siguientes pasos:

1. Ingresamos a Mi PC → Propiedades.

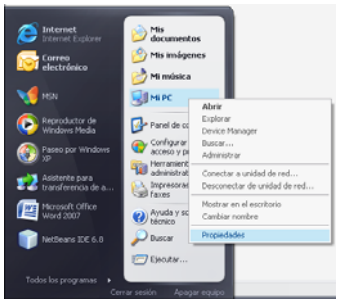


Figura A1.14: Captura de la pantalla para ingresar a propiedades de MI PC.

2. En la ventana “Propiedades del Sistema” seleccionamos “Opciones Avanzadas”, de la cual elegimos la opción “Variables de Entorno”.

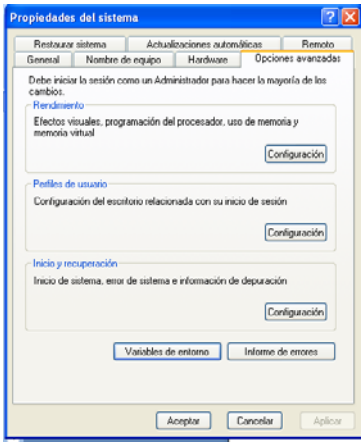


Figura A1.15: Pantalla de “Propiedades del Sistema”.

3. Una vez visualizada la ventana de la “Variable de Entorno” procedemos a la creación de la variable de entorno, para ello se agregan en la variable PATH las direcciones donde se encuentran ubicadas las librerías TBB y OpenCV:

C:\opencv\build\common\tbb\lib\ia32\vc10
C:\opencv\build\x86\vc10\lib

Figura A1.16: Pantalla de Variables de entorno.