SEPARACIÓN DEL PLANO DE TEXTO EN IMÁGENES GRÁFICAS EN 256 TONOS DE GRIS

POR

LUIS GERARDO DE LA FRAGA

INGENIERO INDUSTRIAL EN ELECTRÓNICA

INSTITUTO TECNOLÓGICO DE VERACRUZ, 1991

TESIS SOMETIDA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS EN LA ESPECIALIDAD DE ELECTRÓNICA EN EL INSTITUTO NACIONAL DE ASTROFÍSICA, ÓPTICA Y ELECTRÓNICA.

SUPERVISADA POR: DR. JORGE MARTÍNEZ CARBALLIDO

DEPARTAMENTO DE: ELECTRÓNICA

TONANTZINTLA, PUE.

1994

SEPARACIÓN DEL PLANO DE TEXTO EN IMÁGENES GRÁFICAS EN 256 TONOS DE GRIS

LUIS GERARDO DE LA FRAGA

ELECTRÓNICA

INSTITUTO NACIONAL DE ASTROFÍSICA, ÓPTICA Y ELECTRÓNICA

1994

SUMARIO

La extracción de caracteres en imágenes de documentos con escala de grises que tienen como fondo pinturas, sombras, áreas brillantes, manchas y zonas borrosas, es una operación frecuente pero que resulta crítica en el análisis de imágenes, el reconocimiento óptico de caracteres (OCR, por sus siglas en inglés), en transmisión de imágenes y sistemas de videoconferencia.

En este trabajo se presentan cuatro algoritmos que ayudan a resolver el problema arriba expuesto. Se revisan dos algoritmos ya publicados, uno de estos se mejora. Se sugiera un esquema diferente para otro algoritmo ya propuesto. Otro algoritmo es nuevo y es apropiado para automatizarse.

Para la verificación de los algoritmos, estos se programaron en lenguaje C. Se presentan las imágenes binarias, resultantes de aplicar el algoritmo clásico se segmentación por nivel de umbral global y los cuatro algoritmos revisados, a cuatro imágenes originales que presentan mala calidad. Se realiza una prueba estadística de sus efectividades con un programa de OCR comercial.

A mi abuela

Cecilia de la Fraga

Agradecimientos:

Al INAOE por ser la institución donde logré la maestría

A CONACyT por brindarme el apoyo económico sin el cual hubiese sido imposible continuar estos estudios.

Al Dr. Jorge Martínez Carballido por la primera revisión de este trabajo

A mis asesores M.C. Arturo Díaz Pérez y M.C. Daniel Ortíz Arroyo por los comentarios y sugerencias para la elaboración de este trabajo.

A Jorge Torres por la revisión del manejador de memoria usado en el programa desplegador de imágenes.

A Carlos Escamilla por ayudarme en el trabajo fotográfico para la exposición de este trabajo.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

INTRODUCCIÓN

1.1 Motivación

Los documentos son el medio de información primario en la sociedad informática de hoy. Los símbolos comunes que representan información sobre documentos son escritos a mano o con máquinas que imprimen caracteres y gráficos. Con la disponibilidad de las computadoras, terminales de video, discos ópticos, máquinas de fax, impresoras láser y scanners, estos símbolos pueden ser generados, transmitidos, accesados, almacenados, desplegados, impresos y reabsorbidos automáticamente. Como sustituto al manejo manual, los sistemas de análisis de imágenes de documentos pueden reconocer los símbolos sobre los documentos y organizarlos en otro tipo de archivo de datos con los cuales pueden manipularse con varios procesadores de palabras y programas capaces de dejar gráficos o con bases de datos. Varios de estos sistemas han aparecido recientemente [8, 21] o han sido propuestos en revistas académicas o bien en el mercado comercial

Los sistemas van desde paquetes de programas para reconocimiento de caracteres, llamadas comúnmente OCR's (Optical Character Recognition), para PC (Personal Computer) [21] hasta sistemas especiales que incluyen reconocimiento automático de patrones [20] y sistemas gráficos de vectorización para dibujos en ingeniería [8].

La aplicación de tales sistemas están grandemente limitada por sus requerimientos de documentos con buena calidad. En algunas ocasiones, documentos con esas características de buena calidad o alto contraste no se encuentran en las aplicaciones reales. Algunos problemas que contribuyen a tener una calidad pobre en las imágenes de documentos son:

- Caracteres brillantes en varios colores.
- Caracteres manchados o borrosos.
- Cambios no uniformes en los colores del documento original dado su gran tiempo de almacenamiento.
- El documento tiene una calidad pobre de escritura o impresión.
- Se presentan sombras generadas por las condiciones de iluminación escasa cuando la imagen fue capturada.
- Existe un solo documento original deteriorado imposible de reproducir.
- Se tienen textos combinados con fondos uniformes de patrones geométricos o con fondos no uniformes de pinturas, tales como formas oficiales, cheques bancarios, pagarés, etc.

Dada las grandes variaciones que se tienen en los casos arriba expuestos, aunándose a la gran variedad de formas de los caracteres de texto que se tienen que reconocer: distintos tipos, tamaños y colores, todavía se requiere mucho esfuerzo para encontrar soluciones generales a estos problemas.

1.2 Objetivo

El objetivo del trabajo de esta tesis consiste en buscar métodos adecuados para extraer las formas de los caracteres en imágenes de documentos que presentan uno, o más, de los problemas expuestos en la sección 1.1.

Estos métodos permitirán hacer un reconocimiento de caracteres más eficiente que el logrado con la imagen original.

1.3 Organización de la tesis

Para entender los términos que se manejan y conceptuar el trabajo, en el Capítulo 2 se da una introducción básica al procesamiento digital de imágenes: la definición de imagen, los pasos que involucra un trabajo de procesamiento de imágenes , los dispositivos y programas requeridos. También en el Capítulo 2 se sitúa el trabajo dentro del área del procesamiento digital de imágenes y los dispositivos y programas usados. En el Capítulo 3 se repasan algunas técnicas del procesamiento digital de imágenes que ayudarán a resolver el problema; debe ponerse especial atención a este capítulo, da la teoría del filtrado espacial y es la base de los cuatro algoritmos desarrollados. Estos algoritmos se describen en detalle en el Capítulo 4. Los algoritmos se consideran rápidos y no se involucra la reconstrucción o restauración de las imágenes. La evaluación de los algoritmos se presentan en el Capítulo 5. Finalmente las conclusiones y contribuciones de este trabajo se encuentran en el Capítulo 6.

Este trabajo no pretende realizar una utilería de cómputo específica. Los algoritmos propuestos solo se programaron para verificar su efectividad. Aún así de considera que conforman un soporte para futuros trabajos en esta área y una recopilación de ellos de resume en los Apéndices I, II y III.

Capítulo 2

CONCEPTOS GENERALES

El trabajo de esta tesis involucra el procesamiento digital de imágenes.

El término procesamiento digital de imágenes versa sobre la manipulación y análisis de imágenes versa sobre la manipulación y análisis de imágenes por computadora [19]. De aquí en adelante se usará el termino procesado de imágenes refiriéndose a procesamiento digital de imágenes.

El procesamiento de imagen puede considerarse como un tipo especial del procesamiento digital en dos dimensiones, el cual se usa para revelar información sobre imágenes y que involucra hardware, software y soporte teórico.

En la sección 2.1 se da la definición de imagen. En la sección 2.2 se describen las áreas en que se divide el procesamiento de imágenes vistas estas áreas como un modelo general del procesado de imagen. En la sección 2.3 se revisan los dispositivos y paquetes de cómputo necesarios que conforman un sistema para realizar procesado de imagen. En la sección 2.4 se concretiza el modelo de procesado usado en esta tesis y que define el trabajo desarrollado. En la sección 2.5 se numeran los dispositivos y programas del sistema de procesamiento utilizado.

2.1 Definición de una imagen digital

A continuación se definirá el término imagen correspondiente a una imagen monocromática y es algo diferente que la definición para una imagen en color, pero correspondiente a las imágenes que fueron usadas en el trabajo de tesis.

El término imagen se refiere a una función bidimensional de intensidad

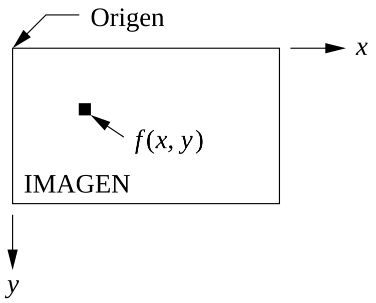

Figura 2.1: Convención de ejes usada

de luz $f(x, y)$, donde x y y denotan las coordenadas espaciales y el valor de f en cualquier punto (x, y) es proporcional al brillo (o nivel de gris) de la imagen en ese punto [19]. La Fig. 2.1 muestra la convención de coordenadas que se usará.

Una imagen digital es una imagen $f(x, y)$ que ha sido discretizada en coordenadas espaciales y en brillo. Una imagen digital puede considerarse como una matriz cuyos índices del renglón y columna identifican un punto en la imagen y el correspondiente valor del elemento de la matriz que identifica el nivel de intensidad de luz en ese punto. Los elementos de tal arreglo digital son llamados elementos de imagen, elementos de pintura, pixeles o pels (estos dos últimos son abreviaturas del inglés "picture elements").

Para trabajar con números en la computadora, el nivel de brillo, o valor de cada pixel, es cuantizado a códigos binarios enteros positivos (el brillo no puede ser negativo). El número de niveles de cuantización está determinado por la relación.

$$L = 2^B \tag{2.1}$$

donde B representa el número de bits necesarios para cada muestra. Esto es, con 5 bits de cuantización en la intensidad se pueden representar 32 niveles de gris (2=32). Para tener una imagen de buena calidad para el ojo humano es necesario tener como mínimo 64 niveles de gris [13, p. 162]. Para una apreciación fina se usa una cuantización estándar a 8 bits, esto es, 256 niveles de gris. A una imagen cuantizada de esta manera se le conoce como imagen en tonos de gris.

La cuantización a 8 bits (1 byte) se dice estándar porque es el mínimo

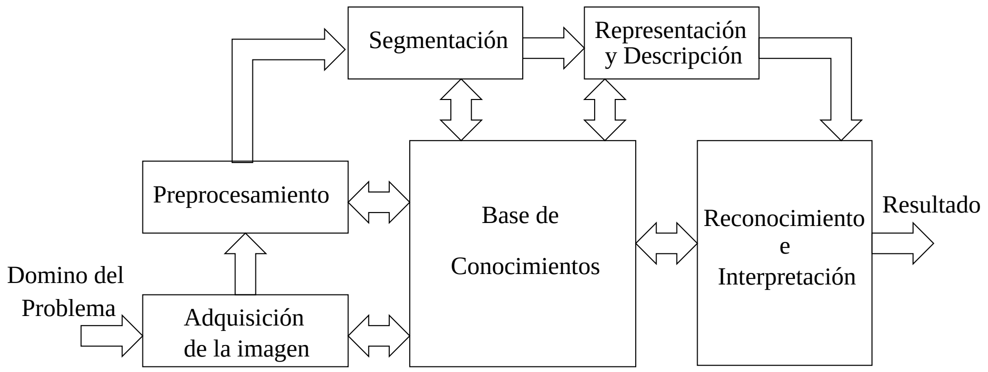

Figura 2.2: Pasos fundamentales del procesamiento de imágenes

número direccionable directamente por la mayoría de los microprocesadores.

Una imagen binaria es una imagen monocromática cuantizada a 1 bit por pixel, esto es, dos niveles, blanco ó negro

2.2 Modelo general para el procesamiento de imágenes

Gonzalez [15], modela los pasos fundamentales que se siguen en el procesado de imágenes como se representa en la Fig. 2.2

Para ayudar a una conceptualización sencilla de este tema, se ilustrará con un ejemplo: se quieren usar las técnicas del procesado de imágenes para leer automáticamente la dirección en sobres de correo. Nuestro dominio del problema (ver Fig. 2.2) en este ejemplo abarca los sobres del correo, y el objetivo es leer la dirección anotada sobre cada sobre. Entonces, la salida deseada en este caso es una serie de caracteres alfanuméricos.

El primer paso del proceso es la adquisición de la imagen, esto es, digitalizarla. Algunos dispositivos con que se puede realizar esto se tratarán en la sección siguiente (2.3).

Después de que la imagen digital ha sido obtenida, el siguiente paso es el preprocesamiento. Su función es mejorar la imagen de manera que se incremente la oportunidad de éxito de los siguientes procesos. El preprocesamiento.típicamente trata con técnicas para realizar el contraste y remover ruido.

El siguiente paso es la segmentación. Definida en forma amplia la segmentación particiona una imagen de entrada en sus partes constituyentes u objetos. Generalmente, la segmentación automática es una de las tareas más difíciles en el procesamiento de imágenes. En términos del reconocimiento de caracteres, el rol de la segmentación es extraer caracteres individuales y palabras del fondo de la imagen.

La descripción llamada también selección de características, trata con extracción de los rasgos que resulta en alguna información cuantitativa de interés o características que son básicas para diferenciar una clase de objetos con otra. En términos del ejemplo, para el reconocimiento de caracteres, descriptores tales como lagos (huecos) y bahías son rasgos poderosos que ayudan a diferenciar una parte del alfabeto de otra.

El último estado comprende al reconocimiento y la interpretación. El reconocimiento es el proceso que etiqueta, o asigna un nombre, a un objeto basándose en la información que proveen sus descriptores. La interpretación involucra la asignación de significado a un conjunto de objetos reconocido. En términos de nuestro ejemplo, identificar un carácter como, digamos, una c, requiere la asociación de los descriptores para este carácter con la etiqueta c.

La base de conocimientos mantiene todo el conocimiento que se tiene acerca del problema tratado. Este conocimiento puede ser simple, teniendo solo las regiones de la imagen donde la información de interés se conoce que está localizada, con lo que se limita la búsqueda que conduce a esa información. La base de conocimientos también puede ser compleja, tal como una lista interrelacionada de todos los defectos mayores posibles en un problema de inspección de materiales o una base de datos de imágenes conteniendo imágenes de satélite en alta resolución de una región en conexión con aplicaciones de detección de cambios. Además de guiar la operación de cada módulo del proceso, la base de conocimientos también controla la interacción entre módulos. Representando con flechas de doble punta en la Fig. no 2.2. Con esto se indica que la comunicación entre módulos del proceso generalmente está basada sobre conocimiento previo de que resultado se podría esperar. Por ejemplo, para que una máquina concluya que una cadena de caracteres es un código postal, el sistema debe estar dotado del conocimiento necesario para reconocer el significado de la localización de la cadena con respecto a otros componentes en un campo correspondiente a una dirección. Este conocimiento no sólo guía la operación de cada módulo, sino también ayuda en las operaciones de retroalimentación entre módulos a través de la base de conocimientos. Como en el caso, una cadena de números en la localización correcta pero que consisten de sólo cuatro caracteres (uno de los cuales pudo no haberse reconocido) podría guiar al módulo de interpretación que "sospeche" que dos caracteres están unidos. Una retroalimentación que mande un mensaje, a través de la base de conocimientos, a la etapa de segmentación para que "mire" otra vez, es un ejemplo de desempeño de tareas en el procesamiento de imágenes.

Es importante hacer notar que los resultados del procesamiento pueden verse en la salida de cualquier paso de la Fig. 2.2. También debe notarse que no todas las aplicaciones del procesado de imágenes requieren la complejidad de interacciones de la Fig. 2.2. Numerosas aplicaciones caen fuera de este esquema. De hecho, no todos los módulos, son siempre necesarios. Por ejemplo, el realce de imágenes para interpretación visual humana rara vez va más allá de la etapa de preprocesamiento. En general, las funciones de procesamiento que incluyen reconocimiento e interpretación están asociadas con aplicaciones de análisis de la imagen en la cual el objetivo es que se extraiga la información de una imagen en forma automática o semiautomática.

2.3 Elementos de los sistemas de procesamiento de imágenes

En la figura 2.3 se muestran los elementos de un sistema de uso general para realizar las operaciones de procesamiento de imágenes.

En la adquisición de imágenes deben existir dos elementos básicos. El primero es algún dispositivo básico que sea sensible a una determinada banda del espectro de energía electromagnético como son las bandas de rayos-x, el ultravioleta, el visible o el infrarrojo, y que produce una señal eléctrica proporcional al nivel de energía sensado. El segundo es el digitalizador que convierte la salida del dispositivo físico de sensado a forma digital.

En esta categoría se agrupan a las cámaras CCDs (Charge-Coupled Devices) que tiene la ventaja de la velocidad de capturación (hasta 1/10,000 seg.) pero un costo elevado, los scanners y cámaras de video.

El almacenamiento es un punto crítico debido a la gran cantidad de información usada. Por ejemplo, una imagen en 8 bits de tamaño 1024×1024 pixeles requieren un megabyte de espacio para su almacenamiento.

En el procesamiento, ya existen computadoras con microprocesadores especializados en procesamiento de imágenes que permiten un manejo rápido de las operaciones de matrices y acceso a memoria para aplicaciones de procesamiento de marcos (frames).

En cuanto al despliegue de las imágenes, se han usado los monitores de T.V. y monitores de computadoras. Los resultados desplegados en el monitor pueden ser fotografiados por una cámara enfocada a la cara del tubo de rayos catódicos o generar directamente una señal de video para grabarse.

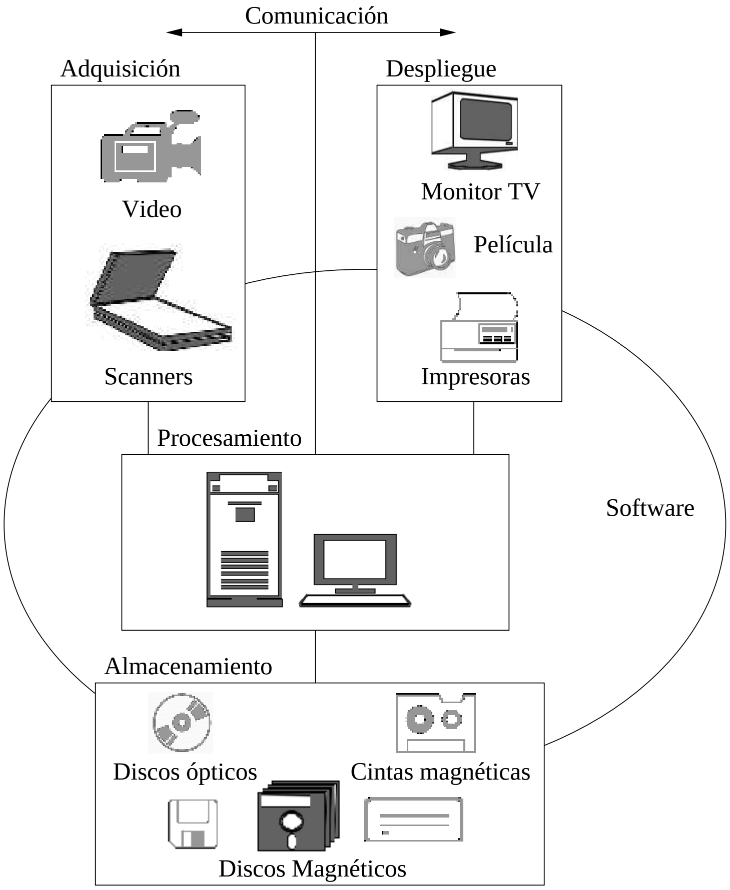

Figura 2.3: Elementos funcionales básicos de un sistema de procesamiento de imágenes: adquisición, almacenamiento, procesado, comunicaciones, despliegue y software. Dentro de cada caja se dan ejemplos de dispositivos usados en tales sistemas.

Para el caso del IBM PC (y compatibles), la salida para desplegado cuenta con dos componentes el adaptador de video (comúnmente conocido como la tarjeta de video) y el monitor de video. El "adaptador" es la tarjeta que se inserta dentro de la computadora y el "monitor" es la pantalla de video donde aparecen los caracteres y gráficos.

Las impresoras no manejan la misma amplitud de tonos de gris o colores que los monitores de computadoras o T.V., o de la película fotográfica, es mas, comúnmente manejan sólo dos tonos: blanco y negro. Es un área de aplicación el desarrollo de algoritmos que permitan representar los detalles visibles de un imagen en papel.

Como se representa en la Fig. 2.3, el fondo de todo el sistema involucra el software, desde el sistema operativo de la computadora, los programas que manejan cada periférico desde la computadora, y los programas de aplicación específicos realizados en algún lenguaje de programación. También existen periféricos especializados y programas ya desarrollados que pueden servir como punto de arranque para desarrollar nuevas aplicaciones.

2.4 Modelo de procesamiento utilizado

En la Fig. 2.4 se presenta el modelo usado que ubica el trabajo realizado dentro del esquema presentado en la sección 2.2.

El dominio del problema de este trabajo de tesis comprende imágenes de documento de mala calidad. Se tomará la consideración de que la mayoría de los documentos presentan los trazos de las letras más obscuros que el fondo. El problema del inciso a) de la sección 1.1, que comprende caracteres de distintos colores, no se contemplará, y los documentos que presenten los otros problemas (caracteres manchados; documentos con colores no uniformes dado su gran tiempo de almacenamiento, documentos con una calidad pobre de escritura o impresión, con sombras o con fondos de patrones geométricos o fondos de pinturas; y la posibilidad de un único documento original deteriorado) tendrán esa restricción de caracteres obscuros sobre un fondo más brillante.

El objetivo, ya expresado en la sección 1.2, es separar los trazos de los caracteres de esas imágenes de mala calidad para su posterior reconocimiento por un programa de reconocimiento de caracteres (OCR). A la salida del OCR tendremos un archivo de texto ASCII.

La imagen de entrada será cuantizada a 256 tonos de gris. Esto es en parte por el scanner [6] con que se cuenta, porque lo importante es el nivel de intensidades de la imagen de entrada y no los colores en ella y por las razones expuestas al final de la sección 2.1. La imagen que entrega el scanner

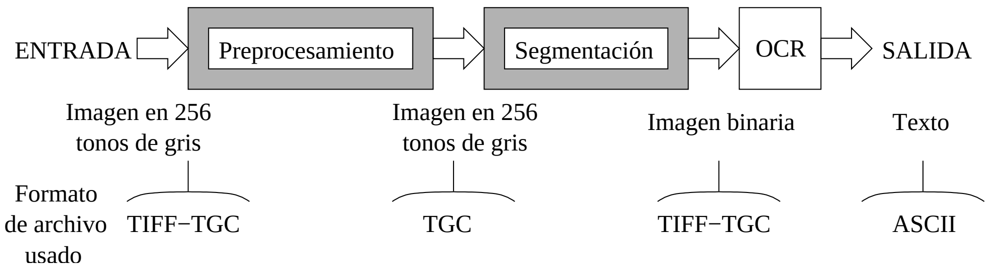

Figura 2.4: Modelo general del trabajo realizado. Los bloques sombreados indican los pasos de los algoritmos desarrollados

estará en un archivo como formato TIFF (para la definición de formato de archivo ver el Apéndice I).

El OCR utilizado en este trabajo es un paquete comercial [21] ya existente, por lo que los pasos de representación y descripción, y reconocimiento e interpretación ya están implementados dentro de él.

Los algoritmos que extraerán el plano de caracteres, entonces, consistirán de técnicas de preprocesamiento y segmentación para imágenes en 256 tonos de gris.

Dada la complejidad del formato TIFF [1], se decidió crear un formato propio para tratar las imágenes, llamado TGC se describe con detalle en el Apéndice 1.

Dado que el OCR ya es una sola entidad a la que no se puede controlar o accesar el código fuente, la posible retroalimentación cerrada queda excluida. También se carece, por ende, de una base de conocimientos. Solo existen ciertas especificaciones para el funcionamiento del OCR [21]:

- Su entrada deben ser imágenes binarias en formato TIFF, por lo que es necesario convertir la imagen binaria TGC de de salida de la etapa de segmentación, como se ve en la Fig.2.2, a una imagen TIFF.

- Reconoce caracteres de tamaño mínimo de 6 puntos a un máximo de 24 puntos. Por comparación, este párrafo está escrito con caracteres de tamaño a 12 puntos; 72 puntos equivalen a una pulgada. Una expansión de los caracteres AaBbCc se muestra en la Fig. 2.5. El texto explicativo bajo cada figura tiene tamaño de 10 puntos. Si se tiene una tamaño de letra más grande que 24 puntos el OCR lo interpreta como un gráfico por lo que no se reconocerá.

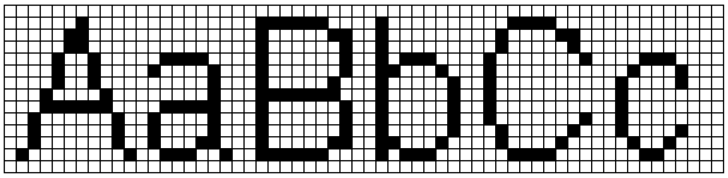

Figura 2.5: Escalamiento de los caracteres AaBbCc. El tamaño, forma y estilo (font) originales de estos caracteres es 12pt Helvética.

- La tecnología del OCR está basada en análisis de contornos y separa automáticamente el texto de los gráficos. Solo reconoce letras impresas o escritas en máquina de escribir. No existe actualmente un OCR general para letras escritas a mano.

- Define un carácter como una forma que puede ser ordenada con otras formas para comunicar información. Los caracteres legibles incluyen las letras del alfabeto, los dígitos decimales 0 a 9, los signos de puntuación y los operadores matemáticos y lógicos (+, −, !, ?, (,), etc.).

A fin de verificar las propiedades de las técnicas aplicadas en el preprocesamiento y segmentación se realiza una prueba de efectividad que consiste en lo siguiente: con algunas imágenes de mala calidad se probarán los cuatro algoritmos propuestos que generan la imagen binaria, correspondiente al plano de caracteres. Se definen tres variables:

n.- Número de caracteres reconocidos visualmente en la imagen original.

n_b.- Número de caracteres reconocidos visualmente en la imagen binaria de salida.

n_r.- Número de caracteres reconocidos por el OCR de la imagen binaria de salida.

Definiremos dos coeficientes:
C_E = razón de caracteres extraídos,

$$C_E = \frac{n_b}{n} \tag{2.2}$$

C_R = razón de caracteres reconocidos,

$$C_R = \frac{n_r}{n} \tag{2.3}$$

Idealmente el valor del coeficiente C_E debe ser igual a uno, esto es, todas las formas de los caracteres de la imagen original sos extraídos. También idealmente el valor del coeficiente C_R debe ser igual a uno, lo que significa que todos los caracteres "vistos" en la imagen original fueron reconocidos por el OCR.

El parámetro objetivo será C_R ya que C_E depende de la apreciación para obtener n_b.

Los resultados se presentarán en el Capítulo 5.

2.5 Dispositivos y paquetes de cómputo utilizados

Los dispositivos que conformaron nuestro sistema para realizar procesamiento de imágenes fue:

- Un scanner HP Scanjet IIc [6].
- Computadoras basadas en microprocesadores 386 ó 486 con adaptador (tarjeta) SuperVGA de Tseng [22] y disco duro.
- Se contó además con el acceso a plataformas SUN a través de la red interna de cómputo.

Los paquetes de cómputo involucrados:

- Conocimiento de los sistemas operativos DOS, Windows y Unix.
- Compiladores y ambientes de programación en lenguaje C.
- Utilización de varios programas, bajo ambiente Windows, para edición de imágenes.

Las imágenes adquiridas con el scanner se guardaron en el disco duro de la computadora (PC o estación de trabajo).

El dispositivo para desplegado de las imágenes esta soportado por los monitores de las computadoras PC. Se crearon dos versiones del programa de desplegado de imágenes TGC, uno con resolución de 320×200 pixeles a 256 colores con el estándar VGA, y el otro permite resoluciones de hasta

640×480 ó 800×600 pixeles, a 256 colores, en Super VGA (ver Apéndice II).

La programación de los algoritmos se hizo en lenguaje de programación C, esto permite la transportabilidad de los programas entre las plataformas SUN y PC. Sobre la PC se utilizó el compilador Quick C 2.5 de Microsoft.

La comunicación entre las estaciones de trabajo y las PC está sostenida por la red interna de cómputo.

El paquete para OCR y otros programas que trabajan con imágenes, corren bajo ambiente Windows. Los programas utilizados fueron: Paintbrush, DoDot [5] y PhotoFinish, este último es un programa editor de imágenes suministrado conjuntamente con el scanner HP ScanJet IIc.

Capítulo 3

TÉCNICAS DE PREPROCESAMIENTO Y SEGMENTACIÓN

Las técnicas de preprocesamiento mejoran la presentación visual de una imagen, comúnmente para interpretación humana.

La técnica de igualación del histograma se trata en la sección 3.1 Esta es una técnica clásica que mejora el contraste de una imagen o que se utiliza para reducir el número de niveles de gris en una imagen.

En la sección 3.2 se revisa la técnica de filtrado espacial. Esta se utiliza para transformaciones de imágenes. El desarrollo de los cuatro algoritmos del capítulo 4, se basan, directamente o con modificaciones, en esta técnica.

Las técnicas de segmentación extraen información de una imagen subdividiéndola en sus partes constituyentes, y son, generalmente el primer paso para el análisis de la imagen, se describen en la sección 3.3. La segmentación autónoma, o automática, es una de las tareas más difíciles del procesamiento de imágenes. Este paso del proceso determina el éxito o fallo final del análisis.

3.1 Igualación del histograma

El histograma de una imagen digital con niveles de gris en la amplitud de $[0, L-1]$, es función discreta

$$p(r_k) = \frac{n_K}{n} \tag{3.1}$$

donde:

r_k es el k-ésimo nivel de gris,

n_k es el número de pixeles de la imagen con tal nivel de gris,

n es el número total de pixeles, y

$k = 0, 1, 2, \ldots, L - 1$ nivel de gris.

Hablando vagamente, $p(r_k)$ da una estimación de la probabilidad de ocurrencia del nivel gris r_k. Una gráfica de esta función para todos los niveles de k provee una descripción global de la apariencia de la imagen. Por ejemplo, la Fig. 3.1 muestra cuatro tipos básicos de imágenes. La Fig. 3.1(a) presenta los niveles de gris concentrados en la zona obscura de la escala de grises, por lo que corresponde a una imagen con toda su apariencia obscura. La Fig. 3.1(b) presenta lo opuesto, corresponde a una imagen brillante. El histograma de la Fig. 3.1(c) tiene forma angosta, lo cual indica una amplitud dinámica pequeña y corresponde a una imagen con bajo contraste. Como todos los tonos de gris están concentrados en la mitad de la escala, la imagen pudiese dar la apariencia de un gris sombrío. Finalmente, la Fig. 3.1(d) muestra un histograma significativamente extendido, corresponde a una imagen con alto contraste. Sin embargo, esta características son globales y no nos dicen nada especifico acerca del contenido de la imagen.

La técnica de nivelación del histograma manipula el histograma de una manera útil. Sea r la variable para representar un nivel gris, que en una imagen de 8-bits por nivel, tomará valores de 0 a 255. Sean representados los niveles de gris en la imagen modificada por q en la misma amplitud de valores. Entonces la ecuación de transformación es:

$$q = T(r) \tag{3.2}$$

donde $T()$ es algún operador de transformación. Esto significa que para cada pixel en la imagen original con el nivel de gris r, el pixel con la misma localidad en la imagen modificada estará dado por el nivel de gris $T(r)$.

En la igualación de histograma, el resultado deseado es que la densidad de probabilidades de q (los niveles de gris en la imagen modificada) sea

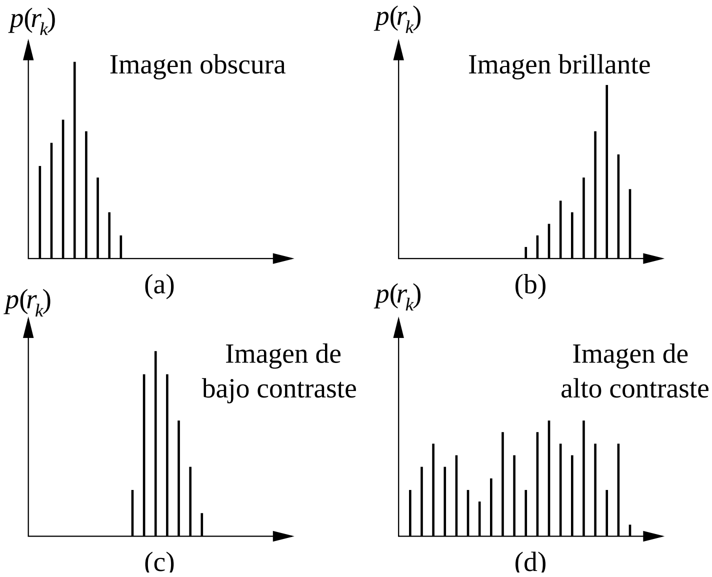

Figura 3.1: Histogramas correspondientes a cuatro tipos básicos de imágenes

uniforme para toda la amplitud de valores de q. La función de densidad de probabilidades puede obtenerse de los valores del histograma de la imagen original.

Usando la función de distribución acumulativa [4] de r se obtiene la mayor densidad de probabilidad uniforme, esta es en forma discreta.

$$q_k = T(r_k) = I_{\max} \sum_{j=0}^{k} p_r(r_j) \tag{3.3}$$

para $k = 0, 1, \ldots, L-1$, y donde $I_{\max} = L-1$ sirve para escalar al máximo nivel de gris.

Esta técnica se aplica para realzar el contraste de una imagen. También se aplica para reducir el número de niveles de gris, de la siguiente manera: para $k = 0, 1, \ldots, L_n - 1$, donde $L_n - 1$ es el nuevo nivel máximo de gris y $L_n < L$, se obtiene una nueva $I_{max} = L_n - 1$ para la Ec. 3.3.

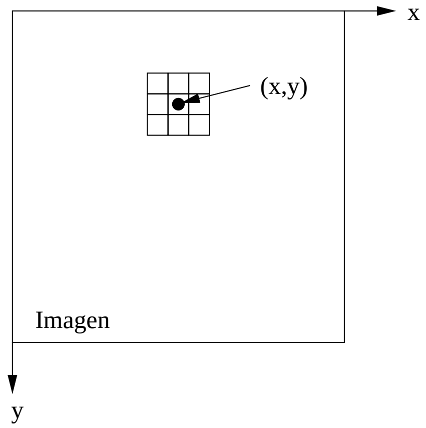

Figura 3.2: Una ventana de tamaño 3×3 sobre un punto (x, y) de una imagen.

3.2 Filtrado espacial

Primero se desarrollarán las ideas en que se basan los métodos en el dominio del espacio y luego se describirán los diferentes filtros espaciales.

3.2.1 Métodos espaciales

Los métodos espaciales operan sobre agregados de pixeles que componen una imagen. Las funciones del procesado de imágenes en el dominio del espacio pueden expresarse como:

$$g(x, y) = T[f(x, y)] \tag{3.4}$$

donde $f(x, y)$ es la imagen de entrada, $g(x, y)$ es la imagen procesada, y T es un operador sobre f, definido sobre una ventana en (x, y). El operador T también puede operar sobre un grupo de imágenes.

La aproximación principal para definir una ventana alrededor de un punto (x, y) es usando una subimagen de área cuadrada o rectangular, centrada en (x, y), como se muestra en la Fig. 3.2. El centro de la subimagen es movido de un pixel a otro pixel de inicio que corresponde a la esquina superior izquierda; aplicando cada operador a cada localidad (x, y) resulta g en esa localidad. Aunque existen otras formas de ventana, tal como la aproximación a un círculo, las formas cuadradas y rectangulares son, por mucho, las más predominantes por su fácil realización.

La forma más simple de T es cuando la ventana es de 1×1. En este caso g depende sólo del valor de f en (x, y), y T se convierte en una función de transformación del nivel de gris (llamado también mapeo) de la forma de la Ec. 3.2. Las ventanas más grandes permiten una gran variedad de funciones de procesamiento. La aplicación más común es en realce de imágenes. Sin tomar en cuenta la aplicación específica, la aproximación general es poner los valores de f en una ventana predefinida de (x, y) y determina el valor de g en (x, y). Una de las aplicaciones de esta formulación está basada en el uso de las llamadas máscaras. Básicamente una máscara es un arreglo pequeño en dos dimensiones (3×3 mínimo) tal como el mostrado en la Fig. 3.3, en la cual los valores de los coeficientes determinan la naturaleza del proceso.

$$\frac{1}{\mathrm{K}} \text{ x } \begin{array}{|c|c|c|} \hline \mathrm{w1} & \mathrm{w2} & \mathrm{w3} \\ \hline \mathrm{w4} & \mathrm{w5} & \mathrm{w6} \\ \hline \mathrm{w7} & \mathrm{w8} & \mathrm{w9} \\ \hline \end{array}$$

Figura 3.3: Una máscara de tamaño 3×3, con coeficientes arbitrarios (pesos).

Las máscaras son frecuentemente referidas como máscaras de convolución espacial relacionándolas con el desarrollo directo del teorema de convolución [15, caps. 3,4]. Estrictamente hablando, esto no es correcto porque la convolución entre dos imágenes involucra el giro de una de estas. El uso de este nombre es correcto sólo cuando la máscara es simétrica con respecto a su origen.

3.2.2 Filtraje

La técnica del filtrado espacial se basa en el uso de máscaras espaciales, usualmente es llamado *filtrado espacial* (opuesto a filtrado en el dominio de la frecuencia usando la transformada discreta de Fourier), y las máscaras en sí mismas son llamadas *filtros espaciales*.

La Fig. 3.4 muestra una sección transversal de filtros pasa-bajos, pasa-altos y pasa-banda circulares simétricos en el dominio de la frecuencia y sus correspondientes filtros espaciales. El eje horizontal para las figuras de arriba corresponden a la frecuencia, y su contraparte de abajo son coordenadas espaciales. El eje horizontal para las figuras de arriba corresponden a la frecuencia, y su contraparte de abajo son usadas como guías para especificar

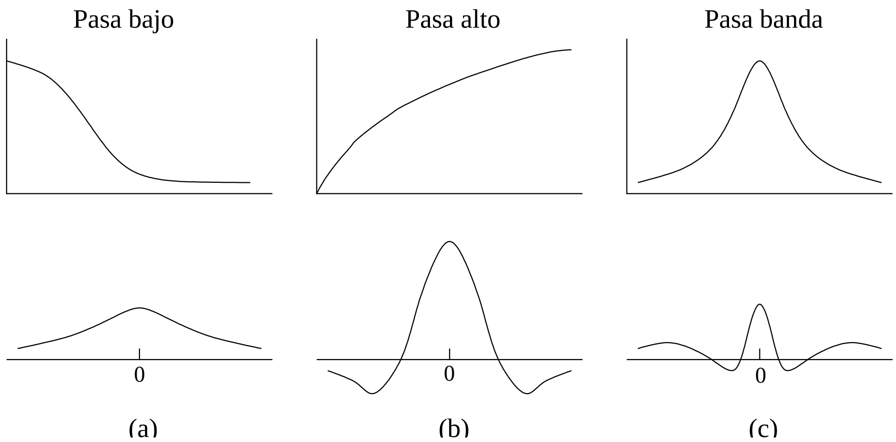

Figura 3.4: Arriba: secciones transversales de las formas básicas para filtros de simetría circular en el dominio de la frecuencia. Abajo: secciones transversales de los correspondientes filtros en el dominio espacial.

filtros espaciales lineales. Sin importar el tipo de filtro usado, la aproximación básica es una suma de productos entre los coeficientes de la máscara y las intensidades de los pixeles bajo la máscara. La Fig. 3.3, muestra una máscara típica de tamaño 3×3. Denotando los pixeles bajo la máscara en cualquier localidad por $z_1, z_2, \ldots, z_9$, la respuesta de una máscara lineal es

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{j=1}^{9} w_j z_j \tag{3.5}$$

El resultado de R en la Ec. 3.5 generalmente queda fuera de la amplitud válida en el nivel de gris, por lo que se escala dividiendo R entre factor K, el mismo que esta representado en la Fig. 3.3.

Los filtros espaciales no lineales también operan sobre ventanas. En general, de cualquier modo, su operación está basada directamente sobre los valores de los pixeles en la ventana bajo consideración y no usan explícitamente los coeficientes en la manera descrita por la Ec. 3.5. La reducción de ruido puede lograrse efectivamente con un filtro no lineal (filtros de suavizamiento) cuya función básica es computar la medida de los valores de niveles de gris en el lindero en el cual el filtro es localizado. Otros ejemplos incluyen al filtro max (con una respuesta $R = \max\{z_k | k = 1, 2, \ldots, 9\}$) el cual es usado para encontrar el punto más brillante en una imagen, y el filtro min, el cual es usado para el propósito opuesto.

3.3 Segmentación

Los algoritmos para segmentar imágenes monocromáticas generalmente se basan en una de las dos propiedades básicas de los valores del nivel de gris: discontinuidad y similitud. En la primera categoría, la aproximación es particionar la imagen basándose en cambios abruptos en el nivel de gris. Las principales áreas de interés en esta categoría son de detección de puntos aislados, de líneas y de bordes en una imagen. La segunda categoría se basa en segmentar por niveles de umbral, en regiones de crecimiento y en corte y pegado de regiones. La técnica por un nivel de umbral global se trata en la subsección 3.3.1. La técnica basada en la detección de bordes se estudia en la subsección 3.3.2. Los otros esquemas para segmentación pueden revisarse en [2, 12, 16].

3.3.1 Segmentación por nivel de umbral global

Suponiendo que el histograma mostrado en la Fig. 3.5(a) corresponde a una imagen, $f(x,y)$, compuesta de objetos brillantes sobre un fondo obscuro, de tal manera que los pixeles del objeto y el fondo tienen niveles de gris agrupados en dos modos dominantes. Un camino obvio para extraer los objetos del fondo es seleccionado un umbral T que separe estos modos. Entonces cualquier punto (x,y) para el cual $f(x,y) > T$ es llamado un punto objeto; de otra manera, el punto es llamado un punto del fondo. La misma aproximación básica clasificada un punto (x,y) como perteneciente a un tipo de objeto $T_1 < f(x,y) \leq T_2$, la otra clase de objeto si $f(x,y) > T_2$ y el fondo si $f(x,y) \leq T_1$, este es llamado multiniveles de umbral.

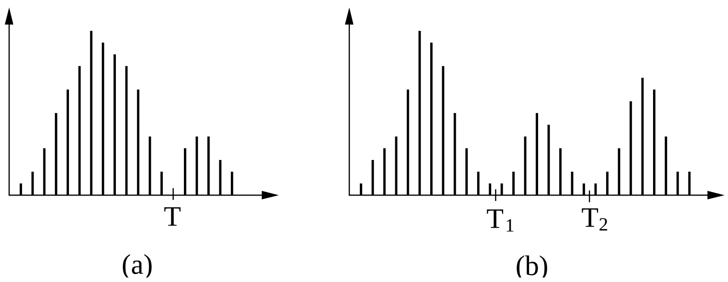

Figura 3.5: Histogramas de niveles de gris que pueden ser particionados por (a) en un umbral simple y (b) en umbrales múltiples.

La técnica más simple de segmentación por nivel de umbral es particionar el histograma de la imagen usando un umbral simple T. La imagen particionada $g(x, y)$ esta definida como:

$$g(x,y) = \begin{cases} 1, & \text{si} \quad f(x,y) > T \\ 0, & \text{si} \quad f(x,y) \leq T \end{cases} \tag{3.6}$$

Entonces los pixeles etiquetados con 1 (o cualquier otro nivel de intensidad conveniente) corresponden a los objetos, mientras que los pixeles etiquetados con 0 corresponden al fondo.

En la práctica, el tipo de umbral global descrito por la Ec. 3.6 es útil en ambientes altamente controlados. Una de las áreas donde es posible aplicarlo es en aplicaciones de inspección industrial, donde el control de la iluminación usualmente es posible y, por lo tanto, se puedan obtener histogramas bimodales.

El tipo de multiniveles de umbral es menos realizable. La razón estriba en la dificultad de establecer umbrales múltiples que aislen efectivamente regiones de interés, especialmente cuando el número de modos del histograma es grande. Típicamente, los problemas de esta naturaleza, si se van a manejar por varios niveles de umbral, se resuelven mejor usando un solo nivel de umbral variable.

3.3.2 Segmentación por selección del umbral basándose en las características de los bordes

Una aproximación para mejorar la forma del histograma es considerar solo aquellos pixeles que están situados sobre o cerca de los bordes entre los objetos y el fondo. Así se hace que el histograma sea menos dependiente del tamaño de los objetos y el fondo. Esto es, si se tiene una imagen con un objeto pequeño, el histograma dará un pico muy grande que correspondería al fondo, pero si se usa solo los límites entre los objetos, el histograma dará, aproximadamente, dos picos de la misma altura. Esto se puede lograr con los operadores gradiente y Laplaciano. La detección de bordes es uno de los métodos más comunes para la detección de discontinuidades en el nivel de gris. Primero se revisará la teoría en que se basa esta técnica y luego se describirá el algoritmo usado.

3.3.3 Formulación básica

Un borde es el límite entre dos regiones con propiedades relativamente distintas en el nivel de gris.

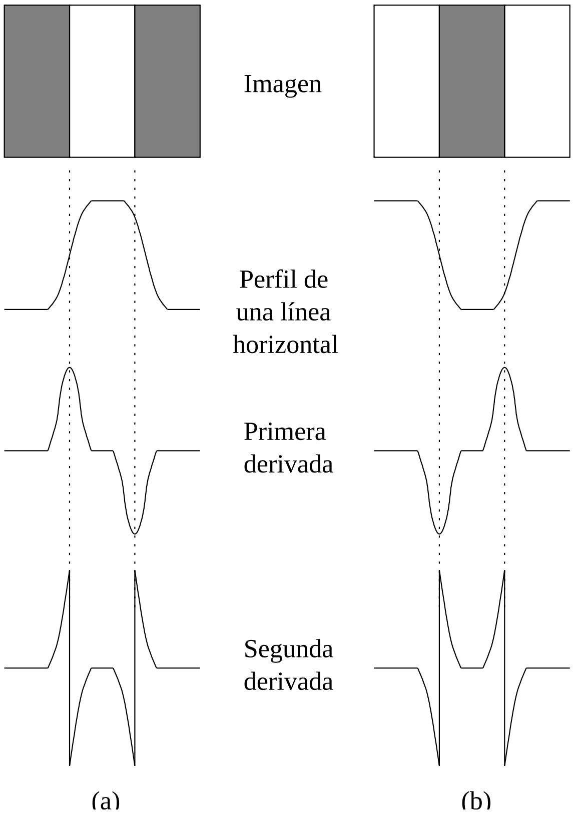

Figura 3.6: Detección de bordes por los operadores derivativos: (a) Una banda brillante sobre un fondo obscuro, (b) Una banda obscura sobre un fondo brillante. Nótese que la segunda derivada tiene un cruce por cero en la localización de cada borde.

Básicamente, la idea fundamental de la mayoría de las técnicas para detectar bordes es la computación de un operador local derivativo. La figura 3.6 ilustra este concepto. La Fig. 3.6(a) muestra la imagen de una banda brillante sobre un fondo obscuro, el perfil de los niveles de gris a lo largo de una línea horizontal, y la primera y segunda derivadas del perfil. Nótese que el perfil de un borde (la transición de obscuro a brillante) está modelado de manera suave, mas que de manera abrupta. Este modelo refleja el hecho de que los bordes en las imágenes digitales son generalmente un poco borrosas como resultado del muestreo.

La Fig. 3.6(b) muestra que la primera derivada del perfil del nivel de gris es positiva en el principal borde de una transición, negativa en el borde siguiente, y, como es de esperarse, cero en las áreas que tienen el nivel de gris constante. La segunda derivada es positiva para la parte de la transición asociada con el lado obscuro del borde, negativa para la parte de la transición asociada con el lado brillante del borde, y cero en las áreas con nivel de gris constante. Por lo tanto, la magnitud de la primera derivada puede usarse para detectar la presencia de un borde en una imagen, y el signo de la segunda derivada puede usarse para determinar si un pixel esta situado sobre el lado obscuro o brillante de un borde. Nótese que la segunda derivada tiene un cruce por cero en el punto medio de la transición del nivel de gris.

Aquí la discusión se limitó a un perfil horizontal en una dimensión, similares argumentos se aplican a un borde en cualquier orientación en una imagen. Simplemente se define un perfil perpendicular a la dirección del borde en cualquier punto deseado y se interpretan los resultados como en la discusión anterior. La primera derivada en cualquier punto de una imagen es obtenida usando la magnitud del gradiente en aquel punto. La segunda derivada es obtenida de manera similar usando el Laplaciano.

3.3.4 Operadores de gradiente

El gradiente es el método más común de diferenciación en aplicaciones de procesamiento de imágenes. Para una función $f(x, y)$, el gradiente de f en las coordenadas (x, y) esta definido como el vector

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix} \tag{3.7}$$

Es bien conocido en análisis de vectores que el vector gradiente apunta en la dirección del cambio máximo de f en (x, y). En la detección de bordes una cantidad importante es la magnitud de este vector, referido generalmente como gradiente y denotado por f, donde:

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2} \tag{3.8}$$

Esta cantidad es igual a la máxima velocidad de incremento de $f(x, y)$ por unidad de distancia en la dirección de f. En la práctica es común aproximar el gradiente con los valores absolutos:

$$\nabla f = |G_x| + |G_y| \tag{3.9}$$

ó como:

$$\nabla f = \max(G_x, G_y) \tag{3.10}$$

las cuales son mucho más simples de realizar, particularmente con hardware dedicado.

La dirección del vector gradiente también es una cantidad importante. Si (x, y) representa el ángulo de dirección del vector (x, y), entonces del análisis de vectores

$$\alpha(x, y) = tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{3.11}$$

donde el ángulo es medido con respecto al eje de la x.

Las derivadas de la Ec. 3.7 pueden implementarse en forma digital de varias maneras. Una es usando los operadores Sobel mostrados en la Fig. 3.7 que tienen la ventaja de proveer además un efecto de suavizamiento. Y debido a que la derivada enfatiza el ruido, el efecto de suavizamiento es una característica particularmente atractiva de los operadores Sobel.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

(a)

−1	0	1
−2	0	2
−1	0	1

(b)

−1	−2	−1
0	0	0
1	2	1

(c)

Figura 3.7: (a) Una región de una imagen de tamaño 3×3 pixels. (b) Máscara usada para computar G_x en el punto central de la región (a). (c) Máscara para computar G_y en el mismo punto. Estas máscaras son referidas frecuentemente como operadores Sobel.

Entonces, la Ec. 3.7 queda usando las máscaras de la Fig. 3.7 como:

$$G_x = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7) \tag{3.12}$$

$$G_y = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3) \tag{3.13}$$

3.3.5 El Laplaciano

El Laplaciano de una función en 2-D $f(x, y)$ es una derivada de segundo orden definida como

$$\nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3.14}$$

Como en el caso del gradiente, Ec. 3.14 puede implementarse de varias maneras en forma digital. Para una región de 3×3, la forma más frecuentemente encontrada es la que se muestra en la Fig. 3.8.

0	1	0
1	−4	1
0	1	0

Figura 3.8: Máscara de 3×3 para computar el Laplaciano

$\nabla^2 f$ de la Ec. 3.14, y viendo las Figs. 3.7(a) y 3.8, se aproxima como:

$$\nabla^2 f = (Z_2 + Z_4 + Z_6 + Z_8) - 4Z_5 \tag{3.15}$$

El requerimiento básico en la definición del Laplaciano digital es que los coeficientes asociados con el pixel central sea positivo y que coeficientes asociados con los pixeles externos sean negativos. Porque el Laplaciano es una derivada, la suma de los coeficientes tiene que ser cero. Por lo tanto la respuesta es cero siempre que el punto tratado y sus linderos tengan el mismo valor.

El Laplaciano responde a las transiciones en intensidad, y este es raramente usado por la detección de bordes por varias razones. Como es una derivada de segundo orden el Laplaciano típicamente es inaceptablemente sensible al ruido. Aún mas, el Laplaciano produce bordes dobles y es incapaz de detectar la dirección del borde. Por estas razones, el Laplaciano usualmente juega un papel secundario como detector para establecer cuando un pixel esta sobre el lado brillante u obscuro de una imagen.

3.3.6 Algoritmo

El gradiente y el Laplaciano pueden usarse para formar una imagen en tres niveles, como sigue:

$$s(x,y) = \begin{cases} 0, & \text{si} \quad \nabla f < T \\ +, & \text{si} \quad \nabla f \geq T \quad \text{y} \quad \nabla^2 f \geq 0 \\ -, & \text{si} \quad \nabla f \geq T \quad \text{y} \quad \nabla^2 f < 0 \end{cases} \tag{3.16}$$

donde los símbolos O, $+, -$ representan cualesquiera de tres distintos niveles de gris, T es un umbral, y el gradiente y el Laplaciano son computadoras en cada punto (x, y). Para un objeto obscuro sobre un fondo brillante el uso de la Ec. 3.16 produce una imagen $s(x, y)$ en el cual todos los pixeles que no están sobre un borde (como los determinados por f al ser menores que T) son etiquetados a 0, todos los pixeles en el lado obscuro de un borde son etiquetados a $+$ y todos los pixeles sobre el lado brillante son etiquetados a $-$. Los símbolos $+$ y $-$ se intercambian para un objeto brillante sobre un fondo obscuro.

La información obtenida por este procedimiento puede usarse para generar una imagen binaria segmentada, en la cual los 1s corresponden al objeto de interés y los 0s corresponden al fondo. La transición (a lo largo de una linea horizontal o vertical) de un fondo brillante a un objeto obscuro debe caracterizarse por la ocurrencia de un '$-$' seguido de un '$+$' en $s(x, y)$. El interior del objeto está compuesto de pixeles que están etiquetados con 'O' ó '$+$'. Finalmente, la transición del objeto al fondo esta caracterizado por la ocurrencia de un '$+$' seguido de un '$-$'. Así una linea horizontal o vertical que contiene una sección de un objeto tiene la siguiente estructura:

$(\dots)(-,+)(\text{O ó } +)(+,-)(\dots)$

donde $(\dots)$ representa cualquier combinación de $+$, $-$ y O. Los paréntesis más internos contienen puntos de objeto que son etiquetados a 1. Todos los demás pixeles de la misma línea son etiquetados a O, con la excepción de cualquier otra secuencia de (O ó $+$) limitados por $(-,+)$ y $(+,-)$.

Capítulo 4

ALGORITMOS PROPUESTOS PARA OBTENER EL PLANO DE CARACTERES

Los algoritmos propuestos se basan en técnicas básicas para la segmentación de imágenes. Se proponen cuatro algoritmos:

1. Por detección de bordes
2. Por nivel lógico
3. Por filtro de formas
4. La técnica de Yasuda-Dubois-Huang (YDH)

El algoritmo 1 viene propuesto por Gonzalez [15], y es semejante al descrito en la sección 3.3.2. El algoritmo 2 fue propuesto por Kamel y Zhao [7] quienes la presentan como superior a las demás pero se han tenido en las pruebas del algoritmo discrepancias con respecto a lo que reportan; se sugiere aquí una técnica mejorada. El algoritmo 3 se propone como nuevo y se obtienen buenos resultados. La técnica YDH [26] fue usada para compresión de imágenes de cheques bancarios.

Los algoritmos 1, 2 y 4 necesitan algunos parámetros para su aplicación. Se propone un ajuste automático para un parámetro del algoritmo 1 y el nuevo algoritmo 3 necesita sólo un parámetro que no resulta crítico.

4.1 ALGORITMO NO. 1 POR DETECCIÓN DE BORDES

Este algoritmo está basado en la técnica de segmentación por selección del umbral basándose en las características de los bordes, tratando en la sección 3.3.2.

Los pasos que involucra el algoritmo por detección de bordes se muestran en la Fig. 4.1. Se puede dividir en dos partes principales: detección de los bordes y la binarización.

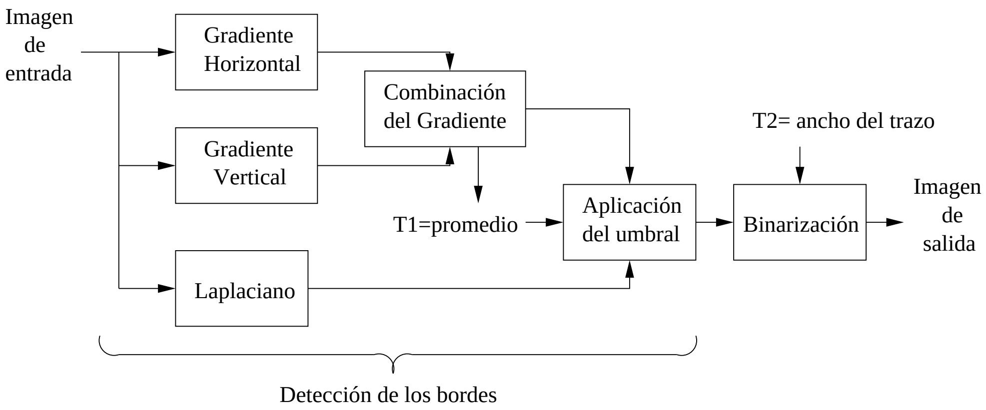

Figura 4.1: Diagrama que muestra los pasos involucrados en el algoritmo basado en la detección de bordes

4.1.1 Detección de los bordes

El gradiente puede calcularse en forma digital de varias maneras. Aquí se usaron los operadores Sobel de la Fig. 3.7. Los operadores Sobel de nuevo mostrados en la Fig. 4.2 tienen la ventaja de proveer además un efecto de suavizamiento y debido a que la derivada enfatiza el ruido, el efecto de suavizamiento es una característica particularmente atractiva de los operadores Sobel. El factor de escalamiento 1/8 (ver Fig. 4.2) suaviza aun más el ruido y mantiene el valor del gradiente entre 0 y 127.

Los gradientes en x y y se combinan usando la Ec. 3.10. De esta manera el gradiente tienen una pendiente suave en las esquinas de los bordes.

El Laplaciano se calcula con la máscara mostrada en la Fig. 4.4. La imagen generada por el Laplaciano es:

$$\frac{1}{8} \text{ x } \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8} \text{ x } \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

(a) (b)

Figura 4.2: (a) Operador Sobel usado para computar el gradiente horizontal. (b) Operador Sobel usado para computar el gradiente vertical.

$$L(x,y) = \begin{cases} 255, & \text{si} \quad \nabla^2 f(x,y) \geq 0 \\ 0, & \text{si} \quad \nabla^2 f(x,y) < 0 \end{cases} \tag{4.1}$$

donde $f(x,y)$ representa la imagen de entrada.

Y se genera una imagen en tres niveles de gris usando la Ec. 3.16 de la siguiente manera:

$$s(x,y) = \begin{cases} 255, & \text{si} \quad \nabla f(x,y) < T_1 \\ 128, & \text{si} \quad \nabla f(x,y) \geq T_1 \quad \text{y} \quad L(x,y) = 255 \\ 192, & \text{si} \quad \nabla f(x,y) \geq T_1 \quad \text{y} \quad L(x,y) = 0 \end{cases} \tag{4.2}$$

Estos tres valores forman la imagen $s(x,y)$ con el fondo blanco (255), donde no hay bordes, un tono medio claro (192) en el lado brillante de un borde y otro tono más obscuro (128) en el lado obscuro del borde; esto produce una imagen en que el contorno exterior de las letras "se ve" en gris claro, el fondo en blanco y el contorno interno de los caracteres queda en gris obscuro, el efecto se muestra en la Fig. 4.3. El por qué de estos valores tendrá su justificación en el siguiente paso de la binarización.

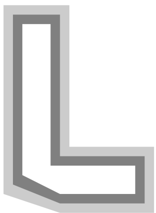

Figura 4.3: Resultado de aplicar la ecuación (4.2) a la forma de un caracter, en el caso de una L. Aquí se muestra la situación ideal "sin ruido".

El valor de T_1, sugerido en [7, pag. 455], es el valor del gradiente localizado en el punto medio del valle en un histograma del gradiente que contenga dos lóbulos principales. No es lo más común tener un histograma con esas características pero se tiene una muy buena aproximación, en cualquier caso, usando el valor medio del gradiente de la imagen entera como valor para el umbral T_1.

4.1.2 Binarización

La información obtenida usando la Ec. (4.2) se puede generar una imagen binaria en la cual los $0's$ (negro) corresponderán al objeto de interés y los $1's$ (blanco) corresponderán al fondo. La transición (a lo largo de una línea vertical u horizontal de $s(x, y)$ en la Ec. 4.2) del fondo brillante al trazo obscuro de la letra, se caracteriza por la ocurrencia del valor 128 seguido por un 192. El interior de la letra está compuesto de pixeles que son 255 ó 128. Finalmente, la transición del trazo obscuro al fondo esta caracterizado por la ocurrencia de un 128 seguida de un 192. De esta manera una linea vertical u horizontal conteniendo la sección de un trazo de una letra tiene la siguiente estructura:

$(\bullet \bullet \bullet)(192, 128)(255 \text{ ó } 128)(128, 192)(\bullet \bullet \bullet)$

donde $(\bullet \bullet \bullet)$ representa cualquier combinación de $128, 192$ ó 255. Los paréntesis más internos contienen los pixeles del trazo de la letra. La distancia entre las ocurrencias $(192, 128)$ y $(128, 192)$ es el "ancho del trazo" y su valor define el parámetro T_2 de la Fig. 4.1. Los pixeles del fondo tienen a no estar limitados por esas secuencias. Entonces, para extraer los caracteres, se examinan las secuencias horizontales y verticales que pasan por el pixel procesado. Si estas dos secuencias están limitadas y uno de las dos, por lo menos, cumple con la restricción del tamaño del ancho del trazo, entonces el pixel procesado es parte de una letra o carácter; de otra manera es un pixel del fondo. Ahora, una implementación eficiente es:

```
/** La imagen cuenta con M columnas y N renglones **/
/** W es el ancho máximo permisible de la secuencia limitada **/

for(row=0;row<N;row++){     /** Se examinan primero los renglones **/
    while (haya_pixeles_limitados){
        if(secuencia_válida){           /* *No encierra un 192 **/
            if(largo_de_secuencia < W)  /** Se cumple restricción **/
```

```
                pixeles_limitados = 128;
            else
                pixeles_limitados = 64;
        }
    }
}

/** Los límites de las secuencias son ahora (192, 128 ó 64) y (128 ó 64,192) **/

for( col=0; col<M; col++ ){                  /** Se examinan las columnas **/
    while( haya_pixeles_limitados ){
        if( secuencia_válida ){         /** No encierra un 192 **/
            if( largo_de_secuencia < W ){
                if( pixeles_limitados < 129 )       /** Ya se marcaron
                                                    horizontalmente **/
                    pixeles_limitados = 0;      /** 0= Negro **/
            }
            else{
                if( pixeles_limitados = 128 )   /** Ya se marcaron
                                                  horizontalmente y
                                                cumplieron restricción **/
                    pixeles_limitados = 0;  /** Negro **/
            }
        }
    }
}

/** Se aplica finalmente un umbral global **/
for( row=0; row<N; row++ ){
    for( col=0; col<M; col++ ){
        if( valor_pixel > 63 )              /** U otro valor entre 1 y 63 **/
            valor_pixel = 255;          /** Blanco **/
    }
}
```

Aquí se tiene 1 byte/píxel, entonces se comprime todavía la imagen a 1 bit/pixel donde el negro es cero y el blanco uno.

Una secuencia no válida es una secuencia limitada pero que contiene algún valor de pixel igual a 192 (solo debe contener valores de 255 o menores a 129).

El valor de T_2 es asociado al ancho máximo del trazo de una letra en

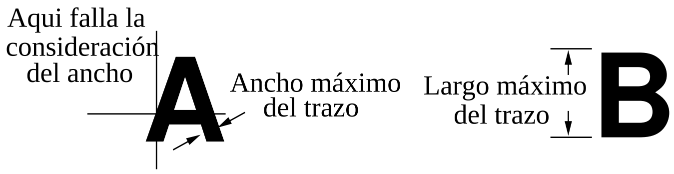

Figura 4.4: Ejemplo de la definición del ancho y largo máximo del trazo.

[7]. Pero como la búsqueda de las secuencias se realiza por renglones y por columnas, es mejor usar el valor el largo máximo del trazo para el parámetro T_2. Esto se observa en la Fig. 4.4.

4.2 ALGORITMO NO. 2: NIVEL LÓGICO

Esta técnica viene propuesta en [7] como superior a las demás, pero hay cierta discrepancia con los resultados que se reportan, lo que se describirá más a detalle posteriormente.

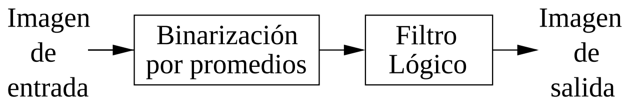

Figura 4.5: Diagrama de las etapas del algoritmo por nivel lógico

Este algoritmo consta de dos etapas como se nota en la Fig. 4.5: Una binarización previa por promedios y un filtro lógico.

Este algoritmo toma ventaja de la suposición de que los caracteres en la mayoría de los escritos son más obscuros que el fondo. Si se digitaliza un escrito de ese tipo y se hace un acercamiento a donde está un trazo de una letra, se tendrá una imagen parecida a la representada en la Fig. 4.6. En esta figura no se ha rellenado el trazo, para permitir la visualización de un pixel dentro del trazo. Alrededor de ese punto, que se encuentra dentro del trazo obscuro, se calcula el promedio por medio de un operador de máscara; el valor del promedio será mayor que el valor del mismo punto dentro del trazo, dado que en la cuantización en tonos de gris, se da un valor cercano a cero (negro) para los tonos obscuros y valores cercanos a 255 (blanco) para los brillantes.

Entonces, para realizar primero la binarización de la imagen supóngase una imagen de entrada $f(x, y)$; la binarización consiste en un operador de

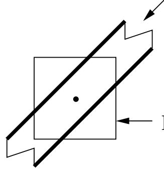

Figura 4.6: Máscara sobre el trazo del caracter

máscara de tamaño $K \times K$ que calcula el promedio alrededor de cada punto (x, y) de la imagen; la imagen binarizada $b(x, y)$ tomará el valor de 1 si la diferencia del promedio calculado alrededor del punto (x, y), menos el mismo valor del pixel, es mayor que un predeterminado umbral T, en cada contrario $b(x, y)$ tomará el valor de 0.

La aplicación de la binarización en imágenes donde los caracteres están pobremente definidos o se tiene ruido generará unos trazos discontinuos. Es aquí donde entra el filtro lógico para la reafirmación de los trazos. En la Fig. 4.7 se muestra la máscara usada en [7], definen un parámetro W que toma el valor del ancho máximo del trazo. Tomando los valores 1 y cero de la imagen generada de la binarización como valores lógicos de cierto y falso, entonces, usando álgebra de Boole [11], el punto central "e" de la máscara tomará el valor de 1 si se cumple que:

$$\begin{aligned} H &= dfai + aibh + bhcg + cgfd = ai(df + bh) + cg(bh + df) \\ &= (ai + cg)(df + bh) \end{aligned} \tag{4.3}$$

La aplicación de la Ec. 4.3 produce un adelgazamiento en los bordes de los trazos donde H no se cumplirá y el adelgazamiento se convierte en "desgaste" con el incremento de W. Ahora, lo que se quiere es una afirmación del trazo, por lo que se pierde W fijándolo igual a uno y se define una nueva condición: la imagen de salida $b_2(x, y)$ tomará el valor de 1 si se cumple (ver Fig. 4.7(b) como apoyo):

$$H = abd + bcf + fhi + dgh = b(ad + cf) + h(fi + dg) \tag{4.4}$$

El algoritmo queda expresado en forma matemática como:

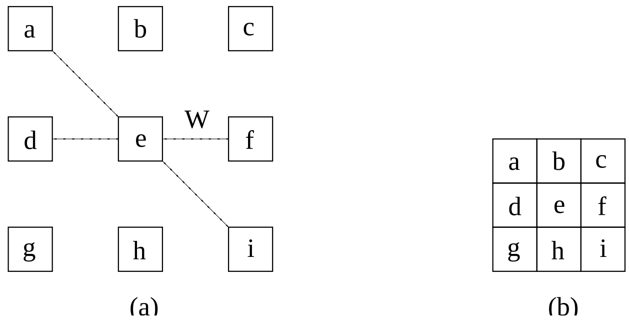

Figura 4.7: (a) Máscara para el filtro lógico sugerido en [7]. (b) Máscara propuesta para el filtro lógico. Nótese que es igual que (a) con $W = 1$.

1º Binarización:

$$b(x,y) = \begin{cases} 1, & \text{si} \quad m(x,y) - f(x,y) > T \\ 0, & \text{de otro modo.} \end{cases} \tag{4.5}$$

donde:

$$m(x,y) = \sum_{j=0}^{K-1} \sum_{j=0}^{K-1} \frac{1}{K^2} f(x + \frac{K-1}{2} - i, y + \frac{K-1}{2} - j) \tag{4.6}$$

2º Filtro lógico:

$$b_2(x,y) = \begin{cases} 1, & \text{si} \quad H \\ 0, & \text{de otro modo.} \end{cases} \tag{4.7}$$

donde $H = b(ad + cf) + h(fi + dg)$
con una máscara como se ve en la Fig. 4.7(b). Esto representa 6 operaciones AND y 3 operaciones OR.

El valor de K debe ser mayor que el ancho del trazo y la determinación del valor de umbral T es difícil determinarlo de manera automática ya que depende directamente del valor de K y del tono de gris del fondo de la imagen. Prácticamente, el valor de K no es tan crítico si se tiene alguna idea del tamaño de los caracteres o se escoge grande. El valor de T sólo se puede determinar generando una imagen de prueba con $T = 0$ y se aproxima T con algún paquete de cómputo que nos permita conocer el valor del tono de gris no deseado de la imagen. Ahora, en imágenes con suficiente contraste y poco ruido se puede prescindir de la segunda etapa del algoritmo.

4.3 ALGORITMO NO. 3: FILTRO DE FORMAS

Ahora se presenta el algoritmo nuevo. Está basado en el algoritmo de la sección 6.2 pero con la eliminación del parámetro T.

La Ec. 4.5 es poderosa y se considera una modificación de una ecuación sugerida en [15, cap. 4 pag. 184], que aplica una transformación basándose en el promedio y la desviación estándar de una ventana, para el realce local del contraste.

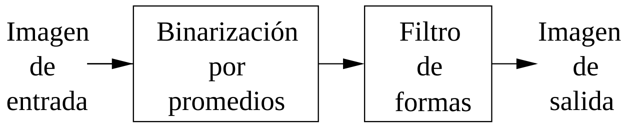

Figura 4.8: Etapas del algoritmo por filtro de formas

Este algoritmo, al contrario de los demás, solo necesita un parámetro, el tamaño de la ventana, que no resulta crítico y, a semejanza de la técnica anterior, consta de dos etapas, como se representa en la Fig. 4.8. La primera etapa es casi igual al algoritmo de la sección 4.2, pero ahora el parámetro T de la Ec. 4.5 tiene valor fijo e igual a cero. También puede eliminarse la segunda etapa si la imagen de entrada tiene un fondo uniforme y poco ruido.

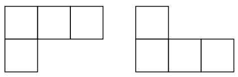

Figura 4.9: Primitivas definidas para el filtro de formas

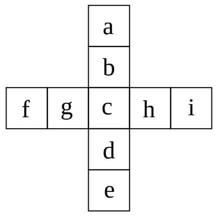

Figura 4.10: Máscara del filtro de formas

El filtro de formas consiste en lo siguiente: empíricamente se hace que los trazos de los caracteres se compongan de dos "primitivas", ilustradas en la Fig. 4.9. Los trazos se suponen compuestas por la agrupación de todas las ocho formas rotadas de estas primitivas.

La aplicación de este filtro se hace por medio de una máscara de la forma presentada en la Fig. 4.10. Tomando como valores lógicos el valor de los pixeles como resultado de la binarización, el punto central "c" se considera parte del trazo si, H se define como:

$$
\begin{aligned}
H &= abcg + abch + fgcb + fgcd + edgc + edch + ihcb + ihcd = \\
&= abc(g+h) + fgc(b+d) + edc(g+h) + ihc(b+d) = \\
&= (g+h)(abc+edc) + (b+d)(fgc+ihc) = \\
&= c(g+h)(ab+ed) + c(b+d)(fg+ih) = \\
&= c[(g+h)(ab+ed) + (b+d)(fg+ih)]. \qquad (4.8)
\end{aligned}
$$

esto representa 5 operaciones OR y 6 operaciones AND, que es un total de 11 operaciones lógicas.

Para la definición de H de la ecuación boleana 4.8 también pueden usarse la lógica complementaria, esto es, detectar los valores que no corresponden a una primitiva (0's en vez de 1's) quedando H como:

$$
\begin{aligned}
H = {} & i'd'e'g'f' + i'h'd'e'f' + a'd'e'g'f' + a'b'e'g'f' + i'a'b'g'f' + \\
& i'h'd'b'f' + a'd'e'i'h' + a'b'e'i'h' = \\
= {} & i'd'e'f'(g'+h') + a'e'g'f'(d'+b') + i'a'b'f'(g'+h') + a'e'i'h'(d'+b') = \\
= {} & i'f'(d'e'+a'b')(g'+h') + a'e'(g'f'+i'h')(d'+b') \qquad (4.9)
\end{aligned}
$$

lo que representa 10 operaciones AND y operaciones OR que es un total de 15 operaciones lógicas.

Matemáticamente el algoritmo queda definido como:

Sea $f(x,y)$ la imagen de entrada,

1º Binarización:

$$
b(x,y) = \begin{cases} 1, & \text{si} m(x,y) - f(x,y) > 0 \\ 0, & \text{de otro modo.} \end{cases} \qquad (4.10)
$$

donde $m(x,y)$ esta definido por la Ec. 4.2.

2º Filtro de formas:

$$
b_2(x,y) = \begin{cases} 1, & \text{si} H \\ 0, & \text{de otro modo.} \end{cases} \qquad (4.11)
$$

donde:

$H = c[(g+h)(ab+ed) + (b+d)(fg+ih)]$

usando una máscara como la mostrada en la Fig. 4.10.

4.4 ALGORITMO NO. 4: TÉCNICA YASUDA-DUBOIS-HUANG

Este algoritmo está basada en operaciones básicas para segmentación de imagen, la Fig. 4.11 muestra los pasos que involucra este algoritmo.

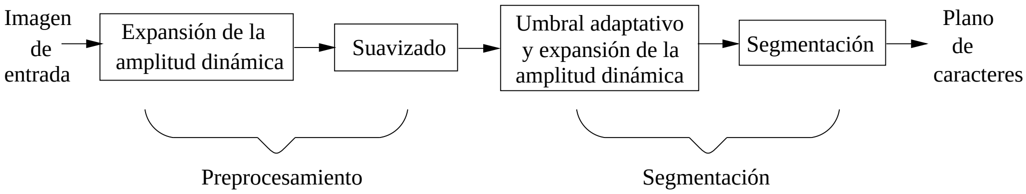

Figura 4.11: Estructura del algoritmo YDH.

El algoritmo consta de cuatro pasos que se puede agrupar en dos fases: un preprocesamiento y luego la segmentación.

4.4.1 Preprocesamiento

Los primeros dos pasos de la Fig. 4.11 forman la fase de preprocesamiento: una expansión en la amplitud dinámica y un suavizado. Primero se determina el tono de gris mínimo y máximo de toda la imagen. Estos valores se usan para expandir linealmente la escala de grises de la imagen. Esta fase es necesaria para normalizar la amplitud de los niveles de gris de la imagen a ser procesada por la parte principal del algoritmo de segmentación. La siguiente fase del preprocesamiento es un filtro suavizador no lineal. Este filtro usa una máscara como en la Fig. 4.7(b). Cuando la diferencia entre los valores máximo y mínimo de los ocho pixeles vecinos a, b, c, d, f, g, h, i es más pequeña que un predeterminado umbral T_1, entonces el valor central se reemplaza por el valor medio de los ocho vecinos. Por otro lado, si esta diferencia no es más pequeña que el valor T_1, entonces no se altera el valor de e. El propósito de este filtro es suavizar las pequeñas variaciones siempre encontrados en la región del fondo y remover pixeles aislados, mientras que se conserva la forma de los bordes.

4.4.2 Segmentación

También consta de dos fases, la primera es aplicar un umbral adaptativo seguido de una expansión de la amplitud dinámica. El propósito de esta fase es hacer el fondo de la imagen más uniforme. Primeramente se divide la imagen en bloques de tamaño $T_2 \times T_2$. En cada bloque se calcula el promedio local y este valor se usa como un umbral para todos los pixeles dentro del bloque. Después de la aplicación del umbral, la amplitud del brillo original, que va desde el mínimo local al promedio local, se expande del mínimo al máximo total (0 a 255, respectivamente). Esto significa que se consideran los pixeles con valor entre el promedio y 255 como pixeles del fondo. En este esquema el tamaño del bloque está determinado por un compromiso entre dos requerimientos contradictorios. Si el tamaño del bloque es muy grande, el fondo todavía contendrá variaciones anchas, y el algoritmo no trabajará. Por otro lado, si el tamaño del bloque es pequeño, el algoritmo confundirá un carácter grande con el fondo y lo removerá de la imagen procesada. En [26] se sugiere el tamaño de 16×16 pixeles por bloque. Para prevenir procesamiento innecesario, la expansión de la amplitud dinámica no se aplica a los bloques que tengan un brillo uniforme. Si la amplitud dinámica original de un bloque es más pequeña que el umbral T_3, entonces el bloque se considera compuesto por pixeles del fondo y se da el valor de 255 a cada pixel dentro del bloque.

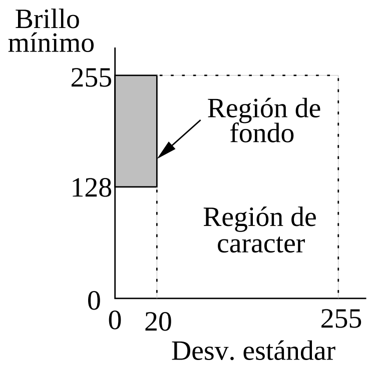

Figura 4.12: Clasificación de los pixeles mapeados en el plano de características.

En la fase de segmentación se miden las propiedades estadísticas de la imagen procesada sobre una ventana de 3×3. Se calcula la desviación estándar y se toma el brillo mínimo de la ventana local, entonces, se divide el plano de características ilustrado en la Fig. 4.12 en las regiones de sombras y no sombras las cuales corresponden al fondo y a los caracteres, respectivamente. El límite entre las dos regiones se determinan por prueba y error.

Matemáticamente todo el algoritmo se expresa como:

1.- PREPROCESADO:

a) Expansión de la amplitud dinámica:

$$g_1(x, y) = \frac{f(x, y) - \min}{\max - \min} \max \tag{4.12}$$

donde:

$f(x, y)$ es la imagen original.

max es el valor del tono máximo de la imagen global.

min es el valor del tono mínimo de la imagen global.

$g_1(x, y)$ es la nueva imagen procesada.

b) Suavizado.

$$g_2(x, y) = \begin{cases} g_1(x, y), & \text{si } \operatorname{amp}(x, y) > T_1 \\ \operatorname{prom}(x, y), & \text{de otro modo} \end{cases} \tag{4.13}$$

donde:
$\operatorname{amp}(x, y) = \max(a, b, c, d, e, f, g, h, i) - \min(a, b, c, d, e, f, g, h, i)$,
con una ventana como en la Fig. 4.7(b).

2.- SEGMENTACIÓN

a) Umbral adaptativo y expansión de la amplitud dinámica.

Se define el siguiente algoritmo:

- Se divide la ventana en bloques no sobrelapados de 16×16 pixeles Para cada bloque:
- Se halla el máximo, max.
- Se halla el mínimo, min .
- Se calcula el promedio pm.

- $if(\max - \min < T_3) \quad g_3(x,y) = 255;$
  else
  $$if(g_2(x,y) \geq pm) \quad g_3(x,y) = 255;$$
  else
  $$g_3(x,y) = \frac{g_2(x,y) - \min}{pm - \min} 255; \tag{4.14}$$

b) Segmentación

$$b(x,y) = \begin{cases} 1, & \text{si} \quad \min(x,y) < T_4 \quad \text{ó} \quad \sigma(x,y) > T_5 \\ 0, & \text{de otro modo} \end{cases} \tag{4.15}$$

donde:

$$\min(x,y) = \min(a,b,c,d,e,f,g,h,i)$$
$$\max(x,y) = \max(a,b,c,d,e,f,g,h,i)$$

con una ventana como en la Fig. 4.7(b).

Significado de los parámetros:

$T_1 \rightarrow$ Dentro de la ventana de 3×3, si la amplitud (el valor máximo menos el mínimo) es menor que T_1, se suaviza el punto central de la ventana sustituyéndolo por el promedio de sus vecinos.

$T_2 \rightarrow$ Se considera fijo e igual a 16.

$T_3 \rightarrow$ Da el valor de umbral para el cual el bloque de 16×16 se considera fondo. Si la amplitud dentro del bloque es menor que T_3, se considera que es fondo y se sustituye por 255 (blanco).

$T_4 \rightarrow$ Valor para la segmentación. Si el valor mínimo dentro de la ventana de 3×3 es menor que T_4 se considera un pixel del carácter.

$T_5 \rightarrow$ Valor para la segmentación. Da el valor de umbral para la desviación estándar dentro de la ventana de 3×3. Si los valores dentro de la ventana son mayores que T_5 se considera el punto central de la ventana como parte de un pixel de caracter.

Capítulo 5

RESULTADOS OBTENIDOS

Para las pruebas cualitativas de los distintos algoritmos se usaron cuatro imágenes:

1) Una imagen de un fragmento de un documento, presentada en la Fig.5.1(a), que tienen mala calidad y sombras mezcladas con caracteres. Esta imagen es la misma que una de las usadas en [7].

El número n caracteres reconocidos visualmente, de las ecuaciones 2.2 y 2.3 es igual a 230 caracteres que corresponden al texto:

> d the number of black pixels
> inmensional ratios are easily ob
> on. This algorithm is described
> this algorithm is similar to the
> he literature [11] [9] in our
> ge the value of each pixel to its
> label but retain the coordinates
> of each componen. Each con
> either rejected or accepted as

no se consideran los signos ortográficos como son los puntos, comas, etc. y los caracteres incompletos en los bordes de la imagen. Cabe recordar que n es un parámetro subjetivo, la calidad de esta imagen es muy mala pero el texto anterior es el que contiene la imagen del documento y es el texto que se "ve".

2) Imagen de un fragmento de un **bono** que contiene como fondo el dibujo de una despensa y caracteres sobreescritos en ella, es la Figura 5.2(a).

El número n de las ecuaciones (2.2) y (2.3) es igual a 61 caracteres que corresponden al texto:

CINCUENTA CENTAVOS N 0 50
3
CH 131

IDAD DE LAS AMERICAS
ATARINA MARTIR CHOLUL

Este texto es más fácil de “ver” que el texto de la imagen del documento.

3 La imagen anterior, el **bono**, pero **suavizada** con dos pasadas de un filtro de promedios usando la máscara 3×3 de la Fig. 5.1; este es el filtro más simple para realizar el suavizado, involucra nueve sumas y una división de números enteros. Esta imagen se ve en la Fig. 5.4(a)

$$\frac{1}{9} \text{ x } \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figura 5.1: Máscara para el filtro de promedios (suavizado).

4) Una imagen de un fragmento de una factura, mostrada en la Fig. 5.5(a). Esta imagen proviene de un fotocopiado deficiente del original irrecuperable.

El número n de las ecuaciones (2.2) y (2.3) es igual a 16 caracteres que corresponden al texto:

SUB TOTAL
IVA
TOTAL

5.1 Pruebas cualitativas

De cada una de las cuatro imágenes originales se obtuvieron seis imágenes binarias del plano de caracteres, obtenidas según:

1) Una conversión a imagen binaria con el paquete de software DoDoT [5] que equivale a la aplicación de un nivel de umbral global. Aquí, DoDoT necesita el parámetro de tanto por ciento (%t) de la amplitud [0-255], a negro que es equivalente al T de la Ec. (3.6) por la relación:

$$T = \%t \cdot 255 \tag{5.1}$$

Este T se sitúa sobre el histograma respectivo de cada imagen original.

2) La imagen binaria resultante de la aplicación del algoritmo 1 por detección de bordes.

3) La imagen binaria de salida del algoritmo 2 por nivel lógico, con la segunda parte del algoritmo aplicando la Ec. 4.3 como se sugiere en [7].

4) Otra imagen binaria como resultado del nuevo algoritmo 2 por nivel lógico, con la segunda parte del algoritmo aplicando la Ec. 4.4, esto es, la nueva segunda fase propuesta del algoritmo.

5) La imagen binaria resultante de la aplicación del nuevo algoritmo 3 por medio del filtro de formas.

6) La imagen binaria resultante del algoritmo propuesto por Yasuda, Dubois y Huang (YDH) [26].

En las Figuras 5.6, 5.7, 5.8 y 5.9 se presentan las seis (6) imágenes binarias resultantes, una imagen por cada inciso anterior, para las cuatro imágenes originales de las Figs.5.1(a), 5.2(a), 5.4(a) y 5.5(a), respectivamente (que son las imágenes del documento, del bono, la imagen suavizado del bono y la imagen de la factura). En estas figuras, las imágenes binarias resultantes se presentan escaladas para suavizar fácilmente el resultado de los distintos algoritmos. Para escudriñar en los detalles finos, en los Apéndices IV, V, VI y VII se presentan las mismas imágenes binarias pero son escalarlas. En la Tabla 5.1 se muestran los valores de los parámetros usados para la obtención de cada imagen binaria.

5.2 Pruebas cuantitativas

En la Tabla 5.2 se presentan los resultados cuantitativos de la prueba con el OCR [21], donde se aplican las Ecs. (2.2) y (2.3). El parámetro n_b de caracteres reconocidos visualmente en la imagen binaria correspondiente, es un parámetro subjetivo y corresponde el número de caracteres que se "ven" en la imagen.

La tabla 5.3 evalúa los cuatro algoritmos propuestos con respecto a tres consideraciones: número de parámetros requeridos, memoria necesaria y tiempo de ejecución.

El valor del parámetro T_1 del algoritmo por detección de bordes se calcula automáticamente ajustándola al valor promedio del gradiente, por lo que sólo

Figura, Algoritmo Aplicado	**Valor de los parámetros**
Imagen del documento Fig. 5.1(a)	
5.6(a) Umbral global	50% a negro
5.6(b) Por detección de bordes	$T_1 = 50$, $T_2 = 20$
5.6(c) Por nivel lógico de [7]	$K = 13$, $T = 1$, $W = 1$
5.6(d) Nuevo nivel lógico.	$K = 13$, $T = 1$
5.6(e) Por filtro de formas	$K = 13$
5.6(f) YDH	$T_1 = 50$, $T_2 = 16$, $T_3 = 12$, $T_4 = 128$, $T_5 = 20$
Imagen del bono, Fig. 5.2(a)	
5.7(a) Umbral global	20% a negro
5.7(b) Por detección de bordes	$T_1 = 21$, $T_2 = 30$
5.7(c) Por nivel lógico de [7]	$K = 11$, $T = 30$, $W = 1$
5.7(d) Nuevo nivel lógico	$K = 11$, $T = 30$
5.7(e) Filtro de formas	$K = 11$
5.7(f) YDH	$T_1 = 50$, $T_2 = 16$, $T_3 = 40$, $T_4 = 128$, $T_5 = 20$
Imagen suavizada del bono, Fig. 5.4(a)	
5.8(a) Umbral global	20% a negro
5.8(b) Por detección de bordes	$T_1 = 10$, $T_2 = 20$
5.8(c) Por nivel lógico de [7]	$K = 11$, $T = 30$, $W = 1$
5.8(d) Nuevo nivel lógico	$K = 11$, $T = 30$
5.8(e) Filtro de formas	$K = 11$
5.8(f) YDHO	$T_1 = 50$, $T_2 = 16$, $T_3 = 100$, $T_4 = 128$, $T_5 = 20$
Imagen de la factura, Fig 5.4(a)	
5.9(a) Umbral global.	65% a negro
5.9(b) Por detección de bordes	$T_1 = 6$, $T_2 = 20$
5.9(c) Por nivel lógico de [7]	$K = 21$, $T = 30$ $W = 1$
5.9(d) Nuevo nivel lógico.	$K = 21$, $T = 30$
5.9(e) Filtro de formas	$K = 21$
5.9(f) YDH	$T_1 = 70$, $T_2 = 16$, $T_3 = 50$, $T_4 = 128$, $T_5 = 20$

Tabla 5.1: Valores de los parámetros usados en las imágenes de prueba para la obtención de las imágenes binarias de salida según el algoritmo aplicado.

Figura	Número de caracteres reconocidos visualmente n_b	Número de caracteres reconocidos por el OCR n_r	Razón de caracteres extraídos $C_E = \frac{n_b}{n}$	Razón de caracteres reconocidos $C_R = \frac{n_r}{n}$
Documento **Fig. 5.1(a)**	$n = 230$			
5.6(a)	66	33	0.287	0.143
5.6(b)	107	56	0.465	0.243
5.6(c)	229	184	0.996	0.800
5.6(d)	229	199	0.996	0.865
5.6(e)	229	192	0.996	0.834
5.6(f)	227	184	0.987	0.800
Bono **Fig. 5.2(a)**	$n = 61$			
5.7(a)	61	0	1.000	0.000
5.7(b)	46	0	0.754	0.000
5.7(c)	61	21	1.000	0.344
5.7(d)	61	22	1.000	0.360
5.7(e)	61	29	1.000	0.475
5.7(f)	61	19	1.000	0.311
Imagen suavizada del bono **Fig. 5.4(a)**	$n = 61$			
5.8(a)	61	43	1.000	0.705
5.8(b)	61	5	1.000	0.082
5.8(c)	58	30	0.951	0.492
5.8(d)	61	32	1.000	0.524
5.8(e)	61	35	1.000	0.574
5.8(f)	60	36	0.984	0.590
Factura **Fig. 5.5(a)**	$n = 16$			
5.9(a)	16	0	1.000	0.000
5.9(b)	16	6	1.000	0.375
5.9(c)	16	6	1.000	0.375
5.9(d)	16	7	1.000	0.438
5.9(e)	16	6	1.000	0.375
5.9(f)	16	0	1.000	0.000

Tabla 5.2: Resultados de las pruebas estadísticas de las imágenes de salida de los distintos algoritmos

Algoritmo	Número de parámetros	Cantidad de memoria	Tiempo de ejecución (seg)
Por detección de bordes	1	$M(N+2)$	14.7
Nuevo nivel lógico	2	$(K+5)M$	8.9
Filtro de formas	1	$(K+9)M$	7.9
YDH	5	$M(N+2)$	26.0

Tabla 5.3: Evaluación de cuatro algoritmos. La memoria se mide como el número de registros necesarios para almacenar resultados intermedios. M, N es el número de columnas y renglones, respectivamente, de la imagen procesada. K es el ancho de la ventana para calcular los promedios. El tiempo de ejecución fue medido sobre la imagen de la factura.

es necesario un solo parámetro 5.3: el valor del largo máximo de un caracter.

La cantidad de memoria es una medida importante para aplicaciones a pequeña escala y bajo costo, como la realización en dispositivos (scanners) y en computadoras personales. Aquí se toma la cantidad de memoria requerida como el número de registros (posiblemente bytes) para almacenar los resultados intermedios en un esquema donde la imagen se lee con un proceso de escaneo óptico por líneas.

El tiempo de ejecución es una medida proporcionalmente inversa a la velocidad de los algoritmos. Esta medición se efectúo sobre la imagen de la factura en una máquina PC 386-DX, con una unidad coprocesadora, a 25 MHz.

5.3 Análisis de los resultados

En esta sección revisaremos las cualidades de cada algoritmo, ventajas y desventajas.

5.3.1 Resultado de la aplicación de un umbral global

Este algoritmo se describió en la sección 3.3.1. Se compara porque es la manera más simple para segmentar una imagen.

Esta técnica se considera mala, aunque las imágenes de las Figs. 5.7(a), 5.8(a) y 5.9(b) presentan mejores resultados visuales, y la Fig. 5.8(a) es la mejor para el reconocimiento, con la razón de caracteres reconocidos $C_R = 0.705$, para el caso de la imagen suavizada del bono; pero para las otras tres imágenes originales resultó el peor, $C_R = 0.143$, para la imagen del documento y $C_R = 0.000$ para el bono y la factura. Esto es debido, como es de suponerse, a que los histogramas de las imágenes, Figs. 5.1(a), 5.2(b), 5.4(b) y 5.5(b), no presentan formas bimodales donde se pueda establecer el valor del umbral para la segmentación, y además a que no sólo los caracteres son obscuros como la figura del fondo en la imagen del bono de la Fig 5.2(a).

También se considera mala esta técnica porque para la generación de la imagen de salida se interactuó directamente con el programa DoDoT [5] para la sintonización del porcentaje de conversión a imagen binaria, ó lo que equivale a encontrar el parámetro T de la Ec. 3.6. En otras palabras, no hay un método general aplicable para la obtención del nivel de umbral para la binarización de la imagen que separe los caracteres y que se pueda aplicar en todos los casos.

5.3.2 Resultados del algoritmo por detección de bordes

Este algoritmo aunque está soportado por fuertes bases matemáticas, no resulta bueno; ni en las apariencias ni, obviamente, para el reconocimiento, como se ve en las Figs 5.6(b), 5.7(b), 5.8(b) y 5.9(b). Esto es debido a que las derivadas amplifican el ruido, dando un fondo “manchado” y caracteres cortados. Además, fue incapaz de extraer los caracteres en la parte obscura de la imagen del documento, ver Figs. 5.6(b). Un suavizado previo, que elimina el ruido, mejora la imagen de salida como se observa comparando las imágenes de las Figs 5.7(b) y 5.8(b) (las generadas del bono y la suavizada del bono).

Este algoritmo resultó malo para el reconocimiento, con la razón de caracteres reconocidos, $C_R < 0.250$, para las imágenes generadas del documento, bono y bono suavizada. El $C_R < 0.375$ para la imagen generada a partir de la factura (Figs. VII.3) representa sólo 6 caracteres reconocidos de 16, y es debido a que aquí los trazos de los caracteres son más gruesos.

Tal vez la aplicación de otros esquemas para la obtención de las derivadas, como los presentados en [7] puedan mejorar la imagen de salida pero no al grado que cualquiera de los otros algoritmos. La forma de obtención de las derivadas aquí aplicadas es la más simple; de otra manera de obtenerlas complicaría este algoritmo y se considera que no es necesario dado que los otros algoritmos presentan mejores resultados.

Optimizando el uso de memoria, suponiendo una imagen de tamaño $M \times N$, se requerirían aproximadamente $M(N+2)$ registros (ver Tabla 5.3). Se guarda toda la imagen original en MN registros. Se pasan los operadores Sobel para calcular el parámetro T_1 (igual al promedio del gradiente). Sobre cada pixel de la imagen original se vuelve a calcular el gradiente y el Laplaciano para aplicar la Ec. 3.16 y generar la imagen en tres tonos. Aquí es necesario contar con 2M registros para almacenar los resultados antes de salvarlos en los registros donde se almacenó la imagen original (ver Fig. 5.2). Finalmente se debe tener toda la imagen de tres tonos para aplicarle el algoritmo de binarización.

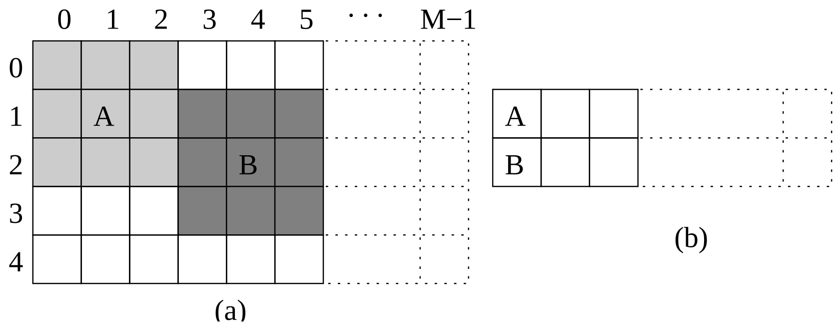

Figura 5.2: Filtraje espacial con una máscara de 3×3 usando registros intermedios. (a) Fragmento de una imagen. Sobre de ella se muestran dos máscaras de tamaño 3×3. M es el número de columnas. (b) Registros intermedios, de tamaño total $2M$. Los resultados del primer paso de la máscara se almacenan en el renglón A de (b). Los resultados de la segunda máscara se almacenan en el renglón B de (b). El tercer paso de la máscara será sobre los renglones 2, 3 y 4 de la imagen, desocupando el renglón 1 de la imagen. Los resultados en el renglón A de (b) se pueden salvar ahora en el renglón 1 de la imagen y los resultados del tercer paso de la máscara se almacenan en el renglón A de (b).

Ventaja: Requieren un solo parámetro.

Desventajas: No es aplicable en el caso de que la imagen presente sombras o mala iluminación. Requiere una cantidad de memoria, aproximadamente igual al tamaño de la imagen original.

5.3.3 Resultados de la aplicación del algoritmo por nivel lógico

Para este caso se hizo además un estudio comparativo entre el algoritmo del nivel lógico sugerido en [7] y el nuevo algoritmo propuesto. Esto para comprobar que el nuevo algoritmo es mejor.

En las Figs. 5.6(c), 5.7(c), 5.8(c) y 5.9(c) se utilizó el algoritmo sugerido en [7]. En las Figs. 5.6(d), 5.7(d), 5.8(d) y 5.9(d) se utilizo el nuevo algoritmo. Los resultados visuales son mejores. El nuevo algoritmo refuerza los trazos de las letras, aunque pueda aumentar también algún ruido del fondo, y elimina el parámetro W que se hace innecesario. Las pruebas con el OCR de la Tabla 5.2 también apoyan esta aseveración, mostrado gráficamente en la Fig. 5.3.

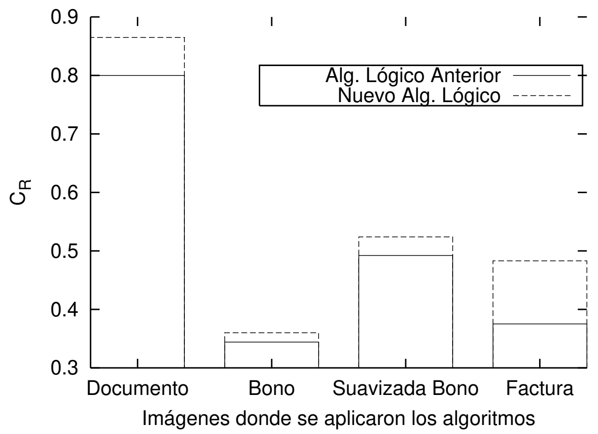

Figura 5.3: Gráfico comparativo de la eficacia para el reconocimiento por el OCR, entre en el algoritmo lógico anterior propuesto en [7] y el nuevo algoritmo lógico sugerido.

La cantidad de memoria que podría utilizar el nuevo algoritmo lógico es aproximadamente $(K + 5)M$ registros, K es el ancho de la ventana para calcular los promedios, M el número de columnas de la imagen original. El filtro lógico se aplica con una máscara de tamaño 3×3 (ver Fig. 5.4), por lo que sólo es necesario cargar $K+2$ renglones de la imagen original. Se necesita además $3M$ registros para almacenar los resultados de los promedios.

La desventaja de este algoritmo es que se llega al mismo esquema de encontrar un nivel de umbral para la obtención del valor de T, el valor presentado en la Tabla 5.1 se encontró con ayuda de DoDoT (que tiene una

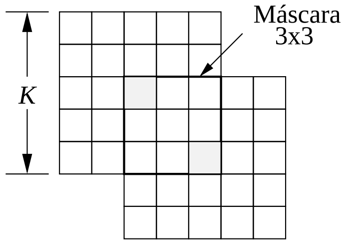

Figura 5.4: Definición de la cantidad mínima de registros que podría utilizar el algoritmo por filtro lógico. Aqui $K = 5$.

utilería para dar el valor RGB de un pixel) sobre la imagen realizada con $T = 0$.

La aplicación de este algoritmo mejora si se aplica un suavizamiento a la imagen de entrada, cuando esta presenta ruido, por lo que también es una desventaja ya que incrementa el tiempo de procesamiento.

La ventaja de este algoritmo es que dio mejores resultados para la prueba con el OCR. También resulta menos complicado que el algoritmo YDH [26], sólo lo forman dos fases y a aquel cuatro, si se quiere utilizar para comprimir imágenes para bases de datos. Es que requiere menor cantidad de memoria ya que utiliza valores locales, viable para aplicaciones en PC. Es un algoritmo rápido, solo detrás del algoritmo por filtro de formas.

5.3.4 Resultados de la aplicación del algoritmo por filtro de formas

Este algoritmo es nuevo y es el mejor. Requiere un sólo parámetro K, que es el ancho de la ventana para la obtención de los promedios alrededor del pixel central y no resulta crítico; debe ser mayor, o mucho mayor, que el ancho del trazo de los caracteres (en la Fig. 4.4, en la pág. 36, se define el ancho del trazo). Su ventaja es que se elimina encontrar el valor del parámetro T del algoritmo por nivel lógico. La fuerza de este algoritmo es precisamente en aprovechar las discontinuidades de la imagen original, esto es, no se necesita aplicar un suavizamiento inicial (comparar las imágenes de las Figs. 5.7(e), 5.8(e) y puede utilizarse directamente para la comprensión de imágenes para bases de datos. Es el algoritmo con menor tiempo de procesamiento. Requiere poca memoria, al igual que el algoritmo por filtro de formas, porque utiliza valores locales. En la Fig. 5.5 se define la máscara del filtro de formas (Fig. 4.10) junto con las máscaras de tamaño $K \times K$ para extraer los promedios; se utilizan $K+4$ renglones de la imagen original y $5M$ registros para almacenar el resultado de la Ec. 4.5, un total de $(K+9)M$ registros.

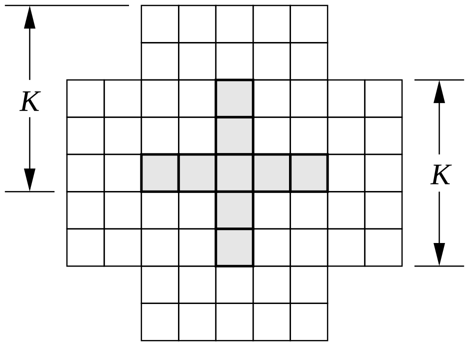

Figura 5.5: Definición de la cantidad mínima de registros que podría utilizar el algoritmo por filtro de formas. Se representa remarcada la forma de la máscara de la Fig. 4.10. K tiene el valor de 5.

En la prueba con el OCR fue el algoritmo mejor. Para la imagen del documento y del bono se obtuvieron los C_R mayores, iguales a 0.834 y 0.475, respectivamente. Para la imagen del bono suavizado quedó abajo del algoritmo YDH, 0.574 contra 0.590, una diferencia de 0.024 que corresponde a tres caracteres reconocidos. Para la imagen de la factura quedo abajo del nuevo algoritmo por nivel lógico pero solo corresponde a un solo caracter reconocido de diferencia (6 de 16 contra 7 de 16 por el nuevo algoritmo lógico).

Su desventaja es que no remueve completamente el ruido del fondo de la imagen y que no se debe aplicar a imágenes suavizadas, como se ve en la imagen de la Fig. 4.8(e) y esto afecta directamente el coeficiente C_R menor que es del YDH para la imagen del bono suavizado.

5.3.5 Resultados de la aplicación del algoritmo propuesto por Yasuda-Dubois-Huang

Se verificó como es propuesta en [26], que sólo se necesita sintonizar dos parámetros T_1 y T_3; T_2 es fijo e igual a 16 y en las pruebas se encontró que los valores para T_4 y T_5 iguales a 128 y 20, respectivamente, no se necesitan cambiar para obtener mejores resultados, como es sugerido en [26].

La imagen de salida mejora si se le aplica previamente un suavizado, cuestión que ya debería considerarse en el paso 1 del algoritmo.

La cantidad de memoria que podría utilizar este algoritmo es igual que la utilizada por el algoritmo por detección de bordes, aproximadamente un número de registros igual al tamaño de la imagen original.

Ventajas: El significado de los parámetros T_1 y T_5 es claro: T_1 es el valor máximo de los pixeles que serán suavizados y T_5 es el valor máximo de la amplitud, dentro de una ventana de 16×16, para considerar los pixeles dentro de esa ventana como pixeles de fondo. Da buenos resultados para el reconocimiento, para la imagen del documento dio un $C_R = 0.800$, igual que el C_R del algortimo del filtro lógico pero abajo de los algoritmos nuevos propuestos; para la imagen del bono tuvo un $C_R = 0.311$, abajo de los algoritmos por nivel lógico y por el filtro de formas; para la imagen suavizada del bono obtuvo el C_R mayor e igual a 0.590, un poco más del 0.574 del filtro de formas.

Desventajas: Se presentan bordes falsos por las ventanas de 16×16, como se ve en la Fig. 4.6(f) (efecto conocido como "tablero de damas"). No se aplica para el reconocimiento para imágenes muy ruidosas como la Fig. 4.9(f) (factura). Es el que conlleva más pasos (cuatro) y por consiguiente mayor tiempo de cómputo. Requiere mucha memoria (se necesita cargar toda la imagen original).

Capítulo 6

CONCLUSIONES Y CONTRIBUCIONES

En este trabajo se han revisado seis algoritmos para la segmentación de imágenes de documentos que presentan mala calidad. Los algoritmos separan las posibles letras del fondo, en imágenes en 256 tonos de gris, y dan como resultado imágenes binarias, osea en blanco y negro; es un paso previo para el Reconocimiento Óptico de Caracteres (OCR). También se pueden utilizar estos algoritmos para la comprensión de imágenes de documentos para bases de datos y/o para transmisión.

El preprocesamiento de la imagen de entrada sólo es necesario cuando la imagen presenta ruido. La eliminación de este ruido se realiza eficazmente con un simple suavizamiento: dos pasadas de un filtro de promedios con una máscara de tamaño 3×3.

De las pruebas realizadas con el OCR, dos algoritmos son los mejores: el algoritmo modificado por nivel lógico y el nuevo algoritmo por filtro de formas. El algoritmo propuesto por Yasuda-Dubois-Huang [26] se considera bueno. El algoritmo por nivel lógico propuesto en [7] se considera regular. Los algoritmos de segmentación por nivel de umbral global y por detección de bordes se consideran malos.

Se reduce el número de parámetros necesarios para la aplicación de los algoritmos: el algoritmo modificado por nivel lógico requiere solo dos parámetros; los algoritmos por filtro de formas y por detección de bordes requieren solo uno. En comparación, el algoritmo por nivel lógico sugerido en [7] necesita tres parámetros, y el de Yasuda-Dobois-Huang necesita cinco. Los algoritmos modificado por nivel lógico y por filtro de formas comparten un parámetro que no resulta crítico: el ancho de la ventana donde se toman los promedios en la vecindad de un pixel.

El nuevo algoritmo por filtro de formas no requiere que la imagen de entrada sea preprocesada.

Una característica fuerte para hallar caracteres en documentos en tonos de gris, es la detección de la diferencia de contraste entre el trazo de un caracter y el fondo, tomando la diferencia del promedio alrededor de un pixel menos el valor del mismo pixel; esto bajo la suposición de que el trazo del caracter es más obscuro que el fondo, como sucede en la inmensa mayoría de los documentos. Esta característica es aprovechada en los dos algoritmos sugeridos: un algoritmo modificado por nivel lógico y otro algoritmo nuevo por un filtro de formas.

Mejorar la eficiencia de los algoritmos propuestos conlleva tratar el sistema global del reconocimiento automático de caracteres que se sugiere en la sección 2.2. Para esto es necesario utilizar un esquema de retroalimentación en todo el sistema y que deberá involucrar sistemas mixtos de Sistemas Expertos, para la toma de decisiones, y probablemente Redes Neuronales, para el reconocimiento de patrones. La aplicación de uno, o varios, de los algoritmos estudiados se situaría en los pasos iniciales. Para esto podría servir la idea del filtro de formas: los trazos de los caracteres se suponen llenos de "primitivas" se extiende esta idea a grupos de "primitivas", a orientaciones de "primitivas" hasta darles significado para cada caracter o reconocer agrupaciones de primitivas con una red neuronal. Se pudiese extender aún más a detectar palabras y/o frases para recuperar los caracteres borrosos.

Existen otros tres algoritmos, considerados malos, reportados en [7]: la técnica de función no lineal , por contraste local y por substracción basada en máscaras, que para su aplicación requieren de 1276, cinco y tres parámetros, respectivamente. Las primeras dos técnicas tienen parámetros difíciles de sintonizar. La técnica de substracción basada en máscaras la reportan en [7] como nueva, pero es de difícil programación. De cualquier manera, los dos nuevos algoritmos propuestos aquí obtienen mejores resultados que cualquiera de esos tres algoritmos (como se consta cualitativamente en las imágenes de prueba con la imagen del documento, Figs 5.6(c) y 5.7(c) y del Apéndice IV, que es la misma usada en [7], además de ser más fáciles en la sintonización de los parámetros y en la codificación.

Se intentó aplicar un filtraje en el dominio de la frecuencia (usando la Transformada Rápida de Fourier) para resolver el problema de la segmentación pero no se logró un resultado concreto. Por otro lado, resulta muy

Figura 6.1: El primer renglón muestra unos caracteres trazados en base de sus bordes, el segundo renglón muestra unso caracteres cuyos trazos no son continuos, los dos últimos renglones muestran caracteres cuyos trazos son “difíciles”

caro computacionalmente el uso de la FFT, requiere operaciones con números reales y gran cantidad de memoria para almacenarlos.

Primeramente en este trabajo se pensó en la segmentación por texturas. Esta técnica la utilizan en [25] y [24] para resolver el problema de separación de dibujos del texto y/o separar regiones con distintos tamaños de letras, en imágenes en tonos de gris. En [25] separan los distintos componentes de una hoja normal de periódico: texto de encabezados, texto de columnas e imágenes y en [24] separan el texto de las figuras en imágenes de manuales de usuario. En estos esquemas, se podría utilizar algún algoritmo, por ejemplo, el del filtro de formas, como paso inicial como sustitución al nivel de umbral adaptativo que proponen, para aislar los trazos de los caracteres.

Queda muy lejana, o insoluble, una solución general al reconocimiento de caracteres de cualquier documento si se considera la complejidad de la escritura humana: caracteres muy parecidos como son la O, Q, 0 ó i, I y 1; los trazos de las letras (ver Fig. 6.1), tipos de letras; colores posibles de los trazos de las letras, orientación de la escritura (comúnmente horizontal); y los posibles problemas en un documento que se describen en la sec. 1.1. Este problema equivaldríaìa al de hacer una máquina que leyera cualquier clase de documentos para personas invidentes.

Apéndice A

Formato para Archivos de Imágenes TGC

Un formato de archivo es la especificación de la estructura del mismo. Cada paquete de cómputo maneja su propio formato para almacenar los datos y existen cientos de formatos; para bases de datos, procesadores de texto y archivos gráficos, algunos de estos han sido estandarizados para intercambio de datos entre distintos sistemas [3].

Los formatos para archivos de imágenes, en general, se pueden dividir en dos grupos [9, 17, 18]:

- Archivos para gráficos de vectores, y
- Archivos para gráficos por mapa de bits.

El primero es usado por el CAD (Computer-Aided Design) y paquetes de dibujo, ejemplos: AutoCAD y Corel Draw. El segundo es crado por scanner y cámaras y también por paquetes de dibujo; ejemplo: PC PaintBrush, donde la imagen es creada en una área de memoria reservada llamada mapa de bits, con algún número de bits correspondiendo a cada uno (o pixel). En los sistemas más simples monocromáticos, un bit representa un punto, o sea, apagado o encendido. Para escala de grises, varios bits son necesarios para representar el número de sombra de intensidad de rojo, verde y azul de cada punto.

Alguno formatos de archivos gráficos comerciales más usados son:

GIF (Graphics Interchance Format).- Creado por CompuServe para usarlo en el almacenamiento de imágenes escaneadas.

WPG De WordPerfect Graphics.

BMP El formato de mapa de bit usado por Microsoft Windows 3 y para el sistema operativo OS/2.

PIC Formato usado por el software de pintado PC Paint/Pictor. Este formato es diferente, aunque mantiene la misma extensión de archivo, que el PIC de los gráficos de Lotus 1-2-3.

TGA de Truevision Targa, el cual es usado en aplicaciones en color de alto nivel, especialmente en retoque y separación de colores de fotografía profesional digital. Soporta imágenes RGB de 24 bits y de 32 bits.

PCX Originado por Z-Soft para su paquete PC Paintbrush.

TIFF (Tag Image File Format) El formato de archivo de imágenes por etiquetas (tags), un formato muy general no especifico para cualquier paquete y que es usado comúnmente para intercambio de información.

IFF/LBM Este formato es nativo para la Commodore Amiga. Cuando Electronics Arts, el creador de la aplicación para la Amiga: Deluxe Paint, portó su programa de aplicación a la PC, cambió su extensión de archivo a LBM.

FITS (Flexible Image Transport System).- Usado para almacenamiento e intercambio de imágenes astronómicas.

JPEG (Join Photographic Expert Group).- El mejor formato, por mucho, para comprensión de imágenes fotográficas.

DXP (Drawing Interchance Format).- Formato para imágenes en ASCII y vectores binarios. Usado para intercambiar datos de imágenes CAD. Aplicado en la mayoría de las aplicaciones en PC para CAD: AutoCAD, FastCAD, Generic CADD, y otras aplicaciones como Corel Draw, Pagrmaker.

A.1 Formato TGC

Este formato se realizó para simplificar el acceso, y por lo tanto hacerlo más rápido para su posterior procesamiento, de un archivo de imagen en mapa

Bytes	Descripción
0-2	Cadena: "TGC"
3	'\0'. Caracter nulo.
4-5	Versión (03 desde el 5 de sep. de 1993)
6-7	Largo de la imagen
8-9	Ancho de la imagen en pixeles
10-11	Tipo de la imagen: 0.- Binaria. Blanco es cero. 1.- Binaria. Negro es cero. 2.- 16 tonos de gris 3.- Paleta de 16 colores 4.- Paleta de 256 colores 5.- 256 tonos de gris 6.- RGB 7.- Matriz de enteros (2 bytes) 8.- Matriz flotantes (4 bytes) 9.- Matriz de dobles (8 bytes) 10.-32 tonos de gris 11.- 64 tonos de gris 12.- 128 tonos de gris
12-15	4 bytes para extensiones futuras.

de bits. TGC desde la versión 1 incluye además el almacenamiento de datos de matrices.

TGC son las iniciales de Tere-Gerardo-Clara que desarrollan la tesis de maestría en el INAOE, 1993.

La imagen es almacenada en código binario con un pequeño encabezado de 16 bytes.

TGC se utilizó primero sobre PC's que utilizan microprocesadores Intel de la serie 80xxx por lo que se usa la convención Little-Endian, esto es, el byte bajo se almacena primero.

A partir del byte 16 se encuentra la imagen almacenada por renglones.

- Los tipos 0-1 se almacena un bit por píxel.
- Los tipos 2 y 3 se almacenan 4 bits por pixel.
- Los tipos 4, 5, 10, 11 y 12 son de 1 byte por pixel.

- El tipo 6 son 3 bytes por pixel.

TGC no maneja aún compresión de datos hasta su versión 03

Los tipos 8 y 9 se crearon para almacenar el resultado de la transformada de Fourier.

La versión 00 y 01 de TGC no manejan las imágenes monocromáticas binarias.

La estructura del encabezado de un archivo TGC es la siguiente:

```
typedef struct {
    char name [4];
    unsigned int version;
    unsigned int witdh;
    unsigned int depth;
    unsigned int tipo;
    unsigned int compress;
    unsigned int anexo;
} TGCHEADER;
```

Para las imágenes tipo 3 y 4, la paleta de colores, en forma RGB de tres bytes por color, se encuentra al final de los datos de la imagen. Para accesarla directamente se puede usar la instrucción en C:

```
fseek( fp_in, ( 16L+(long)depth*(long)bytes), SEEK_SET);
```

donde el número de bytes se calcula según el tipo:

```
switch(tipo) {
    case 0:
    case 1:
            bytes= ( (width+7)/8)<<2;
            break;
    case 2:
    case 3:
            if ( width%2) bytes = width/2+1;
            else bytes=width/2;
            break;
    case 4:
    case 5:
    case 10 :
```

```
        case 11:
        case 12:
                bytes = width;
                break;
        case 6:
                bytes = width*3;        /**tipo=6, RGB **/
    }
```

Apéndice B

Lista de Programas Desarrollados

B.1 Programas para soporte en el manejo de las imágenes TGC

Todos esos programas fueron desarrollados sobre la plataforma PC, y permiten la visualización de una imagen TGC, conversión entre TGC a TIFF y viceversa, y listar sus características.

ltgc.exe

Sintaxis: ltgc [path]nombre
El nombre del archivo sin extensión. Se supone con extensión TGC.
Uso: Lista las características de un imagen TGC:

- Versión de TGC
- Largo y ancho de la imagen en pixeles
- Tipo de imagen

dtgc.exe

Sintaxis: dgtc[path]nombre
El nombre del archivo sin extensión. Se supone con extensión TGC.
Uso: Despliega en pantalla una imagen TGC. El programa cuenta con un manejador de memoria [18] y driver para la tarjeta Super VGA Tseng [22], hasta

una resolución de 640×480 pixeles a 256 colores (tseng.drv), y hasta una resolución de 800×600 pixeles a 256 colores (tseng2.drv).

El manejador de memoria permite guardar la imagen en una zona conocida. Se usa la memoria convencional, o la extendida o virtual en disco duro, la que tenga la capacidad suficiente para contener la totalidad de la imagen.

También se cuenta con el desplegador en el modo 13 del adaptador de video VGA estándar, con la resolución de 320×200 pixeles a 256 colores.

tif.tgc.exe

Sintaxis: tig-tgc [path]nombre
El nombre del archivo sin extensión. Se supone con extensión TIF.
Uso: Escribe un archivo TGC a partir de uno TIFF. El archivo resultante quedará en la dirección especificada por [path] y tendrá el mismo nombre con la nueva extensión TGC.

tgc-tif.exe

Sintaxis tgc.tif[path]nombre El nombre del archivo sin extensión. Se supone con extensión TGC.
Uso: Escribe un archivo TIFF a partir de uno TGC. El archivo nuevo tendrá el mismo nombre pero con extensión TIF y quedará almacenado en la dirección especializada por [path].

ltiff.exe

Sintaxis: ltiff [path]nombre El nombre del archivo sin extensión. Se supone con extensión TIF.
Uso: Lista las etiquetas y valores de un archivo TIFF.

distgc.exe

Sintaxis: distgc
Uso: Es un manejador para el desplegador dtgc.exe Requiere, el mismo subdirectorio donde está el distgc.exe, un archivo distgc.cfg donde se almacena el camino (path) y nombre del archivo dtgc.exe. Ejemplo: `c:\tgc\dtgc.exe`

Si se requiere usar el dtgc con el driver de video para Super VGA es necesario además el archivo et4000.cfg donde se especifica el camino y el nombre del driver de video. Ejemplo:

`c:\tgc\driver\tsend.drv`

para una resolución de hasta 640×480 pixeles a 256 colores, ó

`c:\tgc\driver\tseng2.drv`

para una solución de hasta 800×600 pixeles a 256 colores.

Se puede cambiar de drive con la tecla "I" ó "L" seguido de la letra del drive correspondiente (a,b,c, etc.)

Restricciones: Sólo soporta 64 nombres de imágenes TGC en el mismo directorio, incluyendo los nombres de los subdirectorios.

B.2 Programas involucrados en los algoritmos

En la siguiente descripción se usará:

{Entrada}.- Para indicar el camino y/o nombre de un archivo de imagen TGC, puede ir con o sin extensión.

{Salida}.- Para indicar el camino y/o nombre de un archivo de imagen TGC, puede ir con o sin extensión.

Lo indicado entre corchetes [ ] es opcional.

A menos que se indique lo contrario, cada programa involucra los programas fuente con el mismo nombre que el ejecutable, con extensión %.c por supuesto, y a matrix.c.

B.2.1 Algoritmo 1. Por detección de bordes

grad[.exe] {Entrada} {Salida}

Uso: En {Salida} se almacena el gradiente de la imagen {Entrada} según los operadores Sobel de la Fig. 4.2 y la Ec. 3.10.

laplaz[.exe] {Entrada} {Salida}

Uso: Da en {Salida} la imagen del Laplaciano de la imagen {Entrada} calculada según la Ec. 4.1.

umbral[.exe] **{Entrada} {Grad} {Salida}**

Donde {Grad} es el gradiente de la imagen de {Entrada}, previamente calculado en grad.exe.

Uso: Genera una imagen en tres tonos según la Ec. 4.2. Se pide como dato el valor del parámetro T_1.

bin3[.exe] **{Entrada} {Salida}**

Uso: Genera una imagen en dos tonos (blanco y negro) de la imagen de salida, en tres tonos, del programa anterior. {Salida} esta a 1 byte/pixel para cambiarla a 1 bit/pixel su usa posteriormente el programa binariza.exe.

binariza[.exe] **{Entrada} {Salida}**

sl Uso: Cambia o comprime, la imagen {Entrada} de 1 byte/pixel a la imagen {Salida} en 1 bit/pixel. El tono 255 de {Entrada} se cambia a 1, y todos los demás tonos de {Entrada} son puestos a 0, generando una imagen TGC tipo 1 (uno).

B.2.2 Algoritmo 2. Por nivel lógico

promedios[.exe] **{Entrada} {Salida}**
Uso: Realiza el paso 1 de este algoritmo. {Salida} se genera según la Ec. 4.5, pero el 0 vale 255 y el 1 vale el resultado de la operación de restar el promedio menos el punto central ({Salida} está en 1 byte/pixel). El programa pide como dato el valor de T.

logico[.exe] **{Entrada} {Salida}**

Uso: Paso 2. Filtro lógico según la Ec. 4.1.

logi2[.exe] **{Entrada} {Salida}**

Uso: Paso 2. Filtro lógico mejorado según la Ec. 4.4

La imagen {Salida} está en 1 byte/pixel. Es necesario para el reconocimiento aplicar todavía el programa binariza.exe.

B.2.3 Algoritmo 3. Por filtro de formas

El paso 1 de este algoritmo es aplicando **promedios.exe** con T igual a 0 (cero).

primi[.exe] **{Entrada} {Salida}**

Uso: Paso 2.Filtro de formas según la Ec. 4.9

La imagen {Salida} está en 1 byte/pixel. Es necesario para el reconocimiento aplicar todavía el programa binariza.exe.

B.2.4 Algoritmo 4. Técnica Yasuda-Dobois-Huang.

ydh[.exe] **{Entrada} {Salida}**

Uso: Procesa {Entrada} según los pasos de este algoritmo (Seccion 4.4). El programa pide como datos los parámetros T_1,T_3,T_4 y T_5

.

La imagen {Salida} está en 1 byte/pixel. Es necesario para el reconocimiento aplicar todavía el programa binariza.exe.

B.3 Otros programas realizados

Aquí se incluyen unos programas para mejorar la calidad de la imagen, para modificar el número de tonos de gris y donde se aplica la transformada rápida de Fourier a imágenes TGC.

pasabaja[.exe] **{Entrada} {Salida}**

Aplica dos pasadas a la imagen {Entrada} del filtro espacial de promedios dado por la máscara siguiente:

$$\frac{1}{9} \text{ x } \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

el resultado se almacena en la imagen {Salida}.

nivela[.exe] **{Entrada} {Salida}**

Realiza la nivelación o igualación del histograma de la imagen {Entrada} según la Ec. 3.3.

con32[.exe]**{Entrada}{Salida}**
con64[.exe]**{Entrada}{Salida}**
con128[.exe]**{Entrada}{Salida}**

Estos tres programas generan una imagen {Salida} a 32, 64 y 128 niveles de gris, respectivamente, a partir de la imagen {Entrada} que está en 256 tonos de gris, usando la técnica de igualación de histograma de la Ec. 3.3

grafica[.exe]**{Entrada}{Salida}**

Este programa despliega en pantalla el histograma de una imagen. El formato es muy simple, (es un archivo de texto en DOS):

El primer número almacenado es el número de datos del histograma (o número de niveles de gris). Después van en pares el índice (o tono de gris) y su ocurrencia como un valor real. Este programa fue desarrollado con código fuente de [14]. Ejemplo del formato de un archivo donde se almacena el histograma (es con extensión HST):

```
128
0 1.0000
1 23.0000
2 20.0000
⋮  ⋮
127 0.0000
```

fft_tgc[.exe]**{Entrada}{Salida Real}{Salida Imaginaria}**

Calcula la transformada discreta de Fourier de la imagen {Entrada} por medio del algoritmo de la transformada rápida de Fourier, transladada medio periodo, dando la localización de las bajas frecuencias en el centro de la imagen generada. Da como resultado dos matrices en tipo flotante (números reales), en {Salida Real} se almacena la matriz de números reales y en Salida Imaginaria se almacena la matriz de números imaginarios.

Programas fuente: fft_tgc.c, fft.c, matrix.c.

ifft[.exe]**{Entrada Real}{Entrada Imaginaria}{Salida}**

Calcula la transformada inversa discreta de Fourier, aplicando la FFT en una dimensión.

{Entrada Real} y {Entrada Imaginaria} son los nombres de las matrices real e imaginaria.

Programas fuente: ifft.c, fft.c, matrix.c.

spectrum[.exe]{Entrada Real}{Entrada Imaginaria}{Salida}

Visualiza el espectro de Fourier creando la imagen {Salida} a 256 tonos de gris. Se aplica una comprensión en amplitud dinámica con la siguiente transformación de intensidad:

$$s = c\log(1 + |r|)$$

donde c es una constante de escalamiento, y la función logaritmo realiza la comprensión deseada. Como la función logaritmo no está definida para el valor de cero (0), se suma uno (1) para prevenir el error.

Se ilustra la selección del valor de c con un ejemplo, supóngase que se tiene un espectro de Fourier de una imagen en la amplitud [0, R]= [0, 2.5×10^6]. Los valores de log(1+ —r—), en ese caso, van de 0a 6.4. Y se desea escalar esta amplitud hasta [0, L-1]=[0, 255] para desplegarlo en un sistema de 8-bits, por lo tanto se selecciona el factor de escalamiento $c = 255/6.4$. Esto incrementa de modo significativo los detalles visibles de la imagen.

Programas fuente: spectrum.c, matrix.c.

Además se realizaron las funciones para calcular las estadísticas de segundo orden por el método de suma y diferencia de histogramas [23].

Programas fuente: **es2d.c** y **estatis.c.**

Apéndice C

Funciones Básicas para Imágenes TGC

El tipo de dato básico, incluido de encabezado **matrix.h,** para manejar las imágenes es el siguiente.

```
typedef struct {
        int element_size;
        unsigned int rows;
        unsigned int cols;
        char **ptr;
} MATRIX
```

Donde:

element_size = el número de bytes por dato.

rows= el número de renglones

cols = el número de columnas

ptr= apuntador genérico a un arreglo de apuntadores. ANSI recomienda guardandarlo en tipo void, pero la plataforma SUN usada manejaba la versión de Kernighan & Richie y por lo tanto quedó como tipo char ([10, pág. 220])).

Las primitivas para el manejo de las imagenes TGC, se incluyen en el archivo **matrix.c**, son:

MATRIX* matrix_allocate (int rows, int cols, int element_size);

Reserva la memoria para una imagen.

Entradas: Número de renglones (rows), número de columnas (cols) y de element_size bytes por dato.

Salida: Apuntador a tipo MATRIX si se lleva a cabo. El programa termina si falla la reservación de memoria.

void matrix_free(MATRIX *A);

Libera la memoria reservada para una imagen.

Entrada: Apuntador a tipo MATRIX

Salida: Ninguna

MATRIX *matrix_read(TGCHEADER *htgc, char *name);

Lee una imagen de disco.

Entradas: Apuntador a la estructura del encabezado de una imagen TGC. Apuntador al nombre de la imagen.

Salida: Apuntador al tipo MATRIX donde se almacenó la imagen si se llevó a cabo. Un apuntador a NULO si falla la lectura del archivo en disco o si no es una imagen TGC. El programa termina si falla la reservación de memoria.

void matrix_write(MATRIX *OUT, TGHEADER *htgc, char *name);

Escribe una imagen de disco.

Entradas: Apuntador a tipo MATRIX dinde se almacena la imagen. Apuntador al encabezado de imagen TGC. Apuntador al nombre de la imagen a guardar.

Salida Ninguna. El programa termina si falla la escritura en disco.

En esta función, las imagenes de entrada esta definida a 1 byte/pixel, y la máscara esta definida con enteros, o sea 2 bytes/pixel permitiendo el uso de números positivos y negativos. La imagen de salida está tambien a 1 byte/pixel y para que quede dentro de la amplitud del desplegado (0 a 255), además de dividirse el resultado de la convolución (3.7) entre el factor de escala, se toma su valor absoluto y si este es mayor que 255, se pone a 255. Se utilizó así para reducir la cantidad de memoria utilizada y para hacer la imagen de salida visible.

La función anterior es tal vez la que más se usó, así como se presenta o con modificaciones.

void strmfe (char *new, char *old, char *ext);

Función de cadenas de caracteres. Pega la extensión al nombre del archivo, no importando si el nombre ya tiene extensión.

Entradas: Apuntador al buffer donde se almacenará el nombre con la extensión. Apuntador al nombre del archivo (puede estar incluido el path). Apuntador a los caracteres de extensión de archivo.

Salida: Ninguna.

Esta función es util cuando se trabaja con parámetros de línea y asegura que los archivos de entrada o salida tengan la extensión correcta.

Con excepción de las últimas dos funciones todas las demas son dependientes de la plataforma en que se trabaje. La diferencia es el manejo de archivos (en UNIX existen solo archivos binarios, en DOS existen archivos de texto y binarios) y la manera en como se guarda el encabezado de las imágenes (dado que el formato TGC fue realizado para PC).

En el archivo **imagen.c** se encuentran las funciones para calcular el histograma de una imagen TGC y para almacenarlo en disco. Las funciones son:

float *histogram(MATRIX *IN, int min, int max);

Función que calcula el histograma de una imagen.

Entradas: Apuntador de tipo MATRIX donde se almacena la imagen de entrada. Valor minimo del tono de gris donde se desea computar el histograma. Valor máximo del tono de gris donde se desea computar el histograma.

Salida: Apuntador de tipo float al arreglo donde se creo el histograma.

void write_record (float *_histograma, int n, char *name);

Escribe en disco el histograma.

Entradas: Apuntador al arreglo del histograma. Número de datos del histograma (o número de tonos de gris computados). Apuntador al nombre del archivo (debe ser extensión HST).

Salida: Ninguna El programa termina si no se puede abrir el archivo.

Bibliografía

[1] Aldus Corporation. *TIFF revision 6.0 Aldus Developers Desk*, 1993.

[2] P.C. Chen and T. Pavlidis. Segmentation by texture using a co-ocurrence matrix and a split merge algorithm. *Computer Graphics and Image Processing*, 10:172–182, 1979.

[3] A. Freedman. *The Computer Glossary*. Amacon.

[4] R.C. Gonzales and P. Wintz. *Digital Image Processing*. Adisson-Wesley, 1977.

[5] Halcyon Software. *DoDoT Users Guide Image Processing Program for Microsoft Windows 3.0*.

[6] Hewlett Pakard Company. Edición 2.1. *Cómo utilizar el Scanner HP Scan Jet IIc con Microsoft Windows*, USA, 1991.

[7] M. Kamel and Zhao Aiguo. Extraction of binary charater/graphics images from grayscale document images. *Graphical Model and Image Processing*, 55(3):203–217, 1993.

[8] R. Kasturi and J.Alemany . Information extraction from image of paper-bases maps. *IEEE Trans. Software Eng.*, 14(5):671–675, 1998.

[9] D.C. Kay and J.R. Levine. *Graphics File Formats*. Wincrest/McGraw Hill, 1992.

[10] B.W. Kernighan and D.M Ritchie. *El lenguaje de programación C*. Prentice Hall, 1991.

[11] M. Morris Mano. *Diseño Digital*. Prentice-Hall, 1987.

[12] Theo Pavlidis and Y. Liow. Integrating region growing and edge detection. *IEEE Tran. on Patt. and Match. Intelligence*, 12(3), 1990.

[13] K. Pratt, William. *Digital Image Processing*. John Wiley & Sons, 1991.

[14] Quinn-Curtis. *Science & Engineering Tools for Microsoft C 5.x IPC-MC-006*.

[15] Gonzalez R.C and R.E Woods. *Digital Image Processing*. Adisson-Wesley, 1993.

[16] T.R. Reed and J.M. Hans du Buf. A review of recent texture segmentation and feature extraction techniques. *Image Understanding*, 57(3):359–372, 1993.

[17] Steve Rimmer. *Bit-Mapped Graphics*. Wincrest/McGraw Hill, 1990.

[18] Steve Rimmer. *Supercharged Bitmapped Graphics*. Wincrest/McGraw Hill, 1992.

[19] Azriel Rosenfeld and Avinash C. KaK. *Digital Picture Processing*. Academic Press, 1982.

[20] S. Suzuki and T. Yamada. Maris: Map recognition input system. *Pattern Recognition*, 23(8):919–933, 1990.

[21] SZXI Recognita Corp. *Recognita Plus. The Americas/English OCR Users Guide (MS Windows Envoroment) OCR Software Version 1.2*, 1991.

[22] Tseng Labs Inc. *VGA Enhanced Video Graphics Adapter, Users Manual*.

[23] M. Unser. Sum and difference histograms for texture classification. *IEEE Trans. on Pattern Anal. and Mach Intelligence*, PAMI-8(1), 1986.

[24] F.M. Wahl, K.Y. Wong, and R.G.Casey. Block segementation and text extraction in mixed text/image documents. *Computer Vision, Graphics, and Image Processing*, 20:375–390, 1982.

[25] D. Wang and S.N. Srihari. Classification on newspaper image blocks using texture analysis. *Computer Vision, Graphics, and Image Processing*, 47:327–352, 1989.

[26] Y. Yasuda, M. Dubois, and T.S. Huang. Data compression for check processing machines. *Proceedings of the IEEE*, 68(7):874–885, 1980.