Centro de Investigación Científica y de Educación Superior de Ensenada

Segmentación de Múltiples Objetos en una Imagen basado en Contornos Activos Genéticos

TESIS
MAESTRIA EN CIENCIAS

RAFAEL FELIPE NEVAREZ LOPEZ

ENSENADA BAJA CFA, MEXICO SEPTIEMBRE DE 2006

TESIS DEFENDIDA POR

Rafael Felipe Nevarez López

Y aprobada por el siguiente comité:

Dr. Gustavo Olague Caballero

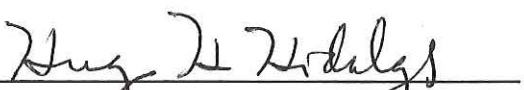
Director del Comité

Dr. Rafael de Jesús Kelly Martínez

Miembro del Comité

Dr. Pedro Negrete Regagnon

Miembro del Comité

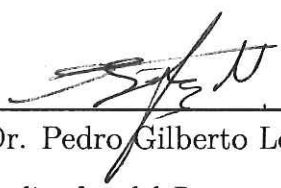


Dr. Hugo Homero Hidalgo Silva

Miembro del Comité

M.C. José Luis Briseño Cervantes

Miembro del Comité



Dr. Pedro Gilberto López Mariscal

Coordinador del Programa de Posgrado en Ciencias de la Computación

Dr. Raúl Ramón Castro Escamilla

Director de Estudios de Posgrado

26 de Septiembre del 2006

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA

POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN

Segmentación de Múltiples Objetos en una Imagen basado en Contornos Activos Genéticos

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS

Presenta:

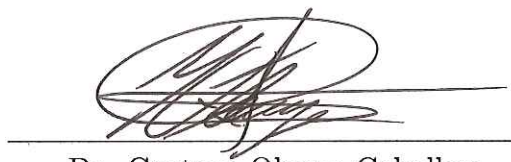
Rafael Felipe Nevarez López

Ensenada, Baja California, México. Septiembre del 2006.

RESUMEN de la tesis de **Rafael Felipe Nevarez López**, presentada como requisito parcial para obtener el grado de MAESTRO EN CIENCIAS en CIENCIAS DE LA COMPUTACIÓN. Ensenada, Baja California. Septiembre del 2006.

Segmentación de Múltiples Objetos en una Imagen basado en Contornos Activos Genéticos

Resumen aprobado por:



Dr. Gustavo Olague Caballero

Director de Tesis

Los modelos de Contornos Activos o *Snakes*, como se conocen, fueron propuestos por Kass *et al.* (1988) y su objetivo principal es la segmentación de objetos de interés en una imagen. Se puede definir una snake como una curva envolvente dinámica, la cual en su esfuerzo por minimizar su energía es atraída hacia características de los objetos, como por ejemplo la intensidad de los pixeles, líneas, esquinas o bordes. Se ha demostrado que esta técnica tiene muchas ventajas sobre otros métodos de segmentación. Sin embargo, existe un número de limitaciones asociado con esta aproximación, como la inicialización, la existencia de múltiples mínimos locales y la selección de los parámetros de elasticidad. Para confrontar estas limitaciones en este trabajo se desarrolla un algoritmo genético para la minimización de la energía de la snake. Además se propone una nueva manera de inicializar snakes utilizando puntos de interés, que permite a la snake tener un mejor desempeño, una mayor rapidez de convergencia, y que puede ser utilizado por cualquier otro algoritmo de minimización. De esta manera la snake genética es capaz de segmentar múltiples objetos en una imagen, con la posibilidad de realizar los cálculos en paralelo para una mayor velocidad de segmentación.

Palabras clave: Contornos Activos o Snakes, Algoritmos Genéticos, Segmentación, Puntos de Interés.

ABSTRACT of the thesis presented by **Rafael Felipe Nevarez López**, as a partial requirement to obtain the MASTER IN SCIENCE degree in COMPUTER SCIENCES. Ensenada, Baja California. September 2006.

Multiple Object Image Segmentation based on Genetic Active Contours

Abstract approved by:

Dr. Gustavo Olague Caballero

Thesis director

Active Contour Models or *Snakes*, like they are also known, were proposed by Kass *et al.* (1988) and its main objective is the image segmentation of interest objects. A Snake is a dynamically evolving curve, which in its effort to minimize its energy gets attracted toward objects features such as pixels intensities, lines, corners or edges. It has been proved that this technique has many advantages over other segmentation methods. However, there are a number of drawbacks associated with this approach, such as the initialization, the existence of multiple local minima, and the selection of elasticity parameters. In this work a genetic algorithm is used for the snake energy minimization in order to confront this limitations. Futhermore a new way for snake initialization is proposed using interest points, that allow the snake to have a better performance, a more quickly convergence, and that it can be used by any other minimization algorithm. In this way the genetic snake is capable to segment multiple objects in an image, with the possibility of parallel computing for greater segmentation velocity.

Keywords: Active Contours or Snakes, Genetic Algorithms, Segmentation, Interest Points.

Dedicado a mis padres

Rafael Felipe Nevarez Galaz
y
Sylvia López Cazares

Agradecimientos

Primeramente agradezco a Dios y a la Virgen de Guadalupe por todo lo que me han dado.

Agradezco a mi esposa Diana, por continuar amándome a pesar que en el último par de años pasé más tiempo sentado en la computadora que con ella.

A mis padres por su apoyo incondicional a pesar de la distancia.

Al Dr. Gustavo Olague, por compartir sus conocimientos y por la ayuda brindada a lo largo de la realización de este trabajo.

A los miembros del Comité de Tesis, por su interés y aportaciones a este trabajo.

Al Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE) y al Consejo Nacional de Ciencia y Tecnología (CONACYT), por darme las facilidades para estudiar en esta institución.

A mis compañeros del laboratorio *EvoVisión*: Cynthia, Leonardo, Eva, Alina, Lupita, León, Juan, Cesar y Eddie.

A los investigadores y mis compañeros de CICESE que han contribuido a este trabajo probablemente de maneras desconocidas para ellos.

Ensenada, México
26 de Septiembre del 2006.

Rafael Felipe Nevarez López

Tabla de Contenido

Capítulo — Página

Resumen ii
Abstract iii
Lista de Figuras ix
Lista de Tablas xii
I Introducción 1
I.1 Visión por computadora . . . 2
I.1.1 Relación con otras áreas . . . 3
I.1.2 Componentes de un sistema de visión . . . 3
I.2 Motivación . . . 7
I.3 Antecedentes . . . 9
I.4 Alcances de esta tesis . . . 12
I.4.1 Objetivo general . . . 13
I.4.2 Objetivos específicos . . . 13
I.5 Organización de esta tesis . . . 13
II Segmentación de Imágenes 15
II.1 Detección de discontinuidades . . . 15
II.1.1 Filtrado espacial . . . 16
II.2 Detección de bordes . . . 18
II.2.1 Gradiente . . . 22
II.2.2 Laplaciano . . . 25
II.3 Conexión de bordes y detección de contornos . . . 27
II.4 Segmentación basada en regiones . . . 28
II.4.1 Crecimiento de regiones . . . 29
II.5 Conclusiones . . . 30
III Contornos Activos 32
III.1 Representación de curvas . . . 32
III.1.1 Representación explícita . . . 33
III.1.2 Representación implícita . . . 34
III.1.3 Representación paramétrica . . . 35
III.2 Splines . . . 36
III.2.1 Interpolación . . . 38
III.2.2 Aproximación . . . 38
III.2.3 Condiciones de continuidad paramétrica . . . 39
III.2.4 Especificaciones de spline . . . 41
III.2.5 Spline cúbica natural . . . 43
III.2.6 Curvas de Bézier . . . 44
III.2.7 B-splines . . . 49

Tabla de Contenido (Continuación)

Capítulo Página

III.3 Información *a priori* . . . 52
III.4 Modelo matemático de la snake . . . 53
III.4.1 Energía interna . . . 54
III.4.2 Energía externa . . . 56
III.5 Funcionamiento de la snake . . . 57
III.6 Trabajo previo . . . 58
III.6.1 Kass . . . 58
III.6.2 Williams y Shah . . . 59
III.6.3 Xu y Prince . . . 61

IV Puntos de Interés **63**
IV.1 Detector de Beaudet . . . 67
IV.2 Detector de Kitchen y Rosenfeld . . . 69
IV.3 Detector de Dreschler y Nagel . . . 70
IV.4 Detector de Wang y Brady . . . 71
IV.5 Detector de Harris y Stephens . . . 72
IV.6 Detector de Förstner . . . 73
IV.7 Detector de Trujillo y Olague . . . 75
IV.8 Conclusiones . . . 77

V Introducción a la Computación Evolutiva **78**
V.1 Paradigmas tradicionales de búsqueda y optimización vs paradigmas de cómputo evolutivo . . . 78
V.2 La evolución natural . . . 80
V.3 Técnicas evolutivas . . . 82
V.3.1 Algoritmos genéticos . . . 82
V.3.2 Programación evolutiva . . . 83
V.3.3 Estrategias de evolución . . . 85
V.3.4 Programación genética . . . 87
V.4 Estructura general . . . 90
V.4.1 Inicialización de la población . . . 91
V.4.2 Evaluación . . . 91
V.4.3 Selección . . . 92
V.4.4 Reproducción . . . 93
V.4.5 Criterio de terminación . . . 103
V.5 Conclusiones . . . 104

VI Inicialización de Contornos Activos utilizando Puntos de Interés **106**
VI.1 Puntos de interés como energía externa . . . 107
VI.2 Casco convexo . . . 113
VI.3 Mínimo círculo contenedor . . . 114

Tabla de Contenido (Continuación)

Capítulo Página

VI.3.1 Experimentación 115
VI.4 Agrupamiento k-medias 117
VI.4.1 Experimentación 119
VI.5 Conclusiones 121
VII Algoritmos Genéticos para la minimización de Snakes **123**
VII.1Formulación 123
VII.2Experimentación 124
VII.3Conclusiones 130
VIII Conclusiones y trabajo futuro **132**
Bibliografía **135**
A Herramientas Computacionales **139**
A.1 OpenCV 139
A.1.1 Instalación 142
A.1.2 Instalación del paquete ffmpeg 144
A.1.3 Cómo compilar programas con OpenCV 145
A.2 IDL 147
A.3 VisLib 148
A.3.1 Instalación 149
A.4 MATLAB 150
A.5 Biblioteca CImg 151
A.5.1 Instalación 151
B Cámara Logitech Quickcam Pro 4000 **153**
B.1 Instalación 154
B.2 SETPWC 156
B.3 Visión estéreo 157

Lista de Figuras

Figura		Página
1	Etapas de la Visión por Computadora.	4
2	Cámara Logitech Quickcam Pro 4000.	7
3	Coordenadas de la imagen. El pequeño recuadro dentro de la imagen muestra una región de 3×3, con su centro en las coordenadas (x, y).	8
4	Animación de una caricatura de un gato. Diferentes snakes siguen las emociones faciales de un actor.	10
5	Monitoreo de tráfico. Con el seguimiento automático de autos, se pueden obtener alertas de accidentes o calcular la velocidad de los autos.	11
6	Representación de una máscara general de tamaño 3×3 para filtrado espacial.	17
7	Una región de tamaño 3×3 de una imagen, representada en términos de variables z que indican el nivel de gris.	17
8	Modelos de borde.	19
9	Dos regiones separadas por un borde vertical, el perfil de nivel de gris y la primera y segunda derivada del perfil.	21
10	Varias máscaras para calcular el gradiente.	25
11	a)Imagen original. b)$\|G_x\|$, componente del gradiente en la dirección x. c)$\|G_y\|$, componente del gradiente en la dirección y. d)Gradiente de la Imagen, $\|G_x\| + \|G_y\|$.	26
12	Máscaras para aproximar el Laplaciano.	27
13	Representación de un objeto tridimensional por pedazos de planos.	33
14	Dos tipos de representación paramétrica para un círculo.	35
15	Ejemplo de spline en dibujo mecánico. Una tira de plástico o metal moldeada por tres pesos ubicados en posiciones específicas.	36
16	Conjunto de siete puntos de control que se interpolan con secciones polinómicas continuas en el sentido de la pieza.	38
17	Conjunto de seis puntos de control que se aproximan con secciones polinómicas continuas en el sentido de la pieza.	39
18	Construcción por piezas de una curva al unir dos segmentos de curva utilizando distintos órdenes de continuidad: a) C^0, b) C^1 y c) C^2.	40
19	Ejemplos de tres curvas de Bézier bidimensionales que se generaron a partir de tres, cuatro y cinco posiciones de puntos de control.	46
20	Las cuatro funciones de combinación de Bézier para las curvas cúbicas.	48
21	Modificación local de una curva B-spline. Al cambiar uno de los puntos de control en la parte (a) se produce la curva de la parte (b), que se modifica sólo en las áreas cercanas al punto de control alterado.	51
22	Imagen de ejemplo que ilustra la importancia del conocimiento *a priori*.	53

Lista de Figuras (Continuación)

Figura Página

23 Efecto de minimizar la energía interna: a) La energía elástica $E_{elastica}$. b) La energía de curvatura $E_{curvatura}$. c) Ambas energías $E_{int} = E_{elastica} + E_{curvatura}$. 55
24 Funcionamiento de una snake: a) Inicialización. b) Deformación. c) Contorno final. 57
25 Ejemplo de la detección de puntos de interés. Los círculos blancos representan los 529 puntos de interés encontrados por el detector de Harris. 64
26 Ejemplo de la respuesta del detector de puntos de interés de Beaudet. . 68
27 Ejemplo de la respuesta del operador de Kitchen y Rosenfeld. 70
28 Ejemplo de la respuesta al detector de puntos de interés de Wang y Brady. 72
29 Ejemplo de la respuesta al detector de puntos de interés de Harris y Stephens. 73
30 Ejemplo de la detección de puntos de interés utilizando el operador de Förstner. 74
31 Ejemplo de la detección de puntos de interés de los operadores evolucionados con Programación Genética. 77
32 Operadores de mutación. 97
33 Operadores de cruzamiento. 100
34 Operador de repartición. 103
35 a) Magnitud del gradiente $|G_x|+|G_y|$. b) Detector de Harris. c) Detector IPGP1. d) Detector Beaudet. 108
36 Snake atraída hacia puntos de interés. a) Puntos de interés detectados por IPGP1. b) Inicialización. c) Deformación. d) Contorno final. 109
37 Snake atraída hacia puntos de interés. a) Puntos de interés detectados por Harris. b) Inicialización. c) Deformación. d) Contorno final. 111
38 Snake atraída hacia puntos de interés. a) Imagen original. b) Puntos de interés detectados por Harris. c) Inicialización. d) Contorno final. . . . 112
39 Casco convexo, analogía con una banda elástica. 114
40 a) Inicialización de una snake utilizando información *a priori*. b) Contorno final. 116
41 a) Inicialización de una snake utilizando el método de mínimo círculo contenedor de los puntos de interés. b) Contorno final. 117
42 a) Imagen original. b) Agrupamiento de los puntos de interés utilizando k-medias ($k = 3$). 120
43 a) Inicialización del mínimo círculo contenedor en la primera pieza. b) Contorno final. 120

Lista de Figuras (Continuación)

Figura Página

44 a) Inicialización del mínimo círculo contenedor en la segunda pieza. b) Contorno final. 121
45 a) Inicialización del mínimo círculo contenedor en la tercera pieza. b) Contorno final. 121
46 Resultado de una snake genética en una imagen sintética. 127
47 Resultado de una snake genética en una imágen real adquirida por una cámara web. 128
48 Resultado de una snake genética en una imagen sintética utilizando una inicialización por puntos de interés y el mínimo círculo contenedor. . . 129
49 Resultado de una snake genética en una imagen real adquirida por una cámara web utilizando una inicialización por puntos de interés y el mínimo círculo contenedor. 130
50 Resultado de una snake genética en una imagen real adquirida por una cámara web utilizando una inicialización por puntos de interés, la técnica de k-medias y el mínimo círculo contenedor. 131

Lista de Tablas

Tabla		Página
I	Últimas versiones estables de OpenCV para Linux.	142
II	Algunas funciones de la utilería SETPWC.	157

Capítulo I

Introducción

El conocimiento del mundo que nos rodea es gracias a nuestros sentidos. La visión es el más avanzado y complejo de nuestros sentidos, que nos provee de una gran cantidad de información acerca del medio ambiente; por lo que no cabe duda que las imágenes juegan un papel muy importante en la percepción humana. A través de la vista podemos conocer los objetos que nos rodean, sus características (forma, tamaño, color), su posición, etc. De esta manera, no es sorprendente que grupos de científicos hayan intentado darle a las máquinas el sentido de la vista desde que las primeras computadoras aparecieron, con la finalidad de que sistemas autónomos puedan interactuar con su medio ambiente. Es así como nace la disciplina de la *Visión por Computadora.*

El campo de la Visión por Computadora tiene como interés principal permitir que las computadoras puedan *ver.* Las tareas de alto nivel de Visión por Computadora tratan de emular distintos tipos de capacidades de visión humana, de tal manera que presenta a la comunidad de investigación un conjunto extenso de problemas abiertos, difíciles e interesantes. Sin embargo, hoy en día estamos muy distantes de haber planteado siquiera una solución concreta a la problemática.

El mundo en que los robots interactuan es tridimensional y dinámico, y se percibe a través de cámaras digitales las cuales modelan el mundo real utilizando principios de geometría. Los modelos matemáticos de formas, tamaños de objetos son algunos de los fundamentos de esta disciplina, que intenta resolver problemas computacionales relacionados con la visión.

Para poder *ver* objetos tal como una persona lo hace, evidentemente fácil, los sistemas de Visión por Computadora deben ser capaces de analizar la forma y movimiento de los objetos en tiempo real. La manera más simple es utilizar la *segmentación* de imágenes para separar los objetos en una imagen. Existen diversos métodos de segmentación; sin embargo, ninguno de ellos resuelve el problema general, de manera que pueda afrontar cualquier circunstancia.

A finales de los años ochenta Michael Kass y sus colegas introducen un nuevo modelo matemático conocido como *Contornos Activos* o *Snakes*. Formularon un modelo visual combinando geometría y física, en particular la dinámica de curvas elásticas, que unifica la representación de la forma y movimiento de objetos. Con lo que los sistemas de visión pueden ser capaces de interpretar secuencias de video en términos de objetos rígidos y no rígidos que se mueven ante la cámara.

Los modelos de contornos activos han ganado popularidad desde entonces, por la gran cantidad de aplicaciones en muchos problemas que conciernen a la segmentación, seguimiento de objetos, descripción de la forma, detección de objetos, obtención de los contornos de los objetos, por mencionar algunos.

I.1 Visión por computadora

La meta final de la Visión por Computadora es utilizar la computadora para emular la visión humana, incluyendo el aprendizaje y la capacidad de realizar inferencias y tomar acciones basadas en entradas visuales. La Visión por Computadora también se conoce como Visión Artificial ya que es una rama de la Inteligencia Artificial, la cual tiene como objetivo emular la inteligencia humana.

Por lo tanto podemos definir la Visión por Computadora como la disciplina cuyo objetivo es *proveer del sentido de la vista a sistemas autónomos para que puedan interactuar de forma eficiente en ambientes complejos* (Forsyth y Ponce, 2002).

I.1.1 Relación con otras áreas

En la actualidad no existe un acuerdo general entre los autores para definir los límites entre la Visión por Computadora y otras áreas relacionadas como el Procesamiento y el Análisis de Imágenes. Una manera útil de hacer ésto, es considerar tres tipos de procesos de cómputo: procesos de bajo, medio y alto nivel.

Los procesos de bajo nivel involucran operaciones primitivas como el tratamiento de la imagen para reducir el ruido, realce del contraste, y la restauración. Un proceso de bajo nivel se caracteriza por el hecho de que la entrada y la salida son imágenes.

El procesamiento de nivel medio involucra tareas como la segmentación (particionar una imagen en regiones u objetos) por diversos métodos, la descripción de los objetos y la clasificación o reconocimiento de cada objeto. El procesamiento de nivel medio se caracteriza por el hecho de que sus entradas generalmente son imágenes, pero sus salidas son atributos extraídos de estas imágenes como bordes, contornos, o la identidad de cada objeto.

Finalmente los procesos de alto nivel involucran funciones cognitivas acerca de los objetos reconocidos. De esta manera podemos decir que la Visión por Computadora se puede dividir en etapas que van desde que una imagen es adquirida hasta el objetivo final, que es lograr la percepción del mundo (ver Figura 1). Ésto no implica que todos los procesos deben ser aplicados a una imagen, sino que la intención es dar una idea de las metodologías que pueden ser aplicadas a las imágenes para diferentes propósitos y objetivos.

I.1.2 Componentes de un sistema de visión

Como se observa en la Figura 1 el componente básico para un sistema de visión es un sistema de adquisición de imágenes digitales, la cual involucra principios de la *geometría proyectiva*. Una imagen puede ser definida como una función bidimensional $f(x, y)$,

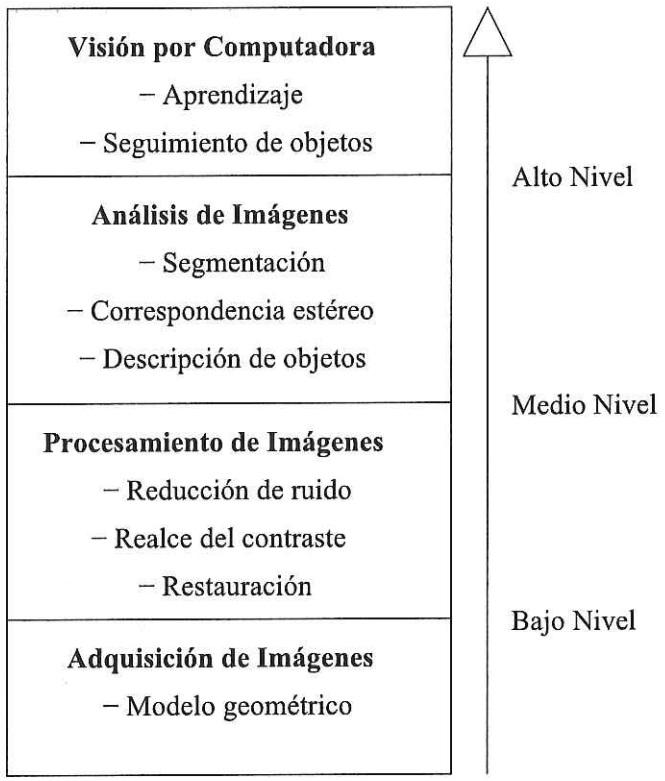


Figura 1: Etapas de la Visión por Computadora.

donde x y y son coordenadas en el plano espacial, y la amplitud de f en cualquier par de coordenadas (x, y) se llama intensidad de nivel de gris en ese punto. Cuando (x, y) y la amplitud de los valores de f son finitos (cantidades discretas), la imagen recibe el nombre de *imagen digital.* Por ejemplo, en una imagen de 8 bits, el mínimo nivel de gris es cero que representa pixeles negros de la imagen y el máximo nivel de gris es 255 que representa los pixeles blancos.

Es importante mencionar que un sistema de visión no sólo se enfoca al problema de la extracción de información desde sensores visuales, sino también tiene como propósito el resolver cómo debe ser extraída, representada y empleada dicha información. A continuación se enumeran los componentes básicos de un sistema de visión típico de propósito general.

1. **Sensor**: dos elementos se requieren para adquirir una imagen digital. El primero es un dispositivo físico el cual es sensible a la energía radiada por un objeto que se desea capturar en una imagen. Y el segundo es un dispositivo *digitalizador*, el cual convierte la salida del dispositivo físico sensor en una forma digital. Por ejemplo, en una cámara digital, los sensores producen una salida eléctrica proporcional a la intensidad de luz y el digitalizador convierte esta salida en datos digitales.

2. **Procesador de imágenes**: generalmente consiste en algún tipo de hardware que sirve para realizar algunas operaciones primitivas. Como por ejemplo, una unidad lógica aritmética (ALU), la cual puede realizar operaciones aritméticas y lógicas en paralelo sobre la imagen completa. Una manera en que una ALU se puede utilizar, es en el promediado de imágenes con el propósito de reducción de ruido, el cual se realiza tan rápidamente conforme se adquieren las imágenes.

3. **Computadora**: algunas computadoras diseñadas especialmente para aplicaciones dedicadas se utilizan para alcanzar el nivel requerido de desempeño; sin embargo, en un sistema de propósito general, cualquier computadora común puede ser utilizada. En este trabajo se utiliza una computadora marca Dell, modelo Optiplex GX270, con un procesador Pentium IV de 3.2 Ghz de velocidad, y 1GB de memoria RAM.

4. **Software**: los programas para tratamiento de imágenes consisten en módulos especializados que pueden realizar tareas específicas. Un paquete bien diseñado también incluye la capacidad para que un usuario pueda escribir código, para que al menos pueda utilizar los módulos especializados. En el desarrollo de esta tesis se emplea el sistema operativo Linux Fedora Core 4, y el lenguaje de programación C++. Información a cerca de los paquetes computacionales que se utlizan puede encontrarse en el Apéndice A.

5. **Sistema de almacenamiento**: una imagen con dimensiones 1024×1024 pixeles, en donde la intensidad de cada pixel está representado por una cadena de 8 bits, requiere un megabyte de espacio de almacenamiento si la imagen no se comprime. Cuando se requiere trabajar con una gran cantidad de imágenes, un dispositivo de almacenamiento adecuado toma mucha importancia. Los sistemas de almacenamiento pueden clasificarse en tres categorías principales: un almacenamiento de corto plazo para utilizar durante el procesamiento, un almacenamiento en línea (discos duros o dispositivos ópticos) para un acceso relativamente rápido, y un almacenamiento en archivo (cintas magnéticas) que se caracteriza por un acceso poco frecuente. Algunos dispositivos de almacenamiento de corto plazo son la memoria de la computadora y las tarjetas especializadas denominadas *memorias temporales* (en inglés frame buffers). Éste último puede almacenar una o más imágenes a las que se puede acceder con rapidez, usualmente a velocidades de video (30 imágenes completas por segundo), y además permite la aproximación (*zoom*), desplazamientos verticales (*scroll*) y horizontales (*pan*) prácticamente de manera instanánea.

6. **Sistema de despliegue**: en la actualidad el principal sistema de despliegue son los monitores, generalmente a color aunque también pueden ser monocromáticos. Los monitores son manejados por tarjetas de gráficos que son una parte integral de una computadora.

El dispositivo que se utiliza en este trabajo para la adquisición de imágenes, es una cámara web Logitech Quickcam Pro 4000 (ver Figura 2), la cual puede generar una imagen a color con una resolución máxima de 640 × 480 pixeles y secuencias de video a una velocidad máxima de 30fps (frames por segundo) con una resolución de 320 × 240 pixeles. Más información a cerca de las características, funcionamiento e instalación de

esta cámara puede ser encontrada en el Apéndice B.

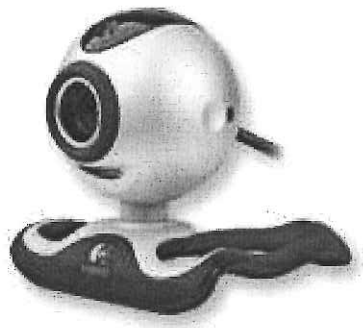


Figura 2: Cámara Logitech Quickcam Pro 4000.

Es muy importante tener presente la forma en que se organizan los pixeles ya que algunas veces se requieren hacer ciertos ajustes para adecuar la información a nuestras necesidades. En el presente trabajo la convención para las coordenadas de las imágenes es de arriba hacia abajo y de izquierda a derecha; es decir, el primer pixel se encuentra en la esquina superior izquierda y el último en la esquina inferior derecha (ver Figura 3).

I.2 Motivación

El problema de segmentación de imágenes se ha estudiado por mucho tiempo en el campo de Análisis de Imágenes. La segmentación se refiere a la idea de identificar objetos en una imagen; sin embargo, la segmentación automática es una de las tareas más difíciles en el procesamiento de imágenes debido a la gran variedad de formas de objetos y calidad de las imágenes. La segmentación juega un papel muy importante para la Visión por Computadora ya que una buena segmentación significa un gran paso hacia

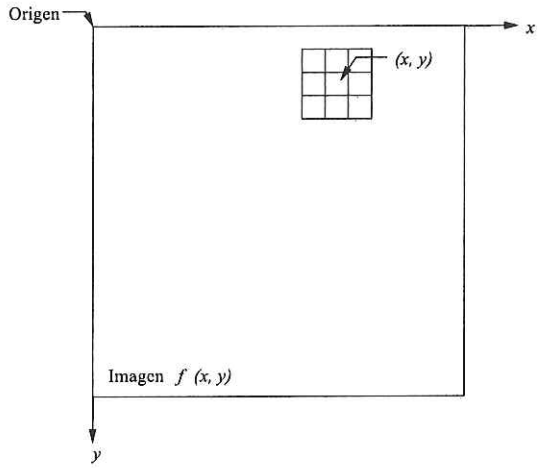


Figura 3: Coordenadas de la imagen. El pequeño recuadro dentro de la imagen muestra una región de 3×3, con su centro en las coordenadas (x, y).

la solución exitosa de problemas de más alto nivel que requieren la identificación individual de objetos, asegurando un mejor desempeño, lo cual tiene muchas aplicaciones importantes como el seguimiento y reconocimiento de objetos.

Varias aproximaciones han tratado de resolver el problema de segmentación. Algunos involucran la utilización de operadores de detección de discontinuidades como puntos, líneas y bordes, mediante el uso de filtros como Prewitt y Sobel. Sin embargo, sus resultados no son fácilmente interpretables para un sistema autónomo por lo que la mayoría de estos métodos necesitan de un posprocesamiento como la transformada Hough o técnicas de teoría de grafos.

Otra aproximación para la segmentación se basa en el crecimiento de región, que opera agrupando pixeles con atributos similares. Dos pixeles son unidos si sus atributos como el nivel de gris, color o textura son suficientemente similares. Cada pixel se compara con sus vecinos inmediatos, si el criterio de similitud se cumple entonces el pixel que se prueba se declara perteneciente a la región y todos lo atributos de la región son actualizados.

Otras aproximaciones incluyen el método de plantillas deformables que consiste en comparar directamente la apariencia de una imagen dada con una plantilla de referencia. La plantilla inicial, que es representativa de la forma que se trata de localizar, se deforma progresivamente con el fin de minimizar un funcional de energía. El criterio de comparación es tan ideal que el mínimo es alcanzado cuando la plantilla converge sobre el objeto de interés. Sin embargo, esta aproximación es útil sólo cuando se busca una forma específica. En el caso más general esta aproximación no es de mucha ayuda.

Por lo cual Kass *et al.* (1988) proponen una nueva aproximación para la segmentación de imágenes llamado Modelos de Contornos Activos. Un Contorno Activo o *Snake*, como se conocen, tiene como objetivo la construcción de una banda elástica que iterativamente se va ajustando a las fronteras de los objetos (Branch y Olague, 2001). Por eso, las snakes prometen una técnica de segmentación que no tiene una inclinación hacia una forma específica. Es por eso aplicable a una gran variedad de circunstancias, lo cual es una de sus ventajas significativas.

A la snake se le asocia un funcional de energía el cual es construído de tal manera que alcanza un valor mínimo cerca de los bordes de los objetos. Por lo tanto, el problema de localizar los contornos de un objeto es reducido a un problema de minimización de energía, el cual se puede resolver con técnicas de optimización.

Desde el trabajo de Kass, las snakes han recibido una amplia popularidad en la comunidad de Visión por Computadora. Rápidamente encontraron su utilización en un gran número de áreas como la visualización médica, animación (ver Figura 4), vigilancia (ver Figura 5), visión robótica, edición de video, y muchas otras (Blake y Isard, 1998).

I.3 Antecedentes

Los contornos activos o snakes se utilizan por la comunidad de Visión por Computadora en una gran variedad de aplicaciones y áreas de investigación; sin embargo, el modelo

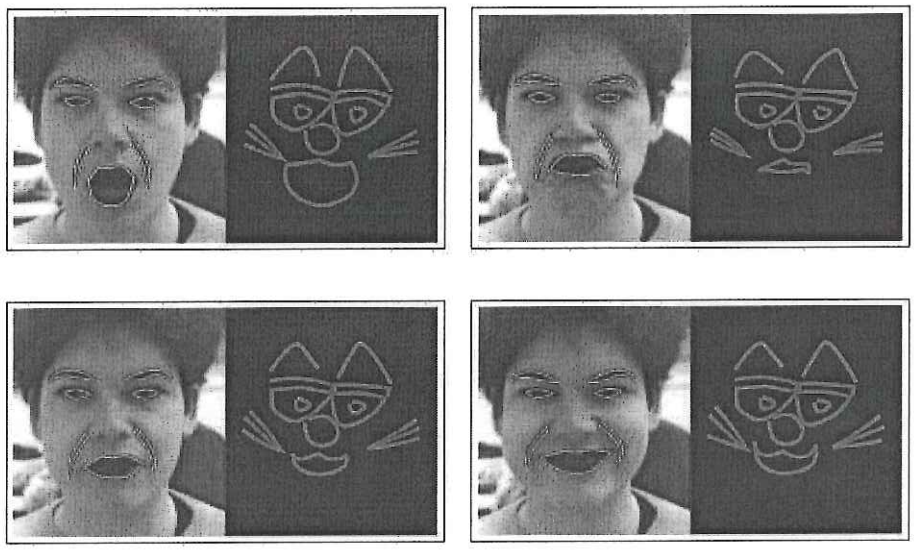


Figura 4: Animación de una caricatura de un gato. Diferentes snakes siguen las emociones faciales de un actor.

tiene algunas limitaciones. Algunos de los problemas con esta aproximación son:

1. Inicialización: al principio los modelos de contornos activos se diseñaron como modelos interactivos (Kass *et al.*, 1988), de tal forma que un usuario inicializa la snake cerca del objeto de interés. Por lo que, en aplicaciones no interactivas, se debe inicializar el contorno lo suficientemente cerca de la estructura de interés para garantizar un buen desempeño.

2. La velocidad de los algoritmos de minimización: Kass utiliza *cálculo de variaciones* para minimizar la energía de las snakes lo cual es eficiente para imágenes estáticas. Sin embargo, en aplicaciones de tiempo real, como el seguimiento de objetos en una secuencia de video, los algoritmos de optimización deben ser lo suficientemente rápidos.

3. Segmentación de objetos complejos: las snakes tradicionales sólo pueden tratar con objetos simples geométricamente y topológicamente, y son inadecuadas para

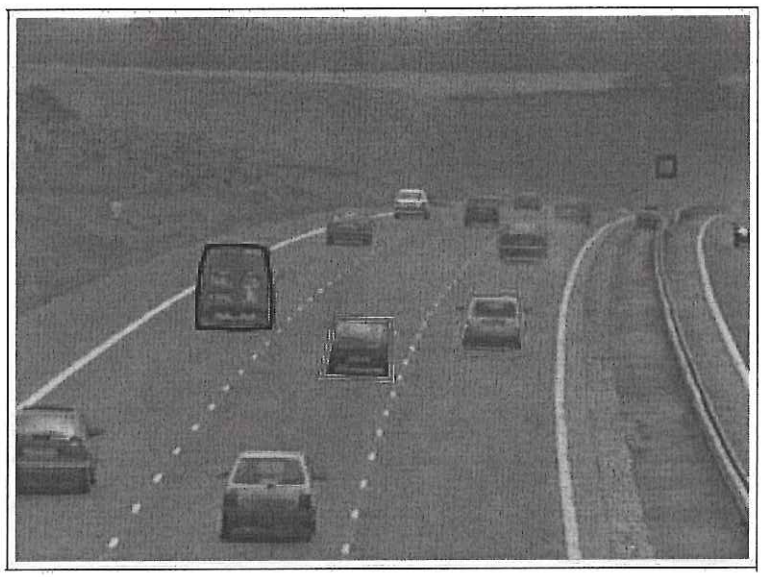


Figura 5: Monitoreo de tráfico. Con el seguimiento automático de autos, se pueden obtener alertas de accidentes o calcular la velocidad de los autos.

objetos con cavidades profundas u objetos con múltiples partes.

Varios métodos se han propuesto para mejorar y automatizar el proceso de segmentación con snakes. A continuación se presentan algunos de los trabajos más importantes en orden cronológico.

Cohen (1991) utiliza una fuerza externa de inflación para expandir una snake a través de bordes espúreos hacia los bordes reales de la estructura del objeto, haciéndola menos sensible a las condiciones iniciales. Este modelo, llamado *balloons*, reduce la sensibilidad a la inicialización; sin embargo, la determinación de la magnitud de la fuerza de inflación debe realizarse de manera experimental.

Williams y Shah (1992) introducen un algoritmo voraz para la minimización de energía, obteniendo un proceso de optimización con mayor velocidad, flexibilidad y simplicidad. Sin embargo, el algoritmo sigue sufriendo de algunas deficiencias topológicas.

McInerney y Terzopoulos (1995) desarrollaron un modelo topológico independiente de la forma que permite a la snake no sólo fluir en formas complejas y con muchas ramificaciones o bifurcaciones, sino además sensar dinámicamente y cambiar su topología conforme se necesita por los datos. Su modelo se conoce como *snakes topológicos adaptables.* También proponen un método de segmentación automático utilizando una rejilla (en inglés *grid*) con múltiples snakes semilla y además el modelo se puede extender a una superficie tridimensional.

Xu y Prince (1997) desarrollaron una nueva energía externa llamada *Flujo de Vectores de Gradiente* (GVF por sus siglas en inglés Gradient Vector Flow). El nombre de snake GVF se debe al campo de fuerza que se genera como una difusión de vectores de gradiente de un mapa de bordes binario o de nivel de gris derivado de la imagen. El campo resultante permite a la snake ser inicializada relativamente lejos o incluso a través del objeto de interés, y obliga a la snake a entrar en regiones cóncavas. Sin embargo, el costo computacional es alto por el cálculo del campo GVF.

MacEachern y Manku (1998) y posteriormente Ballerini (1999) utilizaron algoritmos genéticos para la minimización de la energía de snakes. Mostrando algunas características como la habilidad de manejar grandes espacios de búsqueda, escapar de mínimos locales, y el manejo eficaz de restricciones. Sin embargo, su principal desventaja es que la exactitud en la segmentación depende de los parámetros del algoritmo genético, como el tamaño de la población, número de generaciones, etc. Es decir, entre mayor exactitud se requiera mayor será el costo computacional.

I.4 Alcances de esta tesis

Ahora que se conoce la naturaleza del problema, en esta sección se procede a definir los alcances y limitaciones del presente trabajo en forma de objetivos generales y específicos.

I.4.1 Objetivo general

Implementar un sistema de visión que sea capaz de segmentar múltiples objetos en una escena utilizando la técnica de contornos activos.

I.4.2 Objetivos específicos

- Conocer y analizar los diferentes enfoques de contornos activos.
- Evaluar los diferentes métodos de inicialización para contornos activos.
- Encontrar una manera eficiente y automática de inicialización.
- Evaluar los diferentes tipos de energía externa de las snakes para encontrar los contornos de los objetos.
- Evaluar las diferentes técnicas de minimización de energía
- Desarrollar un programa computacional que implemente los paradigmas de snakes más importantes.

I.5 Organización de esta tesis

El trabajo está organizado en los siguientes capítulos:

- *Introducción*: se definen algunos conceptos básicos y se establece el problema que se desea resolver posicionándolo dentro del campo de la Visión por Computadora.
- *Segmentación de Imágenes*: se introducen algunos métodos de segmentación que se utilizan en el estado del arte.
- *Contornos Activos*: se analiza el modelo matemático de las snakes y los principales trabajos involucrados.

- *Puntos de Interés*: se introducen los principales métodos de detección de puntos de interés.

- *Introducción a la Computación Evolutiva*: se introducen los paradigmas de búsqueda y optimización, basados e inspirados en la evolución natural.

- *Inicialización de Contornos Activos utilizando Puntos de Interés*: se plantea una nueva forma para inicializar snakes.

- *Algoritmos Genéticos para la minimización de Snakes*: se analiza un algoritmo genético para la optimización de la energía de los contornos activos.

- *Apéndice A. Herramientas Computacionales*: se muestran los diferentes paquetes de software empleados en el desarrollo de este trabajo.

- *Apéndice B. Cámara Logitech*: se muestran las características técnicas, funcionamiento, e instalación del sistema de adquisión de imágenes que se emplea en este trabajo.

Capítulo II

Segmentación de Imágenes

El objetivo de la segmentación es dividir una imagen en sus regiones u objetos constituyentes (Gonzalez y Woods, 2002). El nivel de división depende de la aplicación; es decir, la segmentación debe detenerse cuando el objeto de interés en la imagen se ha aislado completamente.

La exactitud en la segmentación puede determinar el éxito o fracaso de un proceso de análisis computarizado. Por lo que en ocasiones se busca tomar ventaja de alguna medida de control sobre el ambiente. Cuando esto no es posible, se intenta seleccionar los tipos de sensores que resalten los objetos de interés y que al mismo tiempo elimine la mayor parte posible de detalle irrelevante de la imagen.

Los algoritmos de segmentación generalmente se basan en dos propiedades básicas de los valores de intensidad de la imagen: la discontinuidad y la similitud. La primera aproximación particiona una imagen basada en cambios abruptos en la intensidad de la imagen. La segunda particiona la imagen en regiones similares de acuerdo a un conjunto de criterios predefinidos. En este capítulo se discuten algunas aproximaciones en las dos categorías mencionadas.

II.1 Detección de discontinuidades

La manera más común para buscar discontinuidades es correr una máscara a través de la imagen. Una máscara, también conocida como *filtro*, *kernel*, *ventana* o *plantilla*, es una subimagen cuyos valores trabajan sobre un vecindario en los pixeles de la imagen de las mismas dimensiones que la subimagen. Los valores de la máscara se conocen

como coeficientes.

El concepto de filtro tiene sus raíces en el uso de la transformada de Fourier para procesamiento de señales en el llamado dominio de frecuencias. Por lo que las operaciones de filtrado que se realizan directamente sobre los pixeles de la imagen se conoce como *filtrado espacial*, para diferenciarlo del filtrado más tradicional en el dominio de frecuencias.

II.1.1 Filtrado espacial

El proceso de filtrado espacial consiste simplemente en mover la máscara punto por punto en una imagen, y en cada punto (x, y) se calcula la respuesta al filtro. Para un filtrado espacial lineal la respuesta está dada por la suma de los productos de los coeficientes de la máscara y los pixeles de la imagen correspondientes en el área cubierta por la máscara.

Para una máscara de tamaño $m \times n$ por lo general se asume que $m = 2a + 1$ y $n = 2b + 1$, donde a y b son enteros positivos, ya que máscaras con dimensión par son poco utilizadas debido a que es difícil establecer su centro. En general, el filtrado espacial de una imagen f de tamaño $M \times N$ con una máscara de tamaño $m \times n$ está dado por la expresión:

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t) \tag{1}$$

El proceso de filtrado lineal anterior es similar a lo que en el dominio de frecuencia se conoce como *convolución*. Por lo cual al realizar un filtrado espacial lineal frecuentemente se dice que se convoluciona una máscara con una imagen.

Cuando sólo es importante la respuesta del filtro y no la manera de implementarla, es común simplificar la notación. Por ejemplo, para una máscara general de tamaño 3×3 que se observa en la Figura 6, la respuesta en cualquier punto (x, y) en la imagen

está dado por

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

donde z_i es el nivel de gris del pixel de la imagen asociado con el coeficiente de la máscara w_i. En la Figura 7 se muestra una versión ampliada de la región de la Figura 3, utilizando una notación con varibles z.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figura 6: Representación de una máscara general de tamaño 3×3 para filtrado espacial.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Figura 7: Una región de tamaño 3×3 de una imagen, representada en términos de variables z que indican el nivel de gris.

Una consideración importante al implementar un filtro espacial es qué pasa cuando el centro del filtro se aproxima al borde de la imagen. Por ejemplo, si se tiene una máscara cuadrada de tamaño $n \times n$, un borde de la máscara coincidirá con el borde de la imagen cuando el centro de la máscara se encuentre a una distancia de $(n-1)/2$

pixeles alejado del borde de la imagen. Si el centro de la máscara se mueve más cerca del borde, una o más columnas o renglones de la máscara se localizará fuera de la imagen.

Existen varias maneras de manejar esta situación. La más simple consiste en limitar las excursiones del centro de la máscara a una distancia no menos de $(n-1)/2$ pixeles del borde. Sin embargo, la imagen resultante es más pequeña que la original, pero todos los pixeles en la imagen filtrada se procesan con la máscara completa.

Si es necesario que la imagen resultante tenga las mismas dimensiones que la original, entonces se puede filtrar la imagen sólo con la sección de la máscara que se encuentre dentro de la imagen. Con esta aproximación se obtienen bandas de pixeles cerca de los bordes de la imagen que se procesan parcialmente con la máscara.

Otras aproximaciones involucran agrandar la imagen al agregar columnas o renglones de ceros u otro nivel de gris, o también se puede agregar réplicas de columnas o renglones. Esto asegura que el tamaño de la imagen filtrada será el mismo que la original, pero los valores del agrandamiento tienen un efecto que es más notorio cuando el tamaño de la máscara se incrementa.

II.2 Detección de bordes

En esta sección se discuten las aproximaciones para implementar la primera y segunda derivada digital para la detección de bordes en una imagen.

La detección de bordes (en inglés *edges*) es la aproximación más utilizada para detectar discontinuidades significativas en nivel de gris. Un borde es un conjunto de pixeles conectados que se encuentran en los límites entre dos regiones. Por lo tanto un borde puede indicar el límite entre un objeto y el fondo de la imagen o también el límite del traslape entre objetos.

El concepto de borde con frecuencia se utiliza en trabajos que tratan con regiones o

fronteras (en inglés *boundaries*). Sin embargo, exite una diferencia entre estos conceptos. Una frontera o contorno, como también se le conoce, es una región finita que forma un camino cerrado y por lo tanto es un concepto global. En cambio, los bordes, como se explica más adelante, se forman con pixeles donde los valores de derivada exceden un umbral determinado. Por lo tanto un borde es un concepto local que se basa en la medida de discontinuidades de niveles de gris en un punto. Es posible unir los puntos de un borde para formar segmentos de borde, y algunas veces estos segmentos se unen de tal manera que corresponden a contornos, pero ésto no es siempre el caso.

Una definición más formal de un borde requiere la habilidad de medir las transiciones de niveles de gris de una manera significativa. Intuitivamente un borde ideal tiene las propiedades del modelo que se muestra en la Figura 8(a). Un borde ideal de acuerdo con este modelo es un conjunto de pixeles conectados (en este caso en dirección vertical), cada uno de los cuales se localiza en un paso ortogonal en la transición de niveles de gris (como se muestra por el perfil horizontal de la Figura 8).

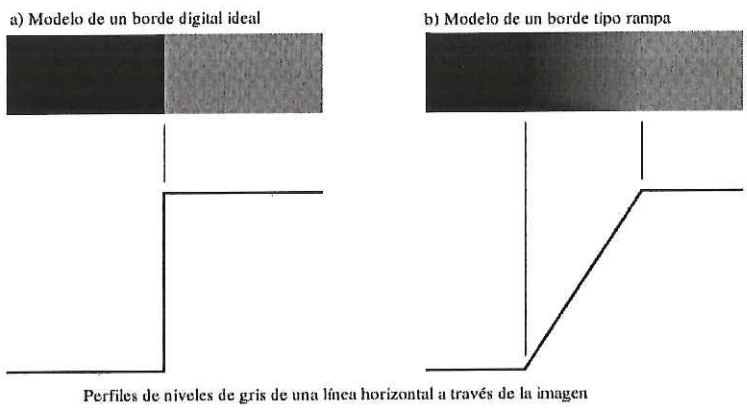


Figura 8: Modelos de borde.

En la práctica la óptica, muestreo, y otras imperfecciones en la adquisición de la

imagen dan por resultado bordes que están suavizados. El grado de suavizado se determina por factores como la calidad del sistema de adquisición, la tasa de muestreo, y las condiciones de iluminación bajo las cuales se adquiere la imagen. De esta manera, los bordes pueden ser modelados con un perfil tipo rampa, como el que se muestra en la Figura 8(b). La pendiente de la rampa es proporcional al grado de suavizado del borde. En este modelo ya no se tiene un camino delgado (de un pixel de grosor), sino que el borde ahora es cualquier punto que contiene la rampa. El grosor de un borde se determina por la longitud de la rampa, que hace una transición de un nivel de gris inicial a uno final.

Debido a que un borde está definido por un cambio en el nivel de gris, un operador que es sensible a este cambio funcionará como un detector de bordes. Por lo que es lógico pensar en el operador diferencial, ya que una derivada puede ser interpretada como la razón de cambio de una función. Un acercamiento de la Figura 8 se muestra en la Figura 9, además se observa un perfil horizontal de niveles de gris del borde entre las dos regiones, y la primera y segunda derivada del perfil. La primera derivada es positiva en los puntos de transición de entrada y salida de la rampa, es constante para los puntos en la rampa y es cero en áreas de nivel de gris constante. La segunda derivada es positiva en la transición que se asocia con el lado oscuro del borde, negativo en la transición que se asocia al lado claro (blanco) del borde, y cero a lo largo de la rampa y en áreas de nivel de gris constante.

Por lo tanto la magnitud de la primera derivada puede utilizarse para detectar la presencia de un borde en un punto de la imagen; es decir, puede utilizarse para determinar si un punto se encuentra en la rampa. De la misma manera, el signo de la segunda derivada puede utilizarse para determinar cuando un pixel se encuentra en el lado oscuro o claro del borde. Otras propiedades adicionales de la segunda derivada alrededor de un borde son:

1. Produce dos valores para cada borde en una imagen.

2. Si se traza una línea recta para unir los valores extremos positivo y negativo de la segunda derivada, la línea cruzará por cero cerca del punto medio del borde. Esta propiedad de cruces por cero es útil para localizar el centro de un borde grueso.

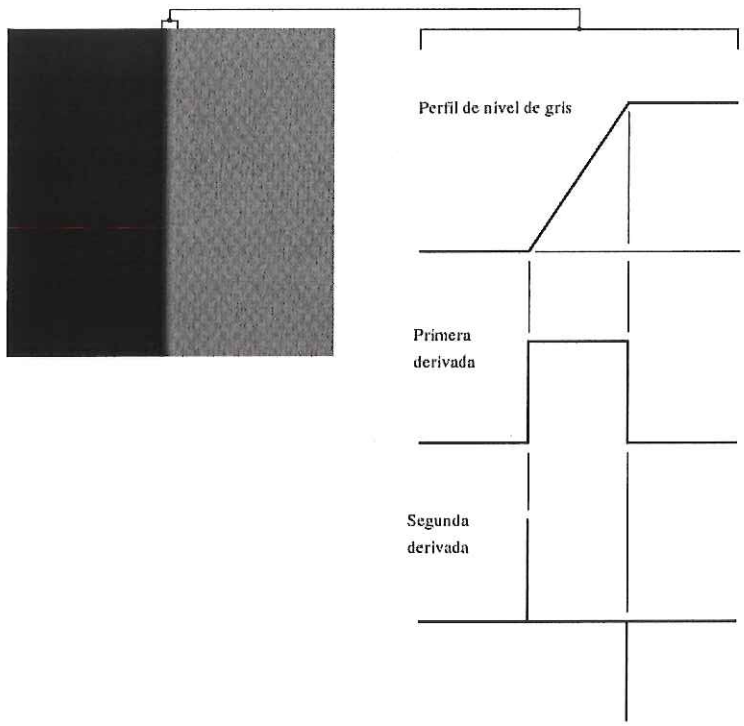


Figura 9: Dos regiones separadas por un borde vertical, el perfil de nivel de gris y la primera y segunda derivada del perfil.

En conclusión, para clasificar un punto como un borde, la transición de los niveles de gris asociados con el punto debe ser suficientemente más fuerte que el fondo en ese punto. Como se trabaja con cálculos locales, el método para determinar si un valor es lo suficientemente fuerte es utilizar un umbral. Entonces, se puede definir que un punto de la imagen es un punto de borde si la derivada de primer orden bidimensional

es mayor que un umbral determinado. Para calcular la derivada de primer orden de una imagen se utiliza el *Gradiente* y para obtener la derivada de segundo orden de una imagen se utiliza el *Laplaciano.*

II.2.1 Gradiente

La derivada de primer orden de una imagen digital se basa en varias aproximaciones de la magnitud de gradiente en dos dimensiones. El gradiente de una imagen $f(x, y)$ en el punto (x, y) se define como un vector

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

Como una imagen es bidimensional, es importante considerar que el cambio de niveles de gris puede ser en muchas direcciones. Por esta razón, las derivadas parciales de una imagen se realizan con respecto a las principales direcciones, en este caso x y y. La magnitud del vector de gradiente es muy importante en la detección de bordes y está dado por

$$\nabla f = mag(\nabla \mathbf{f}) = \sqrt{G_x^2 + G_y^2}$$

La magnitud del gradiente representa la máxima tasa de crecimiento de $f(x, y)$ por unidad de distancia en la dirección de $\nabla \mathbf{f}$. La dirección del vector de gradiente también es importante y está dado por

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

donde θ representa el ángulo de dirección del vector $\nabla \mathbf{f}$ en el pixel (x, y) con respecto al eje x. La dirección de un borde en (x, y) es perpendicular a la dirección del vector gradiente en ese punto.

La implementación para el cálculo de la magnitud de gradiente sobre una imagen completa, no es trivial. Con frecuencia se aproxima utilizando valores absolutos en lugar de una raiz cuadrada, la cual es más sencilla de implementar y reduce la carga computacional.

$$\nabla f \approx |G_y| + |G_x|$$

Por supuesto, una imagen digital no es una función continua, y no puede ser derivada de forma normal. De esta forma y debido a que una imagen es discreta y no continua, para aproximar la derivada de una función digital generalmente se define en términos de diferencia y existen varias formas de definir estas diferencias. Sin embargo, se debe tomar en cuenta que cualquier definición que se utilice debe cumplir con las propiedades de la derivada (ver Figura 9):

1. Debe ser cero a lo largo de segmentos con valores de niveles de gris constante.

2. No debe ser cero a lo largo de la rampa.

Una definición básica de la derivada de primer orden de una función unidimensional $f(x)$ es la diferencia

$$\frac{\partial f}{\partial x} = f(x) - f(x-1)$$

Se puede verificar fácilmente que esta definición satisface las condiciones mencionadas anteriormente. Utilizando la notación de la Figura 7 para denotar los puntos de la imagen en una región de tamaño 3×3, donde el punto central z_5 denota $f(x, y)$, z_1 denota $f(x-1, y-1)$, z_9 denota $f(x+1, y+1)$, etc., podemos obtener el gradiente en las direcciones x y y como

$$G_x = (z_5 - z_4)$$
$$G_y = (z_5 - z_8)$$

Otra manera de aproximar la derivada es utilizando diferencias cruzadas propuesto

por Roberts (1965)

$$G_x = (z_5 - z_9)$$
$$G_y = (z_6 - z_8)$$

Entonces, podemos calcular la magnitud del gradiente como

$$\nabla f \approx |z_5 - z_9| + |z_6 - z_8|$$

Esta ecuación se puede implementar utilizando las máscaras de tamaño 2×2 de la Figura 10. Otra aproximación utilizando máscaras de tamaño 3×3 son los operadores de *Prewitt* que se observan en la Figura 10. Una pequeña variación de esta máscara consiste en utilizar un coeficiente de valor 2 en el centro, con lo que se intenta tener algo de suavizado dándole mayor importancia al punto central. Estas máscaras son conocidas como operadores de *Sobel* y se observan en la Figura 10. En la dicha figura, la primera columna describe las máscaras utilizadas para calcular el gradiente en la dirección x y la segunda columna las máscaras en la dirección y; sin embargo, es posible modificar las máscaras de tamaño 3×3 para que tengan una respuesta más fuerte en las direcciones diagonales.

Se puede observar que los coeficientes de todas las máscaras suman cero, dando una respuesta de cero en áreas con nivel de gris constante, como se espera que suceda con una derivada. Los operadores de Prewitt y Sobel son los que más se utilizan en la práctica para calcular el gradiente de una imagen digital. Sin embargo, la máscara de Sobel tiene una pequeña ventaja en cuanto a las características de eliminación de ruido, un aspecto importante al trabajar con derivadas.

La Figura 11 ilustra la respuesta de los dos componentes del gradiente G_x y G_y utilizando las máscaras de Sobel, así como la magnitud del gradiente de la imagen resultado de la suma de estos componentes. La imagen original (Figura 11(a)) tiene una resolución de 320×240 adquirida con la cámara web Logitech. La dirección de los 2 componentes es evidente en la Figura 11(b) y (c). Se puede observar en particular

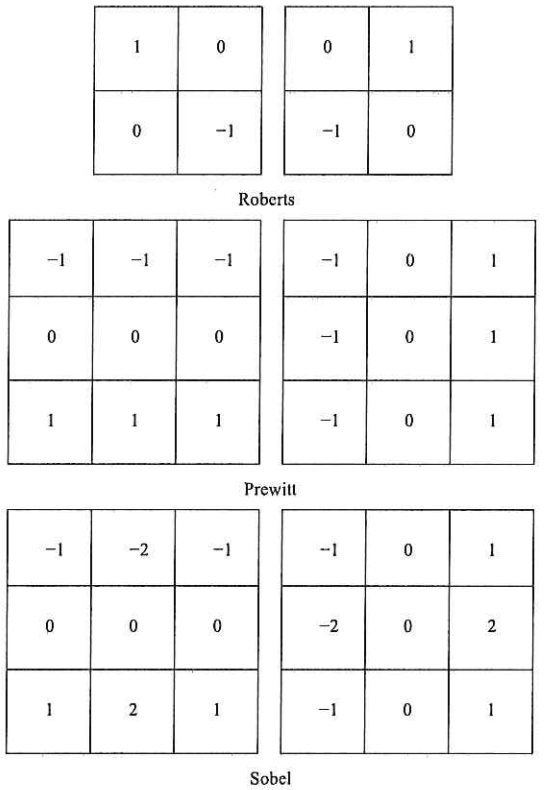


Figura 10: Varias máscaras para calcular el gradiente.

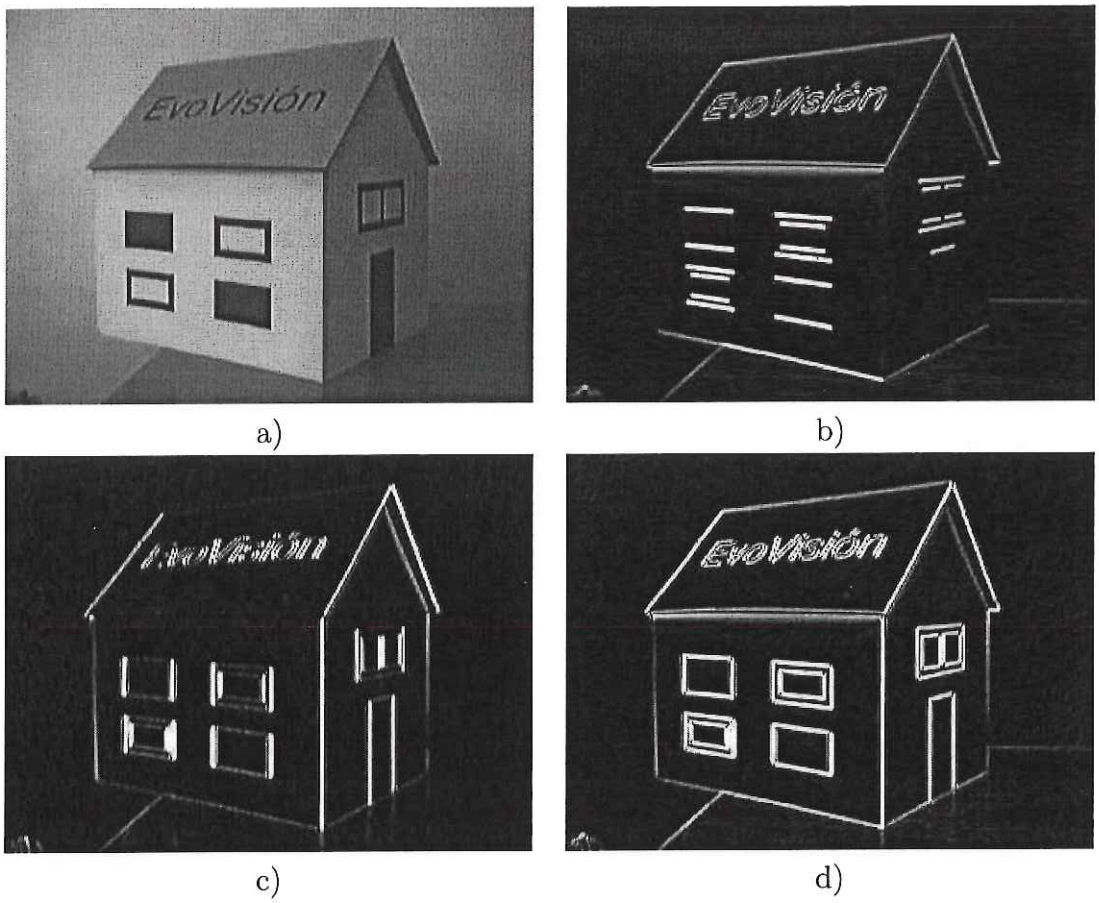
que tan fuertes son los segmentos horizontales en las ventanas de la Figura 11(b). En contraste, la Figura 11(c) realza los componentes verticales, principalmente en la puerta. Por último la Figura 11(d) ilustra la suma de los dos componentes del gradiente.

II.2.2 Laplaciano

Así como la derivada de primer orden se aproxima utilizando el operador de gradiente, la derivada de segundo orden es aproximada utilizando el Laplaciano. El Laplaciano de una función bidimensional $f(x, y)$ está definido por

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

Al igual que el gradiente, el Laplaciano se puede aproximar utilizando máscaras. Los operadores de gradiente requieren la evaluación de dos máscaras de convolución en



a) b)

c) d)

Figura 11: a)Imagen original. b)$|G_x|$, componente del gradiente en la dirección x. c)$|G_y|$, componente del gradiente en la dirección y. d)Gradiente de la Imagen, $|G_x| + |G_y|$.

direcciones perpendiculares, por el contrario, el laplaciano requiere una única máscara. Las dos formas más comunes para una región de tamaño 3×3 se pueden observar en la Figura 12.

El requisito básico para la definición del Laplaciano digital es que el coeficiente central de la máscara sea positivo y los coeficientes exteriores sean negativos, o viceversa. Como el Laplaciano es una derivada, la suma de los coeficientes también debe ser cero.

Generalmente el Laplaciano no se utiliza en su forma original para la detección de bordes por las siguientes razones: por ser una derivada de segundo orden es muy

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Figura 12: Máscaras para aproximar el Laplaciano.

sensible al ruido, la magnitud del Laplaciano produce bordes dobles lo cual no se desea ya que complica la segmentación, y es incapaz de detectar la dirección del borde. Por lo tanto, el papel del Laplaciano en la segmentación consiste en utilizar la propiedad de cruces por cero para la localización de bordes.

II.3 Conexión de bordes y detección de contornos

La detección de bordes no es mas que el primer paso del proceso de segmentación. Idealmente los métodos discutidos en la sección anterior deberían encontrar pixeles pertenecientes sólo a los bordes. Sin embargo, en la práctica este conjunto de pixeles por lo general no caracteriza un borde de forma completa debido a factores como el ruido, huecos en el borde por una iluminación no uniforme, y otros efectos que introducen discontinuidades espurias de la intensidad. Por lo cual el siguiente paso a los algoritmos de detección de bordes consiste en utilizar procedimientos de conexión para enlazar los puntos detectados y formar caminos, y si es posible, curvas cerradas o contornos.

La aproximación más simple para la conexión de bordes es analizar las características de los pixeles en un vecindario pequeño (3×3 o 5×5) a lo largo de todos los puntos (x, y) en una imagen marcados como bordes por una de las técnicas discutidas en la sección anterior. Todos los puntos que son similares de acuerdo a un criterio predefinido se conectan. Las dos propiedades principales que se utilizan para establecer la similitud

de pixeles de borde son:

1. La magnitud del gradiente.

2. La dirección del vector de gradiente.

Por lo tanto un pixel de un borde con coordenadas (x_0, y_0) en un vecindario es similar en magnitud a un pixel (x, y) si

$$|\nabla f(x, y) - \nabla f(x_0, y_0)| \leq E$$

donde E es un umbral mayor que cero. Y de la misma manera la dirección (ángulo) es similar si

$$|\theta(x, y) - \theta(x_0, y_0)| < A$$

donde A es un umbral del ángulo mayor que cero. De esta manera el punto (x, y) es conectado al pixel (x_0, y_0) si ambos criterios de magnitud y dirección se satisfacen.

II.4 Segmentación basada en regiones

El objetivo de la segmentación es particionar una imagen en regiones. En las secciones anteriores se utilizaron técnicas para encontrar las fronteras entre regiones con base a discontinuidades en los niveles de gris. En esta sección se discute una técnica de segmentación que se basa en encontrar las regiones directamente.

Si R representa la región de la imagen completa, la segmentación se puede definir como un proceso que particiona R en n subregiones, $R_1, \ldots, R_n$, de tal manera que

1. $\bigcup_{i=1}^{n} R_i = R$.

2. R_i es una región conectada, $i = 1, \ldots, n$.

3. $R_i \bigcap R_j = \emptyset$ para toda i y j, con $i \neq j$.

4. $P(R_i) = VERDADERO$ para $i = 1, \ldots, n$.

5. $P(R_i \bigcup R_j) = FALSO$ para $i \neq j$.

donde $P(R_i)$ es un predicado lógico que indica el criterio de similitud que se define sobre los puntos en el conjunto R_i y $\emptyset$ es el conjunto vacio.

La condición (1) indica que la segmentación debe ser completa; es decir, todos los pixeles deben pertenecer a una región. La condición (2) requiere que los puntos en una región deben estar conectados de algun modo predefinido. La condición (3) indica que las regiones deben estar separadas. La condición (4) trata de las propiedades que se deben satisfacer por los pixeles en una región segmentada, por ejemplo, el predicado $P(R_i) = VERDADERO$ puede indicar que todos los pixeles en R_i tienen el mismo nivel de gris. Finalmente, la condición (5) indica que las regiones R_i y R_j son diferentes en el sentido del predicado P.

II.4.1 Crecimiento de regiones

Como su nombre lo indica el crecimiento de regiones es un procedimiento que agrupa pixeles o subregiones para formar regiones más grandes empleando un criterio de uniformidad. Se comienza con un grupo reducido de puntos y se van agregando aquellos pixeles adyacentes que tienen propiedades similares (nivel de gris, color, textura) y cuya diferencia no supere cierta tolerancia.

La selección de un conjunto de puntos iniciales es dependiente de la naturaleza del problema. Cuando no hay disponible información *a priori*, el procedimiento consiste en calcular para todos los pixeles el mismo conjunto de propiedades que se utilizan para asignar los pixeles a regiones durante el procedimiento de crecimiento. Si el resultado de estos cálculos forma grupos de valores, los pixeles cuyas propiedades los ubican cerca del centroide de estos grupos pueden utilizarse como semilla.

La selección del criterio de similitud no depende sólo del problema en consideración, sino además del tipo de datos disponibles en la imagen. Otro problema en la técnica de crecimiento de regiones es la formulación de una regla de paro o terminación. Básicamente, el crecimiento de región debe detenerse cuando no hay más pixeles que satisfacen el criterio de similitud. Los criterios como el nivel de gris, color y textura, son locales por naturaleza y no toman en cuenta el historial del crecimiento de región. Por lo que con frecuencia se utilizan criterios como el tamaño, la similitud entre un pixel candidato y los pixeles pertenecientes a la región hasta el momento, y la forma de la región en crecimiento.

II.5 Conclusiones

La segmentación de imágenes es un paso preliminar esencial en la mayoría de las tareas de alto nivel en Visión por Computadora. Como se indica por los ejemplos que se discuten en secciones anteriores, la elección de una técnica de segmentación es dependiente de las características del problema que se desea resolver. Los métodos discutidos en este capítulo sólo son representativos de las técnicas que se utilizan en la práctica; sin embargo, exiten otras técnicas como la segmentación por pirámides, segmentación por watersheds morfológicos, segmentación por umbralización, entre otras.

El siguiente paso a los algoritmos de detección de bordes consiste en utilizar métodos de conexión. En este capítulo se analizó un método común para conectar bordes utilizando información local. Otro método para realizar ésto consiste en utilizar la Transformada Hough. En esta técnica los puntos se conectan determinando primero si caen sobre una curva con una forma específica. Y en donde a diferencia del método de análisis anterior ahora se considera una relación global entre los pixeles.

Otra aproximación global para la conexión de bordes se basa en representar segmentos de bordes en forma de grafos y buscar caminos de bajo costo que correspondan

a los contornos. Esta representación provee una aproximación que tiene un buen desempeño en la presencia de ruido; sin embargo, el procedimiento es considerablemente más complicado y requiere mayor tiempo de procesamiento que la transformada Hough.

En el siguiente capítulo se describe una técnica de segmentación de imágenes que se conoce como modelos de contornos activos. Estos modelos superan algunas deficiencias de algunas técnicas de segmentación descritas en este capítulo, y no necesitan de un posprocesamiento de conexión de segmentos, ya que sus resultados son visualmente interpretables.

Capítulo III

Contornos Activos

Un Contorno Activo se puede definir como una *spline* minimizadora de energía que se permite deformar por una energía externa restrictiva y la influencia de una energía de la imagen que empujan al contorno hacia características de un objeto, como pueden ser los bordes.

A diferencia de otras técnicas de segmentación este modelo de contorno es activo. Siempre se encuentra minimizando su funcional de energía y por lo tanto exhibe una conducta dinámica. Por la manera en que el contorno se desliza mientras minimiza su energía, se le llama *snake*, ya que su movimiento se asemeja al de una serpiente.

El modelo de snake se basa en principios de Geometría y Física, particularmente la dinámica de curvas elásticas. Por lo tanto, este capítulo comienza con la descripción de curvas para representar objetos y después se analizan las curvas paramétricas llamadas splines lo cual ayudará a comprender mejor la formulación de una snake. Por último se describe el modelo matemático de una snake, su funcionamiento y algunos de los trabajos más importantes.

III.1 Representación de curvas

La necesidad de representar curvas y superficies proviene de modelar y/o representar objetos reales. En ambos casos, normalmente no existe un modelo matemático previo del objeto. Una solución a este problema es realizar una aproximación del objeto por pedazos de planos, esferas u otras formas simples de modelar (ver Figura 13), de tal manera que lo que se intenta es que los puntos del modelo se encuentren lo más cercanos

posible a los puntos correspondientes del objeto real.

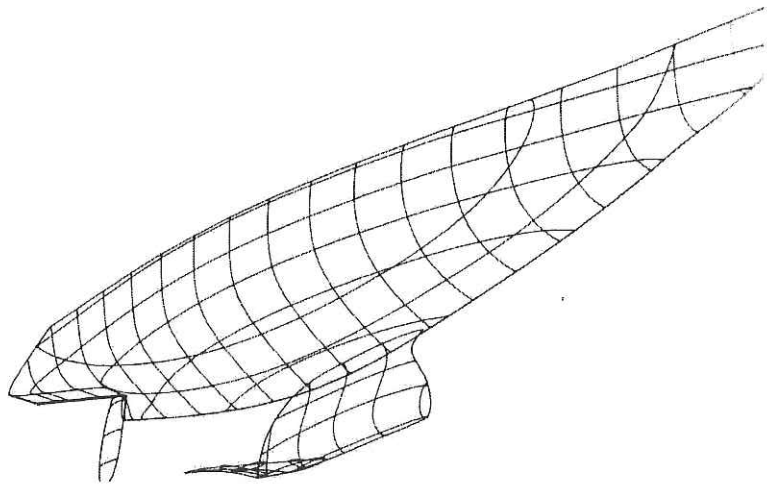


Figura 13: Representación de un objeto tridimensional por pedazos de planos.

Existen tres formas de representar objetos:

1. Explícitamente: $y = f(x)$.
2. Implícitamente: $f(x, y) = 0$.
3. Paramétricamente: $x = x(t)$ y $y = y(t)$.

III.1.1 Representación explícita

En dos dimensiones una curva se puede expresar como una función explícita:

$$y = f(x)$$

Ejemplos:

- Una línea: $y = ax + b$.
- La mitad de un círculo: $y = \sqrt{r^2 - x^2}$.

Las dificultades con este tipo de representación son las siguientes:

1. Es imposible obtener múltiples valores de y para una sola x, por lo que curvas como los círculos o elipses, se deben representar por múltiples segmentos de curvas.

2. Describir curvas con tangentes verticales es difícil, porque una pendiente al infinito es difícil de representar.

III.1.2 Representación implícita

En dos dimensiones una curva se puede representar como soluciones de ecuaciones implícitas de la forma:

$$f(x, y) = 0$$

donde la función f es evaluada en el par (x, y). Ejemplos:

- Una línea: $ax + by + c = 0$
- Un círculo: $x^2 + y^2 - r^2 = 0$

Algunas dificultades con esta representación son:

1. La ecuación dada puede tener más de una solución. Por ejemplo, al modelar un círculo se podría utilizar la ecuación de ejemplo anterior; sin embargo, para modelar la mitad de un círculo se deben agregar restricciones como $x \geq 0$, que no puede ser contenida en la ecuación implícita.

2. Si se requiere unir dos segmentos de curva definidos implícitamente, puede ser difícil determinar si sus direcciones de tangente coinciden en su punto de intersección. La continuidad tangencial es crítica en muchas aplicaciones.

III.1.3 Representación paramétrica

La representación paramétrica para curvas supera los problemas causados por funcionales o formas implícitas. El valor de cada variable espacial se expresa en términos de una variable independiente t, llamada parámetro. En dos dimensiones, una curva paramétrica se describe como:

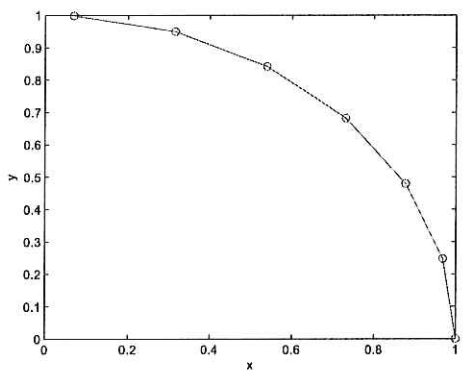
$$p(t) = [x(t), y(t)]^T$$

donde $t_1 \leq t \leq t_2$, que con frecuencia se normaliza de modo que $t_1 = 0$ y $t_2 = 1$.

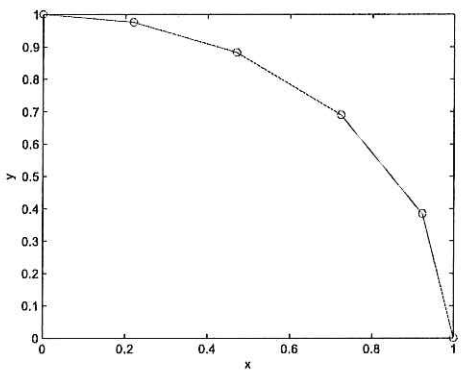
La derivada representa la tangente a la curva y es perpendicular a la normal del punto, y se puede representar de la siguiente forma

$$\frac{dp(t)}{dt} = [x(t)', y(t)']^T$$

Es importante mencionar que muchas figuras geométricas pueden tener más de una representación paramétrica. En la Figura 14 se muestra un ejemplo de dos representaciones paramétricas para formar un cuarto de un círculo.



a) $x = cos(t)$, $y = sen(t)$, $0 \leq t \leq \frac{\Pi}{2}$



b) $x = \frac{1-t^2}{1+t^2}$, $y = \frac{2t}{1+t^2}$, $0 \leq t \leq 1$

Figura 14: Dos tipos de representación paramétrica para un círculo.

III.2 Splines

En la terminología del dibujo mecánico, una spline[1] es una banda flexible que se utiliza para producir una curva suave a través de un conjunto de puntos designados. Varios pesos pequeños se distribuyen a lo largo de la banda para mantenerla en posición sobre la mesa de dibujo mientras se traza la curva (ver Figura 15). El término *curva de spline* al principio se refería a una curva que se traza de esta manera. Esta curva se puede describir en forma matemática con funciones polinómicas cuyas primera y segunda derivadas son continuas a través de las distintas secciones de la curva. En matemáticas las funciones polinómicas o polinomios son una clase importante de funciones simples y suaves. Simples significa que están construidas solamente con operaciones de adición y multiplicación. Suaves significa que son diferenciables infinitamente; es decir, tienen derivadas de todos los órdenes finitos.

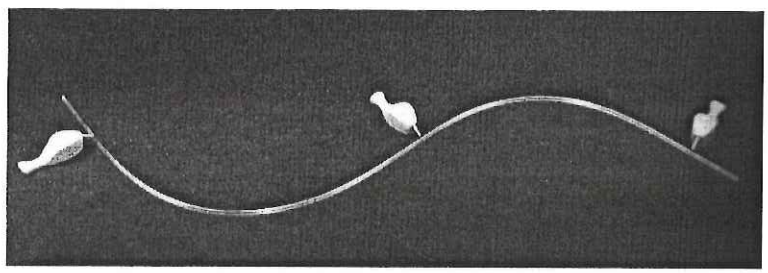


Figura 15: Ejemplo de spline en dibujo mecánico. Una tira de plástico o metal moldeada por tres pesos ubicados en posiciones específicas.

En el área de Gráficas por Computadora (Hearn y Baker, 1995; Foley *et al.*, 1990), el término de curva spline se refiere ahora a cualquier curva compuesta que se forma con

[1]Spline es un término en inglés que significa tira de hule o liga, sin embargo en la literatura en español de Gráficas por Computadora comúnmente se utiliza spline.

secciones polinómicas que satisfacen condiciones específicas de continuidad en la frontera de las piezas. Existen varias clases de especificaciones de spline que se utilizan en aplicaciones gráficas. Cada especificación individual se refiere sólo a un tipo particular de polinomio con ciertas condiciones específicas de frontera.

Las splines se utilizan en las aplicaciones gráficas para diseñar formas curvas y de superficie, para digitalizar trazos para el almacenamiento en la computadora y especificar trayectorias de animación para los objetos o la cámara en una escena. Algunas aplicaciones típicas son el diseño de carrocerías de automóviles, superficies de aeronaves y naves espaciales, cascos de embarcaciones, entre muchas otras.

Considere la curva paramétrica

$$p(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$

esta curva se puede expresar como una curva paramétrica polinomial de grado n de la siguiente forma:

$$p(t) = \sum_{k=0}^{n} t^k B_k$$

Una curva spline se especifica al proporcionar un conjunto de posiciones de coordenadas, que se conocen como *puntos de control*, que establecen la forma general de la curva. Estos puntos de control se ajustan después con funciones polinómicas paramétricas continuas que definen la curva en el sentido de la pieza, y puede ser de dos maneras:

- Por interpolación.
- Por aproximación.

Una curva paramétrica polinomial de grado n requiere $n+1$ puntos de control. Una curva spline se define, modifica y manipula con operaciones en los puntos de control.

Al seleccionar en forma interactiva las posiciones espaciales para los puntos de control, un diseñador puede establecer una curva inicial. Después de que se despliega el ajuste polinomial para un conjunto determinado de puntos de control, el diseñador puede cambiar de posición todos o algunos de los puntos de control para reestructurar la forma de la curva. Además, la curva se puede trasladar, girar o escalar con transformaciones que se aplican a los puntos de control.

III.2.1 Interpolación

Si la spline contiene todos los puntos de control se dice que la curva interpola los puntos; es decir, las secciones polinómicas se ajustan de modo que la curva pasa a través de cada punto de control (ver Figura 16). Este tipo de spline se utiliza por lo general en procesos de digitalización de trazos y especificación de trayectorias de animación.

Figura 16: Conjunto de siete puntos de control que se interpolan con secciones polinómicas continuas en el sentido de la pieza.

III.2.2 Aproximación

Cuando los polinomios se ajustan a la trayectoria general del punto de control sin pasar necesariamente a través de ellos se dice que la curva aproxima los puntos; es decir, la

spline no contiene los puntos de control (ver Figura 17). Este tipo de curva es utilizada en herramientas de diseño para estructurar superficies de objetos.

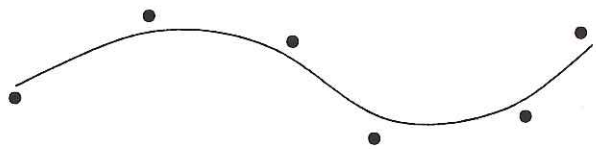


Figura 17: Conjunto de seis puntos de control que se aproximan con secciones polinómicas continuas en el sentido de la pieza.

III.2.3 Condiciones de continuidad paramétrica

Una spline se compone por varias partes de polinomios cúbicos. Para asegurar una transición suave de una sección de una curva paramétrica alrededor de una pieza a la siguiente, se imponen varias condiciones de continuidad en los puntos de conexión entre secciones. Las condiciones de continuidad pueden ser de dos tipos:

- Continuidad paramétrica C^n.
- Continuidad geométrica G^n.

La continuidad paramétrica normalmente es más fuerte que la geométrica, pero existen casos especiales en que G^n no implica C^n.

Continuidad paramétrica

La continuidad paramétrica exige que las derivadas de grado n de las secciones polinomiales coincidan.

- Continuidad de orden cero C^0: sólo implica que las curvas se unen o intersectan. Es decir, el punto final de una sección de la curva es igual al punto inicial de la siguiente sección de la curva.

- Continuidad de primer orden C^1: significa que las primeras derivadas paramétricas (líneas de tangente) para dos secciones de curvas sucesivas son iguales en su punto de unión.

- Continuidad de segundo orden C^2: implica que tanto la primera como la segunda derivada paramétrica (curvatura) de las dos secciones de curvas son las mismas en la intersección.

Las condiciones de continuidad paramétrica de orden superior se definen de manera similar. En la Figura 18 se presentan ejemplos de la continuidad de orden cero, continuidad de primer orden y continuidad de segundo orden. Con C^2 los índices de cambio de los vectores de tangente para las secciones que se conectan son equivalentes en su intersección. Así, la línea de tangente realiza una transición suave de una sección de la curva a la siguiente. Pero con la continuidad de primer orden, los índices de cambio de los vectores de tangente para las dos secciones son muy diferentes, de modo que es posible que las formas generales de las dos secciones adyacentes presenten un cambio abrupto. Con frecuencia, la continuidad de primer orden es suficiente para digitalizar trazos y algunas aplicaciones de diseño. Mientras que la continuidad de segundo orden es útil para establecer trayectorias de animación para el movimiento de la cámara.

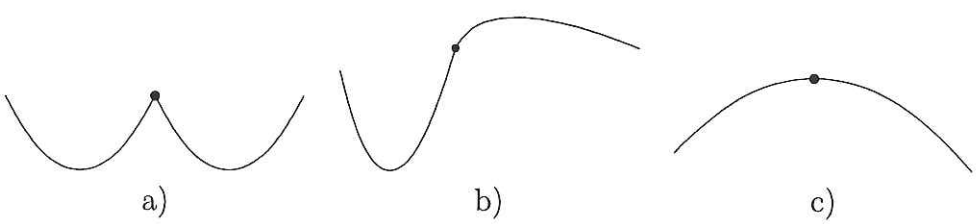


Figura 18: Construcción por piezas de una curva al unir dos segmentos de curva utilizando distintos órdenes de continuidad: a) C^0, b) C^1 y c) C^2.

Continuidad geométrica

Un método alternativo para unir dos secciones de curvas sucesivas consiste en especificar condiciones para la continuidad geométrica, la cual exige que tanto la dirección y sentido de las derivadas de grado n coincida. Aunque en este caso sólo es necesario que las derivadas paramétricas de las dos secciones sean proporcionales entre sí en su frontera común en vez de ser equivalentes. Una curva que se genera con condiciones de continuidad geométrica es similar a una que se genera con continuidad paramétrica, pero con pequeñas diferencias en la forma de la curva.

- Continuidad de orden cero G^0: las curvas se intersectan de la misma manera que C^0.
- Continuidad de primer orden G^1: las tangentes son proporcionales en la intersección de dos secciones sucesivas.
- Continuidad de segundo orden G^2: la primera y segunda derivada de las secciones de curvas son proporcionales en su frontera.

III.2.4 Especificaciones de spline

Existen tres métodos equivalentes para especificar una representación de spline particular:

1. Establecer el conjunto de condiciones de frontera que se imponen en la spline.
2. Establecer la matriz que caracteriza la spline.
3. Establecer el conjunto de *funciones de combinación* (o *funciones base*) que determinan las restricciones geométricas en la curva para calcular posiciones a lo largo de la trayectoria de la curva.

Para ilustrar las tres especificaciones equivalentes anteriores, se utiliza la siguiente representación polinómica cúbica paramétrica:

$$p(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \\ a_z t^3 + b_z t^2 + c_z t + d_z \end{pmatrix}$$

donde $0 \leq t \leq 1$.

Por ejemplo, para la coordenada de x a lo largo de la trayectoria de una sección de la spline, se pueden establecer las condiciones de frontera para esta curva en las coordenadas de extremo $x(0)$ y $x(1)$ y en las primeras derivadas paramétricas en los extremos $x'(0)$ y $x'(1)$. Estas cuatro condiciones de frontera son suficientes para determinar los valores de los cuatro coeficientes a_x, b_x, c_x y d_x.

A partir de las condiciones de frontera, se puede obtener la matriz que caracteriza esta curva de spline como el producto

$$x(t) = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} = T \cdot C$$

donde T es el vector renglón de potencias del parámetro t y C es el vector columna de coeficientes. De esta manera, es posible expresar las condiciones de frontera en forma de matriz y despejar la matriz de coeficientes C como

$$C = M_{spline} \cdot M_{geom}$$

donde M_{geom} es una matriz de columnas de cuatro elementos que contienen los valores de restricción geométrica (condiciones de frontera) en la spline y M_{spline} es la matriz de tamaño 4×4 que transforma los valores de restricción geométrica a los coeficientes polinómicos y ofrece una caracterización para la curva de spline. La matriz M_{geom} contiene los valores de coordenadas del punto de control y otras restricciones geométricas que

se han especificado. Por lo tanto, se puede sustituir la representación de la matriz C y obtener

$$x(t) = T \cdot M_{spline} \cdot M_{geom}$$

La matriz M_{spline} que caracteriza la representación de una spline, también llamada *matriz base*, es de particular utilidad para transformar de una representación de spline a otra.

Por último, se puede ampliar la ecuación anterior con el propósito de obtener una representación polinómica para la coordenada de x en términos de los parámetros de restricción geométrica

$$x(t) = \sum_{k=0}^{3} g_k \cdot BF_k(t)$$

donde g_k son los parámetros de restricción, como las coordenadas de los puntos de control y la pendiente de la curva en los puntos de control, y BF_k son las funciones de combinación polinómica. En las siguientes secciones se analizan algunas splines que se utilizan en forma común, así como su matriz y especificaciones de funciones de combinación.

III.2.5 Spline cúbica natural

Los polinomios cúbicos ofrecen una relación lógica entre la flexibilidad y la velocidad del cálculo. En comparación con los polinomios de orden superior, las splines cúbicas requieren menos cálculos y memoria, a la vez que son más estables. Con respecto a los polinomios de orden inferior, las splines cúbicas son más flexibles para el modelado de formas curvas arbitrarias.

La spline cúbica natural es una curva de interpolación y es una representación matemática de la spline del dibujo mecánico original. Las splines cúbicas naturales tienen una continuidad C^2. Si se tienen $n+1$ puntos de control que se deben ajustar, entonces se tienen n secciones curvas con un total de $4n$ coeficientes polinómicos que

es necesario determinar. En cada uno de los $n-1$ puntos de control interiores, se tienen cuatro condiciones de frontera: las dos secciones curvas en cualquier lado de un punto de control deben tener tanto la primera como la segunda derivadas paramétricas iguales en ese punto de control. Con lo que se obtienen $4n-4$ ecuaciones que se deben satisfacer por los $4n$ coeficientes polinómicos. Se obtiene una ecuación adicional a partir del primer punto de control p_0, la posición de inicio de la curva y otra condición a partir del punto de control p_n, que debe ser el último punto de la curva. Sin embargo, aún se necesitan dos condiciones más para poder determinar los valores para todos los coeficientes. Un método para obtener las dos condiciones adicionales consiste en definir como cero las segundas derivadas p_0 y p_n. Otro planteamiento es el de agregar dos puntos de control simulados, uno en cada extremo de la secuencia original de puntos de control. Es decir, se agregan un punto de control p_{-1} y un punto de control p_{n+1}. De esta manera todos los puntos de control originales son puntos interiores y se tienen las $4n$ condiciones de frontera necesarias.

Una desventaja importante de la spline cúbica natural es que si se altera la posición en cualquier punto de control, se afecta la curva entera. Por lo tanto, las splines cúbicas naturales no permiten un control local, de modo que no es posible reestructurar parte de la curva sin especificar por completo un nuevo conjunto de puntos de control.

III.2.6 Curvas de Bézier

Pierre Bézier, ingeniero francés, desarrolló este método de aproximación de splines para utilizarlo en el diseño de las carrocerías de los automóviles Renault. Las splines de Bézier tienen varias propiedades que hacen que sean muy útiles y convenientes para el diseño de curvas y superficies. Además, es fácil de implementarlas.

En general, es posible ajustar una curva de Bézier para cualquier número de puntos de control. El número de puntos de control que se debe aproximar y su posición relativa

determinan el grado del polinomio de Bézier. Del mismo modo que con las splines de interpolación, se puede especificar una curva de Bézier con condiciones de frontera, con una matriz característica, o con funciones de combinación.

Si se tienen $n+1$ posiciones de puntos de control, es posible combinar estos puntos de coordenadas para producir el siguiente vector de posición $P(t)$, que describe la trayectoria de una función polinómica de Bézier aproximada entre p_0 y p_n

$$P(t) = \sum_{k=0}^{n} p_k BEZ_{k,n}(t)$$

donde $0 \leq t \leq 1$ y p_k corresponde a los puntos de control los cuales se expresan como

$$p_k = (x_k, y_k, z_k)$$

Las funciones de combinación de Bézier $BEZ_{k,n}(t)$ son los polinomios de *Bernstein*:

$$BEZ_{k,n}(t) = C(n,k)t^k(1-t)^{n-k} \tag{2}$$

donde $C(n,k)$ representa los coeficientes del binomio

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Como una regla, una curva de Bézier es un polinomio de grado $n-1$; es decir tres puntos de control generan una parábola, cuatro puntos una curva cúbica y así sucesivamente. La Figura 19 muestra la apariencia de algunas curvas de Bézier para varias selecciones de puntos de control en el plano $xy(z=0)$. Sin embargo, con ciertas posiciones de los puntos de control se pueden obtener polinomios de Bézier degenerados. Por ejemplo, una curva de Bézier que se genera con tres puntos de control colineales es un segmento de línea recta, y un conjunto de puntos de control que se encuentran en la misma posición de coordenadas produce una curva de Bézier que es un solo punto.

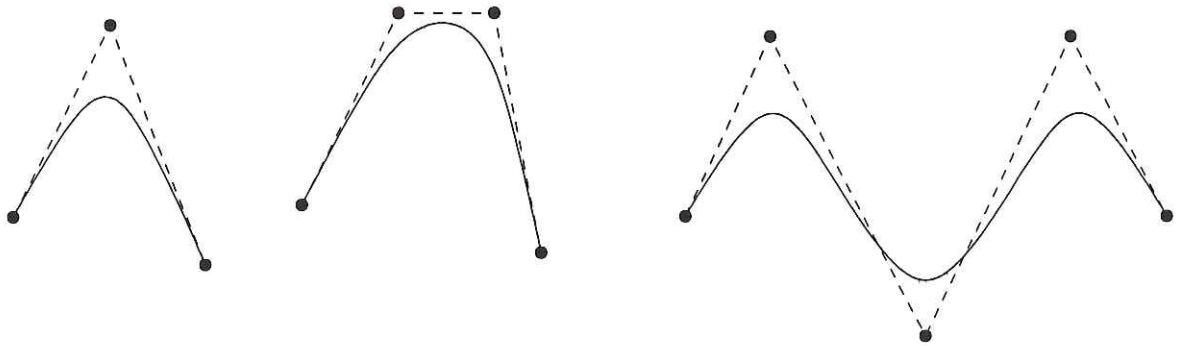


Figura 19: Ejemplos de tres curvas de Bézier bidimensionales que se generaron a partir de tres, cuatro y cinco posiciones de puntos de control.

Propiedades de las curvas de Bézier

Una propiedad muy útil de una curva de Bézier es que siempre pasa a través del primero y el último punto de control. Es decir, las condiciones de frontera en los dos extremos de la curva son

$$P(0) = p_0$$
$$P(1) = p_n$$

Los valores de las primeras derivadas paramétricas de una curva de Bézier en los extremos se pueden calcular a partir de las coordenadas del punto de control como

$$P'(0) = -np_0 + np_1$$
$$P'(1) = -np_{n-1} + np_n$$

Por lo tanto, la pendiente en el principio de la curva es a lo largo de la línea que une los dos últimos extremos. De modo similar, las segundas derivadas paramétricas de una curva de Bézier en los extremos se calculan como

$$P''(0) = n(n-1)[(p_2 - p_1) - (p_1 - p_0)]$$
$$P''(1) = n(n-1)[(p_{n-2} - p_{n-1}) - (p_{n-1} - p_n)]$$

Otra propiedad importante de cualquier curva de Bézier es que cae dentro del casco convexo (ver capítulo 6) de los puntos de control. Ésto se desprende de las propiedades

de las funciones de combinación de Bézier, todas son positivas y su suma siempre es 1,

$$\sum_{k=0}^{n} BEZ_{k,n}(t) = 1$$

de manera que cualquier posición de la curva sólo es la suma ponderada de las posiciones de puntos de control. La propiedad de casco convexo para una curva de Bézier garantiza que el polinomio siga con suavidad los puntos de control sin oscilaciones erráticas.

Curvas cúbicas de Bézier

Las curvas cúbicas de Bézier se generan con cuatro puntos de control. Las cuatro funciones de combinación para curvas cúbicas de Bézier, que se obtienen al sustituir $n = 3$ en la ecuación 2 son

$$BEZ_{0,3}(t) = (1-t)^3$$
$$BEZ_{1,3}(t) = 3t(1-t)^2$$
$$BEZ_{2,3}(t) = 3t^2(1-t)$$
$$BEZ_{3,3}(t) = t^3$$

En la Figura 20 se ilustran los trazos de las cuatro funciones cúbicas de combinación de Bézier. La forma de las funciones de combinación determina la manera en que los puntos de control influyen sobre la forma de la curva para los valores del parámetro t en el rango de 0 a 1. En $t = 0$, la única función de combinación no cero es $BEZ_{0,3}$, que tiene un valor de 1. En $t = 1$, la única función no cero es $BEZ_{3,3}$, con un valor de 1 en ese punto. Por lo que la curva cúbica de Bézier siempre pasa a través de los puntos de control p_0 y p_3. Las otras funciones $BEZ_{1,3}$ y $BEZ_{2,3}$, influyen en la forma de la curva en valores intermedios del parámetro t, de modo que la curva resultante se inclina hacia los puntos p_1 y p_2. La función de combinación $BEZ_{1,3}$ es el máximo en $t = \frac{1}{3}$ y $BEZ_{2,3}$ es el máximo en $t = \frac{2}{3}$.

Se puede observar en la Figura 20 que cada una de las cuatro funciones de combinación es no cero en el rango entero del parámetro t. Por lo que las curvas de Bézier no

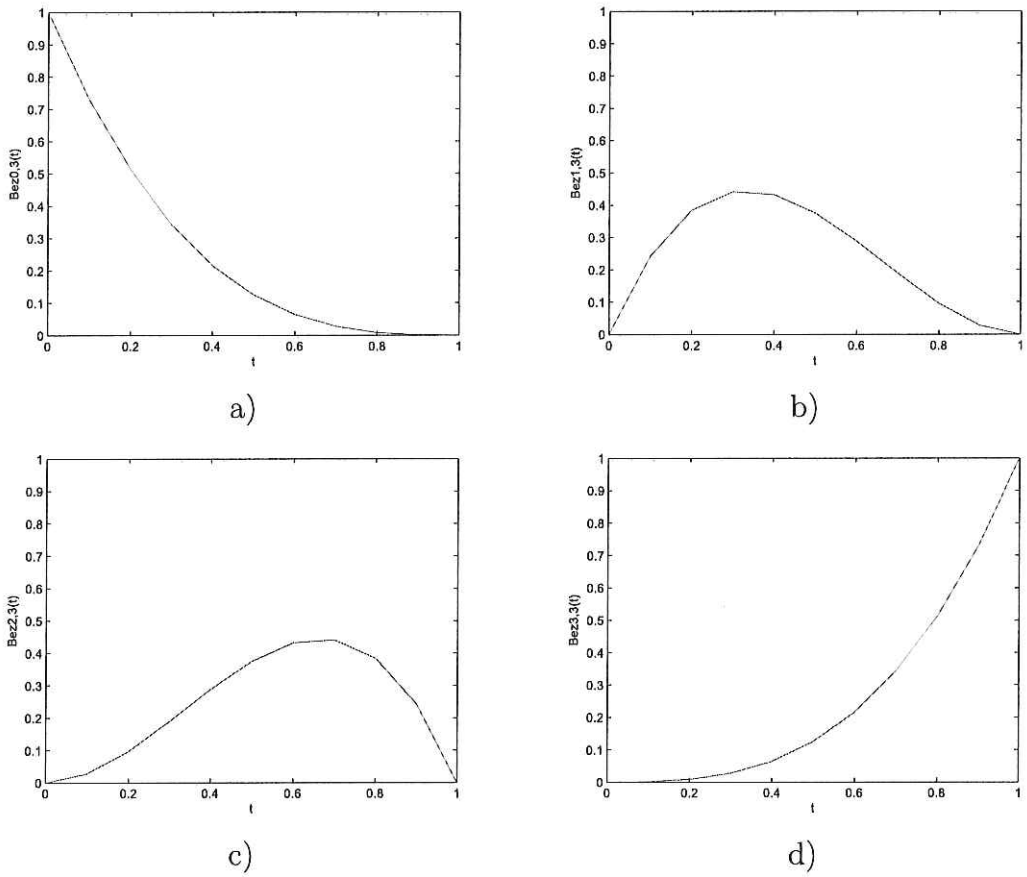


Figura 20: Las cuatro funciones de combinación de Bézier para las curvas cúbicas.

permiten un control local de la forma de la curva. Si se cambia la posición de cualquiera de los puntos de control, se afecta la curva completa.

En las posiciones finales de la curva cúbica de Bézier, las primeras derivadas paramétricas (pendientes) son

$$P'(0) = 3(p_1 - p_0), \quad P'(1) = 3(p_3 - p_2)$$

Y las segundas derivadas paramétricas son

$$P''(0) = 6(p_0 - 2p_1 + p_2), \quad P''(1) = 6(p_1 - 2p_2 + p_3)$$

Al extender las expresiones polinómicas para las funciones de combinación, se puede

representar la función de punto cúbico de Bézier en la forma matricial

$$P(t) = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \cdot M_{BEZ} \cdot \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

donde la matriz de Bézier es

$$M_{BEZ} = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

III.2.7 B-splines

Una spline cúbica natural interpola todos sus puntos de control; sin embargo, cualquier cambio en uno de ellos afecta la curva completa, por lo que son difíciles de manipular. Las B-splines, en cambio, son una clase de spline de aproximación y son las que se utilizan con mayor frecuencia. Las B-splines tienen dos ventajas sobre las splines de Bézier:

1. El grado del polinomio B-spline se puede establecer de manera independiente de la cantidad de puntos de control (con ciertas limitaciones).

2. Las B-splines permiten un control local sobre la forma de una curva de spline.

La desventaja es que las B-splines son más complejas que las splines de Bézier. Se puede escribir una expresión general para el cálculo de las posiciones de las coordenadas a lo largo de una curva B-spline mediante un planteamiento de función de combinación como

$$P(t) = \sum_{k=0}^{n} p_k B_{k,d}(t) \quad t_{min} \leq t \leq t_{max} \quad 2 \leq d \leq n+1$$

donde p_k son un conjunto de entradas de $n+1$ puntos de control. Existen varias diferencias entre esta formulación de B-splines y aquella para las splines de Bézier. El

rango del parámetro t depende de la forma en que se eligen los parámetros de la B-spline. Y las funciones de combinación de B-splines $B_{k,d}$ son polinomios de grado $d-1$, donde el parámetro d se puede elegir como un valor entero en el rango de dos hasta el número de puntos de control $n-1$. En realidad, también se puede establecer el valor de d en 1, pero entonces la curva es sólo un trazo punteado de los puntos de control. El control local para las B-splines se logra al definir las funciones de combinación en subintervalos del rango total de t.

Las funciones de combinación para las curvas B-splines se definen mediante las fórmulas recursivas de *Cox-deBoor*

$$B_{k,1}(t) = \begin{cases} 1, \text{ si } t_k \leq t \leq t_{k+1} \\ 0, \text{ de otro modo} \end{cases}$$

$$B_{k,d}(t) = \frac{t - t_k}{t_{k+d-1} - t_k} B_{k,d-1}(t) + \frac{t_{k+d} - t}{t_{k+d} - t_{k+1}} B_{k+1,d-1}(t)$$

donde cada función de combinación se define en subintervalos d del rango total de t. El conjunto seleccionado de extremos de subintervalo t_j se conoce como *vector de nudo.* Se pueden elegir valores para los extremos de subintervalo al satisfacer la relación $t_j \leq t_{j+1}$. Así los valores para t_{min} y t_{max} dependen de la cantidad de puntos de control que se seleccionen, el valor que se elige para el parámetro d y la forma en que se establecen los subintervalos (vector de nudo). Ya que es posible elegir los elementos del vector de nudo de manera que los denominadores en los cálculos anteriores pueden tener un valor de cero, esta formulación supone que a los términos que se evalúen como una división de cero entre cero se debe asignar el valor cero.

La Figura 21 muestra las características de control local de B-spline. Además del control local, las B-splines permiten variar la cantidad de puntos de control que se utilizan para diseñar una curva sin cambiar el grado del polinomio. También, cualquier cantidad de puntos de control se puede sumar o modificar para manipular las formas de las curvas. De modo similar, se puede aumentar la cantidad de valores en el vector de

nudo para ayudar en el diseño de la curva. Sin embargo, cuando se hace ésto también se necesita sumar los puntos de control ya que el tamaño del vector de nudo depende del parámetro n.

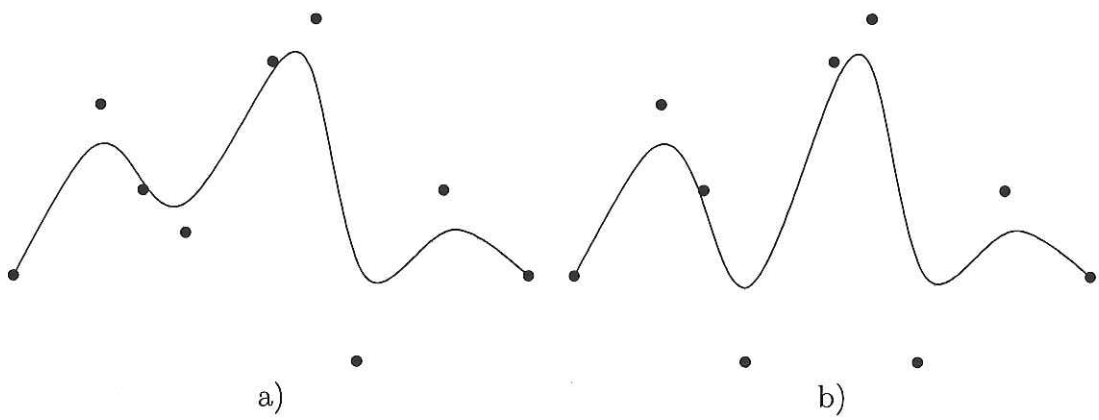


Figura 21: Modificación local de una curva B-spline. Al cambiar uno de los puntos de control en la parte (a) se produce la curva de la parte (b), que se modifica sólo en las áreas cercanas al punto de control alterado.

Las curvas B-spline tienen las propiedades siguientes:

- La curva polinomial tiene un grado $d-1$ y una continuidad C^{d-2} sobre el rango de t.
- Para $n+1$ puntos de control, la curva se describe con $n+1$ funciones de combinación.
- Cada función de combinación $B_{k,d}$ se define sobre los subintervalos d del rango total de t, empezando con el valor de nudo t_k.
- El rango del parámetro t se divide en $n+d$ subintervalos entre los valores $n+d+1$ que se especifican en el vector de nudo.

- Con los valores de nudo que se etiquetan como $\{t_0, ..., t_{n+d}\}$, la curva B-spline que resulta se define sólo en el intervalo desde el valor de nudo t_{d-1} hasta el valor de nudo t_{n+1}.

- Cada sección de la curva spline (entre dos valores de nudo sucesivos) está influenciada por los puntos de control d.

- Cualquier punto de control puede afectar la forma de la mayor parte de las secciones de curva d.

Además, una curva B-spline cae entre el casco convexo en la mayor parte de los puntos de control $d + 1$, de modo que las B-splines se encuentran muy ligadas a las posiciones de entrada. Para cualquier valor de t en el intervalo desde el valor de nudo t_{d-1} hasta t_{n+1}, la suma sobre todas las funciones de base es uno:

$$\sum_{k=0}^{n} B_{k,d}(t) = 1$$

Dadas las posiciones de los puntos de control y el valor del parámetro d, es necesario especificar los valores de nudo para obtener las funciones de combinación al utilizar las relaciones de recursividad. Existen tres clasificaciones generales para los vectores de nudo: uniforme, uniforme abierto y no uniforme. Las B-splines por lo general se describen de acuerdo con la clase de vector de nudo que se selecciona.

III.3 Información *a priori*

Investigadores en psicología de la visión han demostrado que el conocimiento *a priori* (previo) ayuda a la interpretación de una imagen. En ocasiones un observador puede no darle ningún sentido a una imagen hasta que se le da una pista con una sola palabra, entonces el objeto resalta de la imagen. En la Figura 22 se ilustra la idea del conocimiento *a priori*. Si la imagen en la figura nunca ha sido vista con anterioridad, entonces

probablemente signifique poco a primera vista. Sin embargo, con algunas pistas como bicicleta o sombrero, sería posible encontrar a un ciclista con un sombrero mexicano.

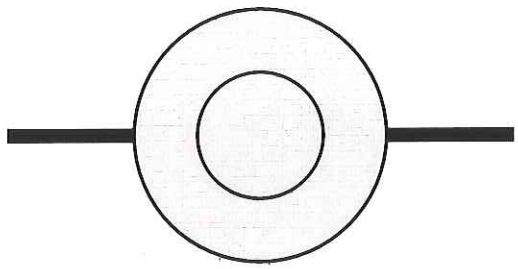


Figura 22: Imagen de ejemplo que ilustra la importancia del conocimiento *a priori*.

La importancia de utilizar conocimiento *a priori* en tareas de visión por computadora radica simplemente en que es muy difícil lograr progresos sin tal conocimiento.

III.4 Modelo matemático de la snake

La posición de la snake se puede representar paramétricamente como $v(s) = (x(s), y(s))$, donde $s \in [0, 1]$. Entonces el funcional de energía de la snake está compuesto por dos componentes:

$$E_{snake} = \int_0^1 [E_{int}(v(s)) + E_{ext}(v(s))]\, ds \tag{3}$$

donde E_{int} representa la energía interna de la snake que sirve para asegurar que la curva permanezca suave y continua, y es asociada a restricciones *a priori*. E_{ext} representa la energía externa que atrae a la snake hacia los objetos, y la cual depende de la imagen y su información *a posteriori*. La energía externa por ejemplo, puede provenir de una interfaz de usuario, mecanismos automáticos, o interpretaciones de alto nivel. La forma final del contorno corresponde al mínimo de su energía.

III.4.1 Energía interna

La energía interna es una característica del contorno por sí mismo y no tiene relación con la imagen. La E_{int} indica las propiedades físicas de la snake. En lugar de esperar que ciertas características del contorno, como la suavidad y continuidad, provengan de la imagen, éstas son impuestas desde el principio de manera *a priori*. La energía interna es definida como:

$$E_{int} = E_{elastica} + E_{curvatura}$$

$$E_{int} = \frac{1}{2}(\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2) \tag{4}$$

donde v_s y v_{ss} son la primera y segunda derivada con respecto a s, respectivamente. Es decir, la energía interna está compuesta por un término de primer orden controlado por un peso $\alpha(s)$ y un término de segundo orden controlado por un peso $\beta(s)$. Ambos parámetros simulan las características físicas del contorno. $\alpha(s)$ controla la tensión mientras que $\beta(s)$ controla la rigidez. Por ejemplo, si se incrementa la magnitud de $\alpha(s)$ aumentará la tensión y se eliminarán los nudos, reduciendo la longitud de la snake. Si se incrementa $\beta(s)$ aumentará la rigidez de curvatura de la snake provocando que la snake sea más suave y menos flexible. Ajustando ambos valores de estas funciones a cero en un punto s, permite discontinuidades en el contorno. Es decir, se permitirá que se presente una esquina. Sin embargo, en la mayoría de las aplicaciones α y β se toman como constantes para todos los puntos a lo largo del contorno.

Para discretizar las pequeñas derivadas de la energía interna, generalmente se hace mediante una aproximación de diferencias finitas, de tal manera que:

$$E_{int} = \frac{1}{2}(\alpha_i|v_i - v_{i-1}|^2 + \beta_i|v_{i-1} - 2v_i + v_{i+1}|^2) \tag{5}$$

Para examinar el comportamiento de los términos de la energía interna, se puede considerar el efecto de cada término individualmente. En el caso de la energía elástica,

como α se toma como una constante, esencialmente se está minimizando la primera derivada que se aproxima (ver ecuación 5) como

$$|v_s(s)|^2 \approx (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2$$

lo cual es equivalente a minimizar la distancia entre los puntos actual y anterior, y que tiene un efecto de encogimiento del contorno.

Similarmente, minimizar la energía de curvatura asegura que la segunda derivada no se incremente mucho. Esta cantidad denota la razón de cambio de la tangente en el punto $v(s)$ (o en forma discreta v_i). Un valor pequeño significa que la tangente de la curva cambia gradualmente; es decir, el contorno se torna curvo muy lentamente.

De esta manera, las energías elástica y de curvatura sirven para mantener el contorno con cierta suavidad. En la Figura 23 se ilustra el efecto de minimizar la energía elástica, la energía de curvatura y la energía interna total; sin la intervención de ninguna energía externa. El contorno externo indica el contorno de inicio, y el contorno interno indica el contorno final después de algunas iteraciones.

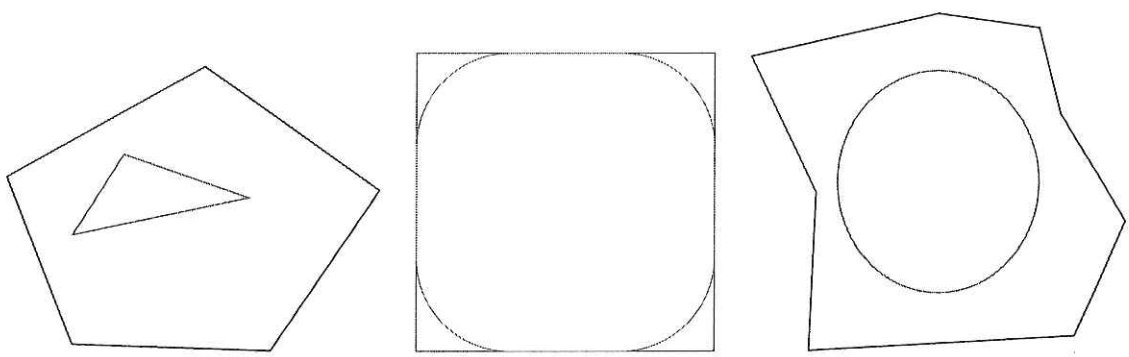


Figura 23: Efecto de minimizar la energía interna: a) La energía elástica $E_{elastica}$. b) La energía de curvatura $E_{curvatura}$. c) Ambas energías $E_{int} = E_{elastica} + E_{curvatura}$.

III.4.2 Energía externa

Se define una función de energía de la imagen de tal manera que toma los valores más pequeños en las características de interés de la imagen como: brillo, bordes, líneas, esquinas, etc.

Existen diferentes tipos de energías externas. La energía de la imagen más simple es la intensidad de los niveles de gris en la imagen por si misma, y se define como sigue:

$$E_{ext} = -\gamma I(x, y) \tag{6}$$

donde $I(x, y)$ es la intensidad de los niveles de gris de la imagen y γ, al igual que α y β en la energía interna, es un peso que sirve para darle la importancia que se requiera a la energía externa. Nótese que el signo negativo de γ hará que la snake sea atraída hacia pixeles de alta intensidad (pixeles claros). De esta manera cambiando el signo a positivo se puede hacer que la snake sea atraída hacia pixeles oscuros. Encontrar los bordes en una imagen también puede ser logrado con un funcional de energía externa simple, que se define a continuación

$$E_{ext} = -\gamma |\nabla I(x, y)|^2 \tag{7}$$

donde ∇ es el operador de gradiente. En este caso la snake será atraída hacia bordes de alta magnitud. Con un pequeño cambio al funcional anterior, otra energía externa muy utilizada es la siguiente:

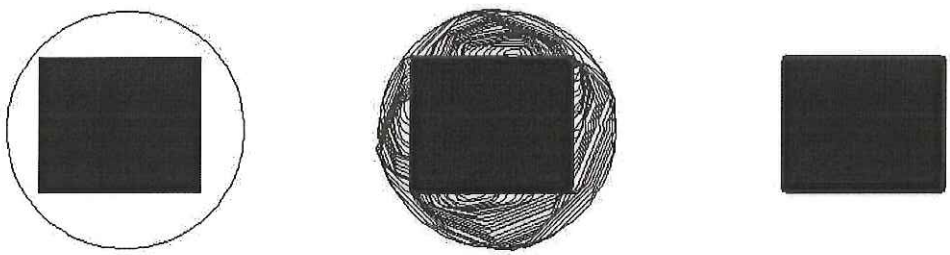
$$E_{ext} = -\gamma |\nabla G_\sigma I(x, y)|^2 \tag{8}$$

donde $G_\sigma I(x, y)$ define la imagen convolucionada por un filtro Guassiano con una desviación estándar σ.

III.5 Funcionamiento de la snake

El funcionamiento general de la snake es muy sencillo. Primeramente un proceso autónomo o un usuario inicializa el contorno en algún lugar cerca del objeto de interés. Después de ésto, la snake iterativamente se comienza a deformar y moverse hacia los bordes del objeto deseado como resultado de la minimización de su energía. Al final se ajusta completamente alrededor del objeto.

En la Figura 24 se ilustra un ejemplo de una snake atraída hacia los bordes de un cuadrado de color negro. En la Figura 24(a) la snake se inicializa como un círculo alrededor del objeto, lo suficientemente cerca como para tener un buen desempeño. Después la snake se comienza a deformar en cada iteración, Figura 24(b). Y por último después de algunas iteraciones la snake se encuentra en los bordes del objeto, Figura 24(c).



a) b) c)

Figura 24: Funcionamiento de una snake: a) Inicialización. b) Deformación. c) Contorno final.

III.6 Trabajo previo

III.6.1 Kass

En 1988 Michael Kass y sus colegas proponen los modelos de snakes a través del siguiente desarrollo. Utilizando las ecuaciones 3 y 4 se puede obtener la siguiente ecuación:

$$E_{total} = \int_0^1 \frac{1}{2}(\alpha|v_s(s)|^2 + \beta|v_{ss}(s)|^2) + E_{ext}(v(s))ds \tag{9}$$

Minimizar esta ecuación pertenece a la clase de problemas llamados *ecuaciones diferenciales de Euler-Lagrange.* Utilizando *cálculo de variaciones* se puede demostrar que la minimización de la energía total debe satisfacer la ecuación de Euler:

$$\alpha v_{ss} - \beta v_{ssss} - \nabla E_{ext} = 0 \tag{10}$$

donde el número de subíndices s indica el orden las derivadas parciales, con respecto al parámetro del contorno s.

Para tener un mejor control sobre la energía externa, se inserta el peso constante γ que se manipula para que la energía externa tenga el mismo orden que la energía interna. Por lo tanto, la ecuación anterior se modifica como sigue:

$$\alpha v_{ss} - \beta v_{ssss} - \gamma\nabla E_{ext} = 0 \tag{11}$$

Esta ecuación se puede separar como las soluciones para las direcciones x y y de la siguiente manera.

$$\alpha x_{ss} - \beta x_{ssss} - \gamma\frac{\partial E_{ext}}{\partial x} = 0 \tag{12}$$

$$\alpha y_{ss} - \beta y_{ssss} - \gamma\frac{\partial E_{ext}}{\partial y} = 0 \tag{13}$$

La ecuación 11, y por consecuencia las ecuaciones 12 y 13, se puede ver como una ecuación de balance de fuerzas como

$$F_{int} + F_{ext} = 0$$

donde

$$F_{int} = \alpha v_{ss} - \beta v_{ssss}$$

y

$$F_{ext} = -\gamma \frac{\partial E_{ext}}{\partial s}$$

Por lo tanto, calculando las ecuaciones 12 y 13 se obtiene la fuerza que actúa sobre un punto particular de la curva y cuya influencia obliga al punto a moverse.

A continuación se describen los pasos del algoritmo para implementar la aproximación de Kass:

1. Para cada punto (x_i, y_i), calcular la energía local de la curva en dicho punto.

2. Estimar las derivadas parciales x_{ss}, x_{ssss}, y_{ss} y y_{ssss} utilizando una aproximación por diferencias finitas.

3. Calcular las fuerzas F_{int} y F_{ext} en las direcciones x y y.

4. Calcular la fuerza total que experimenta el punto (x_i, y_i) en las direcciones x y y, y encontrar el nuevo pixel (x_j, y_j) al que se debe mover.

5. Calcular la energía local de la curva en el nuevo pixel (x_j, y_j).

6. Actualizar el punto (x_i, y_i) por (x_j, y_j) sólo si se obtiene un decremento en la energía local de la curva.

7. Regresar al paso 1 mientras el número de puntos que se movieron sea mayor que un umbral.

III.6.2 Williams y Shah

Williams y Shah (1992) introducen un algoritmo voraz como método de minimización de la energía de la snake, que consiste en examinar un vecindario alrededor de cada

punto que define el contorno de la snake. El algoritmo tiene una complejidad de $O(nm)$, donde n es el número de puntos de la snake y m es el tamaño del vecindario en el cual un punto se puede mover en una sola iteración. Williams y Shah identificaron dos problemas principales que tenía la aproximación de Kass.

Primero, la aproximación de Kass utiliza la primera derivada v_s como el término de la energía elástica; en forma discreta v_s define la distancia entre dos puntos adyacentes del contorno. Por lo tanto, minimizar la energía elástica causa un efecto de encogimiento del contorno, como una liga de hule estirada que se suelta. Como cada punto se mueve en base a consideraciones locales, un punto se empuja hacia al punto anterior por el término de la energía elástica y como resultado se aleja del punto óptimo. Esto causa una racción en cadena de tal modo que los puntos se amontonan cerca de los bordes fuertes de la imagen. Para resolver este problema, en lugar de minimizar la distancia entre los puntos, se minimiza la desviación de la distancia promedio de los puntos

$$d_{avg} - |v_i - v_{i-1}|$$

donde d_{avg} es la distancia promedio entre los puntos. De esta manera los puntos que tienen la distancia cerca del promedio tendrán el valor mínimo.

Otra idea es la normalización de los valores de la energía de la imagen, que puede ser el gradiente. Por ejemplo, en una imagen de 8 bits, la magnitud del gradiente puede tener valores entre 0 y 255. En este caso, existe una diferencia significativa entre un punto con una magnitud de gradiente de 240 y otro con una magnitud de 255. Esto no se refleja cuando los valores son normalizados con respecto al valor máximo 255. Entonces, dada la magnitud de un punto (pix) y el valor máximo (max) y mínimo (min) del gradiente en el vecindario del punto, la fuerza del pixel se normaliza como $(min - pix)/(max - min)$ si $max - min < umbral$ entonces $max = min - umbral$. Si $max - min$ es menor que un umbral, por ejemplo 5, entonces max esta dado por $min - 5$. Esto último previene diferencias grandes en la energía externa donde las

intensidades de los pixeles son más o menos uniformes. Por ejemplo, si los puntos en el vecindario que se examinan tienen una magnitud de gradiente de 47, 48 y 49; entonces los valores normalizados serán 0, -0.5, y -1 respectivamente. Por lo que los puntos de control tendrán una tendencia a quedarse en el punto con intensidad de 49, aunque no exista una gran diferencia entre los pixeles del vecindario. De este modo si se tiene un umbral de 5, se obtendrán los valores normalizados -0.6, -0.8 y -1 respectivamente, los cuales reflejan más la similitud entre los puntos.

El algoritmo ejecuta los siguientes pasos para cada punto que define la snake.

1. Encontrar los pixeles en el vecindario con los valores de gradiente máximo y mínimo.

2. Calcular cada uno de los términos de energía $E_{elastica}$, $E_{curvatura}$ y E_{ext} para cada uno de los pixeles en el vecindario.

3. Normalizar todos los valores de energía.

4. Encontrar el pixel en el vecindario donde la energía total es mínima.

5. Actualizar el punto de control por el nuevo pixel.

6. Continuar con el siguiente punto de control y repetir el proceso.

III.6.3 Xu y Prince

Xu y Prince (1997) proponen una energía externa para snakes, llamada flujo de vectores de gradiente (GVF por sus siglas en ingés Gradient Vector Flow); cuyo nombre se debe al tipo de campo de fuerza que esta energía genera de la imagen para atraer a la snake. El modelo básico de las snakes GVF es identico al propuesto por Kass, donde la única diferencia es la manera en que se calcula la energía externa. La Ecuación (11) define la

ecuación de Euler del contorno como

$$\alpha v_{ss} - \beta v_{ssss} - F_{ext} = 0 \tag{14}$$

Para definir F_{ext}, primero se obtiene el mapa de bordes del gradiente (∇I) de la imagen. Ahora F_{ext} se toma como un vector de la forma $(u(x,y), v(x,y))$ y se evalúa como el campo de vectores que minimiza la siguiente integral

$$E_{ext} = \int\int (u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla I|^2 |F - \nabla I|^2 dxdy \tag{15}$$

En la vecindad de la frontera de un objeto ∇I es muy grande, por lo tanto si se quiere minimizar E_{ext} en esta región se establece $F \approx \nabla I$, que es exactamente lo que se requiere para que la snake tenga un desempeño satisfactorio cuando se inicializa cerca de las fronteras del objeto. Cuando la snake se inicializa muy alejado de los bordes del objeto $\nabla I \rightarrow 0$. Entonces el segundo término de la integral en la ecuación 15 tiende a cero y por lo tanto sólo es necesario calcular el primer término de la ecuación, el cual es una expansión de $\nabla^2 I$. Minimizar este término produce un campo que varía lentamente y que gradualmente converge en la frontera del objeto. La Ecuación (15) se resuelve de la misma manera que en el método de Kass, obteniendo y resolviendo las ecuaciones diferenciales de Euler-Lagrange correspondientes.

Esto asegura que no importa que tan alejado se inicializa el contorno, la snake puede converger rápidamente en la frontera del objeto.

Capítulo IV

Puntos de Interés

Los *puntos de interés* son características de bajo nivel que pueden ser extraídas de una imagen, al igual que otras características comunes como los bordes, *blobs* (regiones con con cierta coherencia), o las esquinas. Estas características de bajo nivel se extraen con el motivo de simplificar el análisis de imágenes, reduciendo la gran cantidad de información contenida en las imágenes que una tarea de visión de alto nivel necesita procesar. Por lo que la detección de puntos de interés es una parte esencial del procesamiento de bajo nivel y de la visión por computadora, ya que se utilizan en sistemas del estado del arte que realizan tareas de visión de alto nivel como: la detección y reconocimiento de objetos, reconstrucción tridimensional, seguimiento, correspondencia, calibración de cámaras, entre otras.

Los puntos de interés se pueden definir como pixeles prominentes de una imagen que exhiben un alto nivel de variación de la señal 2D con respecto a una medida local. Sin embargo, su detección no es trivial, ya que los puntos de interés deben cumplir con ciertas propiedades:

1. **Distintivo**: debe existir una separación global entre los puntos extraídos, de tal manera que los puntos no se deben amontonar en partes aisladas de la imagen. Obviamente este criterio es dependiente de la imagen, y requiere de conocimiento *a priori* del número esperado de puntos y su posición en la imagen.

2. **Inusual**: debe tener alta información de contenido, lo cual indica que los puntos extraídos son únicos cuando se comparan con otros pixeles en un vecindario local.

3. **Invariante**: los puntos detectados deben ser los mismos en el promedio para una imagen bajo diferentes condiciones de ruido.

4. **Estabilidad**: bajo ciertos tipos de transformaciones de la imagen como la rotación, traslación o escalamiento de la imagen, los puntos detectados deben ser los mismos. Probablemente este criterio sea el más importante y es el único en el que existe una métrica ampliamente aceptada: la taza de repetibilidad del detector.

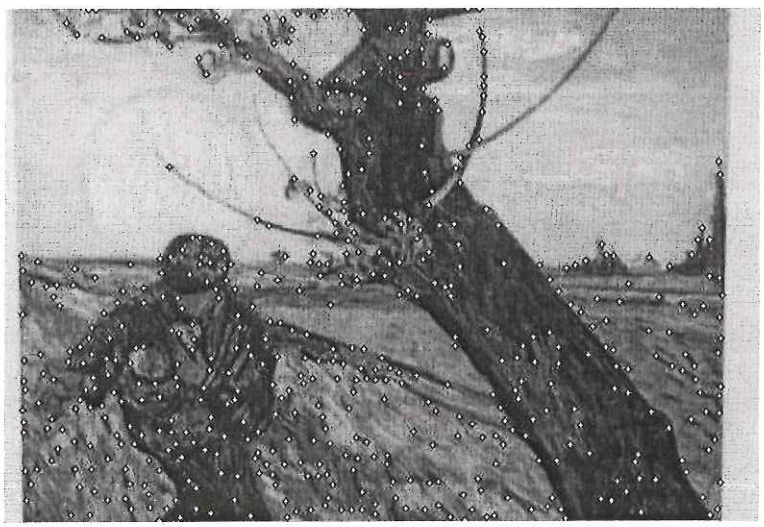


Figura 25: Ejemplo de la detección de puntos de interés. Los círculos blancos representan los 529 puntos de interés encontrados por el detector de Harris.

En la Figura 25 se muestra un ejemplo de la detección de puntos de interés utilizando como imagen un cuadro de Van Gogh. La detección de puntos de interés ha sido resultado de la investigación que concierne a la detección de *esquinas* en una imagen. Las esquinas son características importantes en apareamiento de imágenes, descripción de formas y detección de movimiento, debido a que éstas indican posiciones en la imagen con un alto contenido de información. Aunque no existe una definición exacta

para un punto que representa una esquina en una imagen, podemos reconocer que este punto se encuentra sobre un borde donde la magnitud de gradiente es grande. La información que proporciona la diferenciación de primer orden es ampliamente utilizada en los detectores de esquinas. Los detectores de esquinas pueden ser clasificados en tres clases principales: métodos basados en contornos (puntos de frontera), metodos basados en modelos paramétricos y métodos basados en la intensidad de la imagen (propiedades geométricas).

La clase de detectores de esquinas que operan directamente en la intensidad de la imagen se refiere más apropiadamente como detectores de puntos de interés. Estos operadores definen una función que trabaja en un vecindario local y extrae una esquina o medida de interés para todos los pixeles en la imagen. Esta operación produce una nueva imagen que puede ser llamada imagen de *interés*, a la cual se le aplica una umbralización para extraer los puntos con una medida de interés alta. Aunque conceptualmente este tipo de operadores se diseñaron como detectores de esquinas, sus capacidades de detección no se limitan a puntos que conforman el concepto geométrico de lo que es una *esquina*. Por lo cual, estos operadores extraen todos los puntos donde las variaciones de la intensidad de la imagen son altas con respecto a una medida particular. Es decir, el tipo de características de la imagen que extraen pueden explicarse mejor con un concepto más general de puntos de interés. En este contexto, los puntos de interés son pixeles de la imagen que muestran una propiedad distintiva que los hace útiles para aplicaciones donde puntos específicos de la escena necesitan ser encontrados a través de múltiples imágenes.

En la literatura de visión por computadora se pueden encontrar diferentes tipos de detectores de puntos de interés, los cuales pueden ser agrupados de acuerdo a la manera en que modelan o extraen la información de la imagen. De esta manera es posible identificar 2 grandes grupos.

El primero modela las imágenes como superficies tridimensionales, donde la idea es extraer medidas relacionadas directamente con las curvaturas principales que se calculan alrededor de cada punto. Esto incluye a los detectores propuestos por Dreschler y Nagel (1981); Kitchen y Rosenfeld (1982); Beaudet (1978).

El segundo grupo, siendo el más común, utiliza la distribución del gradiente alrededor de cada punto capturado por la matriz del segundo momento como su medida de interés. El método propuesto por Moravec (1977) establece el primer detector de puntos de interés. Los trabajos de Harris y Stephens (1988); Förstner (1994); Shi y Tomasi (1994); entre otros, extienden el concepto propuesto por Moravec. De este grupo, el operador de Harris se ha convertido en el detector más popular que se utiliza en aplicaciones de visión.

La revisión bibliográfica de puntos de interés en este trabajo no intenta ser exhaustiva. El propósito es revisar los principales trabajos, los métodos que serán discutidos a continuación son:

1. Detector Beaudet.
2. Detector de Kitchen y Rosenfeld.
3. Detector de Dreschler y Nagel.
4. Detector de Wang y Brady.
5. Detector de Harris y Stephens.
6. Detector de Förstner.
7. Detectores de Trujillo y Olague.

IV.1 Detector de Beaudet

Beaudet (1978) propone el Operador Rotacional Invariante (DET), basado en el cálculo del determinante Hessiano de la superficie que representa una esquina. El operador propuesto es el siguiente:

$$K_{Beaudet} = Det(H) = \begin{vmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{vmatrix} \tag{16}$$

donde $I(x, y)$ es la función de intensidad de niveles de gris de una imagen, $Det(H)$ es el determinante del Hessiano de la imagen, I_{xx}, I_{xy} y I_{yy} son las segundas derivadas de $I(x, y)$ con respecto a xx, xy y yy respectivamente.

La localización de la esquina consiste en 2 pasos:

1. Se calculan los extremos del DET $K_{Beaudet}$ y se toman sus valores absolutos correspondientes.

2. Si el valor calculado por el operador de Beaudet sobrepasa un umbral definido por el usuario, entonces la esquina se encuentra en las coordenadas (x, y) del máximo encontrado

Las características de este detector son:

1. Otorga un máximo positivo y un mínimo negativo.

2. El mínimo negativo detecta una posición falsa del punto de esquina, por lo tanto se debe tener cuidado al calcular los extremos.

3. El máximo positivo determina la ubicación de la esquina.

4. No es estable en el espacio de escala lineal Gaussiano. Ésto es, si aplicamos distintos factores de difuminado la ubicación de la esquina se altera.

5. Es sensible al ruido. El DET no contempla algun elemento que suavize las regiones de intensidad dentro y fuera de la esquina.

6. Se requiere definir intuitivamente un umbral para considerar la existencia de una esquina, $max(K_{Beaudet}) \geq Umbral$.

En la Figura 26 se ilustra un ejemplo de la respuesta del detector de Beaudet en una imagen con dimensiones 320 × 240 que contiene tres objetos (dos vehículos de juguete y una caja) sobre una mesa. Los objetos se encuentran forrados con una cuadrícula de manera intencional, para ilustrar la tendencia de los puntos detectados a situarse sobre las esquinas de los cuadros. Se puede observar que la mayoria de los puntos detectados se sitúan únicamente sobre los tres objetos, a excepción de un punto que se encuentra sobre el reloj en la parte inferior de la imagen. En este caso se detectaron 234 puntos de interés, representados por pequeños círculos blancos en la imagen.

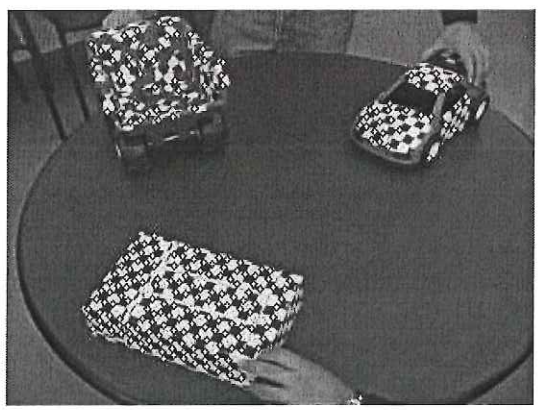


Figura 26: Ejemplo de la respuesta del detector de puntos de interés de Beaudet.

IV.2 Detector de Kitchen y Rosenfeld

Uno de los detectores de esquinas más populares que trabajan con imágenes digitales es el llamado detector K&R propuesto por Kitchen y Rosenfeld (1982), el cual trabaja directamente en la escala de niveles de gris de las imágenes. Su método propone una manera de medir las esquinas basado en el producto de la magnitud del gradiente y en el cambio de su dirección a lo largo de un borde, la cual es la siguiente:

$$K_{K\&R} = \frac{I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_{xy}I_xI_y}{I_x^2 + I_y^2} \tag{17}$$

donde I es la intensidad o nivel de gris de la imagen, I_x denota la derivada parcial de la imagen con respecto a la dirección x, y así sucesivamente.

El detector K&R usa directamente derivadas de primer y segundo orden para obtener la medición de la esquina. Después de que se aplica en cada uno de los puntos de la imagen se realiza una umbralización para aislar las esquinas, la cual consiste en efectuar un mapeo de 0 a 255, que son los valores de niveles de gris, entre los valores resultantes de las mediciones de esquinas. Finalmente, se obtiene una nueva imagen en la cual las esquinas están representadas por los pixeles más luminosos. En la Figura 27 se ilustra un ejemplo de la detección de puntos de interés utilizando el detector K&R, en la imagen con los 3 objetos cuadriculados. En este caso, se detectaron 259 puntos de interés.

Las características del detector son:

1. Existe sólo un mínimo negativo.
2. El máximo del detector K&R está ubicado fuera de la curva que representa la curvatura principal.
3. No es estable en el espacio de escala lineal Gaussiano.
4. Es sensible al ruido.

5. Se requiere definir intuitivamente un valor de umbral para discernir la existencia de una esquina.

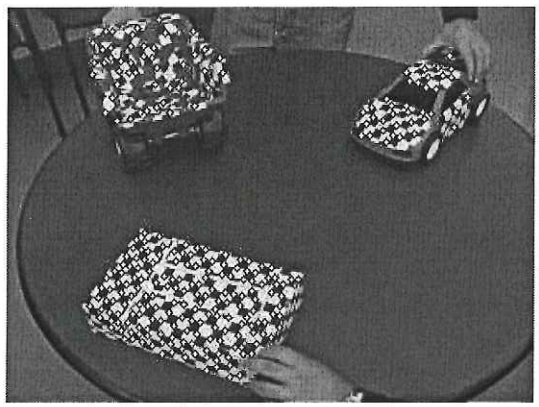


Figura 27: Ejemplo de la respuesta del operador de Kitchen y Rosenfeld.

IV.3 Detector de Dreschler y Nagel

Dreschler y Nagel (1981) proponen un detector basado en la *curvatura Gaussiana principal* de una superficie. Su principal aportación radica en aplicar el concepto de curvatura Gaussiana de una superficie (Geometría Diferencial) con el comportamiento de una esquina en una imagen. El detector que proponen está dado por:

$$K_{D\&N} = \frac{Det(H)}{1 + I_x^2 + I_y^2} \tag{18}$$

donde I_x e I_y son las primeras derivadas parciales de la función de intensidad $I(x, y)$ con respecto a x y y respectivamente. $Det(H)$ es el determinante Hessiano que equivale al detector de Beaudet.

El comportamiento es similar al detector de Beaudet en su forma y propiedades. Sin embargo, el determinante Hessiano no es la expresión exacta que denota la curvatura Gaussiana. El punto de esquina se localiza como sigue:

1. Se obtiene la curvatura Gaussiana dada por $K_{D\&N}$.

2. Se identifica el máximo positivo y el mínimo negativo de la curvatura. El punto máximo positivo representa el extremo cuando $K_{D\&N} > 0$, que se conoce como punto elíptico. El punto mínimo negativo representa el extremo cuando $K_{D\&N} < 0$ y se conoce como punto hiperbólico.

3. El punto de esquina se localiza en la intersección de la línea que une al punto máximo positivo con el mínimo negativo con la curva $K_{D\&N} = 0$ que es la curvatura Gaussiana principal.

Las características del detector son:

1. Sensible al ruido.

2. Se requiere un umbral para discernir la existencia de una esquina.

3. No es estable en el espacio lineal Gaussiano.

IV.4 Detector de Wang y Brady

Wang y Brady (1995) proponen otra medida de curvatura par detectar esquinas. El valor de la curvatura $K_{W\&B}$ es proporcional a la segunda derivada tangencial en la dirección del borde e inversamente proporcional a la magnitud de su velocidad de cambio. El detector está dado por la siguiente expresión:

$$K_{W\&B} = (\nabla^2(I))^2 - S|\nabla(I)|^2 \tag{19}$$

donde ∇^2 es el operador laplaciano, ∇ es el operador gradiente y S es una constante para la supresión de respuesta falsa de una esquina. S se determina empíricamente por el usuario.

La ubicación de la esquina se localiza calculando el máximo del $K_{W\&B}$. Una de las propiedades más importantes del detector W&B radica en su rápida ejecución, como el que se requiere en aplicaciones de tiempo real como la estimación de escenas en movimiento. En la Figura 28 se ilustra la respuesta al detector de Wang y Brady para la imagen con los tres objetos cuadriculados. En este caso se detectaron 158 puntos de interés.

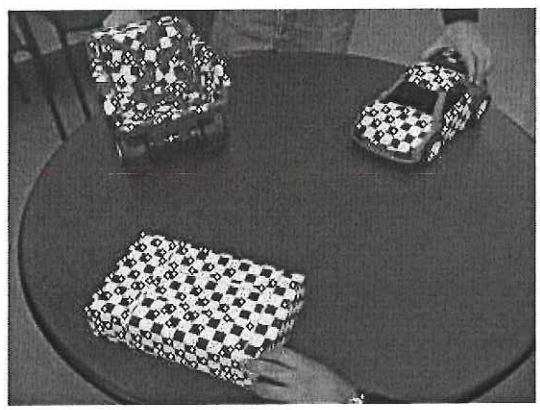


Figura 28: Ejemplo de la respuesta al detector de puntos de interés de Wang y Brady.

IV.5 Detector de Harris y Stephens

Harris y Stephens (1988) proponen un detector basado en el detector de puntos propuesto por Moravec (1977) y de la función de autocorrelación. Los autores estaban interesados en utilizar técnicas de análisis de movimiento para interpretar el ambiente de robots basado en imágenes de una sola cámara móvil. Harris y Stephens necesitaban un método para corresponder puntos en marcos consecutivos de la imagen como el de Moravec, pero estaban interesados en rastrear tanto esquinas como bordes. La matriz

de auto-correlación está dada por:

$$\mu = \begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix}$$

Con lo cual el detector está dado por

$$K_{H\&S} = Det(\mu) - kTr^2(\mu) \tag{20}$$

donde μ es la matriz de auto-correlación y k es la llamada constante de Harris, que en la práctica suele tener un valor aproximado de 0.04. De esta forma el criterio de ubicación del punto de interés se simplifica al tomar los puntos que estén arriba de un umbral definido por el usuario. En la Figura 29 se muestra un ejemplo del detector de H&S a la imagen con tres objetos cuadriculados. En este caso, se detectaron 171 puntos de interés que se observan en círculos blancos en la figura.

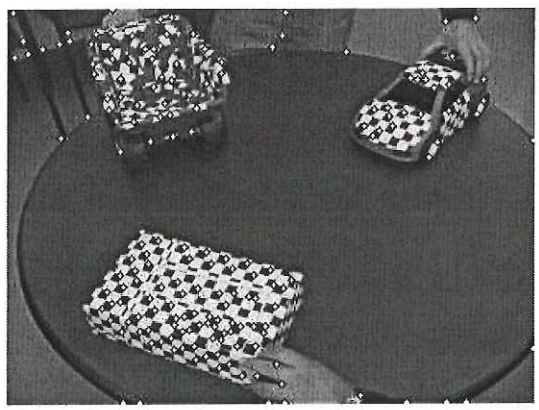


Figura 29: Ejemplo de la respuesta al detector de puntos de interés de Harris y Stephens.

IV.6 Detector de Förstner

El detector de Förstner (1994) puede ser visto como un desarrollo estadístico y analítico, que se formula como un problema de encontrar puntos apropiados para una descripción

de la imagen. Esto implica que dada una región, ésta debe contener estructuras sobresalientes que describan la imagen. Förstner realizó observaciones en problemas como: correspondencia de imágenes por medio de *mínimos cuadrados*, detección de esquinas a partir de la intersección de bordes, la búsqueda del centro de gravedad de una región de la imagen, y la búsqueda del centro de elementos circulares. A partir de estas observaciones encuentra la similitud de los procesos y propone una métrica a partir de la ecuación normal, que es equivalente a la matriz de autocorrelación. En la Figura 30 se muestra un ejemplo de la respuesta al operador de Förstner para detectar puntos de interés en la imagen con tres objetos cuadriculados. En este caso se detectaron 300 puntos, que se observan como pequeños círculos blancos en la imagen.

El detector está dado por

$$K_F = \frac{Det(\mu)}{Tr(\mu)} \tag{21}$$

donde $Tr(\mu)$ es la Traza de la matriz de auto-correlación.

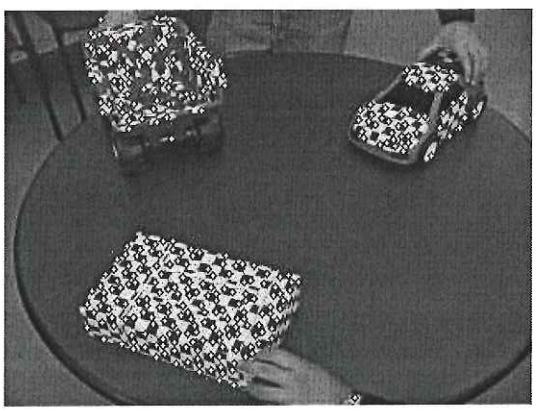


Figura 30: Ejemplo de la detección de puntos de interés utilizando el operador de Förstner.

IV.7 Detector de Trujillo y Olague

Trujillo y Olague (2006) proponen dos nuevos detectores de puntos de interés evolucionando los operadores utilizando Programación Genética (ver Capítulo 5): IPGP1 e IPGP2 (por sus siglas en inglés *Interest Point Genetic Programming*). Donde se propone un nuevo enfoque al problema de detección de puntos de interés como un problema de optimización. El criterio que se utiliza para encontrar la solución es descrito por 3 características principales:

1. **Separación Global** entre los puntos extraídos. Es decir, que los puntos seleccionados como prominentes en una imagen, no sean los mismos.

2. **Alta Información de Contexto** en comparación a otros pixeles.

3. **Estabilidad** sobre ciertos tipos de transformaciones en la imagen, como la rotación. Una métrica importante en este aspecto es la repetibilidad.

El problema es resuelto a través de Programación Genética (GP). Donde cada individuo en la población del GP representa un operador candidato de puntos de interés. Para definir una aplicación de un GP es necesario definir tres elementos: la función aptitud, un conjunto de funciones y un conjunto de terminales.

En este caso la función de aptitud se determina como el producto de la razón de repetibilidad calculada para un conjunto de n imágenes de entrenamiento, las funciones sigmoidales para promover puntos de dispersión a lo largo de las direcciones x y y, y un factor de penalización que reduce la aptitud para los detectores que devuelven un número de puntos menor a los requeridos.

Con las consideraciones anteriores se obtuvo un primer detector dado por

$$K_{IPGP1} = G_{\sigma=2} * (G_{\sigma=1}(I) - I) \tag{22}$$

Nótese que IPGP1 es un detector con una estructura muy sencilla. Lo cual tiene ventajas como un procesamiento rápido y una fácil implementación. Básicamente IPGP1 extrae el punto en dos pasos:

1. Se extraen las frecuencias altas de la imagen. Ésto se lleva a cabo al substraer la imagen original de una imagen que ha sido convolucionada con un filtro Gaussiano, el cual tiene el efecto de suprimir las frecuencias más altas.

2. Se suaviza la imagen resultante de la operación anterior. Es decir se convoluciona la imagen resultante con otro filtro Gaussiano.

IPGP1 tiene algunas características interesantes como:

1. No utilizar derivadas para la extracción de puntos.

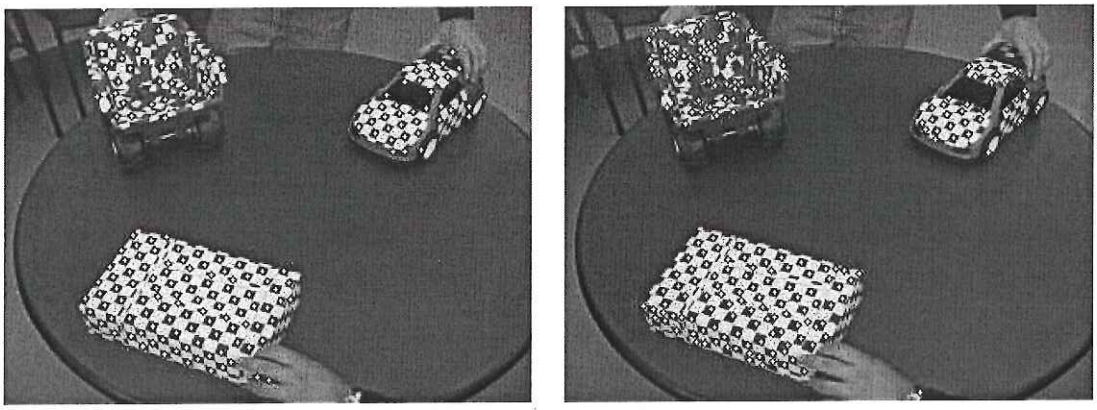
2. Tiene una repetibilidad del 95% que es comparable con detectores como el propuesto por Harris.

El criterio de ubicación del punto es por medio de un umbral indicado por el usuario.

El segundo detector que se obtuvo es el siguiente

$$K_{IPGP2} = (G_{\sigma=1} * (I_{xx} \cdot I_{yy})) - (G_{\sigma=1} * (I_{xy} \cdot I_{yx})) \tag{23}$$

La cual representa una versión modificada del operador DET propuesto por Beaudet. Tiene la misma estructura que el DET, con la diferencia que es promediada alrededor de un vecindario local con una función de suavizado Gaussiano. De esta forma se podría decir que el detector de Beaudet fue redescubierto por la Programación Genética. En la Figura 31 se ilustra la respuesta de los detectores IPGP1 e IPGP2 para la imagen con 3 objetos cuadriculados. Para esta imagen, se detectaron 171 puntos de interés con el operador IPGP1 y 192 puntos con IPGP2.



a) IPGP1 b) IPGP2

Figura 31: Ejemplo de la detección de puntos de interés de los operadores evolucionados con Programación Genética.

IV.8 Conclusiones

En este capítulo se discutieron diversos métodos para detectar puntos de interés. En resumen, una esquina está formada por la unión de dos o más bordes. Un borde es la unión de dos o más regiones en una imagen bidimensional.

En el capítulo VI se analizan los puntos de interés como energía externa para una snake y se plantea una nueva forma de inicializar snakes utilizando los puntos de interés.

Capítulo V

Introducción a la Computación Evolutiva

La *Computación Evolutiva* (EC por sus siglas en inglés de *Evolutionary Computation*) es un área de investigación dentro de las Ciencias de la Computación, y tiene su inspiración en el proceso de la evolución natural tal como la genética biológica y la selección natural.

Este capítulo provee información básica para utilizar las herramientas de Computación Evolutiva para resolver problemas prácticos. Se presentan los conceptos básicos y terminología utilizada, y a continuación los paradigmas que se han desarrollado para ilustrar los conceptos básicos.

V.1 Paradigmas tradicionales de búsqueda y optimización vs paradigmas de cómputo evolutivo

Existen muchos métodos para resolver problemas de búsqueda y optimización. La mayoría de los paradigmas de optimización tradicionales se mueven en el espacio de decisiones de un punto a otro, utilizando alguna regla determinística para encontrar la solución óptima. Una de las desventajas principales de estas técnicas es la tendencia de quedarse atrapado en un óptimo local.

En la mayoría de los casos, un óptimo global existe en un punto del espacio de decisiones. Sin embargo, puede existir un ruido estocástico o caótico que afecte la localización del óptimo global. Algunas veces el óptimo global cambia dinámicamente por influencias externas y frecuentemente existen muy buenos óptimos locales. Por estas y muchas otras razones, a menudo es irracional esperar que algún método de

optimización encuentre el óptimo global (aunque si exista) en un tiempo finito. Lo mejor que se puede esperar es encontrar soluciones muy cercanas al óptimo y que el tiempo que se tome para encontrarlo se aproxime de forma menor a la exponencial con respecto al número de variables.

Los paradigmas de Computación Evolutiva difieren de los paradigmas tradicionales de búsqueda y optimización de cuatro maneras principales:

1. Utilizan una *población* de puntos (soluciones potenciales) en su búsqueda.

2. Utilizan información directa de la *aptitud* en lugar de una función derivada u otro conocimiento relacionado.

3. Utilizan reglas de transición probabilísticas en vez de determinísticas.

4. Las implementaciones de EC generalmente codifican los parámetros en símbolos binarios u otro tipo de símbolos, en lugar de trabajar con los parámetros por si mismos.

Los paradigmas de EC comienzan con una población de puntos, en lugar de una sola solución. Por lo general generan una nueva población con el mismo número de miembros en cada iteración o *generación*. Por lo que muchos máximos o mínimos pueden ser explorados simultáneamente, disminuyendo la probabilidad de quedar atrapado. Operadores como el *cruzamiento* y *mutación* mejoran efectivamente esta capacidad de búsqueda paralela, permitiendo a la búsqueda cambiar de un espacio a otro.

Los paradigmas de EC no requieren información auxiliar al problema, como lo es la función derivada. Muchos paradigmas de búsqueda de *hill-climbing*, por ejemplo, requieren el cálculo de derivadas para poder explorar el máximo local. En los paradigmas de optimización EC la aptitud de cada miembro de la población se calcula del valor de la función que se quiere optimizar, y es común utilizar la función de salida como la

medida de la aptitud. La aptitud es una métrica directa del desempeño de un individuo de la población en la función que se optimiza.

El hecho que los paradigmas EC utilicen reglas probabilísticas no significa que estrictamente se lleve acabo una búsqueda aleatoria. Lo operadores estocásticos se aplican en operaciones que direccionan la búsqueda hacia regiones del espacio de búsqueda que puedan tener los valores de aptitud más altos. Por lo que, por ejemplo, la *selección* de individuos de la población para reproducción frecuentemente se realiza con una probabilidad que es proporcional al valor de aptitud del individuo.

Algunos paradigmas EC, especialmente el *algoritmo genético canónico*, utiliza una codificación especial de los parámetros del problema que se quiere resolver. En los algoritmos genéticos, los parámetros por lo general se codifican como cadenas binarias, sin embargo, cualquier alfabeto finito se puede utilizar. Estas cadenas casi siempre son de una longitud fija con un número fijo de unos y ceros, en el caso de la cadena binaria, que se pueden asignar a cada parámetro. Una longitud fija significa que la longitud de la cadena no varía durante la ejecución de un algoritmo evolutivo. La longitud de la cadena (número de bits para una cadena binaria) asignado a cada parámetro depende del rango máximo y la precisión requerida para el problema que se quiere resolver.

V.2 La evolución natural

Para mover la población de soluciones sobre el espacio de búsqueda, la EC utiliza el principio de evolución natural, donde los individuos más aptos sobreviven y prosperan. Darwin (1859) realizó una extensa investigación sobre la evolución natural e identificó 3 principios:

1. El ciclo reproductivo.

2. La selección natural.

3. La diversidad por variación.

El ciclo reproductivo explica la naturaleza iterativa de la evolución mediante el proceso de nacimiento y muerte. Todos los individuos son mortales, pero a través de la reproducción crean hijos que los reemplazan. Los descendientes forman la siguiente generación de la especie, y algunas veces, forman una nueva variedad de la especie mediante aislamiento geográfico y especialización.

La selección natural es el proceso mediante el cual los individuos se adaptan al entorno que los rodea. En este proceso, los individuos que llevan a cabo sus funciones vitales de una mejor manera tienen mayores probabilidades de sobrevivir, y son ellos los que consiguen reproducirse. De esta manera, en cada generación de individuos aparecen nuevas alteraciones que convierten a los descendientes en mejores o peores individuos que sus padres en la capacidad de adaptarse a su entorno, consiguiendo la diversidad por variación.

El entorno se encarga de decidir si las variantes incorporadas a la especie permanecen o no. Si los descendientes llegan a reproducirse con dichos cambios, tienen la posibilidad de tranmitirse de nuevo a sus propios hijos y el nuevo rasgo puede llegar a conservarse hasta que ya no sea de utilidad a la especie.

Darwin no pudo encontrar la explicación del porqué los individuos tenían la capacidad de variación; sin embargo, Gregor Mendel poco después ofreció la respuesta al introducir las bases de la genética. Cada *gene* posee información codificada sobre un individuo y durante el proceso reproductivo se combinan las características de ambos progenitores para crear a sus descendientes, a lo que se le conoce como *cruzamiento*. De esta manera los hijos resultan ser diferentes de ambos padres y no una réplica de uno de ellos. Además, durante la combinación de genes, periódicamente se cometen errores y en consecuencia se producen *mutaciones* que provocan una nueva diferenciación de los hijos con respecto a los padres.

Estas ideas inspiraron a numerosos científicos a aplicar conceptos similares para la resolución de problemas de búsqueda y optimización en sus respectivos campos. De esta manera, es como nace el Cómputo Evolutivo. Las primeras aplicaciones de estas técnicas evolutivas aparecen en los años setentas, y en la actualidad se han desarrollado un gran número de técnicas consideradas como evolutivas.

V.3 Técnicas evolutivas

Muchos investigadores en el campo de la Computación Evolutiva consideran que ésta incluye cuatro áreas (Kennedy y Eberhart, 2001):

1. Algoritmos Genéticos.
2. Programación Evolutiva.
3. Estrategias de Evolución.
4. Programación Genética.

De estas cuatro metodologías, el mayor trabajo se ha realizado en Algoritmos Genéticos, por lo que las secciones siguientes se enfocan más a esa área. Sin embargo, híbridos de las 4 metodologías así como híbridos con otras herramientas de inteligencia computacional, como las *redes neuronales*, se han vuelto cada vez más populares.

V.3.1 Algoritmos genéticos

Los Algoritmos Genéticos (GAs por sus siglas en inglés Genetic Algorithms) fueron propuestos por John Holland y popularizados por Goldberg. Esta técnica se basa en la genética de los organismos vivos, la cual establece que la estructura fundamental que contiene los rasgos de un individuo se le conoce como *cromosoma*. Uno o más cromosomas especifican un organismo como un todo. Al conjunto de cromosomas se le conoce

como *genotipo*, y a su traducción en rasgos físicos; es decir, al organismo resultante se le llama *fenotipo*. Un cromosoma consta de un cierto número de subestructuras individuales llamadas *genes*. Cada gene codifica una característica particular del organismo; y la localización o *locus* del gene dentro del cromosoma determina cuál de varios valores diferentes de esta característica representa al gene . A los diferentes valores que un gene puede representar se les conoce como *alelos*.

En un GA, cada solución representa un *cromosoma*, y un conjunto de cromosomas forman la población de soluciones. Las variables de decisión que conforman cada cromosoma son codificadas en cadenas binarias, y se les llama *genes*. La posición de cada variable de decisión es el *locus* de la variable, y similarmente, los *alelos* representan los diferentes valores que una variable de decisión puede tomar. El *genotipo* es la concatenación de unos y ceros que forman el cromosoma, mientras que el *fenotipo* es la decodificación del cromosoma en números reales para su evaluación en la función de aptitud.

Los GAs basan su esfuerzo para resolver el problema en alfabetos de baja cardinalidad para codificar la información (generalmente, un alfabeto binario) y en la capacidad del operador de cruzamiento para construir bloques; ésto es, cadenas cortas que pueden ser optimizadas independientemente una de otra para después juntarlas y conformar la solución óptima. La mutación se aplica escasamente mediante eventos aleatorios que consisten en intercambiar el caracter por otro del pequeño alfabeto, además sólo se aplica a un pequeño número del total de la población.

V.3.2 Programación evolutiva

La Programación Evolutiva (EP por sus siglas en inglés Evolutionary Programming) es similar a los Algoritmos Genéticos en el uso de una población de soluciones candidatas para evolucionar una solución óptima en un problema específico. Sin embargo, difieren

en que la EP se deriva de la simulación de la conducta de adaptación en la evolución, mientras que los GAs se derivan de la simulación de la genética. La diferencia es sutil pero importante, los GAs trabajan en el *espacio del genotipo* de la codificación de información, mientras la EP trabaja en *espacio del fenotipo* de conductas observables.

Una aportación significativa a la metodología básica de EP es la característica de autoadaptación, lo cual provee la capacidad de los parámetros estratégicos de evolucionar por si mismos, y por lo tanto direccionar a la mutación en un espacio de búsqueda más prominente. Los tres tipos principales de la Programación Evolutiva se conocen como EP estándar, meta-EP, y Rmeta-EP; que se distinguen por diferentes niveles de autoadaptación (Bentley, 1999).

El procedimiento que por lo general sigue la EP es el siguiente:

1. Inicializar la población.
2. Exponer la población al medio ambiente.
3. Calcular la aptitud para cada miembro.
4. Realizar la mutación aleatoriamente para cada *padre* en la población.
5. Evaluar padres e hijos.
6. Seleccionar miembros para la nueva población.
7. Regresar al paso 2 hasta que un criterio de paro se cumpla.

La población se inicializa de manera aleatoria y su tamaño depende del problema en particular. Generalmente la Programación Evolutiva se basa en la mutación para producir decendencia y no en el cruzamiento. La EP usualmente genera el mismo número de hijos que los padres. Los padres se seleccionan utilizando el método de selección por torneo y sus características son mutadas para producir hijos que son

agregados a la población. Cuando la población se ha duplicado, todos los miembros incluyendo padres e hijos se clasifican por rango y la mejor mitad se mantienen para la siguiente generación.

V.3.3 Estrategias de evolución

La técnica de Estrategias de Evolución (ES por sus siglas en inglés Evolution Strategies) fue desarrollada de manera independiente por Ingo Rechenberg y Schwefel (1995); y se basa en el mecanismo de selección natural de Darwin. En este caso la población se conforma de individuos que compiten con los demás por su derecho a reproducirse y/o sobrevivir a través de las generaciones. Cada individuo se compone de un vector de valores que pertenecen al conjunto de los números reales y representan el conjunto de variables de decisión. Las ES son diferentes a los GAs en dos aspectos principales:

1. Basan su esfuerzo evolutivo en la *fuerza de mutación* y aplican poco los operadores de cruzamiento, de hecho al principio las ES no contaban con un operador de cruzamiento.

2. No codifican la representación del problema, trabajan directamente en el espacio de búsqueda del problema.

Existen diferentes tipos de Estrategias de Evolución:

- **ES Bimiembro (1+1).** Ésta es la forma más simple, en cada generación se toma un sólo padre para crear un hijo aplicando el operador de mutación. Y si el valor de aptitud del hijo resulta mejor que el del padre, se conserva el hijo y se desecha el padre; o viceversa.

- **ES Multimiembro $(\mu+\lambda)$ y (μ, λ).** En este tipo de ES, un tamaño determinado de población μ se utiliza para generar λ hijos. Existen dos formas de aplicar el concepto de multimiembro. En la primera, llamada estrategia *mas*, la población

para la nueva generación se elije de entre los mejores padres e hijos. Mientras que en la segunda, llamada estrategia *coma*, sólo toma en cuenta a los hijos y los padres son desechados después de cada generación. En $(\mu + \lambda)$ los padres pueden sobrevivir a través de las generaciones, lo cual puede controlarse mediante un parámetro κ. De este modo, si $\kappa = n$ se permite que un individuo sobreviva por n generaciones. Nótese que una $(\mu + \lambda)$ con $\kappa = 1$ es en realidad una (μ, λ).

- **ES Autoadaptable**. La autoadaptibilidad permite a las ES ser flexibles y más cercanas a la evolución natural, fue desarrollado por Bäck (Deb, 2001) y se puede aplicar para los dos tipos de ES anteriores. La autoadaptación es recomendable en el caso de que se necesite conocer la solución óptima con una precisión arbitraria; o bien, cuando la función objetivo y/o la solución óptima cambien con el tiempo. Existen tres maneras diferentes de aplicar la autoadaptibilidad en una estrategia de evolución:

  1. *Meta-ES.* Consiste en utilizar dos niveles de evolución; en el primero se optimizan los parámetros del tipo ES Multimiembro, y en la segunda se trabaja con el problema de manera normal, pero con los parámetros que el primer nivel arrojó.

  2. *Matriz de covarianza.* Consiste en registrar el comportamiento de la población para cierto número de generaciones y calcular su varianza y covarianza, modificando los parámetros de los operadores de acuerdo al resultado que se desea en las siguientes generaciones.

  3. *Uso explícito de parámetros de control autoadaptables.* La idea es similar a la aplicación anterior, sólo que aqui las varianzas y covarianzas forman parte del vector solución, evolucionando junto con las variables de decisión. En consecuencia se debe modificar la función aptitud para que se considere las

nuevas variables.

V.3.4 Programación genética

Las tres áreas de la Computación Evolutiva discutidas anteriormente involucran estructuras de individuos que se definien como cadenas. Algunas utilizan cadenas de valores binarios, otras variables codificadas con números reales, pero todas son cadenas o vectores. La Programación Genética (GP por sus siglas en inglés Genetic Programming) trata de evolucionar programas de cómputo jerárquico que se representan generalmente como estructuras de árbol. Además, mientras que las estructuras de indiduos utilizadas en los otros paradigmas generalmente tienen una longitud fija, los programas que se evolucionan por GP usualmente varían en tamaño, forma y complejidad.

Otra perspectiva es que los GP son un subconjunto de los GA que evolucionan programas ejecutables. Las diferencias entre un GP y un GA genérico son las siguientes:

- Los miembros de la población son estructuras ejecutables (generalmente, programas de cómputo) en lugar de cadenas de bits y/o variables.
- La aptitud de un individuo en un GP se mide ejecutándolo.

La meta de un GP es "descubrir" un programa de cómputo en el espacio de programas buscado, que resulta en una salida deseada para un conjunto de entradas predefinidas. En otras palabras, una computadora busca como escribir su propio código.

Cada programa se representa como un árbol, donde las funciones que se definen para el problema aparecen en los puntos internos del árbol, y las variables y constantes se localizan en los puntos externos (hojas) del árbol. La naturaleza de los programas de computadora generados dan a la GP una inherente naturaleza jerárquica.

En la preparación de una implementación de un GP, cinco pasos se toman en cuenta:

1. Especificar el conjunto de terminales.

2. Especificar el conjunto de funciones.

3. Especificar la medida de aptitud.

4. Seleccionar los parámetros de control del sistema.

5. Especificar las condiciones de terminación.

El conjunto de terminales comprende las variables (las variables de estado del sistema) y constantes asociadas con el problema que se desea resolver. Las funciones seleccionadas para el conjunto de funciones son limitadas sólo por el lenguaje de programación que se utiliza para correr el programa evolucionado por el GP. Por lo tanto, pueden incluir funciones matemáticas (*cos*, *exp*, etc.), operaciones aritméticas (+, *, etc.), operadores booleanos (AND, NOT, etc.), operadores condicionales (if-then-else), y funciones iterativas y recursivas. Cada función en el conjunto de funciones requiere un cierto (predefinido) número de argumentos, que se conoce como la *aridad* de la función. Las terminales son funciones con aridad cero. Una de las tareas para especificar el conjunto de funciones es seleccionar un conjunto mínimo capaz de cumplir con la tarea. Esto último lleva a las dos propiedades que se desean en cualquier aplicación de un GP: *cerradura* y *suficiencia*.

Para satisfacer la propiedad de cerradura, cada función debe ser capaz de operar exitosamente en cualquier función en el conjunto de funciones y en cualquier valor de los tipos de datos que pueden tomar los miembros del conjunto de terminales. En ocasiones ésto requiere la definición de casos especiales para las funciones. Las funciones que son redefinidas para que regresen valores válidos son llamados *funciones protegidas*. Si la propiedad de cerradura no se satisface, se debe especificar un método para tratar con los miembros inválidos de la población, y con los miembros cuya aptitud no es aceptable.

Para satisfacer la propiedad de suficiencia, el conjunto de funciones y el conjunto de terminales deben ser lo suficientemente extensos para permitir que se evolucione una

solución. Es decir, alguna combinación de funciones y terminales debe ser capaz de producir una solución. Por lo general se requiere algún conocimiento del problema para poder juzgar cuándo se cumple la propiedad de suficiencia. En algunos problemas, la suficiencia es relativamente fácil de determinar. Por ejemplo, si se utilizan funciones booleanas, el conjunto de funciones AND, OR, NOT es suficiente para cualquier problema. Para otros problemas, puede ser relativamente difícil establecer la suficiencia.

Se ha encontrado que tener más que el mínimo número de funciones puede degradar el desempeño en algunos casos y mejorar significativamente en otros. Sin embargo, el tener muchas terminales por lo general degrada el desempeño (Koza, 1992).

La medida de aptitud usualmente es inversamente proporcional al error producido por la salida del programa. Otras medidas de aptitud también son comunes, como la puntuación que alcanza un programa en una corrida.

Los dos principales parámetros de control son el tamaño de la población y el máximo número de generaciones. Otros parámetros incluyen la probabilidad de reproducción, la probabilidad de cruzamiento, y el máximo tamaño permitido (como la medidad de profundiad o niveles jerárquicos del árbol) en las poblaciones del programa inicial y final.

La condición de terminación por lo general se determina por el número máximo de generaciones. El programa ganador es usualmente el mejor programa (en términos de la aptitud) creado en una determinada generación.

Una vez que los pasos de preparación se completan, el proceso de un GP puede ser implementado de la siguiente manera:

1. Inicializar la población de programas computacionales.

2. Determinar la aptitud para cada programa individual.

3. Realizar la reproducción de acuerdo a los valores de aptitud y la probabilidad de

reproducción.

4. Realizar el cruzamiento de las subexpresiones.

5. Regresar al paso 2 si la condición de terminación no se ha cumplido.

V.4 Estructura general

Independientemente del paradigma que se quiera implementar, las herramientas de la Computación Evolutiva generalmente siguen un procedimiento similar:

1. Inicializar la población

2. Evaluar la aptitud de cada individuo en la población.

3. Seleccionar los individuos para reproducir y formar una nueva población.

4. Reproducir la población utilizando las operaciones evolutivas, como el cruzamiento o mutación.

5. Regresar al paso 2 hasta que un criterio de terminación se cumpla.

Al igual que en la teoría de la evolución natural, en los algoritmos evolutivos los individuos deben comprobar que son aptos para sobrevivir y/o reproducirse mediante un criterio de evaluación al que se le conoce como *función aptitud.* Este criterio califica el lugar que ocupan en el espacio de búsqueda, y aquellos que obtengan los valores más altos tienen mayor probabilidad de ser seleccionados para el proceso de reproducción. La reproducción se lleva a cabo mediante los llamados *operadores evolutivos.* En ella se combina la información de uno o varios individuos (no necesariamente dos) para dar lugar a nuevos individuos. Este proceso se repite iterativamente, es decir, generación tras generación. De esta manera el espacio de búsqueda se va explorando, moviéndose en la dirección que indiquen los mejores individuos.

V.4.1 Inicialización de la población

Por lo general en la inicialización se genera la población con valores aleatorios. Algunas veces, si es factible, se puede inicializar la población con valores relativamente cerca del óptimo. El número total de individuos que se requieren para formar una población depende del problema y el paradigma que se utilice.

La elección de una buena representación de la solución es muy importante para cualquier problema de búsqueda. En el caso de los algoritmos evolutivos es importante que los individuos se diseñen de tal manera que resuelvan el problema satisfactoriamente.

V.4.2 Evaluación

La evaluación de un individuo significa calcular su valor con la función objetivo, penalizándolo si incurre en la violación de alguna restricción, cuantificando así su posición en el espacio de búsqueda. Esta métrica que considera la función objetivo y las restricciones para asignarle un mérito relativo a cada individuo se le conoce como *función de aptitud.* La función de aptitud se diseña específicamente para cada problema y tiene la capacidad de asignar a un individuo un valor de aptitud que indica que tan bueno o malo es como solución del problema.

El valor de la aptitud frecuentemente es proporcional al valor de salida de la función que se quiere optimizar, aunque también puede ser derivada de alguna combinación o número de funciones de salida. La función de aptitud, que también se conoce como *función de evaluación*, toma como entradas las salidas de una o más funciones, dando como salida una probabilidad de reproducción. Algunas veces es necesario transformar la salida de la función para producir un métrica apropiada de aptitud. Para algunos algoritmos evolutivos, sólo un pequeño porcentaje de tiempo computacional es requerido para el algoritmo, y la mayoría del tiempo es para evaluar la aptitud.

V.4.3 Selección

La selección de individuos a través de reproducción sirve para generar una nueva población (que se conoce como *generación*), y por lo general se basa en los valores de aptitud. Entre mayor sea la aptitud, más posibilidades tiene un individuo de ser seleccionado para la nueva generación. Sin embargo, existen otros paradigmas que se consideran evolutivos, que pueden retener a todos los miembros de la población de generación en generación. Es decir, si no hay un operador de selección, entonces todos los miembros de la población producen descendientes con la misma probabilidad.

El principal objetivo del operador de selección es identificar buenas soluciones y determinar que individuos dejarán descendencia y en que cantidad. De esta manera, la búsqueda se orienta hacia aquellas soluciones más prominentes. La selección permite que los individuos más aptos se reproduzcan con mayor frecuencia, negando en ocasiones el derecho de reproducción de algunos individuos menos aptos. Existen varias maneras de llevar a cabo la selección. Las más comunes son: selección por torneo, selección proporcional y selección por rango.

Selección por torneo

La selección por torneo es de las más populares y sencillas. En ella se eligen dos individuos que compiten entre si comparando su valor de aptitud. El derecho de reproducirse lo gana aquel con un valor menor o mayor, según se esté minimizando o maximizando el problema. Este proceso se repite hasta que se obtengan el número de ganadores necesarios para generar la nueva población. La elección de los competidores puede ser sistemática o aleatoria. Independientemente del método de elección, es muy probable que el mismo individuo participe en varias competencias, pudiendo ganar el derecho de reproducirse varias veces. Aquel individuo con el mayor valor de aptitud ganará todos los torneos en los que participe, permitiendo la conservación de sus características en la

nueva población. Mientras que el que tenga menor valor de aptitud perderá los torneos, condenándose a la extinción.

Selección proporcional

En este tipo de selección, el número de veces en que cada individuo obtiene el derecho de reproducción es proporcional a su valor de aptitud, entre mayor valor de aptitud, mayor número de veces lo obtendrá. El mecanismo de este operador funciona como una ruleta, donde la rueda tiene N divisiones que representan el tamaño de la población y el ancho de cada división está en proporción a la función aptitud de cada miembro. La ruleta girará M veces, según el número de padres que necesite la nueva población, eligiendo cada vez al individuo que señale la ruleta.

Selección por rango

La selección proporcional puede llegar a tener un problema de *escalamiento*. Por ejemplo, si en una población existe un individuo con un valor de aptitud mucho más grande que el resto de la población, ocupará la mayor parte de la superficie de la ruleta provocando su selección con demasiada frecuencia, lo cual puede provocar la convergencia hacia un óptimo local. Una manera de evitar este problema es mediante la selección por rango. A cada individuo se le asigna un rango de acuerdo a la magnitud de su valor de aptitud, asignándole al de menor valor el número 1, y al de mayor valor N. Entonces a cada individuo se le asigna un nuevo valor de aptitud proporcional al rango. Después se aplica el operador de selección proporcional con los nuevos valores, de manera normal.

V.4.4 Reproducción

La reproducción consiste en aplicar uno o más operadores (llamados *operadores evolutivos*) sobre los individuos seleccionados (llamados *padres*) en la etapa de selección y

crear los individuos que conformarán la siguiente generación (llamados *hijos*). La aplicación del operador se hace por cada componente del vector de valores del individuo. Los operadores evolutivos y una buena representación de la solución del problema garantizan el buen rendimiento de un algoritmo evolutivo. Al igual que la representación del problema, los operadores evolutivos que se utilicen dependen del tipo de problema que se quiere resolver.

Los operadores más comunes que se aplican son: la *mutación* y el *cruzamiento.* Existen operadores específicos para problemas en los que se desea obtener más de una solución óptima. Uno de ellos es el de *repartición.* Por lo regular, la aplicación de estos operadores es estocástica; es decir, su aplicación depende de una función de probabilidad. Aunque en algunos casos es determinística; es decir, se aplica siempre ya sea a toda la población o sólo a una parte de ella. A continuación se describen los operadores de mutación, cruzamiento y repartición.

Mutación

La mutación altera aleatoriamente la información de sólo un individuo para crear un nuevo hijo. La mutación evita la pérdida de diversidad en la población, por lo que se le considera un operador de *exploración* del espacio de búsqueda. Al igual que en la naturaleza, la mutación incrementa el potencial de variación entre las generaciones de la población. A pesar que la utilización de los operadores evolutivos depende del problema, un operador de mutación debe considerar los siguiente aspectos para obtener un buen rendimiento.

1. **Alcanzabilidad**: ya que el operador realiza movimientos a través del espacio de búsqueda, se espera que pueda trasladar a un individuo de un punto p a un punto q en un cierto número de generaciones g, donde $g < \infty$. La alcanzabilidad se realiza paulatinamente mediante lo que se conoce como "saltos" en el espacio de

búsqueda.

2. **Escalabilidad**: la longitud de los saltos (fuerza de mutación) debe acoplarse a la cercanía del individuo a una solución óptima. Entre más cercano se encuentre, el salto debe ser más pequeño.

A continuación se presentan algunos operadores de mutación.

Mutación binaria

Este tipo de mutación sólo es útil para representaciones con cadenas binarias (0, 1). Consiste en cambiar cada bit del vector de valores del individuo de 1 a 0, o viceversa, de acuerdo a una probabilidad de mutación p_m. Ésto implica generar un número aleatorio por cada bit, lo que puede resultar computacionalmente costoso. Para evitarlo Goldberg (1989) desarrolló una *mutación de reloj*, que después de que un bit ha sido mutado, determina la localización del siguiente bit a mutar con una distribución exponencial. El mecanismo es simple, primero se genera un número aleatorio $r \in [0, 1]$, y después se calcula el siguiente bit a mutar saltándose los siguientes $\eta = -p_m ln(1 - r)$ bits.

Mutación aleatoria

Este operador fue propuesto por Michalewicz (1982). La forma más simple de alterar la información de un individuo es aplicarle una función aleatoria considerando todo el espacio de búsqueda

$$y_i = r_i(x_i^{max} - x_i^{min})$$

donde r_i es un número aleatorio en el intervalo [0, 1] que sigue una distribución uniforme, y x_i^{max}, x_i^{min} son los límites superior e inferior respectivos de la variable de decisión (ver Figura 32(a)).

Mutación no uniforme

También propuesto por Michalewicz (1982). En este caso la probabilidad de crear un nuevo individuo cerca del padre es mayor que crearlo lejos del mismo. Sin embargo, conforme avanzan las generaciones, la probabilidad de crear al individuo más cerca, va creciendo

$$y_i = x_i + \tau(x_i^{max} - x_i^{min})(1 - a_i^{\left(1-\frac{g}{g_{max}}\right)^b})$$

donde τ sólo puede tomar dos valores: 1 o -1, con una probabilidad de 0.5. El parámetro g_{max} es el número máximo de generaciones, mientras que b es un parámetro definido por el usuario. El comportamiento de este operador es similar al de la mutación aleatoria durante las primeras generaciones, mientras que en las últimas se comporta como la *función de Dirac* (ver Figura 32(b)).

Mutación normal

Un operador simple y popular es la mutación normal, la cual utiliza una distribución Gaussiana para provocar la alteración de un individuo. La mutación se aplica por cada componente del vector de valores que compone el individuo

$$y_i = x_i + N(0, \sigma_i)$$

donde el parámetro σ_i determina la fuerza de mutación, y es un valor fijo definido por el usuario. Es muy importante que su valor se establezca correctamente de acuerdo al problema para que el operador funcione correctamente. Nótese que habrá tantos parámetros como componentes en el vector de valores. En el caso de los problemas cuyas variables de decisión están acotadas se debe cuidar que el nuevo individuo no quede fuera del espacio de búsqueda.

Mutación polinomial

La mutación polinomial aplica una distribución polinomial para provocar la alteración en el individuo, en lugar de la distribución Gaussiana

$$y_i = x_i + \delta_i(x_i^{max} - x_i^{min})$$

donde el parámetro δ_i, se calcula a partir de la distribución de probabilidad polinomial $P(\delta) = 0.5(\eta_m + 1)(1 - |\delta|)^{\eta_m}$ como sigue

$$\delta_i = \begin{cases} (2r_i)^{1/(\eta_m+1)} - 1, & si \quad r_i < 0.5 \\ 1 - [2(1 - r_i)]^{1/(\eta_m+1)}, & si \quad r_i \geq 0.5 \end{cases}$$

La distribución polinomial es similar a la distribución de la mutación no uniforme, a excepción del parámetro η_m definido por el usuario, que regula el grado de apertura del área bajo la curva a partir del padre, ver Figura 32(b). Otra diferencia es que con la mutación polinomial la distribución no cambia dinámicamente con las generaciones, así permanece fija.

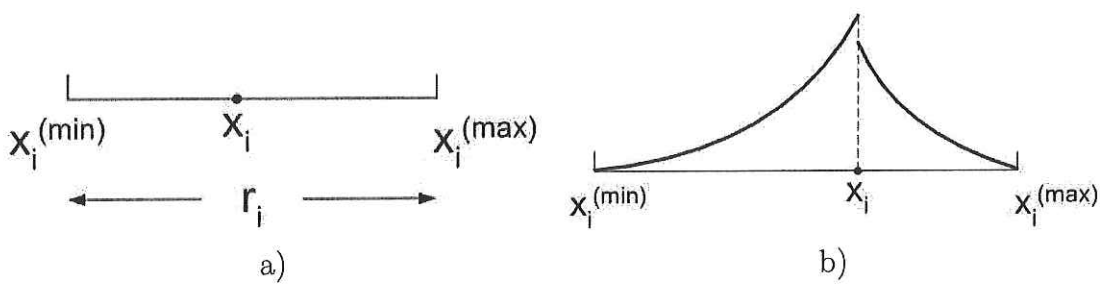


Figura 32: Algunos operadores de mutación. (a) Mutación aleatoria. (b) Mutación no uniforme.

Cruzamiento

El operador de cruzamiento o *recombinación* (*crossover* en inglés), realiza un intercambio de información entre dos o más individuos, dando como resultado uno o dos hijos

dependiendo del tipo de cruzamiento que se realice. La idea central es que diferentes segmentos de información provenientes de padres con un buen grado de adaptación, deberían producir hijos que conserven las similitudes entre los padres y aprovecharlas para realizar una búsqueda minuciosa en los espacios entre ellos. Por esta razón se dice que el cruzamiento es un operador de *explotación*, ya que procura agotar la búsqueda en regiones con mayor valor de aptitud en generaciones sucesivas de selección y cruzamiento. Un buen operador de cruzamiento debe considerar los siguientes aspectos para obtener un buen rendimiento.

1. **Imparcialidad**: cada individuo que tome parte en el cruzamiento debe contribuir en la misma proporción.

2. **Especialización**: un operador de cruzamiento debe diseñarse específicamente de acuerdo al problema que se esté resolviendo.

A continuación se presentan algunos operadores de cruzamiento.

Cruzamiento binario

Se utiliza cuando un individuo se representa por una cadena binaria. Necesita de dos padres y consiste en elegir un punto en la cadena, y a partir de él intercambiar esa porción entre las dos cadenas y así crear dos nuevo individuos (ver Figura 33).

Cruzamiento lineal

También se conoce *baricéntrico*. Consiste en generar una combinación lineal de dos padres para generar un hijo:

$$y_i = (1 - \alpha)x_i^1 + x_i^2$$

donde α es una variable aleatoria en el intervalo de [0, 1] que sigue una distribución uniforme.

Cruzamiento binario simulado

El cruzamiento SBX (por sus siglas en inglés de *Simulated Binary Crossover*) lo propone Deb (2001). Trabaja con dos padres y genera dos hijos, simulando el principio de cruzamiento binario de la siguiente manera:

$$\beta_i = \left| \frac{y_i^2 - y_i^1}{x_i^2 - x_i^1} \right|$$

donde β_i es un factor de dispersión que se define como la razón de la diferencia absoluta de los hijos entre los padres.

Primero se crea un número aleatorio u_i entre 0 y 1. Después, a partir de una función de distribución de probabilidad específica, la ordenada de β_{qi} se encuentra de tal manera que el área bajo la curva con probabilidad entre 0 y B_{qi} es igual al número aleatorio u_i. La función de distribución de probabilidad debe tener el mismo poder de búsqueda que el cruzamiento binario, y es la siguiente

$$P(\beta_i) = \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c} & \text{si } \beta_i \leq 1 \\ 0.5(\eta_c + 1)\dfrac{1}{\beta_i^{\eta_c+2}} & \text{de otra manera} \end{cases}$$

donde η_c es cualquier número no negativo. Un valor grande resulta en una probabilidad alta de que los hijos queden cerca de los padres, mientras que un valor pequeño permite obtener individuos lejanos de los padres, ver Figura 33. Usando esta expresión el valor de β_{qi} puede calcularse en función del área bajo la curva igual a u_i

$$\beta_{qi} = \begin{cases} 2u_i^{1/(\eta_c+1)} & \text{si } u_i < 0.5 \\ \frac{1}{2}(1 - u_i)^{1/(\eta_c+1)} & \text{si } u_i \geq 0.5 \end{cases}$$

Después de obtener la distribución de probabilidad β_{qi}, se procede a calcular los hijos como sigue

$$y_i^1 = 0.5[(1 + \beta_{qi})x_i^1 + (1 - \beta_{qi})x_i^2]$$

$$y_i^2 = 0.5[(1 - \beta_{qi})x_i^1 + (1 + \beta_{qi})x_i^2]$$

Entre mayor es η_c, el área bajo la curva es más estrecha, lo que provoca que el hijo quede cerca del padre. El cruzamiento SBX tiene dos propiedades esenciales:

1. La diferencia entre los hijos está en proporción a los padres.

2. Los hijos cercanos a los padres son monotónicamente más probables de escoger que aquellos que son lejanos.

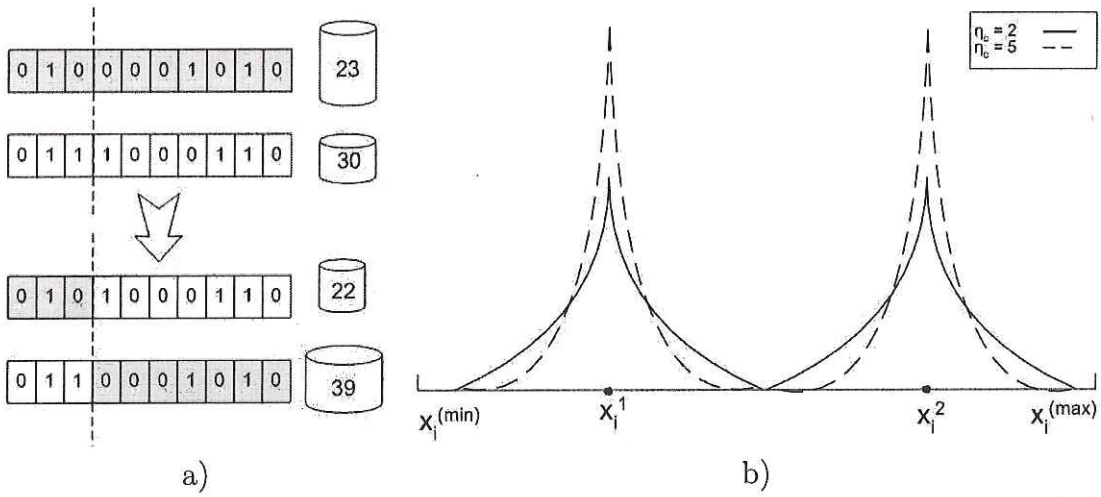


Figura 33: Algunos operadores de cruzamiento. (a) Cruzamiento binario. (b) Cruzamiento SBX.

Operadores de nicho: Repartición

Por sus características, un algoritmo evolutivo explora el espacio de búsqueda hasta converger en una región del espacio. Cuando un problema no requiere encontrar una solución global total, sino varias soluciones locales, se le llama objetivo *multimodal.* Una solución multimodal implica encontrar varias soluciones óptimas. Para evitar la convergencia es necesario introducir un mecanismo para mantener la diversidad de la población durante todo el proceso.

En la naturaleza, los recursos como la comida y el agua presentes en un ecosistema son variados pero limitados y deben compartirse por todas las especies vivas en este ecosistema. Si hay abundancia de recursos las especies prosperan y se multiplican; si hay escasez de estos recursos el número de individuos de las especies disminuye y sólo los más aptos sobreviven. El ecosistema busca por si mismo el equilibrio y no una lucha competitiva, inútil y desgastadora; y lo hace mediante *nichos ecológicos*. El nicho es el papel que cumple cada organismo dentro del ecosistema y depende de sus caracteísticas, su grado de adaptibilidad y su conducta. El nicho ecológico se compone del conjunto de los factores físicos, químicos y fisiológicos que necesita un organismo para vivir.

Inspirado en el principio de nichos ecológicos y en el equilibrio que intenta mantener la naturaleza, Holland (1975) propone el principio de nichos en los algoritmos evolutivos. Aqui los nichos representan una región donde existe una solución óptima, alrededor de la cual se reunen un conjunto de individuos, ver Figura 34. Estos individuos, al compartir el mimo nicho, comparten un recurso común: el valor de aptitud. Entre más individuos conformen un nicho, la cantidad del recurso asignado a cada uno es más pequeña. Este mecanismo se aplica generación tras generación, hasta crear un equilibrio donde el tamaño de la población en un nicho y el número de nichos son proporcionales a la disponibilidad del recurso. El objetivo de los operadores de nicho es que la población se distribuya sobre los mejores lugares dentro del espacio de búsqueda. El número de individuos en un nicho debe ser proporcional a los recursos (aptitud) de ésta. El principio de nichos ecológicos se aplica de maneras diferentes en los algoritmos evolutivos, siendo uno de ellos el operador de repartición (en inglés *sharing*) que propone Goldberg (1989).

En el operador de repartición, el valor de aptitud de un individuo baja de acuerdo al número de individuos dentro de su *zona de influencia* (nicho), con los que debe

compartir recursos. Así el valor de aptitud compartido de un individuo está dado por:

$$apt'_i = \frac{apt_i}{m_i}$$

donde apt_i es el valor de aptitud original y m_i el contador de nicho correspondiente. El contador de nicho representa el número de individuos que comparten el nicho con el individuo i y se calcula mediante la sumatoria de una función de repartición (Sh) del individuo i con respecto a los demás individuos:

$$m_i = \sum_{j=1}^{n} Sh(d_{i,j})$$

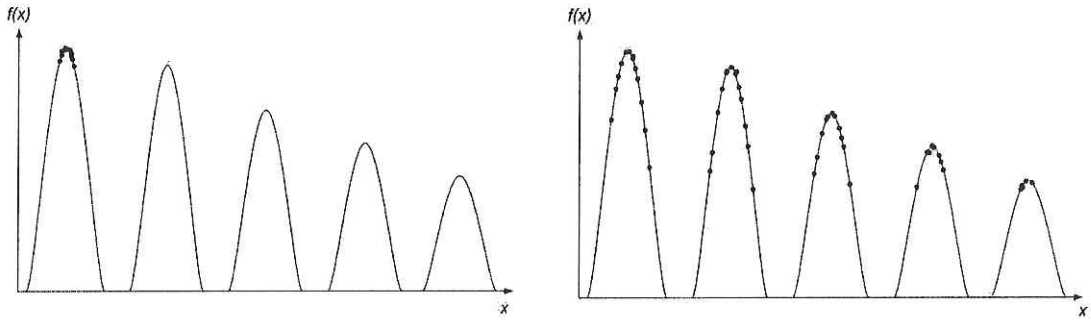
La función de repartición es una métrica de la distancia d entre dos individuos i y j. La función de repartición más utilizada tiene la siguiente forma

$$Sh(d_{i,j}) = \begin{cases} 1 - \frac{d_{i,j}}{\sigma_{rep}} & \text{si } d_{i,j} \leq \sigma_{rep} \\ 0 & \text{de otra manera} \end{cases}$$

donde σ_{rep} representa el radio de la zona de influencia de cada individuo. Por lo general se utiliza la distancia Euclidiana para medir la distancia entre individuos

$$d_{i,j} = \sqrt{\sum_{k=1}^{q} (x_{ik} - x_{jk})^2}$$

donde q representa el número de componentes del vector de variables que conforma cada individuo. Así, cuando muchos individuos se encuentran muy cercanos unos con otros, todos contribuyen a la disminución de sus respectivos valores de aptitud. Como resultado, queda limitada la proliferación de individuos en esa zona del espacio de búsqueda, dando oportunidad a la población emergente de poblar otros nichos. Después de un determinado número de generaciones se habrán poblado todos los nichos y cada uno de éstos poseerá una cantidad estable de individuos.



a) b)

Figura 34: Una función objetivo multimodal. A cada pico se le considera como nicho. a) Resultado de un algoritmo evolutivo sin aplicar operador de repartición. b) Resultado de un algoritmo evolutivo aplicando operador de repartición.

V.4.5 Criterio de terminación

La finalización del algoritmo usualmente se basa en que un individuo alcance una aptitud específica o mantener corriendo el algoritmo por un número determinado de generaciones. El algoritmo evolutivo avanzará generación tras generación evolucionando la población hacia la o las mejores soluciones en el espacio de búsqueda, hasta que se cumpla un criterio de paro para finalizar el algoritmo.

En ocasiones, cuando no se sabe con certeza la región donde se encuentra la solución óptima, o bien cuando el espacio de búsqueda resulta difícil de modelar, es complicado definir un criterio de paro para el algoritmo. Además, la naturaleza estocástica de la población inicial y de la generación de la nueva población agrega un nivel de complejidad mayor. Generalmente, cuando cierto porcentaje de la población converge en un punto del espacio de búsqueda el algoritmo se detiene, aunque este criterio no es aplicable para problemas de objetivo multimodal. También es posible aplicar el criterio de paro cuando el mejor individuo de la población no cambia después de cierto número de generaciones. Otro criterio es definir de manera a *priori* el número de generaciones que el algoritmo llevará a cabo. Éste último es el que se utiliza con mayor frecuencia, ya que

se puede establecer un número grande de generaciones y realizar pruebas estadísticas analizando la convergencia de los valores de aptitud a través de las generaciones; y de esta manera estimar el número de generaciones apropiado y utilizarlo en el futuro.

V.5 Conclusiones

Los algoritmos evolutivos han tenido gran aceptación como algoritmos de búsqueda y optimización en los últimos años. La idea de encontrar soluciones imitando a la naturaleza y aplicar operadores evolutivos como la selección, mutación y cruzamiento han provocado un gran interés, ganándoles terreno a los métodos tradicionales. Esto se debe a las ventajas que ofrecen los algoritmos evolutivos, entre las cuales las más importantes son las siguientes.

1. Son simples y fáciles de implementar. Además son altamente robustos, es decir, que pueden aplicarse a diferentes problemas sin cambios significativos a la estructura del algoritmo.

2. No necesita gran conocimiento a *priori* sobre el problema, el conocimiento emerge a través de las generaciones. No es necesario para el usuario conocer a detalle las restricciones del problema, sean lineales o no lineales.

3. El principio de evolución natural permite una búsqueda global efectiva, basada en la selección de los mejores individuos de una población que cambia generación tras generación.

La diferencia entre los límites de los paradigmas de la Computación Evolutiva, han ido disminuyendo cada vez más con el tiempo. Investigadores de una área están adaptando técnicas y aproximaciones de las otras áreas. Por ejemplo, el concepto de autoadaptación ahora se puede aplicar en las cuatro áreas discutidas.

En el capítulo 7 se propone un algoritmo evolutivo para afrontar el problema de minimización de energía de contornos activos, tratándolo como un problema de optimización. Para tal propósito se ha decidido utilizar algoritmos genéticos, por las razones expuestas en este capítulo y por las ventajas que éstos tienen de no quedarse atrapado en un mínimo local.

Capítulo VI

Inicialización de Contornos Activos utilizando Puntos de Interés

En este capítulo se propone una nueva forma de inicializar snakes utilizando puntos de interés. El problema de inicialización de snakes consiste en que al principio se diseñaron como modelos interactivos (Kass *et al.*, 1988), de tal forma que un usuario inicializa la snake cerca del objeto de interés. Por ejemplo, un usuario puede establecer los puntos que definen la snake utilizando el ratón de la computadora mediante una interfaz gráfica. Por esta razón, en aplicaciones no interactivas, se debe inicializar el contorno lo suficientemente cerca de la estructura de interés para obtener un buen desempeño, lo que garantiza una mejor convergencia en las características de los objetos y una mayor rapidez.

Los métodos más comunes para inicializar snakes consisten principalmente en métodos estadísticos para determinar la posición del objeto de interés o definir alguna región de interés, y otros que inicializan la snake como un polígono o curva (círculo, cuadrados, etc.) pero que necesitan algún conocimiento *a priori* a cerca de la posición, tamaño o forma del objeto de interés. Por lo que la inicialización de snakes sigue siendo un problema abierto que no ha sido resuelto completamente.

Como se describe en la sección III.4.2 se define una energía externa de tal manera que toma valores mínimos en las características de interés de la imagen. En esa misma sección se analizan diversas energías de la imagen como bordes (ecuación 7), y la intensidad de la imagen (ecuación 6). Por lo tanto, como los puntos de interés son las características de más bajo nivel que se pueden utilizar en una imagen, una alternativa

es analizar el desempeño de una snake que sea atraída hacia puntos de interés. Por lo que en la siguiente sección primero se evalúan los puntos de interés como energía externa.

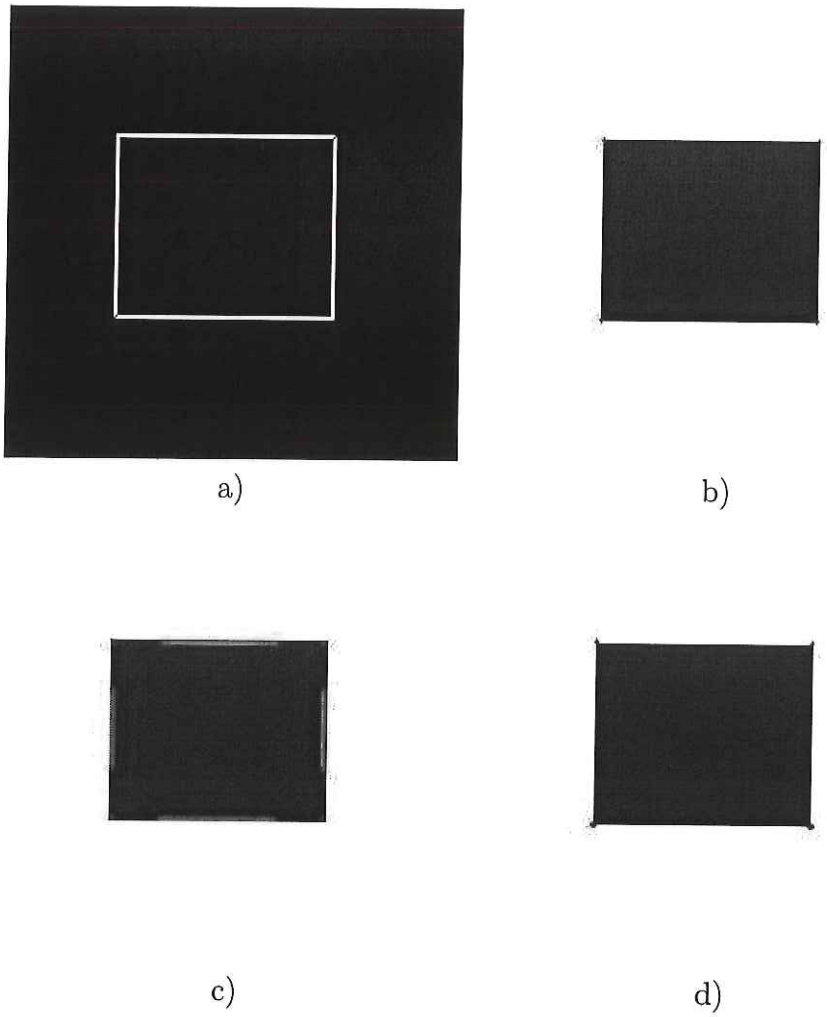
VI.1 Puntos de interés como energía externa

La idea de utilizar puntos de interés como energía externa surge del análisis de los diversos métodos de detección descritos en el capítulo IV. La mayoría de estos detectores tienden a establecer los puntos de interés en pixeles fuertes de los bordes de la imagen, ya que la variación de la señal en estos puntos es bastante alta con respecto a una medida local.

En la Figura 35 se muestra la comparación de los pixeles detectados como bordes de la imagen mediante el operador de Sobel (Figura 35(a)) y los pixeles detectados como puntos de interés. En la figura se utiliza una imagen sintética de dimensiones 256×256 con un cuadro negro en el centro de la imagen. El detector de puntos de interés de Harris (Figura 35(b)) obtuvo 4 puntos, que como se espera por las características de este detector se sitúan en las esquinas del cuadro. El detector evolucionado con programación genética IPGP1 (Figura 35(c)) obtuvo 270 puntos. Y el detector de Beaudet (Figura 35(d)) obtuvo 8 puntos, un par de puntos para cada una de las cuatro esquinas del cuadro.

Como se puede observar en la Figura 35 los puntos de interés detectados por los métodos de Harris, IPGP1 y Beaudet; se sitúan en los bordes de la imagen. Es importante recordar que los métodos de detección de puntos de interés dependen de un umbral, por lo que el número de puntos detectados se puede aumentar incrementando el rango del umbral.

Existen dos maneras de utilizar los puntos de interés como energía externa. La primera consiste en utilizar cualquiera de los detectores, Ecuaciones (16) a (23) sin una



a) b)

c) d)

Figura 35: a) Magnitud del gradiente $|G_x| + |G_y|$. b) Detector de Harris. c) Detector IPGP1. d) Detector Beaudet.

umbralización. Sin embargo, esto prodría hacer que una snake sea atraída hacia puntos de interés falsos.

Y la segunda alternativa consiste en utilizar la imagen resultante de algún detector de puntos de interés con únicamente los puntos detectados, y esta imagen tratarla como la imagen original. Es decir, la energía de la imagen podría estar dada por cualquiera de las ecuaciones 6, 7, 8 o cualquier otro tipo de energía externa simple (como GVF) o compuesta (suma ponderada de varias energías externas).

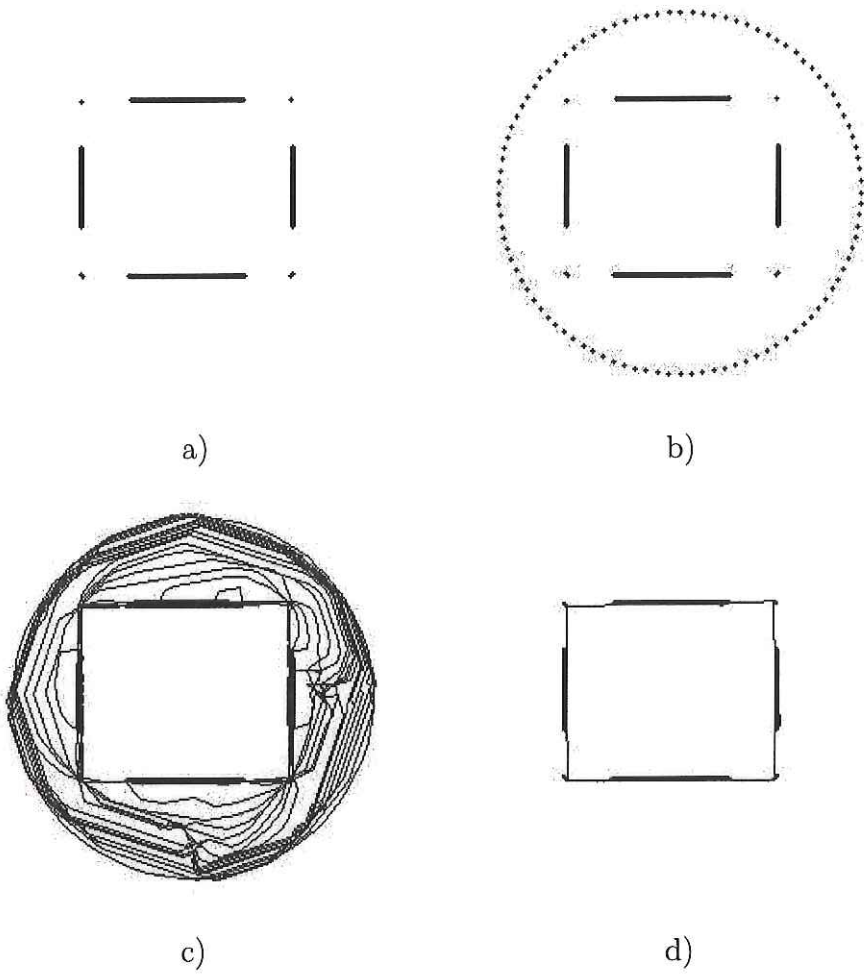


Figura 36: Snake atraída hacia puntos de interés. a) Puntos de interés detectados por IPGP1. b) Inicialización. c) Deformación. d) Contorno final.

La Figura 36 muestra un ejemplo de la snake atraída a los puntos de interés de la imagen del cuadro negro. En este caso se utiliza el detector IPGP1, que como se observa en la Figura 35 fue el que mayor número de puntos obtuvo (270 puntos). La Figura 36(a) muestra la imagen resultante de los puntos detectados por el operador IPGP1. En la Figura 36(b) se observa la inicialización de una snake como un círculo formado por 100 puntos alrededor del objeto. En este caso el círculo tiene un 80% del tamaño de la imagen completa que es de 256×256. En la Figura 36(c) se ilustran las

iteraciones de esta snake mientras minimiza su energía. Y en la Figura 36(d) se muestra la convergencia final de la snake en los puntos de interés. La forma del contorno de la snake entre los puntos de interés está determinado completamente por el término de suavidad de la spline (energía interna).

Los puntos de interés forman una ilusión de la forma de los bordes del objeto original (ver Figura 36(a)), lo que se conoce como *contornos subjetivos* (Kass *et al.*, 1988). Esta es una característica importante del modelo de la snake, ya que la misma snake que puede ser utilizada para encontrar contornos subjetivos de manera muy efectiva también puede encontrar los bordes de una imagen.

En la Figura 37 se ilustra otro ejemplo de una snake atraída hacia puntos de interés, pero en este caso se utiliza el operador de Harris. Como se analiza en la Figura 37(a) el detector de Harris sólo es capaz de obtener 4 puntos ubicados en las esquinas del cuadro negro. En la Figura 37(b) la snake se inicializa de la misma manera que en la Figura 36(b). La Figura 37(c) ilustra la deformación de la snake al minimizar su energía. Y en la Figura 37(d) se muestra el contorno final de la snake sobre los puntos de interés.

Como se observa en la Figura 37(d) la convergencia de la snake con 4 puntos de interés, representa de mejor manera los contornos reales del objeto en comparación con el que se logra en la Figura 36(d) con 270 puntos. Esto se debe a que la imagen del cuadro tiene una forma muy simple, ya que la unión de los cuatro vértices del cuadro con una línea es suficiente para describir su forma. Sin embargo, en figuras más complejas un número pequeño de puntos de interés podría significar la pérdida de información de la forma general del objeto.

Hasta ahora se ha utilizado una imagen sintética simple de un cuadro negro para realizar los experimentos, lo que significa que tanto la snake como los detectores de puntos de interés no se exponen a ningun tipo de ruido de la imagen. Por lo que el siguiente paso consiste en analizar el desempeño de una snake atraída hacia los puntos

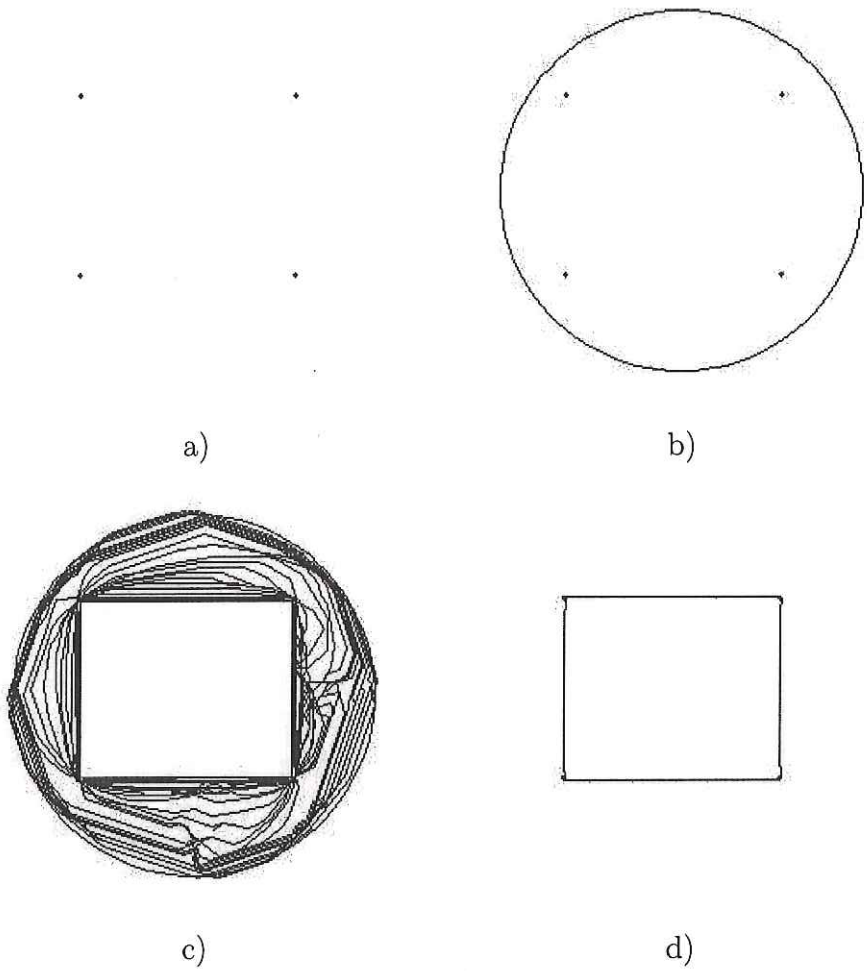


Figura 37: Snake atraída hacia puntos de interés. a) Puntos de interés detectados por Harris. b) Inicialización. c) Deformación. d) Contorno final.

de interés con una imagen adquirida por la cámara web Logitech (ver Apéndice B).

En la Figura 38(a) se muestra una imagen real de un cuadro negro en una hoja blanca para fines de comparación con la imagen sintética. Como se aprecia en esta figura la imagen carece de una iluminación uniforme. La Figura 38(b) muestra los puntos de interés detectados. En este caso el detector de Harris es lo suficiente robusto para detectar de nuevo solamente los 4 puntos de las esquinas del cuadro; sin embargo, algunos detectores podrían variar en el número de puntos que detectan (por ejemplo,

el detector IPGP1 detecta 41 puntos en la imagen real en lugar de los 270 de la imagen sintética). La Figura 38(c) inicializa 100 puntos de la snake de la misma manera que en los ejemplos anteriores. Y la Figura 38(d) muestra el contorno final de la snake.

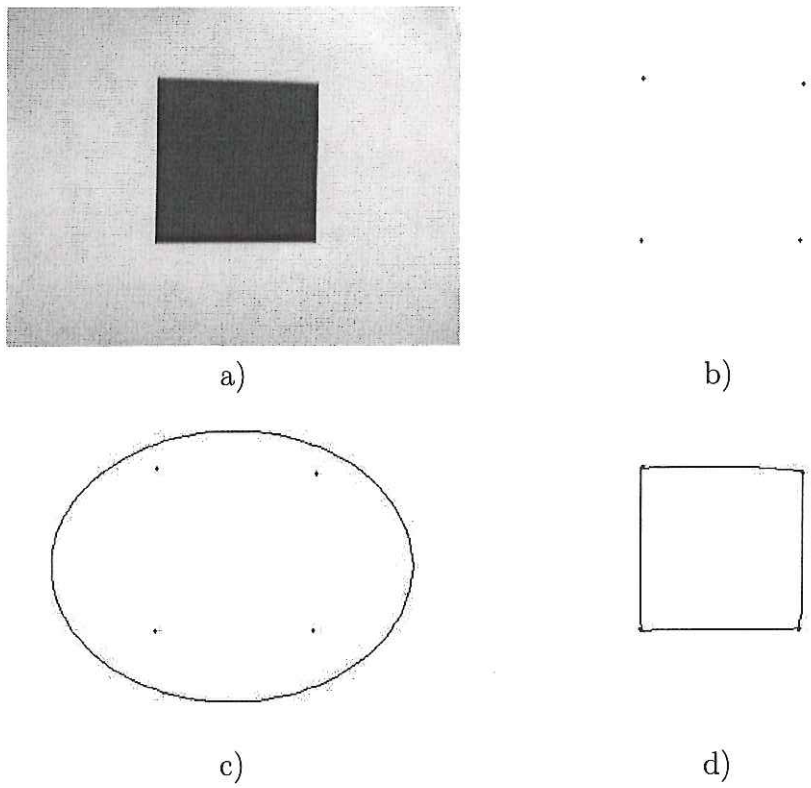


Figura 38: Snake atraída hacia puntos de interés. a) Imagen original. b) Puntos de interés detectados por Harris. c) Inicialización. d) Contorno final.

El uso de los puntos de interés como energía externa de una snake representa una alternativa interesante para encontrar los contornos reales de un objeto simple, utilizando los puntos como contornos subjetivos. Sin embargo, en un objeto más complejo la pérdida de información de los contornos reales entre los puntos de interés puede ser crítica. Por esta razón, una snake atraída hacia puntos de interés podría no describir de forma exacta los contornos reales de un objeto. Además en objetos con textura los

puntos de interés pueden situarse incluso dentro del objeto.

Una solución a este problema podría ser la utilización de una energía externa compuesta por la suma ponderada de una energía para detectar bordes (ecuación 7) y una energía de puntos de interés. De esta manera, la snake será atraída tanto a bordes como puntos de interés.

Sin embargo, una características interesante de los detectores de puntos de interés analizados en el Capítulo IV es que por lo general se sitúan en los objetos de mayor relevancia en las imágenes, ver Figuras 26 a 31. Por lo que intuitivamente se podría buscar algún método para poder encerrar los puntos correspondientes a cada objeto. De esta forma en las siguientes secciones se describe una manera de aprovechar esta propiedad de los puntos de interés para poder encerrar estos puntos e inicializar una snake mucho más cerca del objeto de interés.

VI.2 Casco convexo

Descomponer un contorno en segmentos es muy útil, ya que la descomposición reduce la complejidad del contorno y por lo tanto simplifica el proceso de descripción. Esta aproximación es particularmente atractiva cuando el contorno contiene una o más concavidades que contienen información de la forma. En este caso la utilización del casco convexo de una región encerrada por el contorno es una herramienta eficiente para una descomposición robusta del contorno.

La frontera del polígono convexo que encierra un conjunto de puntos se conoce como *casco convexo* (en inglés convex hull). Una manera de imaginar la forma de un casco convexo consiste en considerar una banda de hule (liga) que se estira alrededor de las posiciones del conjunto de puntos de modo que cada uno de éstos se encuentra ya sea en el perímetro del casco o dentro de éste, ver Figura 39.

Uno de los algoritmos más utilizados para calcular el cierre Q o casco convexo es el

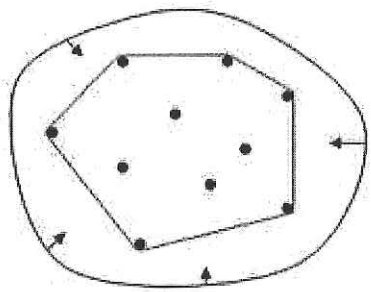


Figura 39: Casco convexo, analogía con una banda elástica.

Algoritmo Sklansky's. Sea $P = p_1, ..., p_n$ un polígono simple de n vértices. El algoritmo procesa secuencialmente cada uno de los vértices, construyendo incrementalmente el cierre Q. Los pasos que sigue el algoritmo son:

1. Se parte de un polígono $P = p_1, ..., p_n, p_1$. Como puede apreciarse el vértice p_1 se encuentra tanto al inicio como al final del polígono.

2. Se insertan en el cierre Q los dos primeros vértices del polígono.

3. Mientras queden vértices en P, se trata cada vértice de la siguiente manera: se borra la cima de la pila, hasta que el vértice que está por debajo de la cima y el vértice no sean un vértice a derecha, entonces se inserta el vértice en la pila.

VI.3 Mínimo círculo contenedor

Trata del problema de encontrar el círculo circunscrito de área mínima para un conjunto de puntos bidimensionales predefinidos.

Puesto que un círculo se puede definir por dos o tres puntos, la solución más obvia (por fuerza bruta), consiste en examinar cada posible pareja y trío de puntos del conjunto, y para cada uno de ellos comprobar que todos los restantes están dentro de la circunferencia que delimitan. Por lo tanto, la complejidad de este algoritmo es $O(n^4)$.

Una mejora al método anterior fue propuesta por (Elzinga y Hearn, 1972), con una complejidad $O(n^2)$.

Preparata y Shamos (1985) propusieron un método haciendo uso de diagramas de Voronoi de los puntos más lejanos con una complejidad de $O(nlog(n))$.

VI.3.1 Experimentación

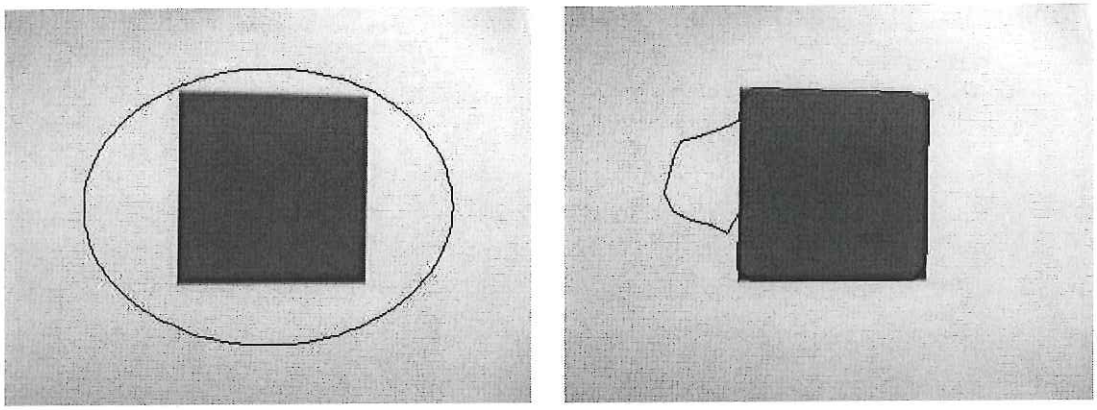
Para fines de comparación primero se analiza un método común para inicializar snakes y después se analiza la inicialización de una snake utilizando el mínimo círculo contenedor.

En la Figura 40 se ilustra una de las maneras más comunes de inicializar un snake, que consiste simplemente en utilizar información *a priori* a cerca de la posición y tamaño del objeto en la imagen. De este modo, el objeto de interés se considera que se encuentra cerca del centro de la imagen y que tiene un tamaño relativo menor al 80% de la imagen. Es decir el objeto se encuentra a una distancia prudente de la cámara, lo suficiente para ser captado completamente dentro de la imagen. Así es posible inicializar una snake en forma de círculo (en caso que la imagen sea cuadrada) de tal manera que el objeto quede completamente dentro.

En la Figura 40(a) la snake se inicializa como una elipse, ya que la resolución de la imagen original en este caso es de 320×240. La elipse tiene un tamaño relativo del 70% de la imagen. La imagen original que se utiliza en este ejemplo es la misma que la que se muestra en la Figura 38(a). Nótese que la snake se inicializa más cerca que el ejemplo de la Figura 38(c) donde se utiliza una elipse con un tamaño del 80% de la imagen, esto con el fin de asegurar un mejor desempeño.

Después de 60 iteraciones la snake se queda atrapada en un mínimo local y es incapaz de salir de ese mínimo en las iteraciones sucesivas. La convergencia final de la snake se muestra en la Figura 40(b).

En la Figura 41 se ilustra un ejemplo de una snake inicializada utilizando los puntos



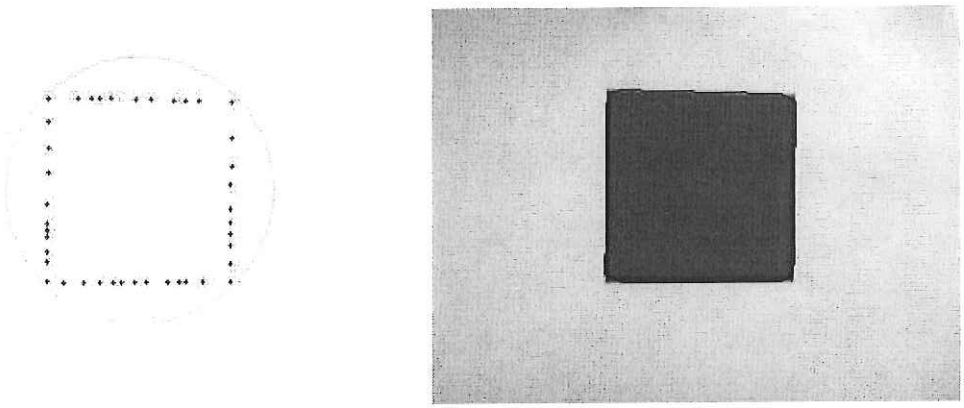
a) b)

Figura 40: a) Inicialización de una snake utilizando información *a priori*. b) Contorno final.

de interés y el mínimo círculo contenedor. La Figura 41(a) muestra los puntos de interés detectados por el operador IPGP1 y el círculo de mínimo tamaño que contiene a estos puntos, con lo cual se inicializa la snake. Nótese lo cerca que la snake se inicializa del objeto en comparación a la inicialización de la Figura 40(a). También se puede observar que por lo general no importa el método de detección de puntos de interés que se utilice o el número de puntos detectados, ya que el mínimo círculo contenedor tendrá en su interior a todos los puntos y por consecuencia al objeto. Por ejemplo, si se utlizara el operador de Harris, éste detectaría básicamente 4 puntos ubicados en las esquinas del cuadro; sin embargo el mínimo círculo contenedor seguiría siendo básicamente el mismo.

La Figura 41(b) muestra la convergencia de la snake depués de sólo 17 iteraciones. En este caso la snake no se queda atrapada en un mínimo local ya que la cercanía de la inicialización al objeto asegura un mejor desempeño. Además de que la snake tiene una mejor convergencia, la rapidez con que se logra también es muy significativa.

Pero los puntos de interés no se limitan a encontrar un solo objeto en una imagen,



a) b)

Figura 41: a) Inicialización de una snake utilizando el método de mínimo círculo contenedor de los puntos de interés. b) Contorno final.

sino que por lo general se ubican en los objetos de mayor relevancia de la imagen. Entonces, en una imagen que contiene más de un objeto, los puntos de interés detectados se podrían clasificar como pertenecientes a cada uno de los objetos. Lo que permitiría inicializar una snake diferente para cada objeto. Esto da lugar al problema de agrupamiento de una serie de puntos predefinidos, para el cual muchos métodos se han propuesto. En la siguiente sección se describe uno de estos métodos, el algoritmo de agrupamiento k-medias, y se analiza su utilidad en la clasificación de puntos de interés.

VI.4 Agrupamiento k-medias

MacQueen (1967) propuso el algoritmo llamado k-medias (en inglés k-means), que es uno de los algoritmos más simples de aprendizaje no supervisados para resolver el problema de agrupamiento.

1. El algoritmo clasifica un conjunto de datos en un número determinado de grupos (k grupos o clusters en inglés), establecidos de manera *a priori*.

2. La idea principal es definir k centroides, uno para cada grupo. Los cuales deben ser

establecidos de manera astuta, ya que diferentes localizaciones causan resultados diferentes. Por lo que la mejor opción es establecerlos lo más separados posible entre ellos.

3. El siguiente paso es tomar cada punto perteneciente al conjunto de datos y asociarlo con el centroide más cercano.

4. Cuando todos los puntos hayan sido asignados, se recalculan las posiciones de los k centroides.

5. Después se repiten los dos pasos anteriores hasta que los centroides ya no se muevan.

Finalmente, el algoritmo minimiza una función objetivo, en este caso la función de error cuadrático.

$$J = \sum_{i=1}^{k} \sum_{j \in S_i} |x_j - \mu_i|^2$$

donde existen k grupos S_i, $i = 1, 2, \ldots, k$ y μ_i es el punto medio o centroide de todos los puntos $x_j \in S_i$.

El algoritmo comienza particionando los puntos de entrada en k conjuntos iniciales, que puede ser de manera aleatoria o utilizando alguna heurística. Entonces se calcula el punto medio o centroide para cada conjunto. Se construye una nueva partición asociando cada punto con el centroide más cercano. Después los centroides son recalculados para los nuevos grupos. Y el algoritmo se repite por la aplicación de los dos pasos anteriores hasta lograr la convergencia que se obtiene cuando los puntos ya no cambian de grupo o cuando los centroides ya no cambian de posición.

El algoritmo es bastante popular ya que converge bastante rápido en la práctica. Se ha observado que el número de iteraciones usualmente es mucho menor que el número de puntos. Sin embargo, recientemente Arthur y Vassilvitskii han mostrado que existen

ciertos conjuntos de puntos para los cuales el algoritmo k-medias toma un tiempo superpolinomial para converger $2^{\Omega(\sqrt{n})}$.

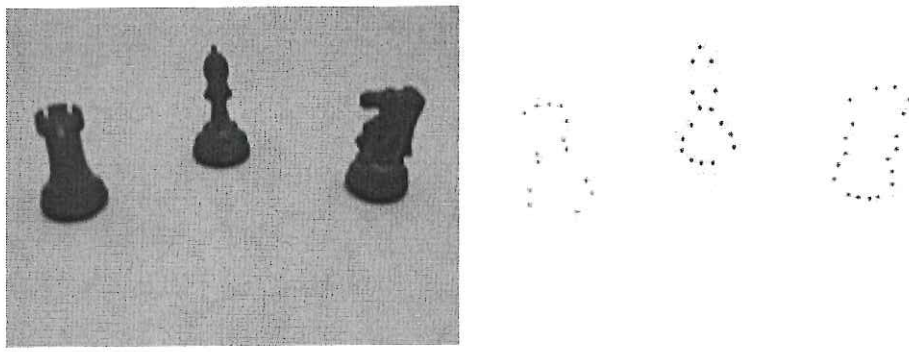
En términos de desempeño el algoritmo no garantiza regresar un óptimo global. La calidad de la solución final depende en gran medida del conjunto de grupos inicial. Sin embargo, ya que el algoritmo es muy rápido, un método común es correr el algoritmo varias veces y regresar el mejor agrupamiento encontrado.

Otra desventaja del algoritmo es que se debe establecer el número de grupos k que se va a buscar de manera *a priori*. Si existen más o menos grupos que los predefinidos, se pueden obtener resultados no deseados. Además, el algoritmo sólo funciona cuando grupos esféricos se encuentran naturalmente disponibles en los datos.

VI.4.1 Experimentación

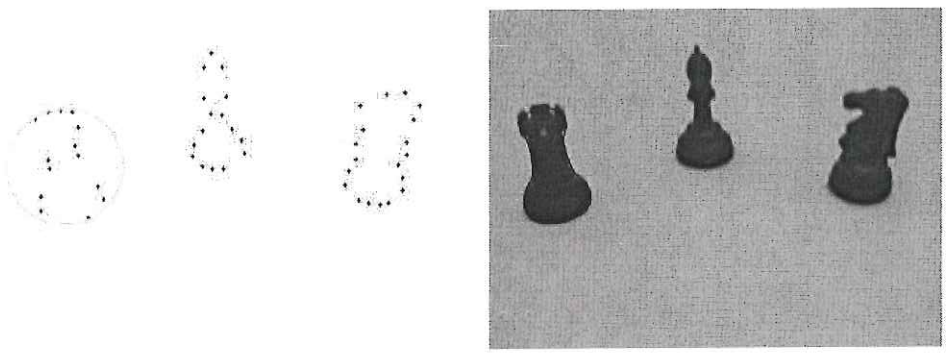
Para ilustrar el uso del algoritmo k-medias se utiliza la imagen de la Figura 42(a) que contiene tres piezas de ajedrez sobre una superficie. En este caso se utiliza el detector IPGP1 para determinar los puntos de interés en la imagen, con el cual se detectaron 49 puntos. Ahora, se conoce que la imagen está compuesta por tres objetos, por lo tanto el parámetro k del algoritmo k-medias debe establecerse igual a 3. Con esto, en la Figura 42(a) se obtiene la clasificación de los puntos de interés en tres grupos, que es lo que se espera del algoritmo. El primer grupo contiene 14 puntos, el segundo 16 puntos, y el tercer grupo contiene 19 puntos.

Después de clasificar los puntos de interés, se necesita calcular el mínimo círculo que contiene los puntos de cada grupo. De esta manera se puede inicializar una snake diferente para cada uno de los grupos, y por lo tanto para cada objeto en la imagen. En la Figura 43(a) se muestra la inicialización de una snake para el primer objeto. Después de sólo 7 iteraciones la snake converge sobre el contorno del objeto de manera muy precisa Figura 43(b).



a) b)

Figura 42: a) Imagen original. b) Agrupamiento de los puntos de interés utilizando k-medias ($k = 3$).

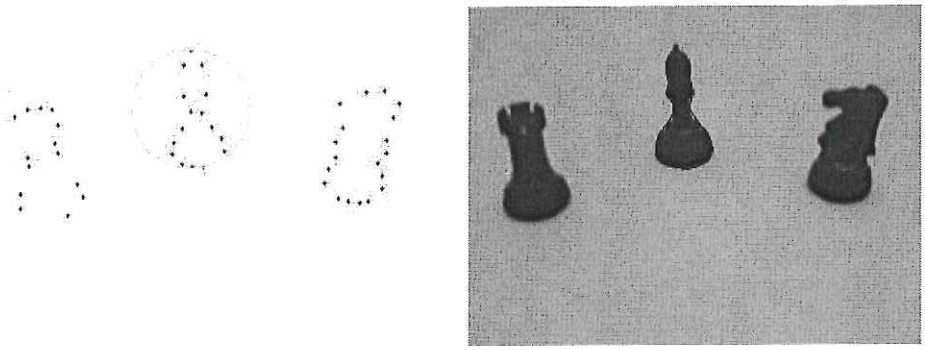


a) b)

Figura 43: a) Inicialización del mínimo círculo contenedor en la primera pieza. b) Contorno final.

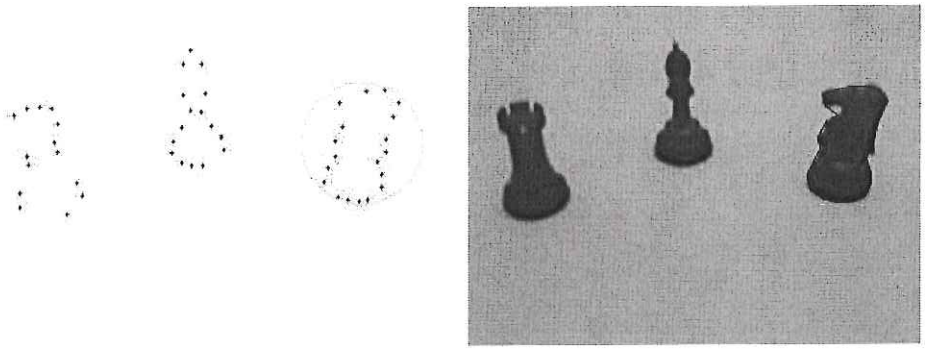
La Figura 44(a) ilustra la inicialización de una snake para la segunda pieza de ajedrez. En este caso, por las características del objeto, la snake trabaja un poco más para encontrar su equilibrio por lo que le toma 27 iteraciones (Figura 44(b)).

Por último, en la Figura 45(a) se observa la inicialización de una snake para la tercera pieza de ajedrez (caballo). Después de sólo 6 iteraciones la snake prácticamente a segmentado el objeto con exactitud (Figura 45(b)).



a) b)

Figura 44: a) Inicialización del mínimo círculo contenedor en la segunda pieza. b) Contorno final.



a) b)

Figura 45: a) Inicialización del mínimo círculo contenedor en la tercera pieza. b) Contorno final.

VI.5 Conclusiones

En este capítulo se ha propuesto una nueva forma de inicialización automática de snakes utilizando puntos de interés y algún método para encerrar los puntos con una figura geométrica de mínimo tamaño, como el mínimo círculo contenedor o el casco convexo. Con lo que se asegura que la snake se inicialice lo bastante cerca del objeto de interés, obteniendo un mejor desempeño de la snake y una mayor velocidad de convergencia, ya que se necesitan menos iteraciones para que la snake encuentre su equilibrio. Además,

los puntos de interés permiten la detección de muchos objetos en una imagen. De tal manera que clasificando los puntos detectados con algún método de agrupamiento, como la técnica de k-medias, se permite inicializar una snake diferente para cada objeto en una imagen.

Capítulo VII

Algoritmos Genéticos para la minimización de Snakes

En este capítulo se propone un procedimiento de minimización de energía de la snake basado en Algoritmos Genéticos (GA). Los modelos de contornos activos es un método efectivo para detectar los bordes de los objetos en una imagen. Sin embargo, existe un número de problemas asociado con esta aproximación como la inicialización y la existencia de mínimos locales.

Para superar estas limitaciones se propone un GA, ya que ofrecen un procedimiento de búsqueda global que han demostrado ser robustos en muchas aplicaciones, y que no se limitan a restricciones como el uso de derivadas en la función objetivo.

VII.1 Formulación

La aplicación de un Algoritmo Genético para cualquier problema práctico requiere la definición de los siguientes componentes:

1. Una representación (cromosoma) de las soluciones del problema.
2. Una evaluación (función objetivo) de los individuos en función de su aptitud.
3. Un método para inicializar la población de soluciones.
4. Los operadores genéticos (mutación, cruzamiento) que producirán nuevos conjuntos de individuos.

5. Los valores de los parámetros utilizados por el algoritmo (tamaño de la población, probabilidad de cruzamiento y mutación, etc.).

6. Un criterio de terminación del algoritmo.

Los últimos tres componentes son independientes del dominio del problema, mientras que los primeros tres componentes son totalmente dependientes del problema y se discuten a continuación.

Los parámetros que se modifican en la optimización genética son las posiciones de los puntos que definen la snake en el plano de la imagen $v(s) = (x(s), y(s))$. Las coordenadas x y y se codifican utilizando coordenadas polares para simplificar la implementación $v_i = (r_i, \theta_i)$, con el origen en el centro de la imagen. De hecho, este punto se debe encontrar dentro del objeto de interés. Las magnitudes r_i se codifican en cromosomas, mientras que los ángulos $\theta_i = 2\Pi_i/n$ donde n es el número de puntos de la snake.

La representación polar introduce un orden en los puntos de la snake y previene que los puntos no se crucen entre sí durante la evolución. De esta manera, los operadores genéticos (mutación y cruzamiento) se pueden implementar directamente con esta representación, y no se requiere una verificación para asegurar que la mutación y el cruzamiento generen individuos válidos.

La función de aptitud que se utiliza es el total de la energía de la snake que se define en la Ecuación (3), donde E_{int} y E_{ext} se definen en las Ecuaciones (4) y (8) respectivamente. La población inicial de las magnitudes r_i se inicializan de manera aleatoria.

VII.2 Experimentación

En los siguientes experimentos se utiliza una selección por torneo (ver sección V.4.3), un cruzamiento SBX ya que los cromosomas se codifican como números reales, una

mutación polinomial, y también un operador de nicho que permite restricciones en la selección por torneo (ver sección V.4.4). El criterio de paro está dado por el número de generaciones.

Determinar el conjunto de parámetros más adecuado tanto para la snake $(\alpha, \beta, \gamma, n)$ como para el algoritmo genético (pm, pc, número de generaciones, y tamaño de población) depende de diversos aspectos de las caracterśticas de la imagen: el ruido en la imagen, el tamaño, el tipo de objeto, etc. Por lo tanto, para una aplicación particular algún tipo de experimentación se requiere para encontrar los mejores parámetros.

Para los parámetros de la snake se observó en los experimentos que valores muy grandes de α tiene el mismo efecto de encogimiento de la snake al igual que con la minimización por otros métodos tradicionales.

El número de puntos de la snake es directamente proporcional a la precisión en la segmentación; es decir, entre mayor sea n mejor se capturarán los contornos de los objetos. Esto se debe a la utilización de coordenadas polares (r_i, θ_i) donde el ángulo θ depende de n $(\theta_i = 2\Pi_i/n)$. Por ejemplo, si $n = 10$ cada ángulo θ se construirá a incrementos de 36 grados $(\theta_1 = 0, \theta_2 = 36, \theta_3 = 72, \ldots, \theta_n = 360)$.

Para los parámetros del algoritmo genético se observó que con una probabilidad de cruzamiento alta y una probabilidad de mutación baja se obtenian mejores resultados. Esto se debe a que la mutación puede traer cambios muy bruscos en la posición de un punto de la snake, lo cual no se desea porque ocasiona que en el contorno final algunos puntos queden muy alejados de sus vecinos. También se observó que la precisión en la segmentación puede ser fácilmente mejorada si se incrementa el tamaño de la población o el número de generaciones. Sin embargo, se decidió mantener estos parámetros relativamente pequeños con el fin de que el algoritmo genético no tenga un gran costo computacional.

Después de estos experimentos se obtuvieron los siguientes valores de los parámetros

que demostraron tener un buen desempeño tanto para imágenes sintéticas como imágenes adquiridas por la cámara Logitech.

- Parámetros de la snake:
  - Número de puntos de la snake = 50
  - $\alpha = 0.5$, $\beta = 0.005$ y $\gamma = -1$
- Parámetros del algoritmo genético
  - Probabilidad de cruzamiento = 0.9
  - Probabilidad de mutación=0.001
  - Número de generaciones= 1000
  - Tamaño de la población=100

A continuación se describen algunos ejemplos de los resultados alcanzados por el algoritmo genético utilizando los valores de los parámetros anteriores. Para fines de comparación, el primer ejemplo se utiliza de nuevo la imagen sintética de un cuadro negro que tiene una resolución de 256 × 256 pixeles. En este caso se utiliza el origen de las coordenadas polares como el centro de la imagen; sin embargo como se observa en la Figura 46 el origen no coincide con el centro del objeto. A pesar de esto la snake genética realiza un buen trabajo segmentando el objeto, y es comparable al resultado obtenido por el algoritmo voraz en la Figura 24.

La siguiente prueba se realiza con una imagen adquirida con la cámara Logitech para evaluar el desempeño de la snake genética al ruido. Como se observa en la Figura 47 la snake genética segmenta de manera muy precisa el objeto y se puede decir que es mejor que el algoritmo voraz para escapar de mínimos locales (ver Figura 40).

Se puede observar en las Figuras 46 y 47 que el origen de las coordenadas polares (pixel de color rojo) se encuentra en el centro de la imagen pero no en el centro del

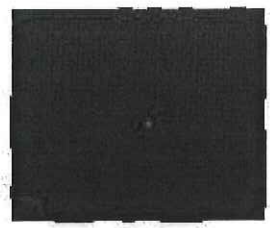


Figura 46: Resultado de una snake genética en una imagen sintética.

objeto. Esto es debido a que utilizando información *a priori* se considera que el objeto se encuentra cerca del centro de la imagen. Aunque esto es suficiente para que el algoritmo genético segmente el objeto de interés, se pueden obtener mejores resultados si el origen se encontrara en el centro del objeto. Además si el origen se encuentra fuera del objeto, el algoritmo genético sólo sería capaz de detectar algunas partes del objeto, obteniendo resultados extraños.

Por lo tanto, una alternativa a este problema es utilizar una inicialización por puntos de interés y el mínimo círculo contenedor (discutido en el capítulo anterior) para encontrar el centro del objeto. Lo cual consiste en aprovechar los datos arrojados por el mínimo círculo contenedor, su centro y la magnitud de su radio. Como se analiza en la sección VI.3.1 por lo general el centro del mínimo círculo contenedor coincide con el centro del objeto, ver Figura 41. De esta manera es posible establecer el origen de las coordenadas polares con el centro del círculo y restringir la evolución del parámetro r_i a la magnitud del radio del círculo, ya que el objeto se encontrará dentro del círculo. Con lo que se diminuirá notablemente el espacio de búsqueda y se obtendrá una mejor

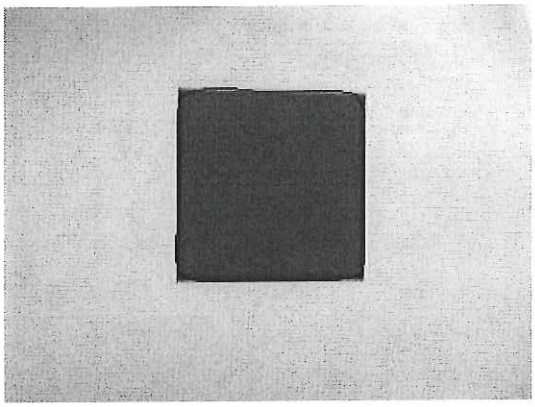


Figura 47: Resultado de una snake genética en una imágen real adquirida por una cámara web.

segmentación con la snake genética.

En las Figura 48 y 49 se ilustra una snake genética inicializada de esta manera. Nótese la posición del origen que en esta ocasión se encuentra prácticamente en el centro de los objetos. Se puede observar en la Figura 48 una notable mejoría en la segmentación en comparación con la Figura 46, donde existía una anomalía en la snake en la parte inferior izquierda del objeto.

Hasta ahora sólo se han utilizado los algoritmos genéticos en snakes para segmentar un solo objeto en una imagen. Sin embargo, segmentar varios objetos en una imagen implica un gran problema en la mayoría de los algoritmos para snakes. Una solución a este problema utilizando snakes genéticas es aplicar el procedimiento de agrupamiento de puntos de interés con la técnica de k-medias discutido al final del capítulo anterior.

En la Figura 50 se utiliza la imagen de las tres piezas de ajedrez para ilustrar lo anterior. Como se observa en la figura el objeto de interés (torre) no se encuentra ni siquiera cerca del centro de la imagen, por lo que una snake genética inicializada en el centro de la imagen sería incapaz de segmentar este objeto. Para inicializar la snake genética, primero se detectan los puntos de interés en la imagen los cuales tenderán a

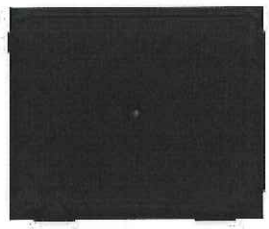


Figura 48: Resultado de una snake genética en una imagen sintética utilizando una inicialización por puntos de interés y el mínimo círculo contenedor.

encontrarse sobre los bordes de las tres piezas de ajedrez. Después de esto se utiliza la técnica de agrupamiento k-medias para clasificar los puntos de interés en tres grupos que perteneceran a cada una de las piezas, ver Figura 42(b). Al tener los puntos clasificados, se escoge uno de los grupos para obtener el mínimo círculo contenedor de los puntos pertenecientes al grupo, ver Figura 43(a). Y por último se obtiene el centro y el radio del círculo para inicializar la snake genética. Los últimos dos pasos se pueden repetir para segmentar de igual manera las dos piezas restantes.

Como se observa en la Figura 50 el centro del mínimo círculo contenedor coincide de manera muy precisa con el centro de la torre (objeto de interés). De esta manera, la snake genética realiza un trabajo muy efectivo para segmentar el objeto muy parecido al obtenido en la Figura 43(b).

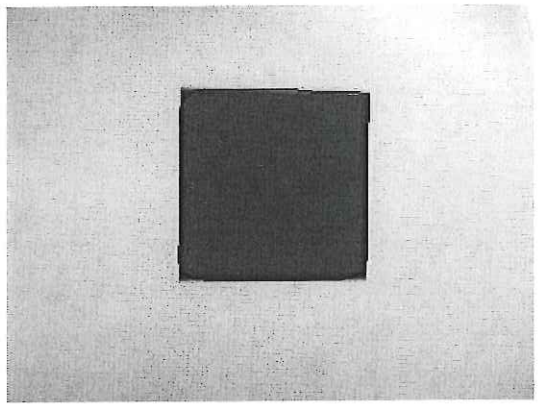


Figura 49: Resultado de una snake genética en una imagen real adquirida por una cámara web utilizando una inicialización por puntos de interés y el mínimo círculo contenedor.

VII.3 Conclusiones

En este capítulo se aplica la evolución por un algoritmo genético a las posiciones de la snake. Al igual que con otros algoritmos, la inicialización por puntos de interés para la snake genética demostró ser también muy útil. Aprovechando los datos arrojados por el mínimo círculo contenedor, como el centro del círculo y la magnitud del radio, se puede encontrar el centroide de los objetos en la imagen y establecer el rango de valores para la evolución de la magnitud de las coordenadas polares disminuyendo así el espacio de búsqueda. Además da la posibilidad de segmentar varios objetos en la imagen utilizando una técnica de agrupamiento para los puntos de interés.

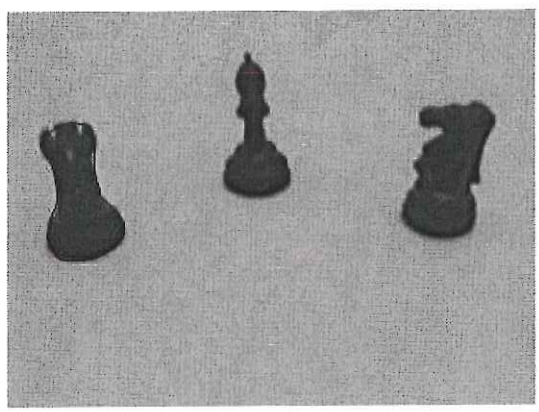


Figura 50: Resultado de una snake genética en una imagen real adquirida por una cámara web utilizando una inicialización por puntos de interés, la técnica de k-medias y el mínimo círculo contenedor.

Capítulo VIII

Conclusiones y trabajo futuro

Los modelos de contornos activos o snakes son un método efectivo para detectar los contornos de objetos en una imagen. Las snakes tienen muchas ventajas sobre otros métodos de segmentación, ya que no necesitan de un posprocesamiento, son más robustas y sus resultados son visualmente interpretables. Por lo tanto, una tarea de más alto nivel puede utilizar estos resultados fácilmente.

Sin embargo, las snakes tienen algunas limitaciones como la inicialización y la existencia de mínimos locales. En este trabajo se ha propuesto una nueva manera de inicializar los contornos activos utilizando puntos de interés, y un algoritmo genético para minimizar la energía de una snake.

El problema de inicialización de los contornos activos consiste en que las snakes tradicionales se utilizaban como modelos interactivos, en donde un usuario inicializa la snake cerca del objeto de interés. Sin embargo, para aplicaciones automatizadas es necesario buscar una manera de inicializar el contorno de manera automática.

La inicialización de snakes utilizando puntos de interés y algún método para encerrar los puntos con alguna figura geométrica de mínimo tamaño, como el mínimo círculo contenedor o el casco convexo, asegura que la snake se inicialice lo bastante cerca del objeto de interés. Con lo que se obtiene un mejor desempeño de la snake y una mayor velocidad de convergencia, ya que se necesitan menos iteraciones para que la snake encuentre su equilibrio. Además, los puntos de interés permiten la detección de muchos objetos en una imagen. De tal manera que clasificando los puntos detectados con algún método de agrupamiento, como la técnica de k-medias, se permite inicializar una snake

diferente para cada objeto en una imagen.

El trabajo futuro en este tipo de inicialización es la paralelización de los algoritmos. De tal forma que una vez clasificados los puntos de interés se permita inicializar y deformar una snake diferente para cada objeto al mismo tiempo. Con lo que se espera mejorar la velocidad en la segmentación de múltiples objetos y poder utilizarse en aplicaciones de tiempo real.

El problema de minimización de energía de la snake consiste en que la snake clásica utiliza cálculo de variaciones para iterativamente minimizar la energía. Algunos de los problemas con esta aproximación es la inicialización y la existencia de mínimos locales. Algunas soluciones se han propuesto para minimizar la energía de la snake como técnicas de templado simulado (en inglés simulated annealing) Strovik (1994), programación dinámica Amini *et al.* (1990), un algoritmo voraz Williams y Shah (1992), entre otros. Sin embargo, estos trabajos requieren búsquedas exhaustivas de las soluciones admisibles, tienen un control complicado de los parámetros que utilizan, o necesitan una inicialización bastante precisa.

Por lo que en este trabajo se propone un algoritmo genético para superar algunas limitaciones del modelo snake, ya que los GA han demostrado ser robustos en muchas aplicaciones. El procedimiento de minimización de la energía basado en algoritmos genéticos supera los problemas asociados con la sensibilidad a la inicialización y no quedarse atrapado en mínimos locales, los cuales son problemas cruciales en las técnicas clásicas.

En este trabajo se aplica la evolución por un algoritmo genético a las posiciones de la snake. Al igual que con otros algoritmos, la inicialización por puntos de interés para la snake genética demostró ser también muy útil. Aprovechando los datos arrojados por el mínimo círculo contenedor, como el centro del círculo y la magnitud del radio, se puede encontrar el centroide de los objetos en la imagen y establecer el rango de

valores para la evolución de la magnitud de las coordenadas polares disminuyendo así el espacio de búsqueda. Además da la posibilidad de segmentar varios objetos en la imagen.

En los experimentos el algoritmo genético demostró ser bastante rápido para segmentar imágenes adquiridas por la cámara Logitech, con un promedio de 3 segundos. Sin embargo, esto no es suficiente para poder ser utilizado en aplicaciones de tiempo real. Por lo que el trabajo futuro en este aspecto podría ser la búsqueda de alternativas para optimizar el costo computacional.

La determinación de los pesos para los diferentes términos de energía sigue siendo un problema abierto importante, y que generalmente se realiza de manera empírica. Por lo que el trabajo futuro de esta técnica podría ser la evolución de los parámetros que gobiernan el comportamiento de la snake.

Bibliografía

Amini, A. A., T. E. Weymouth, y R. Jain 1990. "Using dynamic programming for solving variational problems in vision". IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(9):855-867 p.

Ballerini, L. 1999. "Genetic snakes for medical images segmentation". Lectures Notes in Computer Science, 1596:59-73 p.

Beaudet, P. R. 1978. "Rotationally invariant image operators". En: "Proceedings of the 4th International Joint Conference on Pattern Recognition", volume 4. Kyoto, Japón, 579-583 p.

Bentley, P. J. 1999. "Evolutionary desing by computers". Morgan Kaufmann, San Francisco, primera edición. 446 p.

Blake, A. y M. Isard 1998. "Active contours". Springer, Londres, primera edición. 352 p.

Branch, J. W. y G. Olague 2001. "La visión por computador, una aproximación al estado del arte". DYNA, 68(133):1-16 p.

Cohen, L. D. 1991. "On active contour models and balloons". Computer Vision, Graphics, and Image Processing. (CVGIP): Image Understanding, 53(2):211-218 p.

Darwin, C. 1859. "On the origin of species by means of natural selection". John Murray, Londres, primera edición. 459 p.

Deb, K. 2001. "Multi-objective optimization using evolutionary algorithms". John Wiley & sons, Baffins Lane, Chichester, West Sussex, PO19 1UD, England, primera edición. 497 p.

Dreschler, K. y H. H. Nagel 1981. "Volumetric model and 3d trajectory of a moving car derived from monocular tv frame sequence of a street scene". Computer Vision, Graphics and Image Processing, 20(3):199-228 p.

Elzinga, D. J. y D. W. Hearn 1972. "The minimum convering sphere problem". Management Science, 19(1):96-104 p.

Foley, J. D., A. van Dam, S. K. Feiner, y J. F. Hughes 1990. "Computer graphics: Principles and practice". Addison Wesley, Massachusetts, segunda edición. 1175 p.

Förstner, W. 1994. "A framework for low level feature extraction". Proceedings of the 3rd European Conference on Computer Vision, 3:101-124 p.

Forsyth, D. y J. Ponce 2002. "Computer vision: A modern approach". Prentice Hall, primera edición. 693 p.

Goldberg, D. E. 1989. "Genetic algorithms for search, optimization, and machine learning". Addison-Wesley, Reading, MA. USA, primera edición. 372 p.

Gonzalez, R. y R. Woods 2002. "Digital image processing". Prentice Hall, Inc., Saddle River, New Jersey, segunda edición. 793 p.

Harris, C. y M. Stephens 1988. "A combined corner and edge detector". En: "Alvey Vision Conference", volume 15. Manchester, UK, 147-151 p.

Hearn y Baker 1995. "Gráficas por computadora". Prentice Hall, segunda edición. 686 p.

Holland, J. 1975. "Adaptation in natural and artificial systems". University of Michigan press, Michigan, primera edición. 211 p.

Kass, M., A. Witkin, y D. Terzopoulos 1988. "Snakes: Active contour models". International Journal of Computer Vision, 1(4):321-331 p.

Kennedy, J. y R. C. Eberhart 2001. "Swarm intelligence". Morgan Kaufmann, San Francisco, CA, primera edición. 512 p.

Kitchen, L. y A. Rosenfeld 1982. "Gray-level corner detection". Pattern Recognition Letters, 1(2):95-102 p.

Koza, J. R. 1992. "Genetic programing: On the programming of computers by means of natural selection". MIT press, Cambridge, MA. USA, primera edición. 840 p.

MacEachern, L. A. y T. Manku 1998. "Genetic algorithms for active contour optimization". IEEE International Symposium on Circuits and Systems, 4:229-232 p.

MacQueen, J. B. 1967. "Some methods for classification and analysis of multivariate observations". En: "Proceedings of fifth Berkeley Symposium on Mathematical Statistics and Probability", volume 1. University of California Press. Berkeley, California, 281-297 p.

McInerney, T. y D. Terzopoulos 1995. "Topologically adaptable snakes". En: "Proc. Fifth International Conference on Computer Vision, ICCV'1995". Cambridge, MA, IEEE Computer Society Press. Los Alamitos, CA, 840-845 p.

Michalewicz, Z. 1982. "Genetic algorithms + data structures = evolution programs". W. H. Freeman and Company, New York, primera edición. 385 p.

Moravec, H. P. 1977. "Towards automatic visual obstacle avoidance". IJCAI, 1:584 pp.

Preparata, F. R. y M. I. Shamos 1985. "Computational geometry: An introduction". Springer-Verlag, New York, primera edición. 390 p.

Roberts, L. 1965. "Machine perception of three dimensional solids". In Optical and Electro-Optical Information Processing, Tippet, J.T (ed.), MIT Press, Cambridge, Mass.

Schwefel, H. P. 1995. "Evolution and optimum seeking". Wiley, New York, primera edición. 444 p.

Shi, J. y C. Tomasi 1994. "Good features to track". En: "Conference on Computer Vision and Pattern Recognition (CVPR94)". IEEE. Seatle, 593-600 p.

Strovik, G. 1994. "A bayesian approach to dynamic contours through stochastic sampling and simulated annealing". IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(10):976-986 p.

Trujillo, L. y G. Olague 2006. "Synthesis of interest point detectors through genetic programming". En: "Genetic and Evolutionary Computation Conference (GECCO)". Seattle, Washington, USA, 887-894 p.

Wang, H. y M. Brady 1995. "Real time corner detection algorithms for motion estimation". Image and Vision Computing, 13(9):695-713 p.

Williams, D. J. y M. Shah 1992. "A fast algorithm for active contours and curvature estimation". Computer Vision, Graphics, and Image Processing. (CVGIP): Image Understanding, 55(1):14-26 p.

Xu, C. y J. L. Prince 1997. "Gradient vector flow: A new external force for snakes". IEEE Proceeding in Conference on Computer Vision and Pattern Recognition (CVPR97), 1:66-71 p.

Apéndice A

Herramientas Computacionales

A lo largo de este trabajo se utilizaron diferentes herramientas de software para poder implementar con efectividad los diferentes paradigmas sobre *Snakes*, puntos de interés y otros algoritmos. A continuación se presentan los principales aspectos de los diferentes paquetes de software utilizados para alcanzar diversos objetivos.

A.1 OpenCV

OpenCV es una biblioteca de código abierto para Visión por Computadora (en inglés Open Source Computer Vision) desarrollado por la compañía Intel. La página oficial de OpenCV es:

http://www.intel.com/technology/computing/opencv/

OpenCV contiene una gran colección de algoritmos y código de ejemplo para varios problemas de Visión por Computadora. Así como también funciones en C y clases en C++ que implementan algunos algoritmos populares de Procesamiento de Imágenes y Visión por Computadora.
Descripción general:

- OpenCV utiliza el lenguaje de programación C/C++.
- OpenCV se puede utilizar tanto para el sistema operativo Linux como Windows.
- Es gratis, tanto para uso comercial y no comercial.

OpenCV se enfoca principalmente al procesamiento de imágenes en tiempo real. Un ejemplo de una aplicación exitosa de OpenCV es en el sistema de visión del proyecto *Stanley* de la Universidad de Stanford, ganador del gran reto DARPA 2005 (Defense

Advanced Research Projects Agency - Grand Challenge Race). Las áreas de aplicación incluyen:

- Interfaz humano-computadora (HCI).
- Identificación de objetos.
- Segmentación y reconocimiento.
- Reconocimiento de caras.
- Reconocimiento de gestos.
- Seguimiento de objetos en movimiento.
- Análisis de movimiento.
- Robótica móvil.

Características:

- Manipulación de datos de una imagen (cargar, convertir, copiar).
- Entrada y salida de imágenes y video (entrada basada en nombre de archivo y número de cámara, y salida basada en archivos de imágenes o video).
- Manipulación de matrices y vectores, y rutinas de álgebra lineal (productos, eigen valores, SVD).
- Varios tipos de estructuras de datos dinámicas (listas, colas, conjuntos, árboles).
- Funciones básicas para procesamiento de imágenes (filtrado, detección de bordes, detección de esquinas, muestreo e interpolación, conversión de color, operaciones morfológicas, histogramas).
- Análisis estructural (componentes conectados, procesamiento de contornos, transformada Hough, triangulación Delaunay, aproximación de polígonos).

- Calibración de cámaras (búsqueda y seguimiento de patrones de calibración, estimación de la matriz fundamental, estimación de homografías, correspondencia estéreo).
- Análisis de movimiento (flujo óptico, segmentación de movimiento, seguimiento).
- Reconocimiento de objetos.
- Unidad de interfaz gráfica básica (desplegar imágenes o video, eventos de teclado y mouse, barras de desplazamiento).
- Edición de imágenes (líneas, cónicas, polígonos, texto).

OpenCV está compuesto por diversas librerías enfocadas a realizar funciones específicas. Los módulos de OpenCV son:

- cv - Funciones principales.
- cvaux - Funciones auxiliares (experimentales).
- cxcore - Estructuras de datos y soporte de álgebra lineal.
- highgui - Funciones de la unidad de interfaz gráfica (GUI por sus siglas en inglés Graphics Unit Interface).
- cvcam - Funciones para procesamiento de video de cámaras digitales (sólo disponible para la versión OpenCV-0.9.5).

La documentación de OpenCV se puede adquirir de la misma página donde se descarga (ver siguiente sección), aunque la mayoría de la documentación oficial también se encuentra dentro del paquete de instalación. Sin embargo, mucha de la documentación se encuentra obsoleta por los constantes cambios, renovación, actualizaciones y reparación de *bugs*. Por lo que una buena alternativa para obtener información, aclarar dudas del uso de funciones de OpenCV, y hasta pedir opiniones para resolver problemas diversos de visión es utilizar el sistema grupos de OpenCV. El grupo se encuentra formado en su mayoría por investigadores de las diferentes áreas de visión, estudiantes,

Tabla I: Últimas versiones estables de OpenCV para Linux.

Nombre	Versión	Fecha de liberación
Beta 5	OpenCV-0.9.7	26 de Julio del 2005
Beta 4	OpenCV-0.9.6	18 de Agosto del 2004
Beta 3.1	OpenCV-0.9.5	5 de Marzo del 2003

y aficionados de diversas partes del mundo. Se puede inscribir al grupo de OpenCV a través de la siguiente página, donde el único requisito es tener una cuenta de correo electrónico de *yahoo*.

http://groups.yahoo.com/group/OpenCV/

Otra alternativa para familiarizarse con OpenCV, encontrar tutoriales, cursos y muchos ejemplos de programas es la página:

http://opencvlibrary.sourceforge.net/

A.1.1 Instalación

OpenCV se puede obtener en la página de internet:

http://sourceforge.net/projects/opencvlibrary/

donde se encuentran disponibles diferentes versiones tanto para Linux como para Windows, así como documentación y parches para los paquetes. La Tabla I muestra las últimas versiones estables de OpenCV para Linux y su fecha de liberación. Se puede observar en la tabla que la última versión estable tiene más de un año a la fecha de esta tesis. Por lo que esta versión podría tener problemas de instalación o compatibilidad con nuevas versiones del sistema operativo, o versiones de *kernel* de linux.

Una alternativa es utilizar la versión CVS (por sus siglas en inglés de Concurrent Versions System) para obtener la última actualización de OpenCV y que puede resolver algunos de los problemas mencionados anteriormente. CVS es una herramienta que se utiliza por muchos desarrolladores de software para la administración de cambios en su árbol de código fuente. CVS provee los medios para guardar no sólo la versión actual de

una pieza del código fuente, sino también guardar un registro de todos los cambios (así como quién hizo los cambios) que han ocurrido en el código fuente. La utilización de CVS es particularmente común en proyectos con múltiples desarrolladores, ya que CVS asegura que los cambios hechos por un desarrollador no sean accidentalmente removidos cuando otro desarrollador entregue sus cambios al árbol de código fuente.

Para tener acceso al repositorio CVS, se debe instalar un software especial llamado cliente CVS. Sin embargo, los clientes CVS están disponibles en la mayoría de los sistemas operativos. En este caso la instalación típica de Linux Fedora Core 4 ya cuenta con el cliente CVS, por lo que no es necesario instalarlo.

Para obtener la versión CVS de OpenCV completa es necesario conectarse al repositorio CVS, lo cual se puede hacer como un usuario anónimo con el siguiente conjunto de instrucciones. Cuando se pregunte la contraseña del usuario anónimo, simplemente se presiona la tecla Enter.

```
$ cvs -d:pserver:anonymous@opencvlibrary.cvs.sourceforge.net:/cvsroot/opencv-
library login
$ cvs -z3 -d:pserver:anonymous@opencvlibrary.cvs.sourceforge.net:/cvsroot/op-
encvlibrary co -P opencv
```

En caso que no se requiera obtener el paquete de OpenCV completo, sino solamente un módulo de éste; entonces es necesario especificar el nombre del módulo en lugar de la palabra "opencv" en la última línea de instrucciones. Para determinar los nombres de los módulos creados por este proyecto, se puede examinar el repositorio CVS mediante una aplicación de observación tipo web.

Una vez descargado el paquete OpenCV versión CVS, el siguiente paso es compilarlo e instalarlo. La versión CVS no se encuentra comprimido en un solo archivo, sino que al descargarlo crea un directorio llamado opencv que contiene todos los módulos, archivos de instalación, así como documentación y ejemplos. De esta manera, el procedimiento de instalación consiste en entrar al directorio opencv, configurar, compilar e instalar; como se muestra en las siguientes líneas de comandos:

```
$ cd opencv
$ ./configure
$ make
# make install
```

Los directorios de instalación por omisión son /usr/local/lib y /usr/local/include/-opencv. Por último, es necesario especificar al sistema operativo Linux dónde se encuentran las bibliotecas para poder utilizarlas. Linux utiliza el archivo /etc/ld.so.conf para conocer la dirección de las librerías que se necesitan por un programa en tiempo de ejecución. Por lo que se debe modificar este archivo para que Linux pueda encontrar las nuevas librerías. Para hacer esto se puede utilizar el comando *vi* para modificar el archivo /etc/ld.so.conf y agregar la dirección /usr/local/lib, o también se puede utilizar cualquier otro editor de texto como *emacs*, *gedit*, o *kedit*. Una vez modificado el archivo, se debe utilizar el comando *ldconfig* para que Linux detecte que existe otra dirección para librerías dinámicas.

```
# gedit /etc/ld.so.conf
# /sbin/ldconfig
```

Para asegurarse que OpenCV se instaló correctamente, se puede compilar y ejecutar uno de los ejemplos que ya contiene. Por ejemplo, se puede compilar y ejecutar el programa de ejemplo *convexhull* de la siguiente manera:

```
$ cd opencv/samples/c
$ g++ convexhull.c -I/usr/local/include/opencv -L/usr/local/lib -lhighgui
 -lcvaux -lcxcore -lcv -o convexhull
$ ./convexhull
```

En caso que se desee instalar alguna de las versiones estables, sólo es necesario descomprimir la versión estable correspondiente y seguir los mismos pasos que en la versión CVS. Por ejemplo:

```
$ tar -xvzf opencv-0.9.7.tar.gz
$ cd opencv-0.9.7
```

A.1.2 Instalación del paquete ffmpeg

OpenCV también se puede compilar ligando la librería ffmpeg, que contiene funciones para poder grabar y convertir flujos de audio y video. Lo cual es muy importante si se desea crear secuencias de video de prueba para los algoritmos de visión. Se puede obtener la librería ffmpeg utilizando un cliente SVN (por sus siglas en inglés de SubVersioN). SVN es una aplicación de código abierto para la revisión de programas en desarrollo y

es la versión moderna de reemplazo de CVS. Para descargar ffmpeg utilizando SVN, se utiliza la siguiente línea de comandos:

```
$ svn checkout svn://svn.mplayerhq.hu/ffmpeg/trunk ffmpeg
```

Después de descargar el paquete ffmpeg, se procede a configurarlo, compilarlo e instalarlo de la siguiente manera:

```
$ cd ffmpeg
$ ./configure --enable-shared
$ make
# make install
```

Es importante mencionar que OpenCV es capaz de procesar archivos de video o videos adquiridos por una cámara sin necesidad de utilizar ffmpeg, por lo que sólo es necesario si se requiere grabar archivos de video o convertir un video de un formato a otro (por ejemplo avi a mpeg). Si se requiere utilizar OpenCV con soporte ffmpeg, se debe instalar primero ffmpeg y después OpenCV. De tal manera que al configurar OpenCV se detecte la presencia de ffmpeg.

A.1.3 Cómo compilar programas con OpenCV

Como se aprecia al final de la sección A.1.1 sobre la instalación de OpenCV, la línea de comandos para compilar una aplicación es relativamente larga. Por lo que es posible crear un archivo llamado *Makefile* para utilizar el comando *make* y compilar automáticamente los programas.

Se puede crear el archivo Makefile utilizando cualquier procesador de textos (vi, emacs, gedit) de Linux. En las siguientes líneas se muestra el contenido de un archivo Makefile para compilar programas de OpenCV.

```
CC  = g++
INC = -I/usr/local/include/opencv
LIB = -L/usr/local/lib
OPT = -lhighgui -lcvaux -lcxcore -lcv
BIN = opencv1 opencv2

all:  $(BIN)

opencv1: opencv1.c Makefile
```

```
        $(CC) opencv1.c $(INC) $(LIB) $(OPT) -o $@
opencv2: opencv2.c Makefile
        $(CC) opencv2.c $(INC) $(LIB) $(OPT) -o $@
clean:
        rm *~ $(BIN)
```

Primero se define una variable CC que contiene el compilador que se utiliza para OpenCV, en este caso se necesita el compilador *g++*. La variable INC le indica al compilador donde encontrar los archivos de cabecera (*.h) para OpenCV y la variable LIB indica la dirección de las librerías de OpenCV. La variable OPT indica las librerías que serán utilizadas de la dirección de librerías, en este caso los cuatro módulos de OpenCV deben ser suficientes: highgui, cvaux, cxcore y cv.

Se define una variable BIN que contiene la lista de programas a compilar, y para cada uno se define la lista de archivos que se compilarán cuando se modifiquen. Por ejemplo, el programa opencv1 depende de los archivos opencv1.c y Makefile; por lo que cuando uno de estos archivos tenga un cambio, el siguiente comando *make* ejecutará la compilación de opencv1.

La última parte del archivo es útil para limpiar el directorio (borrar los archivos ejecutables y temporales) con el comando *make clean.* Por defecto el comando *make* realiza un *make all* que compila todos los programas definidos dentro del archivo Makefile, por ejemplo:

```
$ make
  g++ opencv1.c -I/usr/local/include/opencv -L/usr/local/lib -lhighgui
     -lcvaux -lcxcore -lcv -lavcodec -lavutil -o opencv1
  g++ opencv2.c -I/usr/local/include/opencv -L/usr/local/lib -lhighgui
     -lcvaux -lcxcore -lcv -lavcodec -lavutil -o opencv2
```

Sin embargo, es posible compilar individualmente cualquier programa que se encuentre definido en Makefile. Por ejemplo:

```
$ make opencv1
  g++ opencv1.c -I/usr/local/include/opencv -L/usr/local/lib -lhighgui
     -lcvaux -lcxcore -lcv -lavcodec -lavutil -o opencv1
```

compila solamente el archivo opencv1.c si es necesario. Si no existen errores se debe obtener un archivo ejecutable llamado opencv1 que se puede ejecutar como:

```
$ ./opencv1
```

De este modo, cada vez que se cree un nuevo archivo de programa (*.c) se debe definir de la misma manera que los ejemplos opencv1 y opencv2 dentro del archivo Makefile.

A.2 IDL

IDL significa Lenguaje de Datos Interactivo (en inglés Interactive Data Language), que es un software para análisis de datos, visualización, y desarrollo de aplicaciones multi-plataforma (Linux, Windows). IDL integra un lenguaje orientado a matrices, numerosos análisis matemáticos y técnicas de despliegue gráfico; teniendo una gran flexibilidad. Unas cuantas líneas en IDL pueden hacer el trabajo de cientos de líneas en C o Fortran, sin perder flexibilidad o desempeño. Por lo que a IDL se le considera un lenguaje de cuarta generación, ya que es radicalmente más compacto que lenguajes como C o Fortran. Tareas que requieren días o semanas de programación con lenguajes tradicionales puede ser completadas con la utilización de IDL en horas. Un diseñador puede explorar datos interactivamente utilizando los comandos de IDL y crear aplicaciones completas escribiendo programas de IDL.
Las ventajas de IDL incluyen:

- Es un lenguaje estructurado completo que puede ser utilizado para crear funciones sofisticadas, procedimientos y aplicaciones.
- Los operadores y funciones trabajan sobre las matrices completas sin necesidad de utilizar ciclos, simplificando el análisis interactivo y reduciendo el tiempo de programación.
- La inmediata compilación y ejecución de los comandos de IDL proveen una retroalimentación instantánea facilitando la interacción.
- Soporte para hardware de aceleración gráfica basada en OpenGL.
- Gran cantidad de rutinas para análisis numérico y estadístico.
- La entrada y salida de datos es flexible en IDL, ya que puede leer casi cualquier tipo de formato de datos. El soporte incluye:

  - Estándares de imágen comunes: BMP, TIFF, Interfile, JPEG, PICT, PNG, PPM, SRF, X11 Bitmap, y XWD.
  - Formato de datos científicos: CDF, HDF, y NetCDF.
  - Otros formatos de datos: ASCII, Binario, DICOM, DXF, WAV, y XDR.

- Los programas de IDL pueden correr prácticamente en todas las plataformas soportadas (UNIX, VMS, Microsoft Windows, y sistemas Macintosh) con pequeñas o nulas modificaciones. Por lo que la portabilidad de esta aplicación permite el fácil soporte para una variedad de computadoras.

- Rutinas de FORTRAN y C pueden ser ligadas dinámicamente dentro de IDL para agregar cierta funcionalidad especializada. Y alternativamente, programas de C y FORTRAN pueden mandar llamar rutinas de IDL como un librería de subrutinas o motor de despliegue.

A.3 VisLib

Es una biblioteca de procesamiento de visión de alto rendimiento de la compañia ActivMedia Robotics, LLC. VisLib provee una biblioteca rápida y flexible para procesamiento de imágenes y visión de máquina de una sola cámara. Con VisLib se pueden crear rápidamente aplicaciones para detección de movimiento y seguimiento de objetos. VisLib contiene funciones para:

- Adquisición, despliegue y formato de imágenes.

- Procesamiento de imágenes 2D.

- Seguimiento de objetos.

Las funciones de adquisición y despliegue de imágenes son dependientes del hardware. La versión actual de VisLib está diseñada para Linux y puede trabajar con cualquier *framegrabber* soportado por el controlador Bt8xx de Linux y cualquier sevidor X11, el cual provee una visualización de color verdadero (8-24 bit).

Las funciones de procesamiento de imágenes son: la convolución general (máscaras 2D), herramientas de filtrado y operadores morfológicos. Los cuales se han probado y optimizado para velocidad.

Las funciones de seguimiento de objetos están diseñadas para una sola cámara, con seguimiento basado en color. Para empezar, un objeto se define especificando una forma 2D y parámetros de color. Entonces, mientras el objeto se sigue, la actualización de funciones se utilizan para continuamente redefinir estos parámetros. De esta manera, los cambios de adaptación a la forma, color y orientación se realizan directamente.

Las herramientas que se proveen permiten muchos parámetros (características) para la definición de un objeto, incluyendo *blobs* y mapas de bordes. Los objetos se definen inicialmente por crecimiento de regiones con simples condiciones de fronteras; por ejemplo, hasta que se encuentre un cambio máximo en el parámetro especificado del valor de pixel de inicio. Otras rutinas tratan con la actualización de la forma de un objeto, actualización de color, y la redefinición del objeto. La localización de un objeto en cada frame se realiza utilizando la correlación de la forma binaria del objeto con una ventana (subimagen) de seguimiento que ha sido filtrada con el umbral de colores normalizado del objeto.

A.3.1 Instalación

VisLib se puede obtener en la página de internet:

http://robots.mobilerobots.com/

Para instalarlo, se guarda el archivo de distribución en el directorio que se desee, y después se descomprime el paquete y por último se compila. Por ejemplo:

```
# cp vislib-1.8.tgz /usr/local
# cd /usr/local
# tar -zxvf vislib-1.8.tgz
# cd vislib-1.8
# make
```

La librería VisLib provee un gran conjunto de herramientas útiles para procesar imágenes; sin embargo, VisLib no está diseñado para trabajar con cámaras web y por

ello, si se desean utilizar estas herramientas será necesario adquirir una cámara especial, lo que conlleva un costo considerable.

A.4 MATLAB

MATLAB es el nombre abreviado de Laboratorio de Matrices (en inglés MATrix LABoratoy). MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular también puede trabajar con números escalares, tanto reales como complejos.
Características:

- Lenguaje de alto nivel para cálculo técnico.
- Herramientas interactivas para exploración, diseño y resolución de problemas iterativos.
- Funciones matemáticas para álgebra lineal, estadística, análisis de Fourier, filtrado, optimización e integración numérica.
- Funciones gráficas bidimensionales y tridimensionales para visualización de datos.
- Herramientas personalizadas para crear interfaces gráficas de usuario.
- Funciones para integrar los algoritmos basados en MATLAB con aplicaciones y lenguajes externos, tales como C/C++, FORTRAN, Java, COM y Microsoft Excel.

Aplicaciones:

- Procesamiento digital de imágenes (filtrado de imágenes, adquisición, análisis y visualización).
- Cómputo numérico (solución numérica de ecuaciones diferenciales).
- Procesamiento digital de señales y comunicación.
- Modelado de análisis financiero.

- Redes neuronales.

A.5 Biblioteca CImg

CImg es una biblioteca de código abierto en C++ para procesamiento de imágenes, que consiste sólo de un archivo cabecera CImg.h, el cual provee un conjunto de funciones y clases que pueden ser utilizadas desde cualquier programa en C++ para cargar, grabar, procesar y desplegar imágenes. CImg es eficiente y simple de utilizar. Algunas de sus principales características son las siguientes:

- Es altamente portable, ya que puede trabajar en sistemas operativos como Unix/X11, Windows, MacOS X y FreeBSD.

- Contiene algoritmos útiles para procesamiento de imágenes para cargar, grabar, desplegar, escalar, rotar, y filtrar imágenes; herramientas de dibujo (texto, líneas, curvas, elipses, etc.).

- Las imágenes son instanciadas por una clase capaz de representar imágenes de hasta cuatro dimensiones (x, y, z, v), desde señales escalares unidimensionales hasta volúmenes 3D.

- Depende de un mínimo número de librerías, por lo que se puede compilar sólo con las librerías estándar de C y no se necesitan librerías especiales o dependencias complejas.

- Se pueden obtener características especiales con el uso de paquetes como ImageMagick, libpng o libjpeg. Por ejemplo, se puede instalar el paquete ImageMagick o ligar el código de un programa con libpng o libjpeg para poder cargar y grabar imágenes en formato comprimido (GIF, TIF, JPG, PNG).

A.5.1 Instalación

Existen dos maneras diferentes de obtener la librería CImg:

1. La librería se distribuye como un paquete comprimido en formato ZIP que es independiente de la plataforma que se vaya a utlizar. El archivo contiene el código fuente de la librería, así como también varios ejemplos para ilustrar el uso de las funciones y clases de la librería. Se puede obtener el último paquete estable de la siguiente dirección de internet:

   http://sourceforge.net/project/showfiles.php?group_id=96492

2. Al igual que OpenCV también existe una versión CVS. Lo que asegura que se obtendrá la última versión del código disponible, y además facilita la actualización. Sin embargo, es importante mencionar que algunas partes del código en la versión CVS pueden ser experimentales. Para obtener la versión CVS sólo se necesita ejecutar el siguiente par de instrucciones:

```
# export CVS_RSH=ssh
# cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/cimg co CImg
```

Apéndice B

Cámara Logitech Quickcam Pro 4000

El elemento principal de un sistema de visión es el dispositivo de adquisición de imágenes. Sin embargo, existen una gran variedad de dispositivos empleados para este fin: cámaras de video, microdensitómetros, escáners de barrido y matrices fotosensibles de estado sólido, etc. En la actualidad, las cámaras digitales son los elementos que se emplean comúnmente en los sistemas de adquisición de imágenes. Las cámaras contruidas a partir de matries fotosensibles de estado sólido (CCD por sus siglas en inglés de Charge Coupled Device) sensan las variaciones de intensidad de la luz ocurrida en la escena. Dependiendo de la tecnología de los CCD's la imagen adquirida puede ser en tonos de gris o en colores.

La selección adecuada de una cámara muchas veces depende de la aplicación. Así, es posible encontrar muchos tipos de cámaras especiales para tareas específicas de visión, como cámaras infrarrojas, cámaras de alta resolución, etc. La mayoría de las cámaras especializadas tienen la desventaja de que son muy costosas y con frecuencia son difíciles de conseguir. Por lo que una de las soluciones es utilizar cámaras web, para utilizarlas como sistema de adquisición de propósito general. Las cuales son significativamente más baratas y fáciles de conseguir.

Ya que el propósito de este trabajo es utilizar los contornos activos para segmentar objetos en una escena, una cámara web puede ser suficiente para alcanzar estos objetivos. De esta manera en este trabajo se utiliza una cámara marca Logitech, modelo Quickcam Pro 4000. Algunas características de esta cámara son las siguientes:

- Resolución de video de hasta 640×480 pixeles con sensores avanzados CCD VGA.
- Compatible con USB 2.0 y 1.1

- Velocidad de video (en inglés framerate) de hasta 30 imágenes por segundo (fps por sus siglas en inglés frames per second).
- Formato YUV 4:2:0 planar.

El formato de imagen, o paleta como también se conoce, se refiere a la manera en que los datos son almacenados en memoria. Existen muchos tipos de formato por lo que es importante asegurarse que el software que se utilice para manipular y desplegar las imágenes soporte el formato de la cámara. Entre los formatos que más se utilizan en cámaras web son el RGB y el YUV. El formato YUV es un método que define a una señal de video que separa los componentes de luminancia (Y) y crominancia (UV).

B.1 Instalación

El sistema operativo que se utiliza en este trabajo es Linux, en la distribución de Fedora Core 4 (FC4) con un kernel 2.6.11-1.1369-FC4. A diferencia del sistema operativo Windows, en Linux no existe un controlador (en inglés driver) desarrollado por Logitech para esta cámara. Por lo que es necesario utilizar un controlador compatible. Un controlador no es más que un programa de software que permite la comunicación entre un dispositivo y el sistema operativo. En este caso se utiliza el controlador para cámaras Philips (PWC por sus siglas en inglés Philips Webcam Driver). El cual soporta muchas cámaras Philips y compatibles, entre ellas la cámara Logitech Quickcam Pro 4000.

El controlador PWC se puede obtener en la página de internet

http://www.saillard.org/linux/pwc/

En este caso se utiliza la versión 10.0.7a que era la última versión disponible hasta el momento. El código del controlador puede ser compilado directamente dentro del kernel de Linux o como un módulo externo. Sin embargo, éste último es el que se recomienda ya que permite una depuración más fácilmente. El procedimiento para instalar el controlador PWC como un módulo en Linux Fedora Core 4 con kernel 2.6.11.* es el siguiente:

1. FC4 tiene una versión antigua de PWC preinstalada que no funciona para la cámara Logitech, por lo que el primer paso es desinstalar la versión antigua. De este modo, se debe borrar el siguiente directorio y el archivo que contiene, y por último descargar el módulo:

```
rm -R /lib/modules/2.6.11.*/kernel/drivers/usb/media/pwc
/sbin/rmmod pwc
```

2. Se debe descomprimir el paquete pwc-10.0.7a en cualquier directorio (de preferencia el directorio HOME) de la siguiente manera:

```
tar zxvf pwc-10.0.7a.tar.gz
o
tar jxvf pwc-10.0.7a.tar.bz2
```

3. Se entra al directorio y se compila e instala el nuevo controlador como root:

```
su
cd pwc-10.0.7a
make && make install
```

4. Ahora se debe verificar si el nuevo archivo binario pwc.ko se encuentra en la ruta /lib/modules/2.6.11.*/kernel/drivers/usb/media/ y cambiarle los permisos:

```
cd /lib/modules/2.6.11.*/kernel/drivers/usb/media/
chmod 744 pwc.ko
```

5. Y por último se carga el módulo:

```
/sbin/modprobe pwc
```

Una vez instalado el controlador, se puede probar la cámara utilizando alguna aplicación para cámaras web de linux como *xawtv* o *gnomemeeting*.

B.2 SETPWC

SETPWC es una utilería de código abierto para configurar varios aspectos del controlador PWC para cámaras philips, entre los compatibles con la cámara Logitech se encuentran:

- Mostrar la configuración actual.
- Ajustar el framerate.
- Configurar el nivel de compresión.
- Activar el control automático de ganancia.
- Activar compensación de luz de fondo.
- Ajustar el modo de reducción de ruido.

La utilería SETPWC se puede obtener en la página de internet

http://www.vanheusden.com/setpwc/

En este caso se utiliza la versión 1.0, que a la fecha era la última versión disponible. Para instalar la utilería sólo es necesario descomprimir el paquete y compilarlo, de la siguiente manera:

```
tar -xvzf setpwc-1.0.tar.gz
cd setpwc-1.0
make
```

Una vez instalada la utilería se puede probar utilizando el siguiente comando en el directorio donde se instaló SETPWC:

```
./setpwc -p
```

El cual despliega el estado actual de la cámara. Algunas funciones compatibles con la cámara Logitech que son de utilidad se muestran en la Tabla II. Una característica importante de SETPWC es que está escrito en lenguaje de programación C++. Por lo que si se necesita utilizar alguna función de configuración (como el framerate) en

Tabla II: Algunas funciones de la utilería SETPWC.

Parámetro	Función
-d x	Cambiar de cámara.
-f x	Establecer el framerate (3-30).
-c x	Establecer el nivel de compresión (0-3).
-g x	Establecer el control automático de ganancia (0-65535).
-q x	Establecer el nivel de reducción de ruido (0-3).
-n x	Compensación de luz de fondo (0=desactivado, otro= activado)
-x	Reestablecer valores por defecto.
-h	Desplegar todas las opciones disponibles.

alguna aplicación que también utilice C++ (como OpenCV o VISLIB ver Apéndice A), es posible simplemente copiar el código de la función para utilizarlo en nuestra aplicación.

B.3 Visión estéreo

El framerate y resolución de video dependen del ancho de banda del controlador [1] USB. El video utiliza una gran cantidad de bytes para obtener las imágenes de la cámara, y lo hace mediante el controlador USB. De esta manera, en este caso, el ancho de banda USB permite adquirir imágenes de video con una resolución máxima de 640 × 480 pixeles, pero con un framerate de sólo 10fps aun con compresión. Sin embargo, también es posible obtener el máximo framerate de 30fps utilizando una resolución de 320 × 240, lo cual es lo más adecuado para una aplicación de tiempo real.

Una característica importante del controlador PWC es que puede soportar múltiples cámaras web trabajando al mismo tiempo. Sin embargo, al igual que el framerate y resolución de video, el ancho de banda del controlador USB puede limitar el número de cámaras que se pueden utilizar simultáneamente. En teoría es posible trabajar hasta con 5 cámaras en un solo controlador USB, pero únicamente con las resoluciones y framerates más bajos. Dos cámaras con una resolución media es más razonable. Si

[1] En este caso el controlador USB se refiere a un dispositivo de memoria de hardware que no se debe confundir con el controlador (driver) de la cámara que es software.

es necesario utilizar más cámaras simultáneamente, o una resolución y framerate más grandes, se debe instalar un controlador USB extra. Por otro lado, en la actualidad muchas tarjetas madres (en inglés motherboards) para computadoras tienen instalados múltiples controladores. Por lo general las computadoras contienen 2 controladores USB, uno de los cuales está cableado hacia los puertos USB traseros de la computadora y el otro hacia los puertos USB frontales o laterales de la computadora.

De este modo, existen dos maneras de utilizar dos cámaras al mismo tiempo:

1. Si sólo se cuenta con un controlador USB, es posible obtener una resolución de 320×240 y un framerate de 30fps para ambas cámaras utilizando la utilería SETPWC y establecer el nivel de compresión al máximo para ambas cámaras.

2. Si se cuenta con dos controladores USB (uno frontal y otro trasero), esta opción consiste simplemente en conectar las cámaras en puertos USB manejados por diferentes controladores.

Una observación importante acerca de los controladores USB es que éstos generalmente manejan varios puertos USB; es decir, diferentes puertos USB no significa diferentes controladores USB. Por lo que hay que tener cuidado con los diferentes dispositivos que pueden estar conectados a los diferentes puertos USB, ésto incluye teclados USB, escáners, mouse ópticos, etc. los cuales utilizan de igual manera ancho de banda del controlador USB. De esta manera es necesario identificar qué puertos son manejados por un determinado controlador.