FACULTAD DE INGENIERÍA ELÉCTRICA

Tesis en opción al grado académico de Maestra en Ciencias

Reconocimiento de Objetos Poligonales Regulares parcialmente ocultos en imágenes digitales.

Autora: ***Daily Milanés Hermosilla***

Tutor: ***Dr. Ing. Israel Mazaira Morales***

Año 2011

"Año del aniversario 53 de la Revolución."

PENSAMIENTOS

Como el suelo, que, por más rico que sea, no puede dar frutos si no se cultiva, la mente sin cultivo tampoco puede producir.

Añade el hombre conocimientos a conocimientos: nunca el saber es bastante. Si tanto es uno más hombre cuanto más sabe, el más noble empleo será el aprender.

La sabiduría es la capacidad para procesar las señales externas que se reciben en la vida y convertirlas en las brújulas del espíritu.

DEDICATORIA

A mi bebé que amo tanto y está en camino a la vida…

AGRADECIMIENTOS

A mi tutor Mazaira porque sin él no hubiera sido posible la realización de este trabajo.

A mi maravilloso esposo por su entrega y apoyo diario.

A mi madre y mi padre por estar siempre presente

A mi familia por el apoyo brindado.

A los profesores María Elena y Chang por su experiencia.

A todos mis compañeros de trabajo en especial a Saddid, Ania e Irina.

SÍNTESIS

En el presente trabajo se analizan las técnicas más difundidas en el análisis de imágenes digitales para la detección de objetos. Se mencionan algunas de las tendencias actuales de los métodos para la detección de objetos parcialmente ocultos o deformados en imágenes, y se plantean las necesidades de realizar esta investigación la cual tiene como fin el reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales en escala de grises. Se analizan las propiedades que caracterizan a estos tipos de objetos, que permiten identificarlos y diferenciarlos de otros. Se propone un algoritmo basado en información sólo del contorno del objeto para el reconocimiento del mismo y se justifica el uso de los métodos utilizados en el diseño del mismo. El algoritmo contribuirá a una futura implementación de integración robot visión para la manipulación de objetos. Finalmente se corroboran los resultados mediante simulaciones utilizando imágenes creadas por la autora del trabajo utilizando para ello paquetes de programas profesionales. Por último se dan valoraciones de expertos.

ÍNDICE

INTRODUCCIÓN

Durante los años de vida del robot industrial, éste se ha beneficiado de innumerables innovaciones tecnológicas que le han proporcionado mejores prestaciones y que le han permitido abordar con éxito aplicaciones cada vez más variadas y complejas.

Entre estas mejoras se tienen las relativas a nuevos materiales presentes en motores más ligeros y potentes o en la estructura del robot, electrónica más rápida y más integrada, software más poderosos y amigables, y sensores más sofisticados y eficientes.

La mayoría de los robots industriales que hay instalados actualmente en los procesos productivos, están prácticamente incomunicados con el entorno que les rodea. La necesidad de tener programadas las acciones a efectuar, restringe el ambiente de trabajo a unas condiciones estrictas, al igual que a las pinzas o los materiales que se han de manipular.

En la planificación es donde se concentra la mayor parte de la actividad "inteligente" del robot. Sin embargo, un robot no podría hacer nada si no pudiera medir de alguna forma lo que le interesa del medio en el que se desarrolla su actividad.

Para poder realizar esta primera e importante fase, los robots disponen de sensores, los cuales cumplen la misma función en los robots que los órganos sensoriales en la mayoría de los seres vivos. Los sensores les permitan saber donde están, cómo es el lugar en el que están, a qué condiciones físicas se enfrentan, dónde están los objetos con los que deben interactuar, sus parámetros internos, tales como la posición, la velocidad, etc. Sin ellos los robots no podrían localizar objetos para poder cogerlos, evitar obstáculos para no chocarse, comprobar el correcto funcionamiento de una actividad, etc.

Los sensores de robot se clasifican funcionalmente en dos tipos: sensores propioceptivos, que informan sobre el estado interno del robot, midiendo variables de posición y velocidad articulares con fines de control, y sensores exteroceptivos, que informan sobre el entorno del robot y sus variaciones. Los sensores exteroceptivos pueden a su vez clasificarse en dos grandes grupos: sensores de contacto, que responden al contacto físico: tacto, deslizamiento, fuerza de interacción robot-entorno, y sensores de no contacto, basados en la respuesta de un detector a las variaciones en la radiación electromagnética o acústica, entre

los cuales se pueden mencionar los sensores que miden la distancia, el posicionamiento relativo pinza-objeto, el guiado de robots y la identificación y manejo de objetos.

Aunque los sensores de proximidad, contacto y fuerza juegan un papel significativo en la mejora del funcionamiento del robot, se reconoce que la **visión** es la capacidad sensorial más potente del robot.

Al igual que sucede en el ser humano, la capacidad de “ver” dota al robot de un sofisticado mecanismo de percepción, que le permite responder a su entorno de una forma inteligente y flexible. El uso de sistemas sofisticados de percepción, incluyendo la visión, están motivados por la constante necesidad de abaratar los costos, de incrementar flexibilidad, aumentar los campos de aplicación de los sistemas robotizados, así como el de facilitar la comunicación hombre-máquina.

La visión del robot se puede definir como el proceso de extraer, caracterizar e interpretar información de imágenes de un mundo tridimensional.

La integración robot-visión no sólo es un problema de cómo utilizar la información sensorial sino qué información visual debe ser extraída de las imágenes. El elemento básico que permite responder a estas cuestiones es la aplicación y siempre aparece la necesidad de limitar el problema a la aplicación de interés, para poder seleccionar los algoritmos más adecuados o bien para adecuar los algoritmos disponibles, o mejor aún, diseñar métodos novedosos.

Dos características fundamentales destacan los sistemas integrados robot-visión: por un lado la capacidad del reconocimiento y localización de partes en el espacio por medio de una de las técnicas más avanzadas de procesamiento y descripción de información exteroceptiva, y por otro la capacidad de empleo de esta información en la realización automática de tareas de interacción inteligente robot-escena.

El desarrollo de este trabajo se centra en la primera característica, y dentro de ella en el estudio de los métodos existentes para el reconocimiento y localización de objetos en imágenes digitales.

De forma general se recogen métodos en la literatura consultada para la detección de puntos, líneas, contornos de objetos, así como para la extracción de características de los mismos como área, perímetro, etc. Sin embargo estos métodos están basados en la suposición del conocimiento de información total del objeto.

Existen trabajos recientes basados en el conocimiento de información parcial de los objetos, los cuales de forma general reconocen un objeto parcialmente oculto por solapamiento o deterioro mediante la comparación de las características de estos objetos con la de otros almacenados en bases de datos. Esta comparación se realiza de manera automática y los algoritmos desarrollados proporcionan buenos resultados en las aplicaciones en las cuales se han implementado.

Sin embargo es interesante la idea de analizar la forma en la que los seres humanos reconocen objetos aún cuando los mismos no sean visibles totalmente. La razón es precisamente por el aprendizaje de características de cada objeto, las cuales les permiten distinguir unos de otros sin necesidad de compararlos.

El marco de referencia de este trabajo se centra en la implementación de un algoritmo de reconocimiento de objetos parcialmente ocultos de poca complejidad estructural en imágenes, con vistas a una posible integración Visión-Robot en trabajos futuros. Los objetos considerados son figuras geométricas planas regulares, de color uniforme y muy contrastados con el fondo. Estas características permiten el trabajo con imágenes en niveles de gris, que mediante atributos obtenidos a partir del contorno de los objetos será posible el reconocimiento o descripción de los mismos.

El **problema** consiste entonces en la carencia de métodos de análisis y reconocimiento de objetos poligonales regulares a partir de información parcial o incompleta de los mismos.

Para ello de define como **objeto de la investigación** los sistemas de visión.

Como **objetivo de la investigación** el desarrollo de métodos de análisis y reconocimiento de objetos poligonales regulares a partir de información parcial de los mismos.

Como **campo de acción** métodos de reconocimiento de objetos poligonales regulares parcialmente ocultos en sistemas de visión.

Se plantea la siguiente **hipótesis**: si se desarrollan métodos de análisis y reconocimiento de objetos poligonales regulares a partir de información parcial o incompleta de los mismos, se resuelven problemas de reconocimiento de objetos poligonales regulares parcialmente ocultos en sistemas de visión.

Para dar cumplimiento al objetivo de la investigación se trazaron las siguientes tareas:

1. Caracterizar desde el punto de vista gnoseológico, histórico y actual los sistemas de visión y los métodos de reconocimiento de objetos poligonales regulares parcialmente ocultos en sistemas de visión.
2. Desarrollar un método de seguimiento de contorno para la obtención del código de cadena del contorno de objetos.
3. Analizar la representación de rectas digitales en imágenes digitales.
4. Desarrollar un método de reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales en escala de grises.
5. Corroborar el algoritmo desarrollado para el reconocimiento de objetos poligonales regulares parcialmente oculto mediante simulación, así como la obtención de valoraciones de expertos.

Técnicas y métodos empleados en la investigación:

1. Técnicas empíricas.
2. Análisis de fuentes documentales.
3. Observación.
4. Entrevistas a los expertos.
5. Método histórico – lógico.
6. Método de análisis – síntesis.

Aporte de la investigación: Desarrollo e implementación a escala de simulación de un método de análisis y reconocimiento de objetos poligonales regulares a partir de información parcial o incompleta de los mismos en sistemas de visión.

La tesis se encuentra organizada de la siguiente forma: una introducción general en la que se exponen las principales motivaciones que llevaron a la realización de esta investigación y en la cual se encuentra además, la fundamentación del diseño metodológico de la misma. Dos capítulos que constan de introducciones y conclusiones parciales para una mejor comprensión de los objetivos de los mismos, estos a su vez, se encuentran organizados por epígrafes. Finalmente se dan las conclusiones generales, recomendaciones y bibliografía.

CAPÍTULO I

CAPÍTULO I. CARACTERIZACIÓN DE LOS SISTEMAS DE VISIÓN Y LOS MÉTODOS DE ANÁLISIS Y RECONOCIMIENTO DE OBJETOS POLIGONALES REGULARES PARCIALMENTE OCULTOS EN SISTEMAS DE VISIÓN.

INTRODUCCIÓN.

En este capítulo se explica de forma general en que consiste la visión por computadora, así como sus aplicaciones más frecuentes. Se mencionan las técnicas más difundidas para la segmentación de imágenes digitales en escala de grises, en la detección de puntos, líneas, bordes, técnicas para el enlazado de bordes y detección de fronteras. Además se analizan los esquemas de representación para analizar la forma geométrica de los objetos contenidos en las imágenes, así como algunos descriptores de contornos y regiones. Se da una breve definición de polígonos, específicamente los regulares, y se analizan sus características y particularidades que luego serán utilizadas para la detección de los mismos en imágenes digitales en escala de grises. Por último, se da una breve introducción al tema de reconocimiento de objetos parcialmente ocultos, así como algunos comentarios de trabajos recientes relacionados con esta temática. Se analizan las posibles líneas de trabajo para el reconocimiento de objetos poligonales regulares parcialmente ocultos presentes en imágenes digitales en escala de grises.

1.1 CARACTERIZACIÓN GNOSEOLÓGICA DE LOS SISTEMAS DE VISIÓN Y LOS MÉTODOS DE ANÁLISIS Y RECONOCIMIENTO DE OBJETOS POLIGONALES REGULARES.

1.1.1 LOS SISTEMAS DE VISIÓN POR COMPUTADORA.

La visión por computadora es una rama de la inteligencia artificial que tiene por objetivo modelar matemáticamente los procesos de percepción visual en los seres vivos y generar programas que permitan simular estas capacidades visuales por computadora.

La visión por computadora (VC) puede definirse como el proceso de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional.

Estos procesos pueden dividirse en las siguientes etapas principales (Esqueda, 2002 y González, 1987):

- Formación y captación de imágenes digitales.
- Preprocesamiento.
- Segmentación.
- Descripción.
- Reconocimiento e interpretación.

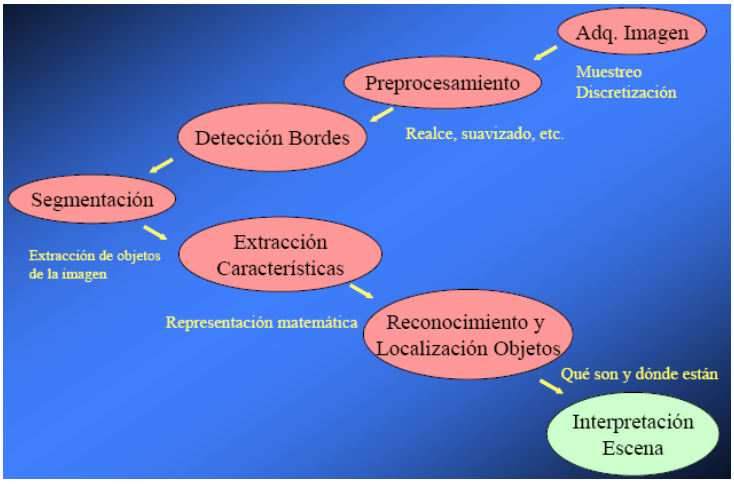


Figura 1.1. Etapas de un sistema de visión por computadora.

La formación es la transformación, mediante procedimientos ópticos, del mundo tridimensional en su imagen en el plano. La captación es el proceso a través del cual se obtiene una señal eléctrica conteniendo la información visual de la imagen y su conversión A/D que permite tenerla disponible en la memoria de la PC.

El preprocesamiento incluye técnicas como reducción de ruidos y realce de detalles.

La segmentación es el proceso que divide una imagen en regiones significativas que sean de interés para cada aplicación en particular.

Mediante los procesos de descripción se extraen características convenientes para diferenciar cada tipo particular de región u objeto.

El reconocimiento es el proceso que clasifica estos objetos y la interpretación le otorga un significado simbólico a un conjunto de objetos reconocidos y a sus relaciones geométricas y topológicas.

En el caso particular de la robótica la visión se utiliza como sensor para cerrar el lazo de control en tareas de interacción inteligente de un robot con su entorno de trabajo y tendría el proceso una etapa adicional denominada comunicación Robot -Visión.

Por tal motivo, la misión de un sistema de visión para robot consiste en proporcionar, a partir de la información visual contenida en una imagen (o serie de imágenes) del entorno significativo del robot, una descripción de dicho entorno, entendiéndose por descripción al vector de atributos que dentro de un espacio de caracterización permite la diferenciación entre clases de objetos para su reconocimiento de manera automática y relativamente inteligente.

APLICACIONES DE LA VISIÓN POR COMPUTADORA.

Cada día es mayor el número de aplicaciones de la visión artificial. En el marco de este trabajo sólo se dará una pequeña pincelada sobre las múltiples aplicaciones en las que la visión artificial se ha aplicado hasta el momento. Obviamente los ejemplos que se presentan son aplicables a cualquier otro proceso y campo industrial o científico diferente. (Gonzalo, 2006. Consulta en [13]).

No obstante cabe diferenciar entre las aplicaciones donde la visión artificial constituye una herramienta por sí sola y aquellas otras en las que es parte de un sistema más general. El primer caso engloba todas aquellas aplicaciones en las que el único sensor presente es el de visión. El segundo caso se refiere a aquellos sistemas multisensoriales tales como los equipos de navegación en robótica.

- **NAVEGACIÓN EN ROBÓTICA**

En este caso, la visión es un elemento de un sistema multisensorial. La información procedente de la visión es validada, comparada y finalmente integrada con el resto de la información proporcionada por otros tipos de sensores. El resultado es la reconstrucción de la escena 3-D, que permite la navegación autónoma del sistema robótico.

No es exclusivo su uso en robótica, sino que podría utilizarse en otras aplicaciones tales como guiado automático de máquinas, la detección y estimación del movimiento de vehículos, etc.

- **BIOLOGÍA, GEOLOGÍA Y METEOROLOGÍA**

En el campo de la biología se podría distinguir entre aplicaciones microscópicas y macroscópicas. En una imagen microscópica se pueden encontrar abundantes organismos, que mediante técnicas de

segmentación orientadas a regiones, podrían ser aislados para su identificación mediante las correspondientes propiedades (tamaño, excentricidad, color, etc.) o para contar el número de microorganismos o células presentes en una imagen. En las imágenes macroscópicas pueden utilizarse las regiones para la identificación de determinados tipos de texturas en vegetales o características de diferentes áreas naturales por su color o el crecimiento de ciertas especies por diferencia de imágenes.

En geología se pueden también detectar movimientos de terrenos captando dos imágenes en diferentes momentos de tiempo para observar la variación mediante una diferencia de imágenes (bajo similares condiciones de iluminación).

En Meteorología se podrían utilizar las técnicas de detección y predicción del movimiento para observar, por ejemplo, la evolución de ciertas masas nubosas, u otros fenómenos meteorológicos, a través de imágenes recibidas vía satélite.

- **MEDICINA**

La comunidad médica utiliza muchas aplicaciones en las que aparece el procesamiento de imágenes, la mayoría de ellas orientadas hacia el diagnóstico de dolencias o enfermedades, entre las que se incluyen radiografías, resonancias magnéticas, tomografías, etc.

Algunas de las aplicaciones en la medicina se han encontrado en:

- El zoom permite ampliar detalles de la imagen que en una primera vista aparecen confusos.
- Obtención del entramado de vasos capilares, o nervios en un determinado tejido mediante la extracción de bordes (González, 1987).
- Detección de lesiones vasculares a partir de angiogramas renales (extensible a otros órganos) en (Jaulent y col, 1997 y Jendrysik y col, 1997).
- Diferenciar tejidos sanos de tejidos cancerígenos o infectados por el color en (Umbaugh, 1998).
- Detección de cánceres de piel mediante técnicas de color y extracción de bordes en (Xu y col., 1999).
- Medida del grosor de venas y arterias en (Wick y col., 1993; Chen y col., 1987 y Schmid-Choenbein y col., 1977).
- Detección de puntos de interés en una radiografía como precursores de la presencia de un tumor en (Low ,1991) o como puntos de referencia en ciertos órganos como el cerebro en (Rohr, 1999).

- Identificación de un nódulo sospechoso en una mamografía por diferencia de contraste y textura en (Trucco y Verri, 1998).
- En neurología para determinar el estado de enfermedad y el grado de deformación de la materia gris del cerebro en enfermos epilépticos mediante el uso de contornos deformables en (Schnabel y Arridge, 1999 y Gee, 1999).
- Detección de costillas en radiografías mediante la transformada de Hough en (Wechsler y Sklansky, 1977).
- Detección de microcalcificaciones en mamografías mediante redes neuronales en (Tsujii y col., 1999).
- Reconstrucción de arterias coronarias utilizando imágenes de angiogramas en (Windyga y col., 1998).

➢ INSPECCIÓN Y CONTROL DE CALIDAD

La inspección de un objeto manufacturado puede tomar muchas formas, que podría involucrar las siguientes tareas:

a) Verificar la presencia de cada característica esperada.

b) Verificar las dimensiones de esas características (por ejemplo radio y longitud de un cilindro).

c) Verificar las interrelaciones entre características (por ejemplo distancias entre centros de gravedad y ángulos entre planos).

La inspección en el sentido más amplio se refiere a la verificación de si un objeto cumple con determinados criterios. Esto implica comparar el objeto con algún objeto modelo que describe las características relevantes del objeto. Para muchos tipos de datos existen tolerancias definidas dentro de las cuales las medidas realizadas pueden considerarse como aceptables.

Una de las finalidades de los controles de calidad consiste en detener la producción de algún producto si el sistema de producción comienza a generar productos que no cumplen con las normas estándares generales, también la verificación de la calidad final del producto para valorar el precio de venta en el mercado, etc.

- **TELEDETECCIÓN**

Es la ciencia que se ocupa, como su nombre indica, de la detección a distancia basada en imágenes de satélite y aéreas fundamentalmente. Esta detección es muy diversa, abarca desde la pura interpretación hasta un tratamiento más avanzado.

Desde el punto de vista de la mera interpretación, estas imágenes son tratables mediante técnicas de filtrado, realzado, extracción de bordes, etc., es decir métodos aplicables a cualquier tipo de imágenes.

Además suele ser útil la detección de cambios en una zona en diferentes instantes de tiempo. Esta aplicación resulta de gran utilidad para la detección de zonas deforestadas, incendios, inundaciones, variación de la edificación, etc.

1.1.2. MÉTODOS DE ANÁLISIS Y RECONOCIMIENTO DE OBJETOS POLIGONALES REGULARES.

Los polígonos son figuras planas, limitadas por una línea poligonal cerrada. Un polígono queda determinado por sus lados, que son los segmentos de la poligonal, por sus vértices, que son los formados por la intersección de dos lados consecutivos y por sus ángulos, que son los que se forman entre dos lados consecutivos. Los polígonos tienen tres o más lados. (Consultas en [1], [2], [3], [4] y [5]).

La palabra **"Polígono"** significa **"varios lados"**; es por esto que el nombre particular de cada polígono está determinado por el número de lados, que es igual al número de ángulos que quedan determinados por dos lados consecutivos.

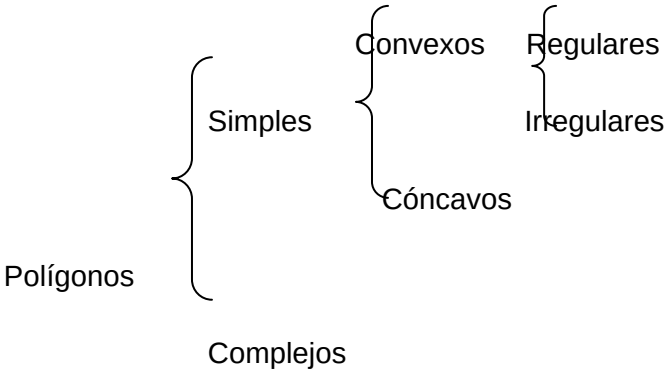
Otro aspecto importante de los polígonos, son sus diagonales. Las diagonales de un polígono, son las líneas que unen dos vértices no consecutivos de éste, por lo tanto la cantidad de diagonales que se le pueden trazar a un polígono, variará según el número de vértices del mismo.

Dependiendo del número de lados que tenga el polígono, recibirá un nombre diferente. En la tabla 1.1 se dan a conocer los nombres de aquellos polígonos que tienen hasta 10 lados.

Número de lados	Nombre del polígono
3	**Triángulo**
4	**Cuadrilátero**
5	**Pentágono**
6	**Hexágono**
7	**Heptágono**
8	**Octágono**
9	**Nonágono**
10	**Decágono**

Tabla 1.1. Nombre de polígonos según el número de lados.

Los polígonos se clasifican según la forma de su contorno en:



- **Simples**: sus aristas no consecutivas no se intersecan.
- **Complejos**: sus aristas no consecutivas se intersecan.
- **Convexos:** al atravesarlo una recta lo corta en un máximo de dos puntos.
- **Cóncavos:** al atravesarlo una recta puede cortarlo en más de dos puntos.
- **Regulares:** tiene sus ángulos y sus lados iguales.
- **Irregulares:** tiene sus ángulos y lados desiguales.
- **Equiláteros:** tiene todos sus lados iguales.
- **Equiángulos:** tiene todos sus ángulos iguales.

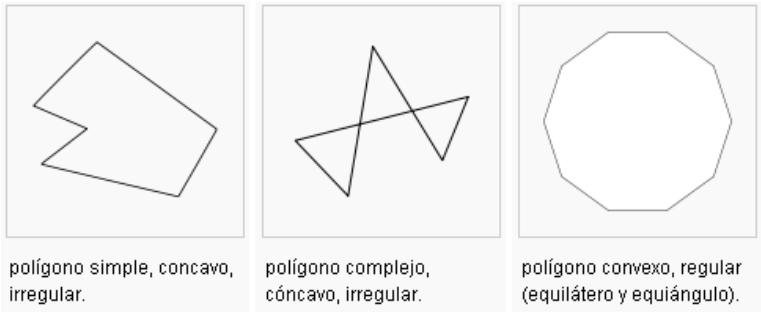


Figura 1.2. Clasificación de polígonos según la forma de su contorno.

Polígonos regulares.

Un polígono regular es un polígono en el que todos los lados tienen la misma longitud y todos los ángulos interiores son de la misma medida.

Se emplea la figura de un hexágono para representar un polígono regular genérico y analizar las características de los polígonos regulares.

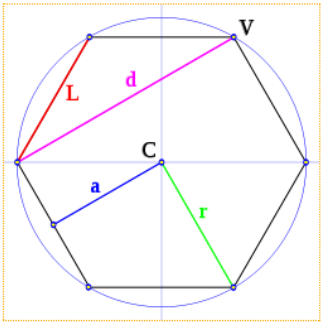


Figura 1.3. Características de los polígonos regulares (hexágono).

Una característica de los polígonos regulares, es que se pueden trazar inscritos en una circunferencia que tocará cada uno de los vértices del polígono. A medida que crece el número de lados de un polígono regular, su apariencia se asemeja cada vez más a la de una circunferencia.

En un polígono regular se distingue:

- Lado, **L**: cada uno de los segmentos que forman el polígono.
- Vértice, **V**: el punto de unión de dos lados consecutivos.
- Centro, **C**: el punto central equidistante de todos los vértices.
- Radio, **r**: el segmento que une el centro del polígono con uno de sus vértices.

- Apotema, **a**: segmento perpendicular a un lado, hasta el centro del polígono.
- Diagonal, **d**: segmento que une dos vértices no contiguos.
- Perímetro, **P**: es la suma de la medida de su contorno.

Dadas las características de los polígonos regulares, se diferencian algunas propiedades que se dan siempre, y que son de gran utilidad para determinar sus propiedades y dimensiones geométricas.

- Los polígonos regulares son **equiláteros**; todos sus lados tienen la misma longitud.
- Todos los ángulos interiores de un polígono regular tienen la misma medida, es decir, son congruentes.
- El **centro** de un polígono regular es un punto equidistante de todos los vértices del polígono.
- Los polígonos se pueden dividir en triángulos cuyos lados son el lado del polígono y los dos segmentos que unen el centro y los vértices (radios).
- El **apotema** es el segmento que une el centro y la mitad de cada lado del polígono.
- El **radio** es el segmento que une el centro y cada vértice.
- El Ángulo interior, β, de un polígono regular mide:

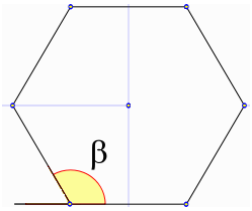


Figura 1.4. Ángulo interior en polígonos regulares.

$\beta = 180.\frac{(n-2)}{n}$ en grados n: número de lados del polígono regular.

$\beta = \pi.\frac{(n-2)}{n}$ en radianes

- La suma de los ángulos interiores, $\sum \beta$, de un polígono regular es de:

$\sum \beta = 180.(n-2)$ en grados

$\sum \beta = \pi.(n-2)$ en radianes

Por ejemplo el ángulo interior de un octágono (8 lados) es: (**8**-2) × 180° / **8** = 135°.

El de un cuadrado es (**4**-2) × 180° / **4** = 90°.

A continuación se muestra la clasificación de algunos polígonos regulares (hasta 10 lados), atendiendo al número de lados.

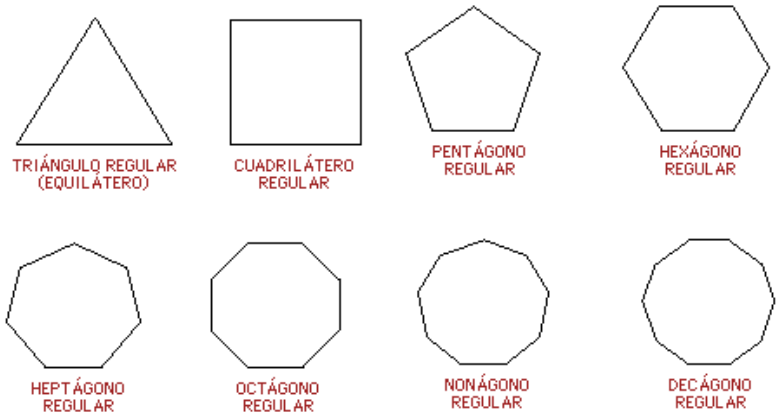


Figura 1.5. Clasificación de polígonos regulares según el número de lados.

Para el reconocimiento del tipo de polígono regular basta tener información de un ángulo interno pues todos los ángulos internos son iguales y para cada tipo de polígono regular el ángulo interior es diferente (tabla 1.2).

Nombre del polígono	Valor de los ángulos interiores
Triángulo	**60º**
Cuadrado	**90º**
Pentágono	**108º**
Hexágono	**120º**
Heptágono	**128º**
Octágono	**135º**

Tabla 1.2. Ángulos interiores de algunos polígonos regulares.

Para conocer el tamaño de un polígono regular o para su reconstrucción, es necesario conocer además la longitud de un lado, pues todos los lados tienen la misma longitud.

En caso que no se tenga información del ángulo interno se puede reconocer el polígono regular teniendo información de un lado completo y parte del lado contiguo, de forma tal que se pueda determinar el ángulo comprendido entre estos dos lados.

1.2 CARACTERIZACIÓN HISTÓRICA DE LOS SISTEMAS DE VISIÓN Y LOS MÉTODOS DE ANÁLISIS Y RECONOCIMIETNO DE OBJETOS POLIGONALES REGULARES.

1.2.1 TÉCNICAS PARA LA SEGMENTACIÓN DE LA IMAGEN.

El primer paso en cualquier proceso de análisis de imágenes es la segmentación. Ésta permite dividir la imagen en las partes u objetos que la forman. El nivel al que se realiza esta subdivisión depende de la aplicación en particular, es decir, la segmentación terminará cuando se hayan detectado todos los objetos de interés para la aplicación.

En general, la segmentación automática es una de las tareas más complicadas dentro del procesado de imagen. La segmentación va a dar lugar en última instancia al éxito o fallo del proceso de análisis. En la mayor parte de los casos, una buena segmentación dará lugar a una solución correcta, por lo que, se debe poner todo el esfuerzo posible en esta etapa del procesamiento de imágenes (Esqueda, 2002; González, 1987 y Martín, 2002. Consultas en [8], [9], [10] y [11]).

Los algoritmos de segmentación se basan en dos propiedades básicas de los niveles de gris de la imagen: discontinuidad y similitud.

Segmentación basada en la detección de discontinuidades: Se intenta dividir la imagen basándose en los cambios bruscos en el nivel de gris. Las áreas de interés en esta categoría son la detección de puntos, líneas y bordes en la imagen.

Segmentación basada en similitud: Intenta construir directamente las regiones que forman la imagen. La idea principal es buscar píxeles que, siendo vecinos en la imagen, compartan valores próximos respecto a alguna propiedad de la imagen, por ejemplo su nivel de gris. Algunas técnicas utilizadas son el procesado local, el procesado global, técnicas de umbralización, el crecimiento de regiones, entre otras.

SEGMENTACIÓN BASADA EN LA DETECCIÓN DE DISCONTINUIDADES.

- DETECCIÓN DE PUNTOS

La detección de puntos aislados es inmediata. Empleando la máscara de convolución siguiente, se dice que se ha detectado un punto en la posición en la cual está centrada la máscara si: $|R| > T$.

donde: R: respuesta a la máscara de la imagen.

T: umbral en niveles de gris.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Básicamente se mide la distancia entre el píxel central y sus vecinos, puesto que un píxel será punto aislado siempre que sea suficientemente distinto de sus vecinos. Solamente se considerarán puntos asilados aquellos cuya diferencia con respecto a sus vecinos sea significativa.

- DETECCIÓN DE LÍNEAS

Para la detección de líneas de ancho un píxel se utilizan las siguientes máscaras:

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Horizontal 45° *Vertical* -45°

Si se pasa la primera de las máscaras a lo largo de la imagen, tendrá mayor respuesta para líneas orientadas horizontalmente. La segunda máscara responderá mejor a líneas orientadas a 45º, la tercera a líneas verticales y la última orientada a -45º.

Estas direcciones se pueden establecer observando que para la dirección de interés las máscaras presentan valores mayores que para otras posibles direcciones. Si se denota con R_1, R_2, R_3 y R_4 las respuestas de las cuatro máscaras para un píxel en particular, entonces si se cumple que $|R_i| > |R_j|$ con $i \neq j$, será más probable que dicho píxel esté asociado a la dirección correspondiente a la máscara *i*.

- DETECCIÓN DE BORDES

La detección de bordes es el procedimiento empleado más habitualmente para la detección de discontinuidades.

Borde: Frontera entre dos regiones con nivel de gris relativamente diferente.

La idea básica detrás de cualquier detector de bordes es el cálculo de un operador local de derivación.

En la siguiente figura se puede ver este concepto. Se observa una imagen de una banda clara sobre un fondo oscuro, el perfil a lo largo de una línea horizontal y la primera y segunda derivada de dicho perfil.

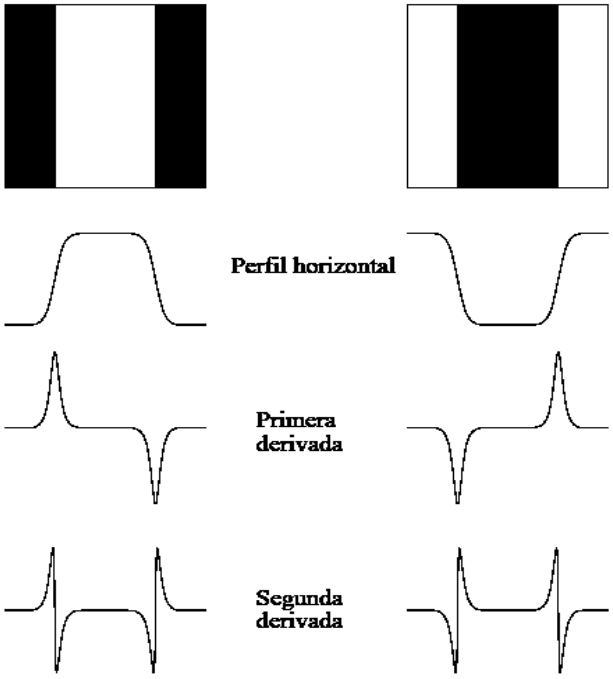


Figura 1.6. Detección de bordes basados en los operadores de derivación.

La primera derivada es positiva para cambio a nivel de gris más claro, negativa en caso contrario y cero en aquellas zonas con nivel de gris uniforme. La segunda derivada presenta valor positivo en la zona oscura de cada borde, valor negativo en la zona clara de cada borde y valor cero en las zonas de valor de gris constante y justo en la posición del borde.

El valor de la magnitud de la primera derivada nos sirve para detectar la presencia de bordes, mientras que el signo de la segunda derivada nos indica si el píxel pertenece a la zona clara o a la

zona oscura. Además la segunda derivada presenta siempre un cruce por cero en el punto medio de la transición. Esto puede ser muy útil para localizar bordes en una imagen.

Para la extensión a dos dimensiones, se define el perfil en la dirección perpendicular a la dirección del borde y la interpretación anterior seguirá siendo válida.

La primera derivada en cualquier punto de la imagen vendrá dada por la magnitud del gradiente, mientras que la segunda derivada vendrá dada por el operador Laplaciano.

Gradiente

El gradiente de una imagen $I(x,y)$ en la posición (x,y) viene dado por el vector:

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$

El vector gradiente siempre apunta en la dirección de la máxima variación de la imagen I en el punto (x,y).

La magnitud del gradiente es muy importante en la detección de bordes, denominado simplemente como gradiente de la imagen, denotado por ∇I y dado por: $\nabla I = \|\nabla I\| = \sqrt{I_x{}^2 + I_y{}^2}$, esta cantidad representa la variación de la imagen por unidad de distancia en la dirección del vector ∇I.

En general, el gradiente se suele aproximar mediante la expresión:

$\nabla I \approx |I_x| + |I_y|$ que es mucho más simple de implementar en la práctica.

La dirección del vector gradiente también es una cantidad importante.

Sea $\alpha(x,y)$ el ángulo del vector ∇I en el punto (x,y), entonces se tiene que: $\alpha(x,y) = \tan^{-1}\frac{I_y(x,y)}{I_x(x,y)}$ donde los ángulos de miden con respecto al eje de las abscisas.

El cálculo del gradiente se basa en obtener las derivadas parciales para cada píxel. Las derivadas se pueden implementar mediante las máscaras de Roberts, Prewitt, Sobel y Freichen, las cuales se muestran a continuación:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Roberts

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Pr*ewitt*

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel

$$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$$

Frei.–Chen

Los operadores de Sobel y Frei-Chen tienen la ventaja de que proporcionan un suavizado además del efecto de derivación pues la derivada acentúa el ruido.

El requisito básico de un operador de derivación es que la suma de los coeficientes de la máscara de una zona uniforme de la imagen sea cero.

Es importante destacar que la imagen gradiente será la suma de las respuestas a cada una de las máscaras, una máscara determina la derivada parcial respecto a *x* y la otra respecto a *y*.

Laplaciano

El Laplaciano de una imagen $I(x, y)$ es una derivada de orden dos definida por:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

En general se suele tomar el valor negativo del Laplaciano. Al igual que en el caso del gradiente se puede implementar mediante el uso de las máscaras de convolución.

Puesto que el Laplaciano es un operador de derivación la suma de los coeficientes debe ser cero. Además el coeficiente asociado con el píxel central debe ser positivo y los demás coeficientes negativos o ceros.

Ejemplo: $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Aunque el Laplaciano responde a transiciones en la intensidad de la imagen, se emplea en pocas ocasiones en la práctica debido a que al ser un operador de segunda derivada es muy sensible a la

presencia de ruido. En general juega un papel secundario en la detección de bordes para determinar si un píxel está en la zona clara o en la zona oscura del borde a través del signo del Laplaciano.

Las técnicas de detección de discontinuidades en la práctica proporcionan píxeles de borde que no suelen caracterizar completamente los contornos o fronteras a los que pertenecen debido a la presencia de ruido, ruptura en los propios contornos producto de la iluminación no uniforme y otros efectos que producen cambios no deseados en las discontinuidades de la intensidad.

SEGMENTACIÓN BASADA EN CRITERIOS DE SIMILITUD.

La segmentación basada en similitud es, en cierta medida, complementaria a la segmentación basada en la búsqueda de discontinuidades. Explota el concepto de similitud respecto a alguna propiedad de la imagen.

En general después de los procedimientos de detección de bordes se suele emplear técnicas de enlazado u otras de detección de contornos designadas para unir los píxeles de bordes en contornos significativos.

1. <u>Técnicas basadas en la frontera</u>

Tipos de detectores de fronteras:

- Basados en criterios locales (seguimiento del contorno).
  Analizan un contorno de vecindad del píxel dado.

  Consideran: Valor del gradiente en el punto y en el entorno.

  Dirección del gradiente en el punto y en el entorno.

- Basados en criterios globales.
  Analizan la imagen en conjunto.

  - Búsqueda heurística.
  - Ajuste de curvas.

BASADOS EN CRITERIOS LOCALES

Se analizan las características de los píxeles en un vecindario pequeño alrededor de cada punto (x, y) donde se ha detectado la presencia de un borde. Todos los puntos similares se enlazan, dando lugar a un contorno o frontera de píxeles que comparten ciertas propiedades en común.

Las dos propiedades que se suelen utilizar son la magnitud del vector gradiente que ha dado lugar al píxel de borde y la dirección del gradiente en ese punto.

Un píxel de borde con coordenadas (x', y') en el vecindario predefinido para el píxel (x, y) va a ser similar en magnitud al píxel (x, y) si:

$\left|\nabla f(x, y) - \nabla f(x', y')\right| \leq T$ *T*: cierto umbral en amplitud.

Un píxel de borde con coordenadas (x', y') en el vecindario predefinido para el píxel (x, y) va a ser similar en ángulo al píxel (x, y) si:

$\left|\alpha(x, y) - \alpha(x', y')\right| \leq A$ *A*: cierto umbral angular.

Se enlazará un punto en el vecindario predefinido para (x', y') con el píxel (x, y) si se cumplen simultáneamente las condiciones de magnitud y de ángulo. Este procedimiento se repite para todos los píxeles de borde de la imagen.

Tras encontrar un píxel de borde que enlaza con el (x, y) se procede a probar con este nuevo píxel de borde, hasta que no se puedan enlazar más píxeles de borde. Todos los píxeles enlazados se etiquetan con cierto valor de gris y se procederá a enlazar los restantes píxeles de borde con otra etiqueta.

Al final habrá tantas etiquetas como conjuntos de píxeles enlazados, y que corresponderán a los contornos o fronteras correspondientes a todos los objetos detectados de la imagen.

BASADOS EN CRITERIOS GLOBALES

En este criterio se enlazan aquellos puntos de borde que caigan sobre una determinada curva con determinada forma. A diferencia del método anterior, se consideran las relaciones globales entre píxeles de borde.

TRANSFORMADA DE HOUGH

Permite detectar curvas o fronteras de un objeto de una imagen. (Martín, 2002; Ospina y Urrea, 2004. Consulta en [15]).

- Detección de rectas:

Sea la ecuación de una recta: $y = ax + b$

Donde:

a: pendiente

b: ordenada en el origen

El método consiste en: conocidos los puntos de borde, estimar los posibles parámetros *a* y *b*.

Por el punto (x_i, y_i) pasan infinitas rectas que satisfacen la ecuación de la recta $y_i = ax_i + b$ para diferentes valores de *a* y *b*.

Sin embargo si se escribe la ecuación en la forma: $b = -x_i a + y_i$ y considerado el plano *ab* (espacio de parámetros) da lugar a una única recta para el par (x_i, y_i) constante.

Si se considera un segundo punto (x_j, y_j), también va a tener su recta asociada en el espacio parámetro. Estas dos rectas se cortarán en el espacio parámetro en un punto (a', b') donde a' es la pendiente y b' la ordenada en el origen de la recta que contiene los puntos (x_i, y_i) y (x_j, y_j) en el plano *xy*, como se puede ver en la siguiente figura:

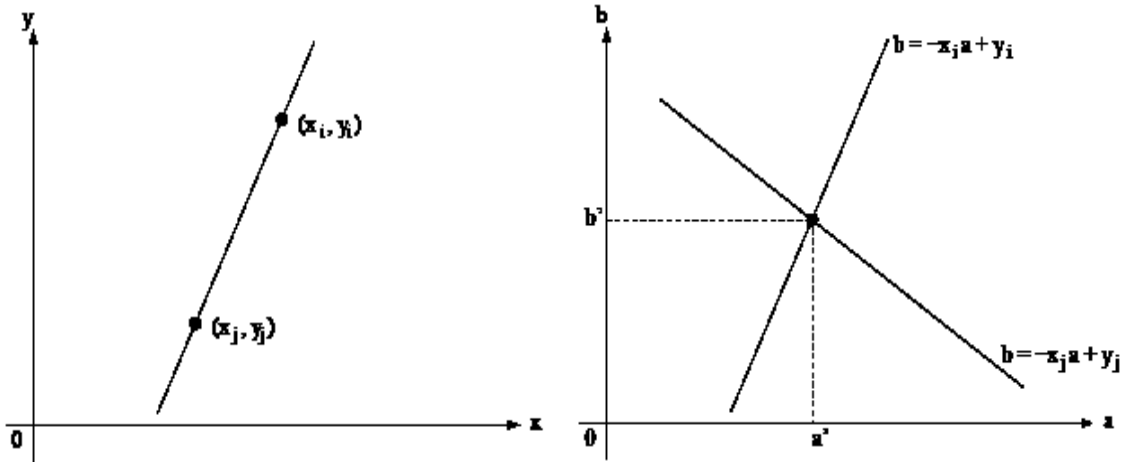


Figura 1.7. Representación del plano *XY* y del espacio de parámetros de una recta.

El atractivo de la transformada de Hough proviene de subdividir el espacio de parámetro en celdas acumuladoras, como se puede ver en la siguiente figura:

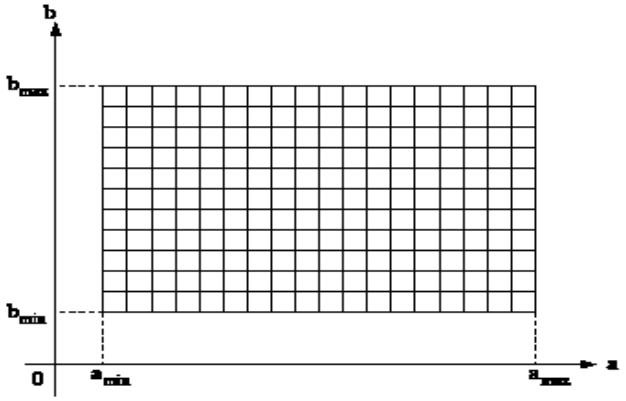


Figura 1.8. Celdas acumuladoras en el espacio de parámetros.

Donde:

$(a_{mín,}a_{máx})$ y $(b_{mín,}b_{máx})$: rangos esperados para la pendiente y la ordenada en el origen.

La celda de coordenadas (i, j) con un valor de acumulador $A(i, j)$ corresponde al cuadrado asociado con las coordenadas (a_i, b_j) del espacio de parámetros. Inicialmente se ponen todas las celdas del acumulador en cero. Entonces para cada punto (x_k, y_k) de la imagen, permitiendo que el parámetro *a* pueda tomar cualquier valor de entre los a_i permitidos, se calcula *b* usando la ecuación $b = -x_k a + y_k$.

Los valores resultantes para el parámetro *b* se redondean hasta los b_j permitidos.

Si para un valor a_p resultó un valor b_p se tiene que: $A(p,q) = A(p,q) + 1$

Al final, un valor de *M* en el acumulador $A(i, j)$ significa que *M* puntos del plano *xy* caen sobre la recta $y_i = ax_i + b$. La precisión en la colinealidad de estos puntos depende del número de celdas del espacio de parámetros.

El problema de utilizar la ecuación de la recta vista anteriormente es que tanto la pendiente como la ordenada en el origen pueden llegar a ser infinito según la recta se hace vertical. Una forma de solventar este problema consiste en utilizar la representación normal de la recta: $x\cos\theta + ysen\theta = \rho$.

En la siguiente figura se puede ver el significado de los nuevos parámetros.

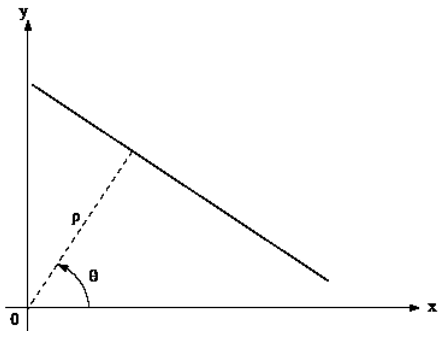


Figura 1.9. Representación normal de una recta.

El uso de la representación para construir la tabla de acumuladores es similar al método explicado para las rectas en la forma pendiente y ordenada en el origen. A cada punto del plano *XY* corresponde ahora una sinusoide en el plano $\rho\theta$ en lugar de una recta. Al igual que antes, M puntos colineales a la recta $x\cos\theta_j + ysen\theta_j = \rho_i$ darán lugar a M sinusoides que se cortan en el punto (ρ_i, θ_j) en el espacio de parámetros.

Incrementando θ y calculando ρ, obtenemos M entradas en el acumulador $A(i, j)$ correspondiente al par (ρ_i, θ_j).

- Detección de curvas

La transformada de Hough es aplicable a cualquier función de la forma: $g(v, c) = 0$

donde v: Vector de coordenadas.

c: Vector de coeficientes.

Por ejemplo, para puntos que caen en un círculo: $(x - c_1)^2 + (y - c_2)^2 = c_3^2$

En este caso se tienen tres parámetros *(c_1, c_2, c_3)*, lo que dará lugar a un espacio de parámetros de tres dimensiones, con celdas en forma de cubo y acumuladores de la forma $A(i, j, k)$. El procedimiento en este caso es para cada punto del plano *xy*, para cada c_1 y para cada c_2 calcular el valor de c_3 y actualizar el acumulador correspondiente a *(c_1, c_2, c_3)*.

Claramente la complejidad de la transformada de Hough es dependiente del tamaño del espacio de parámetros.

2. Técnicas basadas en umbrales (umbralización).

La Umbralización de propiedades (no sólo del nivel de gris o del color) es el proceso de segmentación por similitud más simple, pero resulta muy eficaz en muchas aplicaciones. La idea se basa en suponer que el rango de valores de un objeto, respecto a una determinada propiedad, es relativamente pequeño. Aquellos píxeles que estén en ese rango serán parte del objeto y aquellos fuera de ese rango serán parte del resto de la imagen. (González, 1987 y Martín, 2002; Consultas en [14], [16]).

Resulta especialmente útil cuando la imagen está formada por un único objeto de interés sobre un fondo aproximadamente uniforme. La dinámica de aplicación hace que el proceso de umbralización sea muy rápido computacionalmente pues se trata, simplemente, de recorrer secuencialmente todos los píxeles de la imagen y seleccionar aquellos que son mayores (o menores) que un umbral prefijado. La parte más difícil es encontrar el valor correcto del umbral.

Es posible segmentar la imagen en función de los valores de intensidad de los píxeles para imágenes digitales en escala de grises.

Suponer que el histograma de los niveles de gris de una imagen $I(x, y)$ se muestra en la siguiente figura:

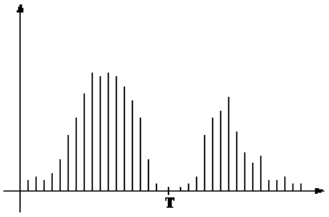


Figura 1.10. Histograma de nivel de gris de una imagen que contiene un objeto.

La imagen está compuesta de objetos claros sobre fondo oscuro de forma tal que los niveles de gris están agrupados en dos modos predominantes.

Una forma de separar los objetos del fondo consiste en seleccionar un umbral *T* que separe esos modos. Entonces cualquier punto (x, y) para el que se cumpla que: $I(x, y) > T$ se lo etiqueta como objetos, en otro caso, como fondo.

En el siguiente caso el histograma está caracterizado por tres modos dominantes. Esto ocurrirá cuando se tenga dos tipos de objetos claros sobre fondo oscuro.

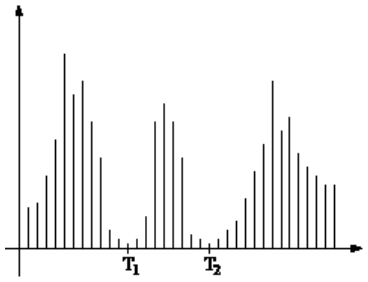


Figura 1.11. Histograma de nivel de gris una imagen que contiene dos objetos.

Se puede utilizar el mismo principio para clasificar cada punto (x, y).

Si $T_1 < I(x, y) \leq T_2$, entonces lo etiqueta como primer objeto, si $I(x, y) > T_2$ como segundo objeto y si $I(x, y) \leq T_1$ como fondo.

En general, este tipo de clasificación con varios umbrales es menos viable, ya que es más difícil determinar esos umbrales que aislen de forma efectiva las regiones de interés, especialmente cuando el número de modos del histograma aumenta. En este caso es mejor emplear umbrales variables.

En general, un método de umbral se puede ver como una operación en la que se hace una prueba a cada píxel con respecto a una función *T* de la forma: $T = T(x, y, p(x, y), I(x, y))$

donde:

$I(x, y)$: nivel de gris del punto (x, y)

$p(x, y)$: propiedad local de ese punto (por ejemplo el nivel de gris medio en una vecindad).

El método del umbral dará lugar a otra imagen $B(x, y)$ definida por:

$$B(x, y) = \begin{cases} 1 & si \quad I(x, y) > T \\ 0 & si \quad I(x, y) \leq T \end{cases}$$

En este caso un píxel con etiqueta 1 de la imagen *B* corresponderá a objetos, mientras que un píxel con etiqueta 0 corresponderá al fondo.

Cuando *T* dependa sólo del nivel de gris se denomina **umbral global.**

Si T depende tanto del nivel de gris como de la propiedad local el umbral se denomina **umbral local**.

Si además T depende de las coordenadas espaciales *x* e *y*, el umbral se denomina **dinámico.**

Umbralización global

Una técnica simple que es a menudo útil para la segmentación de una imagen consiste en la división de la escala de gris en bandas y usar umbrales para determinar regiones o fronteras de los puntos.

Como introducción a esta técnica, suponer que una imagen $f(x, y)$ en escala de niveles de grises tiene el siguiente histograma

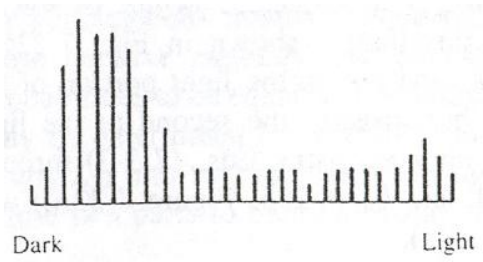


Figura 1.12. Histograma de una imagen en escala de grises.

Se puede observar que un gran numero de píxeles de $f(x, y)$ son oscuros, con presencia de píxeles que son distribuidos en la escala de gris. Este histograma es característico de imágenes que contienen objetos grises sobre un fondo oscuro.

Para encontrar la frontera entre objetos y el fondo, se divide el histograma en dos bandas separadas por un umbral T como se muestra en la figura:

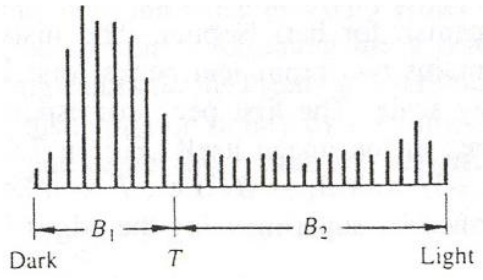


Figura 1.13. Histograma dividido en bandas separadas por un umbral.

Lo importante es seleccionar *T* de modo que la banda B_1 contenga lo más preciso posible, niveles asociados con el fondo, mientras que B_2 contenga los niveles asociados a los objetos.

Un cambio en el nivel de gris de una banda a la otra demuestra la presencia de una frontera.

Para la detección de fronteras en direcciones horizontales y verticales una vez seleccionadas B_1 y B_2 se hace mediante dos pasos:

1. Para cada fila (*x = 0, 1, …, N-1*) crear la fila correspondiente en una nueva imagen $g(x,y)$ usando la siguiente relación para *y = 0 , 1 ,…, N-1*

$$g_1(x,y)=\begin{cases} L_E & si\ el\ nivel\ de\ gris\ de\ f(x,y)\ y\ f(x,y-1)\ están\ en\ diferentes\ bandas\ de\ la\ escala\ de\ gris \\ L_B & en\ caso\ contrario \end{cases}$$

L_E : Niveles de gris especificado para bordes.

L_B : Niveles de gris especificado para fondo.

2. Para cada columna (*y = 0, 1, …, N-1*) crear la columna correspondiente en una nueva imagen $g(x,y)$ usando la siguiente relación para *x = 0 , 1 ,…, N-1*

$$g_2(x,y)=\begin{cases} L_E & si\ el\ nivel\ de\ gris\ de\ f(x,y)\ y\ f(x,y-1)\ están\ en\ diferentes\ bandas\ de\ la\ escala\ de\ gris \\ L_B & en\ caso\ contrario \end{cases}$$

La imagen diseñada que contiene los puntos en la frontera de los objetos diferentes a los del fondo, es obtenida usando la siguiente relación:

$$g(x,y)=\begin{cases} L_E & si\ \ g_1(x,y)\ y\ g_2(x,y)\ \ son\ \ iguales\ \ a\ \ L_E \\ L_B & en\ caso\ contrario \end{cases}$$

Umbralización basada en píxeles de la frontera

Conocido el gradiente *G(x,y)* y el Laplaciano *L(x,y)* de una imagen:

$$S(x,y)=\begin{cases} 0 & si\ \ G(x,y)<T \\ + & si\ \ G(x,y)\geq T\ \ \ y\ \ \ L(x,y)\geq 0 \\ - & si\ \ G(x,y)\geq T\ \ \ y\ \ \ L(x,y)<0 \end{cases}$$

Donde ***0***, **+** y **-** representan 3 niveles distintos de gris, y *T* el umbral.

Ejemplo: Si se tiene una imagen de un objeto oscuro en un fondo claro, se produce una imagen $S(x,y)$ en la cual todos los píxeles que no son puntos de bordes (determinados por $G(x,y)<T$) son etiquetados con ***0***, todos los píxeles que se encuentran en el lado oscuro de un borde son etiquetados con **+** y todos los píxeles que se encuentran en el lado claro de un borde son etiquetados con **-**.

Los símbolos **+** y **-** de la ecuación anterior están invertidos para imágenes que presentan objetos claros sobre fondos oscuros.

3. Técnicas de crecimiento de regiones.

El crecimiento de regiones es un procedimiento mediante el cual se agrupan píxeles o subregiones en regiones mayores. (González, 1987 y Martín, 2002. Consulta en [15]).

El procedimiento más sencillo se denomina agregado de píxeles, que comienza a partir de un conjunto de píxeles semilla, de forma que a partir de cada semilla crecen regiones añadiendo píxeles vecinos que tienen propiedades similares a dicha semilla. El resultado de la segmentación dará lugar a tantas regiones como semillas haya.

Sin embargo, puede darse el caso de que dos de esas semillas correspondan a píxeles de la misma región. En este caso el crecimiento desde una de las semillas absorberá a la otra, que debería ser descartada.

Un ejemplo sencillo, pero muy usado en la práctica de similitud es la diferencia absoluta en el nivel de gris. Fijado un umbral T se va calculando la diferencia en valor absoluto del nivel de gris del píxel en cuestión (en el vecindario de la región crecida hasta el momento) con respecto al nivel de gris de la semilla y si no se supera ese umbral T se añade a la región.

Dos problemas fundamentales en el crecimiento de regiones son: por un lado, la selección de las semillas o puntos de partida que representen adecuadamente a las regiones de interés; y por otro, la elección de las propiedades adecuadas que permitan ir añadiendo píxeles durante el proceso de crecimiento.

La selección de los puntos de partida en muchos casos depende de la naturaleza de la imagen a segmentar. En ausencia de conocimiento del problema, un procedimiento que suele dar buenos resultados consiste en calcular para cada píxel de la imagen las mismas propiedades que luego se emplearán durante el proceso de crecimiento. Si el resultado del cálculo de dichas propiedades da lugar a agrupaciones de los valores para dichas propiedades, se pueden emplear como semillas los píxeles cercanos a los centroides de dichas agrupaciones. Por ejemplo, los píxeles correspondientes a los valores máximos del histograma de nivel de gris de la imagen podrían servir como semillas cuando se emplee como propiedad la similitud en el nivel de gris.

Otro problema adicional en el crecimiento de regiones es la regla de parada. Básicamente, el crecimiento termina cuando no existen más píxeles en el vecindario de la región ya crecida que cumplan el criterio de similitud. Los criterios de similitud mencionados son locales y no tienen en

cuenta la historia del crecimiento de la región. Se pueden emplear criterios adicionales que incrementen la potencia del algoritmo de crecimiento de regiones, utilizando el concepto de similitud entre el píxel candidato y los píxeles de la región crecida hasta ese momento, o usando el tamaño y la forma de la región crecida.

La figura 1.14 (a) muestra una imagen con varias regiones en la que se muestra una única semilla en el interior de una de las regiones a segmentar. El criterio empleado para el crecimiento es que el valor absoluto de la diferencia entre los niveles de gris del píxel candidato y la semilla no exceda el diez por ciento de la diferencia entre el mayor nivel de gris y el menor valor de nivel de gris de la imagen completa. La figura 1.14 (b) muestra el proceso de crecimiento tras unas pocas iteraciones, la figura 1.14 (c) muestra el proceso para la mitad de las iteraciones y la figura 1.14 (d) el resultado del crecimiento de la región para cuando ninguno de los píxeles vecinos de la región cumple el criterio de similitud.

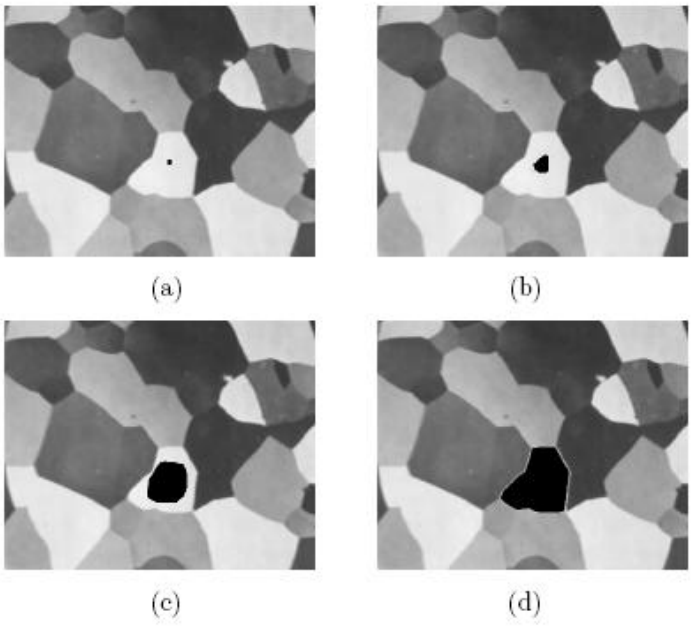


Figura 1.14. Proceso de crecimiento de semillas.

1.2.2 ESQUEMAS DE REPRESENTACIÓN.

Una vez segmentada la imagen y extraído el contorno de los objetos, es necesario analizar la forma geométrica de los mismos utilizando para su representación una estructura de datos compacta, llamada esquemas de representación.

Los descriptores basados en contornos se utilizan cuando el interés principal es describir las formas de los contornos (el reconocimiento posterior se va a basar en como es la forma de los objetos detectados). (González, 1987; Mazo, 2000. Consultas en [6] y [7]).

Asimismo, existen descriptores de contornos y de regiones para el reconocimiento e identificación de los objetos de la imagen.

Los esquemas de representación de formas deben de tener ciertas propiedades deseables:

a) Unicidad: Cada objeto debe tener una única representación.
b) Invariancia frente a transformaciones geométricas, como traslaciones, rotaciones, cambios de escala.
c) Sensibilidad o capacidad para diferenciar objetos casi iguales.

Pueden obtenerse analizando el objeto de dos formas:

Externa: Usan el contorno de los objetos y sus rasgos característicos. Ejemplo: códigos de cadena, firmas, aproximaciones poligonales.

Interna: Describen la región ocupada por el objeto en la imagen, como son el área, perímetro, momentos.

Esquemas de representación externa: Trazado y representación de contornos.

- **FIRMAS**

Es una representación de un contorno mediante una función real unidimensional.

Una de las formas más simples de definir una firma es a través de la distancia desde un punto interior (puede ser el centroide del contorno) a cada uno de los puntos del contorno, como una función del ángulo, tal y como se muestra a continuación:

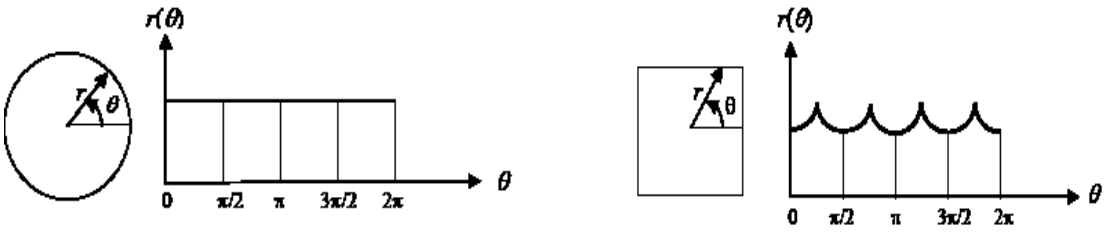


Figura 1.15. Firmas generadas para una circunferencia y para un cuadrado.

Las firmas sólo tienen sentido usarse cuando el vector que se extiende desde el origen corta en un solo punto al contorno. Es invariante frente a traslaciones, pero no frente a rotaciones o cambios de escala.

Para resolver el inconveniente de la invarianza frente a rotaciones, se puede comenzar la representación por el ángulo para el cual la distancia entre el centroide y el punto del centroide es máxima.

Para resolver el inconveniente la invarianza frente a cambio de escala, se puede dividir la función por la distancia máxima al centroide, de forma que la distancia máxima sea uno.

Inconvenientes de las firmas:

1. Es muy sensible a la posición del centroide.

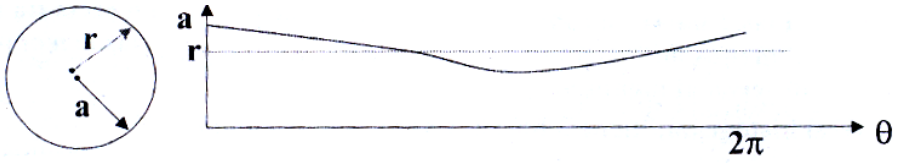


Figura 1.16. Sensibilidad de las firmas a la posición del centroide.

2. Las concavidades pueden dar lugar a una representación multielevada para algunos ángulos (varias distancias para un mismo ángulo)

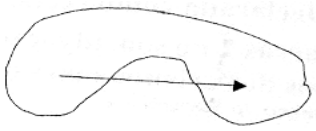


Figura 1.17. Sensibilidad de las firmas en presencia de concavidades.

- **APROXIMACIONES POLIGONALES**

Se trata de aproximar de la mejor manera posible el contorno de un objeto mediante una curva de tramos lineales que constituye un polígono, y donde los vértices de dicho polígono son una representación del contorno del objeto.

Como criterio para evaluar la calidad del ajuste se puede elegir el criterio de mínimos cuadrados.

Suponer la curva que se muestra en la siguiente figura, que va del punto x_1 al punto x_N y sean $x_1, x_2, ..., x_{N-1}$ las coordenadas de $N-2$ puntos intermedios de la curva. Si d_i es el punto del segmento $\overline{x_1, x_N}$ más próximo al punto x_i, $i = 2,3,...N-1$, entonces $|x_i - d_i|$ es el error de aproximación de la curva por el segmento lineal $\overline{x_1, x_N}$ correspondiente al píxel x_i.

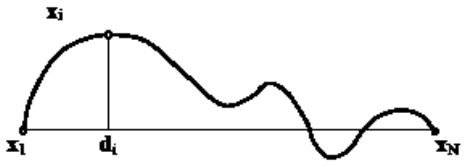


Figura 1.18. Aproximación poligonal.

La aproximación poligonal consistirá en determinar los vértices del polígono de manera que el error total sea mínimo. Sin embargo, encontrar la solución óptima para este problema requiere un costo computacional muy alto.

Para ello hay técnicas alternativas que suministran buenas soluciones y son más rápidas y eficientes.

Las técnicas de partición consisten en ir dividiendo de forma recurrente los tramos de la curva representados por un segmento en dos tramos, representados a su vez cada uno por un segmento. De manera que la medida local de error sea mínima.

Si se utiliza el criterio mínimax, es decir, minimizar el error dado por la expresión $\max_{2 \le i \le N-1} |x_i - d_i|$, entonces para cada tramo de la curva representado por un segmento, se determina el punto de la curva más alejado del segmento y se toma dicho punto como nuevo vértice.

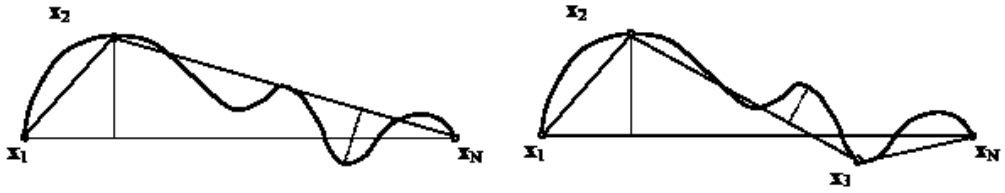


Figura 1.19. Aproximación poligonal por partición sucesiva.

El proceso acaba cuando el error máximo por píxel es menor que cierta cantidad prefijada. Los puntos x_1 y x_N pueden ser los 2 puntos más alejados del contorno. Dichos puntos dividen el contorno en dos partes y el algoritmo se aplica a cada una de ellas.

Obsérvese que en cada caso el error del píxel es igual o menor que el del paso anterior. Conforme se van introduciendo nuevos vértices se mejora la aproximación.

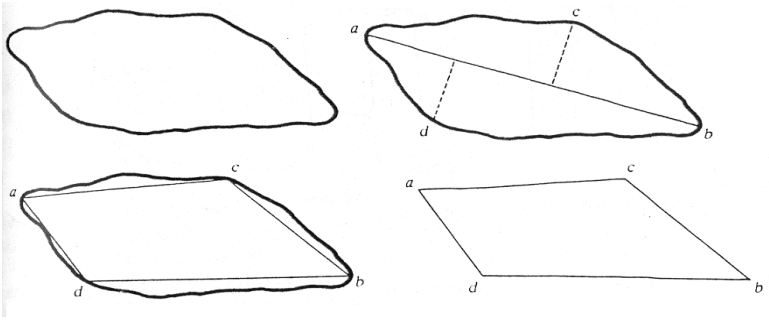


Figura 1.20. Aproximación poligonal para un contorno cerrado por partición sucesiva.

Las técnicas de incorporación operan al contrario, parten de un punto x_1 de la curva y van recorriendo los píxeles de la curva en el sentido de las manecillas del reloj.

Cuando se encuentra el primer píxel digamos x_i, para que el error de x_1 a x_i supere un umbral T (prefijado), se declara dicho punto como vértice del polígono y se repite el proceso comenzando ahora en x_i. Así se genera una secuencia de vértices que constituye la aproximación poligonal.

Esta técnica tiene el inconveniente de que los vértices no suelen ser puntos de inflexión, como ocurre con las técnicas de partición, generando vértices que no coinciden con las esquinas de la curva.

- **CÓDIGOS DE CADENA**

Es un tipo de estructura de datos para representar el contorno de un objeto en una imagen binaria mediante una secuencia de segmentos conectados consecutivamente, de longitud y orientación específica, que conectan píxeles adyacentes.

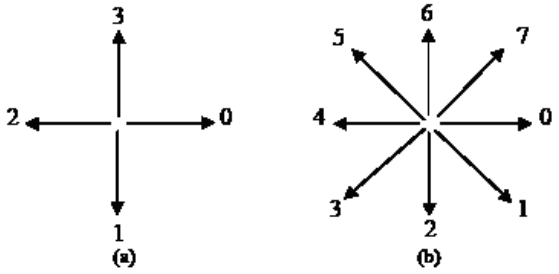


Figura 1.21. Direcciones para entornos de a) 4 vecinos b) 8 vecinos.

La conexión de los segmentos se lleva cabo en entornos de 4 ó 8 vecinos. Cuando se usa un entorno de 4 vecinos se tienen 4 orientaciones para los segmentos y se utilizan los números 0, 1, 2 y 3. Si se usa un entorno de 8 vecinos se tienen 8 orientaciones para los segmentos y se utilizan los números 0, 1, 2, 3, 4, 5, 6 y 7.

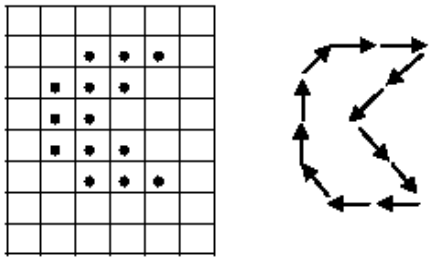


Figura 1.22. Contorno de un objeto y su código de cadena 003311445667.

Es importante notar que el código de cadena depende del punto de partida, es decir que no es invariante a rotación. Este inconveniente se puede eliminar si se considera el código como una lista circular en el caso de contornos cerrados. Además como es conveniente que tenga la propiedad de unicidad entre todas las cadenas posibles según el punto de partida que se haya tomado, se puede elegir aquella que corresponda al menor número entero.

Cuando se trata de objetos con agujeros, se necesita más de un código de cadena para representarlos, y hay que especificar si la cadena corresponde a la parte exterior del objeto o a un agujero.

Ventajas del código de cadena:

1. Es una representación invariante a traslaciones
2. A partir del código de cadena se pueden obtener características del contorno como el perímetro, área, los momentos y los descriptores de Fourier.
3. Es una representación compacta de un objeto binario. Suministra buena compresión de la descripción del contorno ya que cada cadena se puede codificar sólo con 2 bits (para entornos de 4 vecinos), en lugar de coordenadas (x, y) de cada píxel del contorno.

Inconvenientes:

1. La cadena resultante suele ser demasiado larga.
2. Cualquier ruido en el contorno produce segmentos erróneos.

Esquemas de representación interna: Descriptores de contorno y de regiones.

Los descriptores de contorno permiten estudiar la forma geométrica de los contornos de las regiones (objetos) que conforman la imagen digital utilizando para ello estimaciones numéricas que permiten identificar y reconocer los objetos de dicha imagen.

- Algunos descriptores de contorno.

Perímetro: en imágenes binarias viene dado por la longitud de su código de cadena (con 8 direcciones) pero ponderando los pasos diagonales por $\sqrt{2}$ y los horizontales y verticales por 1, es decir, viene dado por el número total de píxel que configuran su contorno pero los píxeles de borde diagonales se ponderan con $\sqrt{2}$.

Diámetro: Viene dado por la distancia Euclídea entre los dos píxeles mas alejados del contorno. Dichos puntos no siempre son únicos, como ocurre en una circunferencia, pero es un descriptor de interés. La recta que pasa por dichos puntos se llama eje mayor de la región.

Centro de gravedad o centroide: Está determinado por el conjunto de píxeles $(x_i, y_i), i = 1,2,...N$, y es el punto $(\bar{x}, \bar{y})$ definido por las siguientes expresiones:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \qquad \bar{y} = \frac{\sum_{i=1}^{N} y_i}{N}$$

- Algunos descriptores de regiones.

Aunque las regiones (objetos) pueden definirse por sus contornos, se estudian además algunas características geométricas y topológicas de las regiones que ayudan a identificar y reconocer los objetos de la imagen digital.

a) Parámetros geométricos

Área: Viene dada por el número de píxeles que la componen. Se puede obtener de forma sencilla por el código de cadena del contorno del objeto.

Centro de gravedad o centroide: Está determinado por el conjunto de píxeles $(x_i, y_i), i = 1,2,...N$, y es el punto $(\bar{x}, \bar{y})$ definido por las siguientes expresiones:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \qquad \bar{y} = \frac{\sum_{i=1}^{N} y_i}{N}$$

El centroide de una región no tiene por qué coincidir con el centroide de su contorno.

b) Parámetros topológicos

La topología es el estudio de configuraciones geométricas con propiedades específicas como la invariancia bajo ciertas transformaciones, por ejemplo la escala.

Compacidad (circularidad): Es un parámetro que no depende del tamaño de la región (como los anteriores), y viene dado por: $c = \frac{A}{p^2}$

Donde:

A: Área de la región

p: Perímetro de la región

El perímetro se eleva al cuadrado para conseguir un parámetro adimensional. Su valor máximo corresponde a los círculos (máxima superficie para un perímetro dado) y vale $1/4\pi = 0.07957$, por ello es una medida de circularidad. Los valores pequeños de este parámetro indican objetos alargados. Además es invariante frente a traslaciones, giros y cambios de escala.

Momentos

La teoría de momentos provee una alternativa usual en la representación de formas de objetos.

Teorema 1.1. Representación de momentos

Sea $f(x, y)$ una función continua, tal que $f(x, y) \neq 0$ solamente en una región finita del plano *XY*, entonces el conjunto infinito de momentos de dicha región m_{pq}, $p, q = 0,1,2,...$ existe y está determinado de manera única por $f(x, y)$, e inversamente los momentos m_{pq} determinan de manera única $f(x, y)$. (Mazaira, 1994).

Definición 1.1. Sea $f(x, y)$ una función continua, acotada y positiva, definida sobre una región *R* del plano *XY*, entonces el momento de orden pq de $f(x, y)$ se define como:

$$m_{pq} = \iint_R x^p y^q f(x,y)dxdy$$

La versión discreta de esta definición, en vez de la doble integral, utiliza una doble sumatoria, quedando definidos los momentos de la siguiente manera:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x,y)$$

Donde $I(x,y)$ representa la función discretizada, definida en el dominio discreto $0 \le x \le M,\ 0 \le y \le N$. Para su aplicación en imágenes binarias $I(x,y)$ puede tomar sólo dos valores (0,1), finalmente en el caso de imágenes binarias la expresión para el cálculo de los momentos queda como sigue:

$$m_{pq} = \sum_x \sum_y x^p y^q$$

en la región en la cual $I(x,y) = 1$.

Obsérvese que m_{00} brinda información del área del objeto y que $\left(m_{10}/m_{00}, m_{01}/m_{00} \right)$ es el centroide del objeto.

Los momentos de orden superior no son invariantes a traslaciones, rotaciones ni escalado.

Los momentos invariantes han sido usados como atributos para describir o caracterizar objetos en procesos de reconocimiento cuando se tiene unicidad de forma pero sin perspectiva, independientemente de su localización, medida y orientación.

Definición 1.2. Momentos centrales.

Sean $\bar{x}$ y $\bar{y}$ las coordenadas del centro de masa de un objeto en la imagen, definidos por:

$$\bar{x} = \frac{m_{10}}{m_{00}}$$

$$\bar{y} = \frac{m_{01}}{m_{00}}$$

Los momentos centrales de orden pq del objeto se definen por:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q$$

Bajo una traslación de coordenadas $x' = x + \alpha, \quad y' = y + \beta$, los momentos centrales permanecen invariantes. Sin embargo, bajo un cambio de escala $x' = \alpha x, \quad y' = \beta y$, los momentos de $f(\alpha x, \beta y)$ cambian a $\mu'_{pq} = \dfrac{\mu_{pq}}{\alpha^{(p+q+2)}}$.

Para obtener un momento invariante a escala se definen los momentos centrales normalizados.

Definición 1.3. Momentos centrales normalizados.

Los momentos centrales normalizados se definen como:

$$\eta_{pq} = \frac{\mu'_{pq}}{(\mu'_{00})^{\gamma}}$$

donde: $\gamma = \dfrac{(p+q+2)}{2}$

Los momentos centrales normalizados son invariantes a traslación y escala, pero no a rotación.

En 1962, Hu (Mazaira, 1994) propuso un conjunto de siete momentos invariantes a traslaciones, cambios de escala y rotación, los cuales están dados por las siguientes expresiones:

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{02})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (3\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] +$$
$$+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

$$\phi_6 = (\eta_{20} - 3\eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{03} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] +$$
$$+ (3\eta_{12} + \eta_{30})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{30})^2\right]$$

Los momentos invariantes al ser atributos de regiones son útiles cuando se cuenta con información total del objeto que se desea reconocer, es decir en aplicaciones en las cuales se desea reconocer objetos que tengan unicidad de forma.

1.3 ESTADO ACTUAL DE LOS MÉTODOS DE RECONOCIMIENTO DE OBJETOS QUE SE ENCUENTREN PARCIALMENTE OCULTOS EN IMÁGENES DIGITALES.

El reconocimiento a partir de información incompleta de objetos en imágenes digitales, constituye uno de los problemas fundamentales y a su vez más difícil de resolver en la visión por computadora.

A modo de ejemplo, no es difícil para cualquier ser humano reconocer que la cantidad de información presente en la imagen que se muestra a continuación, es suficiente para “reconocer” que se trata de un caballo.

El problema consiste entonces en la búsqueda de métodos que permitan de manera automática la detección aún en presencia de solapamiento o deterioro de parte de los objetos de interés.

Figura 1.23. Imagen parcial de un caballo.

A pesar de que se reportan diferentes métodos que tratan de resolver esta tarea, puede considerarse todavía un problema abierto al no existir un algoritmo universal que resulte óptimo en todos los casos. Por lo general cada enfoque de solución se restringe a clases específicas de objetos en presencia de determinadas deformaciones.

Tendencia actual en los métodos de reconocimiento de objetos parcialmente ocultos.

La bibliografía consultada revela que la mayoría de los trabajos realizados en esta temática se centran en el problema de la detección de objetos basado en modelo o patrón, lo cual consiste en: Dado un modelo o patrón de un objeto que se tiene como referencia en una base de datos, el cual puede ser una frontera, un conjunto de puntos característicos o un patrón, localizar en una imagen de entrada un objeto que tenga rasgos similares a los del modelo de referencia.

Es conocido que la detección en objetos de imágenes reales es extremadamente difícil (aún cuando el patrón de forma sea conocido), debido principalmente al ruido en la imagen, ocultamiento y distorsión del objeto buscado, y además es excesiva la complejidad computacional.

Trabajos como (Amit y Kong, 1996; Berg y col., 2005; Clemens y Jacobs, 1991; Felzenszwalb, 2005; Jurie y Schmid, 2004; Mikolajczyk y col., 2003; Olson y Huttenlocher,1997 y Sclaroff y Liu, 2001) tratan esta temática de detección de objetos basados en modelo y extraen características de la imagen a bajo nivel como son las esquinas, bordes, regiones y puntos de interés, luego localizan la frontera del objeto agrupando estos rasgos y a continuación los comparan con los rasgos del modelo de referencia del objeto en la base de datos. En estos trabajos, la complejidad computacional usualmente es muy alta, y el ocultamiento de objetos se plantea como un gran reto para los algoritmos de detección.

En (Yáñez y Azimi, 1999) se presenta un enfoque automatizado para la identificación de objetos parcialmente ocultos. El contorno de cada objeto relevante en la escena analizada se modela mediante el método de aproximación poligonal, cuyos bordes se proyectan en el espacio de Hough. Una red neuronal adaptativa, genera grupos de bordes colineales y/o paralelos, que se usan como base para identificar los objetos parcialmente ocultos, dentro de cada aproximación poligonal. Se reflejan resultados para la detección de partículas en fibras de vidrios en imágenes microscópicas que muestran la rapidez del esquema computacional y las prestaciones ante condiciones diferentes.

En (Horáẽek y col., 2005) se introduce un método para el reconocimiento de objetos binarios considerados de frontera sin discontinuidades utilizando trasformadas afines. El método utiliza descriptores locales invariantes afines de la frontera del objeto como puntos de inflexión y vectores radiales. Primero la forma del objeto se divide en partes las cuales son definidas mediante los puntos de inflexión de la frontera del objeto. Luego la forma de cada parte se describe por un tipo especial de vector radial. Finalmente los parámetros de la deformación afín se estiman y la clasificación se desarrolla por la comparación en el espacio de vectores radiales. En los experimentos realizados se demostró que si la curva tiene puntos de inflexión sobresalientes, estos son usualmente muy estables bajo transformaciones afines y el método trabaja perfectamente. En caso de fronteras oscuras, los

puntos de inflexión pueden ser detectados en diferentes posiciones dependiendo de la transformación particular y/o del ocultamiento, por lo que el algoritmo puede fallar.

En (Krolupper y Flusser, 2007) introducen un método el cual es apropiado para el reconocimiento de objetos parcialmente ocultos representados solamente por su contorno, siempre y cuando los objetos posean puntos de inflexión. Para determinar los puntos de inflexión del contorno del objeto, no se utiliza ningún operador derivada lo que lo hace insensible al ruido, simplemente a cada punto del contorno se le traza un círculo con centro en el punto de análisis, la razón entre el área completa del círculo y el área del círculo que está en la parte inferior del objeto se utiliza para estimar la curvatura del contorno. Si esta razón es igual a 0.5 entonces el punto de análisis es un punto de inflexión o pertenece a una línea recta (Horáẽek y col., 2005). A diferencia de (Horáẽek y col., 2005), este algoritmo utiliza el método de aproximación poligonal para la detección de la forma del objeto a partir de la detección de los puntos de inflexión del objeto. Se realiza una comparación de la forma entre los objetos de las imágenes reales y los de una base de datos. Por último se realiza una comparación con el método propuesto en (Lamdan y col., 1988), concluyendo en las ventajas que tiene el algoritmo propuesto con respecto a la inmunidad al ruido Gaussiano y la exactitud en la detección de los puntos de inflexión.

En (Ge y col., 2008) se presenta un método novedoso basado en modelo para detectar objetos de interés en imágenes reales por macheo de la forma, utilizando solamente el contorno de los objetos. Para localizar un objeto de interés que tiene una forma similar a una frontera de un objeto de referencia dada, el método propuesto integra 3 componentes: agrupado del contorno, macheo de la forma parcial del objeto y la verificación de la frontera. Los resultados muestran que el método puede ser muy útil en aplicaciones donde la precisión de la detección es el objetivo principal y no la detección de todos los objetos presentes en la imagen. Los experimentos demuestran una gran funcionabilidad del método propuesto y se evalúa localizando y reconociendo objetos en un conjunto de imágenes reales con variaciones en la iluminación, ocultamientos de objetos y presencia de ruido.

En los trabajos mencionados se observa un enfoque común de análisis de los objetos, el cual consiste en analizar los mismos solamente teniendo en cuenta su contorno. Los métodos implementados corroboran resultados positivos en las aplicaciones específicas de cada trabajo y rapidez computacional debida precisamente al análisis de los objetos teniendo en cuenta su frontera. Todos los trabajos se basan además en la comparación de los objetos parcialmente ocultos o deformados presentes en las imágenes reales con objetos de referencia almacenados en una base de datos, lo que permite reconocer rasgos característicos de cada objeto sin tener la información total

de los mismos. Este enfoque a criterio de la investigadora es bueno y reporta resultados positivos, sin embargo en este trabajo se abordará el problema de reconocimiento pero desde otro punto de vista, la idea principal es tratar de reproducir en la máquina la forma en que los humanos reconocen los objetos de forma inteligente, sin realizar comparaciones de forma con otros objetos. Los humanos identifican un objeto por los propios atributos que permiten identificar a unos de otros, los cuales aprenden de alguna manera, empleando los sentidos de percepción.

La complejidad del problema radica en encontrar esos atributos que caracterizan esos objetos, y reproducir en la máquina esa inteligencia humana para identificar los mismos. Para llevar a cabo la implementación del reconocimiento de un objeto, lo primero es ir identificando por orden de complejidad las diferentes clases de objetos, y caracterizar los atributos que distinguen unas de las otras.

En base a lo anterior el presente trabajo se centrará en el estudio de imágenes en escala de grises para el reconocimiento de polígonos regulares parcialmente ocultos, utilizando como punto de inicio algunas de las técnicas de procesamiento de imágenes estudiadas anteriormente, como es la segmentación basada en discontinuidad y la descripción del contorno de los objetos utilizando el código de cadena.

La nueva propuesta permite a partir de la determinación del atributo que distingue cada uno de los objetos poligonales regulares (ángulos interiores), brindar información sobre el polígono regular que está parcialmente oculto. Todo lo cual se aborda en el capítulo 2.

CONCLUSIONES

La bibliografía consultada evidenció que no existe en la literatura un método general que permita el reconocimiento de objetos que se encuentren parcialmente ocultos sin necesidad de contar con un modelo o patrón de referencia del mismo.

Para el reconocimiento de objetos poligonales regulares parcialmente ocultos:

1. Basta conocer uno de los ángulos interiores pues todos son iguales y para cada tipo de polígono regular los ángulos interiores son diferentes, o conocer un lado completo del polígono y parte del lado consecutivo de forma tal que se pueda determinar el ángulo comprendido entre estos dos lados.
2. Analizar el contorno de este tipo de objetos es suficiente para determinar características de los mismos que permitan su reconocimiento.
3. El código de cadena es un esquema de representación de contornos sencillo de implementar para obtener una descripción del contorno de estos objetos, y se pueden obtener a partir de él descriptores como área y perímetro.

CAPÍTULO II

INTRODUCCIÓN

En este capítulo se propone un algoritmo para el reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales en escala de grises. El algoritmo propuesto se basa en la detección del contorno del objeto, y el uso del código de cadena para extraer características que permitan su reconocimiento. Se da una breve explicación de la representación de rectas digitales y su código de cadena para una mejor comprensión del algoritmo desarrollado. Se utiliza el método de los mínimos cuadrados para el ajuste de las rectas de los lados y con ello la determinación de la información necesaria para llegar a la conclusión del tipo de polígono regular que está parcialmente oculto. Se corroboran los resultados mediante simulación a través de un ejemplo donde se muestran cada uno de los pasos a seguir para el reconocimiento de un objeto poligonal parcialmente oculto por otro polígono que no necesariamente tiene que ser regular. Por último se dan valoraciones de expertos.

2.1 FUNDAMENTOS TEÓRICOS DE LA PROPUESTA.

En el capítulo anterior se analizaron diferentes métodos para el procesamiento de imágenes digitales. Se mencionaron métodos para el reconocimiento de objetos basados en información sólo del contorno, los cuales proporcionan rapidez y en las aplicaciones mencionadas se demuestran la garantía de trabajar sólo con esta información y no con la de toda la región que ocupa el objeto.

Constituyen aspectos teóricos del fundamento de la propuesta la segmentación de la imagen basada en la detección de discontinuidades (Esqueda, 2002; González, 1987), así como el método del código de cadena (González, 1987; Mazo, 2002) para la descripción del contorno de los objetos.

La segmentación de la imagen basada en la detección de discontinuidades permite detectar los bordes de objetos en imágenes donde existan cambios bruscos en el nivel de gris entre el fondo y el objeto.

La ventaja de representar el contorno del objeto mediante el código de cadena es que al implementar un algoritmo para el seguimiento del contorno, se obtiene directamente el código de cadena del mismo, lo que constituye una gran ventaja pues el algoritmo se hace más rápido.

Con la utilización del método del código de cadena se obtendrá un código equivalente a la posición de cada uno de los píxeles que componen el contorno del objeto, sin necesidad de conocer la posición de cada píxel del contorno, por lo que con su uso se reduce la utilización de la cantidad de memoria de almacenamiento, lo que garantiza eficiencia en el algoritmo propuesto. Este método además permitirá la determinación de la longitud de rectas digitales y el ajuste de las mismas por el método de los mínimos cuadrados sin necesidad de conocer la posición exacta de cada píxel de la recta.

2.2 ALGORITMO PARA EL RECONOCIMIENTO DE OBJETOS POLIGONALES REGULARES PARCIALMENTE OCULTOS EN IMÁGENES DIGITALES EN ESCALA DE GRISES.

En el desarrollo de este trabajo se implementará un algoritmo para el reconocimiento en imágenes digitales en escala de grises de objetos poligonales regulares parcialmente ocultos por otros polígonos que pueden o no ser regulares, basado en el análisis solamente del contorno de los objetos.

Se consideran imágenes digitales en escala de grises de 16 niveles. El cero corresponde al color negro (fondo de la imagen) y el 15 al color blanco (polígono regular parcialmente oculto), los valores del 1 al 14 dan diferentes tonalidades de gris (polígonos que ocultan) los cuales pueden ser o no regulares.

Se considera que los lados del polígono están compuestos como mínimo por 12 píxeles.

Para comprobar el desempeño del algoritmo propuesto se crearon imágenes en escala de grises utilizando el paquete profesional AutoCad 2007.

2.2.1 ALGORITMO DE SEGUIMIENTO DE CONTORNO.

A continuación se muestra el diagrama en bloques del algoritmo de seguimiento del contorno.

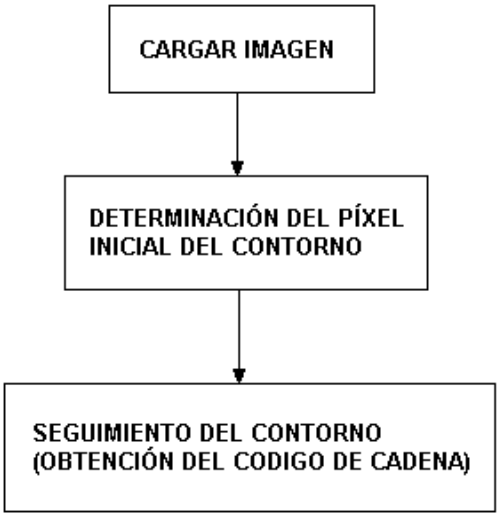


Figura 2. 1. Diagrama en bloques del algoritmo de seguimiento de contorno.

Como se observa el algoritmo de seguimiento de contorno consta de tres pasos fundamentales, que van desde cargar la imagen desde un archivo hasta obtener finalmente el código de cadena del contorno del objeto.

A continuación se explica en qué consiste cada uno de estos bloques, los cuales constituyen el paso inicial para el reconocimiento del polígono regular parcialmente oculto.

CARGAR IMAGEN

Se realiza a través de la función **Imread** del Toolbox de Procesamiento de Imágenes del paquete Matlab 7.0.

Características de la función **Imread:**

Se utiliza para leer imágenes contenidas en archivos gráficos.

Sintaxis: A = IMREAD (FILENAME,FMT)

Si el archivo es una imagen en escala de gris, A será un arreglo bidireccional (N x M). Cada elemento de la matriz se corresponde con el valor del nivel de gris de la imagen en el píxel correspondiente.

Si el archivo es una imagen a color, A es un arreglo tridimensional (N x M x 3).

Los formatos de los archivos pueden ser JPEG, TIFF, GIF, BMP, PNG, HDF, entre otros (ver ayuda del Matlab 7.0).

El píxel inicial de la imagen tiene coordenadas (0,0) y el último elemento (N-1, M-1), en caso de una imagen de tamaño N x M, donde la imagen contiene un total de N filas que van de 0 a N-1, y un total de M columnas que van de 0 a M-1.

DETERMINACIÓN DEL PÍXEL INICIAL DEL CONTORNO.

Una vez cargada la imagen, el primer paso para recorrer el contorno del objeto y obtener el código de cadena del mismo, es determinar un píxel del contorno que corresponda al primer píxel de un lado del objeto. No importa cual lado sea, lo importante es garantizar que el píxel encontrado corresponde al primero de ese lado.

Para una mejor comprensión de lo planteado anteriormente y para conocer algunas características de la conexión de los píxeles en polígonos regulares a continuación se explica la particularidad de las rectas digitales, debido a que los lados de los polígonos no son más que rectas orientadas de formas diferentes y conectadas una a otra.

Rectas digitales. Código de cadena.

El proceso de digitalización de rectas introduce algunas particularidades que es importante tener en cuenta a la hora de identificarlas en imágenes digitales.

En el epígrafe 1.2.2 del capítulo 1 se estableció que el código de cadena es un tipo de estructura de datos para representar el contorno de un objeto en una imagen mediante una secuencia de segmentos conectados consecutivamente, de longitud y orientación específica, que conectan píxeles adyacentes. Se puede considerar la 4 vecindad, o la 8 vecindad del píxel como se muestra en la siguiente figura:

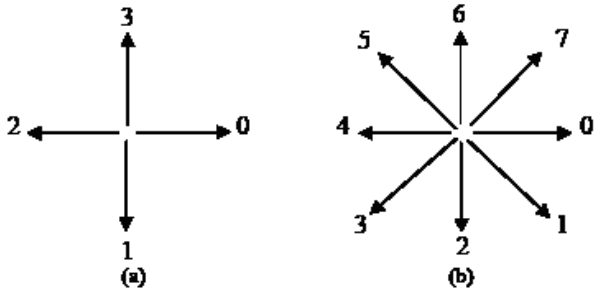


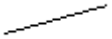
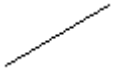
Figura 2.2. Direcciones para entornos de a) 4 vecinos b) 8 vecinos.

Cuando las rectas tienen una orientación de $\pm 45^{\circ}k$, para k=0, 1, 2…, el código de cadena tendrá un sólo segmento, dependiendo de la orientación de la recta, los cuales pueden ser 0, 1, 2, 3, 4, 5, 6 y 7, si se considera la 8 vecindad del píxel o de 0, 1, 2 y 3, si se considera la 4 vecindad del píxel.

En el algoritmo propuesto se utiliza la 8 vecindad del píxel.

Sin embargo cuando la recta tiene una orientación diferente a las mencionadas anteriormente, el código de cadena tendrá dos segmentos, los cuales son consecutivos según las orientaciones mencionadas anteriormente para 4 u 8 vecindades del píxel. La longitud del código de cadena depende de la longitud de la recta, es decir del número de píxeles que contenga la misma.

A modo de ejemplo se muestran diferentes rectas digitales de diversas orientaciones, y su correspondiente código de cadena.

Orientación de la recta	**Representación**	**Código de cadena**
Recta de 0°		000000000000000000
Recta de 15°		0700070007007000 70
Recta de 30°		707077070707707077
Recta de 45°		777777777777777777

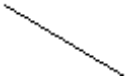
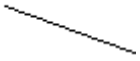
Recta de 60°		767676776767677676
Recta de 75°		7666766676667666676
Recta de 90°		222222222222222222
Recta de 110°		212212122122122121
Recta de 120°		121212112121211212
Recta de 135°		111111111111111111
Recta de 150°		1010110101011011011
Recta de 160°		0100101001001001100
Recta de 180°		0000000000000000000

Tabla 2.1. Representación y código de cadena de algunas rectas digitales.

Las rectas con orientación de (180°+α), donde α es la orientación de rectas entre 0° y 180°, tienen el código de cadena igual que el de las rectas de orientación α, pues al ser sus ángulos de orientación coterminales, los píxeles están conectados de igual forma. Sin embargo cuando se tiene un contorno abierto o cerrado el código de cadena del mismo depende del sentido de recorrido, y aunque el contorno presente rectas orientadas de igual manera, el código de cadena de las mismas no coincide.

En la figura 2.3 se muestra un polígono regular de ocho lados, donde se observa la diferencia en los códigos de cadena de cada lado, aún cuando las rectas que componen el polígono tengan igual orientación.

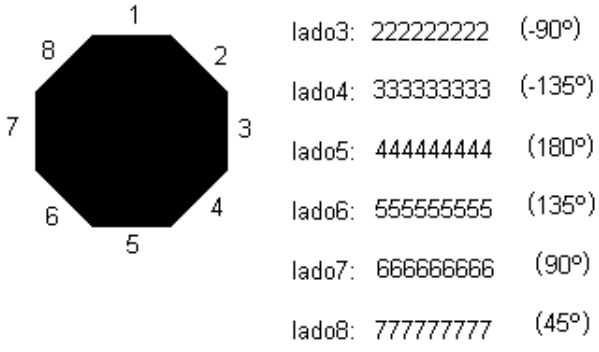


Figura 2.3. Código de cadena de los lados de un polígono de ocho lados.

Para determinar el píxel inicial del contorno, se analizaron diferentes variantes, cada una con sus ventajas y desventajas, lo que permitió elaborar un algoritmo capaz de encontrar el píxel inicial independientemente de la posición y orientación del objeto en la imagen. Los casos analizados corresponden a polígonos regulares que constituyen el objeto de estudio de este trabajo.

Variante 1: Realizar un barrido de la imagen por filas.

Realizar un barrido de la imagen por filas de izquierda a derecha y buscar el primer píxel de la imagen que tiene el nivel de gris correspondiente al color blanco (valor de 15).

Problemas: en la figura 2.4 se muestra la desventaja que tiene encontrar el primer píxel del contorno mediante esta variante en algunos casos. El problema radica en que si se hace un barrido por filas de izquierda a derecha, no necesariamente el primer píxel encontrado corresponde al primer píxel de un lado del polígono, lo cual depende de la orientación del objeto en la imagen. Esto trae como

consecuencia que cuando se obtiene el código de cadena del contorno es imposible determinar el código correspondiente a cada lado del polígono ya que la información inicial del código de cadena del contorno pertenecerá al mismo lado que el correspondiente a la última información del mismo.

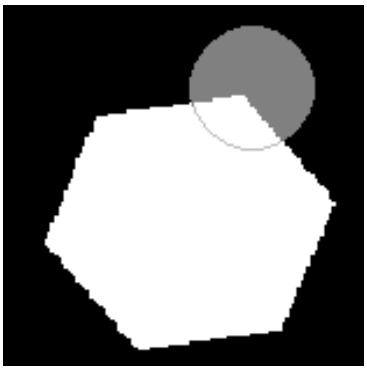


Figura 2.4. Polígono de 6 lados.

Variante2: Realizar un barrido de la imagen por columnas.

Realizar un barrido de la imagen por columnas de arriba hacia abajo y buscar el primer píxel de la imagen que tiene el nivel de gris correspondiente al color blanco (nivel de gris igual a 15).

Problemas: el problema de la variante 1 no aparece en este caso (ver figura 2.5), pero se observa la desventaja que tiene encontrar el primer píxel del contorno mediante esta variante. El problema radica en que si se hace un barrido por columnas de arriba hacia abajo, no necesariamente el primer píxel encontrado corresponde al primer píxel de un lado del polígono. La consecuencia de utilizar esta variante en este y otros casos es la misma que la de la variante 1.

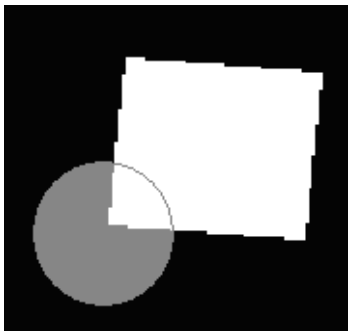


Figura 2.5. Polígono de cuatro lados.

Existen casos en los cuales la variante 1 resuelve el problema y otros en los cuales se resuelven por la variante 2, sin embargo qué hacer si existe una imagen en la cual el objeto está orientado de forma tal que estén presentes ambos problemas? (ver figura 2.6).

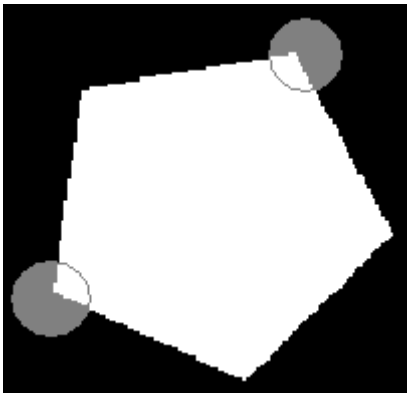


Figura 2.6. Polígono de cinco lados.

Se podría pensar en hacer un barrido de la imagen por columnas y luego por filas, de forma tal que se combinen las dos variantes anteriores, sin embargo el algoritmo podría ser algo lento, y por lo tanto poco eficiente, y aun así existen casos en los cuales no se resolvería el problema como es el caso de la figura 2.6. Entonces además de la combinación de ambas variantes habría que añadir algún otro elemento que resuelva el problema, pero que sea aplicable a cualquier situación, es decir independientemente de la orientación del objeto en la imagen.

Luego de realizar un análisis en imágenes donde se encontraban estos casos de análisis, se decidió realizar lo siguiente: hacer un barrido de la imagen por filas de izquierda a derecha, se encuentra el primer píxel que tenga nivel de gris correspondiente al color blanco (valor de 15) y se valora si el mismo es o no el píxel inicial del contorno.

Para determinar si el píxel encontrado es o no el inicial de un lado del polígono basta con analizar el color (si son blancos o negros) de los vecinos (x+1, y-1) y (x+1, y-2), suponiendo que el píxel de análisis tiene coordenadas (x, y).

El fundamento de esta conclusión de debe a que el código de cadena de las rectas digitales puede tener uno (para rectas de $\pm 45^{\circ}k$) o dos segmentos consecutivos (para el resto de las rectas) como se explicó anteriormente.

En la figura 2.7 se muestran diferentes conexiones de píxeles y se analiza en cada uno de los casos si el píxel inicial encontrado es o no el inicial de un lado. Nótese que cada caso representa la conexión de dos rectas orientadas de forma diferentes, o lo que es lo mismo una parte de un polígono regular o no.

Este constituye un paso muy importante para obtener el código de cadena del contorno del objeto, y con ello la efectividad del algoritmo que se propone en el desarrollo de este trabajo.

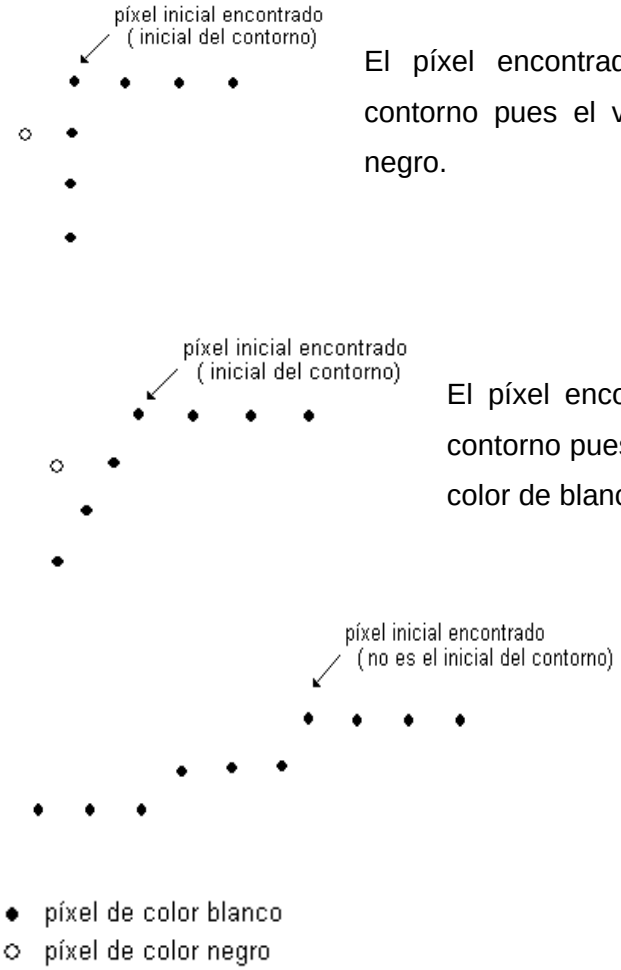


Figura 2.7. Conexiones de píxeles para determinar el píxel inicial del contorno.

De forma general el algoritmo puede utilizarse para la determinación del píxel inicial del contorno de un polígono regular o no, y consiste en lo siguiente:

Se determina el primer píxel del contorno que tenga color blanco por la variante 1 explicada anteriormente, y se considera que el píxel encontrado tiene coordenadas (x, y).

- Si el vecino (x+1, y-1) del píxel encontrado tiene nivel de gris correspondiente al color negro, el píxel encontrado es el inicial del contorno (caso1).
- Si el vecino (x+1, y-1) tiene el color blanco, y el vecino (x+1, y-2) el color negro, el píxel encontrado anteriormente es el inicial del contorno (caso2).
- Si el vecino (x+1, y-1) tiene el color blanco, y el vecino (x+1, y-2) tiene color blanco, entonces se continúa el recorrido por la misma fila del píxel encontrado inicialmente, y el último píxel de la fila que tenga color blanco, corresponde al píxel inicial del contorno (caso3).

SEGUIMIENTO DEL CONTORNO. OBTENCIÓN DEL CÓDIGO DE CADENA.

Una vez encontrado el píxel inicial del contorno del objeto, se recorre el mismo y se obtiene simultáneamente el código de cadena.

Definición 2.1: Sea Ω una región discreta, $\partial\Omega$ su contorno y sea un punto **p** $\in\partial\Omega$, se llama **dirección actual (a_i)** del punto **p** a la dirección en que es encontrado el píxel **p+1** perteneciente al contorno de la región, y **dirección anterior (a_{i-1})** del punto **p** a la dirección en que fue encontrado el punto **p** (ver figura 2.8).

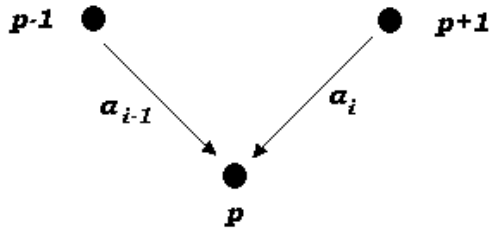


Figura 2.8. Dirección anterior y actual del punto p.

El algoritmo propuesto permite obtener directamente a medida que se recorre el contorno del objeto, el código de cadena. Partiendo del píxel inicial, se sigue el contorno de la región teniendo en cuenta la variación de x y y (coordenadas del píxel de análisis) de acuerdo al valor que va tomando a_{i-1}, según se muestra en la tabla 2.2.

a_{i-1}	X	y
0	+1	0
1	+1	+1
2	0	+1
3	-1	+1
4	-1	0
5	-1	-1

6	0	-1
7	+1	-1

Tabla 2.2. Variación de x y y en función de a_{i-1}.

Analizando las posibles posiciones de los píxeles vecinos con respecto al píxel de análisis (p) teniendo en cuenta la posición en la que fue encontrado el píxel actual (p), y para lograr una mayor rapidez en la ejecución del algoritmo, se concluye que no es necesario preguntar por el nivel de gris de los 8 vecinos del píxel en análisis (p), sólo es necesario preguntar por aquellos que realmente pudieran ser píxeles que estén conectados al píxel de análisis (p) con el mismo nivel de gris, pero siempre teniendo el cuenta la posición (a_{i-1}) en la que fue encontrado el píxel de análisis (p). Las posibles posiciones de los píxeles vecinos, en dependencia de la posición (a_{i-1}) en la que fue encontrado el píxel actual (p) con respecto al anterior (p-1), se muestran en la tabla 2.3.

a_{i-1}	**Posición de los posibles vecinos**
0	7, 0, 1, 2, 3, 4
1	7, 0, 1, 2, 3, 4, 5
2	1, 2, 3, 4, 5, 6
3	1, 2, 3, 4, 5, 6, 7
4	3, 4, 5, 6, 7, 0
5	3, 4, 5, 6, 7, 0, 1
6	5, 6, 7, 0, 1, 2
7	5, 6, 7, 0, 1, 2, 3

Tabla 2.3. Posibles vecinos según la dirección a_{i-1} .

El algoritmo recorre el contorno del objeto poligonal regular parcialmente oculto, siempre buscando de cada píxel del contorno aquel vecino que tenga un color blanco. Como se explicó en la tabla 2.3 no se pregunta por los 8 vecinos del píxel, sólo por aquellos que pueden estar conectados a él con el mismo color (blanco) y siempre en el orden señalado.

El contorno se marca con un color de gris diferente entre 0 y 15, éste valor no influirá en nada, es sólo para resaltar el contorno del objeto con un nivel de gris diferente al del objeto y al del fondo de la imagen. En el desarrollo del algoritmo se realizó el marcado del contorno con un nivel de gris igual a 7, y se muestra en el epígrafe 2.3 donde se corroboran los resultados del algoritmo (figura 2.18).

El algoritmo gráfico para el seguimiento del contorno del objeto y el marcado del mismo se muestra en el anexo 1.

2.2.2 ALGORITMO PARA EL AJUSTE DE LAS RECTAS DE LOS LADOS DEL POLÍGONO REGULAR PARCIALMENTE OCULTO.

AJUSTE DE CURVAS. MÉTODO DE LOS MÍNIMOS CUADRADOS.

Se utilizó el método de los mínimos cuadrados con el fin de ajustar las rectas correspondientes a cada uno de los lados del polígono regular parcialmente oculto.

A continuación una breve explicación del método.

Mínimos cuadrados es una técnica de análisis numérico encuadrada dentro de la optimización matemática, en la que, dados un conjunto de pares (o ternas, etc), se intenta encontrar la función que mejor se aproxime a los datos (un "mejor ajuste"), de acuerdo con el criterio de mínimo error cuadrático (Acosta, 1981).

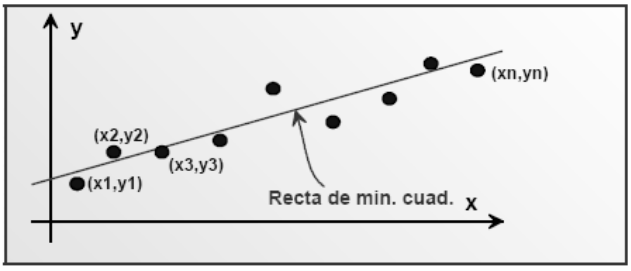


Figura 2.9. Recta de mínimos cuadrados.

El método de los mínimos cuadrados da la mejor recta para ajustar una data, y además implica una única solución.

Para el caso del ajuste mediante una recta, la suma de los cuadrados de las desviaciones puede expresarse como:

$$\sum (y_c - y_0)^2 = \sum (a + bx_0 - y_0)^2$$

Los valores de las constantes *a* y *b* para los cuales la expresión anterior es mínima, pueden determinarse al establecer las derivadas parciales con respecto a cada cantidad, o sea:

Para *a*:

$$\frac{\partial}{\partial a}\sum (a + bx_0 - y_0)^2 = \frac{\partial}{\partial a}\sum (a^2 + b^2 x_0^{\,2} + y_o^{\,2} + 2abx_0 - 2ay_0 - 2bx_0 y_0) = \sum (2a + 2bx_0 - 2y_0)$$

Igualando la expresión anterior a cero, se obtiene:

$$\frac{\partial}{\partial a}\sum (a + bx_0 - y_0)^2 = 2\sum (a + bx_0 - y_0) = 0$$

O sea, $\sum (a + bx_0 - y_0) = 0$ (1)

Para b:

$$\frac{\partial}{\partial b}\sum (a + bx_0 - y_0)^2 = \sum (2bx_0^{\,2} + 2ax_0 - 2x_0 y_0) = 2\sum (bx_0^{\,2} + ax_0 - x_0 y_0)$$

Igualando la expresión anterior a cero, se obtiene:

$$\frac{\partial}{\partial b}\sum (a + bx_0 - y_0)^2 = 2\sum (bx_0^{\,2} + ax_0 - x_0 y_0) = 0$$

O sea, $\sum (bx_0 + bx_0^{\,2} - x_0 y_0) = 0$ (2)

Las ecuaciones (1) y (2) pueden escribirse como:

$$\sum y_0 = \sum a + \sum bx_0$$

$$\sum x_0 y_0 = \sum ax_0 + \sum bx_0^{\,2}$$

Finalmente considerando un número n de puntos:

$$\sum y_0 = na + b\sum x_0 \quad (1')$$

$$\sum x_0 y_0 = a\sum x_0 + b\sum x_0^{\,2} \quad (2')$$

Esquema del procedimiento que se sigue en la búsqueda de la mejor recta por el método de los mínimos cuadrados.

Valores $\begin{cases} x = x_0 \ : \ x_1 \quad x_2 \quad \cdots x_n \\ \\ \end{cases}$

observados $y = y_0 \ : \ y_1 \quad y_2 \quad \cdots y_n$

$$\begin{array}{cccc} x_1 & y_1 & x_1 y_1 & {x_1}^2 \\ x_2 & y_2 & x_2 y_2 & {x_2}^2 \\ x_3 & y_3 & x_3 y_3 & {x_3}^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & x_n y_n & {x_n}^2 \\ \hline \sum x_0 & \sum y_0 & \sum x_0 y_0 & \sum {x_0}^2 \end{array}$$

Sustituyendo los valores en las ecuaciones (1´) y (2´) se obtienen los valores de las constantes a y b:

$$a = \frac{\left(\sum y_0\right)\left(\sum {x_0}^2\right) - \left(\sum x_0\right)\left(\sum x_0 y_0\right)}{n\sum {x_0}^2 - \left(\sum x_0\right)^2}$$

$$b = \frac{n\sum x_0 y_0 - \left(\sum x_0\right)\left(\sum y_0\right)}{n\sum {x_0}^2 - \left(\sum x_0\right)^2}$$

Estos valores sustituidos en la ecuación $y = a + bx$ darán la mejor recta que ajusta los datos.

El método de los mínimos cuadrados no depende de la apreciación visual, ni del agrupamiento, o sea de la forma de agrupar los datos, la precisión del mismo está limitada solamente por la data original.

ALGORITMO PARA EL AJUSTE DE LAS RECTAS .

A partir del código de cadena del contorno del polígono regular parcialmente oculto:

1ro: Dividir el código de cadena del contorno, en el código de cadena perteneciente a cada lado del polígono parcialmente oculto.

2do: Detectar cuales son los lados del polígono parcialmente oculto.

3ro: Para cada lado del polígono parcialmente oculto determinar la pendiente de la recta y la longitud de la misma.

Para el caso de imágenes digitales en escala de grises recordar que la imagen se almacena en forma de matriz donde cada elemento de la misma corresponde al nivel de gris del píxel correspondiente, y además se considera la variación de las coordenadas X y Y como se muestra en la figura 2.10.

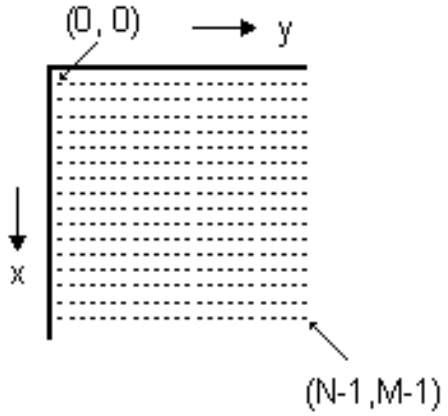


Figura 2.10. Variación de las coordenadas de los píxeles en una imagen de tamaño NxM.

A continuación se explica en detalle cada paso del algoritmo:

Paso1: Dividir el código de cadena del contorno, en el código correspondiente a cada lado del polígono regular parcialmente oculto.

Una vez recorrido el contorno del polígono regular parcialmente oculto y obtenido su código de cadena, para ajustar la recta de cada uno de los lados del mismo es necesario dividirlo en códigos de cadena que identifiquen a cada lado del polígono y poder ajustar de forma independiente cada recta de cada lado.

Para dividir el código de cadena del contorno se tuvo en cuenta las características de la representación de rectas digitales (epígrafe 2.2).

El algoritmo consiste en:

A partir del código de cadena del contorno:

1. Almacenar los 2 primeros elementos del código de cadena del contorno.

Este paso es necesario porque el código de cadena de las rectas pueden tener solamente 1 o 2 segmentos en dependencia de su orientación.

2. Hallar la diferencia entre cada uno de los elementos almacenados y elementos en el código de cadena del contorno a partir del 3ro, considerando:

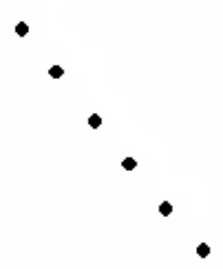
Elemento (i): Elemento actual del código de cadena del contorno.

Elemento (i+1): Elemento siguiente al actual del código de cadena del contorno.

La pertenencia del elemento del código de cadena de contorno (i) al código de cadena de un lado del polígono es verdadera si se cumple alguna de las siguientes condiciones:

- La diferencia entre el primer elemento almacenado y cada uno de los elementos (i) e (i+1) es igual a cero.

  Ejemplo 1: Suponer la posición de los píxeles:



  Código de cadena: 1 1 1 1 1 (a_0 a_1 a_2 a_3 a_4)
  Elementos almacenados: 1 1 (a_0 a_1)
  El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:
  a_0-a_2=0
  a_0-a_3=0
  De esta forma se analiza para cada elemento del código de cadena.
  Conclusión: desde a_0 hasta a_4 pertenecen al mismo lado.

  - El valor absoluto de la diferencia entre el primer elemento almacenado y cada uno de los elementos (i) e (i+1) es igual a 0 o 1 y el valor absoluto de la diferencia entre el segundo elemento almacenado y cada uno de los elementos (i) e (i+1) es igual a 0 o 1. El valor absoluto de la diferencia entre el elemento (i) o (i+1) con respecto al mismo elemento almacenado solo puede ser 1 siempre y cuando el valor absoluto de la diferencia entre el elemento (i) o (i+1) con respecto al otro elemento almacenado sea cero.

  Ejemplo 2: Suponer la posición de los píxeles

  Código de cadena: 3 2 2 3 2 (a_0 a_1 a_2 a_3 a_4)
  Elementos almacenados: 3 2 (a_0 a_1)

  El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

  $|a_0-a_2|=1$ $|a_1-a_2|=0$

  $|a_0-a_3|=0$ $|a_1-a_3|=1$

  De esta forma se analiza para cada elemento del código de cadena.

  Conclusión: desde a_0 hasta a_4 pertenecen al mismo lado.

  Ejemplo 3: Suponer la posición de los píxeles:

  Código de cadena: 1 0 0 0 1 (a_0 a_1 a_2 a_3 a_4)

  Elementos almacenados: 1 0 (a_0 a_1)

  El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

  $|a_0-a_2|=1$ $|a_1-a_2|=0$ $|a_0-a_3|=1$ $|a_1-a_3|=0$

De esta forma se analiza para cada elemento del código de cadena

Conclusión: desde a_0 hasta a_4 pertenecen al mismo lado.

- El valor absoluto de la diferencia entre cada uno de los elementos almacenados y el mismo elemento (i) o (i+1) es igual a 1 y el valor absoluto de la diferencia entre cada uno de los elementos almacenados y el otro elemento (i) o (i+1) es igual a cero.

  Ejemplo 4: Suponer la posición de los píxeles.

  Código de cadena: 4 4 3 4 3 (a_0 a_1 a_2 a_3 a_4)

  Elementos almacenados: 4 4 (a_0 a_1)

  El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

  $|a_0-a_2|=1$ $|a_1-a_2|=1$ $|a_0-a_3|=0$ $|a_1-a_3|=0$

  De esta forma se analiza para cada elemento del código de cadena

  Conclusión: desde a_0 hasta a_4 pertenecen al mismo lado.

- El valor absoluto de la diferencia entre el primer elemento almacenado y cada uno de los elementos (i) e (i+1) es igual a 0 o 7 y el valor absoluto de la diferencia entre el segundo elemento almacenado y cada uno de los elementos (i) e (i+1) es igual a 0 o 7. El valor absoluto de la diferencia entre el elemento (i) o (i+1) con respecto al mismo elemento almacenado solo puede ser 7 siempre y cuando el valor absoluto de la diferencia entre el elemento (i) o (i+1) con respecto al otro elemento almacenado sea cero.

  Ejemplo 5: Suponer la posición de los píxeles.

  Código de cadena: 0 7 7 0 7 (a_0 a_1 a_2 a_3 a_4)

  Elementos almacenados: 0 7 (a_0 a_1)

  El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

  $|a_0-a_2|=7$ $|a_1-a_2|=0$ $|a_0-a_3|=0$ $|a_1-a_3|=7$

  De esta forma se analiza para cada elemento del código de cadena

  Conclusión: desde a_0 hasta a_4 pertenecen al mismo lado.

Ejemplo 6: Suponer la posición de los píxeles.

Código de cadena: 7 0 0 0 7 (a_0 a_1 a_2 a_3 a_4)

Elementos almacenados: 7 0 (a_0 a_1)

El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

$|a_0-a_2|=7$ $|a_1-a_2|=0$ $|a_0-a_3|=7$ $|a_1-a_3|=0$

De esta forma se analiza para cada elemento del código de cadena.

Conclusión: desde a_0 hasta a_4 pertenecen al mismo lado.

- El valor absoluto de la diferencia entre cada uno de los elementos almacenados y el mismo elemento (i) o (i+1) es igual a 7 y el valor absoluto de la diferencia entre cada uno de los elementos almacenados y el otro elemento (i) o (i+1) es igual a cero.

Ejemplo 7: Suponer la posición de los píxeles.

Código de cadena: 7 7 7 0 7 (a_0 a_1 a_2 a_3 a_4)

Elementos almacenados: 7 7 (a_0 a_1)

El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

$|a_0-a_2|=0$ $|a_1-a_2|=0$ $|a_0-a_3|=7$ $|a_1-a_3|=7$

De esta forma se analiza para cada elemento del código de cadena.

Conclusión: desde a_0 hasta a_4 pertenecen al mismo lado.

En caso que no se cumplan las condiciones anteriores no quiere decir que el elemento (i) del código de cadena del contorno no pertenezca al lado, es sólo información de que en caso de pertenecer, hasta ese elemento llega el código de cadena del lado. El elemento (i+1) forma parte del código de cadena de otro lado y las coordenadas del píxel inicial del nuevo lado o final del lado anterior son las mismas y se determinan utilizando el elemento (i) del código de cadena que corresponde a la dirección (a_{i-1}) en la que fue encontrado el píxel actual (p).

Para determinar si el elemento (i) pertenece al código de cadena del lado cuando el elemento (i+1) no pertenece, debe de cumplirse alguna de las siguientes condiciones:

- La diferencia entre el elemento (i) y cada uno de los elementos almacenados sea cero.

Ejemplo 8: Suponer la posición de los píxeles.

Código de cadena: 2 2 2 4 4 4 (a_0 a_1 a_2 a_3 a_4 a_5)

Elementos almacenados: 2 2 (a_0 a_1)

El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

$|a_0-a_2|=0$ $|a_1-a_2|=0$

De esta forma se analiza para cada elemento del código de cadena

Conclusión: desde a_0 hasta a_2 pertenecen al mismo lado, desde a_3 hasta a_5 pertenecen a otro lado.

- El valor absoluto de la diferencia entre el elemento (i) y cada uno de los elementos almacenados sea 0 y 1 o 0 y 7 .

Ejemplo 9: Suponer la posición de los píxeles.

Código de cadena: 2 1 1 3 3 3 (a_0 a_1 a_2 a_3 a_4 a_5)

Elementos almacenados: 2 1 (a_0 a_1)

El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

$|a_0-a_2|=1$

$|a_1-a_2|=0$

De esta forma se analiza para cada elemento del código de cadena.

Conclusión: desde a_0 hasta a_2 pertenecen al mismo lado, desde a_3 hasta a_5 pertenecen a otro lado.

Ejemplo 10: Suponer la posición de los píxeles.

Código de cadena: 7 0 7 6 6 6 (a_0 a_1 a_2 a_3 a_4 a_5)

Elementos almacenados: 7 0 (a_0 a_1)

El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

$|a_0-a_2|=0$

$|a_1-a_2|=7$

De esta forma se analiza para cada elemento del código de cadena.

Conclusión: desde a_0 hasta a_2 pertenecen al mismo lado, desde a_3 hasta a_5 pertenecen a otro lado.

Puede ocurrir que al almacenar los dos primeros elementos del código de cadena estos sean iguales, y al desarrollar el algoritmo aparezca un elemento diferente a los dos almacenados que pertenezca al código de cadena del lado, en este caso se reemplaza el 2do elemento almacenado por el nuevo encontrado y el algoritmo continua.

Ejemplo 11: Suponer la posición de los píxeles.

Código de cadena: 0 0 1 0 0 (a_0 a_1 a_2 a_3 a_4)

Elementos almacenados: 0 0 (a_0 a_1)

El elemento a_2 pertenece al mismo lado que a_0 y a_1 pues:

$|a_0-a_2|=1$ $|a_1-a_2|=1$ $|a_0-a_3|=0$ $|a_1-a_3|=0$

De esta forma se analiza para cada elemento del código de cadena.

Conclusión: desde a_0 hasta a_2 pertenecen al mismo lado. Por tanto, el segundo elemento almacenado se reemplaza por 1.

Paso 2: Detectar cuales son los lados del polígono regular parcialmente oculto.
Considerar los casos que se muestran en la siguiente figura:

Figura 2.11. Polígonos regulares parcialmente ocultos por otros polígonos regulares.

Las imágenes creadas reflejan un polígono regular que está oculto parcialmente por otro polígono regular. En el primer caso se trata de un triángulo que está parcialmente oculto por un cuadrado, el segundo caso un triángulo parcialmente oculto por otro triángulo, el tercer caso un cuadrado oculto por un pentágono y por último un octágono oculto por un hexágono.

El polígono que oculta puede tener un nivel de gris cualquiera con valor entre 1 y 14 (recordar que se está trabajando con imágenes en escala de gris con 16 niveles (entre 0 y 15)).

El problema radica en que luego de haber dividido el código de cadena del contorno del objeto (polígono regular parcialmente oculto) en códigos que representan la conexión de los píxeles de cada

lado, saber identificar cuales son los lados del polígono parcialmente oculto, de forma tal que se pueda desechar información no útil del objeto (lados que pertenecen a la frontera entre los dos polígonos).

Para identificar si un lado del contorno es o no parte del polígono regular parcialmente oculto se calculó la diferencia entre los niveles de gris de cada píxel perteneciente a un lado determinado y 4 de sus vecinos.

Suponer el píxel de análisis del lado con coordenadas (x, y) y los cuatro vecinos con coordenadas: (x, y+1), (x+1, y), (x, y-1), (x-1, y).

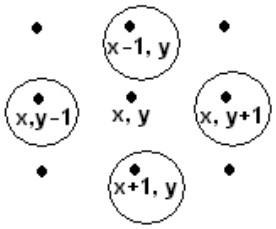
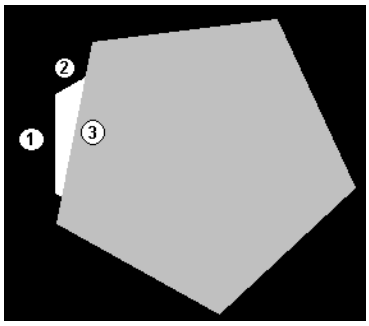


Figura 2.12. Cuatro vecinos del píxel de coordenadas (x, y).

No necesariamente la comparación tiene que hacerse con estos 4 vecinos, se pudo haber realizado con otros, la ventaja de hacerlo por esta vía es que siempre que se está analizando un lado del polígono regular parcialmente oculto que no pertenezca a la frontera con el polígono que lo oculta, la diferencia entre los niveles de gris del píxel de análisis y los cuatro vecinos mostrados en la figura 2.12 será de 0 o 15, ya que algunos de estos vecinos pueden formar parte del fondo de la imagen (color negro igual a cero) o formar parte del polígono parcialmente oculto (color blanco igual a 15). Además en caso que se esté en presencia del análisis de un lado que pertenezca a la frontera de los dos polígonos, como máximo 2 de los cuatro vecinos del píxel de análisis tendrán un nivel de gris entre 1 y 14, pero los otros vecinos tienen nivel de gris de 0 o 15.

El algoritmo está basado en la implementación de la idea expuesta, y realiza el análisis de la siguiente manera: Analizar para cada píxel de cada lado del contorno la diferencia entre los niveles de gris de él y los cuatro vecinos mencionados anteriormente. Si se cumple que todos los píxeles que componen el lado o la mayoría excepto dos como máximo, tienen los cuatro o al menos dos vecinos con niveles de gris de 0 o 15, entonces el lado pertenece al polígono parcialmente oculto. En caso de que no se cumpla esta condición el lado de análisis simplemente no pertenece al lado del polígono que se desea reconocer, y forma parte de la frontera entre ambos polígonos.



1 y 2: Lados del polígono regular parcialmente oculto.
3: Frontera de ambos polígonos.

Figura 2.13. Lados del polígono regular parcialmente oculto.

Paso 3: Para cada lado del polígono determinar la pendiente de la recta y la longitud de la misma.

El ajuste de cada recta de los lados del polígono se realizó siguiendo el mismo procedimiento del método de los mínimos cuadrados explicado anteriormente, con la única diferencia que en la imagen la variación de la abcisa del píxel corresponde a la variación de Y en el método, y la variación de la ordenada del píxel con la variación de X.

El cálculo de cada uno de los parámetros para el ajuste de cada recta que corresponde a los lados del polígono se realiza de acuerdo al procedimiento que se muestra en la figura 2.14, y consiste en que a partir de las coordenadas del píxel inicial de cada lado del polígono y el código de cadena que contiene información de la posición de un píxel respecto al otro obtener las coordenadas de cada píxel. De esta forma se hallan los parámetros de cada recta sin tener información previa de la posición de cada píxel.

Es importante para lograr un buen ajuste de la recta que se calcule con exactitud las coordenadas del píxel inicial de cada lado del polígono y las coordenadas de cada píxel que compone el lado. En la figura 2.14 se muestra como varían la ordenada y abcisa de cada píxel teniendo como información el código de cadena del lado. Las coordenadas del píxel inicial de un lado coinciden con las coordenadas del píxel final del lado anterior.

En el algoritmo se realiza el ajuste de las rectas de los lados que sean información útil del polígono regular parcialmente oculto, es decir, la información brindada en el paso 3 se utiliza para el ajuste solamente de los lados del polígono de interés, lo que garantiza rapidez en la ejecución del algoritmo. Aunque no se realiza el ajuste de los lados que no pertenecen al polígono regular parcialmente oculto, si es necesario recorrer estos lados pues sólo de esta forma se puede determinar las coordenadas del píxel inicial de aquellos lados que pertenezcan al polígono de interés. En caso de no realizarse lo anteriormente explicado, no se podrá tener exactamente las coordenadas del píxel inicial de los lados del polígono regular parcialmente oculto.

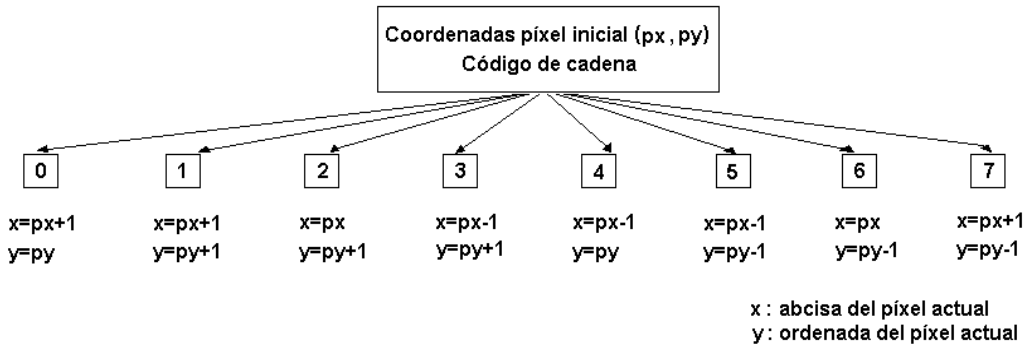


Figura 14. Esquema para la determinación de las coordenadas de los píxeles a partir del código de cadena.

El algoritmo además de ganar en rapidez por lo explicado anteriormente, de forma simultánea calcula la longitud del lado correspondiente.

Para el cálculo de la longitud de cada lado se consideran los píxeles cuadrados, es decir tienen igual ancho que largo y por lo tanto la longitud de la recta se calcula mediante la expresión:

Teniendo en código de cadena de la recta, la longitud de la misma es:

$$L = \#\,segmentos\ verticales\ y\ horizontales + \sqrt{2}.\#\,segmentos\ diagonales$$

Donde:

Segmentos verticales: 2 y 6

Segmentos horizontales: 0 y 4

Segmentos diagonales: 1, 3, 5 y 7

Recordar que cuando se está en presencia de rectas verticales la pendiente de las mismas son infinitas, es decir, la orientación de las rectas es de ±90°. En este tipo de rectas el código de cadena puede ser de la forma: 2222222… ó 6666666… (ver tabla 2.1).

Entonces es muy fácil saber la orientación de algunas rectas conociendo su código de cadena sin necesidad de calcular su pendiente por el método de los mínimos cuadrados, ganando en rapidez la ejecución del algoritmo implementado.

Estas rectas se resumen de la tabla 2.1 y pueden ser:

Código de cadena	Orientación de la recta
000000000000000000	Recta de 0°
444444444444444444	Recta de 180°
222222222222222222	Recta de -90°
666666666666666666	Recta de 90°
777777777777777777	Recta de 45°
333333333333333333	Recta de -135°
111111111111111111	Recta de -45°
555555555555555555	Recta de 135°

Tabla 2.4. Código de cadena de rectas con orientación de ±45kº.

En el algoritmo para cada recta correspondiente a cada lado del polígono se calcula primeramente la media de los elementos del código de cadena, y luego se compara con los valores de 2, 6, 0, 4, 3, 7, 1 y 5 de forma tal que si coincide con algunos de estos valores entonces directamente se sabe la orientación de la recta sin necesidad de calcular la pendiente de la misma por el método de los mínimos cuadrados. Recordar lo explicado en relación al sentido de recorrido y el código de cadena en el epígrafe 2.2.

En el caso que el código de cadena de una recta contenga segmentos 0 y 7, el elemento 0 correspondería a un elemento 8 que no existiría en el código de cadena pues se está trabajando con la 8 vecindad del píxel. En este caso solamente para el cálculo de la media de la recta se sustituyen los elementos 0 por 8. Para las restantes tareas del algoritmo se trabaja con los elementos del código de cadena obtenido inicialmente.

De esta manera se ajustarán sólo las rectas utilizando el método de los mínimos cuadrados que tengan la media del código de cadena diferente a los casos mencionados anteriormente en la tabla 4.

2.2.3 ALGORITMO PARA EL RECONOCIMIENTO DEL POLÍGONO REGULAR PARCIALMENTE OCULTO.

Paso1: Determinar el ángulo interior del polígono regular parcialmente oculto.

Como se analizó en el capítulo 1 epígrafe 1.1.2, los polígonos regulares presentan todos sus ángulos interiores iguales, sus lados iguales y para cada tipo de polígono el ángulo interior es diferente. Por lo que para reconocer un polígono regular basta tener información de un ángulo interno.

En caso que no se tenga información del ángulo interno se puede reconocer el polígono regular teniendo información de un lado completo y parte del lado contiguo, de forma tal que se pueda determinar el ángulo comprendido entre estos dos lados.

ÁNGULO FORMADO ENTRE DOS RECTAS.

Sean l_1 y l_2 dos rectas no verticales, cuyos ángulos de inclinación son θ_1 y θ_2 respectivamente.

Al cortarse las rectas l_1 y l_2 forman cuatro ángulos iguales de dos en dos (figura 2.15), es decir:

$$\beta_1 = \beta_2 = \theta_1 - \theta_2 \quad y \quad \alpha_1 = \alpha_2 = 180^{\circ} - \beta_1$$

Como se conoce el ángulo entre l_1 y l_2 como el ángulo positivo obtenido al rotar la recta l_2 hacia l_1.

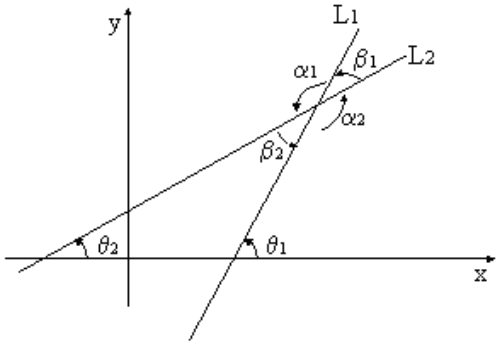


Figura 2.15. Ángulos formados entre las rectas L_1 y L_2.

En este caso, el ángulo entre l_1 y l_2 viene dado por:

$$\beta_1 = \theta_1 - \theta_2$$

El propósito ahora es establecer una relación entre las pendientes de las dos rectas y el ángulo entre ellas.

De la igualdad anterior se tiene:

$$\tan \beta_1 = \tan(\theta_1 - \theta_2)$$

$$\tan \beta_1 = \frac{\tan \theta_1 - \tan \theta_2}{1 + \tan \theta_1 \tan \theta_2}$$

Puesto que $m_1 = \tan \theta_1$ y $m_2 = \tan \theta_2$, entonces la ecuación anterior se puede escribir en la forma:

$$\tan \beta_1 = \frac{m_1 - m_2}{1 + m_1 m_2}$$ donde: m_1: pendiente de la recta L_1

m_2: pendiente de la recta L_2

De esta manera el cálculo de un ángulo interior se puede realizar de dos formas:

1. Conociendo la pendiente de las rectas de dos lados consecutivos, el ángulo entre estas rectas está dado por la ecuación $\beta_1 = \tan^{-1} \frac{m_1 - m_2}{1 + m_1 m_2}$.
2. Conociendo la orientación de las rectas de dos lados consecutivos, el ángulo comprendido entre ellas está dado por la ecuación $\beta_1 = \theta_1 - \theta_2$.

Ambas variantes se pueden utilizar pues son equivalentes.

El problema que presenta la primera variante en que si alguna de las rectas tiene pendiente infinito (recta vertical) el ángulo calculado puede no estar correcto y por lo tanto fallar el algoritmo.

La variante implementada en el algoritmo fue la segunda ya que como se analizó anteriormente conociendo el código de cadena de algunas rectas se puede saber la orientación de las mismas y por lo tanto el proceso de hallar el ángulo formado entre las dos rectas consecutivas es más rápido. En caso que no se trate de ninguno de estos tipos de rectas, entonces mediante la siguiente ecuación se puede determinar el ángulo de orientación de las mismas (en grados) a partir de la pendiente de la misma:

$$\theta = \frac{\tan^{-1}(m) * 180}{\pi}$$ donde m: pendiente de la recta.

¿Entre cuales lados se busca el ángulo interior del polígono regular parcialmente oculto?

Una vez determinados cuales son los lados del polígono parcialmente oculto, entonces con dos de ellos cualesquiera pero que sean consecutivos se puede buscar el ángulo interior del polígono.

El problema es que se tiene que garantizar en el algoritmo que el mismo sea capaz de hallar el ángulo interior del polígono independientemente de la cantidad de información conocida, es decir, calcular de forma correcta el ángulo interior del polígono aún cuando sean conocidos dos, tres, cuatro, etc. lados del mismo.

Como se mencionó anteriormente para conocer el ángulo interior de un polígono regular basta con conocer un lado completo y parte de otro consecutivo, de forma tal que la información parcial del segundo lado sea suficiente para conocer con exactitud la orientación de la recta de ese lado. Sin embargo puede darse el caso que en vez de tener la información de dos lados consecutivos, se tenga de tres o más, y en este caso el algoritmo debe ser capaz de calcular el ángulo interior del polígono y llegar a resultados positivos.

Se establece la condición de conocer un lado completo pues el mismo permite saber el tamaño y otros parámetros que se pueden conocer del polígono. Aunque en el desarrollo de este trabajo no se aborda el tema de la reconstrucción del objeto, sí se prevee una futura implementación para este fin, además que existen aplicaciones en las cuales se requiere conocer el área que ocupa el objeto, así como su perímetro u otras propiedades del mismo y por lo tanto es importante saber la longitud de cada lado del mismo pues una de las características de los polígonos regulares es que precisamente todos sus lados son iguales y conociendo uno de ellos se tiene información de todos los restantes.

En caso de conocerse un lado completo y parte del otro consecutivo, el algoritmo trabaja directamente en la búsqueda del ángulo formado entre estos dos lados.

Ahora, suponer que se tiene el siguiente caso:

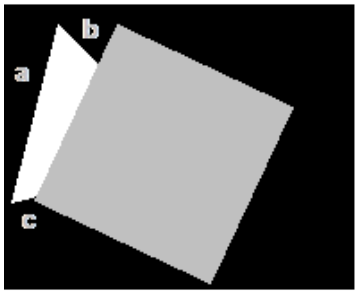


Figura 2.16. Triángulo parcialmente oculto por un cuadrado.

Se trata de un cuadrado ocultando parcialmente un triángulo. Notar que se tiene información de un lado completo y otros dos lados ambos consecutivos al primero.

En este caso el algoritmo realiza lo siguiente:

Luego de conocer cuáles son los lados pertenecientes al polígono regular parcialmente oculto, busca el lado de mayor longitud y además la longitud de los otros lados conocidos y que son consecutivos al lado de mayor longitud. El ángulo se calcula hallando la diferencia de las orientaciones del lado de mayor longitud y la del lado de mayor longitud entre los dos lados consecutivos a él. Para el caso anterior el ángulo se calcula mediante la diferencia en las orientaciones del lado a y b.

Este análisis se realiza para cualquier caso en el que se tenga información superior a la mínima necesaria para reconocer un polígono regular (un lado completo y parte del otro consecutivo).

Paso2: Reconocer el tipo de polígono regular parcialmente oculto.

Una vez calculado el ángulo interior del polígono regular parcialmente oculto, simplemente para identificar el tipo de polígono regular sólo hay que comparar con los diferentes valores de los ángulos de cada uno de los polígonos regulares.

Al trabajar con imágenes digitales y teniendo en cuenta los problemas generados en la digitalización, es necesario considerar cierto margen de variación en el valor de los ángulos interiores de los polígonos regulares.

Se tomó un margen de error para el reconocimiento de los polígonos regulares que garantizó que la variación en los ángulos interiores no sobrepase ±2º, es decir:

- Se trata de un triángulo si el ángulo interno se encuentra en el rango de 58º a 62º.
- Se trata de un cuadrado si el ángulo interno se encuentra en el rango de 88º a 92º.
- Se trata de un pentágono si el ángulo interno se encuentra en el rango de 106º a 110º.
- Se trata de un hexágono si el ángulo interno se encuentra en el rango de 118º a 122º.
- Se trata de un heptágono si el ángulo interno se encuentra en el rango de 126º a 130º.
- Se trata de un octágono si el ángulo interno se encuentra en el rango de 133º a 137º.

2.3 CORROBORACIÓN DE LOS RESULTADOS.

De forma general el algoritmo desarrollado se comprobó en aproximadamente 50 imágenes digitales en escala de grises de 16 niveles. En todas el desempeño del algoritmo fue satisfactorio. Las imágenes fueron creadas por la autora del trabajo utilizando el paquete profesional AutoCad 2007. Se crearon imágenes en las cuales estaban presentes objetos poligonales regulares hasta 8 lados, los cuales estaban parcialmente ocultos por otros polígonos, regulares y no. Se consideraron objetos

poligonales regulares en diferentes posiciones, orientaciones, tamaños, es decir, comprobar la eficiencia del algoritmo ante variaciones de escala, rotaciones y traslaciones.

Para demostrar el desempeño del algoritmo desarrollado se mostrará a través de un ejemplo cada uno de los métodos implementados con sus resultados, considerando en este caso que el polígono que oculta no es regular. Toda la simulación se realizó con el paquete Matlab 7.0.

Ejemplo 12: Considerar la siguiente imagen digital de 16 niveles de gris en la cual se muestra un rectángulo ocultando parcialmente un pentágono. El objetivo es el reconocimiento del polígono regular parcialmente oculto (pentágono).

Figura 2.17. Rectángulo ocultando parcialmente un pentágono.

- Función para cargar la imagen.

  % Leer imagen

  imagen=imread('5ladosrect.bmp');

  imshow (imagen)

  Esta función tiene como entrada la imagen almacenada en el fichero **5ladosrect.bmp** y como salida una variable **imagen** matricial que contiene en sus elementos del valor del nivel de gris correspondiente a cada píxel de la imagen. Se muestra la imagen a través del comando **imshow (imagen)**.

- Función para el seguimiento del contorno y obtención del código de cadena del mismo. (Anexo 2)

  %**Algoritmo de seguimiento del contorno del polígono regular**

[inii,inij,codigo,imagen1] = seguimiento_contorno(imagen);

Figura 2.18. Marcado del contorno del polígono regular parcialmente oculto.

Como se observa en este paso del algoritmo, se recorre el contorno del polígono regular parcialmente oculto, y se marca con un nivel de gris igual a 7. En esta función primero se determina el píxel inicial del contorno y a partir de él se recorre el contorno. El mismo concluye cuando se llega al píxel inicial que ya está marcado con el nivel de gris igual a 7.

- Función para determinar los lados del polígono. (Anexo 3)

  [codigo1,codigo2,codigo3,codigo4,codigo5,codigo6,codigo7,codigo8,codigo9]=lados_del_poligono(codigo);

  En esta función se tiene como entrada el código de cadena del contorno, y como salida el código correspondiente a cada lado del polígono. Se reservan hasta 9 variables para almacenar el código de cadena de los lados del polígono regular pues puede darse el caso que el polígono que oculta aporte un lado más al contorno del polígono regular.

- Función para determinar el ángulo interior del polígono regular (Anexo 4).

  teta=angulo_entre_dos_lados(imagen,inii,inij,codigo1,codigo2,codigo3,codigo4,codigo5,codigo6,codigo7,codigo8, codigo9)

  teta = 109.1498 grados

Esta función tiene como entrada la imagen de análisis, las coordenadas del píxel inicial del contorno y las variables que contienen los códigos de cadena de cada lado del polígono regular parcialmente oculto; como salida el valor del ángulo interior del polígono. En esta función se determinan cuales son los lados del polígono regular parcialmente oculto, se realiza el ajuste de las rectas de estos lados y por último se calcula el valor del ángulo interior del polígono.

- Función para reconocer el tipo de polígono regular.

  El polígono es un pentágono.

  En dependencia del valor del ángulo interior calculado en la función anterior se reconoce el tipo de polígono regular parcialmente oculto. (Anexo 5).

2.4 VALORACIONES DE EXPERTOS.

2.4.1 CARACTERIZACIÓN DE LOS EXPERTOS.

	Título de graduado	**Años de experiencias**	**Grado científico o académico**	**Cargo**	**Conocedor del tema**
experto 1	Ingeniero Electricista	50 años.	Dr	Coordinador de la maestría en Automática de la UCLV.	si
experto 2	Ingeniero en Control Automático	29 años.	Dr	Jefe del Grupo de Automática, Robótica y Percepción (GARP) de la UCLV.	Si
experto 3	Ingeniero en Automática	5 años	MSc		Si
experto 4	Ingeniero en Telecomunicaciones y Electrónica. Licenciado en Ciencias de la Computación.	9 años	**-**		Si

2.4.2 VALORACIONES DE LOS EXPERTOS EMITIDAS CON RESPECTO A LA PROPUESTA DE INVESTIGACIÓN.

Los expertos coinciden en que el trabajo se desarrolla en una temática de gran actualidad al abordar el tema de reconocimiento de objetos en imágenes de ambientes no estructurados, en este caso en el ámbito de figuras planas poligonales regulares de diferentes escalas de grises. El reconocimiento de objetos es una temática muy estudiada en las últimas décadas motivadas por aplicaciones en diferentes campos de la ciencia y la tecnología.

El desarrollar un algoritmo para el reconocimiento de objetos poligonales regulares parcialmente ocultos por la vía desarrollada es una buena idea, sobretodo porque no se requiere mucho tiempo ni memoria para el procesamiento de la información, lo que lo hace ideal para el trabajo en tiempo real, a diferencia de las técnicas basadas en la comparación con objetos ya establecidos en bases de datos, las cuales requieren de un alto equipamiento con apreciable capacidad de memoria y rapidez en sus aplicaciones.

Entre los aspectos novedosos propuestos por la autora se puede mencionar que el algoritmo de seguimiento de contorno es aplicable en la detección y obtención del código de cadena de cualquier objeto de contorno cerrado, no necesariamente para casos de polígonos. El desarrollo de un algoritmo para dividir el código de cadena del contorno de un polígono regular en códigos que identifiquen cada lado de estos tipos de objetos. La utilización del método de mínimos cuadrados para ajustar rectas partiendo del código de cadenas.

Sin embargo plantean la posibilidad futura del análisis de las posibles bondades del método propuesto y su comparación con los métodos de reconocimiento a partir de la utilización de bases de datos, teniendo en cuenta las capacidades actuales de memoria y las velocidades de procesamiento de la información.

Recomiendan que el análisis realizado se extienda a otros tipos de objetos, de forma tal que pueda ser utilizado en otras aplicaciones en las cuales no se tenga necesariamente objetos como los estudiados en este trabajo, incluso en la implementación práctica del algoritmo para tareas de integración robot visión.

Las valoraciones de cada experto están recogidas en los anexos 17, 18, 19 y 20.

CONCLUSIONES

1. Se implementó un algoritmo capaz de reconocer objetos poligonales regulares parcialmente ocultos hasta 8 lados, funcionando correctamente siempre y cuando el número total de lados del polígono regular considerando las fronteras con el polígono que lo oculta no sea mayor que 8.
2. El algoritmo desarrollado necesita como información mínima un lado completo y parte del lado consecutivo, pero trabaja correctamente cuando tiene información adicional a la mínima.
3. El algoritmo de seguimiento de contorno puede utilizarse para detectar y obtener el código de cadena de cualquier objeto de contorno cerrado, no necesariamente para casos de polígonos.
4. El algoritmo propuesto es invariante a rotación, traslación y cambio de escala.
5. Los expertos consultados valoraron de positivo el desarrollo del algoritmo y sus posibles aplicaciones.

CONCLUSIONES

En el análisis de la literatura consultada se evidenciaron las potencialidades de la visión por computadora debido a sus múltiples aplicaciones en numerosas esferas de la vida cotidiana y se caracterizaron los objetos poligonales específicamente los regulares que son los de interés para el desarrollo del trabajo, y se establecieron los atributos invariantes que permiten diferenciarlos unos de los otros. Se realizó un estudio profundo de los diferentes métodos existentes para el análisis de imágenes digitales en cuanto a las etapas de segmentación, extracción de características, reconocimiento y localización de objetos que sean visibles totalmente y se analizaron los resultados de diversos trabajos actuales para la detección de objetos parcialmente ocultos o deformados mediante la técnica de comparación de forma con la de objetos almacenados en bases de datos. Se demostró la ventaja de trabajar sólo con información del contorno de los objetos pues le aporta al método desarrollado rapidez y eficiencia.

El análisis efectuado demostró la necesidad de diseñar e implementar un algoritmo capaz de reconocer objetos poligonales regulares parcialmente ocultos en imágenes digitales utilizando atributos invariantes de este tipo de objetos, sin necesidad de utilizar la técnica de comparación de forma, para lo cual se analizaron las particularidades de la representación de rectas digitales de diversas orientaciones y su código de cadena.

El algoritmo desarrollado para el reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes de 16 niveles de gris, está compuesto por un algoritmo de seguimiento de contorno para la obtención del código de cadena del mismo, y puede utilizarse no sólo para objetos poligonales sino que su uso se puede extender a objetos de cualquier contorno cerrado. La obtención del código de cadena del contorno permitió mediante el uso del método de los mínimos cuadrados realizar el ajuste de las rectas de cada lado del polígono para determinar un ángulo interno y con ello el reconocimiento del tipo de polígono regular.

Las simulaciones realizadas demostraron que:

- El algoritmo propuesto es rápido y eficiente desde el punto de vista computacional, lo cual se comprobó mediante el uso de imágenes creadas por la autora del trabajo.

- El algoritmo desarrollado trabaja perfectamente en el reconocimiento de objetos poligonales regulares de hasta 8 lados, independientemente de la cantidad de información útil conocida del polígono regular.

- El algoritmo propuesto trabaja bajo la consideración que el objeto poligonal que oculta puede tener cualquier nivel de gris entre 1 y 14 y puede ser o no regular, pero siempre deja visible sólo una parte del objeto deseado a reconocer.

- El algoritmo propuesto es invariante a rotación, escala y traslación.

- El algoritmo desarrollado está implementado solamente para el reconocimiento de un objeto presente en la imagen.

- Los expertos que valoraron las propuestas corroboraron la factibilidad y pertinencia del algoritmo desarrollado desde el punto de vista científico.

RECOMENDACIONES

Se recomienda en trabajos futuros:

- Extender el análisis realizado a otro tipo de objetos y a imágenes reales.
- Incluir en el algoritmo desarrollado el reconocimiento de más de un objeto parcialmente oculto.
- Utilizar el método propuesto en tareas de integración robot visión, y otras aplicaciones.

BIBLIOGRAFÍA

1. Acosta, J. F. (1981). Ajuste de curvas experimentales. Editorial Oriente. Cuba.
2. Amit, Y. y Kong, A. (1996). “Graphical templates for model registration”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(3), pp 225– 236.
3. Arrebola, F. y col. (2001). Caracterización de contornos mediante segmentación jerárquica del código de cadena. In proceedings of the XVI national symposium of the URSI. pp. 188-189, España. Disponible en: www.iec.csic.es/ursi/articulos_modernos/articulos**2001**/articulos/210.pdf
4. Benet, G. y col. (1998). Desarrollo de sistemas de robótica móvil en entornos de trabajo industriales.
5. Berg, A. C. y col. (2005). “Shape matching and object recognition using low distortion correspondence”. In IEEE Conference on Computer Vision and Pattern Recognition.
6. Browning, B. y Veloso, M. (2005). Real Time, adaptative color – based robot vision. IROS'05. Estados Unidos.
7. Canny, A. (1986). “Computational Approach to Edge Detection”. IEEE Transaction on Pattern Analysis and Machine Intelligence, 8(6), p 679-698.
8. Chen, P.C.Y. y col. (1987). Analysis of microvascular network in bulbar conjunctiva by image processing. Int. J. Microcirc. Clinical and Experimental, 6, pp 245-255.
9. Clemens, D. y Jacobs, D. (1991). “Space and time bounds on indexing 3D models from 2d images”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(10), pp 1007–1017.
10. Esqueda, J. J. (2002). Fundamentos de procesamiento de imágenes. Conatec’ 2002. Ciudad Madero. México.
11. Felzenszwalb, P. F. (2005). “Representation and detection of deformable shapes”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(2) pp 208–220.
12. Ge, F. y col. (2008). “Template- Based object detection through partial shape matching and boundary verification”. International journal of Signal Processing. Vol 4, No 2. Estados Unidos.
13. Gee, J.C. (1999). “On matching brain volumes”. Pattern Recognition, 32(1), pp 99-111.
14. González, R. (1987). Digital Image Processing. Second Edition. Addison-Wesley Publishing Company. Estados Unidos.
15. Gonzalo, M. (2006). Aplicaciones industriales de la visión por computador. [biblioteca virtual en línea]. <http://gavab.escet.urjc.es/recursos/seminario_07.pdf. >. [Consulta: enero 2009]

16. Horáêek, y col. (2005). Recognition of partially occluded and deformed binary objects. Lecture Note and Computer Science. Vol 3691, p 415-422. República Checa.

17. Jaulent, M.C. y col. (1997). “Application of a fuzzy method for the segmentation of renal angiograms”. Proc. 5th European Congress on Intelligent Techniques and Soft Computing in Aachen, vol. 3, pp. 2355-2359. Germany.

18. Jendrysik, F. y col. (1997). “Fuzzy segmentation method applied to the extraction of kidney boundaries in medical images”. Proc. 5th European Congress on Intelligent Techniques and Soft Computing in Aachen, vol. 3, pp. 2360-2364. Germany

19. Jurie, F. y Schmid, C. (2004). “Scale-invariant shape features for recognition of object categories”. In IEEE Conference on Computer Vision and Pattern Recognition II, pp 90–96.

20. Krolupper, F. y Flusser, J. (2007). “Polygonal shape description for recognition of partially occluded objects”. Pattern Recognition Letter, 28, pp 1002-1011.

21. Lamdan, Y. y col. (1988). “Object Recognition by Affine Invariant Matching”. IEEE Conf. Computer Vision and Pattern Recognition, pp 335–344.

22. Lope, J. y Serradilla, J. (1995). Sistema de localización y posicionamiento de piezas utilizando visión artificial. [Biblioteca virtual en línea]. < http://tesis.pucp.edu.pe/files/PUCP000000000094/SISTEMA%20DE%20VISI%F3N%20Artificial%20para%20el%20reconocimiento%20y%20manipulaci%F3n%20de%20objetos%20utilizando%20un%20brazo%20robot.pdf>. [Consulta: mayo 2010].

23. Low, A. (1991). Computer Vision and Image Processing, McGraw-Hill, New York.

24. Marrón, M. y col. (2000). Diseño de un algoritmo de visión para la navegación de un robot móvil en interiores estructurados. España. [Biblioteca virtual en línea]. < http://dspace.uah.es/jspui/bitstream/10017/460/1/Tesis.pdf>. [Consulta: abril 2010].

25. Martín, M. (2002). Técnicas de segmentación de imagen. [Biblioteca virtual en línea]. <http://poseidon.tel.uva.es/~carlos/ltif10001/segmenclasica.pdf.>. [Consulta: febrero 2009]

26. Mazaira, I. (1994). Manipulación de objetos en movimiento con el robot Unimate S-130, asistido por visión. Tesis en opción al grado científico de Maestro en Ciencias. Centro de Investigación y de estudios av”anzados del IPN. México.

27. Mazo, M. (2000). Visión por computador. España. [biblioteca virtual en línea]. <http://193.146.57.132/depeca/repositorio/asignaturas/32367/**vision**6.ppt>. [Consulta: marzo 2009].

28. Metta, G. y Fitzpatrick, P. (2003). “Erly integration of vision and manipulation”. Epigenetic Robotics. Vol. 11 Issue 2, pp. 109 – 128.

29. Mikolajczyk, K. y col. (2003). Shape recognition with edge-based features. In British Machine Vision Conference.

30. Olson, C. F. y Huttenlocher, D. P. (1997). “Automatic target recognition by matching oriented edge pixels”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(1), pp103–113.

31. Ospina, E. y Urrea, J.P. (2004). “Implementación de la transformada de Hough para la detección de líneas para un sistema de visión de bajo nivel”. Scientia et Technica, Año X, No 24. Disponible en: http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/1536779-84.pdf.

32. Pérez, A. y Solís, H. (2005). Métodos para seguimiento de dedo en tiempo real. [Blibioteca virtual en línea]. <http://newton.azc.uam.mx/mcc/01_esp/11_tesis/tesis/Proyectos_investigacion_computacion_2/Informe%20Proyecto%20v7.pdf>. [Consulta: septiembre 2009].

33. Pinto, R. y Sossa, H. (2002). “Localización automática de objetos 2D y 3D en imágenes”. Computación y Sistemas. Número especial, p.26-34. México.

34. Rahman, M. e Ishikawa, S. (2005). A robust recognition method for partially occluded/destroyed objects. Japón. Disponible en: http://users.cecs.anu.edu.au/~masud/ACCV04.pdf

35. Rohr, K. (1999). “Extraction of 3D anatomical point landmarks based on invariance principles”. Pattern Recognition, 32, pp 3-15.

36. Ros, Ll. y Thomas, F. (1991). Percepción activa para la identificación y localización de patrones 2D. Aplicación a la localización de juntas parcialmente solapadas. 2do Congreso de la AER. España.

37. Schmid-Schoenbein, G.W. y col.. (1977). “The application of stereological principles to morphometry of the microcirculation in different tissues”. Microvascular Research, 14, pp 303-317.

38. Schnabel, J.A. y Arridge, S.R. (1999). “Active shape focusing”. Image Vision Computing, 17, pp 419-428.

39. Sclaroff, S. y Liu, L. (2001). “Deformable shape detection and description via model-based region grouping”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(5), pp 475–489.

40. Soto, J. A. y col. (2005). Generación de trayectorias por Visión para un robot manipulador de 5 grados de libertad. 4to Congreso Nacional de Mecatrónica. México.

41. Trucco, E. y Verri, A. (1998). Introductory Techniques for 3-D Computer Vision. Prentice-Hall, Upper Saddle River. NJ.
42. Tsujii, O. y col. (1999). "Classification of microcalcifications in digital mammograms using trend-oriented radial basis function neural network". Pattern Recognition, 32(5), pp 891-903.
43. Umbaugh, S.E. (1998). A Practical approach using CVIPtools. Computer Vision and Image Processing. Prentice-Hall, Englewood Cliffs. NJ.
44. Valverde, J. (2006). Detección de bordes mediante el algoritmo de Canny. [Biblioteca virtual en línea]. <www.jc-info.blogspot.com/.../deteccin-de-**bordes**-**algoritmo-de-canny**.html.>. [Consulta: febrero 2009]
45. Wechsler, H. y Sklansky, J. (1977). "Finding the rib cage in chest radiographs". Pattern Recognition, 9, pp 21-30.
46. Wick, C.E. y col. (1993). Understanding Microvessels in Two and three dimensions. In C.H. Chen, L.F. Pau and P.S.P. Wang (Eds.) Handbook of pattern recognition, pp. 667-693.
47. Windyga, P. y col. (1998). "Estimation of search-space in 3D coronary artery reconstruction using angiographic biplane images". Pattern Recognition Letters, 19(14), pp 1325-1330.
48. Xu, L. y col. (1999). "Segmentation of skin cancer images". Image and Vision Computing, 17, pp 65-74.
49. Yáñez, O. y Azimi, M. R. (1999). "Unsupervised Clustering in Hough Space for Identification of Partially Occluded Objects". Transactions on pattern analysis and machine intelligence. Vol. 21, No. 9.

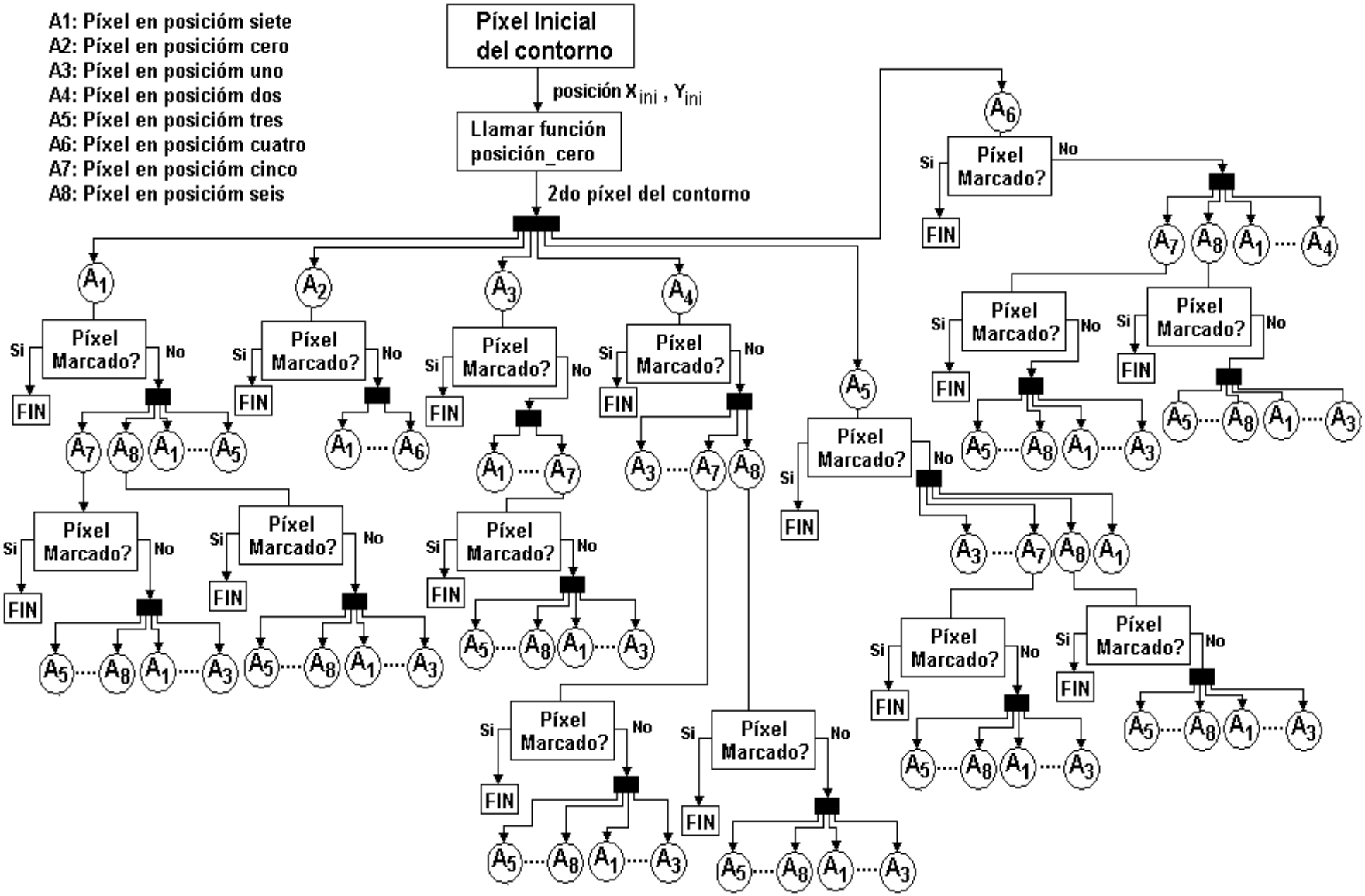
SITIOS CONSULTADOS EN INTERNET:

[1]. http://es.wikipedia.org/wiki/Pol%C3%ADgono_regular. [Consulta: enero 2010].
[2]. http://www.vitutor.com/geo/eso/s_3.html. [Consulta: enero 2010].
[3]. http://es.wikipedia.org/wiki/Pol%C3%ADgono. [Consulta: enero 2010].
[4]. http://www.vitutor.net/2/1/1.html. [Consulta: enero 2010].
[5]. http://huitoto.udea.edu.co/Matematicas/4.5.html. [Consulta: enero 2010].
[6]. http://dsp1.materia.unsl.edu.ar/Representacion%20Descripcion.pdf. [Consulta: noviembre 2009].
[7]. http://campusvirtual.uma.es/tdi/www_netscape/TEMAS/Tdi_12. [Consulta: marzo 2009]
[8]. http://omarsanchez.net/detectdisc.aspx. [Consulta: septiembre 2009].
[9]. http://es.wikipedia.org/wiki/Visi%C3%B3n_artificial. [Consulta: febrero 2009].
[10]. http://isa.umh.es/doct/pava/DeteccionBordes.pdf. [Consulta:septiembre 2009].
[11]. http://www.sc.ehu.es/ccwgrrom/transparencias/transp-vision-1/deteccion-de-bordes.ppt. [Consulta: septiembre 2009].

[12]. http://grvc.us.es/rar/mainFrame/navegacion/navegacion.html. [Consulta: Julio 2009]
[13]. http://www.jasvisio.com/aplicaciones-vision-artificial-industria.html. [Consulta: Julio 2009].
[14]. http://isa.umh.es/asignaturas/rvc/index.html#Documentacion. [Consulta: septiembre 2009].
[15]. http://es.wikipedia.org/wiki/Transformada_de_Hough. [Consulta: enero 2010].
[16]. http://members.fortunecity.es/davidweb2/visart/umbral.html. [Consulta: febrero 2010].

ANEXOS

Anexo 1. Esquema para el algoritmo de seguimiento y marcado del contorno.



Anexo 2. Código de la función para realizar el seguimiento y marcado del contorno. Obtención del código de cadena.

```
function [inii,inij,codigo,imagen1] =
seguimiento_contorno(imagen)

% Inicializacion de las variables
a=0;
b=0;
inii=0;
inij=0;
codcadena=0;
c=0;
n=0;
codigo=[];


% pixel inicial del contorno
[inii,inij,imagen1] = pixel_inicial1(imagen);
a=inii;
b=inij;


%Llamar a la funcion que determina el segundo pixel del
contorno
[a,b,codcadena,n,imagen1] = posicion_cero(a,b,imagen1);

if n==100
   codigo=[codigo,codcadena];
end

while ((a~=inii) || (b~=inij))

      if codcadena==0
         [a,b,codcadena,n,imagen1] = posicion_cero(a,b,imagen1);
         if n==100
            codigo=[codigo,codcadena];
         end

      elseif codcadena==1
            [a,b,codcadena,n,imagen1] =
posicion_uno(a,b,imagen1);
            if n==100
              codigo=[codigo,codcadena];
            end
            elseif codcadena==2
            [a,b,codcadena,n,imagen1] =
posicion_dos(a,b,imagen1);
            if n==100
           codigo=[codigo,codcadena];
            end

      elseif codcadena==3
            [a,b,codcadena,n,imagen1] =
posicion_tres(a,b,imagen1);
            if n==100
               codigo=[codigo,codcadena];
            end
```

```
    elseif codcadena==4
        c=n;
          [a,b,codcadena,n,imagen1] =
posicion_cuatro(a,b,imagen1);
          if n==100
             codigo=[codigo,codcadena];
          elseif (n~=100 && a==inii && b==inij && c==100)
             codigo=[codigo,codcadena];
          end

    elseif codcadena==5
        c=n;
          [a,b,codcadena,n,imagen1] =
posicion_cinco(a,b,imagen1);
          if n==100
             codigo=[codigo,codcadena];
          elseif (n~=100 && a==inii && b==inij && c==100)
             codigo=[codigo,codcadena];
           end

    elseif codcadena==6
        c=n;
          [a,b,codcadena,n,imagen1] =
posicion_seis(a,b,imagen1);
          if n==100
             codigo=[codigo,codcadena];
          elseif (n~=100 && a==inii && b==inij && c==100)
             codigo=[codigo,codcadena];
          end

    elseif codcadena==7
        c=n;
          [a,b,codcadena,n,imagen1] =
posicion_siete(a,b,imagen1);
          if n==100
            codigo=[codigo,codcadena];
          elseif (n~=100 && a==inii && b==inij && c==100)
            codigo=[codigo,codcadena];
          end
    end
end
```

Anexo 3. Código de la función para determinar los códigos de cadena de cada lado del polígono.

```
function [codigo1,codigo2,codigo3,codigo4,codigo5,codigo6,codigo7,codigo8, codigo9] = lados_del_poligono(codigo)

% Inicializacion de las variables

codigo1=[];
codigo2=[];
codigo3=[];
codigo4=[];
codigo5=[];
codigo6=[];
codigo7=[];
codigo8=[];
codigo9=[];

almacen=[];
a=0;
b=0;
c=0;
d=0;
e=0;
f=0;
g=0;
h=0;
j=0;
k=0;
L=0;
U=0;
T=0;
V=0;
W=0;
X=0;

[N,M]=size(codigo);
codigo1=[codigo(1),codigo(2)];
almacen=[codigo(1),codigo(2)];

for i=3:M
    if abs(codigo(i)-almacen(1))==0 &&
abs(codigo(i+1)-almacen(1))==0
        codigo1=[codigo1,codigo(i)];
    elseif abs(codigo(i)-almacen(1))==0 &&
abs(codigo(i+1)-almacen(1))==1 &&
abs(codigo(i)-almacen(2))==1 &&
abs(codigo(i+1)-almacen(2))==0
            codigo1=[codigo1,codigo(i)];

elseif abs(codigo(i)-almacen(1))==0 &&
abs(codigo(i+1)-almacen(1))==1 &&

abs(codigo(i)-almacen(2))==0 &&
abs(codigo(i+1)-almacen(2))==1
            codigo1=[codigo1,codigo(i)];
    elseif abs(codigo(i)-almacen(1))==1 &&
abs(codigo(i+1)-almacen(1))==0 &&
abs(codigo(i)-almacen(2))==0 &&
abs(codigo(i+1)-almacen(2))==1
            codigo1=[codigo1,codigo(i)];
    elseif abs(codigo(i)-almacen(1))==1 &&
abs(codigo(i+1)-almacen(1))==0 &&
abs(codigo(i)-almacen(2))==1 &&
abs(codigo(i+1)-almacen(2))==0
            codigo1=[codigo1,codigo(i)];
    elseif abs(codigo(i)-almacen(1))==7 &&
abs(codigo(i+1)-almacen(1))==0 &&
abs(codigo(i)-almacen(2))==0 &&
abs(codigo(i+1)-almacen(2))==7
            codigo1=[codigo1,codigo(i)];
    elseif abs(codigo(i)-almacen(1))==7 &&
abs(codigo(i+1)-almacen(1))==0 &&
abs(codigo(i)-almacen(2))==7 &&
abs(codigo(i+1)-almacen(2))==0
            codigo1=[codigo1,codigo(i)];
    elseif abs(codigo(i)-almacen(1))==0 &&
abs(codigo(i+1)-almacen(1))==7 &&
abs(codigo(i)-almacen(2))==0 &&
abs(codigo(i+1)-almacen(2))==7
            codigo1=[codigo1,codigo(i)];
```

```
   elseif abs(codigo(i)-almacen(1))==0 &&
abs(codigo(i+1)-almacen(1))==7 &&
abs(codigo(i)-almacen(2))==7 &&
abs(codigo(i+1)-almacen(2))==0
          codigo1=[codigo1,codigo(i)];
   elseif abs(codigo(i)-almacen(1))==7 &&
abs(codigo(i+1)-almacen(1))==7 &&
abs(codigo(i)-almacen(2))==0 &&
abs(codigo(i+1)-almacen(2))==0
           codigo1=[codigo1,codigo(i)];
   elseif abs(codigo(i)-almacen(1))==1 &&
abs(codigo(i+1)-almacen(1))==1 &&
abs(codigo(i)-almacen(2))==0 &&
abs(codigo(i+1)-almacen(2))==0
           codigo1=[codigo1,codigo(i)];
   else
      if abs(codigo(i)-almacen(1))==0 &&
abs(codigo(i)-almacen(2))==0
         codigo1=[codigo1,codigo(i)];
         almacen=[codigo(i+1),codigo(i+2)];
         codigo2=[codigo(i+1),codigo(i+2)];
         j=i+3;
         break
      elseif (abs(codigo(i)-almacen(1))==1 &&
abs(codigo(i)-almacen(2))==0) || (abs(codigo(i)-
almacen(1))==0 && abs(codigo(i)-
almacen(2))==7)
         codigo1=[codigo1,codigo(i)];
         almacen=[codigo(i+1),codigo(i+2)];
         codigo2=[codigo(i+1),codigo(i+2)];
         j=i+3;
         break
      elseif (abs(codigo(i)-almacen(1))==0 &&
abs(codigo(i)-almacen(2))==1) || (abs(codigo(i)-
almacen(1))==7 && abs(codigo(i)-
almacen(2))==0)
         codigo1=[codigo1,codigo(i)];
         almacen=[codigo(i+1),codigo(i+2)];
         codigo2=[codigo(i+1),codigo(i+2)];
         j=i+3;
         break
      else almacen=[codigo(i),codigo(i+1)];
          codigo2=[codigo(i),codigo(i+1)];
          j=i+2;
          break
      end
    end
   if codigo(i)~=almacen(1) &&
codigo(i)~=almacen(2) && a==0
      almacen(2)=codigo(i);
      a=a+1;
   end
end

if j~=0

  for j=j:M-1
    if j~=M-1
      if abs(codigo(j)-almacen(1))==0 &&
abs(codigo(j+1)-almacen(1))==0
         codigo2=[codigo2,codigo(j)];

      elseif abs(codigo(j)-almacen(1))==0 &&
abs(codigo(j+1)-almacen(1))==1 &&
abs(codigo(j)-almacen(2))==1 &&
abs(codigo(j+1)-almacen(2))==0
           codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==0 &&
abs(codigo(j+1)-almacen(1))==1 &&
abs(codigo(j)-almacen(2))==0 &&
abs(codigo(j+1)-almacen(2))==1
           codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==1 &&
abs(codigo(j+1)-almacen(1))==0 &&
abs(codigo(j)-almacen(2))==0 &&
abs(codigo(j+1)-almacen(2))==1
           codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==1 &&
abs(codigo(j+1)-almacen(1))==0 &&
abs(codigo(j)-almacen(2))==1 &&
abs(codigo(j+1)-almacen(2))==0
           codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==7 &&
abs(codigo(j+1)-almacen(1))==0 &&
abs(codigo(j)-almacen(2))==0 &&
abs(codigo(j+1)-almacen(2))==7
           codigo2=[codigo2,codigo(j)];
```

```
      elseif abs(codigo(j)-almacen(1))==7 &&
abs(codigo(j+1)-almacen(1))==0 &&
abs(codigo(j)-almacen(2))==7 &&
abs(codigo(j+1)-almacen(2))==0
            codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==0 &&
abs(codigo(j+1)-almacen(1))==7 &&
abs(codigo(j)-almacen(2))==0 &&
abs(codigo(j+1)-almacen(2))==7
            codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==0 &&
abs(codigo(j+1)-almacen(1))==7 &&
abs(codigo(j)-almacen(2))==7 &&
abs(codigo(j+1)-almacen(2))==0
            codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==7 &&
abs(codigo(j+1)-almacen(1))==7 &&
abs(codigo(j)-almacen(2))==0 &&
abs(codigo(j+1)-almacen(2))==0
            codigo2=[codigo2,codigo(j)];
      elseif abs(codigo(j)-almacen(1))==1 &&
abs(codigo(j+1)-almacen(1))==1 &&
abs(codigo(j)-almacen(2))==0 &&
abs(codigo(j+1)-almacen(2))==0
            codigo2=[codigo2,codigo(j)];
      else
       if abs(codigo(j)-almacen(1))==0 &&
abs(codigo(j)-almacen(2))==0
          codigo2=[codigo2,codigo(j)];
          almacen=[codigo(j+1),codigo(j+2)];
          codigo3=[codigo(j+1),codigo(j+2)];
          k=j+3;
          break
       elseif (abs(codigo(j)-almacen(1))==1 &&
abs(codigo(j)-almacen(2))==0) || (abs(codigo(j)-
almacen(1))==0 && abs(codigo(j)-
almacen(2))==7)
          codigo2=[codigo2,codigo(j)];
          almacen=[codigo(j+1),codigo(j+2)];
          codigo3=[codigo(j+1),codigo(j+2)];
          k=j+3;
          break
       elseif (abs(codigo(j)-almacen(1))==0 &&
abs(codigo(j)-almacen(2))==1) || (abs(codigo(j)-
almacen(1))==7 && abs(codigo(j)-
almacen(2))==0)
          codigo2=[codigo2,codigo(j)];
          almacen=[codigo(j+1),codigo(j+2)];
          codigo3=[codigo(j+1),codigo(j+2)];
          k=j+3;
          break
       else almacen=[codigo(j),codigo(j+1)];
           codigo3=[codigo(j),codigo(j+1)];
           k=j+2;
          break
       end
    end
      if codigo(j)~=almacen(1) &&
codigo(j)~=almacen(2) && b==0
        almacen(2)=codigo(j);
        b=b+1;
      end
    else codigo2=[codigo2,codigo(j),codigo(j+1)];
       break
    end
 end
end

if k~=0
  for k=k:M-1
   if k~=M-1
     if abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k+1)-almacen(1))==0
        codigo3=[codigo3,codigo(k)];

      elseif abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k+1)-almacen(1))==1 &&
abs(codigo(k)-almacen(2))==1 &&
abs(codigo(k+1)-almacen(2))==0
           codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k+1)-almacen(1))==1 &&
abs(codigo(k)-almacen(2))==0 &&
abs(codigo(k+1)-almacen(2))==1
           codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==1 &&
abs(codigo(k+1)-almacen(1))==0 &&
```

```
abs(codigo(k)-almacen(2))==0 &&
abs(codigo(k+1)-almacen(2))==1
            codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==1 &&
abs(codigo(k+1)-almacen(1))==0 &&
abs(codigo(k)-almacen(2))==1 &&
abs(codigo(k+1)-almacen(2))==0
            codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==7 &&
abs(codigo(k+1)-almacen(1))==0 &&
abs(codigo(k)-almacen(2))==0 &&
abs(codigo(k+1)-almacen(2))==7
            codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==7 &&
abs(codigo(k+1)-almacen(1))==0 &&
abs(codigo(k)-almacen(2))==7 &&
abs(codigo(k+1)-almacen(2))==0
            codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k+1)-almacen(1))==7 &&
abs(codigo(k)-almacen(2))==0 &&
abs(codigo(k+1)-almacen(2))==7
            codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k+1)-almacen(1))==7 &&
abs(codigo(k)-almacen(2))==7 &&
abs(codigo(k+1)-almacen(2))==0
            codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==7 &&
abs(codigo(k+1)-almacen(1))==7 &&
abs(codigo(k)-almacen(2))==0 &&
abs(codigo(k+1)-almacen(2))==0
            codigo3=[codigo3,codigo(k)];
      elseif abs(codigo(k)-almacen(1))==1 &&
abs(codigo(k+1)-almacen(1))==1 &&
abs(codigo(k)-almacen(2))==0 &&
abs(codigo(k+1)-almacen(2))==0
            codigo3=[codigo3,codigo(k)];
      else
       if abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k)-almacen(2))==0
          codigo3=[codigo3,codigo(k)];
          almacen=[codigo(k+1),codigo(k+2)];
          codigo4=[codigo(k+1),codigo(k+2)];
          L=k+3;
          break
       elseif (abs(codigo(k)-almacen(1))==1 &&
abs(codigo(k)-almacen(2))==0) ||
(abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k)-almacen(2))==7)
          codigo3=[codigo3,codigo(k)];
          almacen=[codigo(k+1),codigo(k+2)];
          codigo4=[codigo(k+1),codigo(k+2)];
          L=k+3;
          break
       elseif (abs(codigo(k)-almacen(1))==0 &&
abs(codigo(k)-almacen(2))==1) ||
(abs(codigo(k)-almacen(1))==7 &&
abs(codigo(k)-almacen(2))==0)
          codigo3=[codigo3,codigo(k)];
          almacen=[codigo(k+1),codigo(k+2)];
          codigo4=[codigo(k+1),codigo(k+2)];
          L=k+3;
          break
       else almacen=[codigo(k),codigo(k+1)];
           codigo4=[codigo(k),codigo(k+1)];
           L=k+2;
          break
       end
    end
      if codigo(k)~=almacen(1) &&
codigo(k)~=almacen(2) && c==0
        almacen(2)=codigo(k);
        c=c+1;
      end
     else
codigo3=[codigo3,codigo(k),codigo(k+1)];
        break
     end
  end
end

if L~=0
  for L=L:M-1
    if L~=M-1
      if abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L+1)-almacen(1))==0
        codigo4=[codigo4,codigo(L)];
```

```
    elseif abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L+1)-almacen(1))==1 &&
abs(codigo(L)-almacen(2))==1 &&
abs(codigo(L+1)-almacen(2))==0
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L+1)-almacen(1))==1 &&
abs(codigo(L)-almacen(2))==0 &&
abs(codigo(L+1)-almacen(2))==1
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==1 &&
abs(codigo(L+1)-almacen(1))==0 &&
abs(codigo(L)-almacen(2))==0 &&
abs(codigo(L+1)-almacen(2))==1
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==1 &&
abs(codigo(L+1)-almacen(1))==0 &&
abs(codigo(L)-almacen(2))==1 &&
abs(codigo(L+1)-almacen(2))==0
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==7 &&
abs(codigo(L+1)-almacen(1))==0 &&
abs(codigo(L)-almacen(2))==0 &&
abs(codigo(L+1)-almacen(2))==7
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==7 &&
abs(codigo(L+1)-almacen(1))==0 &&
abs(codigo(L)-almacen(2))==7 &&
abs(codigo(L+1)-almacen(2))==0
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L+1)-almacen(1))==7 &&
abs(codigo(L)-almacen(2))==0 &&
abs(codigo(L+1)-almacen(2))==7
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L+1)-almacen(1))==7 &&
abs(codigo(L)-almacen(2))==7 &&
abs(codigo(L+1)-almacen(2))==0
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==7 &&
abs(codigo(L+1)-almacen(1))==7 &&
abs(codigo(L)-almacen(2))==0 &&
abs(codigo(L+1)-almacen(2))==0
        codigo4=[codigo4,codigo(L)];
    elseif abs(codigo(L)-almacen(1))==1 &&
abs(codigo(L+1)-almacen(1))==1 &&
abs(codigo(L)-almacen(2))==0 &&
abs(codigo(L+1)-almacen(2))==0
        codigo4=[codigo4,codigo(L)];
    else
     if abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L)-almacen(2))==0
        codigo4=[codigo4,codigo(L)];
        almacen=[codigo(L+1),codigo(L+2)];
        codigo5=[codigo(L+1),codigo(L+2)];
        T=L+3;
        break
     elseif (abs(codigo(L)-almacen(1))==1 &&
abs(codigo(L)-almacen(2))==0) ||
(abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L)-almacen(2))==7)
        codigo4=[codigo4,codigo(L)];
        almacen=[codigo(L+1),codigo(L+2)];
        codigo5=[codigo(L+1),codigo(L+2)];
        T=L+3;
        break
     elseif (abs(codigo(L)-almacen(1))==0 &&
abs(codigo(L)-almacen(2))==1) ||
(abs(codigo(L)-almacen(1))==7 &&
abs(codigo(L)-almacen(2))==0)
        codigo4=[codigo4,codigo(L)];
        almacen=[codigo(L+1),codigo(L+2)];
        codigo5=[codigo(L+1),codigo(L+2)];
        T=L+3;
        break
     else almacen=[codigo(L),codigo(L+1)];
         codigo5=[codigo(L),codigo(L+1)];
         T=L+2;
        break
     end
  end
   if codigo(L)~=almacen(1) &&
codigo(L)~=almacen(2) && d==0
      almacen(2)=codigo(L);
      d=d+1;
```

```
    end
  else
codigo4=[codigo4,codigo(L),codigo(L+1)];
      break
  end
end
end

if T~=0
 for T=T:M-1
  if T~=M-1
    if abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T+1)-almacen(1))==0
       codigo5=[codigo5,codigo(T)];

   elseif abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T+1)-almacen(1))==1 &&
abs(codigo(T)-almacen(2))==1 &&
abs(codigo(T+1)-almacen(2))==0
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T+1)-almacen(1))==1 &&
abs(codigo(T)-almacen(2))==0 &&
abs(codigo(T+1)-almacen(2))==1
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==1 &&
abs(codigo(T+1)-almacen(1))==0 &&
abs(codigo(T)-almacen(2))==0 &&
abs(codigo(T+1)-almacen(2))==1
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==1 &&
abs(codigo(T+1)-almacen(1))==0 &&
abs(codigo(T)-almacen(2))==1 &&
abs(codigo(T+1)-almacen(2))==0
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==7 &&
abs(codigo(T+1)-almacen(1))==0 &&
abs(codigo(T)-almacen(2))==0 &&
abs(codigo(T+1)-almacen(2))==7
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==7 &&
abs(codigo(T+1)-almacen(1))==0 &&
abs(codigo(T)-almacen(2))==7 &&
abs(codigo(T+1)-almacen(2))==0
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T+1)-almacen(1))==7 &&
abs(codigo(T)-almacen(2))==0 &&
abs(codigo(T+1)-almacen(2))==7
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T+1)-almacen(1))==7 &&
abs(codigo(T)-almacen(2))==7 &&
abs(codigo(T+1)-almacen(2))==0
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==7 &&
abs(codigo(T+1)-almacen(1))==7 &&
abs(codigo(T)-almacen(2))==0 &&
abs(codigo(T+1)-almacen(2))==0
          codigo5=[codigo5,codigo(T)];
     elseif abs(codigo(T)-almacen(1))==1 &&
abs(codigo(T+1)-almacen(1))==1 &&
abs(codigo(T)-almacen(2))==0 &&
abs(codigo(T+1)-almacen(2))==0
          codigo5=[codigo5,codigo(T)];
     else
      if abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T)-almacen(2))==0
          codigo5=[codigo5,codigo(T)];
          almacen=[codigo(T+1),codigo(T+2)];
          codigo6=[codigo(T+1),codigo(T+2)];
          U=T+3;
          break
       elseif (abs(codigo(T)-almacen(1))==1 &&
abs(codigo(T)-almacen(2))==0) ||
(abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T)-almacen(2))==7)
          codigo5=[codigo5,codigo(T)];
          almacen=[codigo(T+1),codigo(T+2)];
          codigo6=[codigo(T+1),codigo(T+2)];
          U=T+3;
          break
       elseif (abs(codigo(T)-almacen(1))==0 &&
abs(codigo(T)-almacen(2))==1) ||
(abs(codigo(T)-almacen(1))==7 &&
abs(codigo(T)-almacen(2))==0)
          codigo5=[codigo5,codigo(T)];
          almacen=[codigo(T+1),codigo(T+2)];
```

```
          codigo6=[codigo(T+1),codigo(T+2)];
          U=T+3;
          break
       else almacen=[codigo(T),codigo(T+1)];
           codigo6=[codigo(T),codigo(T+1)];
           U=T+2;
          break
       end
   end
    if codigo(T)~=almacen(1) &&
codigo(T)~=almacen(2) && e==0
        almacen(2)=codigo(T);
        e=e+1;
    end
   else
codigo5=[codigo5,codigo(T),codigo(T+1)];
       break
   end
 end
end

if U~=0
  for U=U:M-1
    if U~=M-1
      if abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U+1)-almacen(1))==0
         codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U+1)-almacen(1))==1 &&
abs(codigo(U)-almacen(2))==1 &&
abs(codigo(U+1)-almacen(2))==0
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U+1)-almacen(1))==1 &&
abs(codigo(U)-almacen(2))==0 &&
abs(codigo(U+1)-almacen(2))==1
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==1 &&
abs(codigo(U+1)-almacen(1))==0 &&
abs(codigo(U)-almacen(2))==0 &&
abs(codigo(U+1)-almacen(2))==1
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==1 &&
abs(codigo(U+1)-almacen(1))==0 &&
abs(codigo(U)-almacen(2))==1 &&
abs(codigo(U+1)-almacen(2))==0
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==7 &&
abs(codigo(U+1)-almacen(1))==0 &&
abs(codigo(U)-almacen(2))==0 &&
abs(codigo(U+1)-almacen(2))==7
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==7 &&
abs(codigo(U+1)-almacen(1))==0 &&
abs(codigo(U)-almacen(2))==7 &&
abs(codigo(U+1)-almacen(2))==0
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U+1)-almacen(1))==7 &&
abs(codigo(U)-almacen(2))==0 &&
abs(codigo(U+1)-almacen(2))==7
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U+1)-almacen(1))==7 &&
abs(codigo(U)-almacen(2))==7 &&
abs(codigo(U+1)-almacen(2))==0
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==7 &&
abs(codigo(U+1)-almacen(1))==7 &&
abs(codigo(U)-almacen(2))==0 &&
abs(codigo(U+1)-almacen(2))==0
           codigo6=[codigo6,codigo(U)];
      elseif abs(codigo(U)-almacen(1))==1 &&
abs(codigo(U+1)-almacen(1))==1 &&
abs(codigo(U)-almacen(2))==0 &&
abs(codigo(U+1)-almacen(2))==0
           codigo6=[codigo6,codigo(U)];
      else
       if abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U)-almacen(2))==0
          codigo6=[codigo6,codigo(U)];
          almacen=[codigo(U+1),codigo(U+2)];
          codigo7=[codigo(U+1),codigo(U+2)];
          V=U+3;
          break
       elseif (abs(codigo(U)-almacen(1))==1 &&
abs(codigo(U)-almacen(2))==0) ||
```

```
(abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U)-almacen(2))==7)
          codigo6=[codigo6,codigo(U)];
          almacen=[codigo(U+1),codigo(U+2)];
          codigo7=[codigo(U+1),codigo(U+2)];
          V=U+3;
          break
       elseif (abs(codigo(U)-almacen(1))==0 &&
abs(codigo(U)-almacen(2))==1) ||
(abs(codigo(U)-almacen(1))==7 &&
abs(codigo(U)-almacen(2))==0)
          codigo6=[codigo6,codigo(U)];
          almacen=[codigo(U+1),codigo(U+2)];
          codigo7=[codigo(U+1),codigo(U+2)];
          V=U+3;
          break
       else almacen=[codigo(U),codigo(U+1)];
           codigo7=[codigo(U),codigo(U+1)];
           V=U+2;
          break
       end
     end
     if codigo(U)~=almacen(1) &&
codigo(U)~=almacen(2) && f==0
        almacen(2)=codigo(U);
        f=1;
     end
  else
codigo6=[codigo6,codigo(U),codigo(U+1)];
       break
  end
 end
end

if V~=0
  for V=V:M-1
    if V~=M-1
      if abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V+1)-almacen(1))==0
         codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V+1)-almacen(1))==1 &&
abs(codigo(V)-almacen(2))==1 &&
abs(codigo(V+1)-almacen(2))==0
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V+1)-almacen(1))==1 &&
abs(codigo(V)-almacen(2))==0 &&
abs(codigo(V+1)-almacen(2))==1
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==1 &&
abs(codigo(V+1)-almacen(1))==0 &&
abs(codigo(V)-almacen(2))==0 &&
abs(codigo(V+1)-almacen(2))==1
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==1 &&
abs(codigo(V+1)-almacen(1))==0 &&
abs(codigo(V)-almacen(2))==1 &&
abs(codigo(V+1)-almacen(2))==0
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==7 &&
abs(codigo(V+1)-almacen(1))==0 &&
abs(codigo(V)-almacen(2))==0 &&
abs(codigo(V+1)-almacen(2))==7
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==7 &&
abs(codigo(V+1)-almacen(1))==0 &&
abs(codigo(V)-almacen(2))==7 &&
abs(codigo(V+1)-almacen(2))==0
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V+1)-almacen(1))==7 &&
abs(codigo(V)-almacen(2))==0 &&
abs(codigo(V+1)-almacen(2))==7
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V+1)-almacen(1))==7 &&
abs(codigo(V)-almacen(2))==7 &&
abs(codigo(V+1)-almacen(2))==0
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==7 &&
abs(codigo(V+1)-almacen(1))==7 &&
abs(codigo(V)-almacen(2))==0 &&
abs(codigo(V+1)-almacen(2))==0
          codigo7=[codigo7,codigo(V)];
     elseif abs(codigo(V)-almacen(1))==1 &&
abs(codigo(V+1)-almacen(1))==1 &&
```

```
abs(codigo(V)-almacen(2))==0 &&
abs(codigo(V+1)-almacen(2))==0
            codigo7=[codigo7,codigo(V)];
       else
        if abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V)-almacen(2))==0
           codigo7=[codigo7,codigo(V)];
           almacen=[codigo(V+1),codigo(V+2)];
           codigo8=[codigo(V+1),codigo(V+2)];
           W=V+3;
           break
        elseif (abs(codigo(V)-almacen(1))==1 &&
abs(codigo(V)-almacen(2))==0) ||
(abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V)-almacen(2))==7)
           codigo7=[codigo7,codigo(V)];
           almacen=[codigo(V+1),codigo(V+2)];
           codigo8=[codigo(V+1),codigo(V+2)];
           W=V+3;
           break
        elseif (abs(codigo(V)-almacen(1))==0 &&
abs(codigo(V)-almacen(2))==1) ||
(abs(codigo(V)-almacen(1))==7 &&
abs(codigo(V)-almacen(2))==0)
           codigo7=[codigo7,codigo(V)];
           almacen=[codigo(V+1),codigo(V+2)];
           codigo8=[codigo(V+1),codigo(V+2)];
           W=V+3;
           break
        else almacen=[codigo(V),codigo(V+1)];
            codigo8=[codigo(V),codigo(V+1)];
           W=V+2;
           break
        end
    end
     if codigo(V)~=almacen(1) &&
codigo(V)~=almacen(2) && g==0
         almacen(2)=codigo(V);
         g=1;
     end
  else
codigo7=[codigo7,codigo(V),codigo(V+1)];
        break
  end
 end
end

if W~=0
  for W=W:M-1
    if W~=M-1
      if abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W+1)-almacen(1))==0
          codigo8=[codigo8,codigo(W)];
      elseif abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W+1)-almacen(1))==1 &&
abs(codigo(W)-almacen(2))==1 &&
abs(codigo(W+1)-almacen(2))==0
            codigo8=[codigo8,codigo(W)];
       elseif abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W+1)-almacen(1))==1 &&
abs(codigo(W)-almacen(2))==0 &&
abs(codigo(W+1)-almacen(2))==1
            codigo8=[codigo8,codigo(W)];
       elseif abs(codigo(W)-almacen(1))==1 &&
abs(codigo(W+1)-almacen(1))==0 &&
abs(codigo(W)-almacen(2))==0 &&
abs(codigo(W+1)-almacen(2))==1
            codigo8=[codigo8,codigo(W)];
       elseif abs(codigo(W)-almacen(1))==1 &&
abs(codigo(W+1)-almacen(1))==0 &&
abs(codigo(W)-almacen(2))==1 &&
abs(codigo(W+1)-almacen(2))==0
            codigo8=[codigo8,codigo(W)];
       elseif abs(codigo(W)-almacen(1))==7 &&
abs(codigo(W+1)-almacen(1))==0 &&
abs(codigo(W)-almacen(2))==0 &&
abs(codigo(W+1)-almacen(2))==7
            codigo8=[codigo8,codigo(W)];
       elseif abs(codigo(W)-almacen(1))==7 &&
abs(codigo(W+1)-almacen(1))==0 &&
abs(codigo(W)-almacen(2))==7 &&
abs(codigo(W+1)-almacen(2))==0
            codigo8=[codigo8,codigo(W)];
       elseif abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W+1)-almacen(1))==7 &&
abs(codigo(W)-almacen(2))==0 &&
abs(codigo(W+1)-almacen(2))==7
            codigo8=[codigo8,codigo(W)];
```

```
      elseif abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W+1)-almacen(1))==7 &&
abs(codigo(W)-almacen(2))==7 &&
abs(codigo(W+1)-almacen(2))==0
            codigo8=[codigo8,codigo(W)];
      elseif abs(codigo(W)-almacen(1))==7 &&
abs(codigo(W+1)-almacen(1))==7 &&
abs(codigo(W)-almacen(2))==0 &&
abs(codigo(W+1)-almacen(2))==0
            codigo8=[codigo8,codigo(W)];
      elseif abs(codigo(W)-almacen(1))==1 &&
abs(codigo(W+1)-almacen(1))==1 &&
abs(codigo(W)-almacen(2))==0 &&
abs(codigo(W+1)-almacen(2))==0
            codigo8=[codigo8,codigo(W)];
      else
       if abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W)-almacen(2))==0
          codigo8=[codigo8,codigo(W)];
          almacen=[codigo(W+1),codigo(W+2)];
          codigo9=[codigo(W+1),codigo(W+2)];
          X=W+3;
          break
       elseif (abs(codigo(W)-almacen(1))==1 &&
abs(codigo(W)-almacen(2))==0) ||
(abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W)-almacen(2))==7)
          codigo8=[codigo8,codigo(W)];
          almacen=[codigo(W+1),codigo(W+2)];
          codigo9=[codigo(W+1),codigo(W+2)];
          X=W+3;
          break
       elseif (abs(codigo(W)-almacen(1))==0 &&
abs(codigo(W)-almacen(2))==1) ||
(abs(codigo(W)-almacen(1))==7 &&
abs(codigo(W)-almacen(2))==0)
          codigo8=[codigo8,codigo(W)];
          almacen=[codigo(W+1),codigo(W+2)];
          codigo9=[codigo(W+1),codigo(W+2)];
          X=W+3;
          break
       else almacen=[codigo(W),codigo(W+1)];
           codigo9=[codigo(W),codigo(W+1)];
          X=W+2;
            break
        end
      end
       if codigo(W)~=almacen(1) &&
codigo(W)~=almacen(2) && h==0
          almacen(2)=codigo(W);
          h=1;
      end
    else
codigo8=[codigo8,codigo(W),codigo(W+1)];
        break
    end
  end
end
```

Anexo 4. Código de la función para determinar uno de los ángulos interiores del polígono.

```
function teta =
angulo_entre_dos_lados(inii,inij,imagen,codigo
1,codigo2,codigo3,codigo4,codigo5,codigo6,co
digo7,codigo8)

% Inicializacion de las variables
 teta=0;

 alfa1=0;
 alfa2=0;
 alfa3=0;
 alfa4=0;
 alfa5=0;
 alfa6=0;
 alfa7=0;
 alfa8=0;

m1=0;
m2=0;
m3=0;
m4=0;
m5=0;
m6=0;
m7=0;
m8=0;
posx=0;
posy=0;
longitud1=0;
longitud2=0;
longitud3=0;
longitud4=0;
longitud5=0;
longitud6=0;
longitud7=0;
longitud8=0;

[N1,M1]=size(codigo1);
[N2,M2]=size(codigo2);
[N3,M3]=size(codigo3);
[N4,M4]=size(codigo4);
[N5,M5]=size(codigo5);
[N6,M6]=size(codigo6);
[N7,M7]=size(codigo7);
[N8,M8]=size(codigo8);

if N1~=0
   media=media_lados(codigo1);
   if media==0

[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)

if alfa1~=inf
       alfa1=0;
      end


   elseif media==4

[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)
      if alfa1~=inf
       alfa1=0;
      end

   elseif media==2

[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)
      if alfa1~=inf
       alfa1=90;
      end

   elseif media==6

[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)
      if alfa1~=inf
       alfa1=90;
      end

   elseif media==3
```

```
[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)
      if alfa1~=inf
       alfa1=45;
      end

   elseif media==7

[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)
      if alfa1~=inf
       alfa1=45;
      end

   elseif media==1

[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)
      if alfa1~=inf
       alfa1=135;
      end

   elseif media==5

[alfa1,posx,posy,longitud1]=lados_verdaderos(i
nii,inij,codigo1,imagen)
      if alfa1~=inf
       alfa1=135;
      end
   else [m1,longitud1,posx,posy] =
ajuste_rectas_final(inii,inij,codigo1,imagen);
       if m1~=inf
          alfa1=atan(m1)*180/pi;
       else alfa1=inf;
       end
   end

end


if N2~=0
   media=media_lados(codigo2);
    if media==0
```

```
[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
      if alfa2~=inf
       alfa2=0;
      end

   elseif media==4

[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
       if alfa2~=inf
        alfa2=0;
       end

   elseif media==2

[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
       if alfa2~=inf
        alfa2=90;
       end

   elseif media==6

[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
       if alfa2~=inf
        alfa2=90;
       end


   elseif media==3

[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
       if alfa2~=inf
        alfa2=45;
       end


   elseif media==7
```

```
[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
      if alfa2~=inf
       alfa2=45;
      end

   elseif media==1

[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
      if alfa2~=inf
       alfa2=135;
      end

   elseif media==5

[alfa2,posx,posy,longitud2]=lados_verdaderos(
posx,posy,codigo2,imagen)
      if alfa2~=inf
       alfa2=135;
      end
   else [m2,longitud2,posx,posy] =
ajuste_rectas_final(posx,posy,codigo2,imagen)
;
       if m2~=inf
          alfa2=atan(m2)*180/pi;
       else alfa2=inf;
       end
   end
end


if N3~=0
   media=media_lados(codigo3);
    if media==0

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
     if alfa3~=inf
      alfa3=0;
     end

   elseif media==4

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
      if alfa3~=inf
       alfa3=0;
      end

   elseif media==2

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
      if alfa3~=inf
       alfa3=90;
      end

   elseif media==6

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
      if alfa3~=inf
       alfa3=90;
      end

   elseif media==3

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
      if alfa3~=inf
       alfa3=45;
      end

   elseif media==7

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
      if alfa3~=inf
       alfa3=45;
      end

   elseif media==1

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
      if alfa3~=inf
```

```
      alfa3=135;
     end

   elseif media==5

[alfa3,posx,posy,longitud3]=lados_verdaderos(
posx,posy,codigo3,imagen)
     if alfa3~=inf
      alfa3=135;
     end
   else [m3,longitud3,posx,posy] =
ajuste_rectas_final(posx,posy,codigo3,imagen)
;
      if m3~=inf
         alfa3=atan(m3)*180/pi;
       else alfa3=inf;
       end
   end

end


if N4~=0
   media=media_lados(codigo4);
    if media==0

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=0;
     end

   elseif media==4

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=0;
     end

   elseif media==2

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=90;
     end

   elseif media==6

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=90;
     end

   elseif media==3

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=45;
     end

   elseif media==7

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=45;
     end

   elseif media==1

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=135;
     end

   elseif media==5

[alfa4,posx,posy,longitud4]=lados_verdaderos(
posx,posy,codigo4,imagen)
     if alfa4~=inf
      alfa4=135;
     end
```

```
   else [m4,longitud4,posx,posy] =
ajuste_rectas_final(posx,posy,codigo4,imagen)
;
      if m4~=inf
         alfa4=atan(m4)*180/pi;
       else alfa4=inf;
       end
   end
end

if N5~=0
   media=media_lados(codigo5);
   if media==0

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=0;
     end

   elseif media==4

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=0;
     end

   elseif media==2

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=90;
     end

   elseif media==6

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=90;
     end

   elseif media==3

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=45;
     end

   elseif media==7

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=45;
     end

   elseif media==1

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=135;
     end

   elseif media==5

[alfa5,posx,posy,longitud5]=lados_verdaderos(
posx,posy,codigo5,imagen)
     if alfa5~=inf
      alfa5=135;
     end
   else [m5,longitud5,posx,posy] =
ajuste_rectas_final(posx,posy,codigo5,imagen)
;
      if m5~=inf
         alfa5=atan(m5)*180/pi;
       else alfa5=inf;
       end
   end
end

if N6~=0
   media=media_lados(codigo6);
   if media==0
```

```
[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=0;
      end

   elseif media==4

[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=0;
      end

   elseif media==2

[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=90;
      end

   elseif media==6

[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=90;
      end

   elseif media==3

[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=45;
      end

   elseif media==7

[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=45;
      end

   elseif media==1

[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=135;
      end

   elseif media==5

[alfa6,posx,posy,longitud6]=lados_verdaderos(
posx,posy,codigo6,imagen)
      if alfa6~=inf
       alfa6=135;
      end
   else
      [m6,longitud6,posx,posy] =
ajuste_rectas_final(posx,posy,codigo6,imagen)
;
       if m6~=inf
          alfa6=atan(m6)*180/pi;
        else alfa6=inf;
        end
   end
end

if N7~=0
   media=media_lados(codigo7);
   if media==0

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
      if alfa7~=inf
       alfa7=0;
      end

   elseif media==4

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
      if alfa7~=inf
```

```
      alfa7=0;
     end

   elseif media==2

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
     if alfa7~=inf
      alfa7=90;
     end

   elseif media==6

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
     if alfa7~=inf
      alfa7=90;
     end

   elseif media==3

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
     if alfa7~=inf
      alfa7=45;
     end

   elseif media==7

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
     if alfa7~=inf
      alfa7=45;
     end

   elseif media==1

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
     if alfa7~=inf
      alfa7=135;
     end

   elseif media==5

[alfa7,posx,posy,longitud7]=lados_verdaderos(
posx,posy,codigo7,imagen)
     if alfa7~=inf
      alfa7=135;
     end
   else [m7,longitud7,posx,posy] =
ajuste_rectas_final(posx,posy,codigo7,imagen)
;
      if m7~=inf
         alfa7=atan(m7)*180/pi;
       else alfa7=inf;
       end
   end
end

if N8~=0
   media=media_lados(codigo8);
   if media==0

[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
     if alfa8~=inf
      alfa8=0;
     end

   elseif media==4

[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
     if alfa8~=inf
      alfa8=0;
     end

   elseif media==2

[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
     if alfa8~=inf
      alfa8=90;
     end

   elseif media==6
```

```
[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
      if alfa8~=inf
       alfa8=90;
      end

   elseif media==3

[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
      if alfa8~=inf
       alfa8=45;
      end

   elseif media==7

[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
      if alfa8~=inf
       alfa8=45;
      end

   elseif media==1

[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
      if alfa8~=inf
       alfa8=135;
      end

   elseif media==5

[alfa8,posx,posy,longitud8]=lados_verdaderos(
posx,posy,codigo8,imagen)
      if alfa8~=inf
       alfa8=135;
      end
   else [m8,longitud8,posx,posy] =
ajuste_rectas_final(posx,posy,codigo8,imagen)
;
       if m8~=inf
          alfa8=atan(m8)*180/pi;
        else alfa8=inf;
          end
      end
   end
end

if alfa1~=inf
   l=longitud1;
elseif alfa2~=inf
   l=longitud2;
elseif alfa3~=inf
   l=longitud3;
elseif alfa4~=inf
   l=longitud4;
elseif alfa5~=inf
   l=longitud5;
elseif alfa6~=inf
   l=longitud6;
elseif alfa7~=inf
   l=longitud7;
elseif alfa8~=inf
   l=longitud8;
end

%Hallar el lado de mayor longitud
if l==longitud1
   if alfa2~=inf
     if l<longitud2
        l=longitud2;
     end
   end
   if alfa3~=inf
     if l<longitud3
        l=longitud3;
       end
   end
   if alfa4~=inf
     if l<longitud4
        l=longitud4;
       end
   end
   if alfa5~=inf
     if l<longitud5
        l=longitud5;
       end
   end
```

```
    if alfa6~=inf
      if l<longitud6
        l=longitud6;
       end
    end
    if alfa7~=inf
      if l<longitud7
        l=longitud7;
       end
    end
    if alfa8~=inf
      if l<longitud8
        l=longitud8;
       end
    end
end

if l==longitud2
    if alfa3~=inf
      if l<longitud3
        l=longitud3;
      end
    end
    if alfa4~=inf
      if l<longitud4
        l=longitud4;
      end
    end
    if alfa5~=inf
      if l<longitud5
        l=longitud5;
       end
    end
    if alfa6~=inf
      if l<longitud6
        l=longitud6;
       end
    end
    if alfa7~=inf
      if l<longitud7
        l=longitud7;
       end
    end
    if alfa8~=inf
      if l<longitud8
        l=longitud8;
       end
    end
end
if l==longitud3
    if alfa4~=inf
      if l<longitud4
        l=longitud4;
       end
    end
    if alfa5~=inf
      if l<longitud5
        l=longitud5;
       end
    end
    if alfa6~=inf
      if l<longitud6
        l=longitud6;
       end
    end
    if alfa7~=inf
      if l<longitud7
        l=longitud7;
       end
    end
    if alfa8~=inf
      if l<longitud8
        l=longitud8;
       end
    end
end
if l==longitud4
    if alfa5~=inf
      if l<longitud5
        l=longitud5;
       end
    end
    if alfa6~=inf
      if l<longitud6
        l=longitud6;
       end
    end
    if alfa7~=inf
```

```
      if l<longitud7
         l=longitud7;
        end
     end
     if m8~=inf
       if l<longitud8
          l=longitud8;
        end
     end
end
if l==longitud5
     if alfa6~=inf
       if l<longitud6
          l=longitud6;
      end
    end
    if alfa7~=inf
       if l<longitud7
          l=longitud7;
      end
    end
    if alfa8~=inf
       if l<longitud8
         l=longitud8;
       end
   end
end

if l==longitud6
     if alfa7~=inf
       if l<longitud7
         l=longitud7;
       end
     end
       if alfa8~=inf
         if l<longitud8
           l=longitud8;
         end
       end
end

if l==longitud7
    if alfa8~=inf
       if l<longitud8
          l=longitud8;
       end
    end
end

% Hallar el angulo entre dos lados
if l==longitud1
    if M8~=0
       if alfa2~=inf && alfa8~=inf
          if longitud2>longitud8
             teta=alfa2-alfa1
          else teta=alfa1-alfa8
          end
       elseif alfa2~=inf
          teta=alfa2-alfa1
       else teta=alfa1-alfa8
       end
    elseif M7~=0
       if alfa2~=inf && alfa7~=inf
          if longitud2>longitud7
             teta=alfa2-alfa1
          else teta=alfa1-alfa7
          end
       elseif alfa2~=inf
          teta=alfa2-alfa1
       else teta=alfa1-alfa7
       end
    elseif M6~=0
       if alfa2~=inf && alfa6~=inf
          if longitud2>longitud6
             teta=alfa2-alfa1
          else teta=alfa1-alfa6
          end
       elseif alfa2~=inf
          teta=alfa2-alfa1
       else teta=alfa1-alfa6
       end
    elseif M5~=0
       if alfa2~=inf && alfa5~=inf
          if longitud2>longitud5
             teta=alfa2-alfa1
          else teta=alfa1-alfa5
          end
       elseif alfa2~=inf
```

```
            teta=alfa2-alfa1
        else teta=alfa1-alfa5
        end
    elseif M4~=0
        if alfa2~=inf && alfa4~=inf
            if longitud2>longitud4
                teta=alfa2-alfa1
            else teta=alfa1-alfa4
            end
        elseif alfa2~=inf
            teta=alfa2-alfa1
        else teta=alfa1-alfa4
        end
    elseif M3~=0
        if alfa2~=inf && alfa3~=inf
            if longitud2>longitud3
                teta=alfa2-alfa1
            else teta=alfa1-alfa3
            end
        elseif alfa2~=inf
            teta=alfa2-alfa1
        else teta=alfa1-alfa3
        end
    end
end

if l==longitud2
        if alfa1~=inf && alfa3~=inf
            if longitud1>longitud3
                teta=alfa2-alfa1
            else teta=alfa3-alfa2
            end
        elseif alfa1~=inf
            teta=alfa2-alfa1
        else teta=alfa3-alfa2
        end
end

if l==longitud3
    if M4~=0
        if alfa2~=inf && alfa4~=inf
            if longitud2>longitud4
                teta=alfa3-alfa2
            else teta=alfa4-alfa3
            end
        elseif alfa2~=inf
            teta=alfa3-alfa2
        else teta=alfa4-alfa3
        end
    else
        if alfa1~=inf && alfa2~=inf
            if longitud1>longitud2
                teta=alfa1-alfa3
            else teta=alfa3-alfa2
            end
        elseif alfa2~=inf
            teta=alfa3-alfa2
        else teta=alfa1-alfa3
        end
    end
end

if l==longitud4
    if M5~=0
        if alfa5~=inf && alfa3~=inf
            if longitud3>longitud5
                teta=alfa4-alfa3
            else teta=alfa5-alfa4
            end
        elseif alfa3~=inf
            teta=alfa4-alfa3
        else teta=alfa5-alfa4
        end
    else
        if alfa1~=inf && alfa3~=inf
            if longitud1>longitud3
                teta=alfa1-alfa4
            else teta=alfa4-alfa3
            end
        elseif alfa3~=inf
            teta=alfa4-alfa3
        else teta=alfa1-alfa4
        end
    end
end

if l==longitud5
    if M6~=0
```

```
      if alfa6~=inf && alfa4~=inf
         if longitud4>longitud6
            teta=alfa5-alfa4
         else teta=alfa6-alfa5
         end
      elseif alfa4~=inf
         teta=alfa5-alfa4
      else teta=alfa6-alfa5
      end
   else
      if alfa1~=inf && alfa4~=inf
         if longitud1>longitud4
            teta=alfa1-alfa5
         else teta=alfa5-alfa4
         end
      elseif alfa4~=inf
         teta=alfa5-alfa4
      else teta=alfa1-alfa5
      end
   end
end

if l==longitud6
   if M7~=0
      if alfa5~=inf && alfa7~=inf
         if longitud5>longitud7
            teta=alfa6-alfa5
         else teta=alfa7-alfa6
         end
      elseif alfa5~=inf
         teta=alfa7-alfa6
      else teta=alfa6-alfa5
      end
   else
      if alfa1~=inf && alfa5~=inf
         if longitud1>longitud5
            teta=alfa1-alfa6
         else teta=alfa6-alfa5
         end
      elseif alfa5~=inf
         teta=alfa6-alfa5
      else teta=alfa1-alfa6
      end
   end
end

if l==longitud7
   if M8~=0
      if alfa6~=inf && alfa8~=inf
         if longitud6>longitud8
            teta=alfa7-alfa6
         else teta=alfa8-alfa7
         end
      elseif alfa8~=inf
         teta=alfa7-alfa6
      else teta=alfa8-alfa7
      end
   else
      if alfa1~=inf && alfa6~=inf
         if longitud1>longitud6
            teta=alfa1-alfa7
         else teta=alfa7-alfa6
         end
      elseif alfa6~=inf
         teta=alfa7-alfa6
      else teta=alfa1-alfa7
      end
   end
end

if l==longitud8
   if alfa1~=inf && alfa7~=inf
         if longitud1>longitud7
            teta=alfa1-alfa8
         else teta=alfa8-alfa7
         end
   elseif alfa7~=inf
         teta=alfa8-alfa7
   else teta=alfa1-alfa8
   end
end

if -180<teta && teta<0
      teta=180+teta
elseif -270<teta && teta<-180
      teta=360+teta
end
```

Anexo 5. Código del programa principal.

```
% Programa principal

% Inicialización de variables
codigo1=[];
codigo2=[];
codigo3=[];
codigo4=[];
codigo5=[];
codigo6=[];
codigo7=[];
codigo8=[];
codigo9=[];
inii=0;
inij=0;
encontrado=0;

% Leer imagen
imagen=imread('5ladosrotados.bmp');
imshow(imagen)

%Algoritmo de seguimiento del contorno del
poligono
[inii,inij,codigo,imagen1] =
seguimiento_contorno(imagen);
imshow(imagen1)
pause;

% Determinacion del numero de lados del
poligono
[codigo1,codigo2,codigo3,codigo4,codigo5,codi
go6,codigo7,codigo8,codigo9]=lados_del_polig
ono(codigo);

% Determinacion de los angulos entre los
lados del poligono
teta
=angulo_entre_dos_lados(inii,inij,imagen,codig
o1,codigo2,codigo3,codigo4,codigo5,codigo6,c
odigo7,codigo8,codigo9)

if teta>58 && teta<62
   disp('el polígono es un triangulo');
elseif teta>88 && teta<92
   disp('el polígono es un cuadrado');
elseif teta>106 && teta<110
   disp('el polígono es un pentagono');
elseif teta>118 && teta<122
   disp('el polígono es un hexagono');
elseif teta>126 && teta<130
   disp('el polígono es un heptagono');
elseif teta>133 && teta<137
   disp('el polígono es un octagono');
end
```

Anexo 6. Código de la función para determinar el píxel inicial del contorno.

```
function [inii,inij,imagen1] = pixel_inicial1(imagen1)
%inicializacion de las variables

inii=0;
inij=0;
encontrado=0;

%programa para encontrar el pixel inicial del contorno

[N,M] = size(imagen1);

for i=1:N
   for j=1:M
      if imagen1(i,j)==15
         encontrado=1;
         break
      end
   end
   if encontrado==1
      inii=i;
      inij=j;
     break
   end
end

if imagen1(i+1,j-1)==0
   inii=i;
   inij=j;
elseif imagen1(i+1,j-1)==15 && imagen1(i+1,j-2)==0
   inii=i;
   inij=j;
elseif imagen1(i+1,j-1)==15 && imagen1(i+1,j-2)==15
   d=i+1;
   for d=d:M
      if imagen1(d,j)>=0 || imagen1(d,j)<=14
         inii=d-1;
         inij=j;
         break
      end
   end
```

```
end
imagen1(inii,inij)=7;
```

Anexo 7. Código de la función para determinar el píxel siguiente según la dirección a_{i-1}=0.

```
function [a,b,codcadena,n,imagen1] =
posicion_cero(a,b,imagen1)
% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;



if a>1
    if imagen1(a-1,b+1)==15
        codcadena=7;
        n=100;
        imagen1(a-1,b+1)=7;
        i=a-1;
        j=b+1;

    elseif imagen1(a-1,b+1)==7
        codcadena=7;
        n=7;
        i=a-1;
        j=b+1;

    elseif imagen1(a,b+1)==15
        codcadena=0;
        imagen1(a,b+1)=7;
        n=100;
        i=a;
        j=b+1;

    elseif imagen1(a,b+1)==7
        codcadena=0;
        n=0;
        i=a;
        j=b+1;

    elseif imagen1(a+1,b+1)==15
        codcadena=1;
        imagen1(a+1,b+1)=7;
        n=100;
        i=a+1;
        j=b+1;

    elseif imagen1(a+1,b+1)==7
        codcadena=1;
        n=1;
        i=a+1;
        j=b+1;

    elseif imagen1(a+1,b)==15
        codcadena=2;
        imagen1(a+1,b)=7;
        n=100;



        i=a+1;
        j=b;

    elseif imagen1(a+1,b)==7
        codcadena=2;
        n=2;
        i=a+1;
        j=b;

    elseif imagen1(a+1,b-1)==15
        codcadena=3;
        imagen1(a+1,b-1)=7;
        n=100;
        i=a+1;
        j=b-1;

    elseif imagen1(a+1,b-1)==7
        codcadena=3;
        n=3;
        i=a+1;
        j=b-1;

    elseif imagen1(a,b-1)==7
        codcadena=4;
        n=4;
        i=a;
        j=b-1;
    end
end
```

```
if a==1

    if imagen1(a,b+1)==15
        codcadena=0;
        imagen1(a,b+1)=7;
        n=100;
        i=a;
        j=b+1;

    elseif imagen1(a,b+1)==7
        codcadena=0;
        n=0;
        i=a;
        j=b+1;

    elseif imagen1(a+1,b+1)==15
        codcadena=1;
        imagen1(a+1,b+1)=7;
        n=100;
        i=a+1;
        j=b+1;

    elseif imagen1(a+1,b+1)==7
        codcadena=1;
        n=1;
        i=a+1;
        j=b+1;

    elseif imagen1(a+1,b)==15
        codcadena=2;
        imagen1(a+1,b)=7;
        n=100;
        i=a+1;
        j=b;

     elseif imagen1(a+1,b)==7
        codcadena=2;
        n=2;
        i=a+1;
        j=b;

    elseif imagen1(a+1,b-1)==15
        codcadena=3;
        imagen1(a+1,b-1)=7;
        n=100;
        i=a+1;
        j=b-1;

    elseif imagen1(a+1,b-1)==7
        codcadena=3;
        n=3;
        i=a+1;
        j=b-1;

    elseif imagen1(a,b-1)==7
        codcadena=4;
        n=4;
        i=a;
        j=b-1;
    end
end


a=i;
b=j;
```

Anexo 8. Código de la función para determinar el píxel siguiente según la dirección $a_{i-1}=1$

```
function [a,b,codcadena,n,imagen1] =
posicion_uno(a,b,imagen1)

% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;

if imagen1(a-1,b+1)==15
        codcadena=7;
        n=100;
        imagen1(a-1,b+1)=7;
        i=a-1;
        j=b+1;

    elseif imagen1(a-1,b+1)==7
        codcadena=7;
        n=7;
        i=a-1;
        j=b+1;

    elseif imagen1(a,b+1)==15
        codcadena=0;
        n=100;
        imagen1(a,b+1)=7;
        i=a;
        j=b+1;

     elseif imagen1(a,b+1)==7
        codcadena=0;
        n=0;
        i=a;
        j=b+1;

    elseif imagen1(a+1,b+1)==15
        codcadena=1;
        n=100;
        imagen1(a+1,b+1)=7;
        i=a+1;
        j=b+1;

elseif imagen1(a+1,b+1)==15
    codcadena=1;
    n=1;
    i=a+1;
    j=b+1;

elseif imagen1(a+1,b)==15
    codcadena=2;
    imagen1(a+1,b)=7;
    n=100;
    i=a+1;
    j=b;

elseif imagen1(a+1,b)==15
    codcadena=2;
    n=2;
    i=a+1;
    j=b;

elseif imagen1(a+1,b-1)==15
    codcadena=3;
    imagen1(a+1,b-1)=7;
    n=100;
    i=a+1;
    j=b-1;

elseif imagen1(a+1,b-1)==7
    codcadena=3;
    n=3;
    i=a+1;
    j=b-1;

elseif imagen1(a,b-1)==15
    codcadena=4;
    imagen1(a,b-1)=7;
    n=100;
    i=a;
    j=b-1;
```

```
elseif imagen1(a,b-1)==7
    codcadena=4;
    n=4;
    i=a;
    j=b-1;

elseif imagen1(a-1,b-1)==7
    codcadena=5;
    n=5;
    i=a-1;
    j=b-1;

    end
a=i;
b=j;
```

Anexo 9. Código de la función para determinar el píxel siguiente según la dirección a_{i-1}=2.

```
function [a,b,codcadena,n,imagen1] =
posicion_dos(a,b,imagen1)

% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;

if imagen1(a+1,b+1)==15
        codcadena=1;
        n=100;
        imagen1(a+1,b+1)=7;
        i=a+1;
        j=b+1;

    elseif imagen1(a+1,b+1)==7
        codcadena=1;
        n=1;
        i=a+1;
        j=b+1;

    elseif imagen1(a+1,b)==15
        codcadena=2;
        imagen1(a+1,b)=7;
        n=100;
        i=a+1;
        j=b;

    elseif imagen1(a+1,b)==7
        codcadena=2;
        n=2;
        i=a+1;
        j=b;

    elseif imagen1(a+1,b-1)==15
        codcadena=3;
        imagen1(a+1,b-1)=7;
        n=100;
        i=a+1;
        j=b-1;

    elseif imagen1(a+1,b-1)==7
        codcadena=3;
        n=3;
        i=a+1;
        j=b-1;

    elseif imagen1(a,b-1)==15
        codcadena=4;


       imagen1(a,b-1)=7;
       n=100;
        i=a;
        j=b-1;

    elseif imagen1(a,b-1)==7
        codcadena=4;
        n=4;
        i=a;
        j=b-1;

    elseif imagen1(a-1,b-1)==15
        codcadena=5;
        imagen1(a-1,b-1)=7;
        n=100;
        i=a-1;
        j=b-1;

    elseif imagen1(a-1,b-1)==7
        codcadena=5;
        n=5;
        i=a-1;
        j=b-1;

    elseif imagen1(a-1,b)==7
        codcadena=6;
        n=6;
        i=a-1;
        j=b;
```

```
    end                                                    a=i;
                                                           b=j;
```

Anexo 10. Código de la función para determinar el píxel siguiente según la dirección $a_{i-1}=3$.

```
function [a,b,codcadena,n,imagen1] = posicion_tres(a,b,imagen1)

% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;
if imagen1(a+1,b+1)==15
        codcadena=1;
        n=100;
        imagen1(a+1,b+1)=7;
        i=a+1;
        j=b+1;
elseif imagen1(a+1,b+1)==7
        codcadena=1;
        n=1;
        i=a+1;
        j=b+1;
elseif imagen1(a+1,b)==15
        codcadena=2;
        imagen1(a+1,b)=7;
        n=100;
        i=a+1;
        j=b;
elseif imagen1(a+1,b)==7
        codcadena=2;
        n=2;
        i=a+1;
        j=b;
elseif imagen1(a+1,b-1)==15
        codcadena=3;
        imagen1(a+1,b-1)=7;
        n=100;
        i=a+1;
        j=b-1;
elseif imagen1(a+1,b-1)==7
        codcadena=3;
        n=3;
        i=a+1;
        j=b-1;
```

```
elseif imagen1(a,b-1)==15
        codcadena=4;
        imagen1(a,b-1)=7;
        n=100;
        i=a;
        j=b-1;
elseif imagen1(a,b-1)==7
        codcadena=4;
        n=4;
        i=a;
        j=b-1;
elseif imagen1(a-1,b-1)==15
        codcadena=5;
        imagen1(a-1,b-1)=7;
        n=100;
        i=a-1;
        j=b-1;
elseif imagen1(a-1,b-1)==7
        codcadena=5;
        n=5;
        i=a-1;
        j=b-1;
elseif imagen1(a-1,b)==15
        codcadena=6;
        n=100;
        imagen1(a-1,b)=7;
        i=a-1;
        j=b;
elseif imagen1(a-1,b)==7
        codcadena=6;
        n=6;
        i=a-1;
        j=b;
elseif imagen1(a-1,b+1)==7
        n=7;
        codcadena=7;
        i=a-1;
        j=b+1;
end

   a=i;
   b=j;
```

Anexo 11. Código de la función para determinar el píxel siguiente según la dirección $a_{i-1}=4$.

```
function [a,b,codcadena,n,imagen1] = posicion_tres(a,b,imagen1)

% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;

if imagen1(a+1,b-1)==15
        codcadena=3;
        n=100;
        imagen1(a+1,b-1)=7;
        i=a+1;
        j=b-1;

elseif imagen1(a+1,b-1)==7
        codcadena=3;
        n=3;
        i=a+1;
        j=b-1;

elseif imagen1(a,b-1)==15
        codcadena=4;
        imagen1(a,b-1)=7;
        n=100;
        i=a;
        j=b-1;

elseif imagen1(a,b-1)==7
        codcadena=4;
        n=4;
        i=a;
        j=b-1;


elseif imagen1(a-1,b-1)==15
        codcadena=5;
        imagen1(a-1,b-1)=7;
        n=100;
        i=a-1;
        j=b-1;
```

```
elseif imagen1(a-1,b-1)==7
        codcadena=5;
        n=5;
        i=a-1;
        j=b-1;


elseif imagen1(a-1,b)==15
        codcadena=6;
        imagen1(a-1,b)=7;
        n=100;
        i=a-1;
        j=b;

elseif imagen1(a-1,b)==7
        codcadena=6;
        n=6;
        i=a-1;
        j=b;

elseif imagen1(a-1,b+1)==15
        codcadena=7;
        imagen1(a-1,b+1)=7;
        n=100;
        i=a-1;
        j=b+1;

elseif imagen1(a-1,b+1)==7
        codcadena=7;
        n=7;
        i=a-1;
        j=b+1;

elseif imagen1(a,b+1)==7
        n=0;
        codcadena=0;
        i=a;
        j=b+1;
end
   a=i;
   b=j;
```

Anexo 12. Código de la función para determinar el píxel siguiente según la dirección a_{i-1}=5.

```
function [a,b,codcadena,n,imagen1] = posicion_cinco(a,b,imagen1)

% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;

if imagen1(a+1,b-1)==15
        codcadena=3;
        n=100;
        imagen1(a+1,b-1)=7;
        i=a+1;
        j=b-1;
elseif imagen1(a+1,b-1)==7
        codcadena=3;
        n=3;
        i=a+1;
        j=b-1;
elseif imagen1(a,b-1)==15
        codcadena=4;
        imagen1(a,b-1)=7;
        n=100;
        i=a;
        j=b-1;
elseif imagen1(a,b-1)==7
        codcadena=4;
        n=4;
        i=a;
        j=b-1;
elseif imagen1(a-1,b-1)==15
        codcadena=5;
        imagen1(a-1,b-1)=7;
        n=100;
        i=a-1;
        j=b-1;
elseif imagen1(a-1,b-1)==7
        codcadena=5;
        n=5;
        i=a-1;
        j=b-1;
```

```
elseif imagen1(a-1,b)==15
          codcadena=6;
         imagen1(a-1,b)=7;
          n=100;
          i=a-1;
          j=b;

elseif imagen1(a-1,b)==7
          codcadena=6;
          n=6;
          i=a-1;
          j=b;
elseif imagen1(a-1,b+1)==15
          codcadena=7;
          imagen1(a-1,b+1)=7;
          n=100;
          i=a-1;
          j=b+1;
elseif imagen1(a-1,b+1)==7
          codcadena=7;
          n=7;
          i=a-1;
          j=b+1;
elseif imagen1(a,b+1)==15
          codcadena=0;
          imagen1(a,b+1)=7;
          n=100;
          i=a;
          j=b+1;
elseif imagen1(a,b+1)==7
          codcadena=0;
          n=0;
          i=a;
          j=b+1;
elseif imagen1(a+1,b+1)==7
         codcadena=1;
          n=1;
          i=a+1;
          j=b+1;

end

   a=i;
   b=j;
```

Anexo 13. Código de la función para determinar el píxel siguiente según la dirección $a_{i-1}=6$.

```
function [a,b,codcadena,n,imagen1] =
posicion_seis(a,b,imagen1)

% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;

if imagen1(a-1,b-1)==15
        codcadena=5;
        n=100;
        imagen1(a-1,b-1)=7;
        i=a-1;
        j=b-1;

elseif imagen1(a-1,b-1)==7
        codcadena=5;
        n=5;
        i=a-1;
        j=b-1;

elseif imagen1(a-1,b)==15
        codcadena=6;
        imagen1(a-1,b)=7;
        n=100;
        i=a-1;
        j=b;

elseif imagen1(a-1,b)==7
        codcadena=6;
        n=6;
        i=a-1;
        j=b;

elseif imagen1(a-1,b+1)==15
        codcadena=7;
        imagen1(a-1,b+1)=7;
        n=100;
        i=a-1;
        j=b+1;

elseif imagen1(a-1,b+1)==7
        codcadena=7;
        n=7;
        i=a-1;
        j=b+1;

elseif imagen1(a,b+1)==15
        codcadena=0;
        imagen1(a,b+1)=7;
        n=100;
        i=a;
        j=b+1;

elseif imagen1(a,b+1)==7
        codcadena=0;
        n=0;
        i=a;
        j=b+1;

elseif imagen1(a+1,b+1)==15
        codcadena=1;
        imagen1(a+1,b+1)=7;
        n=100;
        i=a+1;
        j=b+1;

elseif imagen1(a+1,b+1)==7
        codcadena=1;
        n=1;
        i=a+1;
        j=b+1;

elseif imagen1(a+1,b)==7
        codcadena=2;
        n=2;
        i=a+1;
        j=b;

end
a=i;
b=j;
```

Anexo 14. Código de la función para determinar el píxel siguiente según la dirección $a_{i-1}=7$.

```
function [a,b,codcadena,n,imagen1] = posicion_siete(a,b,imagen1)

% Inicializacion de las variables
codcadena=0;
i=0;
j=0;
n=0;
if imagen1(a-1,b-1)==15
        codcadena=5;
        n=100;
        imagen1(a-1,b-1)=7;
        i=a-1;
        j=b-1;
elseif imagen1(a-1,b-1)==7
        codcadena=5;
        n=5;
        i=a-1;
        j=b-1;
elseif imagen1(a-1,b)==15
        codcadena=6;
        imagen1(a-1,b)=7;
        n=100;
        i=a-1;
        j=b;
elseif imagen1(a-1,b)==7
        codcadena=6;
        n=6;
        i=a-1;
        j=b;
elseif imagen1(a-1,b+1)==15
        codcadena=7;
        imagen1(a-1,b+1)=7;
        n=100;
        i=a-1;
        j=b+1;
elseif imagen1(a-1,b+1)==7
        codcadena=7;
        n=7;
        i=a-1;
        j=b+1;
elseif imagen1(a,b+1)==15
```

```
            codcadena=0;
            imagen1(a,b+1)=7;
            n=100;
            i=a;
            j=b+1;
 elseif imagen1(a,b+1)==7
            codcadena=0;
            n=0;
            i=a;
            j=b+1;
elseif imagen1(a+1,b+1)==15
            codcadena=1;
            imagen1(a+1,b+1)=7;
            n=100;
            i=a+1;
            j=b+1;
elseif imagen1(a+1,b+1)==7
            codcadena=1;
            n=1;
            i=a+1;
            j=b+1;
elseif imagen1(a+1,b)==15
            codcadena=2;
            imagen1(a+1,b)=7;
            n=100;
            i=a+1;
            j=b;
elseif imagen1(a+1,b)==7
            codcadena=2;
            n=2;
            i=a+1;
            j=b;
elseif imagen1(a+1,b-1)==7
            codcadena=3;
            n=3;
            i=a+1;
            j=b-1;
end

a=i;
b=j;
```

Anexo 15. Código de la función para el ajuste de las rectas por el método de los mínimos cuadrados.

```
function [m,longitud,posx,posy] =
ajuste_rectas_final(c,d,codigorecta,imagen)

% Inicializacion de las variables.

x=0;
y=0;
xo=0;
yo=0;
xo2=0;
xy=0;
m=0; % pendiente
n=0; % ordenada
px=0;
py=0;
posx=0;% coordenada x del pixel de fin de
cadena
posy=0;% coordenada y del pixel de fin de
cadena
a=0;
longitud=0;

% Programa para ajuste de rectas.

imagen=double(imagen);
[N,M]=size(codigorecta);
px=d;
py=c;

for i=1:M
    if codigorecta(i)==0
      x=px+1;
      y=py;
       if     (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
              a=a;
       elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
       elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
              a=a+1;
       elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
       elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
```

```
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
            a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
      end
      xo=x+xo;
      yo=y+yo;
      xo2=x^2+xo2;
      xy=x*y+xy;
      longitud=longitud+1;

   elseif codigorecta(i)==1
      x=px+1;
      y=py+1;
      if     (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
            a=a;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
```

```
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                              a=a+1;
        elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                              a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                              a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                              a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                              a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                              a=a+1;
        end
        xo=x+xo;
        yo=y+yo;
        xo2=x^2+xo2;
        xy=x*y+xy;
        longitud=longitud + sqrt(2);
    elseif codigorecta(i)==2
       x=px;
       y=py+1;
         if      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
```

```
            a=a;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
            a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
            a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
```

```
        end
        xo=x+xo;
        yo=y+yo;
        xo2=x^2+xo2;
        xy=x*y+xy;
        longitud=longitud+1;
    elseif codigorecta(i)==3
        x=px-1;
        y=py+1;
         if      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                a=a;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                            a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-
imagen(y-1,x))<=14) && (abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                            a=a+1;
        elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                            a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                            a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                            a=a+1;
```

```
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
        end
        xo=x+xo;
        yo=y+yo;
        xo2=x^2+xo2;
        xy=x*y+xy;
        longitud=longitud + sqrt(2);
    elseif codigorecta(i)==4
       x=px-1;
       y=py;
         if     (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                a=a;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
```

```
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
        end
        xo=x+xo;
        yo=y+yo;
        xo2=x^2+xo2;
        xy=x*y+xy;
        longitud=longitud+1;
    elseif codigorecta(i)==5
        x=px-1;
        y=py-1;
          if      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                a=a;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                a=a+1;
        elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
        elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
```

```
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
            a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
      end
      xo=x+xo;
      yo=y+yo;
      xo2=x^2+xo2;
      xy=x*y+xy;
      longitud=longitud + sqrt(2);
  elseif codigorecta(i)==6
     x=px;
     y=py-1;
       if     (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
            a=a;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
```

```
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
            a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
            a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
      end
      xo=x+xo;
      yo=y+yo;
      xo2=x^2+xo2;
      xy=x*y+xy;
      longitud=longitud+1;
  elseif codigorecta(i)==7
     x=px+1;
     y=py-1;
       if      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
```

```
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
            a=a;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
            a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif      (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
            a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))==15 || abs(imagen(y,x)-
imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))>0 || abs(imagen(y,x)-
imagen(y,x+1))<=14) && (abs(imagen(y,x)-
imagen(y+1,x))==15 || abs(imagen(y,x)-
imagen(y+1,x))==0) && (abs(imagen(y,x)-
imagen(y-1,x))==15 || abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0 || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))>0 || abs(imagen(y,x)-imagen(y-
1,x))<=14) && (abs(imagen(y,x)-imagen(y,x-
1))==15 || abs(imagen(y,x)-imagen(y,x-1))==0)
                        a=a+1;
      elseif  (abs(imagen(y,x)-
imagen(y,x+1))==15 || abs(imagen(y,x)-
imagen(y,x+1))==0) && (abs(imagen(y,x)-
```

```
imagen(y+1,x))>0 || abs(imagen(y,x)-
imagen(y+1,x))<=14) && (abs(imagen(y,x)-
imagen(y-1,x))==15|| abs(imagen(y,x)-
imagen(y-1,x))==0) && (abs(imagen(y,x)-
imagen(y,x-1))>0  || abs(imagen(y,x)-
imagen(y,x-1))<=14)
                        a=a+1;
        end
        xo=x+xo;
        yo=y+yo;
        xo2=x^2+xo2;
        xy=x*y+xy;
        longitud=longitud + sqrt(2);
    end
    px=x;
    py=y;
end
posx=y;
posy=x;

if a<=2

% Calculo de la pendiente m

m=-(M*xy-xo*yo)/(M*xo2-xo^2);

% Calculo del intercepto con el eje y

n=(yo*xo2-xo*xy)/(M*xo2-xo^2);
else
m=inf;
end
```

Anexo 16. Código de la función para hallar la media del código de cadena de los lados del polígono.

```
function media = media_lados(codigo)

media=0;
suma=0;
codigo1=codigo;
[N,M]=size(codigo1);
a=0;

for i=1:M
  if i<M && a~=1
    if codigo1(i)==0 && codigo1(i+1)==7
    i=1;
    a=1;
    suma=0;
    end
    if codigo1(i)==0 && a==1
         codigo1(i)=8;
    end

  else
      if codigo1(i)==0 && a==1
         codigo1(i)=8;
      end
  end
    suma=suma+codigo1(i);
end
media=suma/M;
```

Anexo 17. Valoración del experto 1.

Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

Valoración de expertos

Título del trabajo: **"Reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales".**

Autora: Daily Milanés Hermosilla

De acuerdo a la descripción ofrecida por la autora acerca del trabajo es posible concluir lo siguiente:

El reconocimiento de objetos es una temática muy estudiada en las últimas décadas motivada por aplicaciones en diferentes campos de la ciencia y la tecnología, en este caso específico en el trabajo con robots equipados con visión.

Entre las técnicas más utilizadas está la comparación con objetos ya establecidos en base de datos de manera que se puedan clasificar de acuerdo a sus características. Además, esta técnica es de aplicación constante en líneas de producción en la actualidad con el fin de establecer parámetros de calidad en los productos

Las mismas técnicas han sido utilizadas en la clasificación de objetos con información limitada requiriéndose para ello equipamiento con apreciable capacidad de memoria y rapidez en sus operaciones, siempre que sea necesario trabajar en tiempo real.

El desarrollar un algoritmo capaz de identificar polígonos regulares parcialmente ocultos es una buena idea, sobre todo porque no se requiere mucho tiempo ni memoria para el procesamiento de la información, lo que lo hace ideal para el trabajo en tiempo real.

La determinación del tipo de polígono regular a partir del sus ángulos internos, conociendo un lado de manera completa y los lados adyacentes de manera limitada, es válida para la mayoría de los polígonos regulares, aunque se excluyen los rombos.

Dada la limitada información que se dispone no es posible conocer cómo se diferencia, en la imagen tomada, la parte visible del polígono oculto del que lo oculta lo que sería más fácil si se dispusiera de dos imágenes tomadas desde ángulos diferentes, en lugar de solamente una.

Considero que el trabajo presentado es un punto inicial en el camino para la clasificación de objetos ocultos, que puede ser aplicado con este fin, aunque con limitaciones en lo relacionado con el tipo de objetos a identificar.

En trabajos futuros debe enfatizarse en el análisis de las posibles bondades del método propuesto y su comparación con los métodos de reconocimiento a partir de la utilización de bases de datos, teniendo en cuenta las capacidades actuales de memoria y las velocidades de procesamiento de información que se han logrado en equipamiento de uso común.

Considero, de acuerdo a la información recibida, que el trabajo reúne los requerimientos para ser defendido como opción para el título de máster.

Universidad Central de Las Villas 10 de Enero del 2011

Dr. José Rafael Abreu García

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

Valoración de expertos del trabajo:

"Reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales".

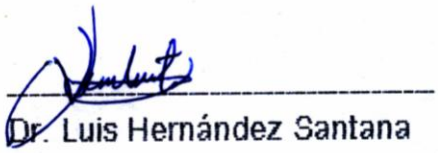
Autora: Daily Milanés Hermosilla

Valoración del trabajo: **"Reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales"** a presentar como Tesis de Maestría en la Universidad de Oriente.

El trabajo se desarrolla en una temática de gran actualidad al abordar el reconocimiento de imágenes en entornos no completamente estructurados, en este caso en el ámbito de figuras poligonales regulares de diferentes escalas de grises. Los resultados descritos son valiosos desde el punto de vista práctico y académico. Esta tarea es imprescindible como paso previo a cualquier aplicación práctica de control servovisual de robots, como está declarado en el trabajo. Para ello es muy importante lograr que los algoritmos de identificación visual sean ejecutados eficientemente, normalmente en un tiempo acorde con la frecuencia de captación de las imágenes por una cámara convencional. Ello permite usar el menor tiempo de muestreo posible en el lazo de control visual, con la menor afectación a la dinámica del lazo. Quizás este análisis es realizado en el trabajo, pero para su evaluación solo he contado con un resumen. Los resultados obtenidos son de interés de nuestro grupo de investigación que desea resolver problemas de control visual de manipuladores en ambientes no estructurados.

Considero que el desarrollo del trabajo centrado en imágenes 2D es adecuado, dada la gran cantidad de aplicaciones prácticas que se pueden desarrollar a partir de imágenes 2D. Recomiendo que además de reconocer el tipo de objeto se obtenga además la orientación del objeto oculto (quizás incluido también en el trabajo), muy importante también para tareas de control de robots.

Considero, a partir del resumen que he recibido, que el trabajo posee méritos para optar por el título de máster.



Dr. Luis Hernández Santana

Anexo 19. Valoración del experto 3.

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

Valoración del trabajo:

"Reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales".

Autora: Daily Milanés Hermosilla

El trabajo presentado por la autora se enmarca en un tema que aparece hoy día entre los más actuales y vigentes en la investigación a nivel mundial. En específico se trabaja en este campo para aplicaciones de diversa índole, siendo uno de las más desarrolladas las que tienen que ver con la robótica, tanto de tipo industrial, empresarial, civil o militar.

Es conocido que las técnicas de visión por computadora han tenido un avance vertiginoso en los últimos años, y la autora trata una componente de la misma, en este caso el reconocimiento de objetos de diferentes escalas de grises con determinadas características.

Los sistemas de visión por computadora utilizan las técnicas de reconocimiento de objetos para interpretar las propiedades de los mismos, normalmente comparan los objetos de estudio con sus bases de datos para tratar de identificarlos como objetos conocidos. Esta tecnología requiere de poderosos procesadores de imágenes con apreciable capacidad de memoria y rapidez para procesar la gran cantidad de datos existentes.

En este caso se tratan de identificar estos objetos por sus características propias, lo que redundaría en una utilización menor de memoria para el procesamiento de la información y en menor tiempo que las técnicas convencionales, lo que permitiría usar el menor tiempo de muestreo posible en el lazo de control visual y poder dedicar mayor tiempo de muestreo para el control en sí. Considero que en el trabajo se debe haber realizado una comparación entre los métodos convencionales y el utilizado aquí para demostrar la validez del mismo.

A pesar de ser un paso preliminar para la identificación de objetos, sería recomendable utilizar en la medida de las posibilidades dos cámaras para obtener una mayor información sobre los objetos requeridos, lo que redundaría en una mejor identificación de los mismos.

Aunque solo se cuenta con parte de la tesis, los resultados descritos son valiosos desde el punto de vista práctico y académico, y se aconseja que sus resultados sean aplicados en un futuro para el posible control servovisual, pues aunque no se tiene como objetivo principal de este trabajo, debe ser a lo que tribute la misma en el futuro.

Considero, a partir de la información recibida, que el trabajo reúne los requerimientos para optar por el título de Máster.

MSc. Diamir De Avila Rodríguez

Anexo 20. Valoración del experto 4

Santiago de Cuba, 17 de Enero del 2011

Valoración de Experto para la opción de Máster del trabajo titulado "**Reconocimiento de objetos poligonales regulares parcialmente ocultos en imágenes digitales**" de la autora Daily Milanés Hermosilla

De acuerdo a la descripción del trabajo ofrecida por la autora se presenta la siguiente valoración:

El reconocimiento de objetos a partir de su forma o apariencia es un campo de mucho auge en los últimos años, debido al desarrollo alcanzado por la tecnología y al creciente número de aplicaciones en la industria y otros sectores de la vida moderna. Específicamente en el área de robótica, en la cual se pretende la manipulación de objetos del entorno, es necesario la determinación de un conjunto de características de los objetos a manipular, como sus dimensiones, formas etc. Las técnicas de visión son especialmente útiles para estas tareas.

El reconocimiento o clasificación de objetos en una imagen a partir de su forma permite caracterizar el contorno de los mismos, con lo que se puede obtener información necesaria para su manipulación. Comúnmente el proceso de clasificación incluye la representación de los contornos a través de un de conjuntos de características de los mismos llamados descriptores, los cuales son seleccionadas a priori, como los momentos, descriptores de Fourier, curvatura, etc. Además es deseable que estos descriptores cumplan ciertas propiedades como la invariancia a transformaciones afines, traslación, rotación y escalado o combinaciones de estas.

El hecho que los objetos puedan aparecer parcialmente oculto complejiza la tarea, muchos de los descriptores usados cuando se detecta el contorno del objeto completamente no son aplicables en estos casos debido a que evalúan el contorno de manera global. Es por esto que los descriptores seleccionados en estos casos deben tener un carácter más local. En la literatura el enfoque aplicado comúnmente a la solución de esta problemática parte de encontrar coincidencias del objeto a clasificar con una colección de objetos de una base de datos. El algoritmo debe ser capaz de confirmar cuando el objeto ocluido es uno de los objetos en dicha base de datos. El proceso de búsqueda en bases de datos incrementa el tiempo de procesamientos de estos algoritmos, lo cual conspira contra la posibilidad de implementar muchos de ellos en tiempo real. Por otro lado los distintos tipos de descriptores pueden ajustarse en mayor o menor medida a los objetos de acuerdo al tipo de contorno. Todos estos aspectos hacen que el reconocimiento de objetos parcialmente ocultos sea un problema fundamental en el área de visión por ordenador. Este problema no ha sido lo suficientemente resulto aún, aunque se haya trabajado en este tema alrededor de veinte años.

En este trabajo se presenta una alternativa diferente a las propuestas encontradas en la literatura, a partir de la restringir del universo de objetos al conjunto de los polígonos regulares, siendo las clases, los distintos tipos de polígonos regulares de acuerdo al número de lados. Esta restricción permite la selección de una única característica (el ángulo interior

entre dos de sus lados) para caracterizar el polígono detectado. Por demás la relación entre esta característica y el tipo de polígono es una expresión analítica conocida, lo cual elimina *la necesidad de una base de datos, con el correspondiente ahorro de memoria y tiempo de* cómputo asociado al proceso de búsqueda. Es importante mencionar que la información necesaria en la imagen para obtener el ángulo es poco (solo un lado completamente visible y otro parcialmente visible) lo cual permite que el fracción de área solapada sea elevado y aún sea posible clasificar correctamente el polígono.

Entre otros aspectos novedosos propuestos por la autora se puede mencionar que el algoritmo de seguimiento de contorno aplicable en la detección y obtención del código de *cadena de cualquier objeto de contorno cerrado, no necesariamente para casos de* polígonos. El desarrollo de un algoritmo para dividir el código de cadena del contorno de un polígono regular en codigos que identifiquen cada lado de estos tipos de objetos. La utilización del método de mínimos cuadrados para ajustar rectas partiendo del código de cadenas.

Como recomendaciones para el trabajo futuro se sigiere la caracterización de método usando indicadores cuantitativos y su comparación con otros métodos en el estado arte. *También es necesaria una evaluación del método usando imágenes reales y el análisis del* efecto que ruido en estas puede provocar al algoritmo. Igualmente se propone la generalización o expansion a otros tipos de figuras geométricas en el plano y en 3D, así mismo como el posible reconocimiento más de una figura por imágenes.

Ing. Lic. Frank Sanabria Macias
Profesor del Centro de Estudios de Neurociencias y Procesamiento de Imágenes y Señales (CENPIS)
Facultad de Ingeniería Eléctrica
Universidad de Oriente