Universidad Nacional Autónoma de México

PROGRAMA DE MAESTRIA Y DOCTORADO EN INGENIERÍA

CREACIÓN E IMPLEMENTACIÓN DE ALGORITMOS PARA EL RECONOCIMIENTO DE FORMAS DE OBJETOS DE MANUFACTURA INTEGRADOS EN MICROCONTROLADORES

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

P R E S E N T A :

MARTÍNEZ MARTÍNEZ FRANCISCO JAVIER

TUTOR: DR. MARIO PEÑA CABRERA

2012

JURADO ASIGNADO:

Presidente: Dr. MARTÍNEZ LÓPEZ JOSÉ ISMAEL

Secretario: Dr. TINOCO MAGAÑA JULIO CÉSAR

Vocal: Dr. PEÑA CABRERA MARIO

1er. Suplente: Dra. NAVARRETE MONTESINOS MARGARITA

2do. Suplente: Dr. PRADO MOLINA JORGE

Lugar donde se realizó la Tesis:

IIMAS – Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
UNAM - Universidad Nacional Autónoma de México

TUTOR DE TESIS:

Dr. PEÑA CABRERA MARIO

_____

FIRMA

Dedicatoria:

A mi familia: a ti Marlene G. Salazar Mtz: por tu apoyo ilimitado, por esas noches de platica y reflexión y por estos años de estar juntos: a ti hijo: Leonardo Javier por tu estado de ánimo tan alegre que siempre transmites generosamente y a mí hijo: Braulio Eleazar por su nacimiento tan esperado y la ternura que provoca.

A mis padres: Javier Martínez y Clara Mtz Saavedra por estar nuevamente conmigo. A Araceli Mtz por ser un verdadero apoyo en los momentos más laboriosos. A Olivia Mtz por sus saludos y mensajes de ánimo constantes (síguelos enviando). Por cortesía, por ser coherente con mis principios… por los años juveniles que pasamos juntos sin que importaran los problemas… a Alonso Aldo.

Agradecimientos:

A la Universidad Nacional Autónoma de México por seguir motivando mi superación profesional y personal nuevamente, por ser y hacerme... "Independiente".

Al Consejo de Estudios de Posgrado (CEP) por la subvención entregada, sin ella no hubiese sido posible la realización de mi posgrado.

Al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) por permitirme hacer uso de sus instalaciones durante la realización de mis estudios de maestría.

Al Dr. Mario Peña Cabrera por aceptar ser mi tutor, por despejar las dudas en los momentos críticos, por los dispositivos y materiales facilitados para la realización de esta tesis.

A la Dra. Margarita N., al Dr. Jorge P., al Dr. Julio César T., al Dr. Ismael Mtz por su tiempo invertido en la revisión del borrador y por sus sugerencias constructivas que mejoraron considerablemente la presentación de esta tesis.

A Dios, las gracias por seguir cuidándonos.

Resumen

El estudio desarrollado en el reconocimiento de formas geométricas propone dos algoritmos para calcular el área y el perímetro de una figura, que pertenece a un cuerpo sólido real, capturada por una imagen y programados en un microcontrolador Atmega32. Para el cálculo del perímetro utiliza el *gradiente digital de Laplace* y para el área recurre a un valor de umbral óptimo para contabilizar el número de pixeles que ocupa la figura y esta información se utilizará para identificar la pieza y obtener otra característica de la misma si así se desea.

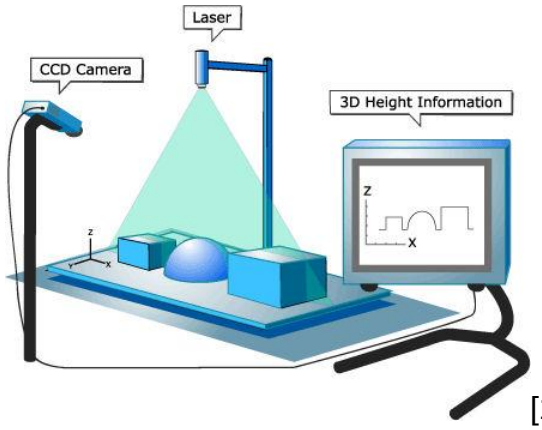
El interés por estudiar estos algoritmos y el procesamiento de imágenes es porque celdas de manufactura utilizan modernos sistemas de visión por computadora para: dar un refinado acabado y/o detectar fallas en el producto, incrementar la producción y reducir el tiempo de fabricación. Utilizando menos recurso humano en el proceso.

Finalmente deseo que este pequeño trabajo sea una introducción para aquellas personas que incursionan en este campo de estudio: “Visión por Computadora”. Y sean bienvenidas las sugerencias o críticas constructivas que podrán mejorarlo aún más.

Índice

Resumen 5
Índice 6
INTRODUCCIÓN 8
1.1 La percepción visual 8
1.2 Visión por computadora 10
1.3 Dispositivos electrónicos de captura de imágenes 11
1.3.1 Los sensores CCD 11
1.3.2 Los sensores CMOS 14
1.4 Objetivos 15
1.5 Planteamiento del problema 17
1.6 Justificación 17
1.7 Integración del sistema para la solución 17
1.7.1 Proceso de captura de la imagen 20
Capítulo 2. FUNDAMENTOS BÁSICOS 22
2.1 Representación digital de imágenes 22
2.2 Digitalización y almacenamiento de la imagen 24
2.3 Reducción de datos 26
Capítulo 3. TEORIA DE ALGORITMOS 28
3.1 Segmentación y representación de formas definidas 28
3.1.1 Regiones de interés 28
3.1.2 Comprensión de datos RLE (Run-Length Encoding) 29
3.2 Características 30
3.2.1 Detección de esquinas 32
3.3 Algoritmos descriptores de formas 33
3.3.1 Detección de borde y curva 34
3.3.1.1 Método del Gradiente 34
3.3.1.2 Método de Sobel 36
3.3.1.3 Método de Laplace 37
3.4 El histograma y su ecualización 38

Capítulo 4. ALGORITMOS 39
4.1 Algoritmos en Matlab 39
4.2 Algoritmos 43
4.2.1 Algoritmo para Área 43
4.2.2 Algoritmo para el Perímetro 44
Capítulo 5. PRUEBAS Y RESULTADOS 46
5.1 Procesamiento de imágenes por computadora 46
5.2 Algoritmos descriptores 52
Capítulo 6. Conclusiones y recomendaciones 61
Bibliografía 63



[16]

INTRODUCCIÓN

1.1 La percepción visual

Un brazo robot en la industria se utiliza para ensamble de partes, moldeo de plástico, soldadura de arco, entre otras funciones. De alguna forma los movimientos que hace el brazo para ubicar algún objeto conocido han sido preestablecidos porque se ha colocado en un lugar estratégico, en la orientación adecuada y la programación se ha enfocado solamente para realizar los movimientos espaciales necesarios y las tareas específicas. La situación cambia, cuando el brazo robot sabe que pieza es, pero no sabe en donde se encuentra o está fuera de posición, entonces, se le agrega la percepción sensorial visual para ubicar el objeto en el área de trabajo, figura 1.1, más las acciones a seguir para manufacturarlo.

Los robots en la actualidad usan la percepción visual para detectar obstáculos en un ambiente controlado. De este modo, asociar una cámara con un sistema electrónico y un brazo mecánico da cierta idea que el sistema se controla por sí mismo. La inteligencia viene dada a través de algoritmos y de sensores electrónicos que tiene el robot asociados al hardware y al software respectivamente. Mientras que los sensores proporcionan información del

ambiente que lo rodea, los algoritmos lo dotan de decisiones lógicas estructuradas por el programador [1]. Los sensores detectan temperatura, presión, humedad, luz, velocidad, aceleración, fuerza y par, la posición de las extremidades o brazos, otros diferentes son: táctiles, acústicos (micrófono), y de proximidad. El radar y el GPS son también utilizados, mientras que el primero, a través de ondas electromagnéticas mide distancias altitudes, direcciones y velocidades de objetos estáticos o móviles, el segundo determina la posición global de un objeto por medio de satélites. En el caso de los algoritmos, estos son la comunicación entre el usuario y el robot, eficientes para la realización de las tareas y están programados en un leguaje de alto nivel como C, Java u otro.

Figura 1.1. Brazo robot y cámara usados en una línea de ensamble.

Aunque los anteriores sensores juegan un rol importante en el trabajo ejecutado por un robot, la visión es la más versátil de las capacidades sensoriales [1]. Mientras la planeación de movimientos son programados para realizar una tarea; el sentido visual, otorga al robot la facultad de decidir libremente, condicionando todas las variables que halla en el ambiente, de esta manera, la inteligencia esta proporcionada en él. El sentido visual se adiciona por medio de una cámara digital, este dispositivo electrónico tiene un sensor de imagen que está construido de tecnologías distintas: CCD o CMOS.

Sistemas de visión industrial modernos realizan medición física: lineal, de diámetros de diferentes tamaños, de curvaturas, cálculo de superficie, altura, cuentan objetos y escalamiento. Se utilizan en inspección: presencia o ausencia de etiquetas. Detección o presencia de defectos, de caracteres alfanuméricos. Identificación de códigos de barras. Presencia de impurezas en fármacos. En este escenario, un sistema de visión se ha extendido, hasta usarlo en ambientes microscópicos, inspección del fondo del mar, estudiar el espacio exterior y cada vez surgen nuevas aplicaciones.

1.2 Visión por computadora

Consiste en reconocer y localizar objetos en el ambiente, mediante el procesamiento de imágenes. Cuyo objetivo es extraer características para su descripción e interpretación. A esta etapa se le llama procesamiento que se le da a una imagen para después, más a detalle, aplicar algoritmos para el reconocimiento de objetos. En la etapa de procesamiento también se aplican algoritmos que mejoran la imagen para la siguiente fase.

La asociación de la cámara con la computadora-monitor es común y se menciona en la literatura como visión por computadora y es aplicada en celdas de manufactura industrial y básicamente se componen de: una computadora, una cámara o CCD, un brazo robot y fuentes de luz, en la figura 1.2 se muestra una celda con los elementos interconectados.

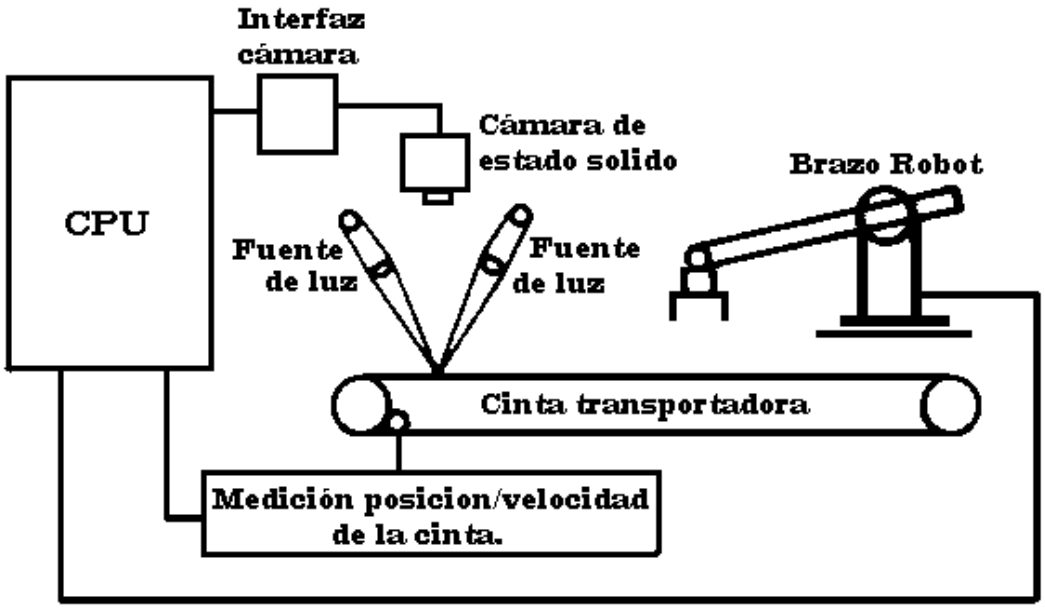


Figura 1.2. Ejemplo de un prototipo con visión artificial para trabajo industrial.

1.3 Dispositivos electrónicos de captura de imágenes

Son dos las tecnologías que se usan para la fabricación de sensores de imágenes: los CCD (Charge Coupled Device) y los CMOS (Complementary Metal Oxide Semiconductor). Ambos se abordan y explican a detalle: como es su arquitectura y funcionamiento. La arquitectura se refiere a la forma de como transfiere la información (luz incidente) un sensor de imagen para su conversión digital.

1.3.1 Los sensores CCD

El Dispositivo de Cargas Acopladas (CCD) fue inventado en 1969 por Willard Boyle y George Smith en los Laboratorios Bell. En el transcurso de 43 años, los CCDs se han usado en una amplia gama de productos.

Típicamente, los pixeles están acomodados en una sola línea (CCDs de matriz lineal) o en una cuadricula de dos dimensiones (CCDs matriz de área cuadrada). La aplicación particular, dicta el tipo de CCD que se usa. Los

escáneres de planos, por ejemplo, usan una matriz lineal y en este caso es necesario mover progresivamente el CCD sobre el objeto que se va a examinar (o viceversa). Las cámaras digitales, por otro lado, usan normalmente un CCD de matriz de área cuadrada, permitiendo así la captura en una sola exposición de una imagen bidimensional completa.

El método más simple para transferir la carga almacenada es el *frame-transfer* (FT) CCD, figura 1.3, básicamente consiste en transferir toda la carga eléctrica del área expuesta a la luz del CCD en un solo tiempo, esta consiste de tres secciones: el área del registro A se usa para la conversión de fotones en cargas eléctricas y almacenarlas durante el tiempo de exposición. Esta forma de distribución en 2 dimensiones es subsecuentemente transferida hacia abajo, al registro B, a otra área del CCD que está cubierta con un metal opaco. Desde el registro B una hilera a la vez es transferida hacia abajo a una línea del CCD (registro C), a partir de aquí las cargas eléctricas son transportadas lateralmente al amplificador de salida. De este modo, se extrae el contenido de este renglón imagen secuencialmente y se capturan múltiples imágenes de una dimensión, con el fin de construir la imagen bidimensional. La desventaja de este método es la sobreexposición de brillo (saturación) en las áreas donde ocurre la transferencia del registro A al B. Esto ocurre porque el registro A permanece sensible a la luz en el transporte vertical. Por esta razón, usan un dispositivo mecánico (shutter) que cubre el registro A, en la transferencia de la carga, al registro B. Otros dos métodos mejorados a partir de este, son: *Interline-transfer* (IT) CCD y el combinado FT-CCD y IT-CCD, figuras 1.3(b) y (c) [2].

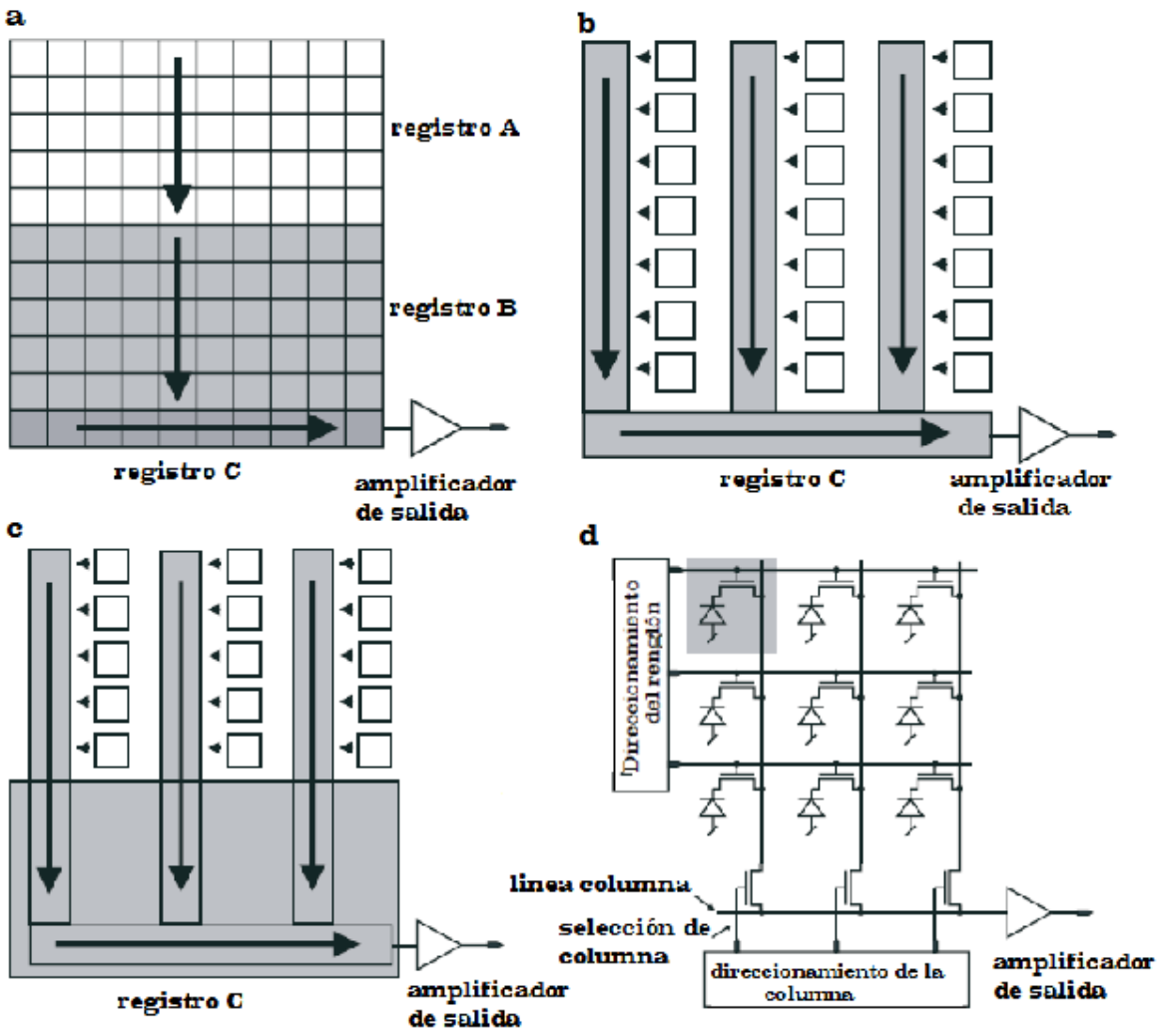


Figura 1.3. Las cuatro arquitecturas más importantes de los sensores de imagen de estado solido: a) Frame-transfer (FT) CCD con sus tres registros; b) Interline-transfer (IT) CCD con columna de luz iluminada y cubiertos para transferencia de carga vertical; c) combinado FT-CCD y IT-CCD; d) arreglo y selección de un fotodiodo y un transistor por pixel.

Un parámetro fundamental de un CCD, es **el número total de pixeles que conforma el área sensible a la luz del dispositivo**. Uno de los primeros CCDs de matriz de área cuadrada, fabricados por Fairchild en 1974, tenía una resolución de 100×100. Hoy, el dispositivo más grande disponible comercialmente es de 9216×9216 o aproximadamente 85 millones de pixeles [30].

La tabla 1.1 muestra dos diferentes dispositivos para una comparación.

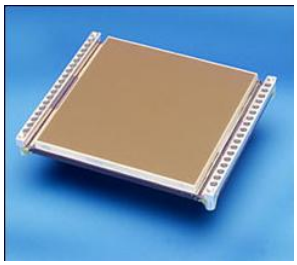
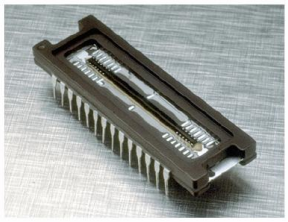
CCD	Características
 a) Matricial	Fabricante: Kodak, CCD 485 Tipo: matriz Pixel: 15-µm x 15 µm Área de imagen: 61.2 x 61.2 mm Cantidad de pixeles: 4096 x 4097 Método de transferencia: Full frame Aplicaciones: industrial, medica y científica [14].
 b) Lineal	Fabricante: Kodak, CCD KLI-2113 Tipo: arreglo lineal, Trilineal RGB Pixel: 14-µm x 14 µm Área de imagen: 29.37 x 0.24 mm Cantidad de pixeles: 2098 x 3 Método de transferencia: Full frame Aplicaciones: idealmente para escaneo a color [15].

Tabla 1.1 CCDs a) Matricial. b) Lineal.

1.3.2 Los sensores CMOS

Los sensores de imagen CMOSs también los encontramos lineales o en dos dimensiones; en el arreglo de los fotodiodos cada uno tiene su propio transistor, como se ilustra en la figura 3.1(d). La forma de operar es así: se asume que los fotodiodos del sensor de imágenes están precargados a un nivel de voltaje, típicamente 5V. Bajo la luz incidente cada uno de los pixeles es descargado y leído por el direccionamiento de la columna y renglón correspondiente a cada transistor. Proporcionando una línea de salida del pixel al amplificador de salida. El amplificador mide cuanta carga es necesaria para ello, y esta carga es idéntica a la fotocarga acumulada en el sitio del pixel (más la carga de corriente oscura). De esta forma, cada pixel es leído individualmente, aleatoriamente, y el tiempo de exposición está completamente bajo el control de direccionamiento electrónico externo [3].

1.4 Objetivos

Realizar dos algoritmos para calcular el área y perímetro de una figura geométrica definida en una imagen digital, figura 1.4, y programarlos en un microcontrolador atmega32 y demostrar el resultado.

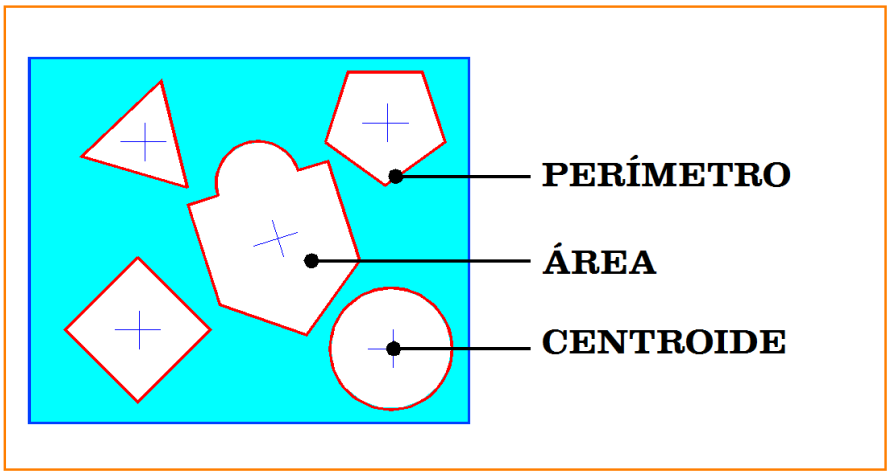


Figura 1.4. Tres características que se extraen de una figura geométrica conocida: el perímetro, el área y el centroide de la pieza.

Para ello, el microcontrolador examina una matriz de datos (imagen) en escala de grises o binaria. Información que incluye dos cosas: su cromatismo y ubicación de cada pixel. Por cromatismo deberá entenderse el color que posee el pixel (monocromático, escala de grises o RGB). Una imagen a color contiene el triple de información que una en escala de grises y naturalmente una imagen monocromática es más fácil de procesar por tener la menor cantidad de información que las anteriores. Los algoritmos descriptores se realizan en leguaje C. Las figuras propuestas para aplicar los algoritmos son: un triángulo, un círculo, un rectángulo y una copa o alguna otra. La figura 1.5 muestra un diagrama de flujo del trabajo de tesis.

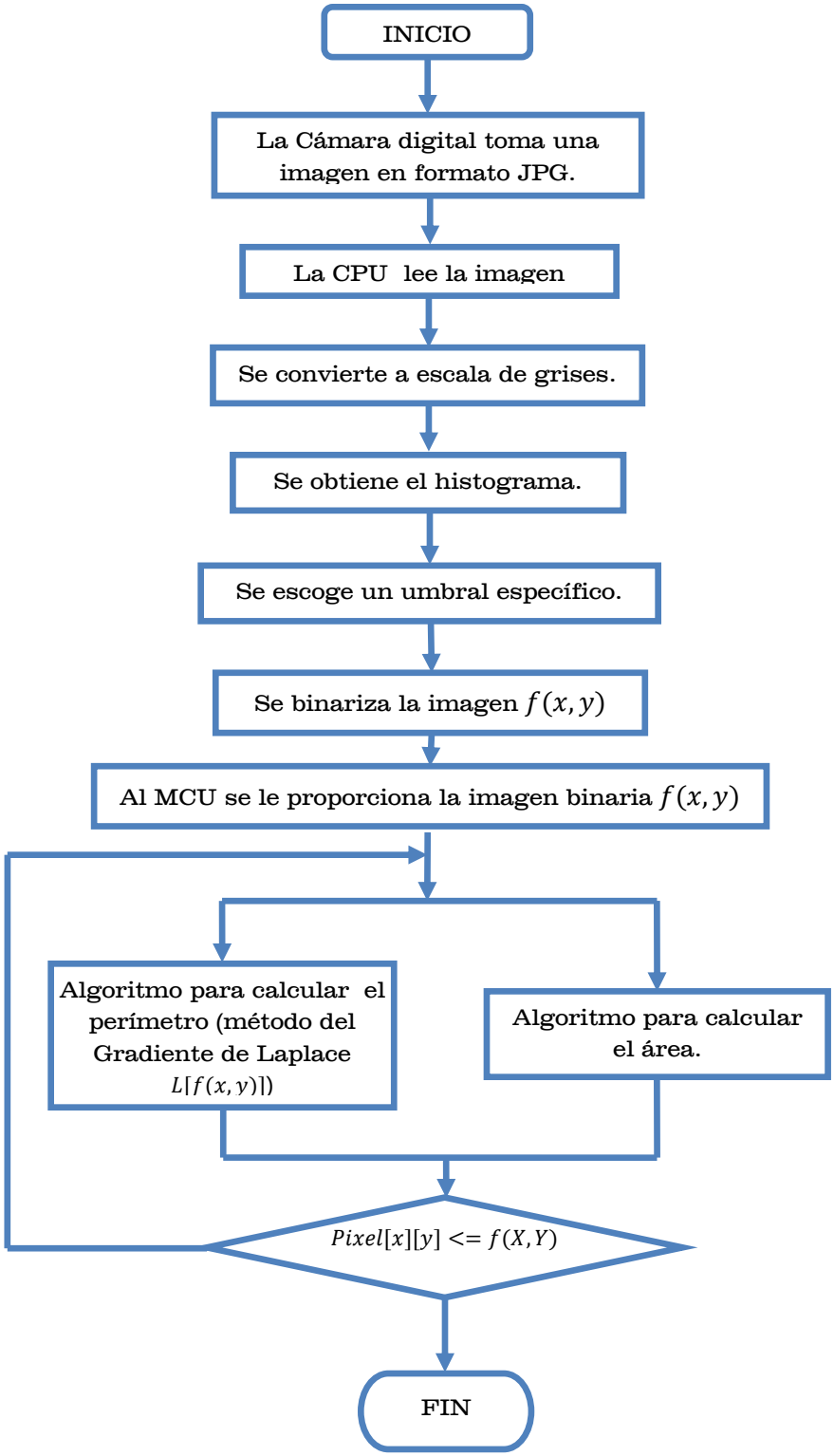


Figura 1.5. Diagrama de flujo que explica el procedimiento que utilizará el microcontrolador para calcular el perímetro y el área de una figura.

1.5 Planteamiento del problema

Cuantiosos programas computacionales dedicados al campo de la visión por computadora, evalúan imágenes para extraer información de un ambiente tridimensional. Los resultados muestran el reconocimiento, con un alto grado de precisión, de piezas de ensamble en una celda de manufactura. Las letras y los colores son también diferenciados, detectados y comparados.

El problema consiste en comprender la estructura de una imagen JPG, la información que contiene y la forma de manipularla para realizar algoritmos descriptores del perímetro y del área para evaluar figuras definidas en una imagen. La cámara utilizada tiene una constitución muy básica: comunicación USART con un nivel TTL ó 5 volts que se adapta a una computadora con un max232 y el software correspondiente o a un microcontrolador Atmega32. Las imágenes capturadas son a color con una extensión de archivo JPG y serán de 160×120 pixeles o de menor tamaño.

Al final de la investigación se escriben las conclusiones de la eficiencia de los descriptores para calcular el área y el perímetro de las figuras planteadas.

1.6 Justificación

Para comprender la estructura y funcionamiento de un sensor de imagen y como se utilizan para el reconocimiento de cuerpos sólidos con características intrínsecas definidas en su forma, contorno, tamaño y color, además de su orientación y posición en un ambiente real, que permita proponer nuevos proyectos en este campo de visión por computadora, académicos y/o personales.

1.7 Integración del sistema para la solución

Básicamente comprende un microcontrolador y la cámara como componentes principales más los algoritmos descriptores, a continuación se citan:

1. MCU Atmega32 (plataforma STK16)

2. Chip MAX232
3. EEPROM AT24C512
4. Cámara a color LINKSPRITE
5. Programador AVRISP

La cámara digital, con interfaz serial UART, esta transfiriendo la imagen a la computadora, que la transforma a escala de grises y posteriormente se obtiene el histograma para escoger un valor de umbral adecuado y se utiliza para binarizar la imagen, ver figura 1.6.

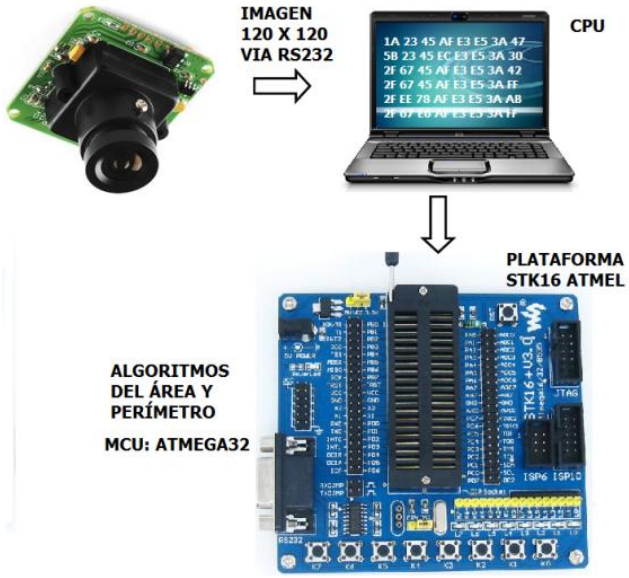


Figura 1.6. Esquemático del Sistema Digital Electrónico.

Después, los valores de intensidad o binarios de toda la matriz se introducen en la EEPROM del MCU, por medio de un arreglo, y este ejecutará los algoritmos *descriptores* del *área y del perímetro* de la figura propuesta. Una breve descripción del hardware implementado para los algoritmos y el control de la cámara se mencionan a continuación:

MCU Atmega32: Constituye la unidad central de procesamiento de la imagen y es el encargado de ejecutar el programa realizado en leguaje C, realizando operaciones lógicas y aritméticas. Será suficiente programar los algoritmos en este microcontrolador de la marca ATMEL de 40 pines. El reloj de

operación es de 16Mhz ó 16MIPS, cuenta con cuatro puertos de entrada-salida: A, B, C y D, tres Timers: 0,1 y 2; convertidor análogo-digital y PWM; comunicación SPI, I2C y USART, comparador y memoria EEPROM de 1kbyte.

MAX232: este componente electrónico se encarga de la transmisión y recepción asíncrona universal (UART) serie entre la cámara y la computadora.

EEPROM: almacena permanentemente imágenes, sonidos o texto y la retiene aún sin una fuente de alimentación. Hay memorias externas que se podrán utilizar, como la EEPROM AT24C512, que proporciona 65,536 bytes de memoria con un protocolo de comunicación de datos serial I2C bidireccional, interfaz que sirve para poder comunicarse con un MCU. Permite la escritura por páginas de 128 bytes cada una (512 páginas en 2.56 segundos) y brinda una protección contra descargas electrostáticas (ESD). Cuenta además con un pin de protección contra escritura por hardware y protección de datos por software. Su ciclo de escritura típico es de $5\ ms$.

AVRISP: Programador de microcontroladores ATMEL, funciona a través del puerto USB. Se utiliza el software AVR Studio que es un Ambiente de Desarrollo Integrado (IDE) para programar y depurar código en lenguaje ensamblador ó C y genera los archivos con extensión HEX que se cargaran a través del AVRISP al microcontrolador o MCU.

La cámara a color LINKSPRITE, ver figura 1.7, adquiere imágenes con el método de comprensión JPEG y su comunicación es serial a nivel TTL.

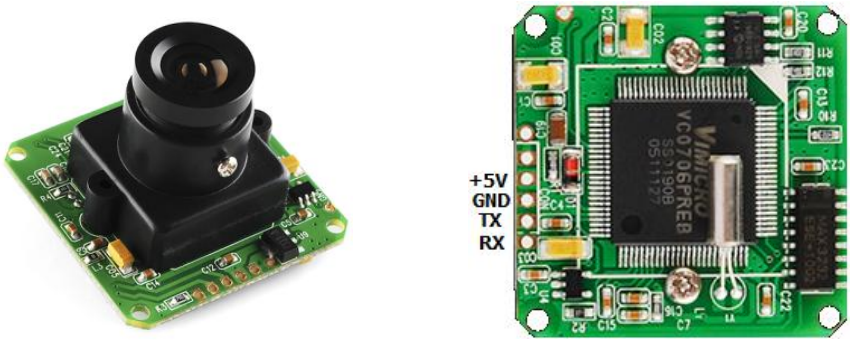


Figura 1.7. (a) Cámara LS-Y201 LINKSPRITE. (b) Distribución de los pines, dimensiones: $\mathbf{32 \times 32\ mm}$.

1.7.1 Proceso de captura de la imagen

El diagrama de flujo, figura 1.8, es la "*inicialización"* y respuesta que envía la cámara al encenderse, indicando que esta lista. El de la figura 1.9, muestra el proceso para capturar una imagen.

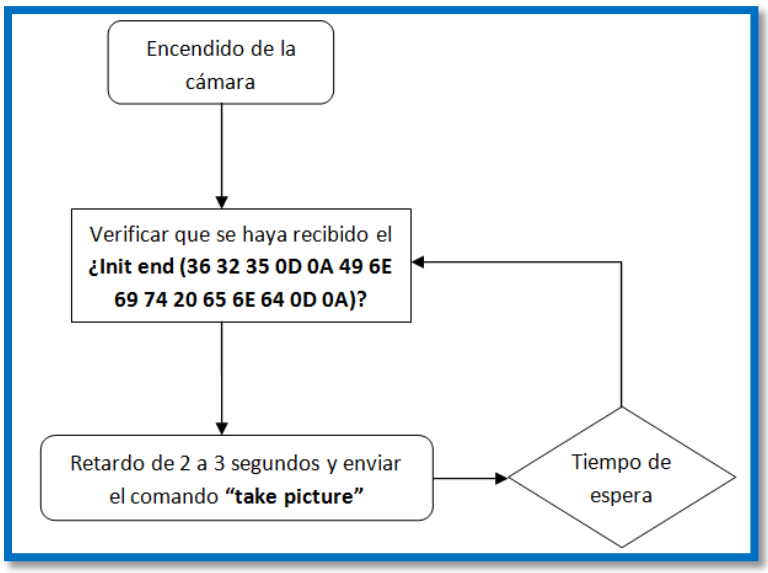


Figura 1.8. Inicialización de la cámara.

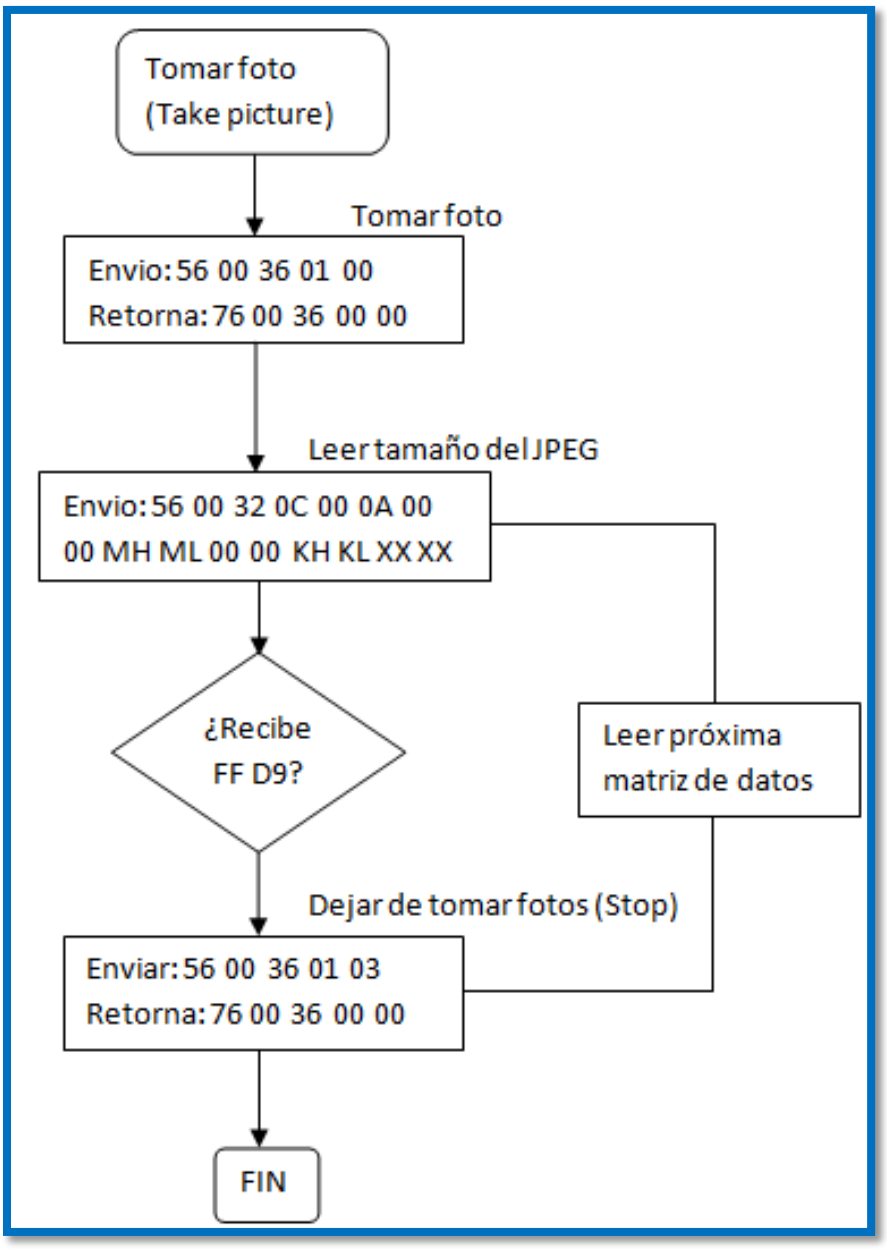
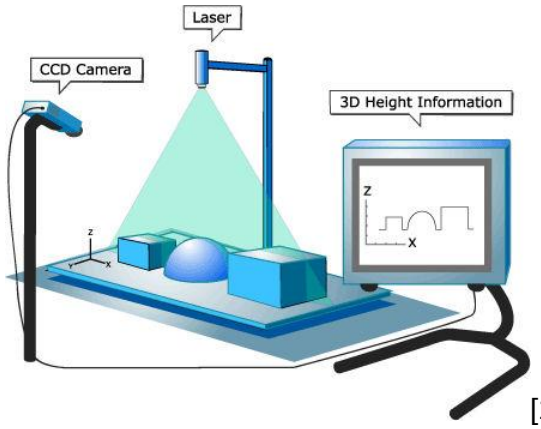


Figura 1.9. Diagrama de flujo para tomar la imagen.



[16]

Capítulo 2. FUNDAMENTOS BÁSICOS

2.1 Representación digital de imágenes

Una imagen es definida como una función de dos dimensiones, $f(x,y)$, donde x y y son coordenadas espaciales respecto al plano. El valor o amplitud que hay en un par coordenado (x,y) es la intensidad de un pixel en ese punto y es una cantidad discreta. A los valores de las amplitudes digitalizadas se le llama cuantización [4,5]. Como la luz es una forma de energía y la amplitud de $f(x,y)$ es diferente de cero y finita, entonces:

$$0 < f(x,y) < \infty \tag{2.1}$$

Las imágenes que percibimos de forma visual, provienen de la luz que reflejan los objetos, de tal forma que la amplitud de $f(x,y)$ tiene dos componentes, una componente es la cantidad de luz que incide sobre la escena vista, mientras que la otra, es la cantidad de luz reflejada por los objetos. Propiamente se llaman componentes de *iluminación* y *reflectancia*, expresadas por $i(x,y)$ y $r(x,y)$ respectivamente. Combinadas como un producto forman $f(x,y)$:

$$f(x,y) = i(x,y)r(x,y) \tag{2.2}$$

donde

$$0 < i(x,y) < \infty \tag{2.3}$$

y

$$0 < r(x,y) < 1 \tag{2.4}$$

Para el caso de la reflectancia, la ecuación (2.4), indica total absorción en cero y total reflectancia en uno.

El resultado del muestreo y la cuantización, es una matriz de números reales, y propiamente se llama imagen digital $f(x,y)$ que tiene M renglones y N columnas de un tamaño de $M \times N$. La figura 2.1 ilustra la convención de coordenadas, el rango de x es de 0 a $M-1$ y el de y de 0 a $N-1$ [4].

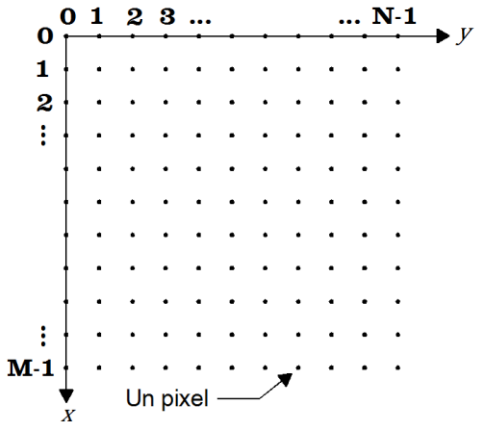


Figura 2.1. Convención de coordenadas [1].

Con el arreglo del sistema de coordenadas anterior, se representa una función de imagen digitalizada así:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.5}$$

Todo el lado derecho de la ecuación (2.5) es por definición la imagen digital [4, 6].

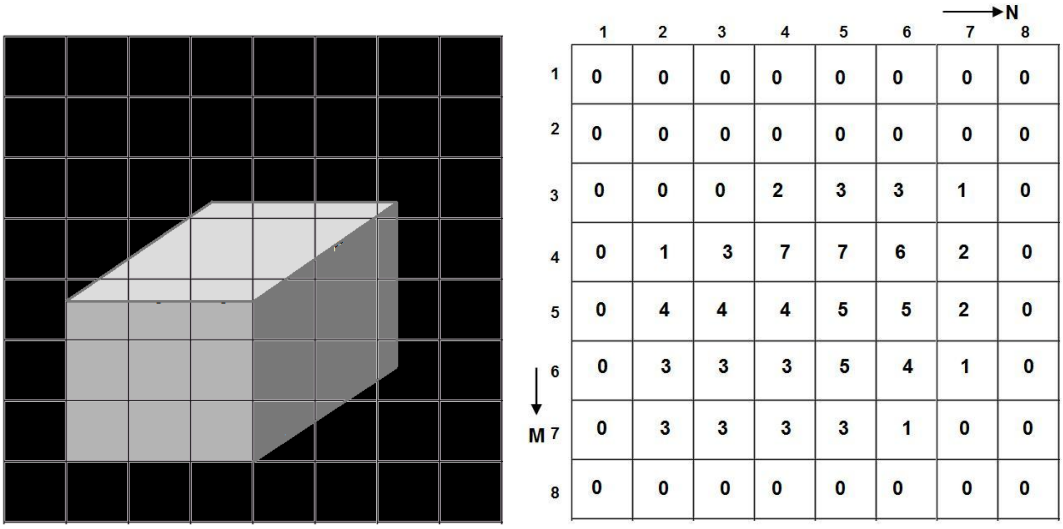
2.2 Digitalización y almacenamiento de la imagen

Después de una conversión análogo/digital, la imagen es almacenada en el buffer o memoria temporal interna de la cámara, pixel por pixel, para ser transferidos a otro dispositivo. La palabra pixel es una contracción inglesa de elemento de imagen (picture element) y es la menor unidad homogénea que hay en una imagen, de tal manera, que los pixeles integran las imágenes digitales. Una imagen que tiene el nivel de cada pixel representado por 8 bits por la conversión a digital, será el tamaño de la palabra o byte que debe tener cada celda de memoria o buffer. Si cada buffer tiene 8 bits de ancho, entonces se representan $2^8 = 256$ niveles de grises, en la figura 2.2 se muestran algunos. Es decir, cada pixel tiene una y solo una variación de gris o un valor de 0 a 255 antes de que se vuelva repetir el proceso de obtener otra imagen. Muchas de las aplicaciones industriales no requieren más de 256 niveles de grises.

Nivel de gris											
Decimal	23	46	69	92	115	138	161	184	207	230	253
Hex	17	2E	45	5C	73	8A	A1	B8	CF	E6	FD

Figura 2.2. Algunos Niveles de Gris, su representación en decimal y hexadecimal.

El nivel de gris de un pixel se representa por el ancho de la palabra y es inferior o superior a 8 bits, dependiendo de los bits de salida del convertidor A/D de la cámara, el microprocesador y memoria externa que se maneje. Para ilustrarlo, se muestra una imagen de 8×8 pixeles en la figura 2.3, que requerirá de 64 posiciones de memoria. Si el convertidor A/D lo hace con solo 3 bits de salida, será posible dar a cada pixel hasta 8 tonos de gris, que, si se enumeran del 0 al 7, normalmente el 0 corresponde al negro y el 7 al blanco o la zona de más luminosidad [7].



M \ N	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	2	3	3	1	0
4	0	1	3	7	7	6	2	0
5	0	4	4	4	5	5	2	0
6	0	3	3	3	5	4	1	0
7	0	3	3	3	3	1	0	0
8	0	0	0	0	0	0	0	0

(a) (b)

Figura 2.3. (a) Imagen de 8x8 pixeles. (b) Representación numérica en nivel de gris, 3 bits (8 niveles de grises).

En este contexto, el uso de colores en las imágenes obedece más a los miles de intensidades y sombras que el ojo humano percibe. En contraste con el nivel de grises que detecta y es de una a dos docenas de sombras de gris en algún punto de la imagen para un observador promedio. El objetivo, entonces, es asignar un color a cada pixel basándose en la intensidad [8].

Los intervalos de colores que se perciben al observar en un ambiente normal, resulta de la mezcla de luz de diferentes longitudes de onda. Referidos como colores primarios, estos son: el rojo (R, $700\,nm$), el verde (G, $546.1\,nm$) y el azul (B, $435.8\,nm$), al combinar cada uno de ellos con los otros en varias proporciones (intensidades), se producen un rango amplio de colores. Se malinterpreta que a partir de ellos se producen *todos los colores*. Las sombras a color no se obtienen por alguna combinación de los colores primarios [8].

Los colores primarios se adicionan para producir los colores secundarios de luz: *magenta* (rojo mas azul), *cián* (verde más azul) y *amarillo* (rojo más verde) [8].

2.3 Reducción de datos

En su constitución física, la cámara digital tiene un sensor de imagen (CCD) y un procesador interno que realiza el primer tratamiento a la imagen antes de transmitirla. Y consiste en reducir, por el método de comprensión JPEG, el volumen de datos. No es parte de esta investigación explicar cómo se lleva a cabo el método de comprensión JPEG, solo se menciona brevemente, la estructura básica de este tipo de archivos. El formato JPEG tiene una estructura en secciones identificados como marcadores. Un marcador es una cadena de dos bytes que comienza con el número hexadecimal FF y un segundo byte identifica el tipo de marcador que es. En la tabla 2.1 se muestran los marcadores comúnmente encontrados en un archivo JPEG [9].

FF D8 – Inicio de imagen
FF E0 – Inicio de solicitud del segmento 0
FF E1 – Inicio de solicitud del segmento 1
:
FF C0 – Inicio del arreglo
FF C4 – Inicio de la tabla del codificador Huffman
FF DB – Inicio de la tabla de cuantización
FF DA – Inicio del escaneo
FF D9 – Fin de la imagen

Tabla 2.1. Marcadores comúnmente encontrados en un archivo JPEG.

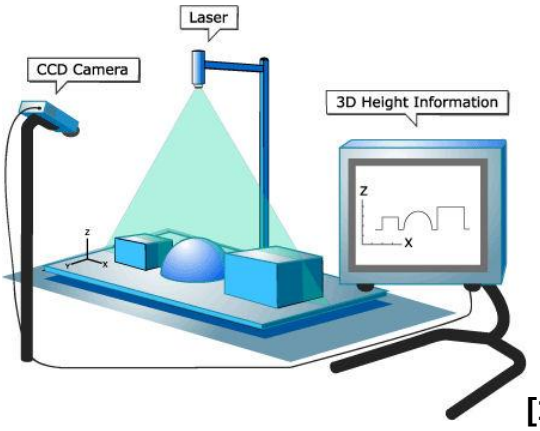
En este contexto, una imagen con extensión JPG, solo indica el uso del método de comprensión JPEG y se necesita un intérprete para decodificarla y así conocer los valores de la matriz de pixeles que forman la imagen.

El siguiente tratamiento de la imagen digital es pasarla a escala de grises y después binarizarla. Cámaras digitales de uso masivo o comercial tienen la opción de tomar imágenes directamente en escala de grises (blanco y negro) e implica colocar el nivel de gris en los tres planos repetidamente por pixel, sin embargo, será imposible extraer sus componentes RGB si se requirieran. La reducción o

comprensión de una imagen en escala de grises o binaria no causa problemas para algunas aplicaciones, por el contrario reducirá más los datos para el procesamiento digital. Por ejemplo:

En una imagen digitalizada de 128 líneas de 128 pixeles cada una, en escala de grises, hay un total de $16{,}384$ pixeles y utilizando un convertidor analógico a digital de 8 bits de salida para obtener el nivel de gris de cada pixel, se tendrá un total de $16{,}384 \times 8 = \mathbf{131{,}072}$ bits de datos y se necesitara 16kbytes de memoria para guardarla. Ahora, la reducción del volumen de datos se da por una conversión de escala de grises a blanco y negro (binaria), almacenando el nivel de cada pixel ($0 = negro$ ó $1 = blanco$), la reducción será de $128 \times 128 \times 1 = \mathbf{16{,}384}$ bits y 2Kbytes de memoria. Como se ve la reducción del volumen de datos es de $131{,}072 - 16{,}384 = \mathbf{114{,}688}$ bits que ya no existen, la diferencia es de siete veces de espacio liberado en la memoria.

En una imagen, hay información redundante o información visual cuantificable que debe eliminarse y es deseable por que no es esencial. Esta pérdida cuantitativa de datos se le llama cuantización. Otra razón, es la gran cantidad de información que hay en ella al momento de ser capturada en un CCD y que tiene que comprimirse. La compresión, es la técnica o método de codificar y decodificar datos sin o con pérdida de información, está última implica, que la recuperación de datos ya no es posible o es irreversible. Aún así, las pérdidas son controladas por el método JPEG, que al final de la comprensión, tiene la información suficiente para reconstruir o decodificar nuevamente una imagen equivalente a la original.



[16]

Capítulo 3. TEORIA DE ALGORITMOS

3.1 Segmentación y representación de formas definidas

La siguiente fase, para reconocer los objetos de una imagen, es separar o segmentar la información de interés de la que no lo es. Una imagen contiene una inmensa cantidad de información, pero gran parte de la misma es redundante para muchas aplicaciones. Por lo tanto, se simplifica la imagen de tal forma, que resulte sencilla y rápida de identificar o reconocer los objetos de interés. Para una persona común, es más sencillo reconocer el esquema de un “objeto” (animal, máquina, etc), que si el propio objeto se encuentra sobre una escena saturada de detalles [7].

3.1.1 Regiones de interés

En visión binaria, un objeto se reconoce por coincidencia o semejanza del área de interés con un patrón o calculando sus características. Cada paso en un análisis de conectividad, se hace en tres pasos discretos: comprensión del código, desarrollo del área de interés y calculando sus características.

3.1.2 Comprensión de datos RLE (Run-Length Encoding)

Esté método de comprensión se basa en reducir una cadena de datos repetida o corrida de caracteres iguales. Para el caso de una imagen, la codificación consiste en llevar la cuenta de los pixeles que tienen un mismo nivel de color y la dirección donde se inicia, ocupando así, menos espacio para guardarla. Además, sirve para extraer aquellas regiones de interés de una imagen que posean ciertas características o umbral determinado.

Mientras una imagen binaria se almacena de manera más compacta como un arreglo de bits, el almacenamiento se reduce aún más usando esta técnica de codificación. Como los pixeles en una imagen binaria tienen solo uno de dos valores, la mayoría de las filas en una imagen contiene corridas de consecutivos 1s ó 0s. En una aplicación típica, un objeto, toma una gran cantidad del área de la imagen y está rodeado por el fondo. El área del objeto son todos 1s mientras que el fondo es todo 0s, como se ve en la figura 3.1.

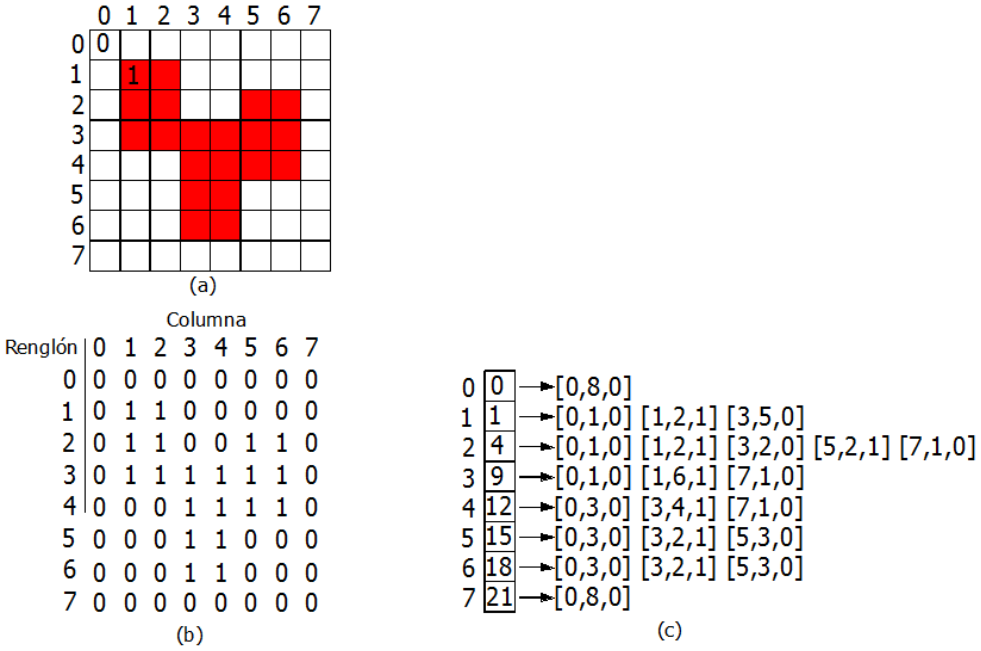


Figura 3.1. Compresión RLE del código de una imagen binaria. a) Imagen. b) Arreglo de pixeles. c) código comprimido RLE.

La compresión RLE toma ventaja de esta coherencia espacial. En lugar de almacenar una fila de pixeles, el número de la columna a la cual la transición toma lugar es la que se almacena. Se usa una estructura de tres elementos para almacenar información acerca de cada transición. La estructura de datos contiene el número de la columna de la corrida de valores consecutivos, la longitud de la corrida y el número de la forma de esos pixeles que pertenecen a esta, figura 3.1(c). Para una imagen, la forma del número es 0 ó 1. Después que la imagen es procesada, cada burbuja (silueta) tiene un número de forma diferente, si hay más de un objeto en la imagen.

La tripleta de datos estructurada contendrá información redundante y puede ser reducida a una dupla que contenga la corrida y número de forma. En el ejemplo de la figura 3.1(c), la comprensión del código toma más espacio que la imagen binaria original.

Un sistema típico de visión con una imagen de 256 por 256 pixeles requiere 8,192 bytes para almacenar la imagen. Con la comprensión de código, una imagen de un círculo de 126 pixeles de diámetro se almacena en tan solo 512 bytes. Reduciendo el volumen de datos considerablemente, también se disminuye el tiempo de procesamiento. La mayoría de algoritmos eficientes para el procesamiento de imágenes binarias, depende en gran parte del hecho que solo las transiciones son significantes, y por tanto,

solo las transiciones necesarias serán procesadas. La comprensión RLE reduce los datos a una secuencia de transiciones, por lo que es una forma de adquisición ideal para estos algoritmos.

3.2 Características

Son divididas en dos categorías: aquellas que son independientes de la ubicación y las que no lo son. Las características dependientes de la ubicación incluyen el centroide y la orientación angular. Las características independientes de la ubicación incluyen el área, perímetro, longitud y el número de huecos en la figura (burbuja).

La característica más simple de calcular es el color, como estaría almacenado en el código de compresión RLE.

Para poder extraer características de un área de interés que se encuentra en una imagen, se utiliza la relación de inclusión, tales como huecos en objetos, son encontrados buscando en el código de comprensión el área de interés y construyendo un árbol de burbujas. La raíz del árbol es el fondo y las ramas son las áreas en el marco. Un rectángulo delimitador es descrito con cuatro valores: mínimo x, máximo x, mínimo y y máximo y, figura 3.2. Los valores y es el primer renglón y último del área de interés. Los valores x son encontrados comparando al inicio de la columna de la primera corrida en cada renglón, y la columna final de la última corrida en cada renglón. De los valores del rectángulo delimitador, podemos calcular la posición del **centroide** (x_c, y_c), la **altura** y el **ancho**.

$$\mathrm{x_c} = \frac{1}{2}(\text{mínimo x} + \text{máximo x}) \tag{3.1}$$

$$y_c = \frac{1}{2}(mínimo\ y + máximo\ y) \tag{3.2}$$

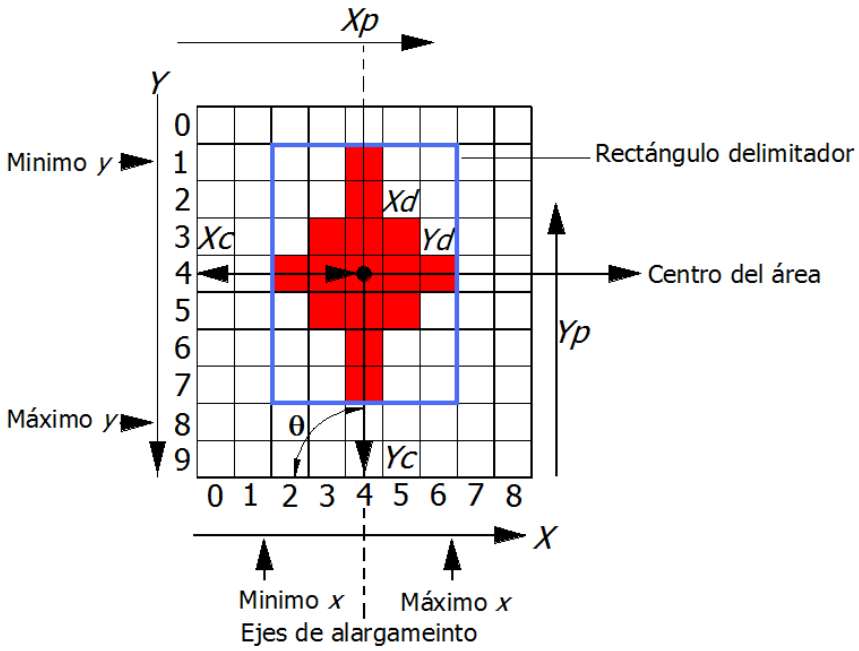


Figura 3.2 Parámetros para cálculo de características.

Mientras que el rectángulo delimitador nos da una medida de su tamaño, esto nos dice muy poco acerca de su forma, aparte si el rectángulo es cuadrado o

alargado. Una medida que captura más información acerca de la complejidad del objeto es la forma y longitud del perímetro del área en cuestión. El **perímetro** es la delimitación entre los pixeles del área en cuestión y los pixeles del fondo. La longitud del perímetro de un área es la longitud de la delimitación, esto es el número de pixeles en la delimitación del patrón. El número de pixeles delimitadores puede ser un acumulado conforme el patrón va creciendo. Un problema con la longitud del perímetro en un objeto reconocido es que el número de pixeles en el límite varía con la orientación del objeto.

El uso de descriptores es para saber algunas características que forman la imagen y que el usuario al aplicar dicho algoritmo descriptor segmentará una parte de la imagen para obtener información como puede ser la detección de borde, color, movimiento, entre otras.

Entonces *segmentar* significa partir los límites de la forma u objeto dentro de piezas simples y entonces la construcción de descripciones de las piezas individuales y de la relación entre las mismas es hecha. La segmentación puede ser calculada por algoritmos semejantes hasta la detección de bordes y de crecimiento de regiones que son usados para segmentar imágenes de dos dimensiones.

El objetivo de la segmentación es ubicar las componentes de una imagen dentro de subconjuntos que correspondan a los objetos físicos en la escena.

3.2.1 Detección de esquinas

Una curva en una fotografía digital se representa por una secuencia de puntos $\{(x_i, y_i)\}$ o de manera más compacta por su *cadena de código* [10,11].

La cadena de código es la codificación del contorno de una curva arbitraria que se representa por una secuencia de vectores de longitud unitaria y dirección definida. Se emplean redes de vectores que tienen 4, 6, 8 y 16 cambios de dirección, en la figura 3.3(b) se presenta una de 8 direcciones. Dados algunos

pares consecutivos sobre la curva, (x_i, y_i), (x_{i+1}, y_{i+1}) hay solo ocho posibles lugares para (x_{i+1}, y_{i+1}) relativo a (x_i, y_i), de manera que la curva puede ser representada por una secuencia de cambios de dirección. La figura 3.3(a) contiene una simple curva digital y su representación de código de cambio de dirección se muestra en la figura 3.3(c). El código es una representación compacta para una curva digital [10,11].

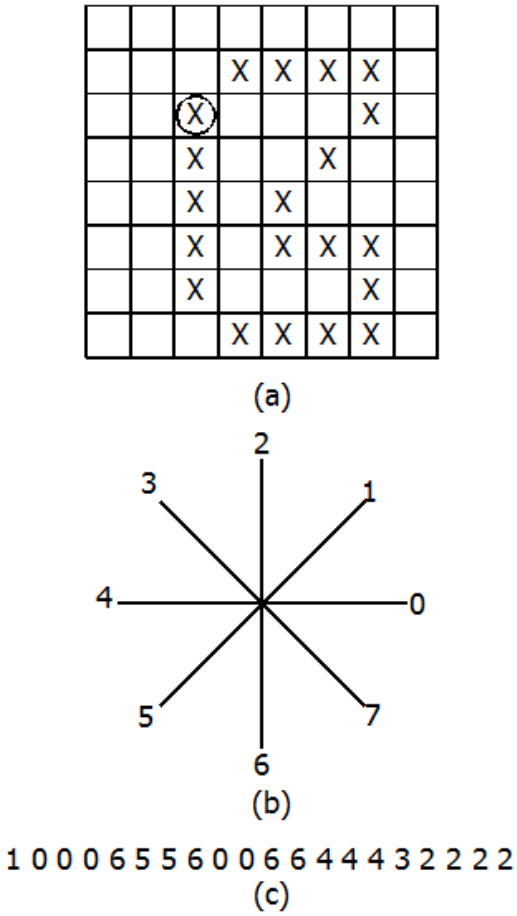


Figura 3.3. Representación del código cadena de una curva digital. (a) Curva digital – el pixel encerrado en un circulo es el punto de inicio; (b) dirección de la cadena código; (c) cadena código para la curva en (a), [14, 25].

3.3 Algoritmos descriptores de formas

El análisis de forma o contorno de un objeto es un problema fundamental en el reconocimiento de patrones y visión por computadora. Con el término "*forma*" nos referimos a las propiedades geométricas invariables entre un conjunto de

características de un objeto. La apariencia de éste en una imagen, puede ser modelada usando representaciones de formas apropiadas de dos dimensiones y entonces tales ejemplos de objetos pueden ser reconocidos y cuantitativamente examinados por comparación de su apariencia actual, con otro de apariencia similar según lo determinado por el modelo.

La representación puede estar basada sobre una descripción de los *límites* del objeto o de las *regiones* que comprende el mismo. La representación puede ser *estructural* en el sentido de la forma, por ejemplo está la de los límites o áreas, esta se rompe dentro de piezas y las propiedades de las piezas son codificadas o puede ser *global* en cuyo caso las propiedades de toda la forma es codificada. Finalmente, las representaciones pueden ser *monolíticas* o *jerárquicas*. La representación jerárquica ordinariamente implica describir la forma en múltiples escalas discretas, mientras que la representación monolítica explícitamente codifica solo propiedades a una sola escala.

3.3.1 Detección de borde y curva

Esta es una de las operaciones fundamentales en el procesamiento de imágenes y un paso inicial para los algoritmos previo a la identificación de objetos. Los bordes tienen cambios de intensidad en una forma discontinua. En dos dimensiones, los bordes tienen una dirección así como una magnitud y el perfil de la intensidad es asumida por ser más o menos uniforme a lo largo del borde [6].

En la literatura se encuentran los métodos de Sobel y Laplace, suficientemente básicos, que encuentran aplicabilidad en la industria. A continuación se explican a detalle los dos.

3.3.1.1 Método del Gradiente

El gradiente de un pixel con coordenadas (x, y) en una imagen $f(x, y)$ es definido como un vector de dos dimensiones [5].

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.3}$$

El vector o gradiente $\mathbf{G}$ apunta en la dirección de la máxima tasa de cambio de f de la ubicación (x, y). Para la detección del borde, sin embargo, estamos interesados en la magnitud de este vector, generalmente referido como el *gradiente* y denotado por $G[f(x,y)]$, donde:

$$G[f(x,y)] = \left[G_x^2 + G_y^2\right]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \tag{3.4}$$

La ecuación (4.4), calcula el gradiente basado en la obtención de las derivadas parciales $\partial f/\partial x$ y $\partial f/\partial y$ para cada pixel. Para hacer esto, implica usar una subimagen o vecindad de 3×3 pixeles que encierra a un pixel central (x, y).

El resultado de esta operación sirve para detectar un borde y está presente cuando la magnitud del gradiente excede un cierto umbral. Presentamos dos formas para calcular el gradiente: el método de Sobel y Laplace. Ambos utilizan los valores de intensidad $I(x, y)$, figura 3.4, en una vecindad de 3×3 pixeles para calcular la magnitud del vector y dirección del borde en los dos métodos. El pixel central (x, y) tiene una intensidad $I(x, y) = a_4$ [6].

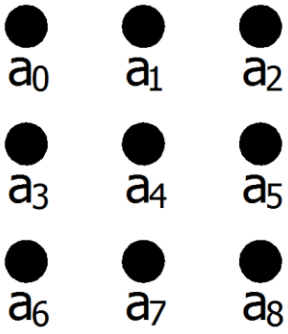


Figura 3.4. Valores de intensidad en una vecindad de 3x3.

3.3.1.2 Método de Sobel

Para calcular el gradiente, se utilizan dos mascaras, que son las mostradas en la figura 3.5.

-1	-2	-1
	(x, y)	
1	2	1

(a)

-1		1
-2	(x, y)	2
-1		1

(b)

Figura 3.5. (a)Mascara para calcular Sy en el pixel central. (b) Mascara para calcular Sx para el mismo pixel.

Estas dos mascaras son comúnmente conocidas como operadores Sobel, al combinarlos, para cualquier pixel con coordenadas (x, y), se obtiene el gradiente en ese punto y hace resaltar el borde [5, 7].

El gradiente esta dado por las componentes S_x y S_y en direcciones x y y, definidas como [11]:

$$S_x = (a_2 + 2a_5 + a_8) - (a_0 + 2a_3 + a_6) \tag{3.5}$$

$$S_y = (a_6 + 2a_7 + a_8) - (a_0 + 2a_1 + a_2) \tag{3.6}$$

$$Magnitud\ del\ gradiente = \sqrt{S_x^2 + S_y^2} \tag{3.7}$$

En la práctica, para utilizarlo en programación de algoritmos, se aproxima la raíz cuadrada de la suma de los cuadrados mediante los valores absolutos de las componentes indicadas.

$$Magnitud\ del\ gradiente = |S_x| + |S_y| \tag{3.8}$$

Esta aproximación es más accesible de realizar. El ángulo queda definido de la siguiente forma:

$$Dirección\ del\ gradiente = \tan^{-1}\left(\frac{S_x}{S_y}\right) \quad (3.9)$$

El operador S_x solo encuentra discontinuidades en la dirección x (líneas verticales), mientras que S_y solo las encuentra en la dirección y (líneas horizontales).

Los resultados de las ecuaciones 3.8 y 3.9 contienen la magnitud y dirección de cada pixel. A partir de esto, es posible segmentar pixeles que forman parte de la frontera de los que no lo son. Para hacerlo, es importante escoger un umbral adecuado.

Al escoger un umbral alto, el contorno de la figura estará más definido, la desventaja es la disminución de pixeles que lo forman y se pierde resolución. Un umbral con valor reducido dará como resultado contornos demasiado gruesos [7].

El método de Sobel, al aplicarlo a una vecindad de 3×3 reduce el efecto del ruido, mientras que los pixeles con intensidad $a_1, a_3, a_5\ y\ a_7$ al duplicar su valor, impide que el borde se oscurezca o desaparezca con el fondo [5, 6]. Se usa más frecuentemente para aplicaciones robóticas.

3.3.1.3 Método de Laplace

El operador de Laplace produce una derivada local de segundo orden de una imagen $f(x, y)$ en escala de grises definida como [12]:

$$L[f(x,y)] = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (3.10)$$

Utiliza también una máscara, figura 3.6, conocida comúnmente como operador de Laplace [12]. Tomando los valores de intensidad de la figura 3.4, se define el operador digital de Laplace en el pixel (x, y) con nivel gris como [12]:

$$L[f(x,y)] = 4a_4 - a_1 - a_3 - a_5 - a_7 \quad (3.11)$$

Esto se debe hacer para cada pixel de la imagen.

	-1	
-1	4	-1
	-1	

Figura 3.6. Operador de Laplace [12].

El detector de borde de Sobel es simple y reducido en cálculo, se desenvuelve bien en escenas de bajo ruido y textura.

3.4 El histograma y su ecualización

El histograma es un gráfico que relaciona los niveles de intensidad de una imagen y el número de pixeles que poseen tal nivel de intensidad, esto es:

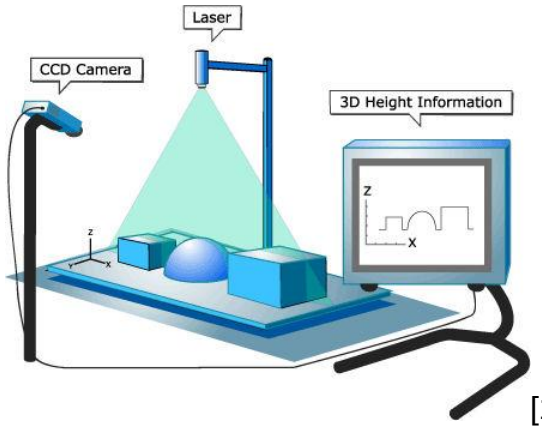
$$f(r_k) = n_k \tag{3.12}$$

Donde r_k representa el valor de la $k - ésima$ intensidad en el intervalo $[0, I_{max}]$ y n_k es el número de píxeles que posee la intensidad r_k. Por ejemplo, una imagen de tipo $uint8, r_k$ tomará los valores entre el intervalo $[0 - 255]$. Representar el histograma de la imagen es programar la ecuación (4.12).

El histograma puede estar normalizado, en tal caso:

$$h(r_k) = f(r_k)/n = n_k/n \tag{3.13}$$

Donde n representa el número total de pixeles de la imagen.



[16]

Capítulo 4. ALGORITMOS

4.1 Algoritmos en Matlab

Se presentan algoritmos en Matlab utilizados en el procesamiento de imágenes para: conocer los valores discretos reales de cada componente RGB, en escala de grises y binaria, además del histograma. En una imagen con formato JPG, los valores de las componentes, rojo, verde y azul (RGB) están en el intervalo de intensidad de 0 hasta 255 cada una. La imagen en escala de grises tiene la tonalidad de intensidades en el mismo intervalo. En una imagen binarizada aparecerán los valores de 0 ó 1 en los pixeles respectivos.

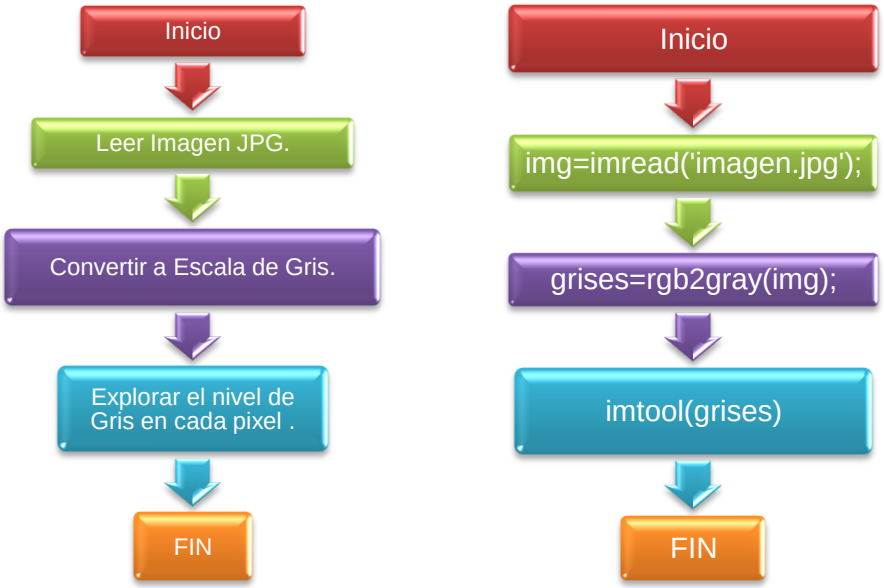
Posteriormente se muestran los algoritmos descriptores en lenguaje C del área y el perímetro.

El diagrama de flujo para leer y explorar el nivel RGB de cada pixel está en la figura 4.1.

(a) (b)

Figura 4.1. (a) Algoritmo para leer una imagen JPG y el nivel de intensidad RGB de los pixeles. (b) Comandos usando Matlab.

A partir de una imagen a color se obtiene la equivalente en escala de grises para explorar las intensidades de gris en la matriz. El diagrama de flujo y los comandos están en la figura 4.2



(a) (b)

Figura 4.2. (a) Algoritmo, (b) comandos usando Matlab para conocer las intensidades de grises

En una imagen en blanco y negro, los pixeles toman los valores, 1 ó 0, respectivamente. Al proceso de convertir una imagen RGB a blanco y negro se le denomina binarización de la imagen. Para obtenerla, es necesario hacer la conversión de una imagen JPG a escala de grises, sacar su histograma y escoger el umbral óptimo que le dará realce a la figura en cuestión, y por último, se exploran los pixeles. El valor del umbral se especifica para designar con un valor de 1 a los pixeles cuyo valor de gris sea mayor al indicado y cero para los demás. En la figura 4.3(a) se muestran los comandos en Matlab para obtener la binarización.

(a) (b)

Figura 4.3. (a) Diagrama de flujo (b) algoritmo en Matlab para binarizar la imagen.

La instrucción “binarización” constituye un algoritmo hecho en Matlab y las instrucciones se muestran en la figura 4.4.

```
function
[I]=Binarizacion(imagen,umbral)
[f,c]=size(imagen);
  for i=1:f
    for j=1:c
      if double(imagen(i,j))<umbral
          I(i,j)=0;
      else
          I(i,j)=1;
       end
    end
end
I=uint8(I);
```

Figura 4.4. Instrucciones para el comando binarización.

En la figura 4.5(a) está el diagrama de flujo para extraer el histograma con los comandos en Matlab, figura 4.5(b).

(a) (b)

Figura 4.5. (a) Diagrama de Flujo para obtener el Histograma, (b) algoritmo en Matlab.

El histograma normalizado representa la probabilidad de gris que hay en un nivel determinado y se calcula en Matlab al aplicar la ecuación con los siguientes comandos:

nkk=nk./sum(nk(:,1)); y bar(rk, nkk, 1);

4.2 Algoritmos

A continuación se muestran los dos algoritmos programados en microcontroladores para calcular el área y el perímetro de una figura. Por la versatilidad en usar instrucciones de decisión, operadores aritméticos, funciones, arreglos y cadenas, así como apuntadores y estructuras están programados en lenguaje C. Todos estos recursos reducen el programa y se ejecuta eficientemente. Se utiliza el compilador WinAvr para crear ambos algoritmos.

4.2.1 Algoritmo para Área

El algoritmo, figura 4.6, calculará el área utilizando un ciclo "for" exterior para revisar primero los pixeles de la fila superior y después pasa a la siguiente hasta revisar todas las filas, posteriormente se inserta otro ciclo "for" anidado para inspeccionar cada pixel de cada columna a la vez. Al terminar de inspeccionar cada pixel de cada renglón y columna, se deposita el resultado en el puerto PORTC. Naturalmente que este resultado tendrá una finalidad, ya sea que sirva para mover un brazo robótico, una cinta transportadora u otra acción de manufactura o ensamble.

```
area = 0;
for(x=0; x<16; x++)
{
        for(y=0; y<16; y++)
                {
                if(imagen[x][y] <= 40) // intervalo valido o umbral
                area++;
                PORTC = area;
                }
        }
```

Figura 4.6. Algoritmo para calcular el área en una imagen de 16x16 pixeles en escala de grises (0-255).

4.2.2 Algoritmo para el Perímetro

Para el perímetro, se utiliza el Método digital de Laplace, figura 4.7. Se toma el valor de la intensidad del pixel central a para multiplicarlo por cuatro y se le sustrae el valor del pixel de su derecha e izquierda así como el valor del pixel superior e inferior inmediatos a él, ecuación (4.1).

	PIXEL c	
PIXEL d	PIXEL a	PIXEL b
	PIXEL e	

Figura 4.7. Ubicación de los pixeles.

$$Laplace = delta = (4 * a) - b - c - d - e \tag{4.1}$$

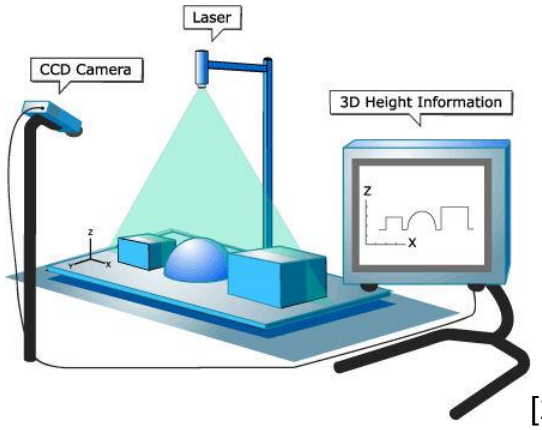
Con una sola corrida sobre todos los pixeles se calcula el perímetro. No hay píxeles más allá de la frontera exterior de una imagen y se evitar un error de acceso fuera de límites definidos. Por lo tanto, el lazo inicia en el segundo renglón y termina una fila antes. Si se requiriera estos dos renglones se ajusta a cero. El algoritmo también limita un valor máximo para el *blanco* (255), de modo que cualquier valor resultante permanece dentro del rango del dato tipo byte.

El algoritmo programado en lenguaje C se observa en la figura 4.8, se utilizan dos ciclos "for", el primero recorre renglones y el segundo columnas. El resultado de Laplace se deposita en la variable "$delta$" tipo int que tiene un intervalo de $-32{,}768$ a $+\ 32{,}767$. Si este es negativo y se encuentra en el intervalo especificado, pertenecerá al perímetro o contorno de la figura. Se lleva la cuenta en otra variable llamada "$perímetro$" y se evalúa el próximo pixel de la derecha con el mismo método de Laplace hasta terminar todos los renglones. El resultado se coloca en el puerto PORTC para activar actuadores en la línea de ensamble de la celda de manufactura.

```
perimetro=0;

for(j = 1; j < (height - 1) ; j++)
{
        for(x=1 ; x < (width - 1) ; x++) //columnas
        {
                                //      b
                                //e     a       c       delta[]=(4*a) -e-c-b-d
                                //      d
                delta = (4 * imagen[j][x]) - imagen[j][x-1] - imagen[j][x+1] - imagen[j-1][x] -
imagen[j+1][x];
                if(delta[y]< (-200) )
                {       perimetro++;  }
                else
                {
                }

                PORTC = perimetro;
        }

}
```

Figura 4.8. Algoritmo para calcular el perímetro en una imagen de 16x16 pixeles en escala de grises (0-255).

El operador digital de Laplace es la instrucción *delta*, donde se ubican los cinco pixeles involucrados.



[16]

Capítulo 5. PRUEBAS Y RESULTADOS

5.1 Procesamiento de imágenes por computadora

Para verificar los marcadores de un JPEG o imagen con formato JPG se hace lo siguiente:

1. Una imagen lo más pequeña posible con extensión JPG.
2. Leer el archivo con un editor hexadecimal (el Frhed hex editor).

La imagen propuesta tiene un tamaño de 1×1, o sea un pixel de color rojo con valores en las componentes $\mathrm{RGB}(255,0,0)$, figura 5.1. Se Observan otras características como: matiz, saturación, luminancia; finalmente el archivo tiene un tamaño de 633 bytes.

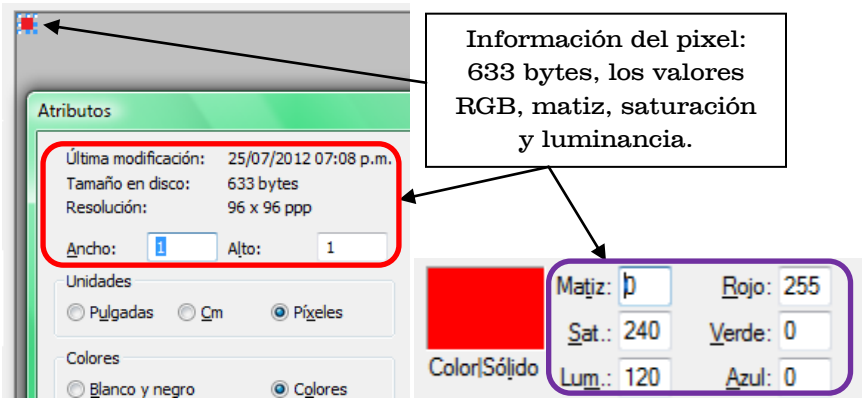


Figura 5.1. Imagen de 1 × 1, color rojo y sus valores RGB.

Al guardarla con la extensión JPG, la comprensión se lleva a cabo. Se abre ahora con el editor hexadecimal, véase la figura 5.2. Y aparecen los marcadores anteriormente mencionados. No se identifica el pixel rojo de forma directa con componentes $RGB\ (255,0,0)$ ó $RGB(FF,00,00)$, es decir, la codificación JPEG es toda la información del pixel rojo y se encuentra en el archivo.

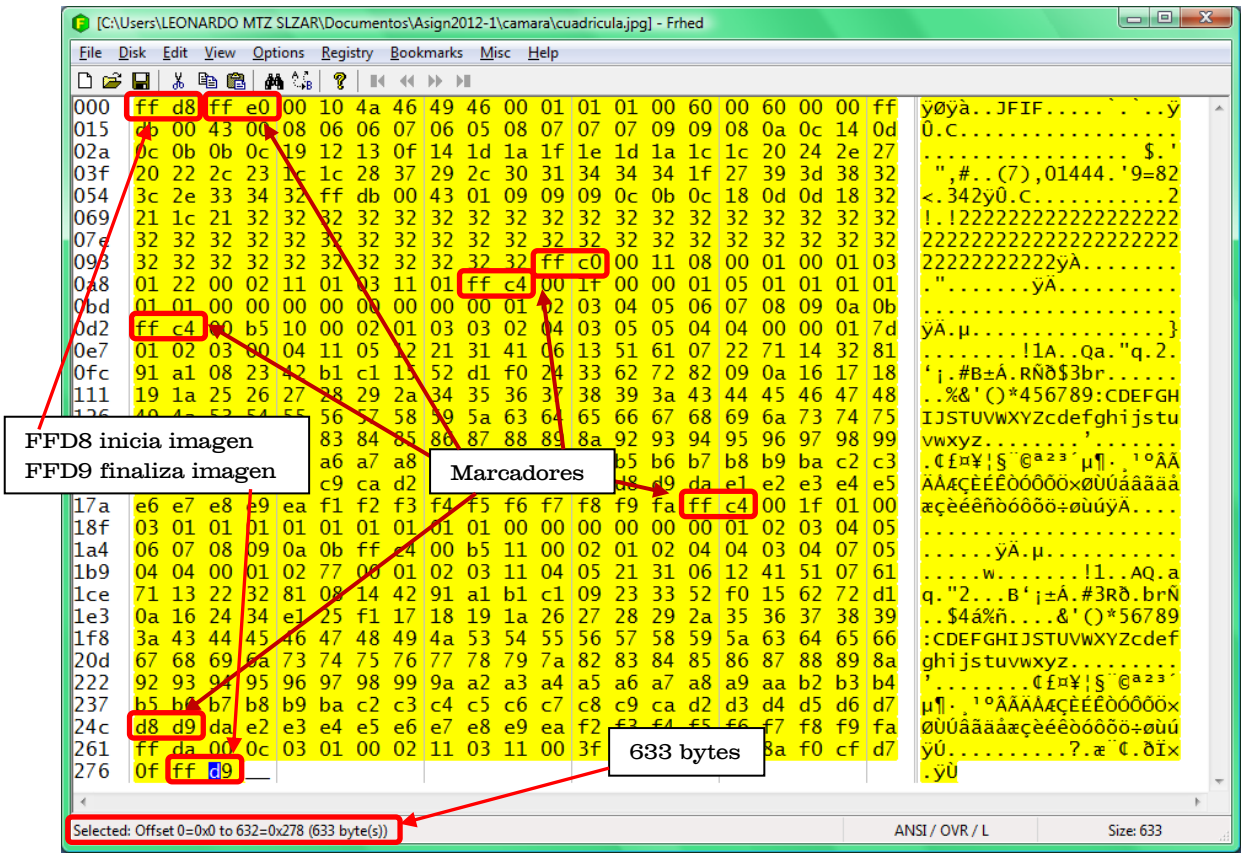


Figura 5.2. Imagen 1x1 pixeles en formato hex.

Ahora, en MATLAB, se decodifica la imagen, se inspecciona cuantos pixeles hay y que valores tienen sus componentes, en la figura 5.3 se observa que el valor de las componentes RGB (237, 27, 36) del pixel varía.

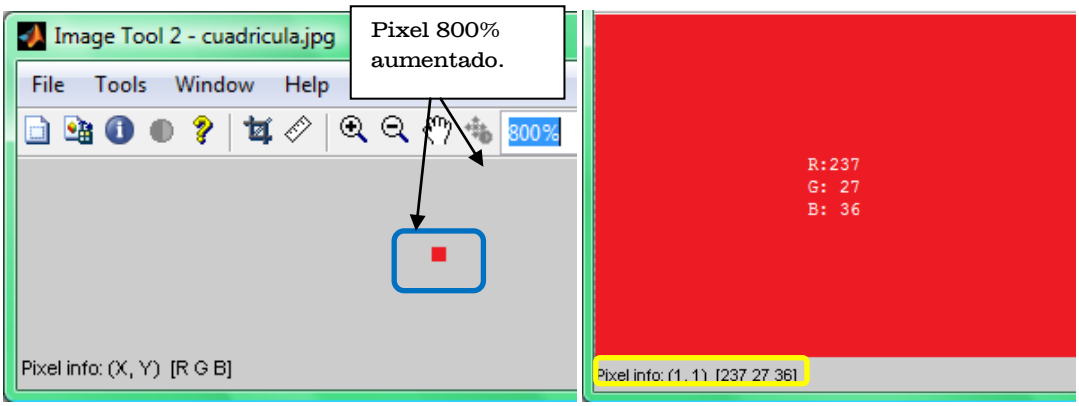


Figura 5.3. Inspección de la imagen-pixel y sus valores RGB.

Se confirma que es solo un pixel. Como se ha comprobado, la comprensión de una imagen tiene una gran cantidad de información, para el caso de 1×1 (un pixel), ocupa más espacio la comprensión en la memoria que los valores de las componentes. Sin embargo, la cantidad de datos en otros archivos de imágenes más grandes llega a ser en millones de bytes. Por lo tanto es necesaria la comprensión.

La figura 5.4 muestra una imagen JPG y se indican dos áreas cuyo nivel RGB se muestra en las figuras 5.5(a) y (b).

Figura 5.4. Imagen JPG del espacio escultórico de la U.N.A.M. Ciudad de México.

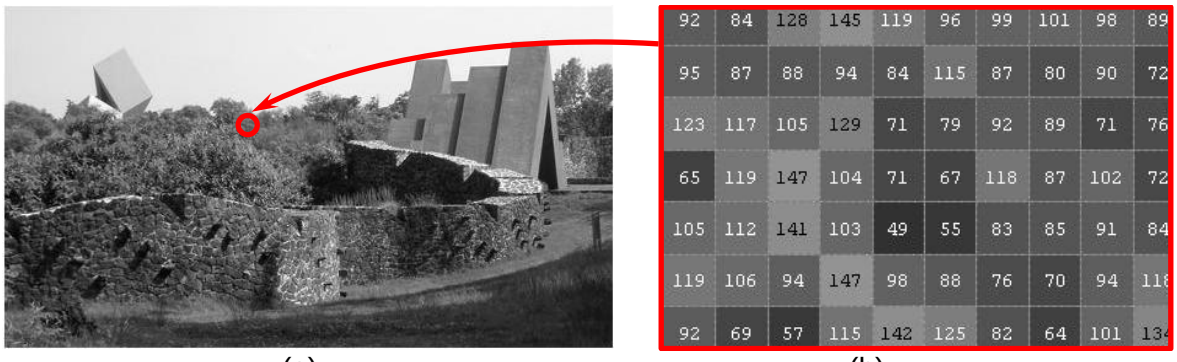
G: 65 B: 59	G: 65 B: 59	G: 43 B: 37	G: 37 B: 40	G: 33 B: 38	G: B:
R:126 G: 77 B: 70	R:109 G: 54 B: 49	R:106 G: 51 B: 46	R: 87 G: 37 B: 38	R: 90 G: 33 B: 40	R: G: B:
R:238 G:224 B:198	R:146 G:117 B:103	R: 88 G: 47 B: 43	R: 92 G: 41 B: 40	R: 95 G: 36 B: 40	R: G: B:
R:253 G:245 B:206	R:255 G:255 B:228	R:200 G:188 B:172	R:119 G: 80 B: 75	R: 83 G: 38 B: 35	R: G: B:

G:69 B:10	G:71 B:17	G:73 B:20	G:67 B:12	G:69 B:10	G: B:
R:42 G:70 B:12	R:46 G:75 B:19	R:47 G:76 B:22	R:43 G:72 B:16	R:45 G:73 B:14	R: G: B:
R:35 G:65 B: 5	R:44 G:73 B:15	R:48 G:77 B:23	R:44 G:71 B:16	R:43 G:71 B:12	R: G: B:
R:34 G:64 B: 2	R:44 G:75 B:16	R:49 G:78 B:22	R:42 G:71 B:15	R:38 G:66 B: 8	R: G: B:

(a) Área 1 (b) Área 2

Figura 5.5. Valores RGB de los pixeles en las áreas indicadas en la figura 4.11.

La misma imagen, figura 5.4, se presenta en escala de grises en la figura 5.6(a) y están los valores de gris en la figura 5.6 (b) para el área especificada.

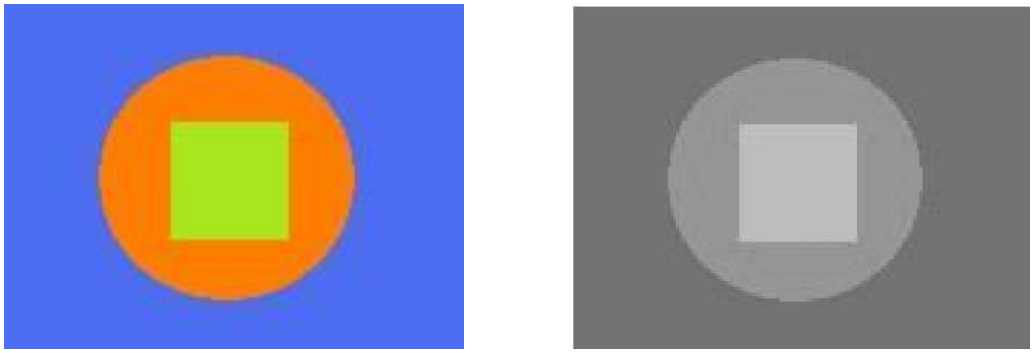


92	84	128	145	119	96	99	101	98	89
95	87	88	94	84	115	87	80	90	72
123	117	105	129	71	79	92	89	71	76
65	119	147	104	71	67	118	87	102	72
105	112	141	103	49	55	83	85	91	84
119	106	94	147	98	88	76	70	94	118
92	69	57	115	142	125	82	64	101	134

(a) (b)

Figura 5.6. (a) Conversión a escala de grises. (b) Valores de gris en el área señalada.

La imagen de la figura 5.7(a) tiene formato JPG y la conversión en escala de grises está en la figura 5.7(b).



(a) (b)

Figura 5.7. (a) Imagen jpg. (b) En escala de grises.

En la figura 5.8 y 5.9 están los histogramas de las figuras 5.6(a) y 5.7(b) respectivamente.

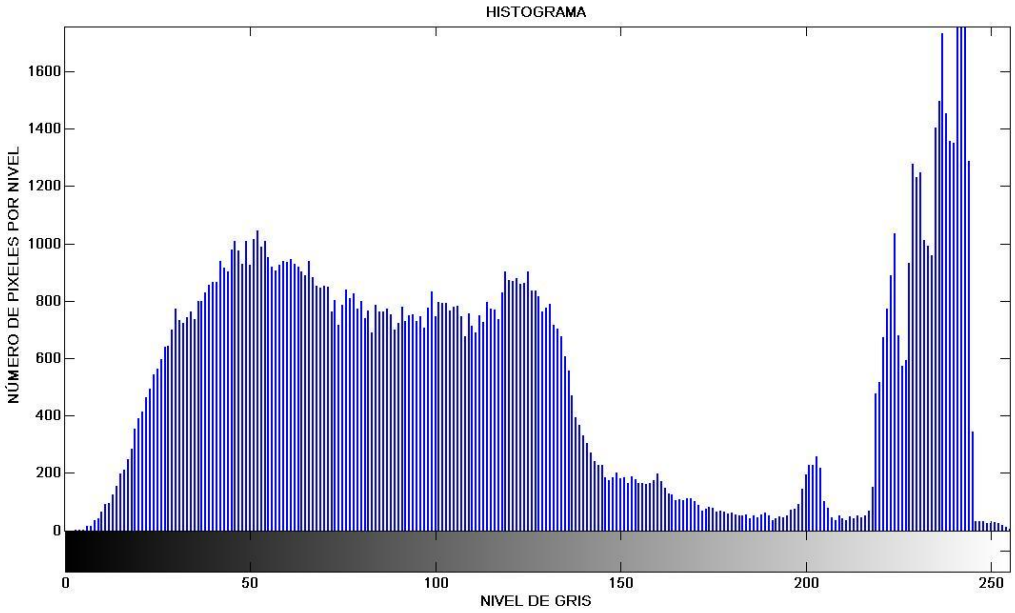


Figura 5.8. Histograma de la figura 5.6(a).

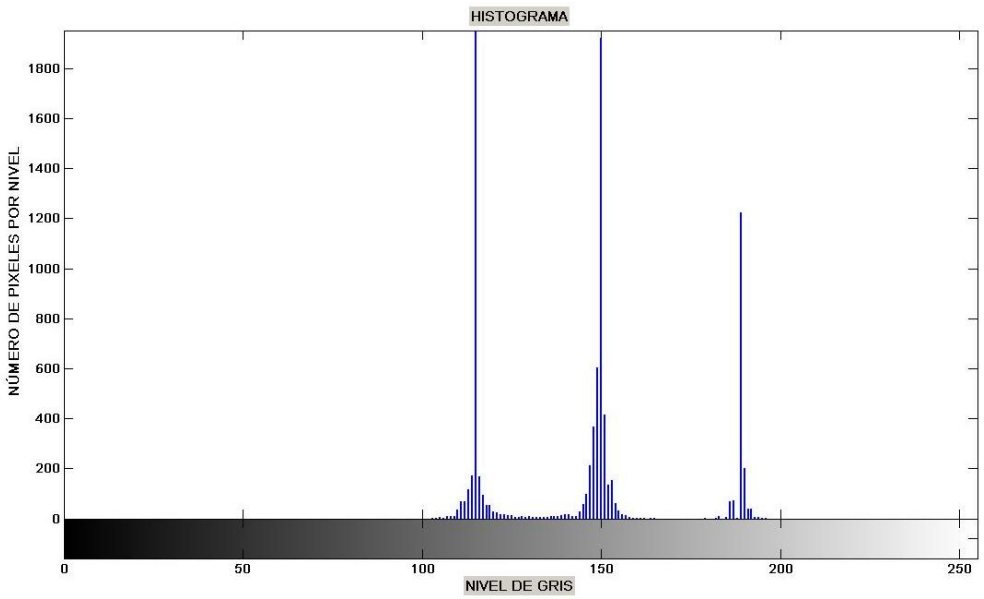


Figura 5.9. Histograma de la figura 5.7(b).

El histograma normalizado representa la probabilidad de gris que hay en un nivel determinado.

El resultado de esta operación se muestra en la figura 5.10.

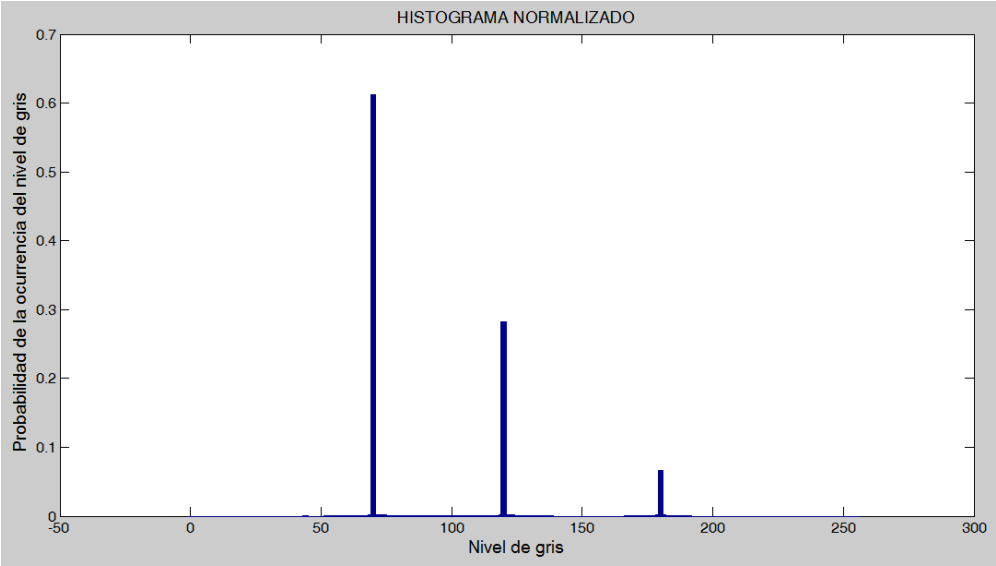


Figura 5.10. Histograma normalizado.

Al observar el histograma de una imagen se decide el umbral optimo para segmentar los pixeles que integraran la figura de aquellos que no y se binariza.

5.2 Algoritmos descriptores

Para las figuras propuestas que se les aplicaran los algoritmos, están en escala de grises, el color blanco y negro se representa con el numeral 255 y 0 respectivamente y la variación de gris de un pixel está en ese intervalo. La matriz imagen propuesta es de 16×16 pixeles, 256 datos. Para que un pixel sea considerado parte de la figura, el algoritmos segmentara los pixeles grises que están en el intervalo valido de 0-40 (0-28 hex). Para un pixel blanco o que forma parte del fondo, validará de 230 hasta 255 del intervalo. Para cada una de las figuras examinadas ambos algoritmos inician en la $fila$ 1 y $columna$ 1, finalizando en $filas - 1$ y $columnas - 1$.

La figura 5.11, muestra el patrón de la imagen (16×16) que se usa, cada pixel está formado por un cuadro blanco o negro. La forma de ubicar un pixel es así: $PIXEL\ (renglón, columna)$.

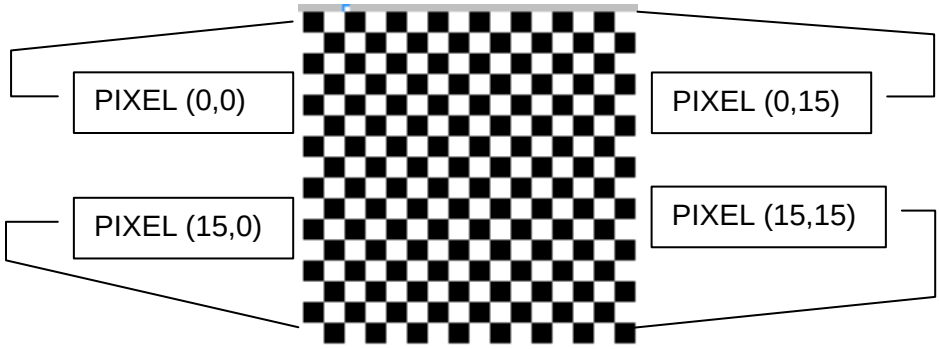
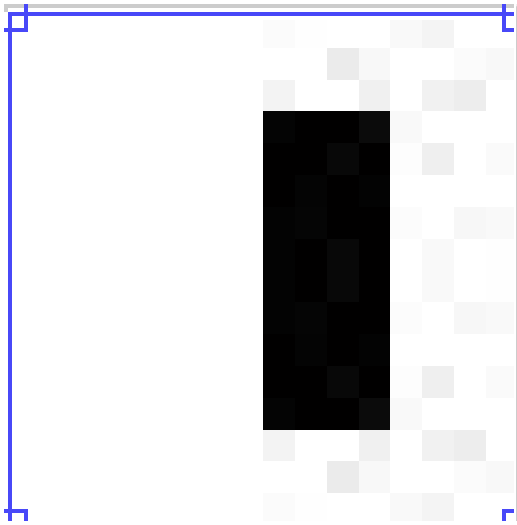


Figura 5.11. Patrón básico para usarlo en el cálculo del área y el perímetro de figuras planas de 2D.

Se proponen las siguientes figuras: rectángulo, triangulo rotado, circulo y una copa. Obteniendo los valores de la matriz de pixeles a través de Matlab para posteriormente introducirlos en una matriz de datos y ejecutados en el programa AVR Studio 4 (Versión 4.18), cada algoritmo calcula el área y perímetro. A continuación los resultados.

Se plantea un rectángulo en color negro, figura 5.12(a), al cual se le calcula el área y el perímetro.



255	255	255	255	255	255	255	255	251	254	255	255	249	244	255	255
255	255	255	255	255	255	255	255	255	255	235	248	255	255	251	248
255	255	255	255	255	255	255	255	243	255	255	240	255	241	237	255
255	255	255	255	255	255	255	255	4	0	0	11	249	255	255	255
255	255	255	255	255	255	255	255	0	0	8	0	253	239	255	250
255	255	255	255	255	255	255	255	0	4	0	3	255	255	255	255
255	255	255	255	255	255	255	255	3	5	0	0	252	255	247	249
255	255	255	255	255	255	255	255	3	0	8	0	255	249	255	254
255	255	255	255	255	255	255	255	3	0	8	0	255	249	255	254
255	255	255	255	255	255	255	255	3	5	0	0	252	255	247	249
255	255	255	255	255	255	255	255	0	4	0	3	255	255	255	255
255	255	255	255	255	255	255	255	0	0	8	0	253	239	255	250
255	255	255	255	255	255	255	255	4	0	0	11	249	255	255	255
255	255	255	255	255	255	255	255	243	255	255	240	255	241	237	255
255	255	255	255	255	255	255	255	255	255	235	248	255	255	251	248
255	255	255	255	255	255	255	255	251	254	255	255	249	244	255	255

(a) (b)

Figura 5.12. (a)Rectángulo en 2D para calcular el área y el centroide. (b)Valores de cada pixel en escala de grises obtenidos en Matlab con el comando Imtool.

Examinando la figura 5.12(b) el valor más alto para un pixel negro es 11 (0x0B) y el valor más bajo para un pixel blanco es 235 (0xEB). Se introducen los valores de los 256 pixeles en una matriz con el nombre de “imagen” como se aprecia en la figura 5.13.

```
unsigned char imagen[16][16] = {

    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFB,0xFE,0xFF,0xFF,0xF9,0xF4,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xEB,0xF8,0xFF,0xFF,0xFF,0xF8},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF3,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x04,0x00,0x00,0x0B,0xF9,0xFF,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x00,0x08,0x00,0xFD,0xEF,0xFF,0xFA},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x04,0x00,0x03,0xFF,0xFF,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x03,0x05,0x00,0x00,0xFF,0xFF,0xF7,0xF9},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x03,0x00,0x08,0x00,0xFF,0xF9,0xFF,0xFE},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x03,0x00,0x08,0x00,0xFF,0xF9,0xFF,0xFE},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x03,0x05,0x00,0x00,0xFC,0xFF,0xF7,0xF9},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x04,0x00,0x03,0xFF,0xFF,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x00,0x08,0x00,0xFD,0xEF,0xFF,0xFA},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x04,0x00,0x00,0x0B,0xF9,0xFF,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF3,0xFF,0xFF,0xF0,0xFF,0xFF,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xEB,0xF8,0xFF,0xFF,0xFF,0xFF},
    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFB,0xFE,0xFF,0xFF,0xF9,0xF4,0xFF,0xFF},};
```

Figura 5.13. Matriz de valores de cada uno de los 256 pixeles de la imagen.

Al aplicar el algoritmo para calcular el área, se producen los siguientes resultados, figura 5.14, es:

$$Área\ del\ rectángulo\ =\ 40\ pixeles$$

$$Perímetro\ del\ rectángulo\ =\ 24\ pixeles$$

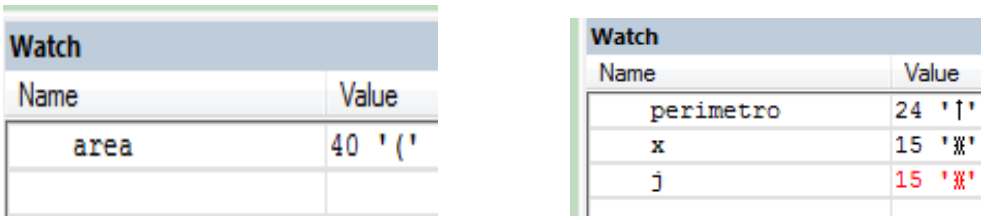
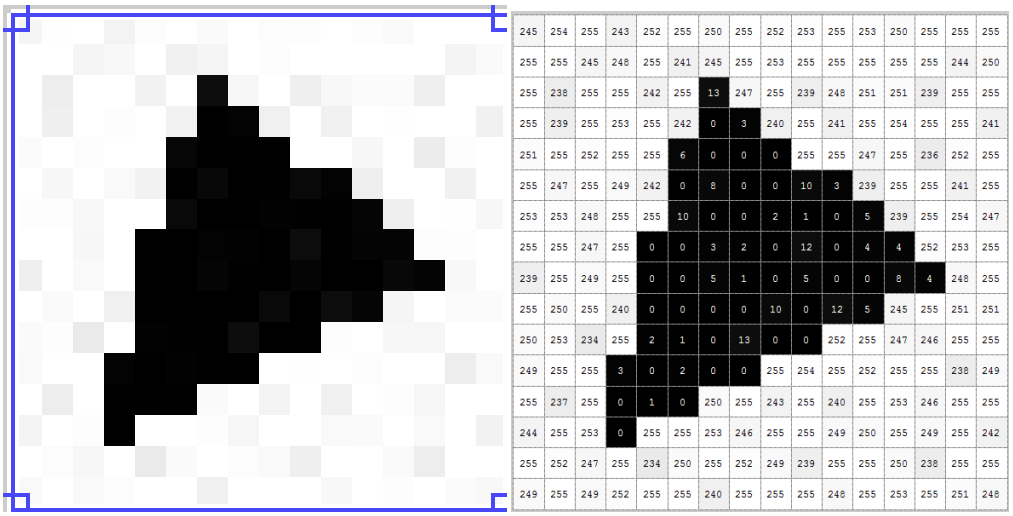


Figura 5.14. Resultado después de correr el algoritmo para calcular el área y el perímetro.

Se propone ahora un triángulo rotado, como se muestra en la figura 5.15.



(a) (b)

Figura 5.15. (a)Triángulo rotado en 2D para calcular el área y el perímetro. (b) Valores de cada pixel en escala de grises obtenidos en Matlab con el comando Imtool.

El valor más alto para un pixel negro es 12 (0x0C) y el más bajo para un pixel blanco es 238 (0xEE). Se introducen los valores de los 256 pixeles en una matriz con el nombre de “imagen”, figura 5.16.

```
unsigned char imagen[16][16] = {
{245,254,255,243,252,255,250,255,252,253,255,253,250,255,255,255},
{255,255,245,248,255,241,245,255,253,255,255,255,255,255,244,250},
{255,238,255,255,242,255,013,247,255,239,248,251,251,239,255,255},
{255,239,255,253,255,242,000,003,240,255,241,255,254,255,255,241},
{251,255,252,255,255,006,000,000,000,255,255,247,255,236,252,255},
{255,247,255,249,242,000,   8,000,000,010,003,239,255,255,241,255},
{253,253,248,255,255,010,000,000,002,001,000,005,239,255,254,247},
{255,255,247,255,000,000,003,002,000,012,000,004,004,252,253,255},
{239,255,249,255,000,000,005,001,000,005,000,000,  8,004,248,255},
{255,250,255,240,000,000,000,000,010,000,012,005,245,255,251,251},
{250,253,234,255,002,001,000,013,000,000,252,255,247,246,255,255},
{249,255,255,003,000,002,000,000,255,254,255,252,255,255,238,249},
{255,237,255,000,001,000,250,255,243,255,240,255,253,246,255,255},
{244,255,253,000,255,255,253,246,255,255,249,250,255,249,255,242}
{255,252,247,255,234,250,255,252,249,239,255,255,250,238,255,255},
{249,255,249,252,255,255,240,255,255,255,248,255,253,255,251,248},};
```

Figura 5.16. Matriz de valores de cada uno de los 256 pixeles de la imagen.

El resultado de ambos algoritmos, figura 5.17, es:

$$Área\ del\ triángulo\ =\ 62\ pixeles$$

$$Perímetro\ del\ triángulo\ =\ 28\ pixeles$$

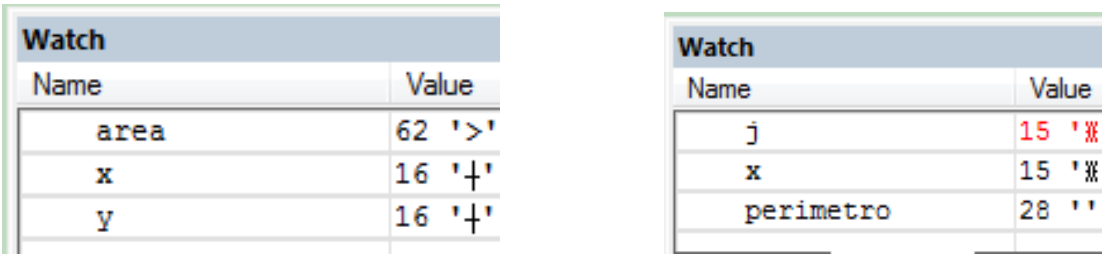
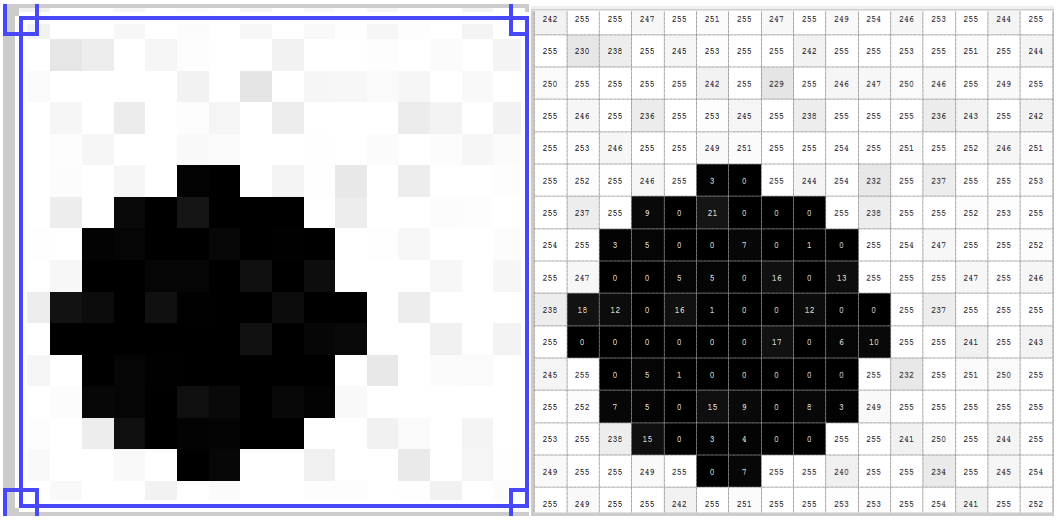


Figura 5.17. Resultado después de correr el algoritmo para calcular el área y perímetro del triángulo rotado.

Se propone ahora un círculo como se muestra en la figura 5.18.



242	255	255	247	255	251	255	247	255	249	254	246	253	255	244	255
255	230	238	255	245	253	255	255	242	255	255	253	255	251	255	244
250	255	255	255	255	242	255	229	255	246	247	250	246	255	249	255
255	246	255	236	255	253	245	255	238	255	255	255	236	243	255	242
255	253	246	255	255	249	251	255	255	254	255	251	255	252	246	251
255	252	255	246	255	3	0	255	244	254	232	255	237	255	255	253
255	237	255	9	0	21	0	0	0	255	238	255	255	252	253	255
254	255	3	5	0	0	7	0	1	0	255	254	247	255	255	252
255	247	0	0	5	5	0	16	0	13	255	255	255	247	255	246
238	18	12	0	16	1	0	0	12	0	0	255	237	255	255	255
255	0	0	0	0	0	0	17	0	6	10	255	255	241	255	243
245	255	0	5	1	0	0	0	0	0	255	232	255	251	250	255
255	252	7	5	0	15	9	0	8	3	249	255	255	255	255	255
253	255	238	15	0	3	4	0	0	255	255	241	250	255	244	255
249	255	255	249	255	0	7	255	255	240	255	255	234	255	245	254
255	249	255	255	242	255	251	255	255	253	253	255	254	241	255	252

(a) (b)

Figura 5.18. (a)Circulo en 2D para calcular el área y el perímetro. (b)Valores de cada pixel en escala de grises obtenidos en Matlab con el comando Imtool.

El valor más alto para un pixel negro es 18 (0x12) y el más bajo para un pixel blanco es 229 (0xE5). Se introducen los valores de los 256 pixeles en una matriz con el nombre de “imagen” como se aprecia en la figura 5.19.

Cálculo del área del círculo:

El área real se calcula utilizando la fórmula $A = \pi r^2$. Observando la figura 6.8(a), el radio del cuerpo circular es de 5 pixeles, de una longitud unitaria. En función del cuerpo circular sobre la matriz de pixeles, resulta:

$$A = \pi 5^2 = 78.54\ unidades\ cuadradas$$

```
unsigned char imagen[16][16] = {
        {242,255,255,247,255,251,255,247,255,249,254,246,253,255,244,255},
        {255,230,238,255,245,253,255,255,242,255,255,253,255,251,255,244},
        {250,255,255,255,255,242,255,229,255,246,247,250,246,255,249,255},
        {255,246,255,236,255,253,245,255,238,255,255,255,236,243,255,242},
        {255,253,246,255,255,249,251,255,255,254,255,251,255,252,246,251},
        {255,252,255,246,255,  3,  0,255,244,254,232,255,237,255,255,253},
        {255,237,255,  9,  0, 21,  0,  0,  0,255,238,255,255,252,253,255},
        {254,255,  3,  5,  0,  0,  7,  0,  1,  0,255,254,247,255,255,252},
        {255,247,  0,  0,  5,  5,  0, 16,  0, 13,255,255,255,247,255,246},
        {238, 18, 12,  0, 16,  1,  0,  0, 12,  0,  0,255,237,255,255,255},
        {255,  0,  0,  0,  0,  0,  0, 17,  0,  6, 10,255,255,241,255,243},
        {245,255,  0,  5,  1,  0,  0,  0,  0,  0,255,232,255,251,250,255},
        {255,252,  7,  5,  0, 15,  9,  0,  8,  3,249,255,255,255,255,255},
        {253,255,238, 15,  0,  3,  4,  0,  0,255,255,241,250,255,244,255},
        {249,255,255,249,255,  0,  7,255,255,240,255,255,234,255,245,254},
        {255,249,255,255,242,255,251,255,255,253,253,255,254,241,255,252},};
```

Figura 5.19. Matriz de valores de cada uno de los 256 pixeles de la imagen.

El algoritmo calcula el área de un objeto contando el número de pixeles que ocupa, resultado, figura 5.20, de:

$$Área\ \ del\ circulo\ =\ 68\ pixeles$$

El error que se origina es de:

$$Error = \frac{78.54 - 68}{68} = 0.155 = 15.5\%$$

Ahora el cálculo del perímetro del círculo, se encuentra al efectuar la siguiente fórmula $P = 2\pi r$.Al aplicarla en la pieza circular, resulta:

$$Perímetro = 2\pi 5 = 31.41\ unidades\ lineales$$

Considerando cada pixel de longitud unitaria, el algoritmo cuenta los pixeles del borde del objeto, resulta:

$$Perímetro\ del\ circulo\ =\ 24\ pixeles$$

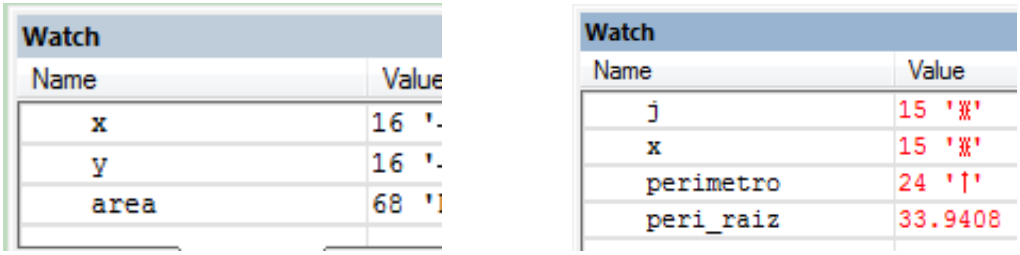


Figura 5.20. Resultado después de correr el algoritmo para calcular el área.

Sin embargo, existe un error:

$$error = \frac{31.41 - 24}{24} = 0.30875 = 30.875\%$$

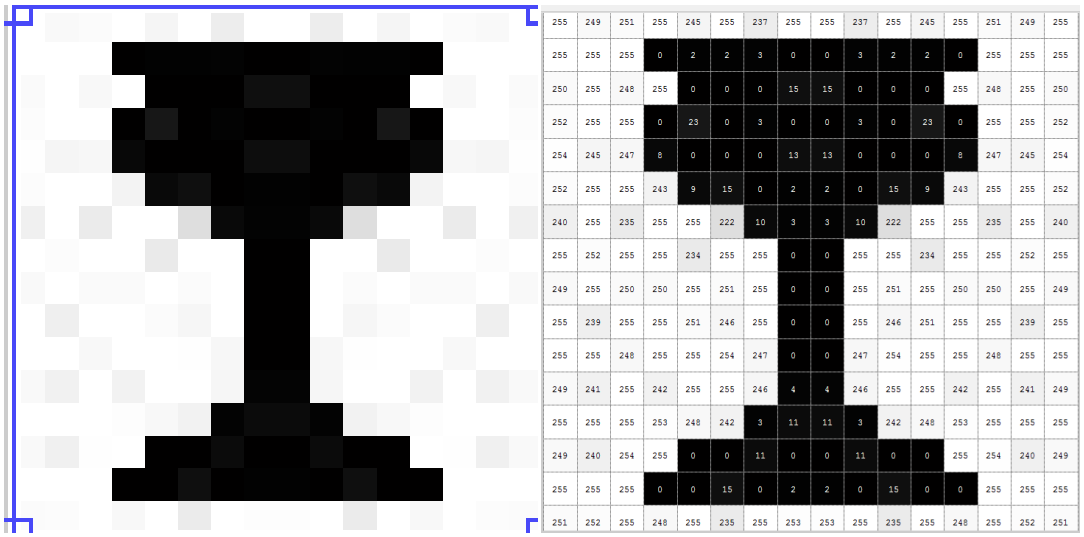
Para mejorar esto, se multiplica por $\sqrt{2}$ como un factor de corrección, lo cual resulta:

$$peri - raiz = 24 \cdot \sqrt{2} = 33.94\ unidades\ lineales$$

El error ahora es de:

$$error = \frac{33.94 - 31.41}{31.41} = 0.08054 = 8.054\%$$

Se propone ahora una copa que se muestra en la figura 5.21.



(a) (b)

Figura 5.21. (a)Copa en 2D para calcular el área y el perímetro. (b)Valores de cada pixel en escala de grises obtenidos en Matlab con el comando Imtool.

El valor más alto para un pixel negro es 18 (0x12) y el valor más bajo para un pixel blanco es 229 (0xE5). Se introducen los valores de los 256 pixeles en una matriz con el nombre de “imagen” como se aprecia en la figura 5.22.

```
unsigned char imagen[16][16] = {

        {255,249,251,255,245,255,237,255,255,237,255,245,255,251,249,255},
        {255,255,255,  0,  2,  2,  3,  0,  0,  3,  2,  2,  0,255,255,255},
        {250,255,248,255,  0,  0,  0, 15, 15,  0,  0,  0,255,248,255,250},
        {252,255,255,  0, 23,  0,  3,  0,  0,  3,  0, 23,  0,255,255,252},
        {254,245,247,  8,  0,  0,  0, 13, 13,  0,  0,  0,  8,247,245,254},
        {252,255,255,243,  9, 15,  0,  2,  2,  0, 15,  9,243,255,255,252},
        {240,255,235,255,255,222, 10,  3,  3, 10,222,255,255,235,255,240},
        {255,252,255,255,234,255,255,  0,  0,255,255,234,255,255,252,255},
        {249,255,250,250,255,251,255,  0,  0,255,251,255,250,250,255,249},
        {255,239,255,255,251,246,255,  0,  0,255,246,251,255,255,239,255},
        {255,255,248,255,255,254,247,  0,  0,247,254,255,255,248,255,255},
        {249,241,255,242,255,255,246,  4,  4,246,255,255,242,255,241,249},
        {255,255,255,253,248,242,  3, 11, 11,  3,242,248,253,255,255,255},
        {249,240,254,255,  0,  0, 11,  0,  0, 11,  0,  0,255,254,240,249},
        {255,255,255,  0,  0, 15,  0,  2,  2,  0, 15,  0,  0,255,255,255},
        {251,252,255,248,255,235,255,253,253,255,235,255,248,255,252,251},};
```

Figura 5.22. Matriz de valores de cada uno de los 256 pixeles de la imagen que forman la copa.

Los siguientes son los resultados de la corrida, figura 5.23:

$$Área\ de\ la\ copa\ =\ 82\ pixeles$$

$$Perimetro\ de\ la\ copa\ =\ 48\ pixeles$$

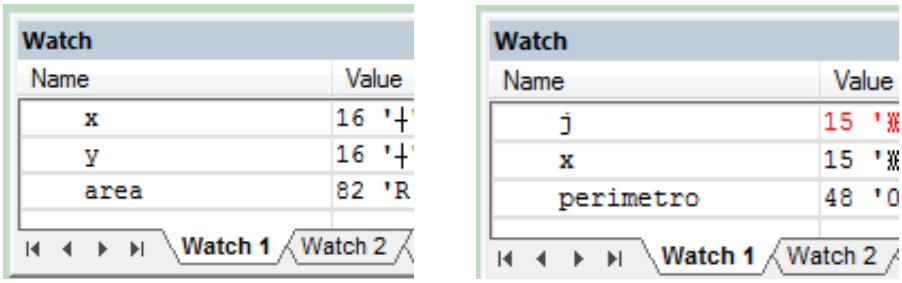


Figura 5.23. Resultado después de correr el algoritmo para calcular el área y perímetro.

En todos los casos se cotejaron los resultados de los algoritmos, contabilizándolos con los patrones de las figuras propuestas.

Para calcular el perímetro en la imagen de la copa se utilizó un intervalo diferente a -200, porque el resultado del perímetro calculado de 44 pixeles es un error, porcentualmente del 8.3%. La causa fue al evaluar algunos pixeles que el resultado es mayor a -200 como es el caso del pixel ubicado en el renglón 5, columna 10, Figura 5.24.

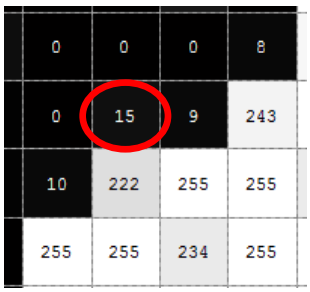
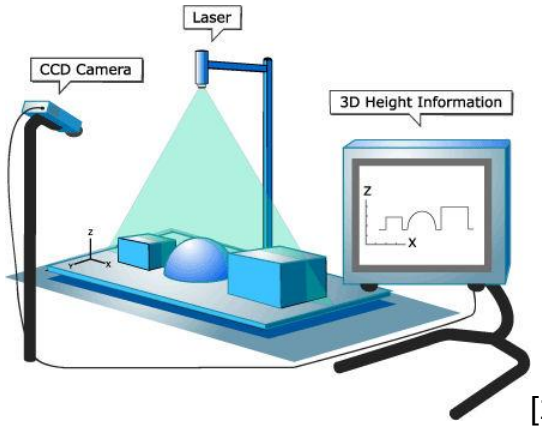


Figura 5.24 Pixel con un valor de Sobel de -171.

Al evaluar el operador de Laplace sobre el pixel circunscrito en rojo (figura 5.24), resulta:

$$Laplace = (4 * 15) - 9 - 0 - 0 - 222 = -171$$

El algoritmo ya no lo tomaba como parte del perímetro, razón por la cual se cambió el intervalo (umbral) de -200 a -160. Aún así, el algoritmo para el perímetro se ejecutó bastante bien en todos los casos.



[16]

Capítulo 6. Conclusiones y recomendaciones

En los capítulos 3 y 4 se plantearon los conceptos básicos, procesamiento y dispositivos para capturar una imagen, y se desarrollaron los algoritmos para alcanzar el objetivo del trabajo de la tesis.

La programación de ambos algoritmos está en lenguaje C y se ejecutan en el microcontrolador atmega32. La metodología seguida fue obtener el histograma de una imagen en escala de grises y escoger un valor de umbral correcto para segmentar los pixeles que forman parte de la figura de los que no son. Después, el algoritmo del área contabiliza los pixeles que cubre cada figura y el algoritmo del perímetro, por el método de Laplace, registra los pixeles que son parte del contorno. El computo del área y perímetro cambia al girar las figuras del: rectángulo, triangulo y copa. Los resultados disminuirán al aumentar la resolución de la imagen y estarán cercanos a los reales, ya que el área de la figura será distribuida entre más pixeles. El objetivo de utilizar estos algoritmos en el procesamiento de imágenes es funcional y operable en un microcontrolador atmega32. El inconveniente, es la dependencia de usar una computadora para transformar la imagen JPG a escala de grises o binaria, y conseguir los valores de la matriz para transferirlos en un arreglo, esto lo hace poco práctico.

Para trabajos futuros será Integrar a la cámara el algoritmo para decodificar archivos JPG, por ejemplo: en un DSP, para obtener los valores de los pixeles de la imagen.

Finalmente, continuación se muestra en la tabla 7.1 los resultados obtenidos.

Imagen de 16x16 (256) pixeles en escala de grises (0 a 255)					
Figura	Área	Perímetro	Tiempo ms	Intervalo p/ área	Intervalo p/ perímetro
Rectángulo	40 pixeles	24 pixeles	4.02	$pixel \leq 40$	$delta(Laplace) \leq -200$
Triángulo	62 pixeles	28 pixeles	4.02	$pixel \leq 40$	$delta(Laplace) \leq -200$
Circulo	68 pixeles	24 pixeles	4.02	$pixel \leq 40$	$delta(Laplace) \leq -200$
Copa	82 pixeles	48 pixeles	4.02	$pixel \leq 40$	$delta(Laplace) \leq -160$

Tabla 7.1. Resultado de cada corrida de algoritmos, tanto para el área como para el perímetro.

Bibliografía

[1] Fu, K.S., Gonzalez, R.C. and Lee, C.S.G. [1987]. *Robotics: Control, sensing, vision, and intelligence.* McGraw-Hill. ISBN:0-07-022625-3

[2] Hauβecker, Horts and J, Bernd. [2000]. *Computer vision and applications.* Academic Press. ISBN: 0-12-379777-2

[3] Gonzalez, R.C., Woods, R.E. and Eddins, S.L. [2004]. *Digital image processing using Matlab.* Pearson Prentice Hall. New Jersey. ISBN: 0-13-008519-7

[4] Bow, Sing-Tze. [1992]. *Pattern Recognition and image preprocessing.* Marcel Dekker. ISBN: 0-8247-8583-5

[5] Angulo, J. Ma. e Iñigo Madrigal, R. [1986]. *Visión artificial por computador.* Paraninfo. ISBN: 84-283-1446-2

[6] Gonzalez, R.C. and Wintz, Paul. [1983]. *Digital image processing.* 6th printing. Addison-Wesley. ISBN: 0-201-03045-4

[7] Pennebaker, William B. [1993]. *JPEG still image data comprenssion standard.* Van Nostrand Reinold. ISBN: 0-442-01272-1

[8] Young, Tzay Y. Fu, King-Sun.[1986]. *Handbook of Pattern Recognition and Image Processing.* Ed. Young, Tzay Y. Fu, King-Sun. ISBN: 0-12-774560-2

[9] Capson, D. and Kitai, R. *Silhouette description and recognition for machine vision.*

[10] Siciliano, Antonio. [2008]. *Matlab data analysis and visualization.* World Scientfic. ISBN: 978-981-283-554-3

[11] Mitra, Sanjit K. [2006]. *Digital Signal Processing. A computer-based approach*. 3rd ed. McGraw-Hill. ISBN 0-07-286546-6

[12] Gonzalez, R.C. and Woods, R.E. [2008]. *Digital image processing.* 3rd ed. Pearson Prentice Hall. New Jersey. ISBN: 0-13-140141-6

[13] Horn, Berthold K.P. [1991]. *Robot vision.* 7th Printing. MIT Press Mc Graw-Hill

◄Fuentes electrónicas►

[14] http://www.fairchildimaging.com/products/fpa/ccd/area/ (consulta: 23/6/2011)

[15] http://www.barnardmicrosystems.com/L4E_linescan.htm (consulta: 17/6/2011)

◄Imágenes►

[16] 3D Vision: Un sistema basado en una PC que utiliza 2 o más cámaras ó 1 cámara con una línea laser. Usado en un sistema de orientación robótica y aplicada en mediciones que no son posibles con una sola cámara fija. Imagen tomada del sitio: ENGATECH We make machine vision work …:

http://www.engatechvision.com/tech.html (Consulta: 01/11/2012)