CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN.

TESIS

SISTEMA FORMAL QUE PERMITE RECONOCER DESCRIPTORES DE OBJETOS BINARIOS

PRESENTA

Osvaldo Arturo Tapia Dueñas

PARA OPTAR POR EL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

TUTOR

Dr. Hermilo Sánchez Cruz
Dr. Ángel Eduardo Muñoz Zavala (Co-Tutor)

COMITÉ TUTORAL

Dr. Hiram Habid López Valdez (Asesor)

Aguascalientes, Ags., a 8 de Abril de 2019

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. José de Jesús Ruiz Gallegos
DECANO DEL CENTRO DE CIENCIAS BÁSICAS
PRESENTE

Por medio de la presente, en mi calidad de tutor designado del estudiante **Osvaldo Arturo Tapia Dueñas** con ID 152131 quien realizó la tesis titulada: **SISTEMA FORMAL QUE PERMITE RECONOCER DESCRIPTORES DE OBJETOS BINARIOS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirla, y así continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y, sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 8 de Abril de 2019

Dr. Hermilo Sánchez Cruz

c.c.p.- Interesado
c.c.p.- Secretaría de Investigación y Posgrado
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. José de Jesús Ruiz Gallegos
DECANO DEL CENTRO DE CIENCIAS BÁSICAS
PRESENTE

Por medio de la presente, en mi calidad de sinodal designado del estudiante **Osvaldo Arturo Tapia Dueñas** con ID 152131 quien realizó la tesis titulada: **SISTEMA FORMAL QUE PERMITE RECONOCER DESCRIPTORES DE OBJETOS BINARIOS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirla, y así continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y, sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 8 de Abril de 2019

Dr. Hiram Habid López Valdez

c.c.p.- Interesado
c.c.p.- Secretaría de Investigación y Posgrado
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. José de Jesús Ruiz Gallegos
DECANO DEL CENTRO DE CIENCIAS BÁSICAS
PRESENTE

Por medio de la presente, en mi calidad de sinodal designado del estudiante **Osvaldo Arturo Tapia Dueñas** con ID 152131 quien realizó la tesis titulada: **SISTEMA FORMAL QUE PERMITE RECONOCER DESCRIPTORES DE OBJETOS BINARIOS**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que él pueda proceder a imprimirla, y así continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y, sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 8 de Abril de 2019

Dr. Ángel Eduardo Muñoz Zavala

c.c.p.- Interesado
c.c.p.- Secretaría de Investigación y Posgrado
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

OSVALDO ARTURO TAPIA DUEÑAS
MAESTRIA EN CIENCIAS CON OPCION A
COMPUTACION, MATEMATICAS APLICADAS

Estimado alumno:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis y/o caso práctico titulado: **"SISTEMA FORMAL QUE PERMITE RECONOCER DESCRIPTORES DE OBJETOS BINARIOS"** hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., a 15 Abril de 2019
"Se lumen proferre"
EL DECANO

M. en C. JOSÉ DE JESÚS RUÍZ GALLEGOS

c.c.p.- Archivo.

Agradecimientos

Esta tesis debe su existencia a la ayuda, apoyo e inspiración de muchas personas. En primer lugar, me gustaría expresar mi sincero agradecimiento y gratitud a mi tutor, Dr. Hermilo Sánchez Cruz , por su dedicación, apoyo, aliento, orientación, paciencia y conocimiento. Muchas gracias por darme la oportunidad de trabajar con usted, de tantas horas de ardua investigación donde me trasmitió tantos conocimientos y consejos en un entorno de trabajo óptimo, por su ayuda en la redacción de la tesis y de dos artículos de investigación en congresos internacionales. Al Dr. Hiram Habid López Valdez por su apoyo, comentarios y sugerencias durante la elaboración de esta tesis. El apoyo financiero proporcionado por el CONACYT. Agradezco a la Universidad Autónoma de Aguascalientes por permitirme ser parte de este programa.

Dedico esta tesis a mis padres y hermano por su fe inquebrantable en mí. Sin su aliento, apoyo, paciencia y estímulo durante todo el período de estudio, este trabajo no hubiera sido posible. Una dedicatoria especial a la memoria de mi hermana que donde quiera que se encuentre quiero seguirle demostrando mi cariño, que en todo momento de la realización de este trabajo estuvieron en mis pensamientos sus palabras de aliento que me dió en vida.

Resumen

En este manuscrito, se proponen nuevos métodos de visión computacional basados en códigos de cadena para disminuir la cantidad de información necesaria para representar la superficie de objetos bidimensionales y tridimensionales. El primer método codifica con el código de cadena F26 las superficies de objetos voxelizados y que no son isomorfos al plano, empleando trayectorias helicoidales por capas y usando el algoritmo A* para obtener el camino más corto para continuar con la trayectoria en la capa subsecuente, sin la modificación de las propiedades topológicas y geométricas del objeto, además de obtener cadenas con longitudes menores a los modelos actuales. El segundo método consiste en detectar puntos dominantes sin realizar ningún análisis explícito de los cambios de curvatura del contorno de un objeto bidimensional, con el cuál se obtiene una mejor aproximación poligonal que los reportados en la literatura, y se propone un nuevo criterio de evaluación para el enfoque poligonal. El tercer método obtiene la medida de disimilitud de Hausdorff entre dos cúmulos de puntos dominantes obtenidos mediante el método anterior usando las n-ésimas capas de los objetos tridimensionales, para obtener esta medida se debe de realizar una traslación y rotación rígida de los ejes principales de un objeto con respecto a otro.

Contenidos

Lista de Figuras

Lista de Tablas

1 Introducción

La visión es uno de los sentidos más importantes que poseen muchos organismos, porque es la ventana al mundo. La visión es la encargada de interpretar el entorno a partir de la luz que es recibida a través de los ojos. Se basa en la traducción de una señal física la cual es la luz emitida o reflejada por loa objetos que nos rodean, esta es captada por los ojos en forma de onda electromagnética, y es transformada por nuestras neuronas en una señal eléctrica, lo que deriva en una proyección de imagen que nuestro cerebro entiende.

El proceso visual humano es de gran interés para la comunidad científica y es estudiado de manera interdisciplinaria. La visión computacional es el estudio de los procesos que realiza el sistema visual con el fin de comprender su funcionamiento y así ser capaz de reproducirlo mediante tćnicas computacionales por medio de máquinas con capacidades similares.

Un área muy ligada a la de visión computacional es la de procesamiento de imágenes. El objetivo de estas áreas es diferente, el procesamiento de imágenes busca mejorar la calidad de las imágenes para su posterior utilización o interpretación. Dado que las imágenes son digitales, es necesario emplear la geometría digital la cuál es una de las áreas de investigación fundamentales en el procesamiento de este tipo de información.

La geometría digital se centra en las propiedades y aplicaciones en el espacio digital o de malla [1–3]. Fue creado debido al desarrollo del procesamiento de imágenes, la visión por computadora y los gráficos por computadora [4–6]. Un buen ejemplo es cómo dibujar una línea digital.

La geometría digital también se ocupa del modelado y reconocimiento de curvas y superficies que se almacenan en las computadoras, se muestrea la imagen, esto es dividirla en regiones cuadradas pequeñas. A este proceso se le conoce como digitalización. A cada una de las pequeñas regiones se les denomina elemento de pixel o pixel. Un pixel es la combinación de una pequeña región y su valor. El término del elemento más pequeño que constituye a un objeto 3D es el voxel, que es la abreviatura de *elemento de volumen*. En ocasiones se le llama al voxel como pixel 3D. En otras palabras, estos pixeles se organizan como matrices 2D o 3D. Estas matrices si se pueden representar digitalmente por medio de las computadoras.

Hoy en día, la representación y el reconocimiento de objetos 3D es un campo muy activo en la visión por computadora. Esto se debe tanto a los aumentos significativos en la potencia de cómputo disponibles en unidades cada vez más pequeñas y de bajo costo. Existe una gran cantidad de aplicaciones que re-

quieren imágenes en 3D para resolver problemas de la vida real, como las imágenes médicas [7, 8], donde las imágenes en 3D desempeñan un papel importante en el apoyo a los expertos para proporcionar diagnósticos más precisos. También existen aplicaciones en la preservación del patrimonio cultural, juegos, construcción mecánica, seguridad y vigilancia, diseño asistido por computadora (es decir, sistemas CAD) y, en general, en visión computacional y reconocimiento de patrones.

Para la representación y el reconocimiento de objetos 3D en el presente trabajo no es necesario procesar todos los descriptores (como el color y la textura) que poseen los objetos, nos centraremos en el descriptor de la forma del objeto, debido a que está fuertemente vinculado a la funcionalidad o identidad del objeto. En la literatura los códigos de cadena, se han empleado como descriptores de forma, estos representan movimientos a través del contorno del objeto mediante una secuencia de segmentos, conectados consecutivamente, de longitud y orientación específica, que conectan celdas adyacentes. La forma en que se visita el contorno del objeto y cuales son los movimientos que pueden ser producidos dan como resultado diferentes c óodigos, estos han sido explotados para la representacioón y compresión, los alcances que se pueden tener con los códigos han atraído la atención de muchos investigadores [9–12].

Para el caso tridimensional, existen importantes propuestas para el uso y la explotación de los códigos de cadena, pero estos no son expltados tanto como los de dos dimensiones, siendo actualmente un campo muy fértil [13–15]. En la literatura, la codificación de objetos tridimensionales de superficies que son isomorfas al plano se presenta [16], sin embargo, en este trabajo abordamos el problema de las superficies codificadas que no son isomorfas al plano, teniendo en cuenta las diferentes geometrías que se presentan, para resolver el problema de encontrar la ruta más corta que permita codificar de manera óptima la transición de una capa a otra, del objeto 3D.

Otra manera de representar el contorno de un objeto 2D o 3D es mediante la aproximación lineal o la aproximación poligonal [17–22]. Esta representación es otro método de codificación, en el cual los puntos de datos (también llamados puntos dominantes) se reducen considerablemente. Attneave [23] encontró que algunos puntos característicos son determinantes en el reconocimiento de la forma. De acuerdo con el trabajo de Atteave, cuando los puntos mencionados se unen mediante líneas rectas, la forma resultante debe ser muy similar a la original. Por supuesto, es inevitable que se pierdan datos, pero esta pérdida es soportable siempre que la información de las principales características de la forma y su topología original no se vean afectadas en lo más mínimo.

Otra de las tareas más importantes en visión computacional es la recuperación de la forma contenida en una base de datos, para lograr esto se han desarrollado motores de búsqueda que hacen una consulta por similitud de contenido. El elemento fundamental de un sistema de recuperación es la coincidencia de formas, que es el proceso de determinar qué tan similares son las dos formas [24]. Los modelos 3D no se recuperan

fácilmente como documentos de texto, pero los métodos de recuperación de formas 3D basados en contenido que utilizan las propiedades de forma de los modelos 3D para buscar modelos similares generalmente tienen un mejor desempeño que los métodos basados en texto [25].

El método de reconocimiento de objetos basado en la forma puede ser el método más completo de reconocimiento de patrones visuales. Esto se debe a que la forma, o el contorno, de un objeto dado es generalmente estable e invariante. Por lo general, la aplicación de este esquema requiere tres pasos. El primero es la detección de contornos, el segundo es la representación de contornos o formas, y el tercero es la comparación de formas. Sin embargo, las enormes dificultades involucradas en la reducción de ruido, la organización de pixeles, la descripción de la forma y la invariabilidad de la postura impiden que este método sea ampliamente utilizado.

Para la comparación de formas se emplea la coincidencia, este es el proceso de determinar qué tan similares son dos formas. A menudo se obtiene calculando una distancia. La recuperación es el proceso de búsqueda y entrega de los resultados de la consulta.

Recientemente, muchos investigadores han estudiado el problema específico de la recuperación de formas en 3D basada en contenido. Además, se puede encontrar una gran cantidad de literatura en los campos relacionados con la visión por computadora, el reconocimiento de objetos y el modelado geométrico. Trabajos de investigación en esta area han sido proporcionados por Besl y Jain [26], Loncaric [27] y Campbell y Flynn [28]. Para una descripción general de los métodos de emparejamiento de formas 2D podemos consultar el artículo de Veltkamp [29]. Desafortunadamente, la mayoría de los métodos 2D no generalizan directamente a la comparación de modelos 3D. El trabajo de Iyer et al. [30] proporciona una amplia visión general de las técnicas de búsqueda de formas en 3D.

1.1 Estructura de la tesis

- Capítulo 2. **Preliminares**.- En este capítulo se explican los diferentes fundamentos de temas teóricos que fueron utilizados a lo largo del presente trabajo, por ejemplo lenguajes formales, códigos de cadena, puntos dominantes, geometría digital.

- Capítulo 3. **Nuevo método de Codificación en 3D**.- En esta sección se describe el método para encadenar por capas la superficie de un objeto 3D, empleando el código de cadena F26 y el algoritmo A* para obtener la ruta más corta entre capas subsecuentes.

- Capítulo 4. **Aproximación poligonal usando un método de resolución múltiple y una gramática libre de contexto**.- En este apartado se describe el método para la detección de puntos dominantes

inmersos en la cadena AF8 del contorno de un objeto 2D y se propone un nuevo criterio de error para evaluar los diferentes métodos que existen en la literatura.

- Capítulo 5. **Aproximación poligonal usando una gramática libre de contexto en objetos 3D**.- En este capítulo desarrollamos un método para comparar dos cúmulos de puntos en 3D normalizando la posición de uno con respecto a otro. Estos cúmulos están constituidos por puntos dominantes obtenidos mediante el método descrito en el capítulo anterior.

- Capítulo 6. **Conclusiones**.- Se explican los resultados de los capítulos 3,4 y 5, así como su relevancia, además de trabajos futuros de cada una de las investigaciones desarrolladas en estos capítulos. Por último una conclusión general de la tesis.

2 Preliminares

En este capítulo damos a conocer los conceptos más importantes utilizados a lo largo de este trabajo, también explicamos en qué consisten y como obtener los diferentes códigos de cadena para 2D y 3D, y una explicación de los momentos centrales.

2.1 Definiciones y conceptos

Definición 2.1.1. Una *medida de disimilitud* se puede formalizar mediante una función definida en pares de descriptores que indican el grado de su parecido. Hablando formalmente, una medida de disimilitud d en un conjunto S es una función de valor no negativo $d : S \times S \to \mathbb{R}^+ \cup \{0\}$. La función d debe ser alguna de las siguientes propiedades [31]:

i. Identidad: Para todo $x \in S, d(x, x) = 0$.

ii. Positividad: Para todo $x \neq y \in S, d(x, y) > 0$.

iii. Simetría: Para todo $x, y \in S, d(x, y) = d(y, x)$.

iv. Desigualdad triangular: Para todo $x, y, z \in S, d(x, z) \leq d(x, y) + d(y, z)$.

v. Invariancia de transformación: para un grupo de transformación elegido G, Para todo $x, y \in S, g \in G, d(g(x), g(y)) = d(x, y)$.

Definición 2.1.2. La *distancia Euclidiana* (d_e) entre dos puntos $p = (p_1, ..., p_T)$ y $q = (q_1, ..., q_T)$ está definida por [32]

$$d(p, q) = \sqrt{\sum_{t=1}^{T} (p_t - q_t)^2}.$$

Definición 2.1.3. La distancia dirigida de Hausdorff $\check{H}$ entre dos conjuntos de puntos $\mathcal{A}$ y $\mathcal{B}$ es la máxima de las distancias entre cada punto $p \in \mathcal{A}$ hasta su vecino más cercano $q \in \mathcal{B}$. Eso es

$$\check{H}(\mathcal{A}, \mathcal{B}) = max_{p \in \mathcal{A}} \{min_{q \in \mathcal{B}} \{||p, q||\}\},$$

donde $||\cdot||$ es cualquier norma, por ejemplo, la función de distancia euclidiana. Tener en cuenta que $\check{H}(\mathcal{A},\mathcal{B}) \neq \check{H}(\mathcal{B},\mathcal{A})$ y, por lo tanto, la distancia dirigida de Hausdorff no es simétrica. La distancia H de Hausdorff es la máxima de las distancias dirigidas de Hausdorff en ambas direcciones y, por lo tanto, es simétrica. H es dada por [33]

$$H(\mathcal{A},\mathcal{B}) = max\{\check{H}(\mathcal{A},\mathcal{B}), \check{H}(\mathcal{B},\mathcal{A})\}.$$

Definición 2.1.4. Un subconjunto $\mathcal{S}$ del plano se llama *convexo* si y solo si para cualquier par de puntos $p, q \in S$ el segmento de línea $\overline{pq}$ está completamente contenido en $\mathcal{S}$ [34].

Definición 2.1.5. Un polígono puede ser formado uniendo todos los puntos de muestreo del contorno a su vez. Extraer tres puntos de muestreo adyacentes a lo largo de la dirección de las agujas del reloj del borde. Supongamos que estos tres puntos son $p_1(x_1, y_1), p_2(x_2, y_2), p_3(x_3, y_3)$. La línea $\overline{p_1p_2}$ es un borde del polígono y la línea $\overline{p_1p_3}$ es una línea diagonal del polígono. Si la línea $\overline{p_1p_3}$ se ubica fuera del polígono, el punto p_2 es un punto cóncavo local, y el ángulo entre la línea $\overline{p_1p_2}$ y la línea $\overline{p_1p_3}$ varía de 0° a 180°, como se muestra en la Fig. 2.1. A este tipo de polígonos se les conoce como *polígonos concavos* [35].

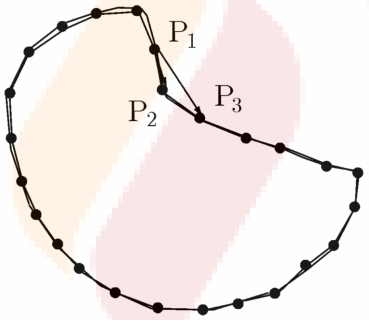


Fig. 2.1: Polígono concavo

Una malla 2D $m \times n$ es una matriz rectangular de puntos.

$$\mathcal{G}_{m,n} = \{(i,j) \in \mathbb{Z}^2 | 1 \leq i \leq m \wedge 1 \leq j \leq n\}.$$

Definición 2.1.6. Una malla rectilínea n-dimensional está representada por $\mathbb{Z}_+^n$, donde $\mathbb{Z}_+$ es el conjunto de enteros positivos. Un elemento de $\mathbb{Z}_+^n$ se abreviará como un *elemento espacial* o *spel*, por ejemplo, un pixel para n = 2 y un voxel para n = 3 [36].

Definición 2.1.7. Un *pixel*, ϱ, es una celda de resolución de una malla 2D con coordenadas cartesianas c(x,y) y un valor de intensidad de la imagen o un vector de valores de imagen asociado con la posición espacial.

Una imagen binaria es una imagen la cual contiene pixeles que toman solamente el valor 0 o 1.

Definición 2.1.8. Se dice que un par de pixeles vecinos están conectados en 4, si tienen un lado en común [2]

$$N_4(\varrho) = \{(i,j),(i+1,j),(i-1,j),(i,j+1),(i,j-1)\},$$

y se dice que un par de pixeles vecinos están conectado en 8, si tienen un lado o una esquina en común

$$N_8(\varrho) = N_4(\varrho) \cup \{(i+1,j+1),(i+1,j-1),(i-1,j+1),(i-1,j-1)\}.$$

Definición 2.1.9. Consideramos la digitalización de los rayos [2]

$$\gamma_{\alpha,\beta} = \{(x, \alpha x + \beta) : 0 \leq x < +\infty\},$$

en el conjunto $\mathbb{N}^2 = \{(i,j) : i,j \in \mathbb{N}\}$ en todos los puntos de la malla con coordenadas enteras no negativas en el plano. Como simplificación , suponemos que $0 \leq \alpha \leq 1$; esto es posible debido a la simetría de la malla. Tal rayo genera una secuencia de puntos de intersección $\rho 0, \rho 1, \rho 2, ...$ de $\gamma_{\alpha,\beta}$ con las líneas verticales de la malla en $w \geq 0$. Sea $(w, \mathfrak{L}_w) \in \mathbb{Z}^2$ el punto de la malla más cercano a ρw. (Si hay dos puntos de la malla más cercanos, tomamos el superior). La función de piso $\lfloor . \rfloor$ específica el número entero más grande que no exceda un valor real dado. Formalmente,

$$\mathfrak{L}_{\alpha,\beta}(w) = \{(w, \mathfrak{L}_w) : w \geq 0 \wedge \mathfrak{L}_w = \lfloor \alpha w + \beta + 0.5 \rfloor\},$$

y $i_{\alpha,\beta} = i_{\alpha,\beta}(0) i_{\alpha,\beta}(1) i_{\alpha,\beta}(2)..$ es un *rayo digital* con pendiente α e intersección β, donde las diferencias entre los sucesivos $\mathfrak{L}_w$ son definidos los códigos de cadena:

$$\mathfrak{L}_{\alpha,\beta}(w) = \mathfrak{L}_{w+1} - \mathfrak{L}_w = \begin{cases} 0, & \text{si } \mathfrak{L}_w = \mathfrak{L}_{w+1} \\ 1, & \text{si } \mathfrak{L}_w = \mathfrak{L}_{w+1} - 1, \end{cases} \quad \text{para } w \geq 0.$$

Definición 2.1.10. La malla definida por $m \times n$ 2-celdas y la relación de adyacencia A_1 (A_0) es isomórfica a la malla definida por $m \times n$ puntos de malla y la relación de adyacencia N_4 (N_8). Cualquiera de estas mallas será denotada por $\mathcal{G}_{m,n}$.
(En general, sea R_1 una relación en un conjunto S_1 y R_2 una relación en un conjunto S_2. Las estructuras $[S_1, R_1]$ y $[S_2, R_2]$ se llaman *isomorfas* si existe un mapeo uno a uno f desde S_1 sobre S_2, tal que pR_1q si y solo si $f(p)R_2f(q)$ para todos los $p, q \in S_1$. La asignación se denomina *isomorfismo.*) [2]

Definición 2.1.11. Los *elementos estructurantes* son pequeños conjuntos en forma de matriz o una subimagen que se usa para interactuar con la imagen que se mejorará. Nos ayuda a definir algunas estructuras de

barrio arbitrarias. Los detalles precisos se pueden obtener eligiendo un elemento estructurador adecuado.

Definición 2.1.12. Un *voxel*, denotado por v ó $v(x,y,z)$, es una celda de resolución de una cuadrícula 3D con coordenadas cartesianas $v(x,y,z)$ y un valor de intensidad $I_v \in \{0,1\}$. Si $I_v = 0$, decimos que el voxel es 0-*voxel*; por el contrario, decimos que es un 1-*voxel* [37].

Definición 2.1.13. Un voxel, v_0 puede compartir sus seis caras, 12 bordes, sus caras y bordes, o todas sus caras, bordes y vértices, dependiendo de sus 6, 12, 18 o 26 -*vecindario*, respectivamente, como se define por los conjuntos dados a continuación.

$$N_6(v_0) = \{v | d_e(v_0, v) = 1\},$$

$$N_{12}(v_0) = \{v | d_e(v_0, v) = \sqrt{2}\},$$

$$N_{18}(v_0) = \{v | d_e(v_0, v) \leq \sqrt{2}\},$$

$$N_{26}(v_0) = \{v | d_e(v_0, v) \leq \sqrt{3}\},$$

donde d_e es la distancia Euclidiana entre v_0 y su voxel vecino, v.

Definición 2.1.14. Un objeto 3D es un componente conectado compuesto por 1-voxeles, que está inmerso en una matriz 3D de N_X columnas, N_Y filas y N_Z capas, y cada capa está compuesta por cero o más regiones: $\mathcal{R}_0^s$, $\mathcal{R}_1^s$, $\cdots$ $\mathcal{R}_m^s$, donde s se refiere a la s-ima capa [2].

Definición 2.1.15. Una región $\mathcal{R}_k^s$, en un segmento dado, debe estar conectada, *es decir*, hay una ruta entre dos 1-voxeles contenidos en ella. Por lo tanto, dado un solo segmento, hay m componentes conectadas. Para nuestros propósitos, los contornos deben estar claramente definidos [2].

Definición 2.1.16. Un *voxel* del contorno de $\mathcal{R}$, es aquel 1-*voxel* que pertenece a $\mathcal{R}$ y es vecino a un 0-*voxel* [2].

Definición 2.1.17. Si c_1 son las coordenadas del voxel v_1, c_2 las coordenadas del voxel v_2, entonces v_1 está cerca de v_2 si y solo si $(c_2 - c_1) \in \mathcal{B} = \{(i,j,k)\}$, con $i,j,k \in \{-1,0,1\} \setminus \{(0,0,0)\}$. El conjunto $\mathcal{B}$ se llama *malla base* [2].

Definición 2.1.18. Un *camino* es una secuencia de voxeles ordenados adyacentemente, $\mathcal{P} = \{v_1, v_2, ..., v_w\}$, de modo que v_1 es adyacente a v_2, v_2 es adyacente a v_3, ..., v_{w-1} es adyacente a v_w. El conjunto de vectores

$\mathcal{P}_{\mathcal{B}} = \{b_1, b_2, ..., b_{w-1}\} \subset \mathcal{B}$ es llamado *camino base de* $\mathcal{P}$ [2].

Definición 2.1.19. El *centroide* de un volumen V (de densidad constante), ($\overline{x}$,$\overline{y}$,$\overline{z}$) es el centro de masa del volumen. Es la posición media (fila, columna, capa) para todos los voxeles en el volumen y es dado por [37]

$$\overline{x} = \frac{1}{\#V} \sum_{(x,y,z)\in V} x, \qquad \overline{y} = \frac{1}{\#V} \sum_{(x,y,z)\in V} y, \qquad \overline{z} = \frac{1}{\#V} \sum_{(x,y,z)\in V} z.$$

Definición 2.1.20. Una gramática libre de contexto es una 4-tupla (V, T, S, P) donde

i. V es un conjunto finito llamado las variables,

ii. T es un conjunto finito, separado de V, llamado los terminales,

iii. P es un conjunto finito de reglas, cada regla es una variable y una serie de variables y terminales,

iv. $S \in V$ es la variable de inicio.

Si u, v y w son cadenas de variables y terminales, y $A \to w$ es una regla de la gramática, decimos que uAv produce uwv, escrito $uAv \Longrightarrow uwv$.

2.2 Métodos de codificación en dos dimensiones

Los códigos de cadena son un conjunto de palabras formadas por un determinado alfabeto que se utilizan para representar la forma que posee un grupo de puntos, los cuáles constituyen una línea recta o una curva. Es por esto que son empleadas para representar el contorno cuando el pixel final es vecino del pixel inicial. Las técnicas de código de cadena se usan ampliamente porque preservan la información y permiten una reducciónn de datos considerable. Estos códigos se pueden ver como una secuencia conectada de segmentos de línea recta con longitudes y direcciones específicas.

2.2.1 Código F8

Freeman [39] creo el primer código de cadena para representar curvas digitales, y sigue siendo la técnica de codificación más utilizada. Este código se mueve a lo largo de una curva digital o una secuencia de pixeles del contorno de una forma, además esta basado en N_8. La dirección de cada movimiento se codifica utilizando un alfabeto $\{l \quad | \quad l = 0, 1, 2, \ldots, 7\}$ que indica un ángulo de $45° \times l$ en sentido de las manecillas del reloj desde el eje x positivo, esto se muestra en la Fig. 2.2. Llamado FDCCE, también es conocido de una forma abreviada como F8. Un ejemplo de como se codifica el contorno de una imagen mediante este código de cadena se muestra en la Fig. 2.3.

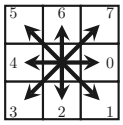


Fig. 2.2: Símbolos de Freeman obtenidos en sentido horario del código F8.

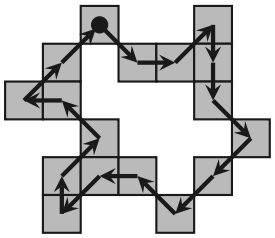


Fig. 2.3: La Figura es recorrida a través de los centros de los píxeles.
$C_{F8} = \{1, 0, 7, 2, 2, 1, 3, 3, 5, 4, 3, 6, 7, 5, 4, 7, 7\}$.

2.2.2 Código F4

Es una versión de cuatro direcciones del código de cadena de Freeman. Originalmente se nombró FFDCC, también es conocido como F4. Se mueve en cuatro direcciones con un alfabeto $\{l \mid l = 0, 1, 2, 3\}$ que denota un ángulo de $90° \times l$ en sentido de las manecillas del reloj desde el eje x positivo, como se muestra en la Fig. 2.4. Un ejemplo de como recorrer el contorno mediante este código se muestra en la Fig. 2.5

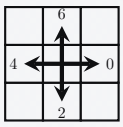


Fig. 2.4: Símbolos de Freeman obtenidos en sentido horario del código F4.

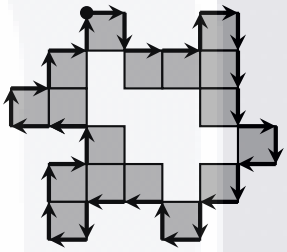


Fig. 2.5: La figura es recorrida por las aristas de los pixeles.
$C_{F4} = \{0, 2, 0, 0, 6, 0, 2, 2, 2, 0, 2, 4, 2, 4, 2, 4, 6, 4, 4, 2, 4, 6, 6, 0, 6, 4, 4, 6, 0, 6, 0, 6\}$.

2.2.3 Código VCC

Bribiesca [40] propone un código de cadena para la representación del contorno de formas llamado VCC(Vertex Chain Code). Se basa en los códigos llamados números de forma [41]. [en 1980] Sánchez-Cruz y López-Valdez [42] demuestran que los símbolos del código VCC se obtienen de analizar dos vectores Fig. 2.6. Si el segundo vector cambia 90 grados a la derecha con respecto al primero, ponemos 1. Si el segundo vector tiene la misma dirección que el primero, ponemos 2. Si el segundo vector cambia 90 grados a la izquierda con respecto al primero, ponemos 3. Los derivados de VCC son E_{VCC}, V_{VCC} y C_{VCC} [12]. El E_{VCC} se obtuvo considerando que el VCC usa dos bits para representar tres símbolos, V_{VCC} surge al considerar una longitud variable de la cadena VCC, y C_{VCC} se basa en la aplicación del algoritmo Huffman en los símbolos del código

VCC. Un ejemplo de como se codifica el contorno de una imagen empleando el código VCC se muestra en la Fig. 2.7.

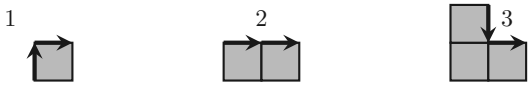


Fig. 2.6: Símbolos del código VCC obtenidos mediante la representación de vectores.

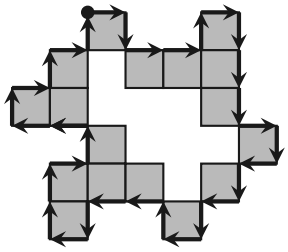


Fig. 2.7: La figura es recorrida por las aristas de los píxeles.C_{VCC} = { 1,1,3,2,3,1,1,2,2,3,1,1,3,1,3,1,1,3,2,3, 1,1,2,1,3,3,2,1,1,3,1,3}.

2.2.4 Código 3OT

Propone Bribiesca [14] un código de cadena para representar curvas 3D, este código solo considera los cambios de dirección ortogonal relativa, y consiste de 5 símbolos, es conocido como 5OT. Establecen Sánchez y Rodríguez [9] un código de cadena en 2D llamado 3OT, esta inspirado en el 5OT de Bribiesca [14], tiene un alfabeto de 3 símbolos Fig. 2.8, ya que solo considera cambios de dirección ortogonales. Esto permite conservar la forma original del objeto binario.

Dado el conjunto $C = \{0, 1, 2\}$, se puede representar cualquier región binaria cerrada por una curva discreta.

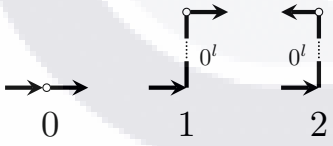


Fig. 2.8: Símbolos del código 3OT

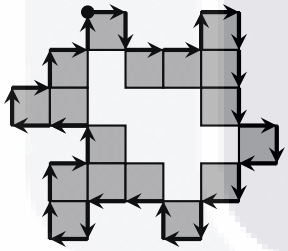


Fig. 2.9: La figura es recorrida por las aristas de los píxeles.
$C_{3OT} = \{2, 1, 0, 2, 1, 2, 0, 0, 1, 1, 2, 1, 1, 1, 1, 2, 1, 0, 2, 1, 2, 0, 2, 1, 2, 0, 1, 2, 1, 1, 1, 1\}$.

El elemento 0 representa que "sigue recto" a través de los segmentos contiguos de línea recta siguiendo la dirección del último segmento. El 1 indica un cambio en el mismo sentido que el vector de referencia. Mientras que 2 significa "regresar" con respecto al vector de referencia . Donde l es cero o un entero positivo, su valor depende del número de pasos dados en una misma dirección. Un ejemplo de como se emplea el código 3OT se muestra en la Fig. 2.9. Se propuso M3OT [43] considerando los símbolos de agrupación del 3OT.

2.2.5 Código AF8

Kui y Žalik [10] proponen un código de cadena donde cada elemento de la cadena está codificado como una diferencia de ángulo relativa entre él y el elemento anterior. Tal código es llamado DFCCE aunque también es conocido como AF8. La Fig. 2.10 muestra los 8 símbolos que componen a este código. El código MDF9 [44] se propuso considerando los patrones de AF8 en piezas de líneas rectas discretas de las formas de contorno. La Fig. 2.11 ejemplifica el uso del código AF8. Este código de cadena es codificado con los símbolos de 'a' hasta 'h' a lo largo de este trabajo para evitar confusión con exponentes numéricos.

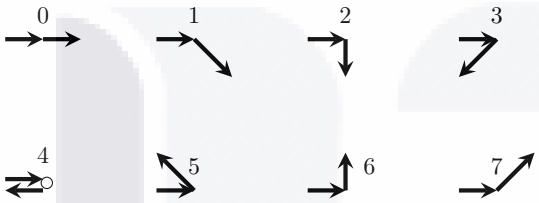


Fig. 2.10: Símbolos del código AF8

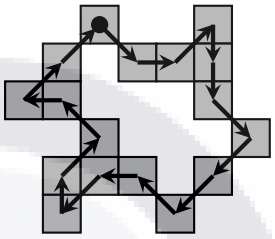


Fig. 2.11: La Figura es recorrida a través de los centros de los píxeles.
C_{AF8}= { 7,7,3,0,7,2,0,2,7,7,5,1,6,7,3,0,2}.

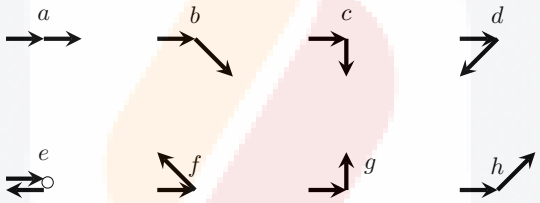


Fig. 2.12: Símbolos del código AF8

2.2.6 Código AAF8

Sánchez-Cruz y López [42] proponen un código de cadena usando una matriz. Este código está inspirado en 3OT, con la diferencia de que este se obtiene al visitar los centros de los pixeles. Lo llaman AAF8 (porque usa dos ángulos para determinar un símbolo). Un ejemplo del uso de este código se da en la Fig. 2.14. A continuación mencionamos algunas características que los códigos F4, 3OT, VCC, F8, AF8 y AAF8 tienen en común.

Nota 1 F4, VCC, 3OT, F8, AF8 y AAF8 tienen las siguientes características comunes:

1. Los pixeles de contorno se visitan en el sentido de las manecillas del reloj.
2. Por simplicidad, el primer pixel se visita en la parte superior izquierda de la figura.

Nota 2 F4, VCC y 3OT tienen las siguientes características en común (ver Fig. **??**):

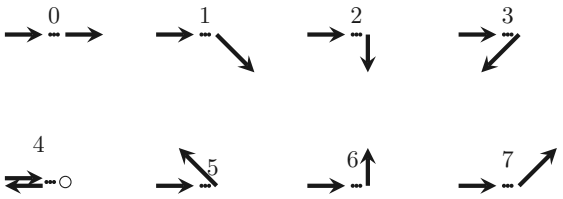


Fig. 2.13: Símbolos del código AAF8

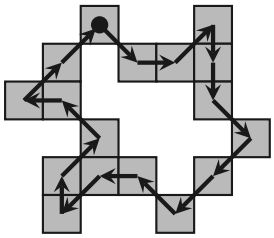


Fig. 2.14: La Figura es recorrida a través de los centros de los píxeles.
$C_{AAF8} = \{1, 6, 2, 8, 2, 1, 8, 4, 1, 6, 2, 4, 7, 5, 2, 8, 5\}$.

1. Se visitan las aristas de los pixeles de contorno.
2. El primer vértice visitado se encuentra en la parte más a la izquierda de la parte superior del primer pixel.
3. Las primeras dos aristas del contorno del primer pixel visitado corresponden a los símbolos 3 y 0 del código F4.
4. El primer cambio de una curva discreta corresponde al símbolo "2" del código 3OT.
5. Los códigos de cadena tienen la misma longitud.

Nota 3 F8, AF8 y AAF8 tienen las siguientes características en comunes (ver Fig. **??**):

1. Se visitan los centros de los pixeles de contorno.
2. El primer centro visitado es el del primer pixel.
3. Los códigos de cadena tienen la misma longitud.

Nota 4 Si F4 o F8 necesita ser codificado / decodificado, vector por vector tiene que ser analizado.

Nota 5 Si es necesario codificar / decodificar VCC o AF8, se deben analizar cada dos vectores consecutivos.

Nota 6 Si 3OT o AAF8 necesita ser codificado / decodificado, cada tres vectores consecutivos deben ser analizados.

Nota 7 Como es usual, los códigos F8 y F4 se usan siguiendo la dirección de las manecillas del reloj (ver Fig. 2.2 y 2.4 respectivamente).

2.3 Métodos de codificación en tres dimensiones

2.3.1 Código F26

Damos símbolos a cada uno de los elementos de la malla base $\mathcal{B}$ de la siguiente manera:

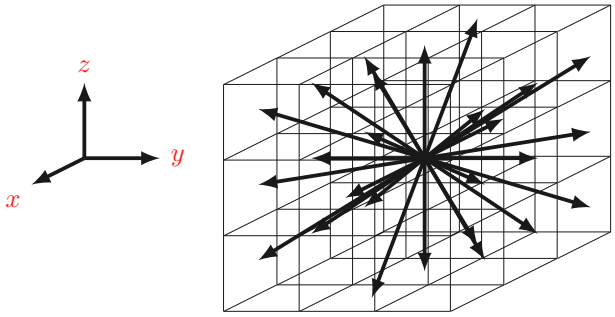


Fig. 2.15: Las 26 direcciones del código Freeman.

$$\underbrace{(1,0,0)}_{\mathbf{a}},\underbrace{(1,1,0)}_{\mathbf{b}},\underbrace{(0,1,0)}_{\mathbf{c}},\underbrace{(-1,1,0)}_{\mathbf{d}},\underbrace{(-1,0,0)}_{\mathbf{e}},\underbrace{(-1,-1,0)}_{\mathbf{f}},\underbrace{(0,-1,0)}_{\mathbf{g}},\underbrace{(1,-1,0)}_{\mathbf{h}},\underbrace{(1,0,1)}_{\mathbf{i}},\underbrace{(1,1,1)}_{\mathbf{j}},\underbrace{(0,1,1)}_{\mathbf{k}},$$

$$\underbrace{(-1,1,1)}_{\mathbf{l}},\underbrace{(-1,0,1)}_{\mathbf{m}},\underbrace{(-1,-1,1)}_{\mathbf{n}},\underbrace{(0,-1,1)}_{\mathbf{o}},\underbrace{(1,-1,1)}_{\mathbf{p}},\underbrace{(1,0,-1)}_{\mathbf{q}},\underbrace{(1,1,-1)}_{\mathbf{r}},\underbrace{(0,1,-1)}_{\mathbf{s}},\underbrace{(-1,1,-1)}_{\mathbf{t}},\underbrace{(-1,0,-1)}_{\mathbf{u}},$$

$$\underbrace{(-1,-1,-1)}_{\mathbf{v}},\underbrace{(0,-1,-1)}_{\mathbf{w}},\underbrace{(1,-1,-1)}_{\mathbf{x}},\underbrace{(0,0,1)}_{\mathbf{y}},\underbrace{(0,0,-1)}_{\mathbf{z}},$$

Entonces, el alfabeto que usamos es F26 = **{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}**, y la codificación es obtenida cuando cada vector b_k (*es decir,* la diferencia de coordenadas de dos voxeles adyacentes) es tomado de $P_{\mathcal{B}} = \{b_1, b_2, ..., b_{n-1}\}$ asociando su respectivo símbolo en F26. Esto es lo que llamamos codificación F26 y esta codificación fué creada por Freeman en 1974 [13]. La Fig. 2.16 muestra un ejemplo de como se obtiene el código de cadena F26, el cuál es: **jagjaaajllm**.

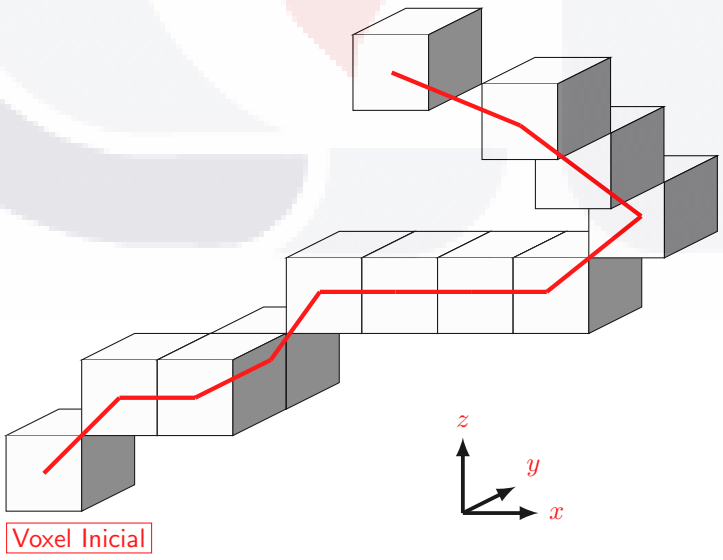


Fig. 2.16: Ejemplo del código de cadena F26 y su código asociado es: **jagjaaajllm**.

2.3.2 Código 5OT

Bribiesca [14] propone un código de cadena llamado ODCCC (Orthogonal Direction Change Chain Code) o también conocido como 5OT, el cuál representa curvas discretas 3D. Define como un cambio de dirección la relación que tienen dos segmentos contiguos de línea recta y un elemento de la cadena es definido por dos cambios de dirección. Solo hay cinco posibles cambios de dirección ortogonal para representar cualquier curva discreta 3D. Los símbolos se muestran en la Fig. 2.17 y en la Fig. 2.18 como se recorren las aristas de los voxeles. Cada dirección está compuesta por tres vectores: un vector de referencia (ref), un vector de soporte (sop) y un vector de cambio de dirección (cam). Un ejemplo que ilustra como se usa este código de cadena se muestra en la Fig. 2.19, su código de cadena asociado es: $\{\widehat{c}\,\widehat{a}\,\widehat{c}\,\widehat{d}\,\widehat{b}\,\widehat{a}\,\widehat{a}\,\widehat{a}\,\widehat{d}\,\widehat{c}\,\widehat{c}\,\widehat{b}\,\widehat{c}\,\widehat{c}\,\widehat{c}\,\widehat{b}\,\widehat{c}\,\widehat{d}\,\widehat{e}\,\widehat{b}\}$.

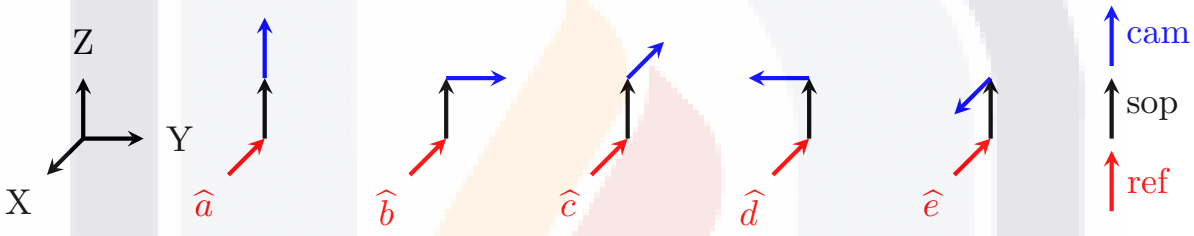


Fig. 2.17: Símbolos del código 5OT.

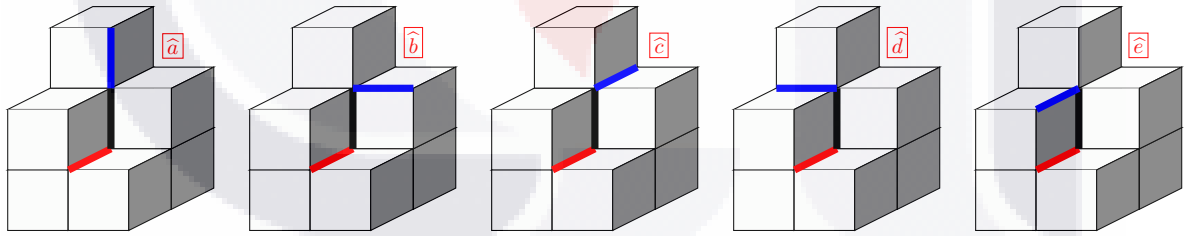


Fig. 2.18: Ejemplos de los símbolos de código de cadena 5OT cuando se recorren las 5 posibles direcciones.

2.3.3 Código 3DRC

Sánchez-Cruz, López-Valdez y Cuevas [15] proponen un nuevo código de cadena para representar curvas discretas en 3D llamado 3DRC. El método se basa en buscar cambios relativos en el espacio Euclidiano 3D, compuesto por tres vectores: un vector de referencia, un vector de soporte y un vector de cambio de dirección. Esta compuesto de un alfabeto de 25 símbolos para representar cualquier curva discreta de cara, borde o

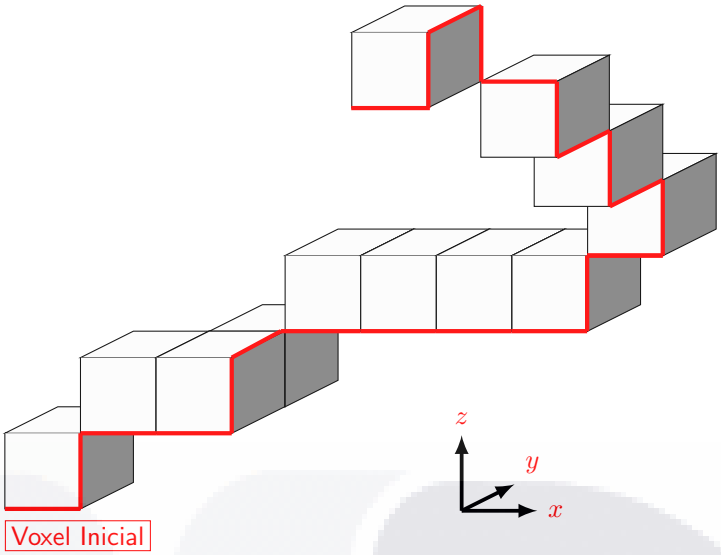


Fig. 2.19: Ejemplo del código de cadena 5OT y su código asociado es: $\{\widehat{c}\,\widehat{a}\,\widehat{c}\,\widehat{d}\,\widehat{b}\,\widehat{a}\,\widehat{a}\,\widehat{a}\,\widehat{d}\,\widehat{c}\,\widehat{c}\,\widehat{b}\,\widehat{c}\,\widehat{c}\,\widehat{c}\,\widehat{b}\,\widehat{c}\,\widehat{d}\,\widehat{e}\,\widehat{b}\}$.

vértice.

Se debe de tener en cuenta que un GEP arbitrario (Una ruta elemental generalizada es un conjunto ordenado de $m \geq 4$ voxeles) no siempre es igual a uno de los elementos de los grupos, pero al realizar tres rotaciones de vox y una rotación habitual, como máximo, obtenemos que cualquier GEP arbitrario es equivalente a uno de los elementos de los grupos. El siguiente resultado nos dice cómo obtener el elemento del grupo que es equivalente a un GEP arbitrario; También muestra qué símbolo corresponde a un GEP.

Lo que se busca con estas rotaciones es alinear los dos voxeles de enmedio de un GEP arbitrario.

Todo está ahora en el comportamiento del vector de cambio. Dependiendo de este vector, eligen el símbolo correspondiente. El símbolo asociado a cam en el Grupo 1, Grupo 2 o Grupo 3, es el símbolo correspondiente a la ruta elemental P. Introducen el símbolo $\widetilde{y}$ que se utiliza para etiquetar una ruta que no tiene cambios de dirección.

El ejemplo dado en la Fig. 2.21 muestra cómo asignar un símbolo a una ruta elemental. Como se puede ver, el símbolo de la ruta es $\widetilde{p}$.

La Fig. 2.22 muestra cuál es código de cadena 3DRC visitando los GEPs de la curva.

2.4 Invariantes

En esta sección estudiaremos algunas de las cantidades o popiedades de los objetos que son insencibles a algún grupo de transformaciones válidos en el espacio que ocupan. Entre las categorías más importantes que se emplean en el estudio de invariantes están las basadas en los momentos centrales, las basadas en las transformadas integrales [45].

Fig. 2.20: Símbolos del código 3DRC.

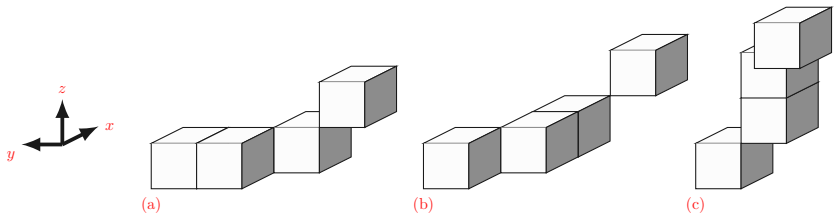


Fig. 2.21: Se debe realizar un conjunto de rotaciones discretas para obtener un símbolo en un arreglo de cuatro voxeles. Para ir de (a) a (b) se realiza una rotación de vox de 45 alrededor de z, y para ir de (b) a (c) se realiza rotación habitual de 90 alrededor de x.

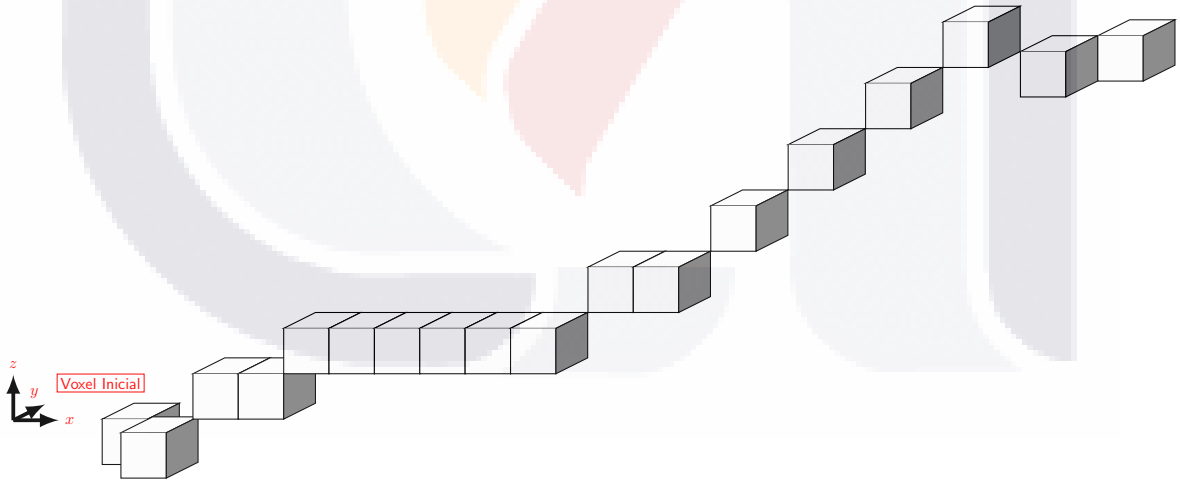


Fig. 2.22: Camino Simple: $P_B = \{\langle 1,1,0\rangle, \langle 1,1,1\rangle, \langle 1,0,0\rangle, \langle 1,0,1\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,1,1\rangle, \langle 1,0,0\rangle, \langle 1,1,1\rangle, \langle 1,1,1\rangle, \langle 1,1,1\rangle, \langle 1,1,1\rangle, \langle 1,1,0\rangle\}$ y su código asociado es: $\{1\widetilde{j}\ 8711\widetilde{j}\ \widetilde{i}\ \widetilde{y}\ \widetilde{y}\ \widetilde{y}\ \widetilde{y}\ \widetilde{p}\ \widetilde{i}\ \widetilde{i}\ \widetilde{y}\ \widetilde{y}\ \widetilde{y}\ \widetilde{j}\ \widetilde{i}\}$.

El inicio del estudio de los invariantes de momentos se remonta al siglo XlX, en el marco de la teoría de grupos y de la teoría de los invariantes algebraicos. La teoría de los invariantes algebraicos fue estudiada a detalle por famosos matemáticos alemanes [46] y se continuo con el desarrollo en el siglo XX en [47] y [48], entre otros. Los invariantes de momento se introdujeron por primera vez en temas de reconocimiento de patrones y procesamiento de imágenes en 1962 [49], cuando Hu empleó los resultados de la teoría de los invariantes algebraicos derivó sus siete invariantes famosos a la rotación de objetos 2D. Sea $\rho(x, y) \leq 0$ una función real y acotada, definida en una región R. Los momentos regulares bidimensionales de orden $(p+q)$ se definen como:

$$m_{pq} = \int_R x^p y^q \rho(x, y) dx dy, \tag{2.4.1}$$

donde R es la región (de la imagen) en la que se aplicará la doble integral y $p, q \in N$.

Los momentos centrales están dados por:

$$\mu_{pq} = \int_R (x - \xi)^p (y - \eta)^q \rho(x, y) dx dy, \tag{2.4.2}$$

donde $\xi = m_{10}/m_{00}$, $\eta = m_{10}/m_{00}$ y $p, q \in N$.

Las relaciones 2.4.1 y 2.4.2 pueden calcularse para el caso discreto, convirtiendo $\int_R$ en dos sumatorias: $\sum \sum_{x,y}$ considerando $\rho(x, y) = 1$ y la longitud del lado de cada *pixel* igual a uno.

Para el caso tridimensional, sólo hay que agregar una suma a los casos discretos de las ecuaciones de momentos regulares y centrales(eq.2.4.1 y eq.2.4.2). Los momentos regulares se definen como:

$$m_{pqr} = \sum_x \sum_y \sum_z x^p y^q z^r. \tag{2.4.3}$$

Y los momentos centrales:

$$m_{pqr} = \sum_x \sum_y \sum_z (x - \xi)^p (y - \eta)^q (z - \varsigma)^r. \tag{2.4.4}$$

2.4.1 Invariancia bajo traslación

Una *traslación de un objeto* es una transformación rígida en $\mathbb{R}^n$. Una traslación mueve cada *spel* del objeto en la misma dirección y por la misma cantidad [50], mantienendo constante las distancias entre cualquier par de spels del objeto.

En 2D los momentos centrales dados por la ecuación 2.4.2 son invariantes bajo traslación. Puede hacerse

una traslación de cada coordenada (x, y) del objeto a través de la siguiente ecuación:

$$\begin{aligned} x' &= x + \alpha \\ y' &= y + \beta \end{aligned} \tag{2.4.5}$$

donde α y β son constantes,

y la ecuación 2.4.2 sigue siendo la misma.

Para el caso 3D puede hacerse una traslación de las coordenadas (x, y, z) a través de la siguiente ecuación:

$$\begin{aligned} x' &= x + \alpha \\ y' &= y + \beta \\ y' &= y + \gamma \end{aligned} \tag{2.4.6}$$

donde α, β y γ son constantes, y la ecuación 2.4.4 se mantiene invariante.

2.4.2 Invariancia bajo rotación

Con el álgebra de invariantes, Hu [49] llega a los siete invariantes ante transformaciones de rotación:

$$\begin{aligned} &\phi_1 = \mu_{20} + \mu_{02}, \\ &\phi_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2, \\ &\phi_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2, \\ &\phi_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2, \\ &\phi_5 = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] + (3\mu_{21} - \mu_{03})(\mu_{21} - \mu_{03})[(\mu_{30} + \mu_{12})^2 \\ &- (\mu_{21} + \mu_{03})^2], \\ &\phi_6 = (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] + 4\mu_{11}^2(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03}) \\ &\phi_7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] - (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 \\ &- (\mu_{21} + \mu_{03})^2], \end{aligned} \tag{2.4.7}$$

Hu aplicó estas ecuaciones por primera vez para reconocer caracteres alfabéticos, sin importar cual es su posición, tamaño y orientación. Una de las limitaciones de este método sucede cuando la resolución de los objetos 2D aumenta, el número de invariantes debe de crecer, por ende si dos objetos tienen invariantes de segundo orden semejantes, las formas pueden ser muy diferentes. Una desventaja que presenta el método es

la sencibilidad al ruido. Sadjadi y Hall [51] utilizaron los momentos en 3D, al aumentar una integral a la ecuación 2.4.1. Encontraron tres invariantes de segundo orden (la suma de los subíndices de las ecuaciones, siempre es igual a dos):

$$\begin{aligned}\phi_1 &= \mu_{200} + \mu_{020} + \mu_{002},\\ \phi_2 &= \mu_{200}\mu_{020} + \mu_{200}\mu_{002} + \mu_{020}\mu_{002} - \mu_{101}^2 - \mu_{110}^2 - \mu_{011}^2,\\ \phi_3 &= \mu_{200}\mu_{020}\mu_{002} - \mu_{002}\mu_{110}^2 + 2\mu_{110}\mu_{101}\mu_{011} - \mu_{020}\mu_{101}^2 - \mu_{200}\mu_{011}^2,\end{aligned} \tag{2.4.8}$$

los cuales son invariantes en rotación. Lo y Don [52] utilizaron esta ecuación, así como los ejes principales para orientar objetos simétricos (cigarro, plato y esfera) y clasificarlos. Aunque, no probaron este método para reconocer objetos irregulares.

2.4.3 Los ejes principales

Los ejes principales han sido ampliamente usados para orientar objetos 2D. En el caso de los objetos 3D hay algunos trabajos, por ejemplo el de Faber y Stokel [53], para quienes calcular los ejes principales les permiten identificar objetos que son identicos excepto por una rotación y/o un factor de escalada homogéneo, dado que los ejes principales tienen un significado geómetrico y analítico mucho más claro que los momentos y las transformadas. Entre las ventajas, Faber y Stokel [53], concluyeron que el uso de los ejes principales, como invariantes, fueron robustos sin importar la resolución de los objetos. Entre las limitaciones que encontraron, observaron que para orientar los objetos 3D es necesario conocer factores de escala y deformación de los objetos, pues existe una ambigüedad en la dirección de los ejes cuando estos se calculan.

Una rotación es una transformación rígida en $\mathbb{R}^3$ llevada a cabo alrededor de un eje de rotación en la que, fisícamente, cada partícula de un objeto dado se mueve alrededor de un eje fijo, conservando siempre una distancia constante con él y en la que las distancias entre todas las partículas del objeto también permanecen constantes.

A contnuación se analiza el caso 3D, pudiéndose reducir los resultados a 2D. El momento angular L de n partículas repsecto al origen de un sistema de coordenadas dado es:

$$\bar{L} = \sum_{j=1}^{n} m_j(\bar{r}_j \times \bar{v}_j), \tag{2.4.9}$$

donde: m_j es la masa del j-ésimo *voxel*, $\bar{r}_j$ es el radio vector cuya magnitud es la distancia de una partícula al eje de rotación y $\bar{v}_j$ es la velocidad. En este caso los objetos se consideran como sólidos rígidos. La velocidad

puede escribirse como:

$$\bar{v}_j = \bar{w}_j \times \bar{r}_j, \tag{2.4.10}$$

donde $\bar{w}_j$ es la velocidad angular. Considerando $w_j = \delta_{ik} w_k$ donde $\delta_{ik} w_k$ es la delta de Kronecker (si $i = k$, entonces $\delta_{ik} w_k = 1$; si $i \neq k$, entonces $k = 0$). Las componentes del momento angular $\bar{L} = (L_1, L_2, L_3)$ son:

$$\bar{L} = w_k \sum_{j=1}^{n} m_j (\delta_{ik} x_l^{(j)} x_l^{(j)} - x_i^{(j)} x_k^{(j)}), \tag{2.4.11}$$

donde: $i, k = [1, 2, 3]$, hay una suma sobre l, mientras que j se refiere a la j-ésima partícula (en el presente trabajo las partículas se consideran como *voxeles* o *pixeles*). La suma de la ecuación 2.4.11 se define como el *tensor de momento-inercia* T_{ik}:

$$T_{ik} = \sum_{j=1}^{n} m_j (\delta_{ik} x_l^{(j)} x_l^{(j)} - x_i^{(j)} x_k^{(j)}), \tag{2.4.12}$$

Usando nueve combinaciones diferentes de esta cantidad (que forma un tendor de segundo orden [54]), la bien conocida expresión de *momentos regulares* para el caso discreto:

$$M_{ijk} = \sum x_1^p x_2^q x_3^r f(x, y, z), \tag{2.4.13}$$

y considerando $m_j = 1$ y $f(x, y, z) = 1$ (debido al hecho de que los objetos considerados son sólidos de densidad constante), se obtiene la *matriz de inercia* dada por:

$$T = \begin{pmatrix} M_{020} + M_{002} & -M_{110} & -M_{101} \\ -M_{110} & M_{002} + M_{200} & -M_{011} \\ -M_{101} & -M_{011} & M_{200} + M_{020} \end{pmatrix} \tag{2.4.14}$$

Si se forma el producto interno entre T_{ik} y cualquier otro vector: $T_{ik} A_k = B_i$, el nuevo vector B con componentes B_i es diferente a A_i. En general, la operación $T_{ik} A_k$ puede rotar y cambiar la magnitud de A. Si se quiere encontrar todos los vectores que no estén rotados debido al producto interno, entonces se tiene que resolver la ecuación $T_{ik} A_k = \lambda A_i$ donde λ es un escalar. Si los vectores arriba mencionados existen, estos constituyen los llamados *eigenvectores* o *ejes principales* del tensor T_{ik}. Para registrar u orientar los objetos

debidamente, en este trabajo utilizamos los ejes principales. La igualdad anterior puede escribirse como:

$$T_{ik}A_k - \lambda A_i = (T_{ik} - \lambda\delta_{ik})A_i = 0,$$

ó

$$\begin{aligned}(T_{11} - \lambda)A_1 + T_{12}A_2 + T_{13}A3 = 0\\ T_{21}A_1 + (T_{22} - \lambda)A_2 + T_{23}A3 = 0\\ T_{31}A_1 + T_{32}A_2 + (T_{33} - \lambda)A3 = 0.\end{aligned} \tag{2.4.15}$$

Este sistema tiene una solución no trivial, si y sólo si su determinante se anula. Cuando se resuelve esta ecuación, se obtiene un polinomio de tercer orden, con tres raíces: $\lambda_1, \lambda_2, \lambda_3$.

No obstante, para objetos irregulares y un eigenvector de tamaño dado, las tres λs pueden ser diferentes.

En general, la λ menor corresponde a la dirección más elongada del objeto y la λ mayor a la dirección donde el objeto se prolonga menos. Notoriamente, otros valores de λ corresponderán a situaciones intermedias. Por lo tanto, las λs dan cierto grado de información sobre la forma del objeto.

3 Nuevo método de Codificación en 3D

A continuación daremos a conocer una representación de un objeto 3D discreto que se describe mediante la codificación helicoidal usando el código de cadena F26, con el fin de tener una compensación entre la compresión y la pérdida de información al representar la forma de un objeto 3D.

3.1 Codificación helicoidal de la forma de un objeto 3D

La codificación helicoidal permite que los códigos de cadena eviten almacenar las coordenadas de inicio tanto como sea posible, mientras la región en la capa actual sea vecina de la región anterior, y exista un camino para ir de n-ésimo voxel $(v_n(s))$ al primer voxel no visitado de la capa siguiente $(v_1(s+1))$ donde s es la s-sima capa. Esto permite recuperar la forma del objeto sin necesidad de saber en qué coordenadas comenzamos a codificar en $v_1(s+1)$.

La codificación helicoidal se lleva a cabo de la siguiente manera:

1. Hacer $s \to 0$ y definir N_Z como el número total de capas de la malla 3D.
2. Mientras $s < N_Z + 1$
3. Visitar el primer voxel no visitado, $v_1(s)$, de la capa actual.Si no existe $s \to s+1$, ir a 2.
4. Codificar el contorno $\mathcal{R}_k^s$ y obtener $v_n(s)$.
5. Encontrar$(v_1(s+1))$. Si no existe $s \to 0$, ir a 3.
6. $s \to s+1$. Ir a 2.

La función Encontrar$(\cdot)$ introducida en el algoritmo anterior se implementa teniendo en cuenta los siguientes casos, que se generan por la geometría del objeto. La Fig. 3.1 muestra un objeto compuesto por dos capas, donde los voxeles oscuros en la capa superior representan el contorno de la región $\mathcal{R}_1^{s+1}$.

Caso 1. $v_1(s+1) \in N_{26}(v_n(s))$ (Fig. 3.1(a)).

Caso 2. Es posible dibujar una línea recta discreta desde $v_n(s)$ a $v_1(s+1)$ como el camino más corto (Fig. 3.1(b)).

Caso 3. No es posible dibujar una línea recta discreta desde $v_n(s)$ a $v_1(s+1)$, ya que hay 0-voxeles entre ellos, por esta razón el camino no se puede crear, a menos que se deba rodear la concavidad. Esto hace que el $v_1(s+1)$ original pueda cambiar, ya que en el i-mo paso, un voxel del contorno de $\mathcal{R}_{k'}^{s+1}$ tiene el menor d_e con respecto al último voxel del camino actual (Fig. 3.1(c)).

Caso 4. Hay dos candidatos $v_1(s+1)$, debido a que ambos tienen la misma distancia con respecto a $v_n(s)$ (Fig. 3.1(d)).

Debemos validar qué $\mathcal{R}_{k'}^{s+1}$ es un vecino de $\mathcal{R}_k^s$, seleccionando el $v_1(s+1)$ correcto utilizando A* para ir de $v_n(s)$ a $v_1(s+1)$. Para resolver estos casos, el algoritmo A* se detalla en la Sección 3.2.

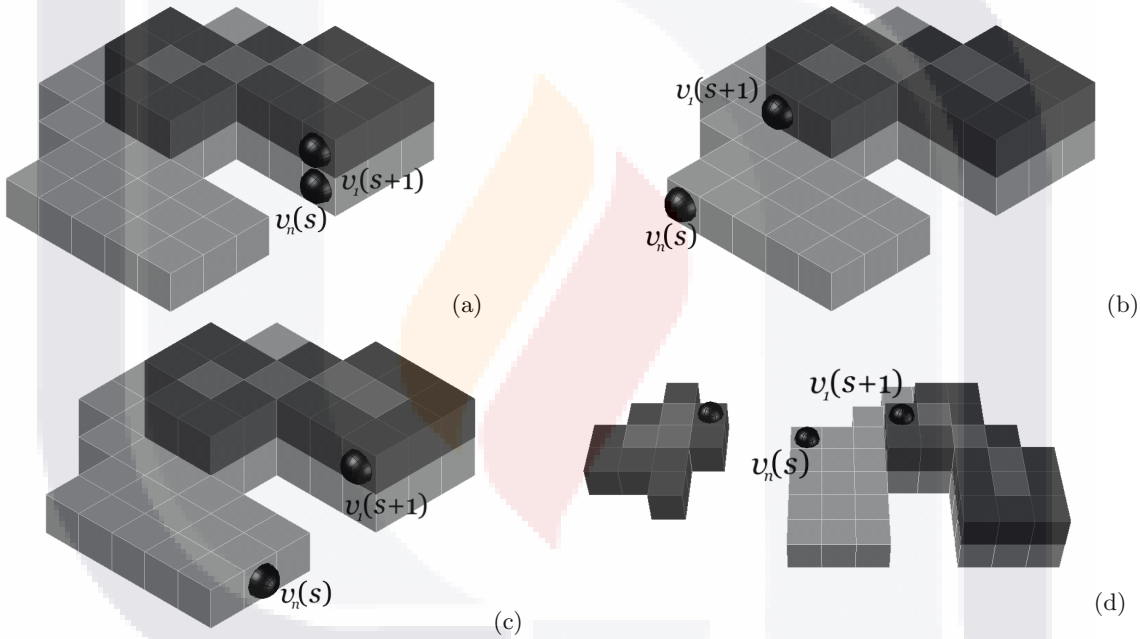


Fig 3.1: Cuatro casos diferentes a considerar para los puntos de inicio y destino debido a la geometría de los objetos.

3.1.1 Símbolos usados de F26

Para codificar nuestro objeto 3D, los 26 símbolos de F26 no son necesarios. Parte de la estrategia es visitar primero cada capa y codificar sus contornos, así podamos usar ocho vectores diferentes para cada región visitada, $\mathcal{R}_k^s$, ya que los vectores apuntan a una de las cuatro caras de cada voxel, más cuatro hacia los bordes. Entonces, los símbolos **{a,b,c,d,e,f,g,h}** son obligatorios. Por otro lado, una vez que se visita el contorno, el siguiente paso es pasar a la capa contigua, *por ejemplo*, de $\mathcal{R}_k^s$ a $\mathcal{R}_{k'}^{s+1}$, por lo tanto, necesitamos vectores

que apuntan a cualquiera de sus cuatro bordes o cuatro vértices, más un símbolo extra correspondiente a la cara superior. Entonces, se usan los símbolos **{i,j,k,l,m,n,o,p,q}**. Por lo tanto, se requieren 17 símbolos, como máximo.

3.2 Algoritmo A*

Algoritmo A* original

El algoritmo A* busca el camino más corto desde un punto incial hasta un punto objetivo. Este puede ser aplicado en el espacio de configuración métrico o topológico [55]. Esta heurística utiliza información relativa al lugar donde se encuentra el objetivo para seleccionar la siguiente dirección que se debe de seguir. La fórmula que se emplea para seleccionar el siguiente punto en el espacio de configuración es:

$$f(\mathrm{p}) = H(p) + G(p) \tag{3.2.1}$$

Donde p es el punto actual, $H(p)$ es la heurística de distancia (manhattan, euclidiana o chebyshev) desde p hasta el punto objetivo y $G(p)$ es el costo acumulado del movimiento de ir desde el estado inicial hasta el estado p. Cada punto adyacente del actual es evaluado mediante la fórmula $f(p)$. El punto con el valor más bajo de $f(p)$ es seleccionado como el siguiente en la secuencia [56].

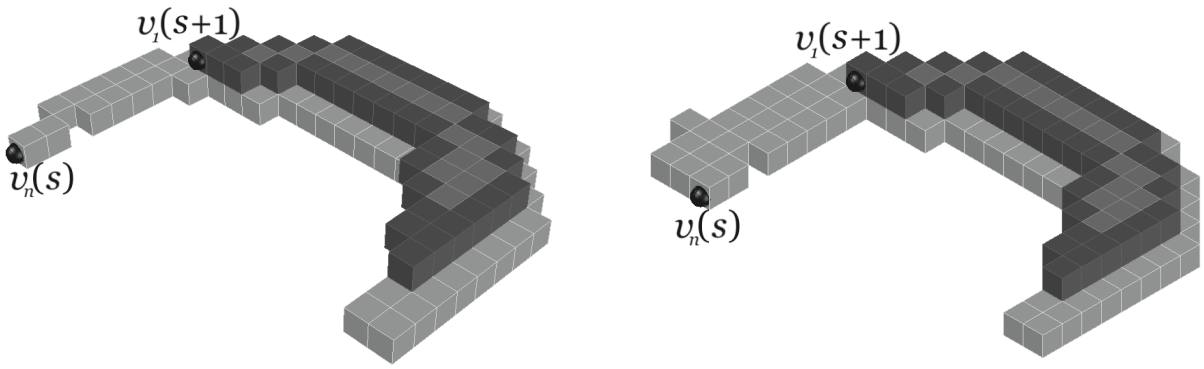
3.3 Modificaciones al Algoritmo A*

3.3.1 Zonas conflictivas para A*

Como queremos codificar la forma de un objeto tridimensional con una curva simple, es decir, colisiones o caminos repetidos no están permitidos, cuando buscamos el camino más corto, evitamos pasar por 1-voxeles que son parte de el contorno, dado que ya fue visitado. Una desventaja de hacer esto es que si una columna o fila solo contiene 1-voxeles que forman parte del contorno, el A* no puede encontrar un camino viable, por lo tanto, debemos validar este hecho antes de usar el algoritmo. Cuando se presenta este caso, usamos uno de los dos elementos estructurantes, uno para agregar 1-voxeles a la columna y el otro para agregar 1-voxeles a la fila.

La Fig. 3.2(a) muestra que yendo desde $v_n(s)$ a $v_1(s+1)$, no hay un camino que satisfaga nuestras condiciones. Para solucionar esto, validamos cada columna o/y fila que exista entre $v_n(s)$ y $v_1(s+1)$. Si la fila o columna tiene uno o menos de tres 1-voxeles, se usa el elemento estructurante. La Fig. 3.2(b) muestra

el resultado de aplicar los elementos estructurantes.



(a) Antes de usar elementos estructurantes

(b) Después de usar elemntos estructurantes

Fig 3.2: Ejemplo de cómo usar elementos estructurantes en una zona conflictiva para usar A*.

3.3.2 Pseudocódigo de nuestro A* modificado

Para nuestros propositos, hemos utilizado A* con adaptaciones para lograr el camino óptimo entre $v_n(s)$ y $v_1(s+1)$. Presentamos las matrices $\mathcal{M}$ y $\mathcal{E}$ compuestas por 1's y 0's para representar las regiones, siguiendo los siguientes pasos.

1. SI hay un camino para ir desde $v_n(s)$ a $v_1(s+1)$ en $\mathcal{R}_k^s$.

   1.1 $\mathcal{M} \to \mathcal{R}_k^s$.

   1.2 $\mathcal{E} \to \mathcal{R}_{k'}^{s+1}$.

2. de lo contrario, si hay un camino para ir desde $v_n(s)$ a $v_1(s+1)$ en $\mathcal{R}_{k'}^{s+1}$.

   2.1 $\mathcal{M} \to \mathcal{R}_{k'}^{s+1}$.

   2.2 $\mathcal{E} \to \mathcal{R}_k^s$.

3. Llenar la matriz H.

   3.1 Llenar la matriz H con $\times$ si $c(i,j)$ de $\mathcal{M}$ es 1, el cuál representa un 1-voxel del contorno, etiquetar con ∞ cualquier otro caso. Llenamos la matriz G de la misma manera que H.

   3.2 Considerar $c(i,j)$ de $v_n(s)$, y hacer $d_{ij} \to 0$.

   3.3 Almacenar $c(i,j)$ en la lista abierta.

   3.4 Hacer

3.4.1 Para cada vecino $c(i', j')$ de $N_8(c(i, j))$, si su valor es igual a ∞ en H y 1 en $\mathcal{M}$, calcular la distancia de Manhattan como sigue:

$$\begin{aligned} d_{i'j'} &\to d_{ij} + 1 \quad if \quad c(i', j') \in N_4(c(i, j)) \\ d_{i'j'} &\to d_{ij} + 2 \quad if \quad c(i', j') \in N_8 c(i, j) \setminus N_4(c(i, j)) \end{aligned} \tag{3.3.1}$$

3.4.2 Cada $c(i', j')$ es almacenado en la lista abierta.

3.4.3 Buscar en la lista aberta $c(i', j')$, tal que $d_{i',j'}$ es el menor y remover $c(i', j')$ de la lista abierta y hacer Eq. 3.3.2.

$$\begin{aligned} d_{ij} &\to d_{i'j'} \\ c(i, j) &\to c(i', j'). \end{aligned} \tag{3.3.2}$$

3.5 Mientras la lista abierta no esté vacía.

4. Llenar la matriz G.

4.1 Obtener $c(i, j) \to v_n(s), d_{ij} \to 0$.

4.2 Almacenar $c(i, j)$ en la lista abierta.

4.3 Hacer

4.3.1 Para cada vecino $c(i', j')$ de $N_8(c(i, j))$, si su valor es igual a ∞ en H y 1 en $\mathcal{M}$, hacer:

$$\begin{aligned} d_{i'j'} &\to d_{ij} + 10 \quad if \quad c(i', j') \in N_4(c(i, j)) \\ d_{i'j'} &\to d_{ij} + 14 \quad if \quad c(i', j') \in N_8(c(i, j)) \setminus N_4(c(i, j)). \end{aligned} \tag{3.3.3}$$

4.3.2 Cada $c(i', j')$ es almacenado en la lista abierta.

4.3.3 Buscar en la lista abierta $c(i', j')$, tal que la suma $d_{i'j'}$ en $G + d_{i'j'}$ en G es el menor y remover $c(i', j')$ de la lista abierta.

4.3.4 Obtenr la distancia Euclidiana del 1-voxel del contorno de $\mathcal{E}$ a d_{ij}. El que contiene la menor distancia es ahora $v_1(s + 1)$.

4.4 Mientras $v_1(s + 1) \notin N_8(c(i, j))$.

5. Obtener el camino más corto.

5.1 Obtener $c(i, j)$ de $v_1(i + 1)$.

5.2 Mientras $v_1(s + 1) \notin N_8(c(i, j))$

5.2.1 Agregar $c(i, j)$ al camino más corto.

5.2.2 Buscar $c(i', j') \in N_8(c(i, j))$, tal que su $d_{i'j'}$ es la menor en G.

5.2.3 Asignar $c(i,j) \rightarrow c(i',j')$.

Para ilustrar un ejemplo, considere la necesidad de ir de $v_n(s)$ a $v_1(s+1)$ a través de un camino óptima, la Fig. 3.1(c) se usa para este propósito. Para lograr esto, sea $\mathcal{R}_k^s$ y $\mathcal{R}_{k'}^{s+1}$ las dos regiones contiguas. Si hay un camino para ir de $v_n(s)$ a $v_1(s+1)$ en $\mathcal{R}_k$, asociamos $\mathcal{M}$ con $\mathcal{R}_k^s$ y $\mathcal{E}$ con $\mathcal{R}_{k'}^{s+1}$ (Figuras 3.3(a) y 3.3(b), respectivamente)

Siguiendo nuestro método A* modificado, el primer paso es llenar la matriz H con '×' si $c(i,j)$ de $\mathcal{M}$ es '1', que representa un 1-voxel del contorno, lo llenamos con ∞ en cualquier otro caso (ver Fig. 3.3(c)). Llenamos la matriz G de la misma manera que H (ver Fig. 3.3(a)).

0	1	1	1	1	1	1
0	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	0	0	0
1	1	1	1	0	0	0
0	1	1	1	1	1	1
0	0	1	1	1	1	1
0	0	1	1	1	1	1

(a)

0	0	0	0	1	1	1
0	0	0	1	1	1	1
0	1	1	1	1	1	1
0	1	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(b)

∞	×	×	×	×	×	×
∞	×	∞	∞	∞	∞	×
×	∞	∞	∞	×	×	×
×	∞	∞	×	∞	∞	∞
×	∞	∞	×	∞	∞	∞
∞	×	∞	∞	×	×	×
∞	∞	×	∞	∞	∞	×
∞	∞	×	×	×	×	×

(c)

Fig 3.3: Matrices (a) $\mathcal{M}$, (b) $\mathcal{E}$, and (c) H, respectively.

El siguiente paso es asignar un número mayor o igual a cero para cada $c(i,j)$, de modo que el valor de $c(i,j)$ sea igual a ∞ en H y '1 'en $\mathcal{M}$. Este número es la distancia de Manhattan. Comenzamos haciendo $d_{ij} = 0$ en (i,j) de G, donde $c(i,j)$ es la coordenada de $v_n(s)$.

La distancia de Manhattan se calcula como se describe en el algoritmo de A* modificado (Eq. (3.3.1)).

Las coordenadas $c(i',j')$ se almacenan en la lista abierta para recordar la celda para visitar a sus vecinos. Buscamos en esta lista las coordenadas $c(i',j')$, tal que $d_{i'j'}$ es la más corta. Eliminamos $c(i',j')$ de la lista y resolvemos las asignaciones de la Eq.(3.3.2). Continuamos con el cálculo de la distancia de Manhattan de

∞	×	×	×	×	×	×
∞	×	∞	∞	∞	∞	×
×	∞	∞	∞	×	×	∞
×	∞	∞	×	∞	∞	∞
×	∞	∞	×	∞	∞	∞
∞	×	∞	34	×	×	×
∞	∞	×	30	20	10	0
∞	∞	×	×	×	×	×

Fig 3.4: Matrix G

∞	×	×	×	×	×	×
∞	×	9	10	11	12	x
×	9	8	9	×	×	14
×	8	7	×	∞	∞	∞
×	7	6	×	∞	∞	∞
∞	×	5	4	×	×	×
∞	∞	×	3	2	1	0
∞	∞	×	×	×	×	×

Fig 3.5: Matrix H.

cada $c(i',j')$ en la lista abierta, hasta que esté vacía. La matriz H se convierte como en la Fig. 3.3(c).

Si el valor de $c(i,j)$ es ∞ en G y '1' en $\mathcal{M}$, asignamos posteriormente a cada $c(i,j)$ un número mayor o igual a cero. Comenzamos haciendo $d_{ij} = 0$ en la celda (i,j) de G, donde $c(i,j)$ es la coordenada de $v_n(s)$.

Cada número se calcula mediante la Eq.(3.3.3).

Las coordenadas $c(i', j')$ se almacenan en la lista abierta para recordar las coordenadas del voxel en el que necesitamos visitar sus vecinos. Buscamos en esta lista las coordenadas $c(i', j')$, de modo que la suma de $d_{i'j'}$ en H mas $d_{i'j'}$ en G es el más pequeño, eliminamos $c(i', j')$ de la lista y resolvemos la Eq. (3.3.3).

Usamos $\mathcal{E}$ para obtener la distancia Euclidiana de cada 1-voxel desde contorno de $\mathcal{R}_k^s$ hasta d_{ij}. El que tiene la menor distancia Euclidiana ahora es $v_1(s+1)$. Continuamos calculando cada $c(i', j')$ que cumpla con las condiciones antes mencionadas, hasta $v_1(s+1) \in N_8(c(i, j))$.

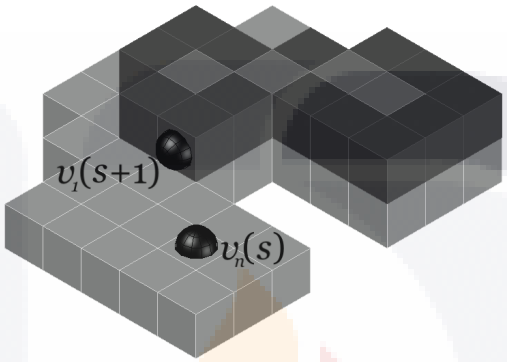


Fig 3.6: Posición final of $v_1(s+1)$.

En este ejemplo, es cierto que al rodear la concavidad, $v_1(s+1)$ cambia (Fig. 3.6), porque hay un voxel $(v_p(s)+1))$ en el contorno representado en $\mathcal{E}$, que es el más cercano al voxel actual $(v_0(s))$. Esto hace que $v_p(s+1)$ sea $v_1(s+1)$. Como en los pasos siguientes, no hay ningún $v_p(s+1)$ que esté más cerca de $v_0(s)$, provoca que $v_1(s+1)$ no cambie hasta el termino del algoritmo.

El último paso es encontrar el camino más corto entre $v_n(s)$ y $v_1(s+1)$. Asignar $c(i, j) \rightarrow v_1(s+1)$. Mientras $v_n(s) \notin N_8(c(i, j))$, hacemos: almacenar $c(i, j)$ en la lista del camino corto. Buscar $c(i', j') \in N_8(c(i, j))$, de modo que $d_{i'j'}$ sea la más pequeña en G. Asignar $c(i, j) \rightarrow c(i', j')$.

Por lo tanto, en nuestro ejemplo, el camino óptimo desde $v_n(s)$ hasta el objetivo definido encontrado, $v_1(s+1)$, es $\mathcal{P} = \{(7,7,1),(7,6,1),(7,5,1),(6,4,1),(5,4,2)\}$, donde (7,7,1) y (5,4,2) son las coordenadas de $v_n(s)$ y $v_1(s+1)$, respectivamente. Finalmente, el código de cadena de la ruta helicoidal que representa los dos contornos es: $C = 771$**ceefeffgghgaaaaacceedcbaadefpefgahhaacceed**.

3.4 Experimentos

Para probar nuestro método, encontramos caminos helicoidales de una muestra de objetos 3D. Utilizamos superficies directamente de voxeles implícitas en un conjunto de datos. Estos modelos fueron tratados previ-

amente para simplificar y eliminar el ruido en los datos.

Los modelos utilizados provienen de diferentes fuentes y nos ayudaron a probar nuestra metodología. Accedimos al sitio del Stanford Computer Graphics Laboratory: http://graphics.stanford.edu/data y el sitio de Suggestive Contour Gallery: http://gfx.cs.princeton.edu/proj/sugcon/models.

La Fig. 3.7 presenta los resultados de la aplicación de nuestro método para obtener la ruta helicoidal que describe la superficie del contorno de cada objeto. Las figuras muestran diferentes colores en su trayectoria helicoidal para indicar los diferentes componentes conectados, que recursivamente fueron visitados para completar el código de cadena. Fig. 3.7(a) Lion tiene 22 colores diferentes, Fig. 3.7(b) Heptoroid tiene 42, Fig. 3.7(c) Drgon tiene 39 y Fig. 3.7(d) Penguin solo tiene 7.

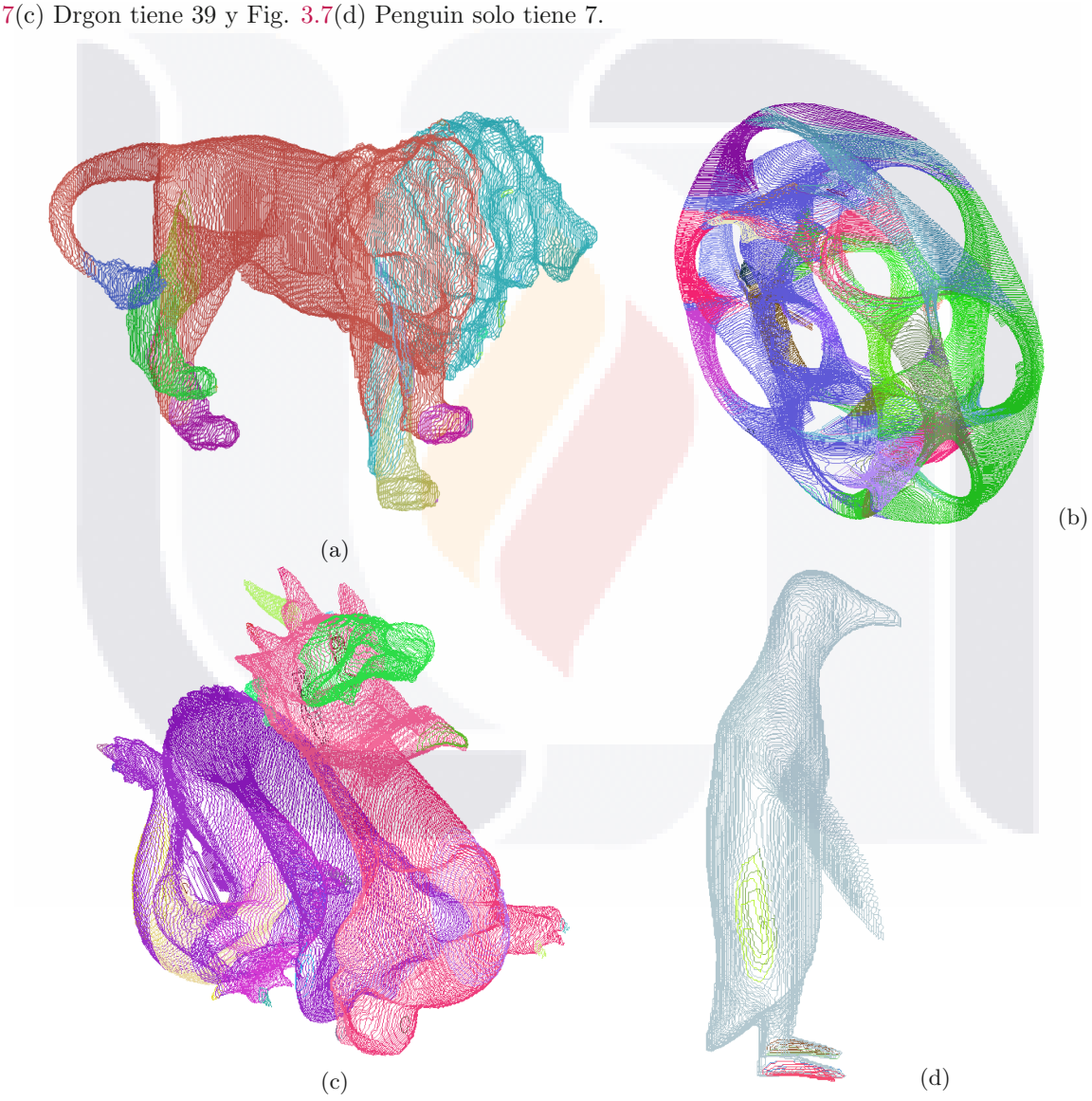


Fig 3.7: Camino helicoidal encontradao en una prueba de muestra: (a) Lion, (b) Heptoroid, (c) Dragon y (d) Penguin

En la Fig. 3.8(izquierda) se muestra el camino helicoidal del objeto llamado Penguin, donde se ejemplifica

el Caso 3 de nuestro A* modificado. En este caso, es necesario rodear la concavidad que existe entre $v_n(s)$ y $v_1(s+1)$. Esto se presenta en la Fig. 3.8(derecha). Esta representación solo contiene los 1-voxeles que forman parte de los contornos de las regiones codificadas.

Por otro lado, Tabla 3.1, presenta las frecuencias de los símbolos de código F26, así como la longitud del código de cadena (l_{F26}) y la cantidad de componentes conectados (cc) que tiene cada objeto. El objeto Heptoroide tiene el código de cadena más largo, mientras que el objeto Penguin tiene el más pequeño. Como se puede ver en la Fig. 3.9, la presencia de los primeros ocho símbolos de F26 es mucho mayor que el resto, y de ellos, los símbolos **a** y **b** son los que parecen más probablemente, seguidos por **g** y **c**. Este tipo de información y análisis es importante para tratar los problemas de reconocimiento y compresión.

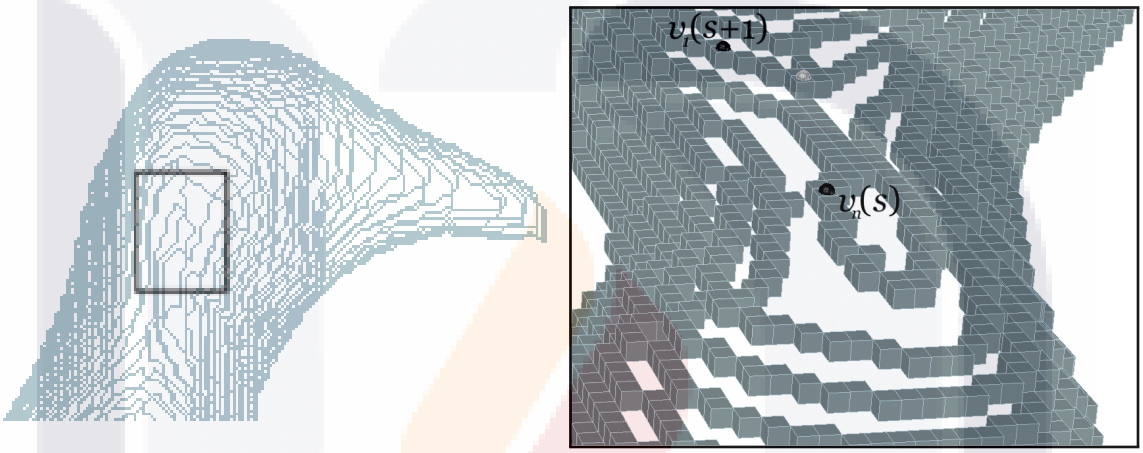


Fig 3.8: A la izquierda, se muestra el camino helicoidal del objeto cabeza de Penguin. A la derecha, el caso 3 se ejemplifica con la representación de voxel del contorno.

Symbol	Objects			
	Dragon	Heptoroid	Lion	Penguin
cc	39	42	22	7
a	15934	20912	10414	13440
b	7332	13232	4165	3204
c	8895	27156	5079	2333
d	9252	13175	4003	2015
e	14939	20723	10419	12438
f	5884	13001	4200	3785
g	12524	27392	4828	2095
h	6871	12724	4153	1615
i	301	734	215	39
j	93	168	42	9
k	4	66	5	5
l	30	105	24	27
m	13	21	10	36
n	47	141	50	24
o	17	14	4	0
p	48	110	33	44
q	31	11	3	17
l_{F26}	82215	149685	47647	41126

Tabla 3.1: Frecuencias de los símbolos F26

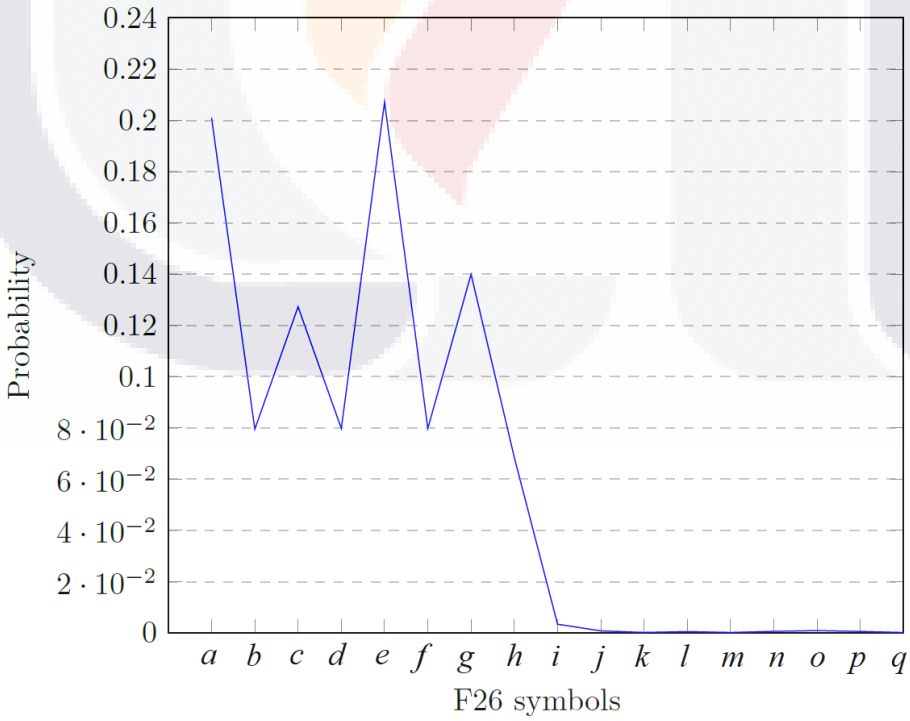


Fig 3.9: Probabilidad de ocurrencia de cada símbolo.

4 Aproximación poligonal usando un método de resolución múltiple y una gramática libre de contexto

La búsqueda de métodos óptimos para encontrar descriptores es una tarea constante en visión computacional y reconocimiento de patrones. Particularmente con respecto a la búsqueda de métodos con aproximación poligonal para representar la forma de objetos binarios, en la que los datos de los vértices, también llamados puntos dominantes (DP, para abreviar), reducen significativamente el almacenamiento de memoria y facilitan el manejo de la información de la forma original. Por supuesto, es inevitable que se pierdan datos, pero esta pérdida es soportable siempre que la información de las principales características de la forma y su topología original no se vean afectadas en lo más mínimo. En este trabajo se presenta una forma alternativa de obtener aproximaciones poligonales, que se basa en el reconocimiento de cadenas AF8 (ver el capítulo Preliminares para comprender el funcionamiento de este código de cadena) que forman parte de una gramática libre de contexto. Además, nuestro método se basa no solo en disminuir significativamente el número de vértices, es decir, los puntos dominantes, sino en buscar un criterio de error que no solo implique el error cuadrático integral o la relación de compresión, sino también la cantidad de información que se pierde. Del contorno original, hay pixeles que no pueden recuperarse en el proceso de decodificación.

4.1 Método

Nos basamos en la búsqueda inicial de candidatos para ser puntos dominantes, donde algunos otros se pueden agregar realizando otra iteración, de modo que tanto el error de cuadrado integral como el parámetro de relación de perdida definido en este trabajo sean los mínimos posibles.

Los pasos de nuestro método multirresolución son:

1. Considerar dos cuadrículas superpuestas, $\mathbb{G}$ y $\mathbb{G}'$, inicialmente de la misma escala, de modo que $\mathbb{G}'$ pueda escalarse mediante un parámetro $\alpha \geq 1$, con $\alpha \in \mathbb{Z}$. Esta escala se realiza a través del origen dado por el centro de masa del objeto binario. Comenzar con $\alpha = 1$.

2. Obtener otro contorno en $\mathbb{G}'$ con la ayuda del original en $\mathbb{G}$ visitando cada celda de ambas cuadrículas y seguir los siguientes pasos.

   2.1 Desde la izquierda y mś arriba, encontrar la primera celda de $\mathbb{G}'$ que contiene 1 pixeles del contorno en $\mathbb{G}$ y marcala. Cubrir el conjunto de celdas marcadas en sentido horario.

   2.2 La siguiente celda para marcar en la vecindad N_8 de $\mathbb{G}'$ es la que tiene el mayor número de 1 pixeles de $\mathbb{G}$.

   2.3 Repetir el último paso hasta que todos los 1 pixeles de $\mathbb{G}$ hayan sido cubiertos.

   2.4 Dado el código de cadena AF8 del contorno, buscar cadenas del conjunto

   $$L = \{xa^{\mathfrak{p}}(bha^{\mathfrak{q}})^{\mathfrak{r}}, xa^{\mathfrak{p}}(hba^{\mathfrak{q}})^{\mathfrak{r}} \mid x \in \{a, b, c, d, e, f, g, h\}\}, \tag{4.1.1}$$

   donde $\mathfrak{p}$, $\mathfrak{q}$, $\mathfrak{r}$ indican el número de veces que se concatenan el símbolo o la subcadena entre paréntesis, x es la etiqueta para los puntos de interrupción y $a, b, \ldots, h$ son símbolos del alfabeto AF8.

   2.5 Una vez que una celda de $\mathbb{G}'$ se ha definido como DP, encuentre el pixel de $\mathbb{G}$ más cercano al centro de la celda de $\mathbb{G}'$.

3. Dados dos puntos de interrupción (x_k, y_k) y (x_{k+1}, y_{k+1}), se define un segmento de línea continua. La distancia entre este segmento y los puntos de las celdas del contorno esta dada por la Eq. (4.1.2).

   $$d^2(p_i, \overline{p_k p_{k+1}}) = \frac{((x_i - x_k)(y_{k+1} - y_k) - (y_i - y_k)(x_{k+1} - x_k))^2}{(x_i - x_{k+1})^2 + (y_k - y_{k+1})^2}. \tag{4.1.2}$$

   Si $\alpha \geq 1$ y, también, el error entre los segmentos de línea dados por los puntos de interrupción y el contorno es mayor que un cierto error tolerable, crear $\alpha \to \alpha/2$ e ir al paso 2. De lo contrario, considerar todos los puntos de interrupción como DPs y detener.

La idea principal de nuestro método es captar lo que podemos decir visualmente que es una pieza de línea recta digital (PLRD). Por supuesto, hay un error si consideramos la definición precisa, sin embargo, funciona para nuestros porpósitos. La Fig. 4.1 presenta un ejemplo de una línea recta visual digital.
Como se puede observar, el código de la cadena AF8 es $C_{AF8} = xaaabhbhbhaabhabhbh$, la cuál también se

puede escribir como $C_{AF8} = xa^3\,\underbrace{bha^0}\,\underbrace{bha^0}\,\underbrace{bha^2}\,\underbrace{bha^1}\,\underbrace{bha^0}\,\underbrace{bha^0}$. Observar que está en la forma dada por L en la Eq.(4.1.1), donde $\mathfrak{p} = 3$, $0 \leq \mathfrak{q} \leq 2$ y $\mathfrak{r} = 6$.

Theorem 4.1.1. *L es un subconjunto de un lenguaje generado por una gramática libre de contexto, GLC.*

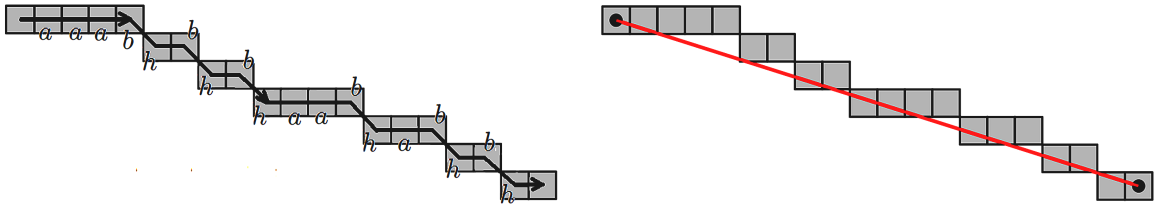


Fig 4.1: Izquierda: una línea recta discreta codificada con el código de cadena AF8. Derecha: una línea recta continua es adaptada .

Proof. Sea una 4-tupla $G = (V, \Sigma_{AF8}, S, P)$, donde las variables V y los símbolos terminales Σ_{AF8} son conjuntos disjuntos, $S \in V$ y P es el conjunto de producciones dadas por las siguientes fórmulas.

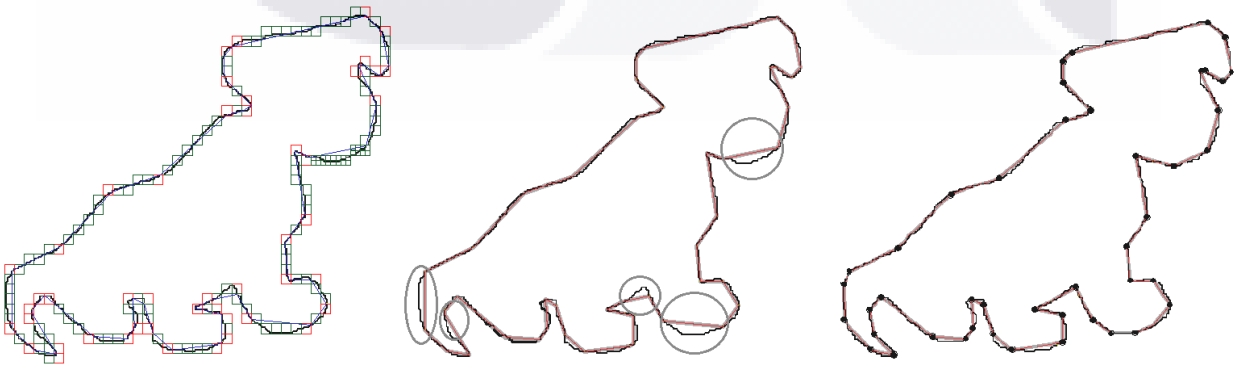
$$S \rightarrow\ xAB|xAC$$

$$B \rightarrow\ bhAB|\ \epsilon$$

$$C \rightarrow\ hbAC\ |\ \epsilon$$

$$A \rightarrow\ aA\ |\ \epsilon$$

Donde ϵ es la cadena vacía. Como se puede observar, esta 4-tupla define una GLC y produce cada una de las cadenas dadas por L en la Eq.(4.1.1). □



(a) (b) (c)

Fig 4.2: Una forma del contorno, (a) la cuadrícula $\mathbb{G}'$ escalada con $\alpha = 16$, (b) aproximación de polígonal, y (c) DPs finales obtenidos.

Sólo para fines ilustrativos, supongamos una forma como la que se muestra en la Fig.4.2(a) con una resolución de 156×144 en $\mathbb{G}$. El contorno está inmerso en la cuadrícula $\mathbb{G}'$ escalado por $\alpha = 4$ (*i.e.* con una resolución de 39×36). Las celdas rojas representan los puntos de ruptura. Por otra parte, en la Fig. 4.2(b), se obtuvo una aproximación polígonal en la primera iteración, obtenida mediante la aplicación de la GLC.

Una vez que se aplica L para encontrar los puntos de interrupción actuales, preguntar si entre los puntos k y $k+1$ el error es tolerable. Si no, aplicamos iterativamente el método de multirresolución y busquemos otros puntos de interrupción; de lo contrario, considere los puntos de interrupción actuales como DPs. En la Fig. 4.2(b) las regiones circunscritas no están bajo un error tolerable, dos de ellas se amplían en la Fig. 4.3 en la que se usan celdas grises para buscar subcadenas de L después de una reducción de $\mathbb{G}'$ con $\alpha = 2$, dividiendo cada una de las celdas de resolución. Las líneas rojas están ahora bajo error tolerable.

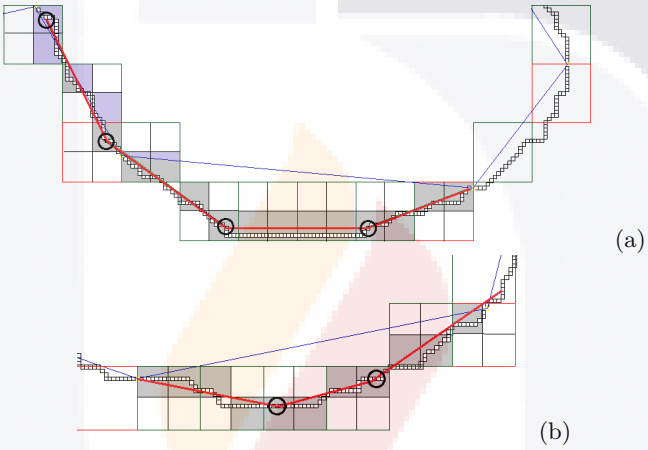


Fig 4.3: Una segunda iteración de nuestro método en regiones donde el error es mayor al tolerable. Los vértices circunscritos son DPs, mientras que las uniones con líneas finas son puntos de interrupción

Una vez que el procedimiento anterior se ha llevado a cabo de manera iterativa, se obtiene un conjunto final de DPs, como se muestra en la Fig. 4.2(c).

4.2 Compensación entre criterios de error comunes

Se ha escrito un número considerable de artículos para encontrar la mejor aproximación poligonal, proponiendo una serie de criterios de error para evaluar los diferentes métodos. Algunos parámetros comúnmente utilizados para evaluar los métodos vienen dados por la relación de compresión (CR, Eq. (4.2.1)) y el error del cuadrado integral (ISE, Eq. (4.2.2)).

$$\mathrm{CR} = \frac{n}{N}. \tag{4.2.1}$$

$$\text{ISE} = \sum_{i=1}^{n} d_i^2. \tag{4.2.2}$$

donde n es el número de pixeles de la forma del contorno y N el número de DPs.

Como señalan Masood y Haq en [57], la calidad de la aproximación poligonal debe medirse en términos de reducción de datos y en la similitud con el contorno original, también. Por supuesto, otro criterio principal es el número de DPs. Sin embargo, a veces este número se sacrifica para obtener una distancia de error menor. En este trabajo, también proponemos considerar la cantidad de pixeles que se pierden (LP) cuando se realiza una decodificación para recuperar la forma. Las razones se dan a continuación.

Una vez que se encuentran los DPs, si se realiza una decodificación, se pueden contar los pixeles perdidos. Por supuesto, la forma de recuperar parte de la forma original está dada por el polígono aproximado. Se obtiene considerando los pixeles que contienen parte de los segmentos rectos continuos dados por pares de DPs. Comenzando con el primer DP, el siguiente pixel a decodificar se elige cuando contiene la longitud del segmento más largo. Si la celda vecina con el segmento más grande coincide con el pixel 1 del contorno original, entonces el pixel no se pierde, de lo contrario es un pixel perdido.

La Fig. 4.4 muestra un ejemplo de pixeles perdidos al descodificar un segmento entre dos DPs, que forman un lado de la aproximación poligonal. Recorriendo las celdas de arriba a abajo y de izquierda a derecha, tenga en cuenta que los 1-pixeles etiquetados de 1 a 4 contienen menos longitud, del segmento continuo, que uno de los vecinos (0-pixeles) del pixel visitado anterior, por lo tanto son pixeles que se pierden en la decodificación, que están marcados en amarillo. Los píxeles grises a la derecha de la Fig. 4.4 son el polígono de aproximación decodificado final. Tener en cuenta, también, que hay un error entre los pixeles recuperados y el segmento continuo, dado por las coordenadas en puntos negros.

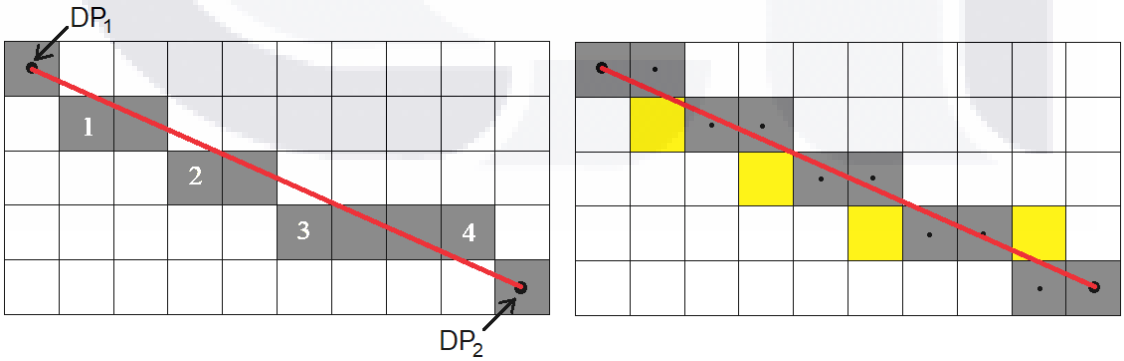


Fig 4.4: Pixeles perdidos en un proceso de decodificación. El segmento continuo rojo es un lado del polígono que se aproxima. Izquierda: algunos pixeles del contorno original en celdas grises; derecha: pixeles perdidos en celdas amarillas.

Considerar el caso en el que se encuentran N DPs. Supongamos que se recupera la forma y se obtiene el contorno original exacto. En este caso, no hay pérdida de información y el método puede considerarse

sin pérdidas. Algo importante a tener en cuenta (como se muestra a la derecha de la Fig. 4.4) es que esto puede ocurrir incluso si ISE $\neq 0$! Si, por otro lado, esos N DP se encuentran en lugares donde el contorno recuperado pierde pixeles, entonces el método es *con pérdidas*.

Supongamos dos modelos de soluciones (modelos con pérdida y sin pérdida) que dan el mismo número de DPs, sin embargo, distribuidos en diferentes lugares. Por supuesto; el valor de CR es el mismo!

Tener en cuenta que $\mathbb{G}'$ no se modifica, *i.e* $\alpha = 1$, y un subconjunto dado por Eq. (4.1.1), aplicado a la forma de Shark. La Fig. 4.5 ilustra la aplicación de L en la que su ISE $= 3.45$ y $N = 105$.

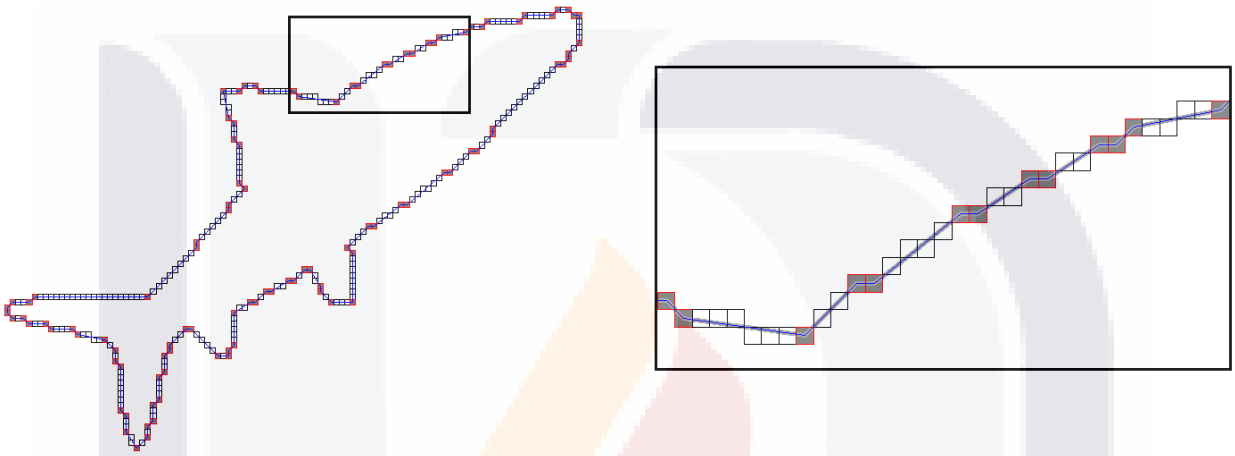


Fig 4.5: Izquierda: forma de Shark. Derecha: enfoque visual para observar que, a pesar del error en la aproximación poligonal, no se pierde ningún pixel. Las celdas sombreadas en los cuadrados rojos representan los DPs.

Mientras que la Tabla 4.1 muestra las diferentes subcadenas obtenidas del contorno de Shark, que son elementos de L. A pesar de que ISE es diferente de cero, y al usar el algoritmo mencionado anteriormente, no se pierde ningún pixel. En este ejemplo, los parámetros en la Eq. (4.1.1) satisface: $\mathfrak{p} \leq 20$, $\mathfrak{p} = \mathfrak{q}$, y $\mathfrak{r} = 1$.

Una vez que hemos analizado estas ambigüedades, nos proponemos considerar la importancia de N e ISE como una suma en una razón de pérdida (LR), pero ponderada por LP completamente en una sola ecuación, dada por la Eq. (4.2.3).

$$\mathrm{LR} = \frac{(N + \mathrm{ISE}) * \mathrm{LP}}{n}, \tag{4.2.3}$$

donde LP (pixeles perdidos) es la cantidad de pixeles perdidos en la decodificación y n la cantidad de pixeles del contorno original. Por esta razón, proponemos considerar la cantidad de pixeles perdidos como parte de la efectividad del método: cuantos menos pixeles pierda el método, mejor. Lo mismo es válido para ISE y DPs, como se expresa en la Eq. (4.2.3).

4.3 Experimentos

Aplicamos nuestro método a un conjunto muestra que aparecen comúnmente en la literatura. Para seleccionar los valores de los parámetros $\mathfrak{p}$ y $\mathfrak{q}$ de nuestro L propuesto, se lee cada cadena del código de la cadena AF8 y se obtiene el número máximo de a's concatenados, mientras que $\mathfrak{r}$ es el resultado de encontrar repeticiones de la forma $bha^{\mathfrak{q}}$ o $hba^{\mathfrak{q}}$.

4.3.1 Primer Conjunto

En esta primera parte, aplicamos a una prueba de muestra comunmente usada, y en este caso el parámetro $\alpha = 1$, *ie.* No se realiza ninguna escala debido a la resolución muy baja de la prueba de muestra. Los códigos de cadena de cada muestra son los que se presentan en la Tabla 4.2.

Nuestro método propuesto se comparó e implementó, tomando nuestros errores tolerables de los encontrados por Naser *et al.* [58], Masood [59] y Madrid-Cuevas *et al.* [60], utilizando los parámetros $(\mathfrak{p}, \mathfrak{q}, \mathfrak{r}) = (4, 4, 2)$

Subcadena	Como en L	Subcadena	Como en L	Subcadena	Como en L	Subcadena	Como en L
ba	xa^1	ha	xa^1	$baaaaaaa$	xa^7	$bahba$	xa^1hba^1
b	xa^0	b	xa^0	h	xa^0	caa	xa^2
h	xa^0	hbh	xa^0bha^0	ba	xa^1	h	xa^0
$caaaa$	xa^4	b	xa^0	ha	xa^1	ba	xa^1
ba	xa^1	ha	xa^1	$habha$	xa^1bha^1	b	xa^0
h	xa^0	b	xa^0	b	xa^0	$haaaaa$	xa^5
b	xa^0	hbh	xbh	$haaa$	xa^3	b	xa^0
b	xa^0	$haaaaa$	xa^5	b	xa^0	$haabhaa$	xa^2bha^2
$haaaaaaaaaaa$	xa^{11}	b	xa^0	haa	xa^2	haa	xa^2
h	xa^0	h	xa^0	b	xa^0	b	xa^0
baa	xa^2	ca	xa^1	ha	xa^1	$habha$	xa^1bha^1
b	xa^0	$baaaa$	xa^4	b	xa^0	b	xa^0
$haaa$	xa^3	h	xa^0	b	xa^0	hbh	xa^0bha^0
b	xa^0	haa	xa^2	b	xa^0	b	xa^0
$haabhaa$	xa^2bha^2	h	xa^0	baa	xa^2	hbh	xa^0bha^0
b	xa^0	b	xa^0	h	xa^0	b	xa^0
$habha$	xa^1bha^1	haa	xa^2	ba^{20}	xa^{20}	h	xa^0
b	xa^0	b	xa^0	$haaaaaaaa$	xa^8	$bahba$	xa^1hba^1
$haaa$	xa^3	$haaa$	xa^3	h	xa^0	h	xa^0
d	xa^0	b	xa^0	$baaaaa$	xa^5	ba	xa^1
$baaaaaaaa$	xa^8	haa	xa^2	h	xa^0	h	xa^0
$caaa$	xa^3	b	xa^0	baa	xa^2	$baaaaa$	xa^5
ba	xa^1	ha	xa^1	g	xa^0	h	xa^0
b	xa^0	ba	xa^1	$baaaaaa$	xa^6	$baaaa$	xa^4
hbh	xa^0bha^0	cbh	xbh	h	xa^0	h	xa^0
h	xa^0	baa	xa^2	baa	xa^2		
h	xa^0	h	xa^0	h	xa^0		

Tabla 4.1: Las subcadenas del código de la cadena de Shark que forman parte del conjunto L cuando LP = 0.

para cromosoma y hoja, y (6,6,1) para aproximaciones poligonales de semicírculo, respectivamente.

En nuestros experimentos, encontramos un resultado interesante: la cantidad de pixeles que se pierden al decodificar la forma es menor con nuestro método que con los de la literatura. La Tabla 4.3 muestra los resultados de la aplicación de nuestro método en comparación con los otros métodos de aproximación poligonales recientes mencionados anteriormente, mientras que en la Fig. 4.6 se presenta una comparación visual de los diferentes métodos.

4.3.2 Segundo conjunto

En esta subsección, mostramos la aplicación de nuestro método, para objetos con mayor longitud en formas del contorno. Comparamos nuestro método propuesto con Algo 1, APS (aplicando el proceso de simplificación automática) y FDP (fijando el número deseado de puntos dominantes) informado recientemente por Nasser *et al.* [58].

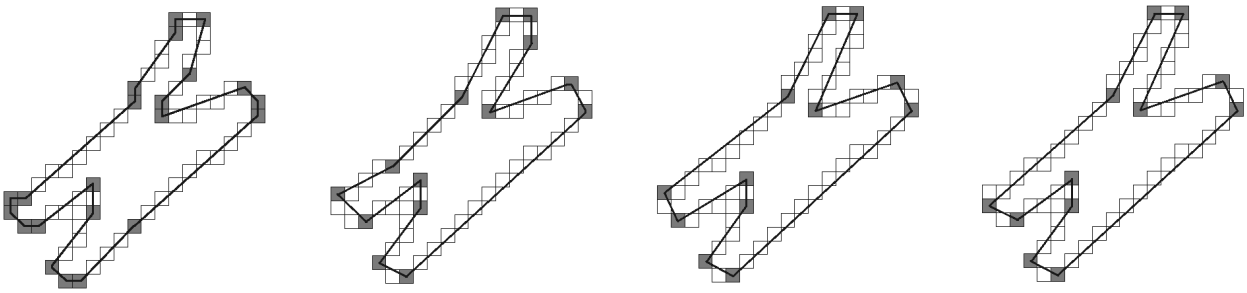
Usando la Eq. (4.1.1), se encontraron parámetros. Para Shark: $(\mathfrak{p}, \mathfrak{q}, \mathfrak{r}) = (20, 20, 7)$ para la Cup: $(\mathfrak{p}, \mathfrak{q}, \mathfrak{r}) = (19, 19, 1)$ y para Stingray: $(\mathfrak{p}, \mathfrak{q}, \mathfrak{r}) = (4, 4, 1)$. La tabla 4.4 muestra los resultados con los criterios de error definidos. Cup y Stingray son formas muy ruidosas, y se aplicó un proceso de multirresolución, usando el método iterativamente desde $\alpha = 4$, hasta $\alpha = 1$.

Forma	Código de cadena
Chromosome	cacabhbahgahbhbbbbaabhaaaaaaaaabbcahbhafabhbbbchabhaaaahbahbh
Semicircle	baaaaaaabhabaaaaabahbaaaaaaabgbhbabhchbabhbfaaabhbabhbaaab hbbabhbaaafbhbabhchbabhbgbaaaaaaabhabaaaaabah
Leaf	daabhabafaahbadhabhbafahabcabhbhbaaaahaahbaaaaaaaaccaaaaaaaa aahbagabhbaaaaabhdbhaaafabhbhdbhaaafabhbaaehaafaabhdbaafaaaa

Tabla 4.2: Códigos de cadena de las formas de muestra.

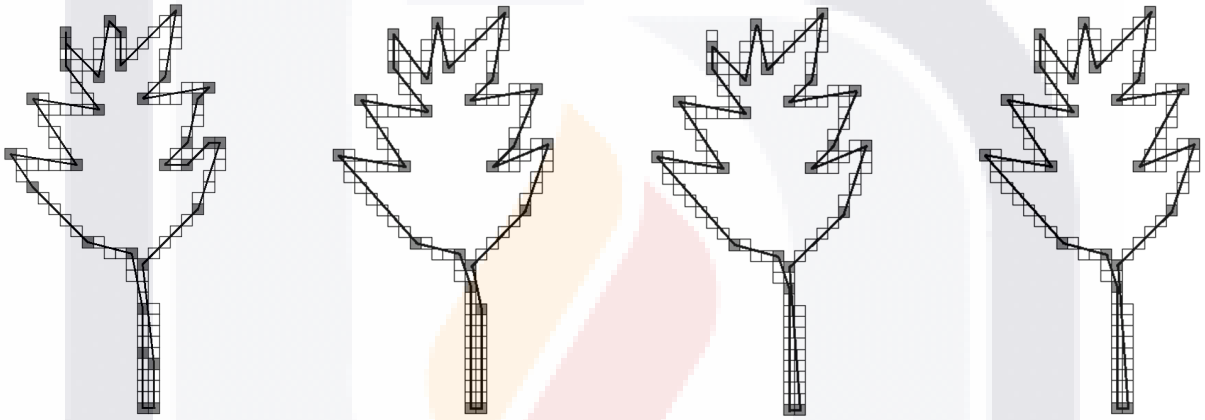
Forma	Método	N	CR	ISE	LP	LR
Chromosome	Proposed	22	2.73	**3.78**	**8**	3.44
n=60	Nasser *et al.* (2018)	14	4.28	4.97	10	**3.16**
	Masood (2008)	12	5	7.76	15	4.94
	Madrid-Cuevas (20016)	12	0.2	5.82	13	3.86
Leaf	Proposed	31	3.87	**8.42**	**11**	**3.61**
n=120	Nasser	24	4.95	9.96	15	4.25
	Masood	23	5.22	10.61	18	5.04
	Madrid-Cuevas	22	5.45	11.16	18	4.97
Semicircle	Proposed	25	4.08	**6.32**	**6**	**1.84**
n=102	Nasser	23	4.43	7.63	11	3.30
	Masood	22	4.64	8.61	17	5.10
	Madrid-Cuevas	10	10.2	40.79	47	23.40

Tabla 4.3: Comparaciones cuantitativas con otros métodos de aproximación poligonal.



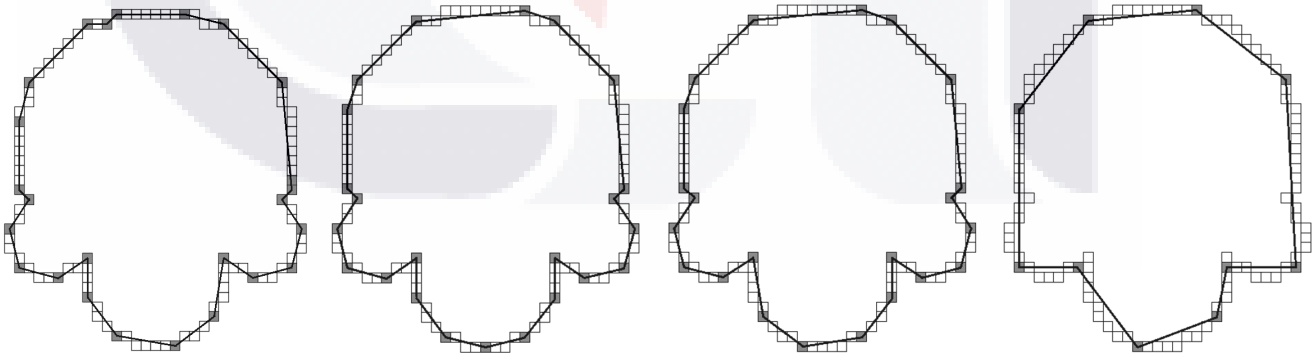
(a) Proposed (b) Nasser *et al.* (c) Masood (d) Madrid-Cuevas *et al.*

Chromosome



(a) Proposed (b) Nasser *et al.* (c) Masood (d) Madrid-Cuevas *et al.*

Leaf



(a) Proposed (b) Nasser *et al.* (c) Masood (d) Madrid-C. *et al.*

Semicircle

Fig 4.6: Puntos dominantes de las tres formas.

Fig. 4.7 muestra las regiones donde se utilizó la multiresolución, mientras que Fig. 4.8 muestra una comparación de nuestro método con los de Nasser*et al.*

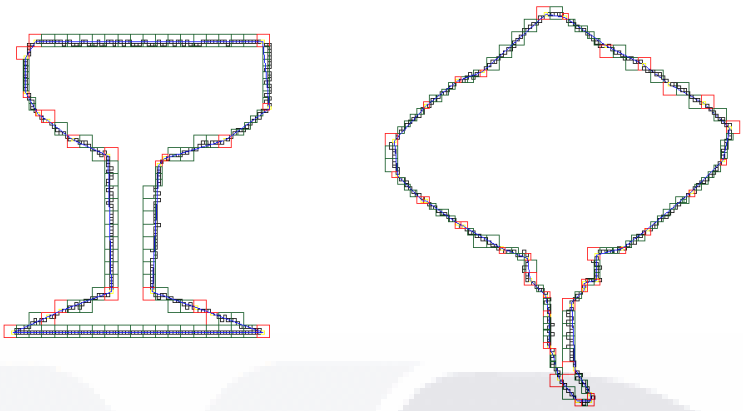


Fig 4.7: Método de multiresolución aplicado a formas de Cup y Stingray.

Forma	Método	N	ISE	CR	LP	LR
Shark	Propuesto	37	**45.46**	7.92	**73**	**419.07**
$n = 293$	Algo 1	23	79.41	12.73	137	854.01
$\alpha = 1$	APS	21	75.52	13.95	133	719.90
	FDP	19	105.09	15.42	135	920.00
Cup	Propuesto	20	**93.22**	20.25	**105**	**483.36**
$n = 405$	Algo 1	21	160.70	19.29	167	1391.55
$\alpha = 4, 2$	APS	11	238.45	36.82	220	1424.81
	FDP	17	159.56	23.82	180	1205.54
Stingray	Propuesto	37	**84.36**	8.86	**122**	**1160.98**
$n = 328$	Algo 1	25	118.66	13.12	180	1732.86
$\alpha = 4, 2, 1$	APS	23	121.38	14.26	160	1447.56
	FDP	20	165.75	16.40	180	1865.82

Tabla 4.4: Comparaciones cuantitativas del segundo conjunto con otros métodos de aproximación poligonal.

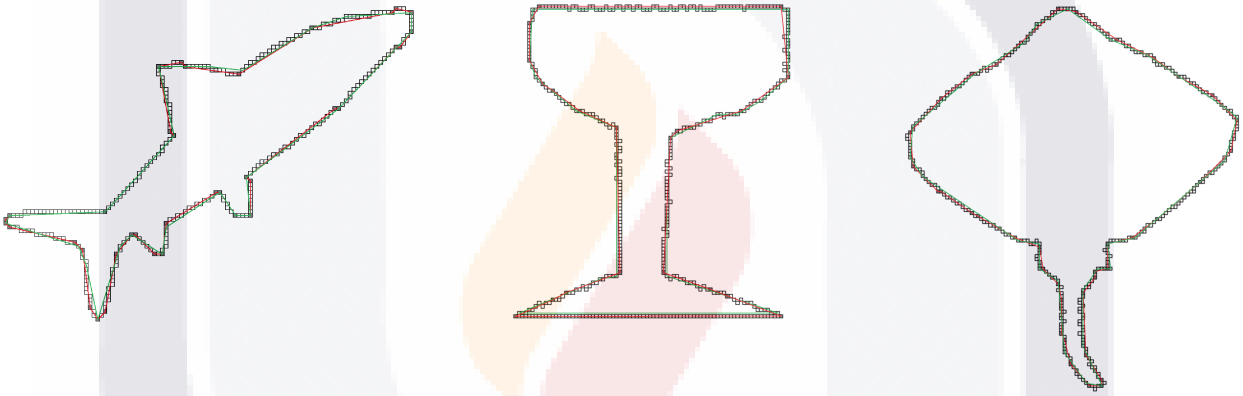


Fig 4.8: La aproximación poligonal en rojo es de nuestro método propuesto, mientras que en verde está el dado por el Algoritmo 1 de Nasser *et al.*

5 Aproximación poligonal usando una gramática libre de contexto en objetos 3D

En este capítulo se define un nuevo método para comparar dos objetos 3D que normaliza la posición de uno con respecto al otro, esto con el fin de ser invariante a la rotación en el espacio. Nuestro método hace una concordancia global, porque comparamos objetos que presentan transformaciones rigidas. Además de ser eficiente, ya que compara solo puntos dominantes obtenidos mediante el lenguaje (Eq. 4.1.1). Recordemos que este lenguaje esta desarrollado para objetos bidimensionales, por lo tanto no es necesario usar todas las capas que componen cada objeto 3D, sino solo cada n capas que no sean consecutivas o cercanas, dado que provocan repetición de DPs en la misma posición salvo por el cambio de capa, por el contrario si las capas se encuentran muy alejadas entre sí, pueden ocasionar la pérdida de información valiosa (secciones de un objeto que lo distinguen de otro), esta característica hace que nuestro método sea robusto.

5.1 Método

Para medir qué tan similares son dos objetos, es necesario calcular distancias entre pares de descriptores usando una medida de disimilitud. Se utiliza el método descrito en el capítulo anterior para obtener los PD's en cada n-ésima capa de un objeto 3D, posteriormente se realizan transformaciones (traslación y rotación) para hacer que sus ejes principales se alineen entre ellos, con el fin de saber que tan cercano esta un objeto con respecto al otro. Los pasos para comparar dos objetos 3D ($\mathcal{A}$, $\mathcal{B}$) son:

1. Obtener el cúmulo de DPs de $\mathcal{A}$ y de $\mathcal{B}$.

2. Hacer una traslación rigida de los objetos DPs de los $\mathcal{A}$ y $\mathcal{B}$ para que coincidan sus centros de masa.

3. Realizar una transformación de rotación rígida de los DPs con el fin de que estén orientados conforme a sus ejes principales.

4. Calcular la distancia de Hausdorff que hay entre los DPs de $\mathcal{A}$ y $\mathcal{B}$.

Ninguna de las transformaciones dadas por los dos primeros incisos del método anterior modifica la forma de los objetos, esto es debido a que existen functores llamados *invariantes*, es decir, son los valores que se mantienen iguales a pesar de que el objeto ha sufrido cambios derivados de los incisos 1 y 2.

5.2 Experimentos

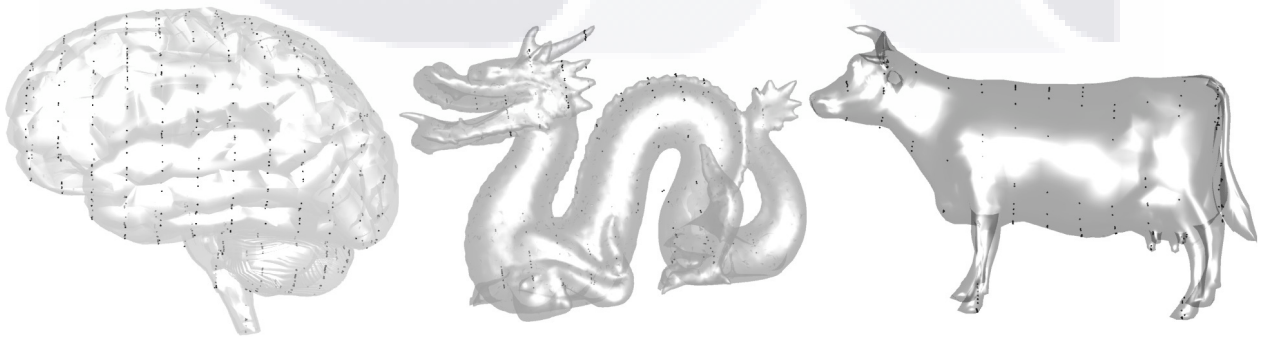
Para probar nuestro método, utilizamos una muestra de seis objetos 3D (Blade, Brain, Cow, Dragon, Heptoroid, Lion), los cuales fueron previamente tratados. Los procesos previos consistieron en: rotar el objeto 15°, 35° y 45° sobre el eje x, voxelizar cada objeto así como sus 3 rotaciones y eliminación de ruido en los datos.

Los modelos utilizados provienen de diferentes fuentes y nos ayudaron a probar nuestra metodología. Accedimos al sitio del Stanford Computer Graphics Laboratory: http://graphics.stanford.edu/data y el sitio de Suggestive Contour Gallery: http://gfx.cs.princeton.edu/proj/sugcon/models.

Calculamos la distancia de disimilitud entre las rotaciones de cada objeto, así como las distancias entre las rotaciones de un objeto con las rotaciones de los demás objetos de la muestra con el fin de determinar si el método era capaz de encontrar la menor distancia de disimilitud en las diferentes rotaciones del objeto. Para lograr esto, usamos el método descrito tanto en el capítulo actual como el del capítulo anterior. Recordemos que la Eq. 4.1.1 requiere la definición de los valores que tendrán los parámetros $\mathfrak{p}, \mathfrak{q}, \mathfrak{r}$, en este experimento los valores utilizados son (20,20,10) respectivamente, $\mathfrak{p}$ es el mayor número de a's concatedanas presente en la mayor cantidad de regiones conectadas en las n-ésimas capas, para el caso de la repetición de bh o hb concatenado con a's ($\mathfrak{r}$), se eligió el valor de 10 porque se encuentra en casi todas las regiones conectadas de las n-ésimas capas y no genera un error (ISE) muy grande ,y dado que los objetos son 3D tenemos que seleccionar cada cuantas capas usaremos la Eq. 4.1.1, para no perder información valiosa y no obtener información repetida por capa, se ha optado por calcular los DPs cada 20va capa ($n = 20$). Los valores de los 4 parámetros son utilizados en todos los objetos de la muestra.

Notamos que los cúmulos de puntos de las rotaciones del objeto Lion (Fig. 5.1) conservan la cabeza, las patas, torzo y estómago, pero no con la misma cantidad de puntos dominantes. Los puntos dominantes que conforman las patas poseen una distancia mayor o menor entre ellos, además el número que las componen son diferentes.

Fig 5.1: Cúmulo de puntos dominantes obtenidos en las rotaciones de Lion.(a) Objeto Lion 3D original, (b) rotación a 0° con respecto al eje x,(c) rotación a 15° con respecto al eje x,(d) rotación a 35° con respecto al eje x,(e) rotación a 45° con respecto al eje x



(a)Brain con sus PDs (b)Dragon con sus PDs (c)Cow con sus PDs

Fig 5.2: Ejemplos de Puntos dominantes encontrados en diferentes objetos 3D

La Fig. 5.2 muestra los resultados obtenidos utilizando la Eq.(4.1.1), en la Fig. 5.2 (a) notamos que el cúmulo de puntos es muy denso en todo el objeto, debido a que esta compuesto en su mayoría por concavidades y la Tabla 5.1 nos muestra que es el objeto que contiene el mayor número de puntos dominates. Debido a que la Fig. 5.2 (b) contiene concavidades con poca curvatura, su cumulo de puntos es poco denso, salvo en los cuernos y en la parte superior del cuerpo, que es donde presenta concavidades con mucha curvatura, con un número de puntos dominantes que son menos de la mitad que el objeto Brain (ver Tabla 5.1). La Fig. 5.2 (c) es la que presenta un menor número de puntos dominantes, ya que esta figura es muy regular, con pocas concavidades y estas en su mayoría con poca curvatura, a excepción de las patas, donde es una curvatura es mayor y por ende contiene un denso conjunto de puntos dominantes. Cow es el objeto que presenta el menor número de puntos dominantes de la muestra como lo se ve en la Tabla 5.1. Inferimos que los otros objetos con topologías que poseen pocas concavidades son Blade y Lion, dado que sus cúmulos contienen pocos puntos dominantes esto se ve reflejado en la Tabla 5.1, mientras que el objeto Heptoroid contiene bastantes puntos dominantes en su cúmulo de puntos, debido a la complejidad de su topología, la cuál no solo contiene concavidades si no también túneles.

Objeto	Rotación	Num. puntos dominates
Blade	0°	429
	15°	402
	35°	520
	45°	506
Brain	0°	1635
	15°	1693
	35°	1693
	45°	1707
Cow	0°	337
	15°	324
	35°	350
	45°	341
Dragon	0°	644
	15°	647
	35°	694
	45°	720
Heptoroid	0°	956
	15°	1092
	35°	1069
	45°	1083
Lion	0°	414
	15°	419
	35°	421
	45°	426

Tabla 5.1: Número de puntos dominantes que contienen los cúmulos de las diferentes rotaciones de los objetos 3D de la muestra.

Analizando la información de la Fig. 5.3 vemos que en todos los casos el eje mayor es el de color azul, el mediano el verde y el menor el rojo. Empatando esta información con la Tabla 7.2 llegamos a la conclusión

que el eje menor es el que tiene la longitud mayor en su eigenvalor, el eje mayo el que tiene la longitud menor en su eigenvalor y el eje mediano el que su longitud se encuentra en medio de las otras dos.

Lion	Blade			
	0°	15°	35°	45°
0°	80.025	78.113	95.876	89.856
15°	83.646	80.676	98.564	93.358
35°	74.103	71.375	90.830	85.457
45°	75.203	76.222	92.377	89.287

Tabla 5.2: Distancia de Hausdorff entre Lion con Blade

Lion	Brain			
	0°	15°	35°	45°
0°	81.117	78.901	77.480	77.492
15°	81.544	79.880	78.674	78.262
35°	78.550	81.286	79.723	80.493
45°	81.098	81.661	80.273	81.876

Tabla 5.3: Distancia de Hausdorff entre Lion con Brain

Lion	Cow			
	0°	15°	35°	45°
0°	46.521	47.857	47.050	47.440
15°	47.551	50.200	46.267	50.710
35°	46.810	50.437	45.562	50.400
45°	45.799	49.906	48.536	44.524

Tabla 5.4: Distancia de Hausdorff entre Lion con Cow

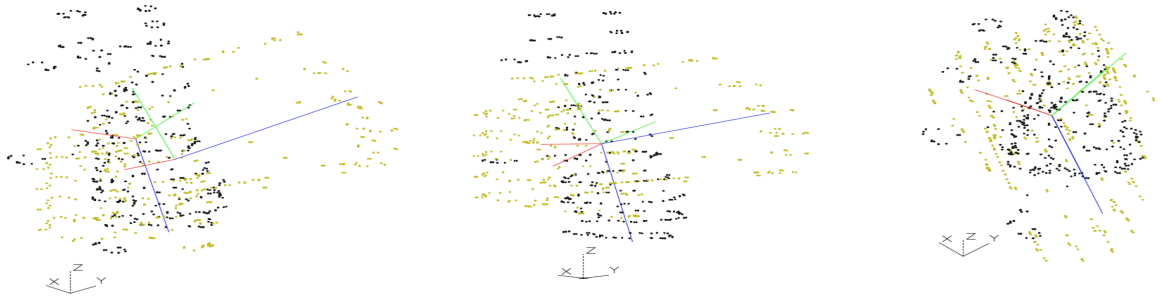
Lion	Dragon			
	0°	15°	35°	45°
0°	51.084	54.774	56.078	59.311
15°	51.861	55.163	55.871	59.079
35°	53.556	47.771	49.639	51.825
45°	54.126	48.682	48.653	48.707

Tabla 5.5: Distancia de Hausdorff entre Lion con Dragon

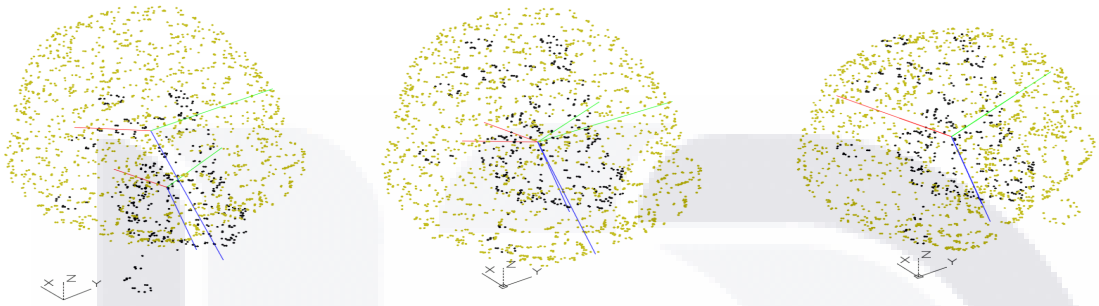
Lion	Heptoroid			
	0°	15°	35°	45°
0°	76.034	74.215	59.224	73.302
15°	75.613	74.449	62.576	73.453
35°	74.563	77.166	60.141	71.990
45°	76.434	74.740	56.617	73.962

Tabla 5.6: Distancia de Hausdorff entre Lion con Heptoroid

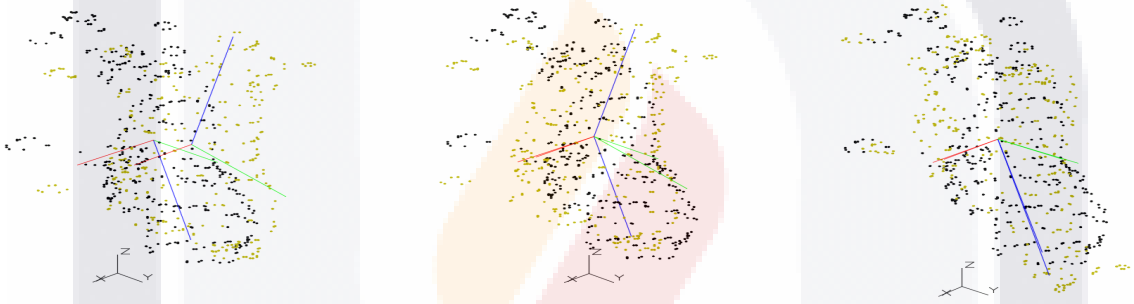
Los objetos que tienen una distancia Hausdorff menor con respecto a Lion son Cow (Tabla 5.4) y Dragon (Tabla 5.5), dado que tienen una topología similar, contienen una cabeza, torso y extremidades. Gráficamente



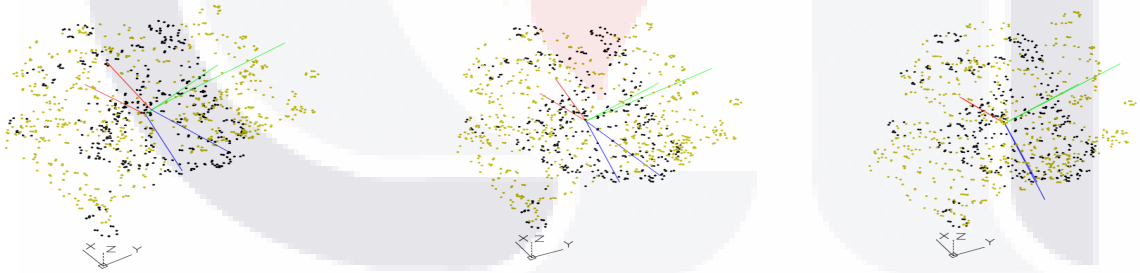
(a)Orientación de Blade con respecto a Lion.



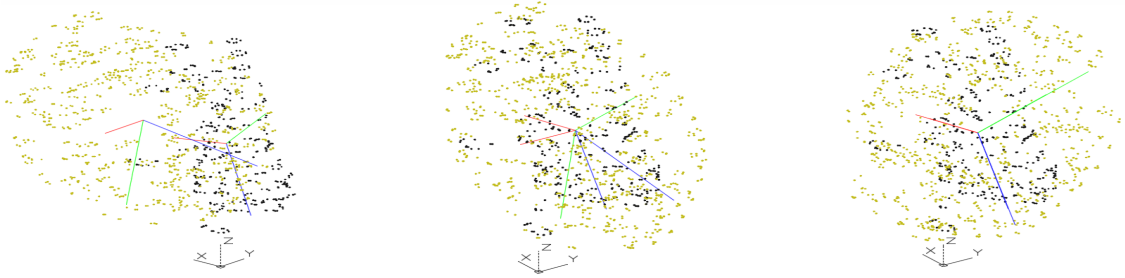
(b)Orientación de Brain con respecto a Lion.



(c)Orientación de Cow con respecto a Lion.



(d)Orientación de Dragon con respecto a Lion.



(e)Orientación de Heptoroid con respecto a Lion.

Fig 5.3: Ejemplos de las diferentes etapas que se deben de realizar para alinear los objetos con respecto a Lion, la columna de la izquierda se muestran los objetos sin ninguna alineación entre ellos, en la columna de enmedio se alinean sus centros de masa, y en la columna de la derecha se rotan los ejes principales de los objetos con respecto a Lion hasta que están alineados.

Lion	Lion			
	0°	15°	35°	45°
0°	0	17.376	18.353	27.650
15°	17.376	0	18.962	31.967
35°	18.353	18.962	0	25.912
45°	27.650	31.967	25.912	0

Tabla 5.7: Distancia de Hausdorff entre Lion con sus diferentes rotaciones al rededor del eje X

vemos que tanto en la Fig. 5.3 (c) y Fig. 5.3 (d) se extienen sus torsos sobre el eje z y apuntan sus ejes principales menores en dirección negativa a ese eje.

Heptoroid	Dragon			
	0°	15°	35°	45°
0°	59.040	62.343	62.422	62.469
15°	55.488	54.599	55.208	57.823
35°	50.167	52.704	53.135	51.426
45°	66.163	62.791	60.401	59.053

Tabla 5.8: Distancia de Hausdorff entre Heptoroid con Dragon

Heptoroid	Heptoroid			
	0°	15°	35°	45°
0°	0	50.550	46.716	27.643
15°	50.550	0	53.710	46.757
35°	46.716	53.710	0	44.281
45°	27.643	46.757	44.281	0

Tabla 5.9: Distancia de Hausdorff entre Heptoroid con sus diferentes rotaciones al rededor del eje X

Observando el renglón 3 y la columna 1 de la Tabla 5.8 con respecto al renglón 2 y columna 3 de la Tabla 5.9, vemos que Heptoroid rotado a 35° tiene una disimilitud de 50.166728 con respecto a Dragon sin rotar, mientras que la disimilitud entre Heptoroid rotado a 15° y Heptoroid rotado a 35° es de 53.710093, en este caso la menor disimilitud no se obtiene entre las diferentes rotaciones del objeto Heptoroid. Por lo tanto la confiabilidad de este método sobre este conjunto de objetos es del 83%, dado que no logró obtener las menores distancias del objeto Heptoroid con sus diferentes rotaciones, sin embargo el objeto Heptoroid tiene una topología diferente a los demás. Así que habría que agregar un procedimiento más, que es un algoritmo que sea capaz de detectar túneles y cavidades con el fin de obtener una medida de similitud pequeña en objetos que presenten este tipo de topologías.

6 Conclusiones

En esta tesis presentamos tres métodos para disminuir la información del contorno de objetos 2D y 3D. El primero utiliza el recorrido helicoidal [16] y el algoritmo A* [56] para encadenar el contorno de capas subsecuentes de cada región conectada mediante el código de cadena F26. El segundo se basa en una gramática libre de contexto la cuál busca puntos dominantes inmersos en la cadena del contorno de un objeto 2D codificada mediante el código de cadena AF8. El tercero se basa en un modelo de trasnformación y rotación rígidas para encontrar la disimilitud entre dos objetos 3D, donde los objetos 3D están compuestos de puntos dominantes obtenidos mediante el método anterior. Evaluamos los métodos 1 y 3 utilizando las bases de datos del Stanford Computer Graphics Laboratory y el sitio de Suggestive Contour Gallery, mientras que para el método 2 utilizamos las imágenes 2D presentadas en los artículos [57, 58].

El primer método nos permite codificar superficies de objetos basados en voxeles que no son isomorfos al plano, por medio de caminos helicoidales. Una de las ventajas de este método es que la longitud del código de la cadena se optimiza visitando los centros de los voxeles que conforman los contornos, así como encontrando las trayectorias más cortas entre las capas, preservando las propiedades topológicas y geométricas del objeto. En el segundo método no se realiza ningún análisis explícito de los cambios de curvatura, hemos propuesto un nuevo método para detectar puntos dominantes y, en consecuencia, una aproximación poligonal, con un error que mejora los modelos actuales. Además, se propuso un nuevo criterio de evaluación para el enfoque poligonal, basado en la pérdida de pixeles en la decodificación. En el tercer método la validación experimental dirigida a demostrar la capacidad del algoritmo para reconocer objetos realizando la traslación y rotación rígidas de los ejes principales de un objeto con respecto a otro, para posteriormente obtener la medida de disimilitud usando la distancia de Hausdorff ha producido resultados muy buenos con un 83 % de certeza, esto significa que solamente no fue capaz de diferenciar entre las rotaciones del objeto y el original, ya que obtuvo distancias similares con las rotaciones de objetos diferentes, esta dificultad fue producida por la topología del objeto.

6.1 Trabajos Futuros

Se sugieren realizar los siguientes trabajos futuros:

Para el método 1, analizar el código de cadena de la trayectoria helicoidal, encontrar puntos dominantes y, por lo tanto, reducir en mayor medida la información de la forma del objeto sin perder información valiosa sobre ella.

Aplicar el segundo método a formas de mayor resolución y con mayor cantidad de ruido. Por otro lado, decidimos encontrar el pixel más cercano al centro de una celda $\mathbb{G}'$, sin embargo, puede que no sea el óptimo. Un estudio a través de técnicas metaheruísticas puede ser el adecuado para esta tarea.

Para realizar la tarea de alinear los ejes principales empleamos el programa ACAD, este proceso puede ser más exacto si usamos las matrices que realizan rotaciones sobre los tres ejes(x,y,z). Adicionar un proceso que permita identificar tanto cavidades como túneles presentes en las topologías de los objetos 3D. Agregar alguna técnica metaheruística para seleccionar sobre que n-ésimas capas utilizar el método 2 con el fin de no perder información importante del objeto 3D y obtener la menor cantidad de puntos dominantes.

7 Anexos

Blade				
Rotación	Eigenvectores			Eigenvalores
0°	0.865	-0.502	-0.001	3.345E+06
	0.037	0.062	0.998	2.88242E+06
	-0.500	-0.863	0.072	1.033E+06
15°	0.833	-0.502	0.233	3.108E+06
	-0.146	0.206	0.968	2.754E+06
	-0.534	-0.840	0.099	868164
35°	0.738	-0.387	0.553	4.742E+06
	-0.465	0.303	0.832	4.234E+06
	-0.490	-0.871	0.044	1.176E+06
45°	0.725	-0.208	0.656	5.163E+06
	-0.605	0.261	0.752	4.60E+06
	-0.328	-0.943	0.064	1.277E+06

Tabla 7.1: Eigenvectores y Eigenvalores del objeto Blade y sus diferentes rotaciones al rededor del eje x

Brain				
Rotación	Eigenvectores			Eigenvalores
0°	0.799	-0.083	0.596	1.27E+07
	0.083	0.997	0.028	1.186E+07
	-0.596	0.027	0.803	9.370E+06
15°	0.819	0.140	0.558	1.275E+07
	-0.012	0.974	-0.226	1.214E+07
	-0.574	0.179	0.799	9.756E+06
35°	0.791	0.441	0.424	1.273E+07
	-0.279	0.877	-0.391	1.220E+07
	-0.544	0.191	0.817	9.567E+06
45°	0.865	0.363	0.348	1.243E+07
	-0.162364	0.856431	-0.490065	1.21832E+07
	-0.475465	0.367	0.780	9.761E+06

Tabla 7.2: Eigenvectores y Eigenvalores del objeto Brain y sus diferentes rotaciones al rededor del eje x

Cow				
Rotación	Eigenvectores			Eigenvalores
0°	1.000	0.010	0.002	2.396E+06
	-0.010	0.971	0.240	1.910E+06
	0.001	-0.239	0.971	755057
15°	0.968	0.245	0.053	2.276E+06
	-0.251	0.946	0.205	1.874E+06
	-0.002	-0.212	0.978	687897
35°	0.833	0.537	0.131	2.393E+06
	-0.552	0.815	0.173	1.966E+06
	-0.013	-0.217	0.976	726407
45°	0.701	0.690	0.181	2.331E+06
	-0.713	0.677	0.183	1.865E+06
	0.003	-0.257	0.966	731025

Tabla 7.3: Eigenvectores y Eigenvalores del objeto Cow y sus diferentes rotaciones al rededor del eje x

Dragon				
Rotación	Eigenvectores			Eigenvalores
0°	0.991	-0.127	0.032	4.689E+06
	0.128	0.886	-0.446	3.427E+06
	0.029	0.446	0.894	2.052E+06
15°	0.993	0.074	-0.092	4.780E+06
	-0.110	0.861	-0.497	3.56E+06
	0.042	0.503	0.863	2.020E+06
35°	0.901	0.342	-0.266	5.000E+06
	-0.431	0.764	-0.480	3.712E+06
	0.040	0.547	0.836	2.097E+06
45°	0.811	0.454	-0.369	5.091E+06
	-0.582	0.694	-0.424	3.817E+06
	0.064	0.558	0.827	2.125E+06

Tabla 7.4: Eigenvectores y Eigenvalores del objeto Dragon y sus diferentes rotaciones al rededor del eje x

Heptoroid				
Rotación	Eigenvectores			Eigenvalores
0°	0.010	0.165	0.986	9.188E+06
	-1.000	0.009	0.009	5.940E+06
	-0.007	-0.986	0.165	4.504E+06
15°	0.245	-0.065	0.967	1.019E+07
	0.969	0.008	-0.248	7.020E+06
	0.008	0.998	0.065	4.641E+06
35°	0.577	-0.0886	-0.812	7.256E+06
	0.817	0.065	0.573	5.965E+06
	0.002	-0.994	0.110	2.495E+06
45°	0.705	-0.025	-0.709	1.033E+07
	0.709	0.012	0.705	7.129E+06
	-0.009	-1.000	0.027	4.720E+06

Tabla 7.5: Eigenvectores y Eigenvalores del objeto Heptoroid y sus diferentes rotaciones al rededor del eje x

Lion				
Rotación	Eigenvectores			Eigenvalores
0°	0.981	0.180	0.072	2.524E+06
	-0.161	0.963	-0.216	2.171E+06
	-0.108	0.200	0.974	690154
15°	0.907	0.421	0.005	2.44624E+06
	-0.409	0.883	-0.229	2.109E+06
	-0.101	0.205	0.974	690819
35°	0.766	0.639	-0.079	2.417E+06
	-0.637	0.734	-0.235	2.112E+06
	-0.092	0.231	0.969	673232
45°	0.626	0.762	-0.165	2.149E+06
	-0.776	0.590	-0.224	1.851E+06
	-0.073	0.268	0.961	654541

Tabla 7.6: Eigenvectores y Eigenvalores del objeto Lion y sus diferentes rotaciones al rededor del eje x

Blade				
Rotación	λ	X	Y	Z
0°	1	-2893195.212	-123398.955	1674250.236
	2	1446621.58	-178606.599	2486813.869
	3	905.289	-1030401.776	-74470.091
15°	1	-2588702.786	453289.911	1659330.726
	2	1382963.040	-567834.303	2312806.298
	3	-201835.050	-839915.265	-85480.010
35°	1	-3498941.516	2202603.971	2321740.280
	2	1639094.134	-1282894.939	3687455.850
	3	-650328.164	-978865.770	-51425.929
45°	1	-3744960.996	3125698.664	1691111.129
	2	955061.162	-1200362.467	4333864.342
	3	-838037.845	-960149.620	-81222.483

Tabla 7.7: Ejes principales de Blade

Brain				
Rotación	λ	X	Y	Z
0°	1	-10152325.504	-1048791.266	7576401.430
	2	984004.621	-11812681.608	-316857.233
	3	-5584089.597	-263226.172	-7519341.393
15°	1	-10435392.119	150104.474	7318645.475
	2	-1692634.683	-11827367.730	-2171157.421
	3	-5434644.056	2208878.658	-7794590.745
35°	1	-10067125.346	3548642.823	6928592.344
	2	-5379918.341	-10702100.957	-2335862.968
	3	-4056523.122	3744560.967	-7812051.901
45°	1	-10751168.080	2019048.540	5912368.252
	2	-4419976.600	-10433972.870	-4474615.508
	3	-3392411.408	4783491.525	-7802558.453

Tabla 7.8: Ejes principales de Brain

Cow

Rotación	λ	X	Y	Z
0°	1	-2395718.294	23820.808	-1001.0512
	2	-18511.343	-1851011.814	456572.376
	3	-1409.266	-180674.023	-732985.673
15°	1	-2202608.841	570232.748	4173.055
	2	-458303.210	-1773533.529	397303.426
	3	-37636.398	-140723.874	-672167.891
35°	1	-1994433.129	1321951.303	32299.056
	2	-1055376.881	-1602814.734	426160.302
	3	-94850.782	-125875.440	-708961.631
45°	1	-1634568.013	1662326.047	-7649.978
	2	-1285788.256	-1262200.491	479654.693
	3	-132322.116	-133419.113	-706323.460

Tabla 7.9: Ejes principales de Cow

Dragon

Rotación	λ	X	Y	Z
0°	1	-4648554.347	-598612.568	-134188.690
	2	435190.990	-3035643.994	-1528862.619
	3	-64776.787	915198.561	-1835264.823
15°	1	-4746938.909	526740.781	-200489.084
	2	-264349.087	-3066334.298	-1793224.718
	3	185108.713	1003802.488	-1743408.398
35°	1	-4506880.550	2156427.308	-197608.208
	2	-1267838.418	-2836317.767	-2031853.991
	3	558435.452	1006356.149	-1752914.444
45°	1	-4128964.485	2960604.828	-325786.032
	2	-1732355.478	-2650791.019	-2131266.824
	3	784366.273	900477.593	-1757210.180

Tabla 7.10: Ejes principales de Dragon

Heptoroid

Rotación	λ	X	Y	Z
0°	1	-93226.137	9186838.618	67824.447
	2	-977084.597	-53049.368	5858192.643
	3	-4442729.823	-39635.236	-741354.686
15°	1	-2521466.953	-9868407.076	-83425.962
	2	455482.243	-56975.087	-7004936.195
	3	-4486543.921	1149048.025	-300879.761
35°	1	-4187017.406	-5925829.317	-14973.814
	2	528275.214	-388187.731	5928854.108
	3	2025883.702	-1430578.014	-274057.323
45°	1	-7283944.943	-7322567.515	89336.048
	2	176192.590	-88239.259	7126754.347
	3	3344468.118	-3328131.744	-123762.787

Tabla 7.11: Ejes principales de Heptoroid

Lion				
Rotación	λ	X	Y	Z
0°	1	-2476320.157	406785.795	272929.745
	2	-391218.083	-2091008.020	-434072.720
	3	-49545.448	148899.443	-671971.930
15°	1	-2218565.090	1000474.734	247044.385
	2	-888154.714	-1863025.287	-432375.818
	3	-3602.512	157943.190	-672418.225
35°	1	-1850298.000	1538812.970	222625.774
	2	-1348361.891	-1550958.525	-487449.855
	3	53501.817	158627.551	-651999.007
45°	1	-1345836.926	1667878.846	156555.119
	2	-1410054.121	-1091320.449	-496524.581
	3	108287.120	146381.664	-628630.814

Tabla 7.12: Ejes principales de Lion

Blade	Brain			
	0°	15°	35°	45°
0°	0	28.841	40.306	31.368
15°	28.841	0	45.300	31.420
35°	40.310	45.300	0	40.661
45°	31.368	31.420	40.661	0

Tabla 7.13: Distancia de Hausdorff entre Blade con sus diferentes rotaciones al rededor del eje x

Blade	Blade			
	0°	15°	35°	45°
0°	87.190	88.743	83.982	88.368
15°	83.834	80.494	77.188	86.756
35°	78.490	78.418	80.392	76.134
45°	80.313	83.359	81.640	79.597

Tabla 7.14: Distancia de Hausdorff entre Blade con Brain

Blade	Cow			
	0°	15°	35°	45°
0°	60.515	62.393	62.721	60.497
15°	57.129	59.012	59.214	59.554
35°	67.335	66.730	69.865	64.837
45°	61.415	61.097	63.681	58.014

Tabla 7.15: Distancia de Hausdorff entre Blade con Cow

Blade	Dragon			
	0°	15°	35°	45°
0°	64.738	67.599	66.021	62.679
15°	62.772	65.847	65.149	68.039
35°	80.422	75.910	74.287	78.422
45°	73.732	71.047	68.710	72.689

Tabla 7.16: Distancia de Hausdorff entre Blade con Dragon

Blade	Heptoroid			
	0°	15°	35°	45°
0°	70.4015	70.105	57.856	69.539
15°	67.821	76.776	63.060	77.3460
35°	67.294	81.830	60.542	66.034
45°	74.711	78.337	77.433	77.154

Tabla 7.17: Distancia de Hausdorff entre Blade con Heptoroid

Blade	Lion			
	0°	15°	35°	45°
0°	80.025	83.646	74.103	75.203
15°	78.113	80.676	71.375	76.222
35°	95.876	98.564	90.830	92.377
45°	89.856	93.358	85.457	89.287

Tabla 7.18: Distancia de Hausdorff entre Blade con Lion

Brain	Blade			
	0°	15°	35°	45°
0°	87.190	83.834	78.490	80.313
15°	88.743	80.494	78.418	83.359
35°	83.982	77.188	80.392	81.640
45°	88.368	86.756	76.134	79.597

Tabla 7.19: Distancia de Hausdorff entre Brain con Blade

Brain	Brain			
	0°	15°	35°	45°
0°	0	25.264	27.871	36.522
15°	25.264	0	24.241	35.760
35°	27.871	24.241	0	24.848
45°	36.522	35.760	24.848	0

Tabla 7.20: Distancia de Hausdorff entre Brain con sus diferentes rotaciones al rededor del eje x

Brain	Cow			
	0°	15°	35°	45°
0°	78.394	77.432	79.045	77.479
15°	78.201	77.024	78.983	77.407
35°	75.931	74.810	76.961	75.171
45°	78.841	76.266	78.212	76.326

Tabla 7.21: Distancia de Hausdorff entre Brain con Cow

Brain	Dragon			
	0°	15°	35°	45°
0°	93.928	94.534	93.834	96.924
15°	92.044	92.963	94.284	95.016
35°	90.580	92.075	89.994	89.184
45°	96.079	93.428	91.788	93.436

Tabla 7.22: Distancia de Hausdorff entre Brain con Dragon

Brain	Heptoroid			
	0°	15°	35°	45°
0°	91.578	87.336	88.273	92.174
15°	87.494	85.141	84.555	90.090
35°	84.708	81.894	84.857	85.853
45°	89.779	85.319	81.450	86.833

Tabla 7.23: Distancia de Hausdorff entre Brain con Heptoroid

Brain	Lion			
	0°	15°	35°	45°
0°	81.117	81.544	78.550	81.020
15°	78.901	79.880	81.286	81.661
35°	77.480	78.674	79.723	80.273
45°	77.492	78.262	80.493	81.873

Tabla 7.24: Distancia de Hausdorff entre Brain con Lion

Cow	Blade			
	0°	15°	35°	45°
0°	60.515	57.129	67.335	61.415
15°	62.393	59.011552	66.730	61.097
35°	62.721	59.214	69.865	63.681
45°	60.497	59.554	64.837	58.014

Tabla 7.25: Distancia de Hausdorff entre Cow con Blade

Cow	Brain			
	0°	15°	35°	45°
0°	78.394	78.201	75.931	78.841
15°	77.432	77.024	74.810	76.266
35°	79.045	78.983	76.961	78.212
45°	77.479	77.407	75.171	76.326

Tabla 7.26: Distancia de Hausdorff entre Cow con Brain

Cow	Cow			
	0°	15°	35°	45°
0°	0	23.407	19.419	19.542
15°	23.407	0	22.740	15.988
35°	19.419	22.740	0	18.779
45°	19.542	15.988	18.779	0

Tabla 7.27: Distancia de Hausdorff entre Cow con sus diferentes rotaciones al rededor del eje x

Cow	Dragon			
	0°	15°	35°	45°
0°	55.711	52.457	48.511	47.243
15°	54.324	50.447	46.039	51.183
35°	55.297	52.463	48.359	47.542
45°	53.468	49.427	44.940	48.510

Tabla 7.28: Distancia de Hausdorff entre Cow con Dragon

Cow	Heptoroid			
	0°	15°	35°	45°
0°	73.450	74.862	56.296	73.466
15°	73.929	72.998	55.907	72.976
35°	74.572	72.997	58.415	73.979
45°	70.622	74.755	53.781	70.535

Tabla 7.29: Distancia de Hausdorff entre Cow con Heptoroid

Cow	Lion			
	0°	15°	35°	45°
0°	46.521	47.551	46.810	45.799
15°	47.857	50.200	50.437	49.906
35°	47.050275	46.267	45.562	48.536
45°	47.440441	50.710	50.400	44.524

Tabla 7.30: Distancia de Hausdorff entre Cow con Lion

Dragon	Blade			
	0°	15°	35°	45°
0°	64.738	62.772	80.422	73.732
15°	67.599	65.847	75.910	71.047
35°	66.021	65.149	74.287	68.710
45°	62.679	68.039	78.422	72.689

Tabla 7.31: Distancia de Hausdorff entre Dragon con Blade

Dragon	Brain			
	0°	15°	35°	45°
0°	93.928	92.044	90.580	96.080
15°	94.534	92.963	92.075	93.428
35°	93.834	94.284	89.994	91.788
45°	96.924	95.016	89.184	93.436

Tabla 7.32: Distancia de Hausdorff entre Dragon con Brain

Dragon	Cow			
	0°	15°	35°	45°
0°	55.711	54.324	55.297	53.467
15°	52.457	50.447	52.463	49.426
35°	48.511	46.038	48.359	44.940
45°	47.243	51.183	47.542	48.510

Tabla 7.33: Distancia de Hausdorff entre Dragon con Cow

Dragon	Dragon			
	0°	15°	35°	45°
0°	0	20.514	24.290	24.307
15°	20.514	0	23.223	22.681
35°	24.290	23.223	0	16.258
45°	24.307	22.681	16.258	0

Tabla 7.34: Distancia de Hausdorff entre Dragon con sus diferentes rotaciones al rededor del eje x

Dragon	Heptoroid			
	0°	15°	35°	45°
0°	59.040	55.488	50.167	66.163
15°	62.343	54.599	52.704	62.791
35°	62.422	55.208	53.135	60.401
45°	62.469	57.823	51.426	59.053

Tabla 7.35: Distancia de Hausdorff entre Dragon con Heptoroid

Dragon	Lion			
	0°	15°	35°	45°
0°	51.084	51.861	53.556	54.126
15°	54.774	55.163	47.771	48.682
35°	56.078	55.871	49.639	48.653
45°	59.311	59.079	51.825	48.707

Tabla 7.36: Distancia de Hausdorff entre Dragon con Lion

Heptoroid	Blade			
	0°	15°	35°	45°
0°	70.401	67.821	67.294	74.711
15°	70.104	76.776	81.830	78.337
35°	57.855	63.060	60.542	77.432
45°	69.539	77.350	66.034	77.154

Tabla 7.37: Distancia de Hausdorff entre Heptoroid con Blade

Heptoroid	Brain			
	0°	15°	35°	45°
0°	91.578	87.494	84.708	89.778
15°	87.336	85.141	81.894	85.319
35°	88.273	84.555	84.857	81.450
45°	92.174	90.090	85.853	86.833

Tabla 7.38: Distancia de Hausdorff entre Heptoroid con Brain

Heptoroid	Cow			
	0°	15°	35°	45°
0°	73.450	73.929	74.572	70.622
15°	74.862	72.998	72.997	74.755
35°	56.296	55.907	58.415	53.780
45°	73.466	72.976	73.978	70.535

Tabla 7.39: Distancia de Hausdorff entre Heptoroid con Cow

Heptoroid	Lion			
	0°	15°	35°	45°
0°	76.034	75.613	74.563	76.434
15°	74.215	74.449	77.166	74.740
35°	59.224	62.576	60.141	56.617
45°	73.302	73.453	71.990	73.962

Tabla 7.40: Distancia de Hausdorff entre Heptoroid con Lion

Referencias

[1] Chen Li. *Discrete Surfaces and Manifolds: A Theory of Digital-Discrete Geometry and Topology*. Scientific and Practical Computing, 2004.

[2] Klette R. and Rosenfeld A. *Digital Geometry*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, San Francisco, 2004.

[3] Latecki Logan Jan. *Discrete Representation of Spatial Objects in Computer Vision*. Kluwer Academic Publisher, 1998.

[4] Chen Li M. *Digital Functions and Data Reconstruction: Digital-Discrete Methods*. Springer Publishing Company, Incorporated, 2012.

[5] Gonzalez Rafael C. and Woods Richard E. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008.

[6] Theo Pavlidis. *Algorithms for Graphics and Image Processing*. Computer Science Press, Rockville, USA, 1982.

[7] Boucetta A. and Melkemi K. E. Hand shape recognition using hu and legendre moments. In *Proceedings of the 6th International Conference on Security of Information and Networks*, SIN '13, pages 272–276, New York, NY, USA, 2013. ACM.

[8] Pattanachai N., Covavisaruch N., and Sinthanayothin C. Tooth recognition in dental radiographs via hu's moment invariants. In *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pages 1–4, May 2012.

[9] Sánchez-Cruz H. and Rodríguez-Dagnino R.M. Compressing bilevel images by means of a three-bit chain code. *Optical Engineering*, 44:44 – 44 – 8, 2005.

[10] Yong K.L. and Alik B. An efficient chain code with huffman coding. *Pattern Recognition*, 38(4):553 – 557, 2005.

[11] Echávarri L., Aguinaga R., Neri-Calderón A., and Rodriguez-Dagnino R.M. Compression rates comparison of entropy coding for three-bit chain codes of bilevel images. *Optical Engineering*, 46:46 – 46 – 7, 2007.

[12] Yong K.L., Wei W., Peng J.W., and Alik B. Compressed vertex chain codes. *Pattern Recognition*, 40(11):2908 – 2913, 2007.

[13] Freeman H. Computer processing of line-drawing images. *ACM Comput. Surv.*, 6(1):57–97, March 1974.

[14] Bribiesca E. A chain code for representing 3d curves. *Pattern Recognition*, 33(5):755 – 765, 2000.

[15] Sánchez-Cruz H., López-Valdez H., and Cuevas F.J. A new relative chain code in 3d. *Pattern Recogn.*, 47(2):769–788, February 2014.

[16] Salazar J.M. and Bribiesca E. Compression of three-dimensional surfaces by means of chain coding. *Optical Engineering*, 54:54 – 54 – 12, 2015.

[17] Chau C. P. and Siu W. C. New nonparametric dominant point detection algorithm. *IEE Proceedings - Vision, Image and Signal Processing*, 148(5):363–374, Oct 2001.

[18] Cronin Terence M. A boundary concavity code to support dominant point detection. *Pattern Recognition Letters*, 20(6):617 – 634, 1999.

[19] Marji Majed and Siy Pepe. A new algorithm for dominant points detection and polygonization of digital curves. *Pattern Recognition*, 36(10):2239 – 2251, 2003.

[20] Park Hyungjun and Lee Joo-Haeng. Error-bounded b-spline curve approximation based on dominant point selection. In *International Conference on Computer Graphics, Imaging and Visualization (CGIV'05)*, pages 437–446, July 2005.

[21] Rattarangsi A. and Chin R. T. Scale-based detection of corners of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(4):430–449, April 1992.

[22] Sarfraz Muhammad, Asim M.R., and Masood Asif. Piecewise polygonal approximation of digital curves. volume 8, pages 991– 996, 08 2004.

[23] Attneave Fred. Attneave, f. informational aspects of visual perception. psychol. rev. 61, 183-193. *Psychological review*, 61:183–93, 06 1954.

[24] Tangelder Johan W. H. and Remco C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441, Dec 2007.

[25] Min Patrick, Kazhdan Michael, and Funkhouser Thomas. A comparison of text and shape matching for retrieval of online 3d models. In Rachel Heery and Liz Lyon, editors, *Research and Advanced Technology for Digital Libraries*, pages 209–220, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[26] Besl Paul J. and Jain Ramesh C. Three-dimensional object recognition. *ACM Comput. Surv.*, 17(1):75–145, March 1985.

[27] Loncaric Sven. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983 – 1001, 1998.

[28] Campbell Richard J. and Flynn Patrick J. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81(2):166 – 210, 2001.

[29] Veltkamp Remco C. and Hagedoorn Michiel. *State of the Art in Shape Matching*, pages 87–119. Springer London, London, 2001.

[30] Iyer Natraj, Jayanti Subramaniam, Lou Kuiyang, Kalyanaraman Yagnanarayanan, and Ramani Karthik. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509 – 530, 2005. Geometric Modeling and Processing 2004.

[31] Tangelder J. W. H. and Veltkamp R. C. A survey of content based 3d shape retrieval methods. In *Proceedings Shape Modeling Applications, 2004.*, pages 145–156, June 2004.

[32] Haralick Robert M. and Shapiro Linda G. Glossary of computer vision terms. *Pattern Recogn.*, 24(1):69–93, December 1990.

[33] Taha Abdel Aziz and Hanbury Allan. An efficient algorithm for calculating the exact hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(11):2153–2163, November 2015.

[34] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[35] Fu Rong and Shen Hong. An algorithm for determining concave vertex of object based on vector product. *Journal of Electronics (China)*, 27(2):212–217, Mar 2010.

[36] Nadeem Syed Ahmed, Hoffman Eric A., and Saha Punam K. Path-gradient – a theory of computing full intensity-transition between two points. In Marcelo Mendoza and Sergio Velastín, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 399–407, Cham, 2018. Springer International Publishing.

[37] Sánchez-Cruz H. and Bribiesca E. A method of optimum transformation of 3d objects used as a measure of shape dissimilarity. *Image and Vision Computing*, 21(12):1027 – 1036, 2003.

[38] Eugene Xavier S.P. *Theory of Automata, formal languages and computation.* New age international (P) Limited, University Bookstore, B-74,new delhi, India, 1st ed. edition, 2005.

[39] Freeman H. On the encoding of arbitrary geometric configurations. 10:260 – 268, 07 1961.

[40] Bribiesca E. A new chain code. *Pattern Recognition*, 32(2):235 – 251, 1999.

[41] Bribiesca Ernesto and Guzman Adolfo. How to describe pure form and how to measure differences in shapes using shape numbers. *Pattern Recognition*, 12(2):101 – 112, 1980.

[42] Sánchez-Cruzand H. and Lopez H. Equivalence of chain codes. *journal of electronic imaging*, 23:1–11, 02 2014.

[43] Hermilo Sánchez-Cruz, Marcos López-Cruces, and Hector Puga. A proposal modification of the 3ot chain code. In *Proceedings of the Tenth IASTED International Conference on Computer Graphics and Imaging*, CGIM '08, pages 6–11, Anaheim, CA, USA, 2008. ACTA Press.

[44] Sánchez-Cruz H. Proposing a new code by considering pieces of discrete straight lines in contour shapes. *Journal of Visual Communication and Image Representation*, 21(4):311 – 324, 2010.

[45] Wood Jeffrey. Invariant pattern recognition: A review. *Pattern Recognition*, 29:1–17, 01 1996.

[46] Gurevich G. B. *Foundations of the Theory of Algebraic Invariants.* Groningen, The Netherlands: Nordhoff, 1st ed. edition, 1964.

[47] Hilbert David. *Theory of Algebraic Invariants.* Cambridge: CambridgeUniversity Press, 1st ed. edition, 1993.

[48] Schur Issai. *Vorlesungen über Invariantentheorie.* Springer-Verlag, 1st ed. edition, 1968.

[49] Hu Ming-Kuei. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, February 1962.

[50] Karush W. *Webster's New World Dictionary of Mathematics.* Simon & Schuster, Inc., New York, 1989.

[51] Sadjadi F. A. and Hall E. L. Three-dimensional moment invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(2):127–136, March 1980.

[52] Lo C. and Don H. 3-d moment forms: their construction and application to object identification and positioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1053–1064, Oct 1989.

[53] Faber T. L. and Stokely E. M. Orientation of 3-d structures in medical images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):626–633, Sep. 1988.

[54] Borisenko A. I. and Tarapov I. E.

[55] Cui S.G., Wang H., and Yang L. A simulation study of a-star algorithm for robot path planning. pages 506–509, 01 2012.

[56] Duchó F., Babinec A., Kajan M., Be P., Florek M., Fico T., and Juri L. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59 – 69, 2014. Modelling of Mechanical and Mechatronic Systems.

[57] Masood Asif and Haq Shaiq A. A novel approach to polygonal approximation of digital curves. *J. Vis. Comun. Image Represent.*, 18(3):264–274, June 2007.

[58] Nasser Hayat, Ngo Phuc, and Debled-Rennesson Isabelle. Dominant point detection based on discrete curve structure and applications. *Journal of Computer and System Sciences*, 95:177–192, 2018.

[59] Masood Asif. Dominant point detection by reverse polygonization of digital curves. *Image Vision Comput.*, 26(5):702–715, May 2008.

[60] Madrid-Cuevas F.J., Aguilera-Aguilera E.J., Carmona-Poyato A., Muñoz-Salinas R., Medina-Carnicer R., and Fernández-García N.L. An efficient unsupervised method for obtaining polygonal approximations of closed digital planar curves. *Journal of Visual Communication and Image Representation*, 39:152 – 163, 2016.

8 Glosario

- **Alfabeto**.- Se define como un conjunto finito de símbolos.
- **Algoritmo**.- Un algoritmo es una secuencia de pasos instructivos ordenados para resolver un problema. El algoritmo puede diseñarse para que sea muy complicado si no podemos encontrar una fórmula o ecuación simple para resolver el problema.
- **Cadena**.- Una cadena C es una secuencia ordenada de elementos, y está representada por, $C = "a_1a_2 \ldots a_n = \{a_i : 1 \geq i \geq n\}$.
- **Código**.- Consideremos el CÓDIGO de un alfabeto de un número finito de símbolos, CÓDIGO $= \{a_1, a_2, \ldots, a_n\}$
- **Componentes Conectados**.- Cualquier elemento de la malla está conectado a sí mismo, y dos elementos p y q de la malla están conectados si existe una secuencia de elementos $(p_0, \ldots, p_n)$ donde $n \geq 0$ tal que $p_0 = p, p_n = q$, y p_{i+1} es adyacente a $p_i (0 \leq i \leq n)$. Los elementos p y q se llaman conectados si hay un camino de p a q. En particular, para la adyacencia α ($\alpha \in \{0, 1, 2, 3, 4, 6, 8, 18, 26\}$), usamos los términos "camino α" y "α-conectado".
- **Contorno**.- El contorno de S es el conjunto de pixeles de S que tienen 4-vecinos en $\overline{S}$.
- **Curva**.- Una curva simple es una curva en la que cada punto p tiene un índice de bifurcación 2. Un arco simple es una curva en la que cada punto p tiene un índice de bifurcación 2 excepto por dos puntos finales, que tienen un índice de bifurcación 1, o una curva simple con uno de sus puntos etiquetados como un punto final.
- **Distancia**.- En geometría euclidiana: la distancia entre dos puntos es igual a la longitud del segmento de línea recta definido por los dos puntos.
- **Eficiencia**.- Para colecciones de formas grandes, es ineficiente hacer coincidir secuencialmente todos los objetos en la base de datos con el objeto de consulta. Debido a que la recuperación debe ser rápida, se necesitan estructuras de búsqueda de indexación eficientes para respaldar la recuperación eficiente.

- **Emparejamiento parcial**.- A diferencia de la concordancia de formas global, la concordancia parcial encuentra una forma de la cual una parte es similar a una parte de otra forma.

- **Forma**.- La forma se refiere a la forma del objeto, y se considera que un objeto es una entidad geométrica compuesta de celdas o poliedros, para 2D o 3D, respectivamente.

- **Geometría Digital**.- La geometría digital es el estudio de propiedades geométricas o topológicas de conjuntos de píxeles o voxeles. Con frecuencia, intenta obtener información cuantitativa sobre los objetos analizando imágenes digitalizadas (2D o 3D) en las que los objetos están representados por tales conjuntos.

- **Lenguaje**.- Es cualquier conjunto de cadenas sobre un alfabeto Σ

- **Morfología matemática**.- La morfologá matemática se ocupa de operaciones que reemplazan el valor de un pixel p con el máximo o mínimo de los valores de un conjunto de vecinos de p.

- **Normalización de postura**.- A falta de conocimientos previos, los modelos 3D tienen una escala, orientación y posición arbitrarias en el espacio 3D. Dado que no todas las medidas de disimilitud son invariantes en la rotaciń y la traslación, puede ser necesario colocar los modelos 3D en un sistema de coordenadas canónico. Este debe ser el mismo para una copia trasladada, rotada o escalada del modelo.

- **Poder discriminativo**.- Un descriptor de forma debe capturar las propiedades que discriminan bien los objetos. Sin embargo, el juicio de la similitud de las formas de dos objetos 3D es algo subjetivo, dependiendo de la preferencia del usuario o la aplicación en cuestión.

- **Polígono**.- Un arco poligonal (finito, conectado) en el espacio euclidiano $\mathbb{E}^n$ es una secuencia finita de puntos $(p_1, p_2, \ldots, p_n)$ donde $n \geq 3$, que define $n-1$ segmentos de línea recta $p_i p_{i+1} (i = 1, 2, \ldots, n-1)$. Los p_is se denominan vértices del arco y los segmentos de línea se denominan bordes. El arco poligonal forma un circuito $\langle p_1, p_2, \ldots, p_n \rangle$ si agregamos el borde np_np_1.

- **Punto**.- Un punto indica la posición en el espacio y no tiene tamaño ni magnitud.

- **Robustez**.- A menudo es deseable que un descriptor de forma sea insensible al ruido y pequeñas características adicionales, y robusto contra las degeneraciones topológicas arbitrarias, por ejemplo. Si se obtiene mediante escaneo láser. Además, si un modelo se ofrece en múltiples niveles de detalle, las representaciones de diferentes niveles no deben diferir significativamente del modelo original.

- **Subcadena**.- z es una subcadena de w si z aparece consecutivamente dentro de w.

- **Topología**.- El origen de la topología se identifica a menudo con el teorema de Descartes-Euler $\alpha_0 - \alpha_1 + \alpha_2 = 2$, que se estableció originalmente con respecto a los números $\alpha_2, \alpha_1 y \alpha_0$ de caras, bordes, y

vértices de un poliedro convexo. (Un *poliedro* convexo es un conjunto delimitado no vacío que es una intersección finita de muchos espacios intermedios).

- **Vector**.- Un punto de malla p puede identificarse con un vector p = $\overrightarrow{op}$ que comienza en el origen o y termina en p; También podemos considerar los vectores $\overrightarrow{pq}$ de un punto de malla a otro.
  El espacio vectorial 2D $[\mathbb{R}^2, +, \cdot, \mathbb{R}]$ sobre los números reales se define mediante una operación de suma $(x_1, x_2)+(y_1, y_2) = (x_1+y_1, x_2+y_2)$ para todos $(x_1, y_1), (x_2, y_2)$ en $\mathbb{R}^2$ y una operación de multiplicación escalar a $\cdot(x, y) = (a_x, a_y)$ para todo $a \in \mathbb{R}$ y todo $(x, y) \in \mathbb{R}$. Los espacios vectoriales de dimensión superior se definen de manera análoga utilizando n-tuplas en lugar de pares.