Universidad Nacional Autónoma de México

Facultad de Ciencias

Cálculo del perímetro de curvas digitales usando geometría de espacios localmente finitos.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
PEDRO VEGA GALAVIZ

DIRECTOR DE TESIS:
DRA. MARÍA DE LUZ GASCA SOTO

Mexico, D.F. 2014

Índice general

Índice de figuras

Introducción

El presente trabajo ha sido desarrollado a partir de la investigación del Dr. Kovalevsky, quien ha publicado diversos artículos en el campo de la Topología y Geometría digital. El trabajo de Kovalevsky es un interesante enfoque con estrictos fundamentos teóricos de la Topología digital clásica, la cual fue iniciada por A. Rosenfeld. La teoría propuesta por Rosenfled funcionaba a pesar de presentar problemas con los conjuntos conexos. Sin embargo, fue este problema el que llevó a otros autores a buscar una nueva solución, en paricular, el Dr. Kovalevsky desarrolló una nueva teoría basada en la topología digital de los espacios localmente finitos.

Los objetivos del presente trabajo son los de conocer estos nuevos conceptos y teoría para introducir dos nuevos algoritmos para el cálculo de la longitud de curvas digitales o el perímetro de regiones digitales.
En la literatura relacionada con el análisis de imágenes, se recomiendan diferentes métodos para el cálculo de la longitud de curvas digitales o el perímetro de regiones digitales; sin embargo, estos métodos son imprecisos y, aún más importante, la precisión de estos algoritmos no puede ser mejorada incrementando la resolución de la curva o la región digital.
El enfoque de Kovalevsky facilita la representación de los objetos en la computadora; además, permite evitar el problema de los conjunto conexos. Lo anterior permite analizar, diseñar e implementar los algoritmos usando la topología digital de espacios finitos. Los dos metodos que se analizan son:

(i) El método DDS, por sus siglas en inglés, el cual se basa en una partición de la curva digital en segmentos digitales de líneas rectas.

(ii) El método MLP, por sus siglas en inglés, el cual está caracterizado por el cálculo del polígono de mínima longitud en una frontera abierta de una región digital.

Para ambas técnicas se sabe que la longitud de la curva mesurable converge hacia el verdadero valor si una región Euclidiana es digitalizada con el aumento de resolución.

La estructura de la tesis es la siguiente: en el primer capítulo se presenta una breve explicación de los antecedentes del trabajo.

En el segundo capítulo se dan las bases teóricas de la topología digital de los espacios localmente finitos, comenzando por definir los axiomas, definiciones de esta teoría y la relación entre estos nuevos axiomas y los de la topología digital clásica. Se verán las propiedades de los espacios localmente finitos, los cuales darán las bases teóricas para los

complejos de celdas abstractas, concepto que se verá en ese mismo capítulo.

En el tercer capítulo se explicará la teoría requerida para entender cada uno de los algoritmos que trataremos en la tesis, utilizando los conceptos definidos en el capítulo anterior. En el caso del algoritmo DDS se definen los segmentos digitales de líneas rectas. Para el algoritmo MLP se define el polígono de longitud mínima.

En el cuarto capítulo se da una implementación de ambos algoritmos mencionando los detalles que se realizaron de cada uno, presentando el pseudocódigo y ejemplos de ambos.

En el Apéndice A se presentan los conceptos básicos de la topología clásica que son necesarios para poder entender la topología digital clásica y la topología digital de los espacios localmente finitos, conceptos como frontera, vecindad conjuntos abiertos y cerrados, además de la definición de espacios topológicos.

En el Apéndice B se dan los conceptos básicos de la topología digital para conocer la teoría que antecedió a la de los espacios localmente finitos.

En el Apéndice C se estudian los conceptos relacionados con la orientación de los complejos de celdas abstractas que son particularmente necesarios para poder describir las estructuras de datos que la topología de los complejos de celdas abstractas utilizan para poder aplicar los conceptos nuevos de celdas de manera más eficiente.

En el Apéndice D se dan los conceptos de la orientacion de los complejos. Estas deficiones son necesarias para poder establecer las estructuras de datos que se ven en el siguiente Apéndice.

En el Apéndice E se analizan las estructuras de datos con las cuales podemos almacenar las imágenes digitalizadas para su manipulación posterior, además de que se proporcionan estrucuturas de datos especificamente creadas para poder utilizar y manipular de forma eficiente las celdas.

Capítulo 1

Antecedentes

La Geometría digital es el estudio de las propiedades geométricas de los objetos digitalizados o imágenes digitalizadas, se encarga de las definiciones de las propiedades así como de los algoritmos para calcular o determinar las propiedades, por ejemplo la convexidad digital, la rectitud digital, o planaridad digital.

La Topología digital se encarga del estudio de las propiedades de naturaleza topológica, particularmente propiedades que involucren conceptos de conexidad o propiedades de adyacencia, pero que no dependen del tamaño o la forma y proporciona algoritmos que calculan o preservan estas propiedades, por ejemplo la conectividad y las fronteras de los objetos digitales. La topología digital se fundamenta en la topología algebraica. Las propiedades topológicas juegan un papel importante en el análisis de las imágenes en 2 y 3 dimensiones. Uno de los autores más prominentes de este campo fue el Dr. Azriel Rosenfeld, quien es considerado el fundador tanto de la geometría digital como de la topología digital. En la geometría digital a cada pixel o voxel se le asocia un punto con valor 1 o 0. Si un punto tiene valor 0 se le llama punto blanco y si tiene valor 1 se le llama punto negro. Con esto se estableció el concepto de adyacencia, que se ha descrito en el Apéndice B.1, pag. 61. Sin embargo, su teoría presentó algunas dificultades relacionadas con la definición de conjuntos conexos, conocida como la paradoja de Jordan.

La paradoja de Jordan básicamante consiste en que si para todo par de pixeles o voxeles, si usamos la $4-$adyacencia, entonces los puntos negros están totalmente desconectados, en cambio, si usamos la $8-$adyacencia, entonces los puntos negros forman una curva de Jordan pero no separan a los puntos blancos. Sin embargo, la solución de la paradoja consiste en usar una adyancencia para los puntos negros y otra para los puntos blancos, así si los puntos negros usan $8-$adyacencia, los puntos blancos usan $4-$adyacencia o viceversa.

A pesar de que se propuso la anterior manera de poder sobrellevar estos problemas, el Dr. Kovlevsky quedó insatisfecho con esta solución, debido a que ésta no se podía aplicar a imágenes de color y al hecho de que se cuenta con tantas y diferentes definiciones de fronteras de las regiones, ya que al usar distintos tipos de adyacencias, la frontera varía según la adyacencia que se use.

Debido a lo anterior se trató de encontrar una mejor solución; sin embargo, este inten-

to no resultó ser exitoso. Fue entonces que se presentó un nuevo concepto de geometría digital, el cual está basado en la topología de los espacios localmente finitos y además es independiente de la geometría euclidiana.

Es aquí que empezamos a introducir el concepto de complejos de celdas abstractas. Veamos que los complejos de celdas abstractas o complejos AC son un caso particular de los espacios localmente finitos. Un complejo abstracto es un conjunto de celdas abstractas (el concepto de celdas abstractasse dara en la siguientes secciones) con un entero no negativo asignado a cada celda, a este valor se le llama la dimensión de la celda. El conjunto es provisto con una relación binaria asimétrica, irreflexiva y transitiva llamada relación de ligado. Dada esta relación de ligado, una celda sólo puede ligar a otra celda de mayor dimensión. Un complejo abstracto es un espacio topológico de acuerdo con los axiomas clásicos. Una gran ventaja de utilizar complejos abstractos es que al ser un espacio topológico finito puede representarse directamente en la computadora. Todas las propiedades topológicas de los complejos y sus elementos pueden ser calculadas directamente en la computadora.

Capítulo 2

Topología Digital de Espacios Finitos

En este capítulo empezaremos por definir los nuevos axiomas que utilizaremos en la teoría de los espacios topológicos aplicables en el cálculo del procesamiento de imágenes. Se iniciará definiendo nuevos conjuntos de axiomas, además de que se presentan algunos teoremas y consecuencias de esta teoría. Continuaremos con la demostración de que un espacio topológico que satisface los axiomas propuestos es un caso particular de un espacio topológico clásico, mostraremos que las propiedades más importantes de los espacios localmente finitos satisfacen los axiomas de los espacio ALF y que la relación de vecindad debe ser antisimétrica y transitiva.

Cabe mencionar que conceptos básicos de Topología son descritos en el Apéndice A y de manera similar conceptos elementales de la Topología Digital son expuestos en el Apéndice B.

2.1. Axiomas de la Topología Digital.

Se comenzará por definir los axiomas y definiciones de topología que necesitaremos, los cuales son diferentes a los de la topología clásica. Un espacio que es explícitamente representado en la computadora es llamado espacio localmente finito. Cada uno de los elementos de un espacio localmente finito tiene una vecindad que consiste de un número finito de elementos. A estos elementos no se les denomina puntos, a diferencia de la Topología Digital Clásica, ya que un espacio localmente finito debe poseer elementos con diversas propiedades topológicas, las cuales tienen diferentes notaciones. Los conceptos anteriores se definiran a continuación; este material está basado en el libro *Geometry of Locally Finite Spaces* del Dr. Kovalevsky.

2.1.1. Axiomas y definiciones

Definición 2.1 (Espacio Localmente Finito)
Se dice que un espacio S es **localmente finito** si para cada elemento $e \in S$ es posible asignarle a e subconjuntos denominados vecindades, los cuales pueden ser finitos.

Consideraremos un caso particular de un espacio localmente finito que satisface los siguientes axiomas, al cual denotaremos como un **espacio** ALF.

Axioma 2.1 Para todo elemento e del espacio S existen subconjuntos que contienen a e denonimados vecindades de e. La intersección de dos vecindades de e es, nuevamente, una vecindad de e.

Ya que el espacio es localmente finito, existe la **vecindad más pequeña** de e, ésta es la intersección de todas las vecindades de e. Entonces cada vecindad de e contiene a la vecindad más pequeña, que denotaremos como $SN(e)$.

Axioma 2.2 Hay elementos en el espacio que en su vecindad más pequeña tiene al menos un elemento.

Definición 2.2 (Incidencia)
Si $b \in SN(a)$ ó $a \in SN(b)$ entonces los elementos a y b son llamados **incidentes** el uno del otro.

La definición anterior nos dice que si b está en la vecindad más pequeña de a o bien si a está en la vecindad más pequeña de b entonces a y b son incidentes. De esta forma, tenemos que la relación de incidencia es simétrica y reflexiva pues $a \in SN(a)$

Definición 2.3 (Trayectoria de incidencia)
Sea T un subconjunto del espacio S. Una secuencia $a_1, a_2, ..., a_k$, donde $a_i \in T$, $1 \leq i \leq k$, en la cual cada dos elementos subsecuentes son incidentes el uno al otro es llamada una **trayectoria de incidencia** en T de a_1 a a_k.

Definición 2.4 (Conexo)
Se dice que, elementos incidentes son **directamente conexos**. Un subconjunto T del espacio S es **conexo** si y sólo si para cada dos elementos de T existe una trayectoria de incidencia que los contiene. Tal trayectoria está completamente en T.

Definición 2.5 (Frontera)
El **borde topológico**, también llamado **frontera**, de un subconjunto no vacío T es el conjunto de todos los elementos e de S, tal que cada vecindad de e contiene elementos tanto de T como del complemento $S \setminus T$.

Denotaremos a la frontera de $T \subseteq S$ como $Fr(T, S)$ y la llamaremos la frontera de T en S. En el caso de un espacio localmente finito es posible reemplazar cada vecindad por la vecindad más pequeña, pues de acuerdo al Axioma 2.1 cada vecindad de a contiene la vecindad más pequeña de a.

Definición 2.6 (Relación de vecindad)
La relación de vecindad N es una relación binaria en el conjunto de elementos del espacio S. El par ordenado (a, b) está en N si y sólo si $a \in SN(b)$.

También podemos escribir aNb para (a, b) en N.

Definición 2.7 (Oponentes)
Los elementos de un par (a, b) en la frontera $Fr(T, S)$ de un subconjunto $T \subset S$ son **oponentes** el uno del otro si a pertenece a $SN(b)$, b pertenece a $SN(a)$, uno de ellos pertenece a T y el otro a su complemento $S \setminus T$.

Definición 2.8 (Frontera gruesa y delgada)
La frontera $Fr(T, S)$ de un subconjunto T de un espacio S es llamada **gruesa** si contiene al menos un par de oponentes. De otra manera es llamada **delgada**.

En las localidades donde hay pares de oponentes en la frontera se dice que la frontera es doble: Hay dos subconjuntos de la frontera los cuales son paralelos el uno al otro. Estos subconjuntos son llamados **borde** y **co-borde**. En la Figura 2.1 se muestran algunos ejemplos de las fronteras, en la parte inferior de la misma Figura se muestran las vecindades más pequeñas de los diferentes elementos de los espacios.

En la Figura 2.1(a) el espacio S consiste de solo cuadrados, el subconjuntoc T es el conjunto de cuadrados coloreados. Los elementos de la frontera $Fr(T, S)$ estan marcados con circulos blancos o negros; los cuadrados con circulos negros petenecen a T, mientras que los cuadrados con circulos blancos pertenecen a $S \setminus T$. Cada elemento de la frontera tiene al menos un oponente, los pares de oponentes se muestran conectados por lineas. De acuerdo a la Definición 2.8 esta frontera es gruesa. Por otro lado en la Figura 2.1(b) el espacio consiste de puntos, representados por circulos, lineas y cuadrados, el subconjunto T es el conjunto de los cuadrados coloreados, las lineas no punteadas y los puntos negros. La frontera $Fr(T, S)$ esta compuesta solo de los circulos negros grandes y las lineas gruesas. La linea gruesa $K \in T$ y el circulo blanco $Q \in S \setminus T$ pertenecen a diferentes subconjuntos por lo tanto no son oponentes ya que $Q \notin SN(K)$.

Axioma 2.3 La frontera $Fr(T, S)$ de cualquier subconjunto $T \subset S$ es delgada.

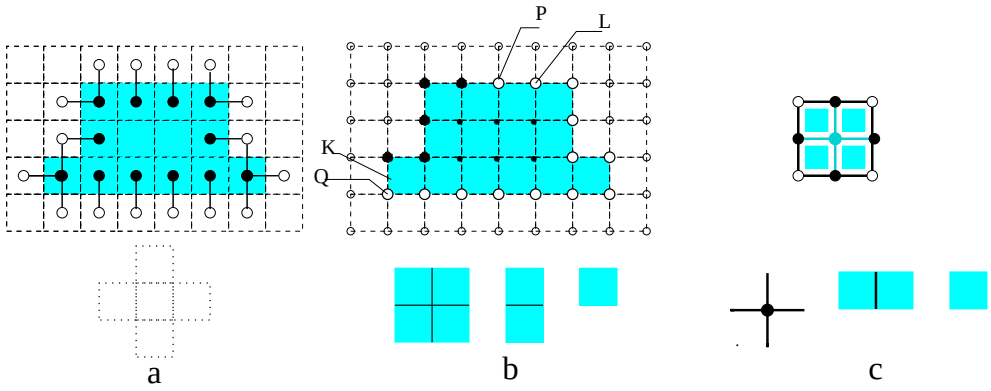

Figura 2.1: Ejemplo de fronteras.
(a)Una frontera gruesa (b)una frontera delgada (c)una frontera con vacíos.

De acuerdo a la Definición 2.5, la frontera de T es la misma a la frontera de su complemento $S \setminus T$. Otra propiedad importante de la frontera es que no tiene vacíos. Esto significa que la frontera de una frontera F es la misma que F. Se debe observar que la frontera $Fr(F, S)$, con $F = Fr(T, S)$, es diferente de F sólo si la relación de vecindad es no transitiva, esto es importante para demostrar que las vecindades más pequeñas que satisfacen nuestros axiomas son subconjuntos abiertos del espacio. La Figura 2.1(c) tiene vacios representados por los circulos blancos. La Figura muestra un espacio S que consiste de puntos, lineas y cuadrados. El $SN(P)$ de un punto P contiene las lineas incidentes a P pero no los cuadrados. El SN de una linea contiene uno o dos cuadrados, mientras que el SN de un cuadrado es el mismo cuadrado. El subconjunto T es representado por los elementos coloreados, su frontera $Fr(T, S)$ consiste de los circulos y lineas negras. Los puntos blancos no pertenecen a $F = Fr(T, S)$ por que sus SN no intersectan a T. Sin embargos, $Fr(F, S)$ contiene los puntos blancos porque sus SN intersectan ambos F y su complemento. Entonces en este caso la frontera $F = Fr(T, S)$ es diferente de $Fr(F, S)$.

Axioma 2.4 Se tiene que $Fr(Fr(T, S)) = Fr(T, S)$. Es decir, la frontera de $Fr(T, S)$ es la misma que $Fr(T, S)$.

La topología de un espacio S queda definida como la colección de subconjuntos abiertos que satisfacen los siguites axiomas:

Axioma C1 El conjunto total S y el subconjunto ϕ son abiertos.

Axioma C2 La unión de cualesquiera subconjuntos abiertos es abierta.

Axioma C3 La intersección de un número finito de subconjunto abiertos es abierta.

También se incluye el axioma de separación de la topología clásica:

Axioma C4 El espacio tiene la propiedad de separación.

Hay, al menos, tres versiones de la propiedad de separación y del Axioma C4:

Axioma T0 Para cualesquiera dos puntos distintos x y y hay un subconjunto abierto que contiene exactamente uno de los puntos.

Axioma T1 Para cualesquiera dos puntos distintos x y y hay un subconjunto abierto que contiene a x pero no a y y otro subconjunto abierto conteniendo a y pero no a x.

Axioma T2 Para cualesquiera dos puntos distintos x y y hay subconjuntos abiertos que no se intersectan conteniendo exactamente uno de los puntos.

La Figura 2.2 ilustra las tres versiones de la propiedad de separación.
Un espacio con la propiedad de separación T2, se le conoce como espacio de Hausdorff.

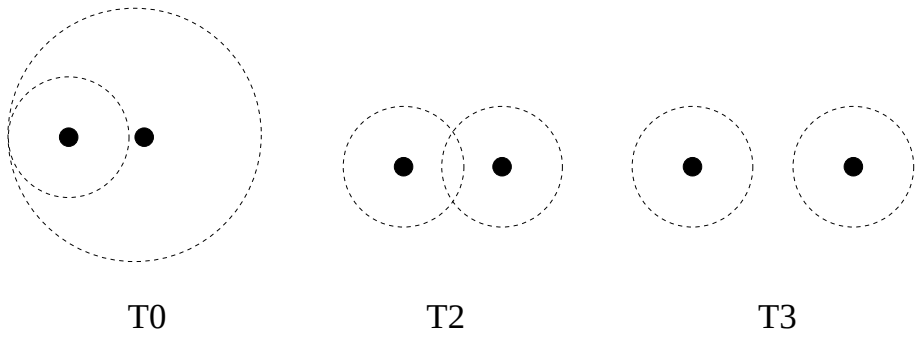

Figura 2.2: Ejemplo de los axiomas de separación

A continuación se presentan algunos resultados que muestran la relacion de los axiomas sugeridos con los axiomas clásicos.

Teorema 2.1 Un espacio localmente finito satisface el Axioma 2.3, si y sólo si la relación de vecindad N del espacio S es antisimétrica.

Definición 2.9 (Abierto)
Un conjunto $O \subset S$ es **abierto** en S si no contiene elementos de su frontera $Fr(O, S)$. Un subconjunto $C \subset S$ es **cerrado** en S si contiene todos los elementos de $Fr(C, S)$.

Lema 2.1 Un subconjunto $T \subset S$ es abierto de acuerdo a la Definición 2.9, si y sólo si contiene para cada $a \in S$ también a su vecindad más pequeña $SN(a)$.

Teorema 2.2 Los subconjuntos de un espacio ALF S satisface los axiomas clásicos $C1, C2$ y $C3$ ya que son abiertos, de acuerdo a la Definición 2.9, y, por tanto, son abiertos en el sentido clásico.

2.2. Propiedades de los espacios ALF

Un espacio ALF es un espacio topológico en el sentido clásico y la relación de vecindad es antisimétrica, de acuerdo con el Teorema 2.1 . Consideremos la relación binaria $a \neq b$ y $a \in SN(b)$. Se le llama usualmente relación de ligado la denotamos con B y decimos que b liga a o a es ligado por b. Esta notación refleja el hecho que $b \in Fr(a, S)$.

Definición 2.10 (Relación de ligado)
La relación binaria entre dos elementos a y b, con $a \neq b$ y $a \in SN(b)$, de un espacio ALF se llama **relación de ligado**.[1]

La relación de ligado es irreflexiva. De acuerdo al Teorema 2.1, debe ser asimétrica.

Lema 2.2 (Mínimo y máximo)
Si S es un espacio ALF, la relación de ligado B es transitiva, entonces S contiene elementos los cuales no están ligados por otros elementos y contiene elementos, que no ligan a ningún elemento. Los primeros se llaman elementos **mínimos** y los segundos **máximos**.

Lema 2.3 Sea T un subconjunto de S. Si la relación de ligado B es transitiva, entonces $Fr(T, S)$ no contiene elementos máximos de S y para cualquier elemento a de S el subconjunto $SN(a)$ contiene al menos un elemento máximo de S.

Teorema 2.3 Un espacio localmente finito satisface el Axioma 2.4, si y sólo si la relación de ligado es transitiva.

El teorema anterior indica que la relación de ligado es transitiva si y sólo si se tiene que $Fr(Fr(T, S)) = Fr(T, S)$ para todo subconjunto $T \subset S$.

Corolario 2.1 (Orden parcial)
La relación B, que es irreflexiva, asimétrica y transitiva, es un **orden parcial** irreflexible y se denota como $a < b$ en lugar de aBb.

Se tienen las siguientes propiedades sobre la vecindad más pequeña.

Corolario 2.2
a) La vecindad más pequeña de cualquier elemento a de un espacio ALF es abierta de acuerdo a la Definición 2.9 y en el sentido clásico, resulta ser el subconjunto abierto más pequeño que contiene al elemento a.
b) La vecindad más pequeña $SN(a)$ de una celda a contiene todos los elementos ligados por a.
c) La vecindad más pequeña en un espacio ALF satisface el Axioma T1 clásico pero no el Axioma T2.

En la siguiente sección consideraremos el caso particular de los espacios localmente finitos conocidos como complejos de celdas abstractas. Comenzaremos por definir los conceptos de estos nuevos espacios que nos ayudarán a desarrollar la nueva teoría de imágenes.

[1] Usaremos relación de ligado como traducción de *bounding relation.*

2.3. Topológia de los Complejos

Definición 2.11 (Menor Vecindad Abierta)
El subconjunto abierto más pequeño de un espacio ALF S que contiene al elemento $a \in S$ es llamado **vecindad abierta más pequeña** de a en S y es denotada por $SON(a, S)$.

De acuerdo con el Corolario 2.2, $SON(a, S) = SN(a)$.

Definición 2.12 (Faceta)
Un elemento del espacio a es llamado una **faceta** del elemento b si $b \in SON(a, S)$. Además, si $a = b$, entonces a es una faceta no propia de b. La **relación faceta** es reflexiva, antisimétrica y transitiva. Entonces, es un orden parcial reflexivo en S y se denota por $a \leq b$.

De acuerdo al Corolario 2.2, la relación de vecindad N en un espacio ALF es la inversa a la relación faceta aNb significa que $a \in SN(b)$ mientras b es una faceta de a. Requerimos de la relación reflexiva faceta junto con la relación irreflexiva de ligado porque hace que la definición de productos cartesianos del espacio ALF sea más simple, mientras que trabajar con complejos resulta más fácil con la relación de ligado.

Un caso particular de los espacio ALF, el cual es muy apropiado para aplicaciones en el cómputo de imágenes, e caracteriza por una relación orden parcial entre los elementos del espacio y por una característica adicional: La **función de dimensión** $dim(a)$, la cual asigna un entero no negativo a cada elemento de espacio, así que si $b \in SON(a, S)$ entonces $dim(a) \leq dim(b)$. Este tipo de espacio localmente finito es llamado **complejo de celda abstracta** o complejo AC. Sus elementos son llamados **celdas**. Si $dim(a) = k$, entonces a es llamada una **celda k–dimensional** o **k–celda**. La dimensión de un complejo está definida por la dimensión más grande de sus celdas.

Definición 2.13 (AC)
Un **complejo de celda abstracta** $C = (E, B, dim)$ es un conjunto de E de elementos abstractos (celdas) con una relación binaria asimétrica, irreflexiva y transitiva $B \subset E \times E$ llamada relación de ligado y con una función de dimensión $dim : E \rightarrow I$, la función va del conjunto E al conjunto I de enteros no negativos tal que $dim(e') < dim(e'')$ para todos los pares $(e', e'') \in B$. Las celdas no son subconjunto de ningún otro conjunto.

Gráficamente, las $0-$celdas son denotadas por discos pequeños o cuadros, las $1-$celdas son denotadas por segmentos de línea, las $2-$celdas por el interior de los polígonos y las $3-$celdas por el interior de poliedros. En la Figura 2.3(a) el complejo consiste de solo $0-$celdas y $1-$celdas. En la Figura 2.3(b) el complejo consiste de $0-$celdas, $1-$celdas y $2-$celdas. Finalmente, en la Figura 2.3(b) el complejo consiste de $0-$celdas, $1-$celdas, $2-$celdas y $3-$celdas.

La dimensión de las celdas representa el orden parcial correspondiente a la relación de ligado. Llamemos a la secuencia $a < b < ... < k$ de celdas de un complejo C, en el cual cada celda liga a la siguiente, una **trayectoria de ligado** de a a k en C. La longitud de

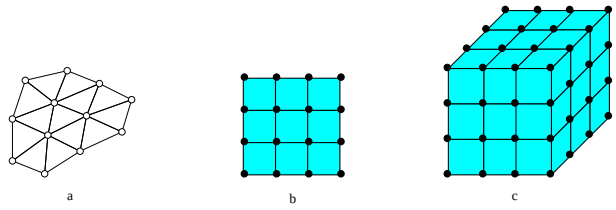

Figura 2.3: Ejemplos de complejos:
(a) uno–dimensional; (b) dos–dimensional; (c) tres–dimensional.

la trayectoria de ligado es el número de celdas en la secuencia menos 1 y se le llama la **longitud de la trayectoria de ligado**.

Definición 2.14 (Dimensión de una celda)
La **dimensión de una celda** c de un complejo C, $dim(c, C)$, es la longitud de la trayectoria de ligado más larga de cualquier elemento del complejo C al elemento c.

De acuerdo con la definición anterior, la dimensión de una celda está dada de manera relativa a un subcomplejo que contiene la celda c porque la longitud de la trayectoria de ligado más larga puede diferir en diversos o varios subcomplejos. La dimensión $dim(c, S)$ de una celda c relativa a un subcomplejo $S \subset C$, $c \in S$, es la longitud de la trayectoria de ligado más larga de un mínimo elemento de S al elemento c en de S. La dimensión de un subcomplejo $S \subset C$ relativa a C es la dimensión más grande de las celdas de S relativa al complejo C, mientras que la dimensión de S es la longitud de la trayectoria de ligado más larga en el subcomplejo S.

Nótese que la relación faceta de cualquier complejo AC es un orden parcial. La dimensión de los elementos del espacio es un propiedad importante. Usando dimensiones prevenimos algunos errores los cuales pueden ocurrir cuando se usa un espacio localmente finito sin dimensión.

Definición 2.15 ($K-$complejo)
Un **K–complejo** es una pareja (E, B) donde E es un conjunto abstracto de elementos llamados celdas y B es una relación asimétrica, irreflexiva transitiva y binaria en E llamada relación de ligado. Un entero no negativo es asignado a cada celda de un $k-$complejo. Este número es llamado dimensión de la celda y es definido por la longitud de las trayectorias de ligado de acuerdo a la Definición 2.14. Las celdas no son subconjuntos de otra celda.

La Figura 2.4 muestra un complejo y su relaciones de ligado de sus celdas. La celda p es un elemento del espacio con dimensión igual a 0.Una de las trayectorias de ligado mas largas de p a v es (p,e,f,v). Su longitud es 3, por lo tanto $dim(v) = 3$.

Un $k-$complejo es un caso particular de un complejo AC. Ya que un $k-$complejo es definido de manera única por el conjunto E y la relación binaria B simétrica, irreflexiva y transitiva, se puede representar por una gráfica dirigida sin lazos ni ciclos.Un $k-$complejo es un espacio ALF. Los $k-$complejos difieren de los complejos abstractos clásicos por la

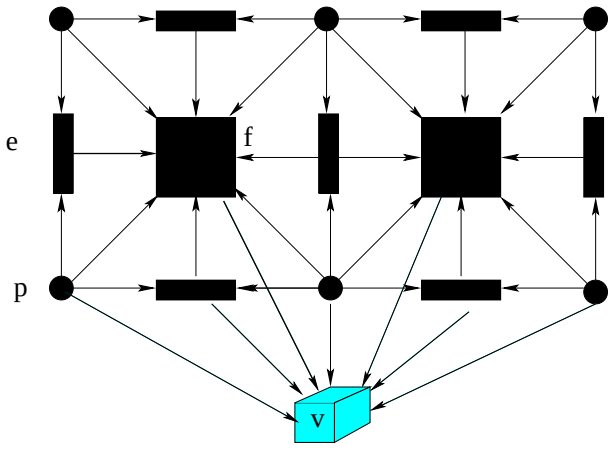

Figura 2.4: Ejemplo de un complejo con la relación de ligado representada por flechas
Una flecha apunta de a a b si a liga a b.

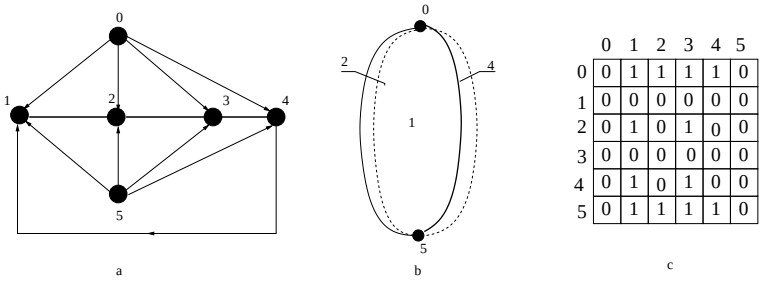

Figura 2.5: Representacion de un complejo.

definición de la dimensión de las celdas. Una celda sólo puede ligar a otra celda de mayor dimensión.

La Figura 2.5 (a) muestra ejemplos de cómo se pueden representar los complejos por medio de digráficas; (b) por medio de un dibujo espacial y (c) por medio de una matriz de ligado. Los nodos de la digráfica representan las celdas, mientras que las aristas dirigidas representan la relación de ligado. La digráfica puede representarse por una matriz de adyacencia la cual, en el caso de los complejos se llama matriz de ligado. Cada fila y y cada columna x de la matriz corresponde a una celda del complejo. Un elemento de la matriz (x, y) igual a 1 indica que la celda y liga a la celda x. La matriz es asimétrica.

Definición 2.16 (Subcomplejo)
Un **subcomplejo** $S = (E', B')$ de un $k-$complejo $C = (E, B)$ es un subcomplejo cuyo conjunto E' es un subconjunto de E y la relación B' es una intersección de B con $E' \times E'$. La dimensión de las celdas de S son definidas de acuerdo a la Definición 2.14, y pueden ser diferentes de las dimensiones de las mismas celdas en C.

Como un subcomplejo de C es definido de manera única por el complejo C y el subconjunto E', es posible aplicar las operaciones de conjuntos como unión, intersección y complemento a los subcomplejos.

Como en cualquier espacio topológico, algunos subconjuntos de un complejo son abiertos. Por ejemplo, considere la Figura 2.6, los subconjuntos $\{f_1, l, f_2\}$ y $\{f_1, l, f_2, l_1, f_3, l_2, v_1\}$ son abiertos en el complejo C. El subconjunto $T = \{f_1, l\}$ no es abierto en C pues la celda l liga a f_2 la cual no está contenida en T.

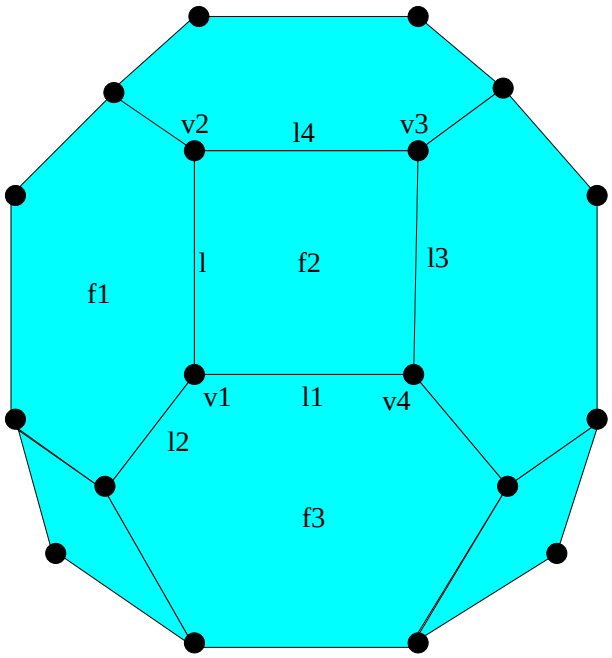

Figura 2.6: La superficie de un poliedro considerada como un complejo.

Una $n-$celda c^n de un complejo $n-$dimensional C^n es un subconjunto abierto de C^n ya que c^n no liga celdas de C^n. Una $0-$celda c^0 de algún complejo C es cerrado en algún subconjunto C ya que no existen celdas que liguen a c^0.

El complemento $T = C \setminus A$ de un subcomplejo $A \subset C$ abierto en C es cerrado en C y viceversa. La propiedad de un subcomplejo S de ser abierto o cerrado es relativa, pues depende del complejo.

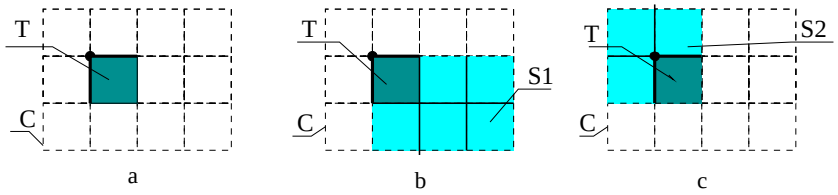

Figura 2.7: Ejemplos de subcomplejos, los cuales son abiertos o cerrados dependiendo del complejo.

En la Figura 2.7 (a) el complejo $2-$dimensional C está representado por una malla de líneas punteadas. El subcomplejo $T \subset C$, que está formado por el cuadro, las dos líneas y el vértice negro, no es ni abierto ni cerrado en C, esto se debe a que las celdas de T

ligan a algunas celdas que no pertenecen a T y están ligadas por algunas otras celdas en $C \setminus T$. En la Figura 2.7 (b) tenemos que $T \subset S_1$, es abierto en S_1, ya que contiene todas las celdas de S_1 ligadas por las celdas de T, pero no aquellas en C. En la Figura 2.7 (c) tenemos que $T \subset S_2$, es cerrado en S_2, ya que contiene todas las celdas de S_2 que liga alguna celda de T.

Teorema 2.4 (Ligado Abierto)
Un subcomplejo $S \subset C$ es abierto en C, de acuerdo a la Definición 2.9, si y sólo si contiene a cada celda de C ligada por alguna celda de S.

Teorema 2.5 (Ligado Cerrado)
Un subcomplejo $S \subset C$ es cerrado en C, de acuerdo a la Definición 2.9, si y sólo si contiene a cada celda de C que liga a alguna celda de S.

Corolario 2.3 Si S es un subconjunto abierto de C y la celda pertenece a S, entonces $SON(c, C) \subset S$.

Corolario 2.4 Si una celda c_2 pertenece a la vecindad de otra celda c_1, $c_2 \in SON(c_1, S)$, entonces existe una vecindad c_2 que pertenece a la vecindad de c_1. En cualquier caso:

$$c_2 \in SON(c_1, S) \rightarrow SON(c_2, S) \subset SON(c_1, S)$$

Definición 2.17 (Cerradura)
Sean t y T subconjuntos del espacio S tal que $t \subseteq T \subseteq S$. El conjunto que contiene a cada celda de T que ligan a a es llamado la **cerradura** de t en T y se denota como $Cl(t, T)$. El conjunto $t \setminus Fr(t, T)$ es llamado el **interior de t en T** y es denotado por $Int(t, T)$. Al conjunto $T \setminus t \setminus Fr(t, T)$ se le conoce como el **exterior** de t en T y es denotado por $Ext(t, T)$.

La cerradura no sólo se usa para los subconjunto sino también para las celdas individuales.

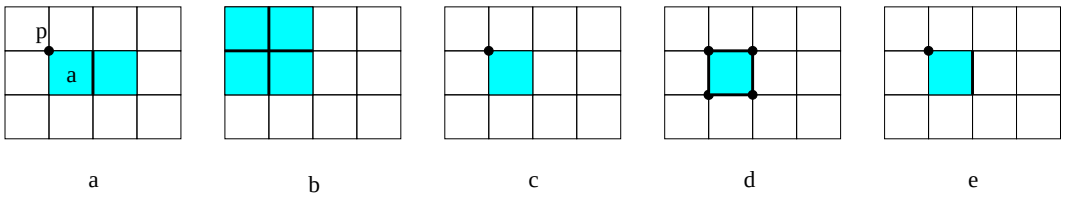

Figura 2.8: Ejemplos de SON y cerraduras dependiendo del complejo.

En la Figura 2.8 (a) tenemos el subcomplejo $S \subset C$ representado por dos cuadros sombreados, una línea gruesa y un vértice. En la Figura 2.8 (b) se muestra el $SON(p, C)$ de la $0-$celda p en C, mientras que en la Figura 2.8 (c) se muestra el $SON(p, S)$ de p en S. En la Figura 2.8 (d) se muestra la cerradura $Cl(a, C)$ de la celda a en C, mientras que en la Figura 2.8 (e) se muestra $Cl(a, C)$ de a en S.

Definición 2.18 (Complejos Incidentes)
Dos subcomplejos S_1 y S_2 de un complejo C son **incidentes** el uno del otro si dos celdas incidentes $x \in S_1$ y $y \in S_2$ existen en C.

Definición 2.19 (Dual)
Un complejo D es llamado $\boldsymbol{n}$**–dual** de C cuando se obtienen de un complejo $n-$dimensional C cambiando el orden de dos celdas en cada par de la relación de ligado B y reemplazando la dimensión de cada celda por $n - d$.

Si C^n es un complejo $n-$dimensional y D es $n-$dual de C^n, entonces D es también $n-$dimensional. Cada subconjunto abierto de C^n corresponde a un subconjunto cerrado del dual D y viceversa.

De acuerdo al **principio de Dualidad**, subconjuntos abiertos y cerrados de un espacio topológico tienen propiedades simétricas. Cuando reemplazamos la noción de abierto por cerrado, cerrado por abierto e intercambiamos intersección por unión, entonces los axiomas del espacio topológico son válidos.

Una de las nociones topológicas más importantes para las aplicaciones, es la de **conexividad**. La conexividad en los complejos es definida como la cerradura transitiva de la relación de incidencia o por una trayectoria de incidentes.

Definición 2.20 (Conexo)
Un espacio topológico es **conexo** si no puede ser representado como la unión de dos subconjuntos no vacíos, abiertos y disjuntos.

Teorema 2.6 La Definición 2.20 y la Definición 2.4 aplicadas a los complejos AC son equivalentes.

Se debe notar que cualquier subconjunto S de un espacio C puede ser considerado a su vez un espacio cuyos subconjuntos abiertos son la intersección de los subconjuntos abiertos del espacio C con el subconjunto S. Entonces la Definición 2.20 y el Teorema 2.6 se pueden aplicar a los subconjuntos de un espacio y, en particular, a subcomplejos. En ese caso se requiere de una Definición 2.20 más precisa: Un subconjunto S de un espacio topológico es **conexo** si no puede ser representado como una unión de dos subconjuntos no vacíos disconjuntos que son abiertos en S^n.

La **conexidad** de las celdas es una relación de equivalencia es reflexiva, simétrica y transitiva. Sus clases de equivalencia son llamadas componentes. Conforme al Teorema 2.4, una componente de un conjunto S abierto es abierta en S y ,por otro lado, de acuerdo al Teorema 2.5 el complemento de un subconjunto el cual es abierto en S es cerrado en S. Por lo tanto, una componente de un conjunto S es abierta y cerrada en S. Entonces la Definición 2.20 se puede reformular como: Un espacio topológico es conexo si no puede ser representado como la unión de dos subconjuntos no vacíos, disjuntos y cerrados.

Definición 2.21 ($n-$dimensional homogéneo) Un complejo $n-$dimensional C^n es llamado $n-$**dimensional homogéneo** cuando cada una de sus celdas de dimensión menor que liga al menos una $n-$celda del complejo C^n.

Definición 2.22 (Trayectoria $n-$dimensional)
Una trayectoria de incidencia en un complejo C^n es llamada una **trayectoria** $n-$**dimensional** en C^n cuando es de la forma

$$x_0^n x_1^{n-1} x_2^n ... x_{l-1}^{n-1} x_l^n$$

donde x_i es una celda $n-$dimensional y x_i^{n-1} es una celda $(n-1)-$dimensional de C^n.

Definición 2.23 (Fuertemente conexo)
Un complejo $n-$dimensional C homogéneo es llamado **fuertemente conexo** si para cualesquiera dos celdas $n-$dimensionales de C existe en el complejo C, una trayectoria $n-$dimensional que contiene esas dos celdas.

Definición 2.24 (Frontera)
La **frontera** $Fr(S, C)$ de un subcomplejo $S \subset C$ en C es el subcomplejo de C que contiene todas las celdas c del complejo C tal que sus $SON(c, C)$ en C contiene tanto celdas de S como celdas del complemento $C \setminus S$.

Corolario 2.5 La frontera $Fr(S, C^n)$ de un subcomplejo de un complejo C^n $n-$dimensional no contiene celda $n-$dimensionales.

Definición 2.25 (Borde combinatorio)
Considere un complejo $n-$dimensional C^n y su subcomplejo S^k $k-$dimensional con $S^k \subset C^n$. La cerradura en C^n del conjunto de todas las $(k-1)-$celdas de S^k, cada una de las cuales ligan exactamente una $k-$celda en S^k es llamado **borde combinatorio** de S^k. Es usual denotar al borde de S por δS.

Definición 2.26 (Frontera abierta)
Considera un subcomplejo S de un complejo C. La **frontera abierta** $Of(S, C)$ de S en el complejo C es el subcomplejo de C que contiene cada celda $c \in C$ cuya cerradura $Cl(c, C)$ contiene ambas celdas tanto de S como las celdas del complemento $C \setminus S$.

Se emplea la noción de frontera y frontera abierta no sólo para los subconjuntos sino también para las celdas individuales.

2.4. Complejos Cartesianos y Coordenadas Combinatorias

Considera un espacio S cuyos elementos componen una secuencia

$$A = (e_0, e_1, e_2, ..., e_{2m}); m \geq 1.$$

Cada e_i con un índice par i liga los elementos e_{i-1} y e_{i+1} que tienen índices impares. Entonces A se convierte en un espacio ALF $1-$dimensional conexo. Asignamos dimensión a los elementos de A de acuerdo con la Definición 2.14. Entonces, S se convierte en un complejo AC $1-$dimensional. Cada elemento con un índice par es cerrado y cada índice impar es abierto.

Cada celda cerrada es una faceta propia de una o dos celdas abiertas. Los índices de A son llamados **coordenadas combinatorias** de las celdas en un espacio $1-$dimensional.

Definición 2.27 (Complejo Cartesiano)
Un complejo AC de dimensión muy grande que se define como el producto cartesiano de complejos $1-$dimensional se llamado **complejo cartesiano** AC.

El conjunto de celdas de un complejo cartesiano $n-$dimensional AC C^n es el producto cartesiano de n conjuntos de celdas de complejos AC $1-$dimensionales.
Estos complejos son los ejes coordenados de C^n el cual es un espacio $n-$dimensional. Se denotan con A_i con $i = 1, 2, ..., n$. Una celda c del complejo cartesiano AC $n-$dimensional C^n es una $n-$tupla $c = (a_1, a_2, ..., a_n)$ de celdas a_i de los ejes correspondientes: $a_i \in A_i$. Las celdas a_i son llamadas **componentes** de c.

Definición 2.28 (Relación facial)
La **relación facial** del complejo cartesiano $n-$dimensional C^n se define de la siguiente manera: la $n-$tupla $(a_1, a_2, ..., a_n)$ es una faceta de otra $n-$tupla $(b_1, b_2, ..., b_n)$ si y sólo si, la celda $a_i \in A_i$ es una faceta de la celda $b_i \in A_i$, para toda $i = 1, 2, ..., n$.

De acuerdo a las Definiciónes 2.10 y 2.12, la n-tupla $(a_1, a_2, ..., a_n)$ liga a otra n-tupla $(b_1, b_2, ..., b_n)$ si y sólo si es diferente de $(b_1, b_2, ..., b_n)$ y es su faceta.

En la Figura 2.9 se puede ver que las coordenadas de las $0-$celdas P_1, P_2 y P_3 son ambas pares, las coordenadas de las $1-$celdas C_1, C_2 y C_3 al menos una es impar y ;finalmente, las coordenadas de las $2-$celdas F_1, F_2 y F_3 son ambas impares. Dado lo anterior se puede inferir fácilmente la dimensión de una celda por medio de sus coordenadas, lo que da paso al siguiente teorema:

Teorema 2.7 (Celda $k-$dimensional)
La dimensión de una celda $c = (a_1, a_2, ..., a_n)$ en un complejo cartesiano C es igual al número de sus componentes a_i, $i = 1, 2, ..., n$, las cuales son abiertos en sus ejes.

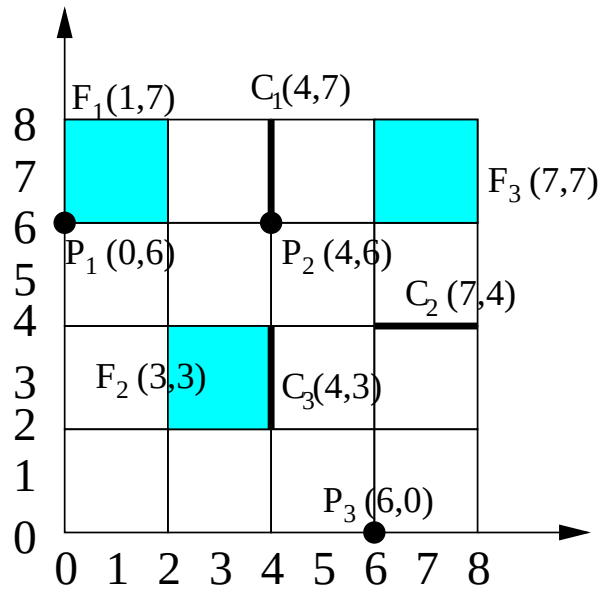

Figura 2.9: Ejemplo de un complejo cartesiano 2−dimensional.

De acuerdo con el convenio de que una componente a_i de la celda $c = (a_1, a_2, ..., a_n)$ es abierta si y sólo si a_i es un número impar, podemos decir que la dimensión de una celda c en el espacio C^n es igual al número de coordenadas impares de la celda c. Las celdas de dimensión más grande n en C^n son llamadas **celdas principales** de C^n.

Definición 2.29 (Celda Interna)
Una celda c de un complejo cartesiano finito $n-$dimensional C^n se conoce como una **celda interna** de C^n si y sólo si la $i-$ésima coordenada de c satisface la relación $\min_i<x_i<\max_i$, donde $\min_i$ es el mínimo y $\max_i$ es el máximo valor de la $i-$ésima coordenada de c con respecto a todas las celdas del complejo C^n.

Según la definición anterior, las celdas internas de la Figura 2.9 son P_2, F_2 y C_3.

Lema 2.4 (Lema NIC1)
Una celda $k-$dimensional c^k de un complejo cartesiano $n-$dimensional C^n liga la celda c^m $m-$dimensional si y sólo si todas las coordenadas impares de c^k son iguales a la correspondiente coordenada de c^m y exactamente $m-k$ de las coordenadas pares de c^k difieren de la correspondiente coordenada de c^m por uno.

Teorema 2.8 (Teorema NIC)
Una celda interna $k-$dimensional c^k de un complejo cartesiano $n-$dimensional C^n liga exactamente $C_{n-k}^{m-k} \times 2^{m-k}$ celdas $m-$dimensional de C^m y está ligado por exactamente $C_k^{k-j} \times 2^{k-j}$ celdas $j-$dimensionales de C^n.

Es apropiado en ocasiones emplear las coordenadas semi-combinatorias conocidas: Una componente con una coordenada x semi-entera es una faceta de dos componentes con

coordenadas enteras $x^{+}_{-}\frac{1}{2}$. Una componente con una coordenada entera es una faceta (no propia) por sí misma. Esta versión es útil en los casos cuando una celda compleja se compara con una imagen digitalizada.

2.5. Complejos AC comparados con otros Espacios Localmente Finitos

Se puede observar que para cualquier complejo Euclidiano hay un complejo AC isomorfo, pero el inverso no es verdad. Una celda $k-$dimensional c^k del complejo AC C la cual liga la celda $m-$dimensional c^m de C corresponde a una faceta $k-$dimensional f^k de una celda $m-$dimensional e^m de un complejo Euclidiano. Similarmente, para cualquier complejo simple existe un complejo isomorfo AC cuyas $2-$celdas están ligadas por exactamente tres $1-$celdas y cuyas $3-$celdas están ligadas por exactamente cuatro $2-$celdas.

Los espacios localmente finitos llamados espacios **Khalimsky**, son similares a los complejos cartesianos AC. Un espacio Khalimsky $1-$dimensional es una secuencia alternante de *puntos delgados* y *puntos gordos* mientras la vecindad más pequeña de un punto delgado es el punto en sí y que un punto gordo contiene además del punto en sí, otros dos puntos delgados.

Tal como un espacio es equivalente a un complejo $1-$dimensional: los puntos delgados corresponden a las $1-$celdas y los puntos gordos a las $0-$celdas. El orden parcial corresponde a la relación faceta. Los espacios Khalimsky de dimensiones más altas son definidos como productos cartesianos de espacios $1-$dimensionales. Para un espacio Khalimsky $n-$dimensional obviamente existe un complejo AC cartesiano equivalente. La diferencia importante entre estos dos espacios es que no hay una dimensión asignada a los elementos de un espacio Khalimsky. Sin embargo, la noción de dimensión de las celdas es muy importante, ignorar la dimensión de las celdas puede llevar a equivocaciones y errores. Todos los posibles espacios localmente finitos tienen mucho en común, como se indica a continuación:

Teorema 2.9 (Teorema IAC) Para cualquier espacio topológico localmente finito que satisface los axiomas propuestos existe un complejo AC isomorfo.

Capítulo 3

Entorno del problema

Para el análisis de imagen se recomiendan diferentes fórmulas que combinan números de configuraciones del espacio local a lo largo de la curva o a lo largo de la frontera de una región digital para estimar la longitud de una curva digital o del perímetro de un objeto digital plano. En general, se sabe que estos métodos son imprecisos y, lo más importante, es que la precisión no puede ser mejorada incrementando la resolución de la digitalización.

Desde el punto de vista de la convergencia, la estimación de un perímetro y la del área de una región plana se comportan de manera muy diferente: El área puede ser estimada contando el número de elementos en el espacio, tales como puntos en la malla o cuadrados que contiene el objeto. Esta estimación converge al área real para regiones en el plano Euclidiano. En cambio, estimar la longitud de una curva de Jordan no puede estar basada en el conteo de elementos del espacio, tales como aristas o diagonales de la malla, ya que al escalarse podría modificarse la resolución de la malla y la aproximación del perimetro podría ser muy diferente para diversas orientaciones, de curvas idénticas.

Durante el desarrollo de esta tesis analizaremos y compararemos dos técnicas: el método DSS el cual se basa en la partición de una curva digital en segmentos rectos, y el método MLP, que se caracteriza por el cálculo del polígono de longitud mínima en una frontera abierta de una región digital. Ambos métodos nos permiten calcular la longitud de una curva o el perímetro, respectivamente, de curvas digitales o de conjuntos digitales simplemente conexos.

Para ambas técnicas se sabe que la medida de la longitud de la curva converge al valor real si una región convexa, en el plano Euclidiano, es digitalizada con resolución incremental de la malla. Estos teoremas fundamentan la teoría de este trabajo.

Los algoritmos DDS y MLP presentados son de tiempo lineal. Ambos se describen detalladamente y se ejemplifican. Además, también se comparan con respecto a la velocidad de convergecia y el número de segmentos generados.

3.1. Fundamentos del Trabajo

Las regiones digitales que usaremos estarán descritas por medio de **complejos de celdas abstractas** AC [1], sea $C = [E, B, dim]$ un AC, entonces C tiene un conjunto E de elementos o celdas, además, posee una relación binaria antisimétrica, irreflexiva y transitiva B que es llamada relación de ligado. Si $B(e', e'')$ entonces decimos que e' liga a e'', para toda $e', e'' \in E$. La dim es una función entera que específica la dimensión de un elemento $e \in E$. Una $k-$celda es una celda de dimensión k. Una celda sólo puede ligar a otra celda de una dimensión mayor. Si una celda liga a otra, entonces se dice que son **incidentes** la una a la otra. Una celda es incidente a sí misma. Con esto podemos ver que la relación de incidencia es simétrica, reflexiva y no es transitiva. La envolvente transitiva de la relación de incidencia es la **relación de conexividad**, definiendo así la noción topológica de subconjuntos conexos en los complejos de celdas abstractas AC.

3.2. Línea frontera y Frontera abierta de una región digital.

Los algoritmos a analizar en esta sección pretenden calcular la longitud de una curva abierta o cerrada en el análisis de imágenes. Comenzamos por tomar a las curvas cerradas como las fronteras de las regiones y las curvas abiertas como subconjuntos de las fronteras. Una región digital es el resultado de la digitalización de una región análoga. Una curva digital es la línea frontera (o un subconjunto conexo de una línea frontera) de una región digital. Para tener una expecificación exacta de las nociones de las regiones digitales y sus fronteras, haremos referencia a diversos conceptos básicos de la topología de los complejos de celdas abstractas descritos en el Capítulo 2.

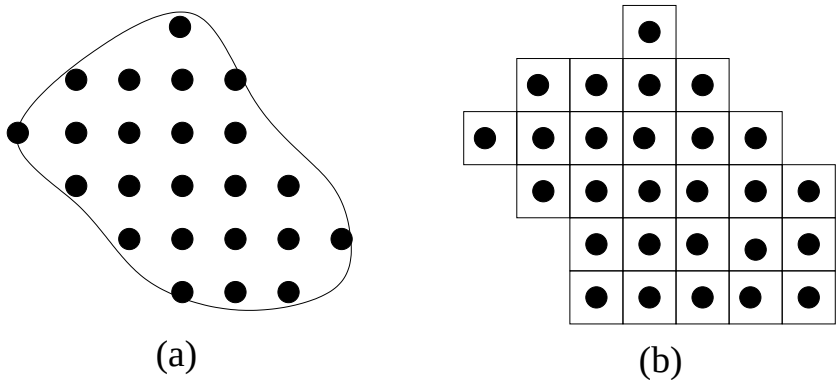

Figura 3.1: Ejemplo de una región y su analogo digital.

En la Figura 3.1(a) se muestra una región sin digitalizar y mientras que en la (b) se muestra el analogo digital de la misma región al utilizar alguna estructura de datos para

[1] Presentado en el Capítulo 3, Definición 2.13.

representarla en la computadora, las cuales se analizaran posteriormente.
Consideraremos a una imagen digital 2−dimensional como un espacio cartesiano AC 2−dimensional. Un espacio cartesiano AC 2−dimensional es un producto cartesiano (producto cruz) de dos AC 1−dimensionales que pueden considerarse ejes coordenados. Contiene celdas de dimensión 0, 1 y 2. Llamamos a las 2−celdas *pixeles*; las 1−celdas, *cracks*[2], y las 0−celdas, *puntos*.

Gráficamente representamos a los pixeles como cuadrados, a los cracks como los lados de esos cuadrados y los puntos representan vértices de los cuadros permitiéndonos visualizar la relación de ligado de un AC: un crack liga a un pixel si el crack es representado como un lado del cuadro que representa al pixel, así un punto liga a un crack si el punto representa uno de los puntos finales del segmento de línea que representa al crack y un punto liga a un pixel si el punto representa un vértice de cuadro que representa al pixel. En un complejo cartesiano las celdas se organizan en filas y columnas para formar una malla regular. Esta red se pueden dibujar ortogonalmente. Hay que aclarar que esta manera de representar los cartesianos AC no implica que las celdas de un AC son subconjuntos de un plano Euclidiano. Las celdas de un AC no están compuestas por objetos más pequeños, son unidades invisibles de diferentes categorías definidas por su dimensión.

La idea principal que llevó a usar los AC, es tener una estructura que contenga un número finito de elementos que se puedan almacenar y procesar por la computadora y que poseen una topología en el sentido clásico.

Un subcomplejo S es **homogéneamente 2−dimensional** si alguna celda en S es una 2−celda o bien liga a una 2−celda de S. Una **región digital** es un subcomplejo conexo homogéneamente 2−dimensional cuyo complemento es también conexo y homogéneamente 2−dimensional.

Es fácil ver que la vecindad más pequeña de un pixel, consiste del propio pixel. La vecindad más pequeña de un crack contiene al crack y a los dos pixeles que están ligados por él. La vecindad más pequeña de un punto contiene al mismo punto y los cuatro pixeles ligados por él. Por lo tanto, la frontera de una región digital consiste de cracks y puntos y no contiene pixeles.

El problema que consideramos consiste en estimar el perímetro de una preimagen de una región digital S mientras que analizamos a S. El estimado se espera que sea lo *más cercano* al perímetro del la preimagen de S, el cual se asume que es una región en el plano Euclidiano, teniendo una curva de Jordan[3] como su frontera. El objetivo, en palabras más precisas, consiste en asegurar la convergencia hacia el perímetro verdadero.

La **línea delimitante o línea frontera** de una región digital S está definida como un secuencia alternante de cracks y puntos, los cuales cubren todas las celdas en δS y donde cada celda es seguida por una celda incidente a ella. La línea delimitante es una curva poligonal cíclica cerrada y se puede dividir en segmentos digitales de líneas rectas (DSS). El método DSS, por sus siglas en inglés, es definido mediante la partición de una frontera en segmentos digitales de líneas rectas cada uno de longitud máxima.

[2]Hay que aclarar que *crack* y arista son dos conceptos distintos, pues la aristas representa lados de un polígono y los cracks solo lados de los pixeles.

[3]Una curva de Jordan corresponde a cualquier curva en topología.

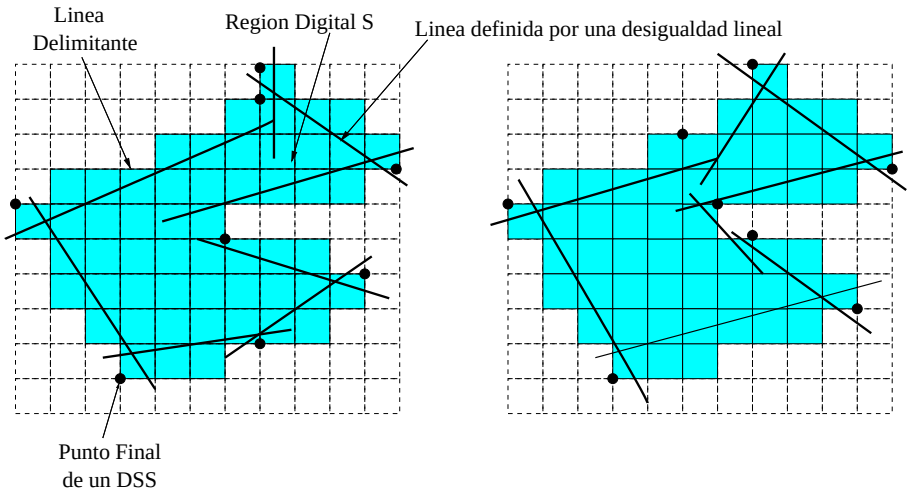

Figura 3.2: Ejemplos de partición de una línea delimitante en DSS de longitud máxima.

En la Figura 3.2 podemos ver la partición de la línea delimitante de la región digital S. Observamos que en ambos casos la partición es distinta aunque tienen el mismo vértice inicial, recordemos que los resultados de la partición pueden variar dependiendo de la posición inicial y orientación que se elijan. En este caso el primer ejemplo nos muestra la partición en sentido de las manecillas del reloj y el segundo en sentido contrario.

Específicamente, el algoritmo DSS traza la línea delimitante crack por crack y la subdivide en segmentos digitales rectos de longitud máxima. El algoritmo detecta para cada longitud máxima DSS las coordenadas de sus puntos extremos y calcula la longitud de cada DSS como la distancia euclidiana entre estos puntos. La suma de las longitudes de estos segmentos de línea es finalmente usada como el estimado DSS del perímetro de la región digital dada.

El método MLP,por sus inglés, originalmente, se define usando la noción de una malla continua. Sin embargo, trabajar con una **frontera abierta** de una región digital también es adecuado. La **frontera abierta** de una región S es el conjunto de todos los cracks y pixeles cuya cerradura topológica intersecta tanto a la región S como al complemento $C \setminus S$. Esto difiere de la línea delimitante la cual consiste de puntos y cracks.

Dos pixeles e_1, e_2 se dice que están **conectados por aristas** si no son idénticos y si existe (exactamente) un crack incidente a ambos pixeles. Una frontera abierta es llamada **simple** si cada $2-$celda e en esta frontera abierta está conectada por arista a dos $2-$celdas adicionales en tal frontera abierta. Únicamente los cracks *entre* dos $2-$celdas conectadas por aristas de una frontera abierta están contenidas en esta frontera abierta y ningún otro crack.

Una **malla continua del ciclo cerrado simple 1–dimensional** corresponde a la cerradura de una **frontera simple** cerrada. Una frontera abierta de una región digital contiene (al menos) una frontera abierta simple. Si la frontera abierta de una región digital S es homeomorfa al ciclo cerrado[4] entonces se cumple que todas las celdas de la frontera

[4]El autor hace referencia al termino *annulus* que de manera formal se refiere a una figura geométrica

de esta frontera abierta puede ser trazada por dos secuencias alternantes de cracks y puntos; es decir, la línea delimitante L_1 de la región dada y una segunda línea L_2 a la cual denominados **línea delimitación** de la región dada.

El método MLP consiste en encontrar la curva poligonal más corta, llamada MLP por sus siglas en inglés, que cae completamente en la cerradura de la frontera abierta de una región digital S y rodea la línea delimitante de S. La curva MLP de una cerradura de una frontera abierta de una región digital es definida de manera única si esta frontera abierta es homeomorfa al ciclo cerrado. La longitud de esta MLP es finalmente usada como el estimado MLP del perímetro de la región digital.

En la Figura 3.3 hay dos ejemplos que ilustran nuestra definición de MLP. En ambas figuras podemos ver una región digital S, en la figura de la derecha encontramos la frontera abierta de S y su polígono de longitud máxima; en la figura de la izquierda la región digital presenta dos huecos, el primero de ancho 1 y el segundo de ancho 2, esto nos muestra los problemas que podemos encontrar al calcular su polígono de longitud máxima, además vemos la línea delimitante L_1 y la línea delimitación L_2 de la región digital expandida.

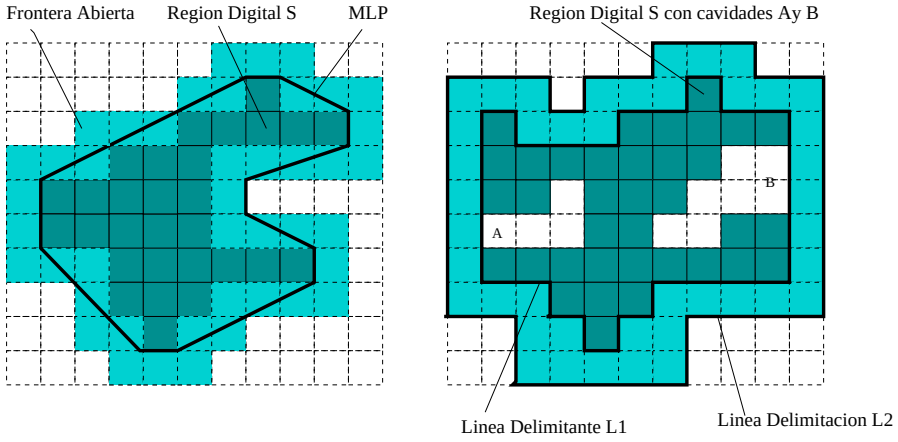

Figura 3.3: Ejemplos de MLP.

3.3. Segmentos Digitales de Líneas Rectas

Los segmentos digitales de líneas rectas se pueden definir a partir de la noción de un semiplano. Por esta razón, tendremos que asignar coordenadas a todas las celdas del complejo C. En general, la teoría de los AC permite la especificación de las coordenadas topológicas de las celdas: Supongamos un complejo $1-$dimensional que consiste de una secuencia alternante de $0-$celdas y $1-$celdas. Estas celdas son enumeradas de manera única por enteros que especifican las coordenadas topológicas para todas las $0-$celdas y $1-$celdas. El producto cruz de dos complejos $1-$dimensionales es isomorfo a un complejo

con forma de anillo, es decir, un ciclo cerrado.

2−dimensional C introduciendo tupla de enteros como las coordenadas topológicas para todas las 0−celdas, 1−celdas y 2−celdas.

Definición 3.1 (Semiplano digital)
Sea C un complejo. Un **semiplano digital** en C es el conjunto de todas las celdas cuyas coordenadas satisfacen una desigualdad lineal dada.

Hay que observar que un semiplano digital satisface la definición de una región digital. Así, la línea delimitante de un semiplano digital queda definida.

Definición 3.2 (Segmentos digitales de línea recta o DSS)
Un **segmentos digitales de línea recta** o **DSS** es un subconjunto conexo de la línea delimitante de un semiplano digital. Se sigue que un DSS no contiene pixeles, solamente contiene cracks y puntos, como se ha indicado anteriormente para las líneas delimitantes de las regiónes digitales en general.

Ésta nos es la única manera de definir los segmentos digitales de línea recta en espacios finitos. En principio hay, al menos, dos tipos de DSS considerados en la literatura de la geometría digital: el primero es el DSS como se definió anteriormente que consiste de cracks y puntos, el cual también se puede llamar DSS $frontera$, y el segundo, llamado **segmentos digitales de línea recta visual o DSS visual** es una secuencia de pixeles cuyas coordenadas topológicas satisfacen dos desigualdades lineales. Este último tipo de DSS es extensamente utilizado, ya que es fácil de visualizar en una computadora mientras que el DSS frontera requiere de esfuerzos adicionales para poder visualizarlo.

El DSS frontera describe las líneas delimitantes de las regiónes y, siendo curvas matemáticas de ancho cero, que se puede usar fácilmente en soluciones de diversos problemas en geometría digital. Los DSS frontera se han elegido para resolver el problema de la subdivisión de una curva digital dada en DSS de longitud máxima, donde una curva digital es una secuencia alternante de puntos y cracks. Ésta puede o no ser un ciclo cerrado.

Consideramos a la curva digital como dirigida. Esto significa que todo crack tiene un punto de inicio y un punto final siguiendo una orientación global única de la curva. La arista más larga en la envolvente convexa del conjunto de puntos de un DSS es llamado su **base**. Cualquier punto de un DSS tiene una distancia restringida desde la base. Si la base se encuentra a la derecha del DSS (de acuerdo a la orientación del DSS) entonces es llamada **base derecha**. En este caso la **base izquierda** queda definida como el lado del envolvente convexa paralelo a la base derecha. La base izquierda puede degenerarse a únicamente un sólo vértice.

Del mismo modo, si el lado más largo de la envolvente convexa se encuentre a la izquierda del DSS entonces se llama la base izquierda, y la base derecha queda entonces definida como el lado del envolvente convexa que es paralelo a la base izquierda. Finalmente, contiene un único vértice.

En la Figura 3.4 se muestra un ejemplo de una curva digital dirigida a la cual calculamos su cierre convexo y así podamos encontrar la base de esta curva, de acuerdo a la definición

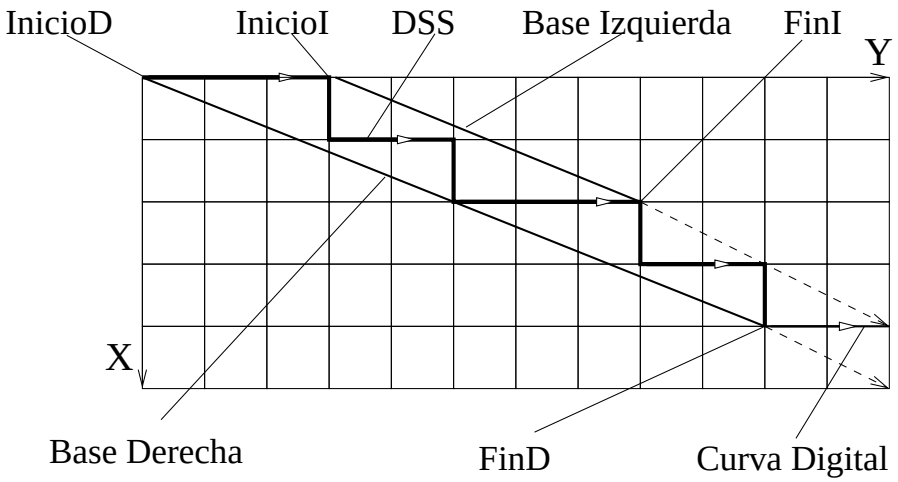

Figura 3.4: Ejemplo del cálculo de las bases.

podemos ver que la base es una base derecha y con esto podemos identificar la base izquierda que por definición es la arista paralela a la base derecha.

Usamos un sistema de coordenadas cartesianas en el plano donde los puntos de la malla (0−celdas) tienen tuplas de números enteros como coordenadas. Todos los puntos de la malla(X, Y) de un DSS satisfacen las siguientes desigualdades:

$$0 \leq V \cdot X - U \cdot Y + W \leq |U| + |V| - 1$$

donde (U, V) es el vector tangencial paralelo a la base derecha del DSS teniendo componentes enteras primas mutuas y W es un entero de valor constante para el DSS y elegido de tal manera que $V \cdot X - U \cdot Y + W$ sea igual a cero para cualquier punto de la malla (X, Y) que se encuentran en la base derecha. Denotaremos el valor $V \cdot X - U \cdot Y + W$ como $H(X, Y)$. Tenemos entonces que $0 \leq H(X, Y) \leq |U| + |V| - 1$. También se satisface que si $H(X, Y)$ es igual a menos uno lo cual significa que el punto (X,Y) no pertenece al DSS con los parámetros dados U, V y W, entonces los parámetros pueden ser actualizados de tal manera que todos los puntos anteriores del DSS, así como los puntos recientes (X, Y), pertenezcan a un DSS caracterizado por los parámetros actualizados. Este método de actualización es implementado en la función $DSS\_Cont()$.

El valor $H(X, Y) = -1$ corresponde a un **delimitante exterior derecho** [5], esto es, a un punto, el cual se encuentra a la derecha de la base derecha. De manera similar, podría haber un **delimitante exterior izquierdo**. En este caso $H(X, Y)$ es igual a $|U|+|V|$. En los casos en que $H(X, Y)<-1$ o que $H(X, Y)<|U| + |V|$ no hay la posibilidad de ajustar el parámetro; esto es: no existe un DSS que contenga a todos los puntos anteriores junto con el punto actual (X, Y).

[5]Utilizaremos el término delimitante exterior como traducción del término *outliner*

3.4. Polígono de longitud mínima

Para una línea L en el plano Euclidiano sea P_L la zona cerrada poligonal delimitada por la línea L. Para la línea delimitante L_1 y la línea de delimitación L_2 se tiene que $L_1 \subseteq P^{\circ}_{L_2}$. Suponemos que una curva poligonal es trazada en sentido contrario a las manecillas del reloj. Un vértice de estas curva poligonal es un **vértice convexo** si la curva gira a la izquierda en este vértice, un **vértice cóncavo** en el caso de que gire a la derecha, y un **vértice colineal** en comparación con ambos vértices vecinos en otro caso. Para tal decisión es común usar el valor del determinante:

$$det(p_1, p_2, p_3) = (X_2 - X_1)(Y_3 - Y_1)(X_3 - X_1)$$

para la secuencia de tres puntos $p_i = (X_i, Y_i)$, con $i = 1, 2, 3$; la cual es negativa para los giros a la izquierda, positiva para los giros a la derecha, e igual a cero para las situaciones colineales suponiendo un sistema coordenado XY. Los vértices del MLP en $G = P_{L_2} \setminus P^{\circ}_{L_1}$ pertenece a cualquiera de los vértices convexos de L_1 o a los vértices cóncavos de L_2.

Supongamos que tenemos dos curvas de Jordan L_1, L_2. Decimos que un rayo $\overrightarrow{pq}$ **toca primero** a L_1 si y sólo si el rayo comienza en p, pasa por q, intersecta a L_1 en un punto r, y no intersecta a L_2 entre q y r. De manera similar, definimos una situación donde el rayo $\overrightarrow{pq}$ **toca primero** a L_2.

Un rayo intersecta a una curva si hay puntos de la curva en la izquierda y así y en la derecha del rayo, y el conjunto intersección del rayo y la curva es no vacía. Hay que notar que el rayo puede ser incidente con todo un segmento recto de la curva, esto es que el punto r puede ser cualquier punto de tal segmento en este caso. Ahora supongamos una línea delimitante L_1 y una línea delimitación de L_2 de una frontera abierta y sea p_0, p_1, p_{n-1} el MLP rodeando a L_1 en la cerradura de estas frontera abierta. Entonces, satisface que el rayo $\overrightarrow{p_i p}_{i+1(mod\ n)}$ toca a L_1 primero si $p_{i+1(mod\ n)} \in L_2$, o el rayo $\overrightarrow{p_i p}_{i+1(mod\ n)}$ toca L_2 primero si $p_{i+1(mod\ n)} \in L_1$, para $i = 0, 1, ..., n-1$.

Capítulo 4

Algoritmos para determinar la longitud de curvas digitales

En este capítulo describimos con detalle los algoritmos DSS y el MLP, dara la implementacion de ambos señalando detalles especificos y mostrando ejemplos de cada uno. Ademas, para representar las imagenes en la computadora se usara la estructura de la malla estandar, que se vera mas adelante.

En aplicaciones para el análisis de imágenes la secuencia es generada durante el proceso de rastreo, crack por crack, una línea delimitante de una región digitalizada. La entrada de ambos algoritmos es una secuencia de N puntos de una malla $P[1..N]$ en una línea delimitante de una región digital S, cada punto es descrito por la estructura de datos $(P[i].X, P[i].Y)$ que contiene las coordenadas enteras del $i-$ésimo punto $P[i]$. Ya que una línea delimitante es un ciclo cerrado $P[N]$ debe ser igual a $P[1]$. Como la secuencia de puntos de entrada de una malla se supone que sigue a una línea delimitante de una región digital, las coordenadas de los dos subsecuentes puntos en la secuencia con índices i e $i+1$ deben satisfacer la condición $[X_{i+1} - X_i] + [Y_{i+1} - Y_i] = 1$.

Para ambos algoritmos la salida es un valor $Perim$ el cual es el estimado DSS o el estimado MLP del perímetro de una región digital S. En los Apendices E,2 a E,5 se presentan la teoria y las estructuras de datos que proporcionan de manera explicita la información topológica para los complejos 2D y 3D respectivamente, sin embargo estas estructuras no son completamente necesarias para resolver el problema.

4.1. Malla Estandar.

Las imágenes de 2 y 3 dimensiones son almacenadas en la computadora usualmente en arreglos de la dimensión correspondiente. Cada elemento del arreglo contiene ya sea un valor de gris, un color o una densidad. A esta estructura se le conoce como **malla estándar**. Esta estructura de datos no está diseñada para soportar operaciones topológicas, sin embargo es posible realizarlas sin cambiar la estructura. Por ejemplo, es posible trazar y codificar la frontera de una región en una imagen $2-$dimensional a pesar de la apa-

rente dificultad que presenta con relación a la Definición 2.5(Frontera), que consiste de 0−celdas y 1−celdas, mientras que la malla contiene sólo pixeles los cuales se interpretan como 2−celdas. Esta interpretación se debe a que un pixel contiene una característica óptica como la brillantez o el color que es proporcional a cierta área. Podemos ver que el rastreo en la malla estándar es posible pues el concepto de un complejo AC es la manera de ver a las propiedades topológicas de una imagen digitalizada en lugar de una manera de codificarlas.

Cuando pensamos en una imagen 2−dimensional como un complejo AC 2−dimensional, esté contiene celdas de dimensión cero a dos. Las 2−celdas (pixeles) en la malla estándar tienen coordenadas enteras las cuales, a diferencia de las coordenadas topológicas, que son siempre impares, pueden tomar cualquier valor entero, par o impar. Los pixeles son representados explícitamente en la malla estándar, mientras que las 0−celdas y 1−celdas son representadas implícitamente.

La correspondencia entre las coordenadas combinatorias y estándar está definida por la **Regla de Asignación de Coordenadas**. La regla nos dice que cada pixel f de una imagen 2D recibe una 0−celda asignada a este como una celda *propia*. Esta es la 0−celda situada en la esquina de f la cual es la más cercana a las coordenadas de origen. También se la asignan las 2−celdas incidentes a la 0−celda como propias. Así cada pixel recibe tres celdas propias de baja dimensión, además todas las celdas de f reciben las mismas coordenadas de f.

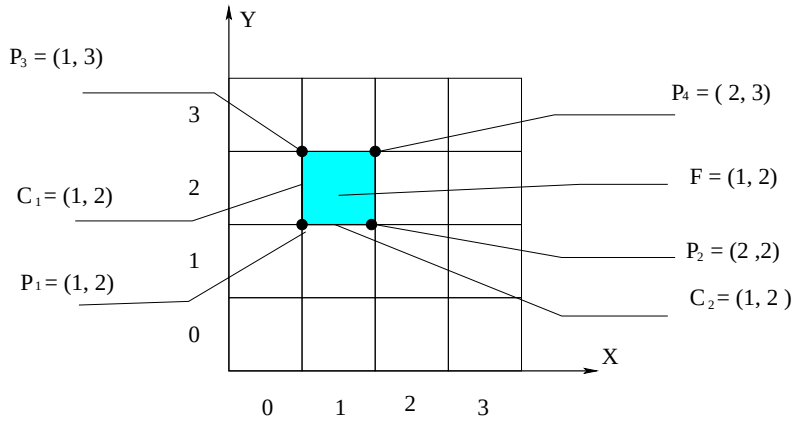

Figura 4.1: Coordenadas no combinatorias de celdas de dimension menor.

En la Figura 4.1 podemos observar la Regla de Asignación de Coordenadas aplicada al pixel F, donde la 0−celda P_1 y las 1−celdas C_1 y C_2 corresponden a sus celdas propias y, por lo tanto, tiene las mismas coordenadas que el pixel.
La regla de asignación de coordenadas puede extenderse para abarcar celdas 3−dimensionales de la siguiente manera: cada voxel obtiene siete celdas propias de baja dimensión las cuales son asignadas de manera similar a la del caso 2−dimensional. A éstas se les asignan las mismas coordenadas de las de su correspondiente voxel. En el caso n−dimensional el número de las celdas propias de una celda n−dimensional es $2^n - 1$. En el Apéndice E,1 se presenta otra forma para representar las imagenes en las computadoras.

4.2. Trazado de la frontera en imagenes 2D.

El trazado de la frontera se vuelve extremadamente simple cuando pensamos en una imagen 2D como un complejo 2D. La estrategia del algoritmo consiste en lo siguiente: en cada punto P (0−celda) de la frontera se busca el siguiente crack (1−celda) C de la frontera incidente a P y se da un paso a lo largo del crack hacia el siguiente punto de la frontera. Se repite este procedimiento hasta alcanzar el punto inicial nuevamente. El algoritmo TRACE sigue la frontera de una región del primer plano comenzando y terminando en el mismo punto dado (x, y). Como utilizamos la malla estandar los puntos y cracks son representados implicitamente. TRACE siempre comienza en la direccion del eje Y positivo.

Después de cada movimiento a lo largo del crack C, de la frontera los valores de los pixeles R y L del $SON(P)$, con P el punto final de C, deben probarse ya que los valores de los otros pixeles del $SON(P)$ ya han sido probados durante el movimiento previo.

Nótese que los cracks en un complejo cartesiano 2D sólo tiene cuatro diferentes direcciones. Por lo tanto, la variable *direction* toma los valores del 0 al 3. La Figura 4.2 (a) muestra las cuatro posibles direcciones de un crack; en (b) se presentan los pixeles derecho e izquierdo.
El punto que corresponde a la esquina de un pixel, el cual es el más cercano al origen, tiene las mismas coordenadas que el pixel.

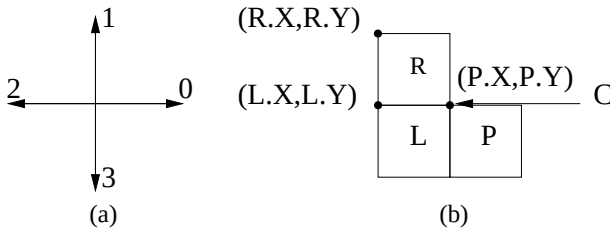

Figura 4.2: (a)Las cuatro direcciones de un crack y (b) los pixeles derecho e izquierdo.

Listado 1 Trace

```
Trace(array image; int x, y){
  P.X = x; P.Y = y; direction = 3;
  do{
    R = P + right[direction];// R es el pixel derecho.
    L = P + left[direction];// L es el pixel izquierdo.
    if(image[R] == foreground)
        direction = (direction + 3) mod 4;// giro derecho.
    else{
        if(image[R] == background)
           direction = (direction + 1) mod 4;//giro izquierdo.
    }
    P = P + step[direction];// un movimiento en la nueva direccion.
  }while( P.X != x || P.Y != y );// end while
}// end Trace
```

El algoritmo TRACE toma como parámetro un arreglo 2D $Image[NX, NY]$ que representa una imagen 2D, en malla estandar, cuyos elementos contienen valores de grises o colores. Las variables P, R, L y los elementos de los arreglos $right[4]$, $left[4]$ y $step[4]$ son estructuras representando vectores 2D con coordenadas enteras. La operación '+' se refiere a la suma de vectores.

Implementación

Antes de empezar a calcular la frontera hay que recordar que en la computadora las coordenadas se presentan de una manera distinta a la que estamos acostumbrados. En la Figura 4.3 se observan ambas representaciones de las coordenadas.

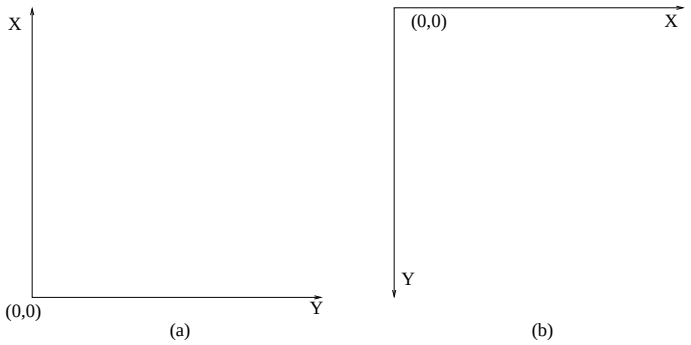

Figura 4.3: Representación de las coordenadas: (a) tradicional; (b) computacional.

Esto es muy importante ya que si recordamos **Regla de Asignación de Coordenadas**, ésta nos dice que cada pixel de una imagen 2D tiene una celda *propia*,la cual es, la $0-$celda que está en la esquina más cerca al origen. Por lo tanto, en las coordenadas de la computadora la celda propia de los pixeles se encuentra en la esquina superior izquierda. En la Figura 4.4 se muestran las celdas propias tanto en las coordenadas usales como las de la computadora.

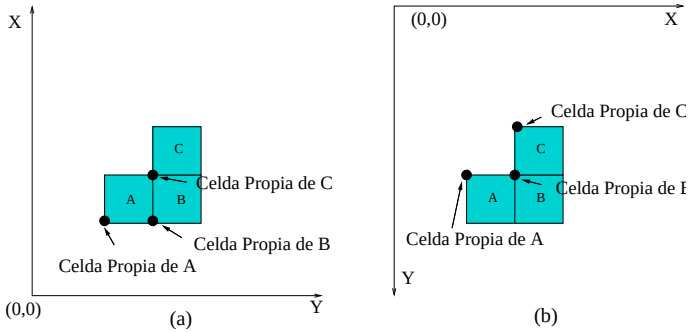

Figura 4.4: Las celdas propias de los pixeles: (a) coordenadas normales, (b) coordenadas de la computadora.

De acuerdo con la regla de asignación de coordenadas la 0−celda propia tiene las mismas coordenadas de su pixel. Sin embargo, una 0−celda, en coordenadas combinatorias, son pares y los pixeles de la imagen tienen coordenadas tanto pares como impares. Afortunadamente basta con multiplicar por dos las coordenadas del pixel para obtener las de la 0−celda. A pesar de lo anterior, podemos utilizar directamente las coordenadas del pixel sin que afecte el procedimiento.

Ahora que hemos establecido las condiciones en la que se presentan los datos, podemos pasar al algoritmo. Dentro de las variables de algoritmo tenemos los arreglos $right[4]$, $left[4]$ y $step[4]$. Los primeros dos arreglos contiene los pixeles derechos e izquierdos de la 0−celda con respecto a la direccion actual, en la Figura 4.5 podemos ver las 4 direcciones posibles y en las demas podemos ver los pixeles derecho e izquierdo correspondientes a esa dirección, en algunas ocasiones R o L es igual a P. El arreglo *step*, en cambio, nos dice cuanto nos movemos en la dirección actual para llegar a la siguiente 0−celda en la frontera.

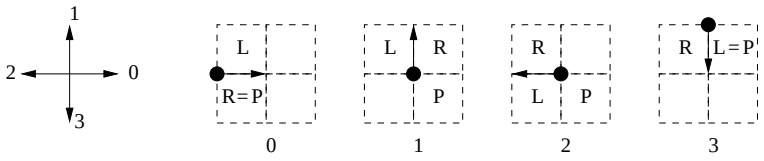

Figura 4.5: Las direcciones posibles y las posiciones de los pixeles derecho e izquierdo.

En la Figura 4.6 (a), comenzamos el trazado del frontera en la 0−celda propia $I = (7, 1)$ con la dirección 3, entonces de acuerdo con el algoritmo vemos si R es igual al valor del primer plano y si I es igual al valor del fondo, en este caso R es igual al valor de fondo e I es igual al valor del primer plano, por lo tanto no hay un cambio de dirección y avanzamos, en la dirección 3, la cantidad que nos indique *step*, que es una unidad en la coordenada Y, entonces tenemos que la 0−celda propia del pixel (7,2) está en la frontera.

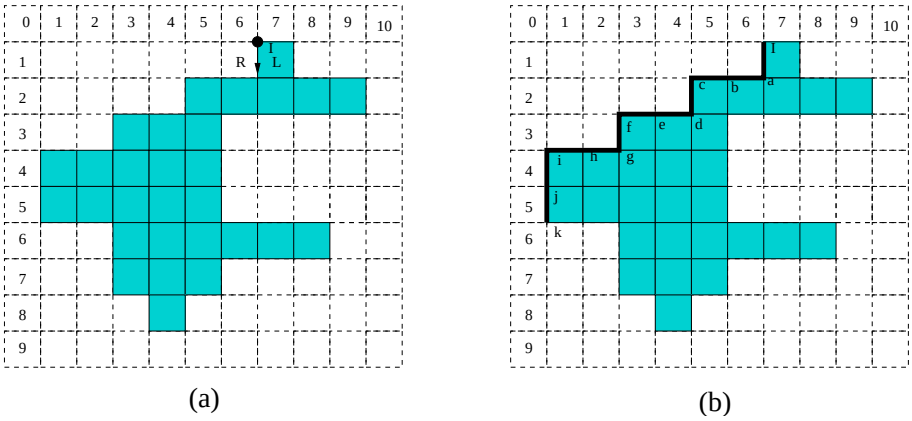

Figura 4.6: Ejemplo del trazado de la frontera.

Al inicio del siguiente ciclo comenzamos en el pixel (7,2) con la dirección es 3. Obtenemos los valores de R e I, revisamos si corresponden al valor del primer plano y del fondo,

respectivamente; en este caso R es igual al valor del primer plano y entonces tenemos que hacer un cambio de dirección. Al final tenemos que la nueva dirección es 2, ahora avanzamos en este dirección la cantidad que *step* indique en esta dirección, que es -1 en la coordenada X, y por lo tanto la $0-$celda propia del pixel (6,2) está en la frontera. En la Figura 4.6 (b) podemos ver el resultado del algoritmo después de 11 ciclos, de esta manera seguimos hasta alcanzar nuevamente el vértice inicial. Finalmente, en la Figura 4.7 podemos ver la frontera contruida por el algoritmo.

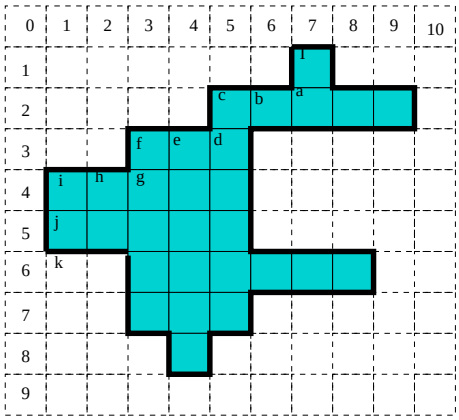

Figura 4.7: La frontera final de la figura.

4.3. Algoritmo DSS.

La tarea consiste en el rastreo de una curva digital $P[1:N]$ crack por crack y de manera secuencial dividirla en DSS de máxima longitud. El algoritmo DSS realiza los siguientes pasos:

(i) Inicializar un DSS con un crack;

(ii) Revisar si el punto actual de la malla aún pertenece al último DSS, durante la prueba de rastreo de la línea delimitante;

(iii) Ajustar el parámetro del DSS, si el punto actual no pertenece al DSS, de tal manera que el DSS contenga todos los puntos previos; esto es, desde el principio del último DSS hasta el punto anterior, así como el último punto;

(iv) Cerrar el último DSS y generar un mensaje de término, si no hay ajuste posible.

El algoritmo DSS es presentado en el Listado 2.

Listado 2 Algoritmo DSS.

```
DSS_Perim(array P){
        N_DSS = -1;//Indice del primer DSS distinto 0.
        Perim = 0.0;
        for(i=2 to N){
                dir = F(P[i]-P[i-1]);//Direccion del ultimo paso
                                     //especificada por la funcion F()
                if(i=2){
                        BRK = 1;//Inicializacion del parametro al inicio.
                }else{
                        BRK = DSS_Count(P[i],dir);//
                }
                if(BRK > 0)
                {
                        StartL = StartR = P[i-1];
                        EndL = EndR = P[i];
                        Tang = P[i] - P[i-1];
                        DIR[1] = dir;
                        DIR[2] = -1;//Se desconoce DIR[2].
                        HR = 0;//HR es igual a H(X,Y)
                        Vertex = P[i-1];
                        N_DSS = N_DSS + 1;
                        if( i > 2){
                                //Distancia Euclidiana.
                                Perim = Perim + Distance(Vertex, P[i]);
                        }//end if
                }//end if
        }//end for
        N_DSS = N_DSS + 1;
        Perim = Perim + Distance(Vertex,P[i]);
}// end DSS_Perim
```

El algoritmo usa las siguientes variables:

1. $DIR[1:2]$: arreglo que contiene ambas direcciones permitidas por el último DSS;

2. $StartL$, $StartR$, $EndL$, $EndR$: puntos iniciales y finales de las bases izquierda y derecha, respectivamente cada punto es descrito por una estructura de datos (int:X,int:Y) que contiene sus coordenadas;

3. $Vertex$: punto final del DSS previo, es descrito por una estructura (X, Y);

4. $Tang$: vector tangencial que especifica la dirección de las bases del reciente DSS y está descrito por una estructura similar, con $Tang.X = U$ y $Tang.Y = V$;

5. HR: valor entero de $H(X, Y)$ del lado izquierdo de la desigualdad del semiplano Euclidiano cuya frontera o borde es incidente con la base derecha del DSS mientras HR sea positivo para los puntos de la malla del DSS; y

6. N_DSS: índice (etiqueta) del DSS que se acaba de detectar,
   i : índice del último punto,
   dir: dirección del último crack, y
   BRK: bandera para comenzar el siguiente DSS si $BRK = 1$.

La función DSS_Count determina si el DSS actual puede ser extendido o no. Si sí entonces los parámetros deben ser ajustados, si es necesario.

La función entera DSS_Cont es llamada varias veces por el algoritmo $DSS\_Perimeter$, tiene como entrada el último punto $P[i]$ descrito por la estructura de datos $(P.X, P.Y)$ que contiene las coordenadas de $P[i]$ y la dirección dir del último crack cuyo punto final es $P[i]$. La salida es igual a 0 si $P[i]$ pertenece al último DSS ajustado, o es igual a 1 si no existe un DSS que contenga el último punto $P[i]$ y todos los puntos anteriores. El Listado 3 presenta un bosquejo del algoritmo DSS_Count.

Listado 3 Algoritmo DSS_Count.

```
DSS_Count(array P,int  dir){
        if(DIR[2] < 0 && dir = DIR[1]){
                EndL = EndR = P; return 0; //valor permitido de HR }
        if(DIR[2] >= 0 && dir != DIR[1] && dir != DIR[2])
                return 1; //Una tercera direccion esta prohibida
        if(DIR[2] < 0 && dir != DIR[1]) DIR[2] = dir;
        //Se encontro una segunda direccion
        //Actualizar el valor del lado izquierdo de la desigualdad
        switch(dir){
                //HR es igual a H(X,Y)
                case 0: HR = HR + Tang.Y; break;
                case 1: HR = HR - Tang.X; break;
                case 2: HR = HR - Tang.Y; break;
                case 3: HR = HR + Tang.X; break; }
        if(HR > 0 && HR < abs(Tang.X) + abs(Tang.Y) - 1)
                return 0; //Valor permitido
        if(HR < -1 || HR > abs(Tang.X) + abs(Tang.Y))
                return 1; //Valor no ajustable
        if(HR = 0){ //P esta en la base derecha
                EndR = P; return  0;}
        if(HR = abs(Tang.X) + abs(Tang.Y) - 1){
                //P esta en la base izquierda
                EndL = P; return 0; }
        if(HR = -1 ){//P es un outliner derecho
                EndR = P; StartL = EndL;  Tang = P - StartR;
                HR = 0; return 0; }
        if(HR = abs(Tang.X) + abs(Tang.Y)){
                //P es outliner izquierdo
                EndL = P; StartR = EndR; Tang = P - StartL;
                HR = (P.X - StartR.X) * Tang.Y -
                     ( P.Y - StartR.Y ) * Tang.X;
                return 0; }
}//end DSS_Count
```

Implementación

Analizando el algoritmo DSS se ecuentra la función F que indica la dirección del último paso especificado. Para poder implementar la función, se necesita primero establecer las direcciones que se van a usar.

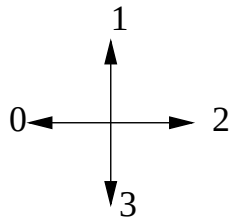

Figura 4.8: Las cuatro direcciones del algoritmo DSS

En la Figura 4.8 se muestran las direcciones que se utilizan en el algoritmo DSS. Una vez establecidos estos valores se puede implementar la función.
El algoritmo recibe como parámetro dos celdas de las cuales se quiere determinar la dirección. Para esto se resta a la celda $P[i]$ el valor de la celda $P[i-1]$ y la se almacena en otra celda auxiliar. Si la coordenada X de esta celda es 0 entonces la dirección es determinada por la coordenada Y. Si Y es menor que 0 la dirección es 1; en cambio, si Y es manor que 0 entonces la dirección es 3. Si la coordenada Y de la celda es igual a 0 entonces la dirección es determinada por la coordenada Y. Si Y es menor que 0 la dirección es 0; en cambio, si Y es mayor que 0 entonces la dirección es 2.

Listado 4 Funcion F

```
F(celda A, celda B){
        tmp = A - B;
        if( tmp.X = 0){
                if( tmp.Y > 0 ) return 3;
                if( tmp.Y < 0 ) return 1;
        }else{
                if(tmp.Y == 0)
                {
                        if( tmp.X > 0 ) return 2;
                        if( tmp.X < 0 ) return 0;
                }
        }
}// end F
```

Entonces, el procedimiento para calcular el perímetro de la Figura 4.9 es el siguiente: se inicia el algoritmo en la 0−celda I con coordenadas (7,1), en este caso por conveniencia se emplean los coordenadas combinatorias en todas las 0−celdas de la frontera, por lo tanto se tiene al final que las coordenadas de la 0−celda son (14,2). El ciclo 1 comienza por inicializar la bandera N_DSS con -1 y el perímetro con 0; después se calcula la dirección

actual con la función F con las celdas (14,2) y (14,0), la cual es 3. Como se está en el primer ciclo $i = 2$, por lo tanto se obtiene $BRK = 1$. Como $BRK > 0$, entonces se tienen los siguientes valores:

$SR = (14, 0)$, $ER = (14, 2)$, $SL = (14, 0)$, $EL = (14, 2)$, $Tang = (0, 2)$,
$Vertex = (14, 0)$, $Dir[1] = 3$, $Dir[2] = -1$, $HR = 0$, $N\_DSS = 0$.

Finalmente, como i es menor que 2 aun no calculamos el perímetro.

Al inicio del siguiente ciclo se tiene que $i = 3$ y la dirección dada por las 0−celdas (12,2) y (14,2) es $dir = 0$. Como i es distinto de 2 entonces se calcula el valor de BRK con DSS_Count. La función DSS_Count toma como parámetros la celda y dirección actuales (12,2) y dir=0. Al comienzo de la función se observa que $Dir[2] < 0$ y $dir \neq Dir[1]$ así que $Dir[2] = dir = 0$. Como $dir = 0$ entonces $HR = HR - Tang.Y$, $HR = 2$. Sin embargo, como estamos usando coordendas combinatorias, lo mejor es convertirlas en coordenadas cartesianas dividiéndolas por 2. Así, $HR = 2$. El valor de $HR = 2$ es igual a $|Tang.X|$ + $|Tang.Y|$, tomando en cuenta que los valores de $Tang$ son divididos por 2, entonces se deben ajustar los parámetros de DSS actual para incluir la nueva celda, de estas manera tenemos que:

$Dir[1] = 3$, $Dir[2] = 0$, $HR = -2$, $BRK = 0$, $SR = (14, 2)$, $ER = (14, 2)$,
$SL = (14, 0)$, $EL = (12, 2)$, $Tang = (-2, 2)$, $Vertex = (14, 0)$, $N\_DSS = 0$.

Volviendo a la ejecución del algoritmo del perímetro como $BRK = 0$ entonces se termina el ciclo actual. Al final del tercer ciclo se observan los valores:

$Dir[1] = 3$, $Dir[2] = -1$, $HR = 0$, $BRK = 1$, $N\_DSS = 1$, $SR = (12, 2)$,
$ER = (12, 4)$, $SL = (12, 2)$, $EL = (12, 4)$, $Tang = (0, 2)$, $Vertex = (12, 2)$

Como $N\_DSS = 1$, significa que ha comenzado un nuevo DSS y se calcula la distancia de (14,0) a (12,2), sumando este valor al perímentro actual.

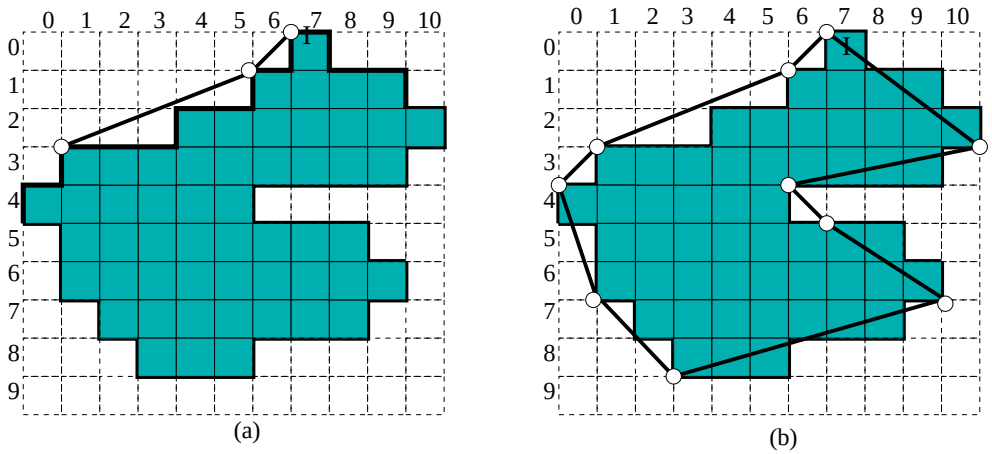

Figura 4.9: Ejemplo del algoritmo DSS.

El Figura 4.9 (a) se observa el algoritmo al final del ciclo 10. La Figura 4.9 (b) se muestra todos los DSS de la figura original.

4.4. Algoritmo MLP.

Este método se basa en encontrar el polígono de longitud de mínima o MLP, por sus siglas en ingles, el cual está contenido en la cerradura de una frontera abierta de una región digital S y la cual circunscribe la línea delimitante de la región. La tarea también puede especificarse como calcular un MLP en un conjunto compacto poliédrico del plano euclidiano que tiene dos curvas poligonales disjuntas isotéticas; es decir, que sólo tiene aristas paralelas a los ejes de coordenadas, como su frontera o borde, la línea delimitante L_1, la curva digital dada $P[1 : N]$, como un *borde o frontera interna* y la línea de demarcación L_2, una curva digital $Q[1 : M]$ calculada a partir de $P[1 : N]$, como su *borde o frontera externa*. Mediante el prepocesamiento aseguramos que ambas curvas definen la frontera o borde de una frontera abierta simple, una malla continua de ciclo cerrado simple $1-$dimensional.

Las propiedades descritas en la Sección 4.4 nos dan las bases teóricas de la correctez del algoritmo lineal para el problema MLP. El algoritmo de *Dos Insectos* [1] es el siguiente:

Hay dos *insectos*, Blanco y Negro. Ambos pueden moverse hacia adelante en una dirección, digamos en contra de las manecillas del reloj. El punto de inicio del primer recorrido es el vértice superior izquierdo de la borde o frontera interna el cual es también el primero del MLP. Sólo entonces, el Negro está en la línea delimitante. Después del inicio del recorrido, al Negro sólo se le permite moverse a los vértices cóncavos de la línea de demarcación y al Blanco sólo se le permite moverse a los vértices convexos de la línea delimitante, y para cualquier nueva posición de un *insecto*, ambas líneas rectas *Vértice Inicial* a Negro y *Vértice Inicial* a Blanco no se les permiten intersectar ni a la línea de demarcación ni a la línea delimitante. Los *insectos* se mueven de manera alternante, en la medida en que a ambos se les permite moverse y detenerse si ningún movimiento adicional es posible. El Blanco inicia el primer recorrido. Todos los recorridos se inician por el ganador del recorrido anterior. Si un *insecto* no se pudo mover en absoluto durante el recorrido el otro *insecto* es el ganador. Si ambos se mueven, entonces el ganador es el aquel cuyo rayo *Vértice Inicial* a *insecto* toca primero la línea del otro *insecto*. La posición del *insecto* ganador es la del siguiente vértice del MLP y también es el vértice inicial para el siguiente recorrido. El otro *insecto* inicia desde su última posición aceptada en el recorrido anterior. La secuencia de recorridos se detiene si el Blanco alcanza el primer vértice del MLP de nuevo.

Podemos ver en la Figura 4.10 un ejemplo del algoritmo MLP, en la primera figura se muestra al ganador del primer recorrido, el insecto Blanco. En la segunda figura, el ganador del quinto recorrido es el insecto Negro, el rayo que va del vértice de inicio al Negro, toca primero a L_1 en el punto indicado.

Para la implementación de este algoritmo se debe especificar:

(i) cómo clasificar vértices de la línea en convexos, cóncavos o puntos colineales;

[1] El algoritmo original se llama $two - beetle$.

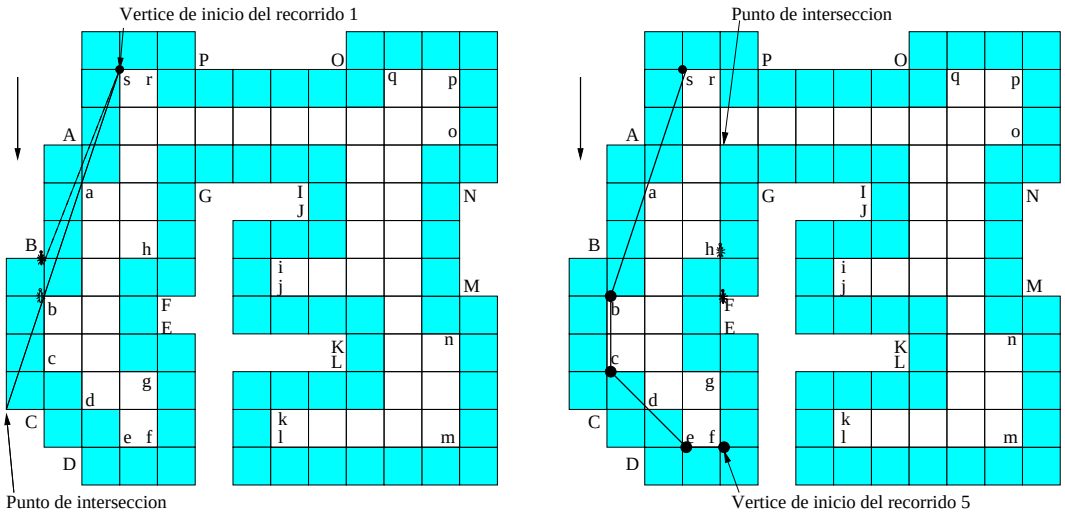

Figura 4.10: Ejemplos del algoritmo MLP.

(ii) cómo asegurar que los segmentos de las rectas *Vértice Inicial* a *Insecto* no intersecten a L_1 o L_2; y

(iii) decidir si un rayo *Vértice Inicial* a *Insecto* toca L_1 o L_2. El intervalo de búsqueda se limita en realidad a la posición final del otro *insecto* y su siguiente posición de prueba; esto es, el siguiente vértice cóncavo o convexo.

La decisión (i) es fácil de implementar usando los valores direccionales $det(p_1, p_2, p_3)$. Los procedimientos para la decisiones (ii) y (iii) usa las variables:

1. SV: el vértice de inicio del recorrido;

2. W: última posición aceptada de Blanco,
   TW: siguiente posición de prueba de Blanco,
   NW: número de pasos de Blanco en el recorrido hasta el momento,
   $Wmove$: bandera, con valor 1 si se esperan más posibles movimiento de Blanco en este recorrido; y con valor 0 si no se esperan más movimientos;

3. $B, TB, NB, Bmove$: variables correspondientes para Negro;

4. BAY y PEN: banderas para situaciones *bahía* o *península*, respectivamente; [2];

5. $NextMove$: indica quien hará el siguiente movimiento Blanco o Negro.

Además, sean: $next\_convex(P, i)$: índice del siguiente vértice convexo, módulo N, en P que sea distinto a i; y $next\_concave(Q, i)$: índice del siguiente vértice cóncavo, módulo M, en Q, distinto a i. El listado del Algoritmo 5 nos muestra la prueba para la siguiente posicion del blanco utilizando las variables descritas anteriormente.

[2]Si la imagen tiene forma de península o bahía.

Listado 5 NextWhite

```
NextWhite(array P, Cell2 SV, Cell2 W, Cell2 B, Cell2 TW, Cell2 TB,
          int BAY, int PEN, int NW, int Wmove, String NextMove){

        nextConvex = next_convex(P, nextConvex);
        TW = P[nextConvex];
        if( NB = 0 || det(SV,W,TW) != rightTurn &&
            NW = 0 || det(SV,W,TW) != leftTurn &&
            PEN = 0 ||
            un 4-vecino de TB tiene un indice >= nextConvex){
                W = TW;
                NW = NW + 1;
                NextMove = Black;

        }//end if
        else{
                if( NB > 0 && det(SV,BTW) = rightTurn )
                {
                        BAY = 1;
                }//end if
                Wmove = 1;
        }//end else
}//end NextWhite
```

La prueba para la siguiente posición Negro es análoga a este procedimiento e incluye una posible detección de una *situación de península* dando lugar a $PEN := 1$.

La decisión (iii) sobre el insecto ganador se muestra en el listado del Algoritmo 6.

Listado 6 Metodo para determinar el ganador del recorrido.

```
if( NB = 0 || det(SV,W,TB) = leftTurn && det(SV,W,B) != leftTurn)
{
        Winner = White;
}//end if
else{
        Winner = Black;
}//end else
```

La posición del insecto ganador es el siguiente vértice del MLP y la nueva posición inicial del siguiente recorrido. Finalmente, el perímetro estimado MLP es la longitud del polígono longitud mínima.

Implementación

A diferencia del algoritmo DSS que sólo requiere la frontera, el algoritmo MLP además requiere de la línea de demarcación, esta línea se calcula a partir de la frontera abierta. Por lo tanto, es necesario describir un método para el cálculo de la línea demarcación. El método propuesto sigue la misma estrategia usada en el cálculo de la frontera anterior, excepto que en este caso se utiliza una variable y arreglo adicional: P^* y $P^*[4]$.

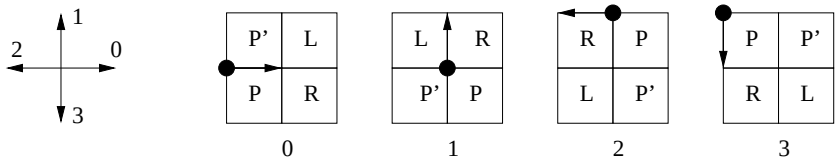

Figura 4.11: Las direcciones y las posiciones de los pixeles R, L y P^*.

En la Figura 4.11 (a) se ilustran las cuatro direcciones posibles, que son las mismas que se usan en el cálculo de la frontera; en (b) se presentan los pixeles R, L y P^* correspondientes a cada dirección, se observa que P^* siempre está al lado de P.

Listado 7 Trace

```
Trace(array image; int x, y){
  P.X = x - 1; P.Y = y - 1; direction = 3;
  do{
    R = P + right[direction];// R es el pixel derecho.
    L = P + left[direction];// L es el pixel izquierdo.
    P^{*} = P + p^{*}[direction];//P^{*} es el pixel que esta siempre
                                 //al lado de P
    if( image[R] = foreground &
        image[L] = foreground &
        image[P^{*}]=foreground )
        direction = (direction + 3) mod 4;// giro derecho.
    else{
        if( image[R] = background &
            image[L] = background &
            image[P^{*}] = background )
           direction = (direction + 1) mod 4;//giro izquierdo.
    }
    P = P + step[direction];// un movimiento en la nueva direccion.
  }while( P.X != x || P.Y != y );// end while
}// end Trace
```

Como se observa el algoritmo propuesto para el cálculo de la línea de demarcación es practicamente igual, excepto que en éste debemos calcular los valores de los tres pixels R, L y P^*, ya que el pixel P no se encuentra en la figura sino que está en la frontera abierta por lo tanto el número de pixeles con igual valor debe ser menor de tres, ya que si los tres pixels fueran iguales, o se está fuera de la frontera abierta o se está totalmente contenido

en la imagen. En Figura 4.12 a se tiene el caso en el que los tres pixeles son de fondo, así que se debe cambiar de dirección, que en este caso es la dirección 3, en Figura 4.12 (b) se tiene el caso en el que los tres pixeles son del primer plano y, por lo tanto, se debe cambiar también la dirección a 3.

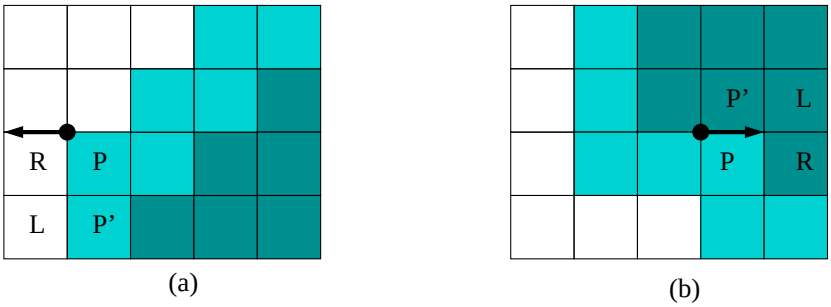

Figura 4.12: Ejemplos de porque se necesita verificar los 3 vertices.

Una vez se ha definido el método para calcular la línea de demarcación, se pueda pasar a ver la implementación del algoritmo MLP. Antes de comenzar a revisar el algoritmo hay que tener en consideración lo siguiente:

- Los valores de las banderas BAY y PEN se mantienen en cada carrera y no se reinician al iniciar una nueva carrera.

- Para poder determinar el insecto ganador tanto $Bmove$ como $Wmove$ deben ser igual a cero.

- Si alguna de las variables $Bmove$ o $Wmove$ es igual a cero pero la contraria es distinta de cero, entonces el insecto distinto de cero aún puede moverse.

- Si W es distinto de TW usamos el valor actual de TW, en caso contrario calculamos el valor de TW con la función $Next\_Convex$.

- Si B es distinto de TB usamos el valor actual de TB, en caso contrario calculamos el valor de TB con la función $Next\_Concave$.

Una vez que se tiene esto, se puede pasar a implementar el algoritmo. Primero se deben ver las funciones $Next\_Convex$ y $Next\_Concave$. Para ambas funciones se usa el determinante: si el valor es positivo, se tienen giros a la derecha, esto significa que, se encontró un vértice concavo; si el valor es negativo, se tienen giros a la izquierda, ya que se encontró un vértice convexo; si su valor es igual a 0, entonces los vértices son colineales.

La función recibe como parámetros la frontera P de la imagen y el índice $nextConvex$ del último vértice convexo aceptado después va recorriendo la frontera en busca del siguiente vértice convexo; para esto se recorre la frontera comprobando que el determinante de las $0-$celdas $nextConvex$, $nextConvex + 1$ y $nextConvex + 2$ sea menor que cero, si el determinante es menor que cero entonces la función regresa $nextConvex + 1$; de lo

contrario, se debe pasar al siguiente vértice de la frontera y repetir el proceso hasta haber recorrido toda la frontera.

Listado 8 Next_Convex

```
Next_Convex( array P, int nextConvex ){
   count = 0;
   while( count<P.Length ){
        if( Det(P[nextConvex],
                P[nextConvex+1%8],
                P[nextConvex+2%8])<0 )
           return nextConvex+1%8;
        count++; nextConvex=(nextConvex+1)% P.Length;
   }
}// end Next_Convex
```

La función *Next_Concave* sigue exactamente el mismo procedimiento, excepto que el determinante debe ser que mayor que cero. Los parámentros que recibe la función *Next_Concave* son la línea de demarcación Q y el índice *nextConcave*. Finalmente, el algoritmo MLPPerim se muestra en el Listado 9.

Listado 9 MLPPerim

```
MLPPerim( array P){
   //Inicializacion de las variables
   Perim=0; SV = P[0]; W = P[0]; B = P[0]; TW = P[0]; TB = P[0];
   NextConvex = 0; NextConcave = 0; NextMove = white; PEN = 0;
   BAY = 0; NW = 0; NB = 0; Wmove = 1; Bmove = 1; initial=W;
   do{
        if( NextMove=white && Wmove!=0 )
          NextWhite(p);
        if( NextMove=black && Bmove!=0 )
          NextBlack(q);
        if( Wmove = 0 && Bmove = 0){
          if( NB=0 ||  Det(SV,W,TB)<0 & Det(SV,W,B)>= 0 ){
             Winner=W; SV=W; NextMove=white;
          }else{
             Winner=B; SV=B; NextMove=black;
          }
          perim = perim + EuclidianDistance(SV,Winner) ;
        }
   }while(W!=P[0] & W!=B);
   perim = perim + EuclidianDistance(SV,initial) ;
}// end Next_Convex
```

A continuación ejemplificamos el procedimiento para calcular el perímetro de la Figura 4.13 con el algoritmo MLP. La asignación inicial de valores es la siguiente:
SV=(14,2), W=(14,2), TW=(14,2), NW=0, B=(14,2),

TB=(14,2), NB=0, $Wmove$=1, $Bmove$=1, $NextMove$=$White$,
$NextConvex$=0, $NextConcave$=0, Bay=0, Pen=0, $Perim$=0.0.

Solo al inicio del algoritmo es la única ocasion en la que el insecto negro está en la frontera. Como se está iniciando entonces el primero en avanzar es el insecto blanco, por lo tanto se llama al método $NextWhite$, al final del método las variables quedan con los siguientes valores:

SV=(14,2), W=(10,4), TW=(10,4), NW=1, B=(14,2),
TB=(14,2), NB=0, $Wmove$=1, $Bmove$=1, $NextMove$=$Black$,
$NextConvex$=3, $NextConcave$=0, Bay=0, Pen=0.

Como $Bmove$=1 y $NextMove$=$Black$, el insecto negro puede moverse entonces se llama a la función $NextBlack$, entonces se tiene los valores:

SV=(14,2), W=(10,4), TW=(10,4), NW=1, B=(12,2),
TB=(12,2), NB=1, $Wmove$=1, $Bmove$=1, $NextMove$=$White$,
$NextConvex$=3, $NextConcave$=1, Bay=0, Pen=0.

Como $Wmove$=1 y $NextMove$=$White$ entonces el insecto blanco aun pueden moverse, se realiza nuevamente la llamada a $NextWhite$, las variables quedan con los siguientes valores:

SV=(14,2), W=(6,6), TW=(6,6), NW=2, B=(12,2),
TB=(12,2), NB=1, $Wmove$=1, $Bmove$=1, $NextMove$=$Black$,
$NextConvex$=6, $NextConcave$=1, Bay=0, Pen=0.

Se continua de hasta que los valores de $Wmove$ y $Bmove$ sean ambos 0. En la cuarta llamada de $NextWhite$ los valores son:

SV=(14,2), W=(2,8), TW=(2,12), NW=3, B=(4,6),
TB=(4,6), NB=3, $Wmove$=0, $Bmove$=1, $NextMove$=$Black$,
$NextConvex$=11, $NextConcave$=7, Bay=0, Pen=0,

sin embargo como $Bmove$=1, el insecto negro aun puede moverse y entonces la carrera aun no termina. En la siguiente llamada de $NextBlack$ tenemos los valores:

SV=(14,2), W=(2,8), TW=(2,12), NW=3, B=(4,6),
TB=(4,14), NB=3, $Wmove$=0, $Bmove$=0, $NextMove$=$Black$,
$NextConvex$=11, $NextConcave$=15, Bay=0, Pen=1.

Como $Wmove$=0 y $Bmove$=0 entonces se puede determinar al ganador de la primera carrera de la forma espeficada anteriormente, por lo tanto el ganador, en el ejemplo, es el insecto blanco y se actualiza el valor de $Perim$=$Distance(SV, W)$ y se tiene ahora que SV=W, inializamos a NB=0, NW=0, $Bmove$=1 y $Wmove$=1. Como el insecto blanco ganó la carrera anterior éste el primero en comenzar en la nueva carrera y queda marcado como vértice del MLP. Se realiza este procedimiento hasta que W sea igual al elemento inicial de la frontera y B sea igual a W.

En la Figura 4.13 de la derecha se puede ver el resultado de la primera carrera y en la de la izquierda está el MLP final.

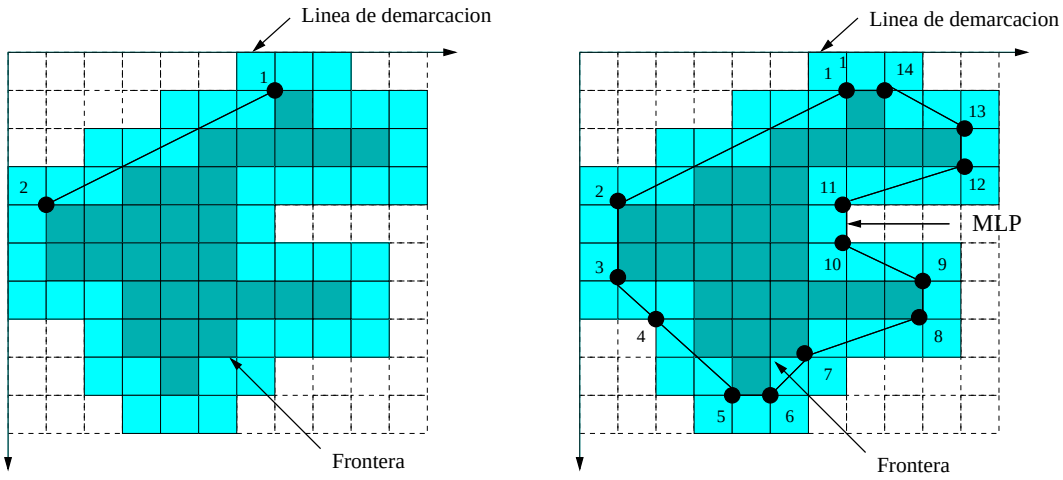

Figura 4.13: Ejemplo del algoritmo MLP.

4.5. Comparacion entre ambos algoritmos.

Como se muestra en el Listado 2, el algoritmo DSS recorre la frontera celda por celda, determinando si la $0-$celda actual está permitida en el DSS actual o se puede ajustar el DSS para incluirla. En el caso en que no está permitida o no se puede ajustar el DSS, entonces se inicia un nuevo DSS que comienza en la $0-$celda actual. Se continua de esta manera hasta haber recorrido toda la frontera. Por lo tanto, el algoritmo DSS es de orden n, con n el numero de $0-$celdas en la frontera de la imagen.

El algoritmo MLP, que se muestra en el Listado 9 no sólo recorre la frontera sino también la línea de demarcación. Esto se debe a que el insecto blanco se mueve en los n' vértices convexos de la frontera, mientras que el insecto negro se mueve en los m' vértices cóncavos de la línea de demarcación. El algoritmo va intercalando el movimiento de ambos insectos ajustando los parámetros en cada ocasión, hasta que el insecto blanco regresa a la posición inicial. Por lo tanto, el algoritmo MLP es de orden $n+m$, donde m es el número de $0-$celdas de la línea de demarcación y n el numero de $0-$celdas de la frontera. Así que ambos algoritmos MLP y DSS calculan el perímetro de las imágenes en tiempo lineal. Sin embargo, en general, el algoritmo DSS es más rápido que MLP con respecto al tiempo de ejecución. Ya que éste último revisa más elementos de la imagen y además calcula la línea de demarcación.

Un punto importante a considerar es que la partición en DSS de una imagen no es única, ya que depende de la posición inicial que se tome en la imagen, además de la orientación, ya sea en sentido de las manecillas del reloj o en sentido contrario. En cambio, el MLP está definido de manera única.

En la Figura 4.14 (a) se observa el MLP de la imagen y en la Figura 4.14 (b) la particion DSS de la misma. Se puede observar que el MLP está más cercano a la frontera mientras que el DSS varía mucho de ésta en algunos puntos, ya sea que se omiten partes de la

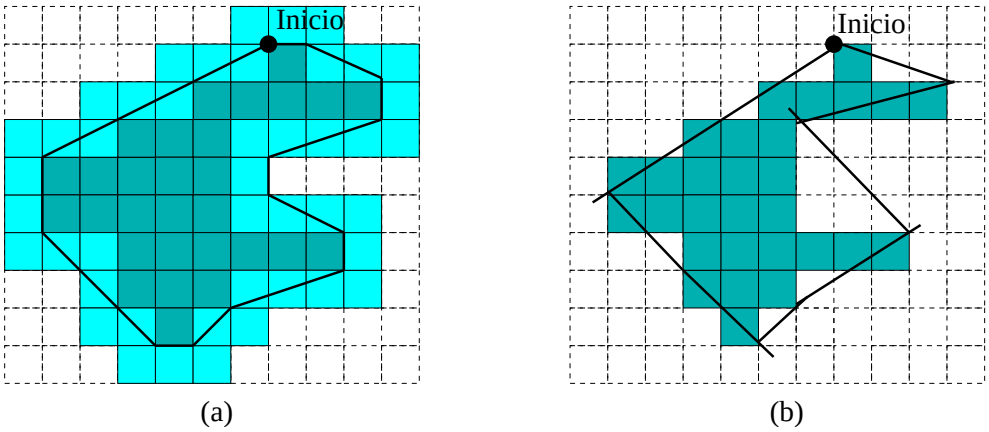

Figura 4.14: Ejemplo de aplicar el algoritmo MLP (a) y el DSS (b) a la misma imagen.

imagen o tramos de la misma. En general MLP está más cercano a la frontera.

A pesar de que MLP es más cercano a la frontera, tiene problemas al tratar con imágenes que tengan huecos de ancho 1 o 2. En la Figura 4.15 (a), al calcular la línea de demarcación, entonces, por definición, las $0-$celdas y $1-$celdas que ligan a los pixeles que están en los huecos A y B no pertenecen a la línea de demarcación. Aunque estos píxeles estén contenidos en la frontera, al ejecutarse el algoritmo MLP omite por completo tales píxeles, esto se debe a que la $0-$celda O es la única que corresponde a un vértice convexo en el hueco A, pero O no puede ganar el recorrido, de lo contrario el rayo del vértice inicial al insecto blanco intersectaría la frontera, hecho no permitido por el algoritmo. Lo mismo ocurre para las $0-$celdas del hueco B. En este caso la aproximación del perímetro se aleja más del valor real. En cambio en la Figura 4.15 (b), el algoritmo DSS no omite los huecos A y B, al no tener las restricciones del MLP, por lo tanto su aproximación al perímetro de la imagen es mejor que la del MLP, en este ejemplo.

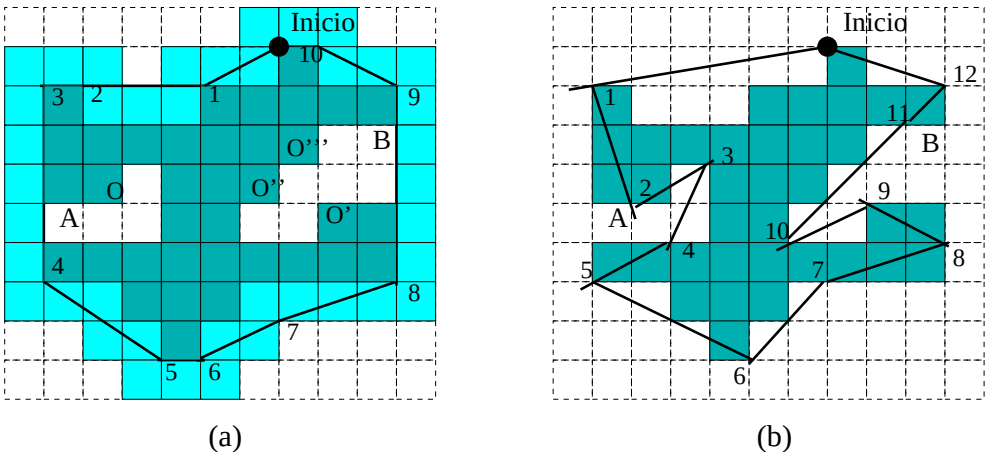

Figura 4.15: Resultado de aplicar el algoritmo MLP (a) y el DSS (b).

Se puede concluir que tanto el algoritmo MLP y DSS ofrecen métodos de tiempo lineal, cada uno de los cuales brinda una buena aproximación al perímetro de las imágenes.

4.6. Métodos para Calcular el Perímetro en la Topología Digital Clásica.

En esta sección se describiran brevemente diferentes enfoques para determinar el perímetro de imagenes digitales, en la topología digital clásica.

Las aproximaciones para el cálculo del perímetro se basan en la reconstrucción del contorno, o borde, y la medición subsecuente de la longitud de éste o bien el conteo de los enlaces de la frontera, los cuales tiene la misma longitud del tamaño píxel.

El primer algoritmo propuesto por Rosenfeld en [2] para el cálculo del perímetro está basado en el conteo de los enlaces de la frontera. Si tenemos una componente conexa C de $1s$, entonces la frontera se define como el conjunto (C, D) de todos los puntos de C que son adyacentes a D, donde D es una componente conexa de $0s$ adyacente a C. Hay que asumir que C es $4-$conexo y que D es $8-$conexo, el procedimiento para el caso opuesto es análogo. Entonces el perímetro $p(C, D)$ se refiere al número de movimientos desde un punto en C al siguiente usando el algoritmo de recorrido estándar de la frontera. Por lo tanto, es necesario ver cómo se define el recorrido de la frontera o borde. La forma estándar de la topología digital clásica para hacer el recorrido del borde dada por Rosenfeld en [1] es la siguiente: Supongamos que tenemos un elemento inicial del borde x_0 de una imagen digitalizada S, para determinar el elemento siguiente x_1, tomamos un y_1 cualquiera del conjunto $F_{x_0} \cap \bar{S}$, donde F_{x_0} es la $4-$vecindad de x_0 y sean los elementos de E_{x_0}, en sentido contrario al reloj, comenzando desde $y_1, y_2, ..., y_s$. Si ninguno de los elementos y_3, y_5, y_7 es 1 entonces x_0 es el único elemento de S. En cualquier otro caso, sea y_{2i+1} el primero de los elementos el cual sea 1; entonces tomamos a $x_1 = y_{2i+1}$ si $y_{2i} = 0$ y $x_1 = y_{2i}$ si $y_{2i} = 1$. En cualquier caso x_1 tiene un $4-$vecino el cual es 0, así que x_1 es un elemento del borde. Este método es bastante simple pero tiende a tomar muchos pasos adicionales y puede visitar a algunos elementos dos veces, además longitud de este método es más grande que el perímetro original en la mayoría de los casos.

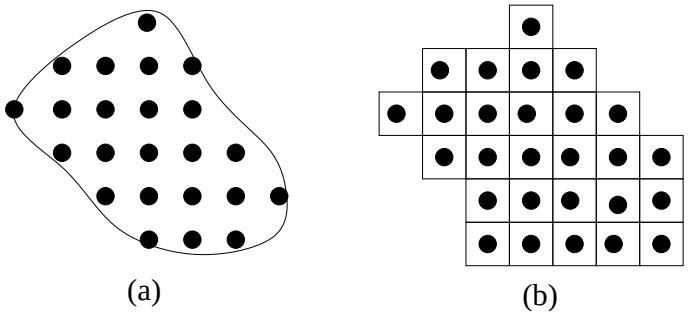

Figura 4.16: Ejemplo de una imagen digitalizada.

La figura 4.16 (a) muestra una imagen y (b) su digitalización, en este caso se puede ver

claramente que la digitalización tiene un perímetro mayor que la imagen original. Los autores Jack Koplowitz y Alfred M. Brucksein, en [4], lo que proponen para mitigar este problema es multiplicar la longitud de la frontera digitalizada por un factor que produciría un error esperado de cero entre el verdadero valor y la aproximación calculada. Este estimador de perímetro tiene una gran variabilidad en el error inducido estimado.
Además de lo anterior se proponen otro método para calcular el perímetro que reduce esta variabilidad, propone calcular el perímetro usando dos parámetros, el número de enlaces y el número de puntos esquina. Esto porque las esquinas contribuyen al sobre valor de la longitud de la frontera. Entonces podemos tener al perímetro de la forma

$$\hat{L} = S_l * (No.enlaces) - S_c * (No.esquinas) = \psi_n * N_n + \psi_c * N_C$$

donde N_n es el número de puntos no esquina y N_c es el número de puntos esquina de la frontera digitalizada. Los factores de peso S_l y S_c o su equivalente ψ_n y $\psi_n c$, respectivamente, se determinan bajo algún criterio que se desee, como el insesgamiento. Encontrar los números N_n y N_c pueden determinarse fácilmente en un ángulo arbitrario θ. Finalmente podemos decir que la longitud de una línea recta L tiene la longitud digitalizada de $L * cos(\theta) + L * sen(\theta)$. Los coeficientes ψ_n y $\psi_n c$ son determinados de tal manera que el error promedio y el el MSE ($mean - squareerror$) sean cero para las líneas rectas distribuidas uniformemente. Los valores óptimos para los coeficientes, dados por los autores, son $\psi_n = \pi * (\sqrt{2}+1)/8$ y $\psi_c = \pi * (\sqrt{2}+2)/8$. Este mismo desempeño se puede obtener cortando las esquinas; esto es, remplazando los enlaces perpendiculares con un enlace diagonal para pasar de 4 direcciones a 8 direcciones. Finalmente, se multiplica el perímetro digitalizado por el factor $(\sqrt{2}-1) * 8/\pi$ para producir un error esperado de cero.

Otros autores mencionan otra forma para calcular el perímetro usando la estructura de datos quadtree. Este metodo hace uso de la correlación $2-$dimensional universalmente existe entre los elementos en cualquier imagen, para poder reducir los datos requeridos para representar una imagen. Sin embargo, la necesidad de insistir en la simetría de las divisiones del quadtree lo perjudica y, por lo tanto, reduce la eficiencia de los algoritmos basados en quadtrees para calcular el perímetro.
Finalmente el autor Huang Wei en [3] propone otra forma para calcular el perímetro es conocida la representación de imagen no simétrica de anti-empaquetado o $NAIR$[3], por sus siglas en inglés. Este método se basa en el modelo no simetrico de anti-empaquetamiento o NAM[4], por sus siglas en inglés, que es la contra parte del problema de empaquetamiento NP-Dificil. En NAM se nos da un patrón y la finalidad es encontrar la secuencia de algunos prototipos predefinidos de tal manera que la disposición de estos ejemplares cubran completamente el patrón sin sobreponerse. $NAIR$ es el resultado de aplicar NAM a una imagen. El $NAIR$ de una imagen se denota como una lista lineal $Q = (p1, p2, ..pk)$ donde cada pi en Q es una instancia de cierto prototipo δj en $\Delta = \{\delta 1, \delta 2, ..., \delta n\}$, a esta lista también se le conoce como la lista $NAIR$. De forma general, pi se denota como

[3]Non-symmetry Anti-packing Image Representation
[4]Non-symmetry Anti-packing pattern representation Model

cuádrupla $(prototipo, ps, vA)$ donde prototipo es un ejemplar de pi, ps son las coordenadas del punto de inicio de pi, este punto es el píxel más alto a la izquierda; v es el color de pi, y A es un conjunto de parámetros de la forma de pi. Sin embargo, la figura que se uso en el artículo original era un rectángulo y solo se consideran imágenes binarias, se recomienda que se ignoren los parámetros v y prototipo, pues sólo usaran prototipos de forma rectangular además de que se utilizaron imágenes binarias en el articulo original. Por lo tanto la cuádrupla solo contiene las coordenadas del rectángulo y sus dimensiones, entonces la cuádrupla queda así (x , y, l, w). Todo los anterior nos ayuda a generar el $NAIR$ de la imagen.
La definición del perímetro que se ha escogido para trabajar en ella es:

$$C = \sum_{i=0}^{4} (i * wi)$$

donde wi es el número de píxeles negros cuyos vecinos $4-$conexos i son píxeles blancos. Para poder calcular el perímetro C de una imagen binaria representada por $NAIR$, recorremos la lista $NAIR$ Q y para cada rectángulo p en Q sumamos el perímetro de p a C y entonces se le resta a C la longitud de los lados compartidos por p y sus rectángulos adyacentes. Sin embargo, cada vez que visitemos un rectángulo nuevo tendremos que realizar la búsqueda de los adyacentes y restar las longitudes de los lados compartidos incluso de los ya visitados, a pesar de que estas búsquedas se pueden hacer eficientes la meta de ambas es la misma que es la de restar la longitudes compartidas de p a C.

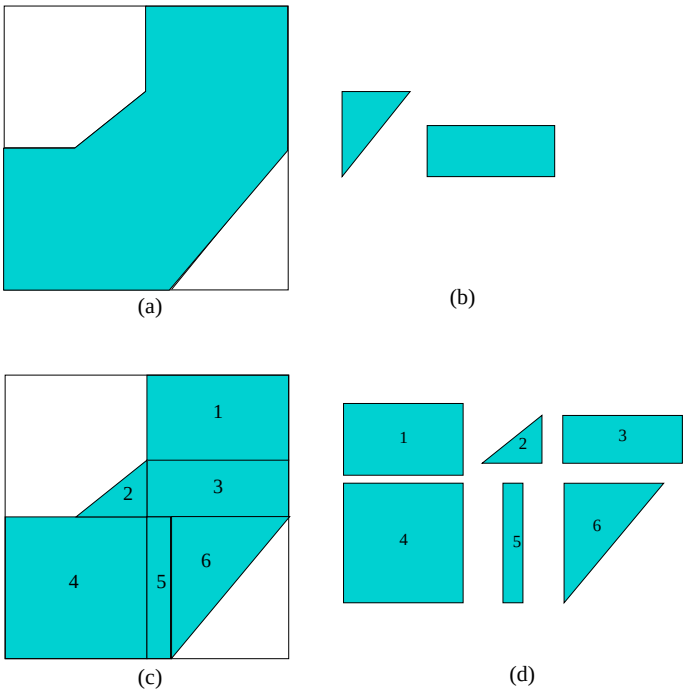

Figura 4.17: Ejemplo de un $NAIR$.

Entonces en lugar de realizar dos búsquedas, lo que se hace es simplemente resta el doble de la longitudes compartidas de los rectángulos a p empezando por el fondo a la derecha y al siguiente paso se reducen las búsquedas del rectángulo adyacente a p. Por lo tanto, el algoritmo recorre la lista $NAIR\ Q$ y va sumando los perímetros de los rectángulos como se indico anteriormente hasta recorrer toda la lista Q. La Figura 4.17 (a) se muestra una imagen digilizada, en (c) esta la division asimetrica de la imagen usando los prototipos predefinidos en (b) y en (d) esta la lista NAIR generada.
Sin embargo, en los $NAIR$ se pierde la relación espacial explícita entre las instancias lo que hace que calcular el perímetro de los rectángulos no sea sencillo. Por lo tanto, es necesario introducir una representación especial que permita restaurar esta relación de la lista $NAIR$ para el cálculo del perímetro, a esta representación se le llama L^2G. La representación L^2G de una imagen $2^n \times 2^n$ consiste de dos arreglos 2^n de listas lineales, denominados S_x y S_y. Para cada $i = 0, 1, ..., 2^n$, $S_x[i]$ es una lista lineal que se usa para almacenar índices de los rectágulos cuyas coordenadas de x del punto de inicio son i. Para poder acelerar la búsqueda los índices son ordenados de tal manera que la coordenada y del punto de inicio de los rectángulos está en orden ascendente. S_y es igual que S_x, excepto que la lista lineal indexa las ordenadas en lugar de las abscisas. Sin L^2G, se tiene que recorrer toda la lista $Q\ NAIR$ para poder encontrar todos los rectágulos adyacentes de p. Sin embargo, con el uso de L^2G, el número de los rectángulos a los que se accede de accesos se reduce notablemente.

Conclusiones

El cálculo del perímetro en curvas y regiones digitales es una tarea que parece sencilla, sin embargo es más complicada de lo que se ve. Los primeros métodos son imprecisos y no mejoran a pesar de que se incremente la resolución de la imagen. Se fueron generando nuevas estrategias para tratar estos problemas y se propusieron nuevos métodos y algoritmos que trataban de ser más aproximados al valor real del perímetro de la imagen digitalizada. Todos estos métodos utilizan como base la topología digital clásica propuesta por el Dr. Azriel Rosenfeld, esta topología presenta problemas, como la paradoja de Jordan y el hecho de que contamos con más de una definición de frontera.

Los métodos propuestos en el trabajo están basados en la topología digital de espacios finitos, esta teoría es totalmente nueva y ajena a la topología digital clásica, por lo que se necesitó explorarla primero antes de comenzar a estudiar los algoritmos. Debido a que la topología digital de espacios finitos resulta ser muy extensa, se acotó su estudio para que sólo se abarcaran las definiciones estrictamente necesarias como la definición de celda y la de función de ligado, que resultan ser muy importantes poder realizar nuestros algoritmos, pues a través de ellos podemos definir la frontera de una imagen que es necesaria para el cálculo del perímetro en los algoritmos MLP y DSS. Esta topología a diferencia de la clásica, no presenta ni ambigüedad ni paradojas, y en cambio se tiene la ventaja de que los complejos AC pueden representarse directamente en la computadora.

Por facilidad se decidió que las imágenes serían binarias, estos es; imágenes en blanco y negro, además de con una sola componente conexa, los algoritmos son igualmente válidos para imágenes de color y con más de una componente. Las imágenes son representadas usando la malla estándar en lugar de la malla combinatoria, pues en general demanda menos espacio, a pesar de que en la malla combinatoria podemos obtener la información de las celdas directamente o podemos deducirla fácilmente.

Tanto el algoritmo DSS como el MLP son relativamente fáciles de entender. Para los dos algoritmos se requiere del cálculo de la frontera y en particular para el MLP se requiere además de la línea de demarcación, esto puede hacer parecer como una desventaja para el algoritmo MLP, sin embargo el cálculo tanto de la frontera como de la línea de demaración se realiza en un tiempo de ejecución lineal. Las implementaciones que se dan en el trabajo de ambos se basan en el pseudocódigo del Dr. Kovalevsky; y, en los dos casos, se tuvieron que realizar algunos pasos extras que no estaban dentro de la descripción del algoritmo como el control de las banderas en el algoritmo MLP en cada ciclo nos indican que insecto se mueve y cuando se tiene una situacion de península o bahía; así como la elección de las direcciones del algortimo DSS que permiten especificar la dirección del

último paso de éste.
Ambos algoritmos propuestos son de tiempo lineal y convergen al valor real conforme se aumenta la resolución de la imagen, a pesar de estos, cada uno de los algoritmos presenta ventajas y desventajas con respecto al otro. El algoritmo DSS es más rápido en ejecución que el MLP, pero el MLP converge más rápido a la frontera de la imagen, y por lo tanto a el perímetro. También está el hecho de que MLP no puede lidiar con imágenes que tengan hoyo de longitud 1 ó 2, mientras que el DSS no tiene problemas con ningún tipo de imagen.
En el Apéndice E se mencionan las estructuras de datos que en particular se utilizan en la topología digital de espacios finitos para facilitar el manejo de las celdas, sin embargo el uso de un arreglo simple junto con la representación de malla estándar de la imagen es suficiente para resolver el problema.

Otros algoritmos para calcular el perímetro basados en la primera aproximación dada por Rosenfeld tienen el gran problema de que la aproximación, en la mayoría de los casos, resulta ser mayor que el original. Otros autores para mejorar este cálculo multiplican este resultado por un estimador de perímetro para poder acercarse más a el perímetro real. A pesar de que se cuenta con otra versión de esta estrategia para el cálculo perímetro que toma puntos esquina y no esquina de igual forma aun se necesita multiplicar el resultado por un factor para mejor la aproximación. Tanto DSS como MLP a diferencia de los métodos anteriores convergen al valor real conforme se aumenta la resolución de la imagen y no requieren de estos factores para mejorar su aproximación.
Otro algoritmo basado en quadtrees se tiene el problema de la simetría, lo que reduce su rendimiento, además de que depende totalmente de la implementación de los quadtrees. Mientras que el último algoritmo es sencillo de entender pero presenta el problema tener que construir una malla especial conocida como L^2G para facilitarse el cálculo del perímetro de las formas básicas en las que subdivide una imagen. En los casos anteriores el algoritmo depende de la implementacion de una estructura de datos a diferencia de los metodos DSS y MLP que cuentan con estructuras de datos que ayudan a la representacion de la informacion topológica, sin embargo no son indispensables para el algoritmo.

Los algoritmos MLP y DSS son muy simples bajo la teoría dada, no necesitan de algún factor para mejorar su aproximación, y si bien podemos implementar sus propias estructuras de datos, no son esenciales ni dependen de la implementación de éstas al momento de calcular el perímetro a diferencia de los algoritmos basados en topología digital clásica u otras estrategias.

Apéndice A

Topología

Las definiciones y conceptos de esta sección fueron tomados del libro *Topología Básica* de C. Prieto.

A.1. Espacios Euclidianos

Con el fin de familiarizarnos con la notación del texto, en esta sección presentaremos una serie de ejemplos de espacios topológicos canónicos que jueguen un papel muy importante, tanto en la topología como en otras ramas de las matemáticas. R y C designarán, como es costumbre, los números reales y los complejos. Si n es mayor o igual que 1, R^n será el **espacio euclidiano** de dimensión n, es decir, $R^n = \{x = (x_1, ..., x_n) \mid x_i \in R, i = 1, ..., n\}$, con sus operaciones usuales: si $x, y \in R^n$ y $r \in R$, entonces, $x + y = (x_1 + y_1, ..., x_n + y_n)$, $x - y = (x_1 - y_1, ..., x_n - y_n)$ y $r \cdot x = (r \cdot x_1, ..., r \cdot x_n)$ y la **norma** está dada por $|x| = \sqrt{x_1^2 + ... + x_n^2}$. Se define **la distancia entre dos puntos** simplemente como $|y - x|$. R^0 representa al espacio euclidiano consistente de un solo punto, el 0, que se puede ver como subespacio del espacio R^n. Se tiene una identidad canónica $R^n \times R^m = R^{n+m}$ dada por $((x_1, ..., x_n), (y_1, .., y_m) = (x_1, ..., x_n, y_1, ...y_m))$. Así mismo, se considera a R^n de forma canónica como un subespacio de R^{n+1} identificándolo con el subespacio $R^n \times 0$. También se tiene una identificación canónica de R^2 con los números complejos C haciendo $(x + y) = x + i \cdot y$, donde i es la raíz cuadrada de -1.

Definición A.1 Sea $n \geq 0$. Consideraremos los siguiente espacios:

$R^+ = \{ x \in R \mid 0 \leq x \}$, la **semirecta no negativa**.

$B^n = \{ x \in R^n \mid |x| \leq 1 \}$, la **bola unitaria** de dimensión n.

$S^{n-1} = \{ x \in R^n \mid |x| = 1\}$, la **esfera unitaria** de dimensión $n - 1$.

Si $n = 2$, entonces S^1 es el **círculo unitario**.

$B^{\circ n} = \{ x \in R^n \mid |x| < 1\}$, la **célula unitaria** o **bola unitaria abierta** de dimensión n.

$I^n = \{ x \in R^n \mid 0 \leq x_i \leq 1, 1 \leq y \leq n\}$, el **cubo unitario** de dimensión n.

$\delta I^n = \{ x \in I^n \mid x_i = 0 \text{ o } 1 \text{ para alguna } i\}$, la **frontera** de I^n en R^n.

$I = I^1 = [0,1] \subset R$, el **intervalo unitatio**.

Brevemente, a B^n se le conoce como la $n-$bola unitaria, a S^{n-1} como la $n-1$ esfera unitaria, a $B^{\circ n}$ como la $n-$célula unitaria o $n-$bola unitaria abierta, y a I^n como el $n-$cubo unitario.

A.2. Espacios Métricos

En R^n con la métrica o distancia usual, definida en A.1, se define el concepto de *conjunto abierto*. Este concepto de métrica y su concepto asociado de conjunto abierto puede generalizarse a cualquier conjunto provisto de una función que se comporte de manera análoga a la función distancia en R^n. Esto permite estudiar una serie de propiedades de los conjuntos provistos de una métrica, que serán comunes a las de los espacios euclidianos y a las de cualquiera de sus subespacios.

Definición A.2 Un **espacio métrico** consta de un conjunto X y de una función $d : X \times X \to R^+$, llamada **métrica**, que satisface los siguientes axiomas:

$M1$: $d(x,y) = 0 \Leftrightarrow x = y$

$M2$: $d(x,y) = d(y,x) \forall x, y \in X$

$M3$: $d(x,y) \leq d(x,z) + d(y,z), \forall x, y, z \in X$

A esta ultima desiguladad se le conoce como la *desigualdad del triángulo*.

Definición A.3 Sea X un espacio vectorial real. Una **norma** en X se define como una función que a cada $x \in X$ asocia un número real $\| x \| \in R^+$, la **norma** de x, tal que

$No1$: $\| x \| = 0 \Leftrightarrow x = 0$.

$No2$: $\| rx \| = |r| \cdot \| x \|, r \in R$.

$No3$: $\| x - y \| \leq \| x \| + \| y \|$

Un espacio vectorial X provisto de una norma es llamado **espacio vectorial normado**.

A.3. Vecindades y conjuntos abiertos.

En un espacio métrico es posible hablar de vecindades con un único punto.

Definición A.4 Sea $x \in X$, $\epsilon \in R$, $\epsilon > 0$. Definimos a la **bola abierta de radio** ϵ con centro en x como:

$$B_\epsilon(x) = \{y \in X \mid d(x, y) < \epsilon\}$$

Definimos una **vecindad** de x como un conjunto $U \subseteq X$, tal que existe $\epsilon > 0$ y se cumple $B_\epsilon(x) \subseteq U$.

Definición A.5 Dadas dos métricas d y $d\prime$ en un conjunto X, decimos que son **equivalentes** si para cada punto $x \in X$ ambas determinan las mismas vecindades de x.

Proposición A.1 Dos métricas d y $d\prime$ en X son equivalentes $\Leftrightarrow$ para cada punto $x \in X$ se cumple lo siguiente: Dada $\epsilon > 0$ existe $\epsilon\prime > 0$ tal que $d\prime(x, y) < \epsilon\prime$ implica $d(x, y) < \epsilon$; y dada $\delta\prime > 0$ existe $\delta > 0$ tal que $d(x, y) < \delta$ implica $d\prime(x, y) < \delta\prime$.

Definición A.6 Declaremos a un subconjunto A de un espacio métrico X como **abierto** si A es vecindad de x, $\forall x \in A$. En otras palabras, $A \subseteq X$ es abierto $\Leftrightarrow \forall, x \in A$ existe $\epsilon > 0$ tal que $B_\epsilon(x) \subseteq A$.

Proposición A.2 Para un espacio métrico X se cumplen las siguientes afirmaciones:

$A1$: Si $\{A_i\}_{i \in I}$ es una familia de abiertos en X, para cualquier conjunto de índices I, entonces $\cup_{i \in I} A_i$ es abierto en X.

$A2$: Si $\{A_i\}_{i \in I}$ es una familia $finita$ (I finito) de abiertos en X, entonces $\cup_{i \in I} A_i$ es abierto en X.

A.4. Espacios Topológicos

Definición A.7 Sea X un conjunto. Una **topología** en X es una familia A de subconjuntos en X, llamados **abiertos**, tal que cumple con los siguientes axiomas:

$A1$: Si $\{A_i\}_{i \in I} \subseteq A$, I arbitrario, entonces $\cup_{i \in I} A_i \in A$.

$A2$: Si $\{A1\}_{i \in I} \subseteq A$, I finito, entonces $\cap_{i \in I} A_i \in A$.

En particular, el conjunto completo X, por ser la intersección de una familia vacía, y el conjunto $\emptyset$, por ser la unión de una familia vacía, están en A. A la pareja **(X,A)** se le denomina **espacio topológico**, al cual denotaremos sólo por X cuando no haya confusión con respecto a la estructura topológica dada por sus abiertos. Al conjunto X lo llamaremos el **conjunto subyacente** del espacio topológico.

Definición A.8 Sea X un espacio topológico y sea $X \in X$. Definimos una **vecindad** de x como un conjunto $U \subseteq X$ para el que existe un abierto A en X, tal que $x \in A \subseteq U$.

Teorema A.1 Sea X un espacio topológico. $A \subseteq X$ es abierto $\Leftrightarrow$ A es vecindad de todos sus puntos.

Al conjunto de vecindades de un punto x en un espacio topológico X como N_x^X o simplemente N_x. A $N=\{N_x | x \in X\}$ lo llamaremos **sistema de vecindades** de la topología de X.

Proposición A.3 El sistema de vecindades N de la topología X satisface las siguientes condiciones:

$V1$: $V \in N_x$, $V \subseteq U \Rightarrow U \in N_x$.

$V2$: $V_i \in N_x$, $i \in I$, I finito, $\Rightarrow \cap_{i \in I} V_i \in N_x$.

$V3$: $V \in N_x \Rightarrow x \in V$.

$V4$: $U \in N_x \Rightarrow \exists\, V \in N_x$, tal que $U \in N_y\ \forall y \in V$.

Definición A.9 Sea X un conjunto y, para cada $x \in X$, sea N_x una familia que satisface las condiciones de la proposición anterior. A la colección $N=\{N_x \text{ — } x \in X\}$ la llamaremos un **sistema de vecindades** en X.

Un sistema de vecindades en un conjunto X determina una topología en X en el siguiente sentido.

Teorema A.2 Sean X un conjunto y N un sistema de vecindades en él. Entonces existe una única topología en X que tiene precisamente a N como el sistema de vecindades de ella.

Definición A.10 Sea X un espacio topológico y sea $A \subseteq X$. Definimos el **interior** de A como

$$A^\circ = \{x \in A | A \in N_x\}$$

A un punto x $\in A^\circ$ se le llama **punto interior**.

A.5. Conjuntos cerrados

Los conjuntos que llamaremos cerrados en un espacio topológico X, al igual que los abiertos, determinarán su topología. Sin embargo, sus propiedades son muy diferentes a las de los abiertos, por lo que resulta importante estudiarlos por separado.

Definición A.11 Un punto x en un espacio topológico X es un **punto de contacto** de un conjunto A contenido en X si toda vecindad V de x en X es tal que la intersección V con A es no vacía. Sea

$$\bar{A} = \{x \in X | \text{ x es punto de contacto de } A\}$$

A $\bar{A}$ se le llama la **cerradura** o **envolvente cerrada** de A. Ahora veamos algunas propiedades de la cerradura.

Proposición A.4 La cerradura satisface las siguientes condiciones:

(a) $\bar{A} = X \setminus (X \setminus A)^\circ$.

(b) $X \setminus \bar{A} = (X \setminus A)^\circ$.

Así, si $X \setminus A$ es abierto entonces

$$X \setminus A = (X \setminus A)^\circ = X \setminus \bar{A}$$

y, por lo tanto, $\bar{A} = A$.

Definición A.12 Un conjunto A en un espacio topológico X se dice que es **cerrado** si $X \setminus A$ es abierto.

Podemos ver que A es cerrado si y sólo si $A = \bar{A}$. En particular, tenemos que X y $\emptyset$ son abierto y cerrado a la vez. En un espacio discreto, todos los conjuntos son cerrados y abiertos, mientras que en un espacio indiscreto, los únicos conjuntos cerrados son X y $\emptyset$.

Definición A.13 Sea X un espacio topológico y sea $A \subset X$. Un punto $x \in \bar{A} \cap X \setminus A$ se denomina **punto frontera** de A (y de $X \setminus A$); a $\bar{A} \cap X \setminus A$ se le llama la **frontera** de A (y de $X \setminus A$) y se le denota por δA.

Se puede observar que la frontera de un conjunto A es cerrada. Además, la frontera de un conjunto A no está contenida en A.

Definición A.14 Sean X un espacio topológico y A un subconjunto de X. Un punto $x \in A$ es **aislado** si tiene un vecindad U en X, tal que $U \cap A = \{x\}$.

En particular, un punto aislado de un espacio topológico X es un punto que, como conjunto es abierto en X.

Definición A.15 Sean X un espacio topológico y A un subconjunto de X. Un punto $x \in X$ se denomina **punto de acumulación** de A si toda vecindad V de x en X es tal que $(V \cap A) - \{x\} = \emptyset$.

Proposición A.5 Dado un punto x en A, x es aislado o x es de acumulación de A

A.6. Homeomorfismo

Un espacio topológico es un conjunto con una estructura adicional dada por sus conjuntos abiertos (o, por sus conjuntos cerrados). Se puede hablar de que los espacios topológicos son "isomorfos" cuando existe una biyección entre sus conjuntos subyacentes, que respeta la "estructura topológica".

Definición A.16 Sean X y Y espacios topológicos y sea $f : X \to Y$. Decimos que f es un **homeomorfismo** si es una función continua y biyectiva y su inversa $f^{-1} : Y \to X$ también es una función continua. Si existe tal homeomorfismo se dice que lo espacios X y Y son **homeomorfos**, y se denota como $f : X \to^{\approx} Y$ o como $X \cong Y$.

Un homeomorfismo es una biyección que respeta la estructura topológica.

Teorema A.3 Sean X y Y espacios topológicos y $f : X \to Y$. Son equivalentes:

(a) f es un homeomorfismo.

(b) f es biyectiva y $f(A)$ es abierto en Y si y sólo si A es abierto en X.

(c) f es biyectiva y $f(A)$ es cerrado en Y si y sólo si A es cerrado en X.

El teorema anterior afirma que f no sólo establece una biyección entre los puntos de X y los puntos de Y, sino que también entre los abiertos de uno y del otro, esto también se afirma para los cerrados. Un homeomorfismo establece una equivalencia, no sólo de los conjuntos sino también de sus estructuras topológicas.

Nótese que si f es continua y biyectiva, no necesariamente es un homeomorfismo. Por ejemplo, si X es un espacio discreto con más de un elemento y Y es indiscreto, con el mismo conjunto subyacente que X, entonces la identidad es continua y biyectiva, pero no es un homeomorfismo. La relación de homeomorfismo es una relación de equivalencia.

Apéndice B

Topología Digital Clásica

En el proceso de imágenes y cómputo gráfico, un objeto en el plano o en el 3−espacio se aproxima digitalmente por un conjunto de pixeles o voxeles. La topología digital estudia las propiedades de este conjunto de pixeles o voxeles que corresponde a las propiedades topológicas del objeto original. Este material esta basado en el libro *Topological Algorithms for digital Image Processing* del Dr. Azriel Rosenfeld.

B.1. Conceptos Básicos

Las imágenes que se consideran en la topología digital son arreglos binarios, cuyos elementos tienen valor de 0 ó 1. Se puede generalizar la topología digital para que se puedan aceptar arreglos de imágenes de escalas de grises cuyos elementos tiene valores de $0 \leq x \leq 1$.
A los elementos de un arreglo de una imagen 2−dimensional se les llama **pixele**s; para una imagen 3−dimensional se les llama **voxeles**.

Para evitar considerar la frontera del arreglo de la imagen asumimos que el arreglo no tiene límites en todas las direcciones. Esto permite imágenes en las cuales un número infinito de pixeles o voxeles tiene valor 1. Sin embargo, las imágenes son frecuentemente definidas de manera que se evita esta posibilidad.

Asociamos a cada pixel o voxel con un **punto latice**, esto es, un punto con coordenadas enteras en el plano o en 3−espacio. Se escribe Z^2 para el conjunto de puntos latice en el plano y Z^3 para el conjunto de puntos latice en el 3−espacio.

Dos puntos latice de Z^2 se dicen que son 8−**adyacente** si son distintos y cada coordenada de uno difiere de la correspondiente coordenada del otro por a lo más 1; dos puntos latice son 4−**adyacente** si son 8−adyacente y difieren en solo una de sus coordenadas. En Z^3 dos puntos latice son 26−adyacentes si son distintos y cada coordenada de uno difiere de la correspondiente coordenada del otro por a lo más 1, 18−adyacente si son 26−adyacente y difieren en a los más dos de sus coordenadas y 6−adyecentes si son 26−adyecentes y difieren en sólo una coordenada. Para $n = 4, 8, 6, 18, 26$ un n−**vecino** de un punto latice p es un punto q que es n−adyacente a p.

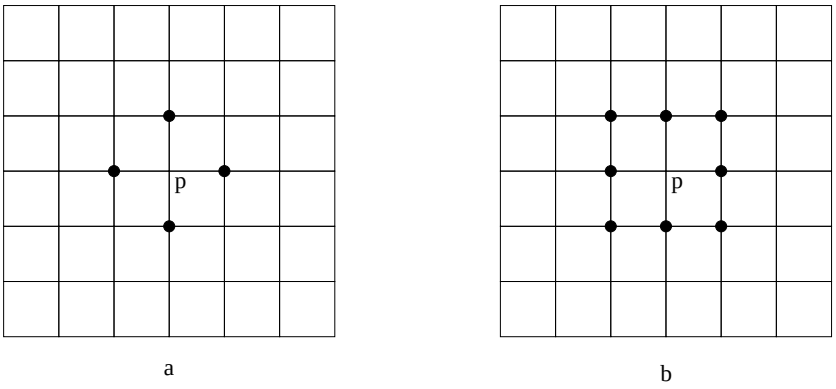

Figura B.1: El la figura (a) se muestra los 4−vecinos de p y en la figura (b) su muestran los 8−vecinos.

Si p es un punto latice en Z^2 entonces $N(p)$ denota el conjunto que consiste de p y sus 8−**vecinos**. Si p es un punto latice en Z^3 entonces $N(p)$ denota el conjunto que consiste de p y sus 26−**vecinos**. Un punto latice asociado con un pixel o voxel que tiene valor 1 en un imagen se le llama **punto negro**; un punto latice asociado con un pixel o voxel que tiene valor 0 se le llama **punto blanco**.

B.2. Imagen Digital

Uno de los puntos de estudio de la topología digital fue la idea de usar diferentes relaciónes de adyacencia para los puntos negros y blancos. La razón para esto es la de evitar paradojas, por ejemplo si una 4−adyacencia es usada para todos los pares de puntos de la Figura B.2, entonces todos los puntos negros están totalmente desconectados, pero igual separan el punto central blanco de los otros puntos blancos. En cambio, si se utiliza una 8−adyacencia para todos los pares de puntos, entonces los puntos negros forman el análogo discreto de una curva de Jordan pero no separan los puntos blancos.

Estas dificultades se resuelven si usamos 4−adyacencia para los puntos blancos y 8−adyacencia para los negros o viceversa. En tres dimensiones, paradojas análogas son evitadas de manera similar si la 6−adyacencia es usada para los puntos blancos y la 18−adyacencia o la 26−adyacencia para los puntos negros o viceversa.

Definición B.1 Una **imagen digital** es una cuadrupla (V, m, n, B), donde $V = Z^2$ o $V = Z^3$, $B \subseteq V$, y donde $(m, n) = (4, 8)$ o $(8, 4)$ si $V = Z^2$, y $(m, n) = (6, 26)$, $(26, 6)$, $(6, 18)$ o $(18, 6)$ si $V = Z^3$.

Una imagen digital $P = (V, m, n, B)$ se dice que es 2−dimensional o 3−dimensional según sea $V = Z^2$ o $V = Z^3$, y también se le llama una (m, n)−imagen digital. Los elementos de V son llamados puntos de (o en) P. Los puntos de B son llamados **puntos negros** de P; los puntos en $V \setminus B$ son llamados los **puntos blancos** de P. Usualmente

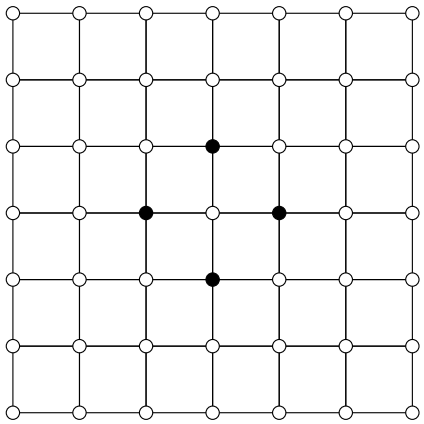

Figura B.2: Paradoja de la conexidad.

B es un conjunto finito; si es así, entonces P se dice que es finita.

Dos puntos negros de P se dicen que son $P-$adyacentes si son $m-$adyacentes, y dos puntos blancos o un punto blanco y un blanco se dice que son $P-$adyacentes si son $n-$adyacentes. Consideremos a $\bullet$ como P, o cualquier de los números enteros 4,8,18 o 26. Si un punto p es $\bullet-$adyacente a un punto q entonces decimos que p es un $\bullet$–**vecino** de q. Un punto p se dice que es $\bullet-$adyacente a un conjunto de puntos S si p es $\bullet-$adyacente a algún punto en S. Dos conjuntos de puntos S y T se dice que son $\bullet-$adyacentes si algún punto en S es $\bullet-$adyacente a algún punto en T. Un punto negro se dice que es $\bullet$–**aislado** si no es adyacente a otro punto negro.

Un conjunto de puntos es $\bullet$–**conexo** si no es la unión de dos conjuntos disjuntos no vacíos que no son $\bullet-$adyacentes. Una $\bullet$–**componente** de un conjunto de puntos no vacío S es un subconjunto maximal $\bullet-$conexo de S. Entonces una $\bullet-$componente de S es un subconjunto no vacío $\bullet-$conexo de S que no es $\bullet-$adyacente a otro punto en S.

Para $P = (V, m, n, B)$, una $P-$componente de B es una **componente negra** de P. Una $P-$componente de $V \setminus B$ es una **componente blanca** de P. Una componente blanca finita se le llama un **hoyo** de P en 2D y una **cavidad** de P en 3D. Si P es una imagen digital entonces tiene una única componente blanca infinita, esta componente se le llama el **fondo** de P, sin embargo, este término se refiere al conjunto de todos los puntos blancos. Al conjunto de todos los puntos negros se le llama el **primer plano** de P.

Una $\bullet$–**trayectoria** es una secuencia $\langle p_1, p_2, ..., p_k \rangle$ de puntos con $k \leq 1$ en la cual cada punto p_i es $\bullet-$adyacente a p_{i+1} $1 \leq i < k$. Si $p_1{=}p_k$ entonces la $\bullet-$trayectoria es **cerrada**. La **trayectoria de un solo punto** $\langle p_0 \rangle$ es un caso especial de una $\bullet-$trayectoria cerrada. Una $\bullet-$trayectoria desde p a q es una $\bullet-$trayectoria cuyos puntos inicial y final son respectivamente p y q. Dos puntos p y q que están en la misma $\bullet-$componente de un conjunto de puntos S si y sólo si existe una $\bullet-$trayectoria en S de p a q.

Una $\bullet$–**curva simple cerrada** es un conjunto finito de puntos $\bullet-$conexo en el cual cada punto es $\bullet-$adyacente a exactamente otros dos puntos en el conjunto. Un $\bullet$–**arco** es un conjunto finito de puntos $\bullet-$conexo en el cual hay dos puntos, llamados finales, cada uno de los cuales es $\bullet-$adyacentes a sólo un punto en el conjunto y en el cual cada punto

que no es un punto final es $\bullet-$adyacente a exactamente dos puntos en el conjunto.

Un punto negro en una imagen digital P se le llama un **punto borde** si es adyacente a uno o más puntos blancos; de otra manera se le llama **punto interior**. El **borde** de una componente negra C de P es el conjunto de todos los puntos bordes en C. El **interior** de una componente negra C de P es el conjunto de todos los puntos interiores en C. El borde de una componente negra C con repecto a una componente blanca D es el conjunto de puntos en C que son adyacentes a D.

La frontera entre dos conjuntos disjuntos P y Q se define como el conjunto de todos los pares (p, q) tales que $p \in P$, $q \in Q$ y p es $4-$adyacente o $6-$adyacente a q. Un conjunto conexo de puntos X en P se dice que **rodea** a un conjunto de puntos Y en P si cada punto en Y está contenido en una $P-$componente finita de $V \setminus X$. En una imagen digital finita la componente blanca única e infinita rodea a todos los otros puntos.

Una componente blanca de P la cual es adyacente y es rodeada por una componente negra C de P se le llama **hoyo** de C si P es una imagen digital 2D y una cavidad de C si P es una imagen digital 3D. En el caso 2D, si una componente negra C no tiene hoyos, entonces decimos que C es una **componente simplemente conexa**.

Apéndice C

Definiciones necesarias para las estructuras de datos

A continuación se presentan las definiciones necesarias para poder establecer la teoría y necesaria para las estructuras de datos lista de celdas 2−dimensional y 3−dimensional, que encontramos en el Apendice E. Este material está basado en diferentes definiciones que se pueden encontrar en el libro *Geometry of Locally Finite Spaces* del Dr. Kovalevsky. Además de las definiciones de la vecindad abierta $SON(c, M)$ y la cerradura $Cl(c, M)$ necesitaremos las siguientes definiciones:

$$SON^*(c, M) = SON(c, M) - \{c\}$$

y

$$Cl^*(c, M) = Cl(c, M) - \{c\}$$

Definición C.1 (Bola AC cerrada 0−dimensional)
Una **bola AC cerrada** 0−**dimensional** (una 0−bola) es un k−complejo, referencia, que consiste de una sola celda y se denota por B^0. El k−complejo que resulta de la unión de dos distintas 0−bolas se llama una esfera 0−dimensional y se denota por S^0.

Definición C.2 (Esfera AC 1−dimensional)
Una **esfera AC** 1−**dimensional** es un k−complejo conexo 1−dimensional en el cual cada celda es incidente exactamente a otras dos celdas y se denota por S^1.

Definición C.3 (Celda Propia) Una celda m−dimensional c^m de un complejo n−dimensional C^n, $n \leq m$, se llama una **celda propia** de C^n si el complejo $Cl^*(c^m, C^n)$ es vacío (si $m = 0$) o es una esfera $(m-1)$−dimensional S^{m-1} y el complejo $SON^*(c^m, C^n)$ es vacío (si $m = n$) o es una esfera $(n-m-1)$−dimensional S^{n-m-1}.

Definición C.4 (Complejo propio)
Un complejo C es **propio** si todas sus celdas son celdas propias de C.

Definición C.5 (Bola AC cerrada $m-$dimensional)
Una **bola AC cerrada $m-$dimensional**, $0 < m \leq n$, denotada por B^m, es la cerradura de una celda propia $m-$dimensional de un complejo $n-$dimensional o es la unión de dos bolas cerradas $m-$dimensional B_1^m y B_2^m, mientras $\delta B_1^m \cap \delta B_2^m$ es una bola cerrada $(m-1)-$dimensional.

Definición C.6 (Esfera AC $m-$dimensional)
Una **esfera AC $m-$dimensional**, $m \leq 2$, denotada por S^m, es la unión de dos bolas cerradas $m-$dimensional B_1^m y B_2^m intersectándose sólo en sus bordes completos:

$$S^m = B_1^m \cup B_2^m$$

mientras

$$\delta B_1^m = \delta B_2^m$$

y

$$(B_1^m - \delta B_1^m) \cap (B_1^m - \delta B_1^m) = \varnothing$$

Definición C.7 (Bola abierta $m-$dimensional)
Una **bola abierta $m-$dimensional** O^m es el subcomplejo de un complejo $n-$dimensional que es n-dual a una bola cerrada $m-$dimensional.

Definición C.8 (Estructura de incidencia)
La **estructura de incidencia** de una celda $c \in C^n$ es el complejo

$$SON^*(c, C^n) \cup Cl^*(c, C^n)$$

Apéndice D

Orientación de los complejos AC

Los siguientes conceptos son necesarios para poder definir las esctructuras de datos lista de celdas 2−dimensional y 3−dimensional, que se encuentran en el Apendice E. El material está basado en el libro *Geometry of Locally Finite Spaces* del Dr.Kovalevsky.

La noción de orientación se definió originalmente para los complejos Euclidianos. Esta noción de orientación fue adaptada para los complejos simples. Sin embargo, ninguna de estas aproximaciones se puede aplicar directamente a los complejos de celdas abstractas. Por lo tanto, la noción de orientación permanece indefinida para los complejos abstractos en general. Sin embargo, podemos usar una orientación para los complejos propios. En el caso de un subcomplejo 2−dimensional cerrado es natural el definir la orientación de una 1−celda, especificando cual de las dos 0−celdas incidentes a ella es su punto inicial y su punto final. La orientación de un 2−celda se define como la opuesta en el sentido de circulación de las 1−celdas orientadas en su borde o frontera.

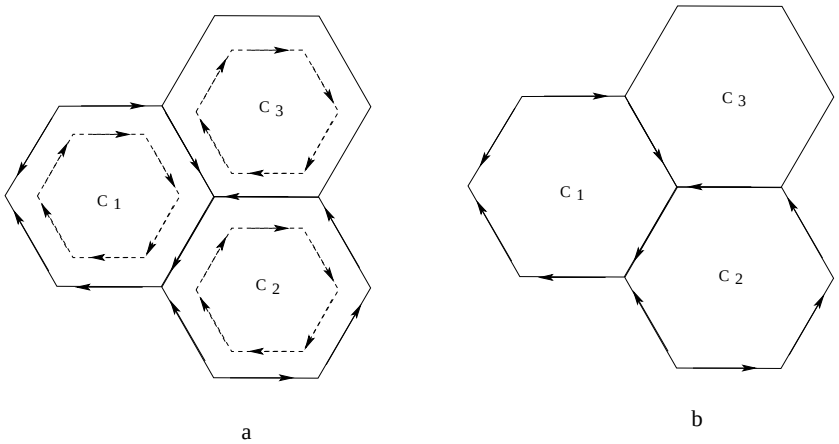

Figura D.1: Mostrando la necesidad de la orientación inducida.

Podemos observar en la Figura D.1 que nos es imposible el definir la orientación para todas las 1−celdas de tal manera que los sentidos de la circulación de las tres 2−celdas estén de acuerdo con las 1−celdas. Para poder solucionar esto se introduce, junto con la orientación intrínseca de las 1−celdas, las orientaciones relativas a una 2−celda espécifica.

Se les conoce como las orientaciones inducidas de las 1−celdas en una 2−celda. Una orientación inducida de una 1−celda es igual a su orientación intrínseca o la orientación opuesta a la intrínseca. Entonces la orientación de una 2−celda puede estar definida en el sentido de circulación de las orientaciones inducidas.

Una 0−celda no tiene orietación intrínseca, sin embargo tiene orientacioń inducida relativa a cada 1−celda incidente a ella. Esto significa que si la orientación de una 1−celda se cambia, entonces su punto inicial se convierte en su punto final y viceversa. Además una misma 0−celda puede pude ser el punto inicial de una 1−celda y el punto final de otra. Entonces esta 0−celda tiene diferentes orientaciones inducidas en estas dos 1−celdas.

El operador que transforma una orientación intrínseca de una celda a a su orientación inducida en una celda b sólo puede realizar una de dos operaciones: Puede hacer la orientación inducida de a igual a su orientación intrínseca o igual a la opuesta. La elección entre estas dos posibilidades se puede hacer por medio de un parámetro que toma dos valores diferentes. Es común usar emplear el valor de $+1$ o -1, el valor -1 significa *cambio* y el $+1$ significa *sin cambio*. Este parámetro se llama *valor de ligado de la celda a en la celda b* y se denota por $\epsilon(a, b)$. Si b es una 1−celda y a es una 0−celda incidente a b, entonces $\epsilon(a, b) = 1$ si a es el punto inicial de b y $\epsilon(a, b) =$ -1 de lo contrario. La elección para cada 1−celda del complejo puede hacerse arbitrariamente.

Para asignar una orientación a una 2−celda c^2 es necesario encontrar todos los valores de todas las 1−celdas en el borde de c^2 tal que la orientación inducida de cualesquiera dos 1−celdas c_1^1 y c_2^1 que tiene a la 0−celda c^0 incidente a ellas sean *coherentes*, lo cual significa que esta 0−celda debe ser el punto final de una de las 1−celda y el punto inicial de la otra. La siguiente figura nos muestra los casos posibles.

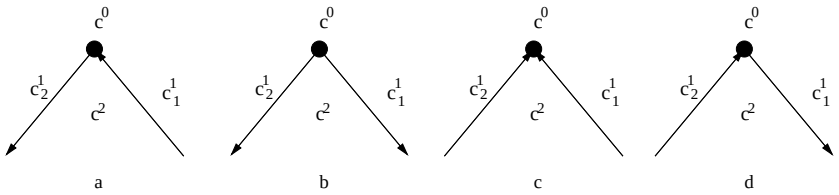

Figura D.2: Ilustrando las elecciones de los valores de ligado.

Si $\epsilon(c^0, c_1^1) = -1$ y $\epsilon(c^0, c_2^1) = +1$, como se muestra en la Figura D.2 (a), entonces las orientaciones intrísecas de c_1^1 y c_2^1 son coherentes y ambos valores $\epsilon(c_1^1, c^2)$ y $\epsilon(c_2^1, c^0)$ se pueden establecer a $+1$, esto significa que la orientación inducida de c_1^1 y de c_2^1 es igual a sus orientaciones intrínsecas. El otro posible caso es establecer ambos valores de $\epsilon(c^0, c_1^1)$ y $\epsilon(c^0, c_2^1)$ a -1. En este caso ambas orientaciones inducidas se convierten en la opuesta a sus orientaciones intrínsecas y permanecen coherentes. De manera similar ocurre en la Figura D.2 (d).

En los casos para las Figuras D.2 (b, c), los valores de $\epsilon(c^0, c_1^1)$ y $\epsilon(c^0, c_2^1)$ son iguales el uno al otro. Entonces los valores de $\epsilon(c_1^1, c^2)$ y $\epsilon(c_2^1, c^2)$ deben ser diferentes, ya sea uno igual a $+1$ y el otro a -1 o viceversa.

La regla general nos dice que: Exactamente un par de $\epsilon(c^0, c_1^2)$, $\epsilon(c^0, c_2^2)$ y $\epsilon(c_1^2, c^2)$,

$\epsilon(c_2^2, c^2)$ debe tener valores distintos. Es posible expresar esta regla como:

$$\epsilon(c^0, c_1^1) * \epsilon(c_1^1, c^2) + \epsilon(c^0, c_2^1) * \epsilon(c_1^2, c^2) = 0$$

La definición de la orientación en el caso general de una celda de dimensión mayor a 1 se puede presentar por las siguientes definiciones formales, las cuales están basadas en el siguiente teorema:

Teorema D.1 (dos $(m-1)-$dimensional) Sea c^n un propio o semi-propio $n-$dimensional de C^n. Sea c^m una celda $m-$dimensional de C^n y c^{m-2} una celda $(m-2)-$dimensional que liga a c^m. Entonces hay en C^n exactamente dos celdas $(m-1)-$dimensional incidentes a ambas c^m y c^{m-2}.

Todas las nociones que se tienen para un $2-$celda se pueden generalizar para el caso de una $m-$celda de un complejo propio o semi-propio $n-$dimensional con $2 \leq m \leq n$.

Definición D.1 Un mapeo del conjunto de $(m-1)-$celdas que ligan a una $m-$celda c^m en el conjunto $\{+1, -1\}$ se llama **función de ligado** de la $m-$celda c^m. El valor de la función de ligado para la $(m-1)-$celda c^{m-1} se llama el **valor de ligado** de c^{m-1} en c^m y se denota por $\epsilon(c^{m-1}, c^m)$.

Por lo tanto, la función de ligado de la celda c^m es el conjunto de pares (c_i, s_i) donde c_i, con $i = 1, 2, ..., k$, son celdas de δc^m y s_i son los enteros iguales a $+1$ o -1. La función de ligado la podemos denotar de manera simbólica como $(\overset{+}{-}c_1, \overset{+}{-}c_2, ..., \overset{+}{-}c_k)$, donde el signo $+$ o $-$ representa el signo de s_i. El valor de s_i se llama la *orientación inducida de* c_i *en* c^m. El orden de los pares (c_i, s_i) y de los símbolos $\overset{+}{-}c_i$ en la función de ligado de una celda no tiene importancia, como en los complejo simplistas, por la tanto podemos aplicar este concepto a complejos no simplistas.

Una $0-$celda no tiene función de ligado. La función de ligado de una $1-$celda puede ser $(+c_1, -c_2)$ o $(-c_1, +c_2)$. Las funciones de ligado de celdas de mayor dimensión deben satisfacer la regla que se dio para $2-$celdas y que ahora podemos generalizar para cualquier $m-$celda.

Definición D.2 Una función de ligado de una $m-$celda c^m se llama **coherente por pares** si cualquier $(m-2)-$celda c^{m-2} que liga a c^m cumple con:

$$\epsilon(c^{m-2}, c_1^{m-1}) * \epsilon(c_1^{m-1}, c^m) + \epsilon(c^{m-2}, c_2^{m-1}) * \epsilon(c_2^{m-1}, c^m) = 0$$

donde c_1^{m-1} y c_2^{m-1} son dos $(m-1)-$celdas incidentes a c^m y c^{m-2}.

Una función de ligado coherente por pares de la celda c^m define de manera única su orientación relativa a todas las celdas que ligan a c^m.

Teorema D.2 (2 orientaciones) Si las funciones de ligado para todas las $(m-1)-$celdas que ligan a la $m-$celda c^m están definidas, entonces la celda c^m puede tener a lo más dos diferentes orientaciones, esto es, que a lo más pude tener dos funciones de ligado coherente por pares.

Definición D.3 Dos celdas a^m y b^m que tiene en común una $(m-1)-$celda incidente z^{m-1} se llaman **coherentemente orientada** a lo largo de z^{m-1} si las orientaciones inducidas de z^{m-1} en a^m y en b^m son opuestas la una a la otra.

Definición D.4 Un complejo $n-$dimensional K^n se llama **orientable**, si las orientaciones de todas las celdas principales existen tal que cada par de celdas principales fuertemente conexas de K^n está coherentemente orientada lo largo de su $(m-1)-$celda común.

Apéndice E

Estructuras de Datos

En esta sección se presentan algunas estructuras de datos utiles las cuales permiten poder acceder de manera más fácil a las propiedades topológicas de las imágenes. El material que se presenta a continuacion está basado en el libro *Geometry of Locally Finite Spaces* del Dr. Kovalevsky.

E.1. Malla Combinatoria

Para representar una imagen como un complejo cartesiano empleamos la **Malla Combinatoria**. Ésta es nuevamente un arreglo 2−dimensional o 3−dimensional, en el cual un elemento de la memoria es asignado a cada celda de los índices del arreglo. Cada eje coordenado es un complejo 1−dimensional. Las 0−celdas de los ejes tienen coordenadas pares, las 1−celdas tienen coordenadas impares. La dimensión y la orientación (si está definida) de cualquier celda puede ser calculada a partir de sus coordenadas combinatorias. La dimensión de cualquier celda es el número de sus coordenadas impares, la orientación es especificada indicando cual de las coordenadas son impares.

En la Figura E.1b, la celda c tiene 1 coordenada impar y es la coordenada X. Por lo tanto, es una celda 1−dimensional ortogonal al eje X. La 2−celda f tiene dos y la 0−celda p no tiene coordenadas impares. En los complejo 3−dimensional la orientación de las 2−celdas está dada de manera similar: si la i−ésima coordenada de una 2−celda C, es la única, entonces la normal de C es paralela al i−ésimo eje coordenado.

Las ventajas de la malla combinatoria palidecen por el hecho de que esta estructura demanda 2^n más espacio en memoria. También el tiempo necesario para el proceso es mucho mayor, por ello es razonable sólo usar la malla combinatoria para imágenes no muy grandes o para propósitos de investigación donde el tiempo de proceso no es importante.

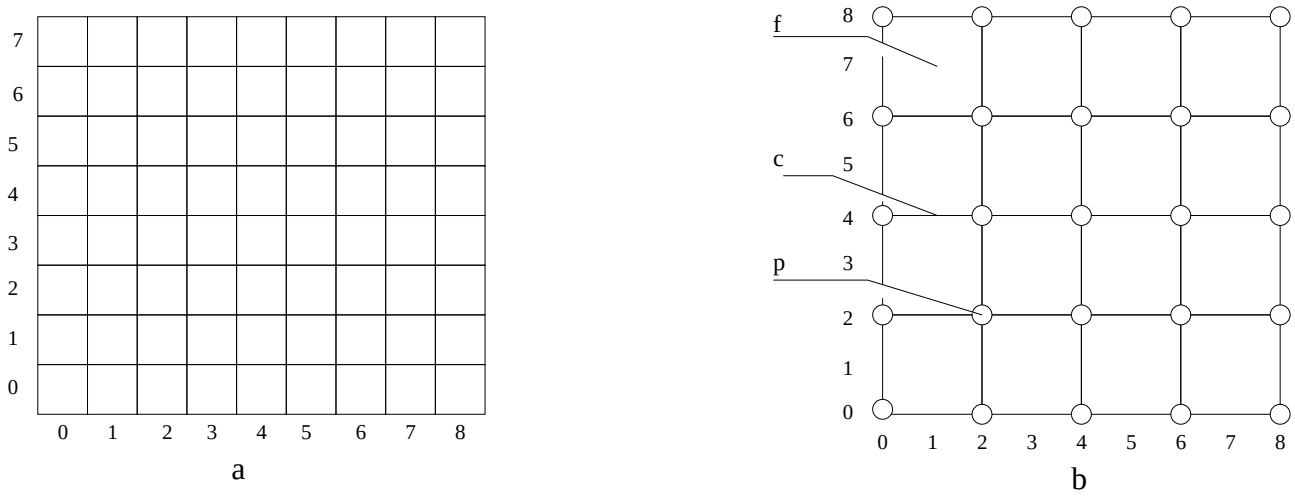

Figura E.1: La malla estándar (a) y la malla combinatoria (b).

E.2. Estructuras de Datos usando Listas de Elementos del Espacio

Una estructura de datos, diseñada para representar eficientemente la información topológica de una imagen digital debe satisfacer las siguientes demandas:

1. La estructura debe tener información topológica suficiente para obtener conocimiento de las relaciones topológicas entre las partes de la imagen o de una escena 3D sin una búsqueda. A las relaciones topológicas pertenecen principalmente la relación de incidencia y adyacencia.

2. La estructura debe ser capaz de representar correctamente complejos no propios, los cuales se emplean frecuentemente en las investigaciones topológicas porque contienen muchos menos elementos que los complejos propios.

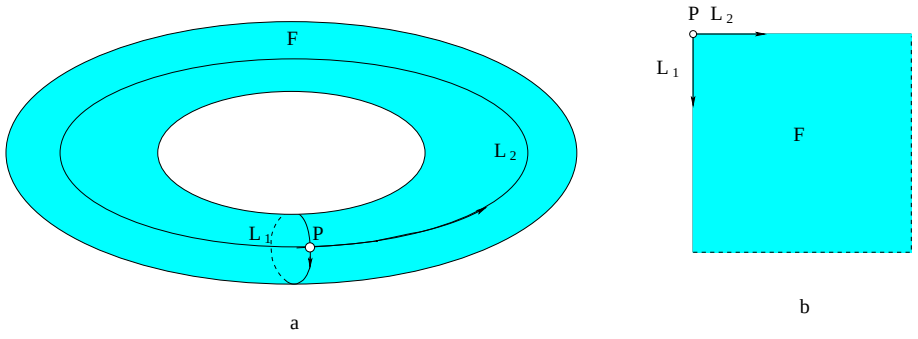

Figura E.2: Representación de la superficie de un toro (a) y de un complejo simple (b).

La superficie de un toro se puede representar como un complejo que consiste de una 0−celda dos 1−celdas y una 2−celda como en la Figura E.2. Esta representación es muy simple, sin embargo, si se tratara de interpretar como un complejo AC se presentan varios problemas, pues los complejos correspondientes a la Figura E.2 (a, b) son el mismo, ya que tienen el mismo conjunto de 4 celdas la misma relación de ligado y las mismas dimensiones de las celdas, la diferencia entre los dos complejos es que cada una de las celdas L_1 y L_2 en a ligan a la 2−celda dos veces en ambos lados.

Ya que el objetivo es el de considerar un acercamiento puramente combinatorio con ninguna relación a un espacio euclidiano, consideramos la posibilidad de usar la noción de celdas propias.

Las estructuras de datos conocidas no cumplen con los requisitos 1 y 2. La matriz de incidencias permite codificar cualquier complejo de celda propia. Contiene la información topológica compleja; sin embargo, no es adecuada para codificar los complejos no-propios. Además de eso, su uso no es económico: contiene en el caso de un complejo n−dimensional

$$\sum_{k=1}^{n} N_{k-1} * N_k$$

elementos donde N_k es el número de celdas k−dimensionales. Este número es muy grande para casos relevantes. Por estas razones, las estructuras de datos que usan listas **lineales** en lugar de **cuadráticas** de elementos del espacio son preferibles.

Una de las estructuras más populares en graficación por computadora y modelado geométrico 3D es la **Representación de ligado**. Esta estructura permite seguir fácilmente el borde de una cara 2D de un cuerpo. Sin embargo, para encontrar cuáles cuerpos en una escena 3D son adyacentes el uno al otro demanda una búsqueda exhaustiva a través de la descripción de todos los vértices de todos los cuerpos en la escena.

La estructura de datos conocida como **mapa n-G** o **mapa Topológico** tiene mucha similitud con la estructura de **Lista de Celdas**, que se definirá más adelante. La principal diferencia consiste en el uso de dos copias de cada celda de dimensión más grande que cero, diferenciándose sólo por su orientación mientras que en la lista de celdas de dimensión dos y tres la orientación es definida simplemente por el signo del índice correspondiente.[1] Por lo tanto, estas estructuras son más complicadas que la lista de celdas.

E.3. Bloques Complejo

Antes de poder ver cómo está compuesta la lista de celdas 2−dimensional necesitamos ver primero el concepto de bloques complejos.

Al realizar análisis de imágenes se tiene que lidiar con complejos que contienen miles de celdas. En éstos, es recomendable subdividir un complejo dado en varios grandes subcomplejos, cada uno de los cuales puede considerarse como uniforme con respecto a una

[1]Estas copias son llamadas "*dardos*" y "*medias aristas*". El autor emplea junto con los dardos la noción de celdas.

propiedad específica para una aplicación concreta. Por ejemplo, conjuntos de pixeles de un color constante. Las particiones de un complejo que satisfacen ciertas condiciones son llamadas bloques complejos. Consideremos la partición R de un complejo $n-$dimensional A en subcomplejos $k-$dimensionales S_i^k con $k = 0, 1, ..., n$, $i = 0, 1, ..., m$. Subconjuntos con $k = 0$ son algunas de las $0-$celdas s de A; cada subconjunto con $k > 0$ es combinatoriamente homeomorfo a una bola abierta $k-$dimensional.

Es posible definir un nuevo complejo AC B cuyas celdas corresponden a los subcomplejos S_i^k anteriores. Este complejo se denomina **bloque complejo** de A. Sus celdas $b_i \in B$, $i = 0, 1, ..., m$ son llamadas **bloques de celdas**; los subcomplejos S_i^k que corresponden a los bloques de celdas son llamados **bloques**. Por lo tanto; los bloques de celdas son elementos de un bloque complejo mientras que los bloques son subcomplejos del complejo original A. La función de dimensión Dim del bloque complejo de A y su relación de ligado BR son definidas de esta manera:
La dimensión $Dim(b_i, B)$ del $i-$ésimo bloque de celda b_i es igual a la dimensión del bloque correspondiente S_i^k en A; el $i-$ésimo bloque de dimensión k liga el $j-$ésimo bloque con dimensión $m > k$ si hay en A dos celdas $c_1 \in S_i^k$ y $c_2 \in S_i^k$ tal que c_1 liga a c_2. Esta definición de dimensión Dim está de acuerdo con la Definición 2.14.

Definición E.1 (Bloque $k-$dimensional)
La tripleta $B(A) = (EB, BR, Dim)$ se llama un bloque complejo de A si existe una partición de A en subcomplejos S_i^k, mientras que cada uno de los S_i^k es combinatoriamente homeomorfo a una bola abierta $k-$dimensional. EB es el conjunto de los bloques de celdas. Cada bloque de celda $b_i \in EB$ corresponde a un subcomplejo S_i^k llamado **bloque $k-$dimensional** o **k-bloque** de A. El par ordenado (b_i, b_j) de bloques de celdas está en la relación de ligado BR si existen dos celdas en A $c_1 \in S_i^k$ y $c_2 \in S_i^k$ tal que c_1 liga a c_2. La función de dimensión Dim asigna a cada bloque de celda b_i de $B(A)$ la dimensión del bloque correspondiente S_i^k.

Los bloques de celdas y los bloques complejos se pueden orientar de acuerdo con el concepto ya dado. La definición de bloque de celdas incidente de un bloque complejo se da a partir de la definición de relación de ligado BR que se dio anteriormente y de la definición de subcomplejos incidentes: Dos bloques de celdas $b(i)$ y $b(j)$ corresponden a subcomplejos S_i^k y S_j^m son incidentes el uno al otro si bien son idénticos o uno de ellos liga al otro de acuerdo con la relación de ligado BR.

Un ejemplo de un bloque complejo $2-$dimensional se muestra en la Figura E.3, las tres áreas de color en la figura (a) son bloques de celdas $2-$dimensionales. Los bloques de celdas $0-$dimensionales se muestran como los círculos negros grandes, mientras que los bloques de celdas $1-$dimensionales son las lineas sólidas que van de un bloque de celda $0-$dimensional a otro. En la Figura E.3 (b) se observa el complejo original cuyos subcomplejos $2-$dimensional corresponden a los bloques de celdas del mismo color. Los triángulos, líneas y circulos negros pequeños son celdas del complejo original. Las líneas poligonales sólidas en la figura (b) son bloques $1-$dimensionales que corresponden a los bloques de celdas $1-$dimensionales de la Figura E.3 (a).

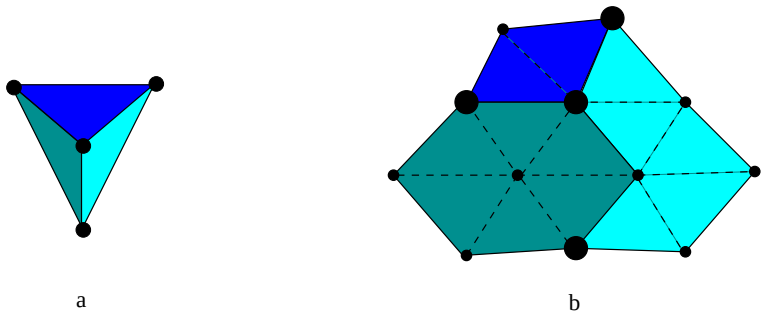

Figura E.3: Ejemplo de un bloque complejo 2−dimensional (a) y su complejo subyacente dividido.

E.4. Lista de Celdas 2 dimensional

Una estructura de datos 2D que satisface las condiciones de la Seccion 5.3 se le conoce como **Lista de Celdas**. La peculiaridad de esta estructura es que la información topológica está representada de manera explícita en ella: es posible obtener directamente la información acerca de las regiones frontera, los puntos finales y las regiones incidentes de líneas sin realizar una búsqueda. La información acerca de las adyacencias está disponible con una búsqueda local restringida ya que .ªdyacente"significa ïncidente a un elemento incidente".

La estructura de datos lista de celdas está basada en la noción topológica de un bloque complejo para complejos AC que definimos anteriormente. Los bloques son subcomplejos del complejo que representa la imagen. Debemos considerar además de estos bloques, las regiones. Antes de poder definir regiones es necesario definir lo que es un sólido

Definición E.2 (Subcomplejo Sólido)
Un subcomplejo S^n de un complejo $n-$dimensional C^n es llamado **sólido** si y sólo si es homogéneamente $n-$dimensional y contiene el subcomplejo $\text{Int}(Cl(S^n, C^n), C^n)$.

Ahora que hemos definido lo que es un sólido hay que definir lo que es una región:

Definición E.3 (Región)
Una región es un subconjunto $n-$dimensional de un sólido conexo abierto del espacio $n-$dimensional.

A diferencia de un bloque, una región no tiene que ser homeomorfa a una $n-$bola abierta. Hay un bloque de celdas correspondiente a cada bloque. Los bloques de celdas son elementos del complejo AC llamado el bloque complejo. Cada bloque de celdas $k-$dimensional ($k-$bloque de celdas), con $k > 0$, está acotado por algún $(k-1)-$bloque de celdas; cada $k-$bloque de celdas con $k < n$, donde n es la dimensión del complejo original, acota algunas $(k+1)-$bloques de celdas. En un bloque complejo 2D, los bloques de celdas 2D corresponden a las regiones. Este último debe ser especificado por algún procedimiento de segmentación asignando a todos los pixeles de una región, la etiqueta de esa

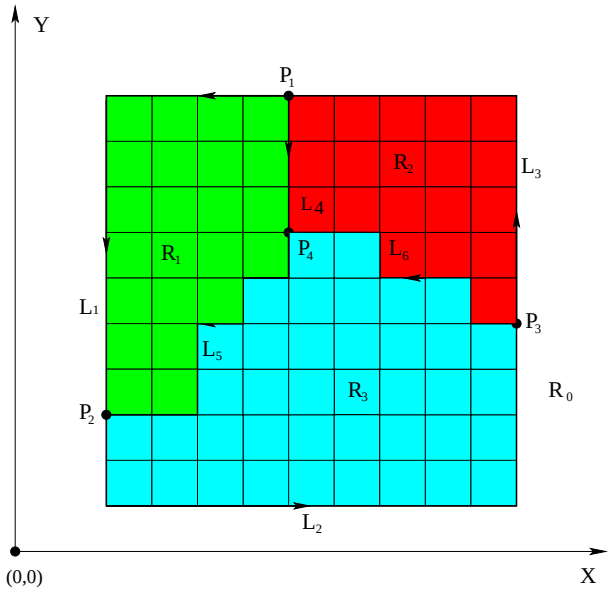

Figura E.4: Ejemplo de un bloque complejo 2−dimensional P_i son los 0−bloques(puntos), L_i son los 1−bloques(lineas) y los R_i son los 2−bloques(regiones).

región. El 0−bloque de celdas corresponden a puntos de bifurcación donde se encuentran tres o más regiones; los 1−bloques de celdas corresponden a segmentos de las frontera de las regiones. Un 1−bloque de celdas con orientación está acotado por dos 0−bloques de celdas llamados punto inicial y final.

Un bloque complejo 2−dimensional puede ser descrito por una lista de celdas 2D. Su versión completa consiste de cuatro listas parciales. Cada fila de la lista de k−bloques de celdas corresponde a un k−bloque de celda b^k y contiene los índices de todos los bloques de celdas con orientación incidentes a b^k. En las siguientes tablas se muestra un ejemplo de la lista de celdas describiendo el bloque de celdas de la Figura E.4.

Índice	X	Y	N_{linea}	Lineas
P_1	6	10	3	$+L_1, -L_3, +L_4$
P_2	2	3	3	$-L_1, +L_2, -L_5$
P_3	11	5	3	$-L_2, +L_3, +L_6$
P_4	6	7	3	$-L_4, +L_5, -L_6$

La tabla anterior nos muestra la lista parcial de los 0−bloques de celdas, esta lista contiene tanto datos que corresponden a la información geométrica que nos permiten la reconstrucción de la imagen original, así como la información topológica. Las columnas X y Y corresponden a las coodernadas de cada punto de bifurcación en la imagen original. Si sólo se desea la información topológica estas columnas pueden ser omitidas. La lista también indica para cada 0−bloque de celdas P_i el número de todos los 1−bloques de celdas N_{linea} incidentes a P_i y la lista de los 1−bloques con signo. El signo negativo de un índice L_k en la fila de P_i indica que P_i es el punto final de L_k.

La siguiente tabla nos muestra la lista parcial de los 1−bloques de celdas.

Índices	Punto		Región		Métrica	
	Inicial	Final	Derecha	Izquierda	Inicio	Final
L_1	P_1	P_2	R_0	R_1	M_1	M_1
L_2	P_2	P_3	R_0	R_3	M_2	M_3
L_3	P_3	P_1	R_0	R_2	M_4	M_4
L_4	P_1	P_4	R_1	R_2	-	-
L_5	P_4	P_2	R_1	R_3	M_5	M_5
L_6	P_3	P_4	R_2	R_3	-	-

Las últimas dos columnas en la lista parcial de los 1−bloques de celdas contienen los índices de los llamados puntos métricos, estos puntos no son puntos de bifurcación. Sus coordenadas están en la lista parcial de los datos de la métrica. Estos son vértices que corresponden a una línea poligonal aproximada al correspondiente segmento del borde de una región. Los puntos métricos se encuentran entre los puntos finales de una línea y son necesarios para describir características geométricas de la imagen representada por el bloque complejo y sirven para la reconstrucción de la imagen. Las coordenadas pueden omitirse en una versión puramente topológica de la lista de celdas. La siguiente tabla nos muestra los puntos métricos junto con sus etiquetas y coordenadas.

Etiqueta	X	Y
M_1	2	10
M_2	2	1
M_3	11	1
M_4	11	10
M_5	3	3

En la lista parcial de los 2−bloques de celdas la variable N_{CI} representa el número de bloques de celdas en la frontera del correspondiente 2−bloque celda. Los índices de estos bloques son mostrados en la última columna como una lista encadenada. La lista encadenada también define el orden de los bloques de celdas en la frontera. El uso de la lista encadenada es ventajoso durante la creación automática de la lista de celdas, ya que hace más fácil los cambios en está. Tan pronto como la lista de celdas está disponible, cada lista encadenada puede ser remplazada por un arreglo el cual toma menos memoria.

Índice	N_{CI}	Apuntador	Lista Encadenada
R_0	6	$Z_1 \rightarrow$	$P_1 \rightarrow +L_1 \rightarrow P_2 \rightarrow +L_2 \rightarrow P_3 \rightarrow +L_3 \rightarrow P_1$
R_1	6	$Z_2 \rightarrow$	$P_1 \rightarrow +L_4 \rightarrow P_4 \rightarrow +L_5 \rightarrow P_2 \rightarrow -L_1 \rightarrow P_1$
R_2	6	$Z_3 \rightarrow$	$P_1 \rightarrow -L_3 \rightarrow P_3 \rightarrow +L_6 \rightarrow P_4 \rightarrow -L_4 \rightarrow P_1$
R_3	6	$Z_4 \rightarrow$	$P_1 \rightarrow -L_5 \rightarrow P_4 \rightarrow -L_6 \rightarrow P_3 \rightarrow -L_2 \rightarrow P_2$

Modificaciones de la lista de celdas son posibles dependiendo de la especificación del problema a resolver. Algunas modificaciones de la lista de celdas 2D son:

1. Es posible construir cualquiera de las dos: una lista de celdas geométrica o una lista de celdas topológica la cual no contiene coordenadas. Esta última lista representa sólo datos acerca de la relaciones de ligado de los bloques y alguna información adicional en el caso de un complejo no-propio.

2. Si una imagen contiene regiones con "hoyos", los cuales son regiones más pequeñas de algún color diferente en el interior de una región más grande, entonces la partición de la imagen no corresponde a un bloque complejo. De hecho, un bloque complejo debe ser, por definición, homeomorfo a una bola abierta y una región con hoyos no satisface esta condición. En tales casos la partición se considera como un bloque complejo no propio. Es posible describir tal complejo mediante una lista de celdas. Entonces es apropiado, pero no necesario, acompañar la lista de celdas por una estructura la cual llamaremos **árbol de inclusión**. Un árbol de inclusión es una gráfica dirigida, cuyos vértices corresponden a regiones, mientras una arista dirigida apunta desde una región incluida a otra. Un árbol de inclusión puede ser descrito por una estructura en la cual un conjunto de tres índices $(padre, hijo, hermano)$ es asignado a cada vértice V. El índice *padre* apunta a la región directamente incluida de V, el índice *hermano* (el cual indica *el siguiente hermano mayor*) apunta a alguna región diferente de V e incluido por el padre de V. El índice $hijo$ (el cual es el $hijo$ *mayor*) apunta a una de las regiones incluidas en V.

3. Es posible reducir el espacio en memoria que se requiere para un lista de celdas mientras excluyamos la redundancia. Una lista redundante contiene ambas relaciones, *ligaa* y *es ligado por*, lo cual es necesario para encontrar las relaciones topológicas de los bloques sin una búsqueda. Esta propiedad de una lista redundante, hace el análisis de la estructura de un imagen muy rápida. Sin embargo, para reconstruir la imagen desde su lista de celdas es suficiente con tener la relación de *es ligado*. En una lista reducida la sublista del $0-$bloque de celdas permanece sin cambios, en la sublista del $1-$bloque de celdas las columnas correspondientes a los puntos finales son descartadas. La sublista del 2-bloque de celdas es completamente descartada.

Recordando el problema de la representación de complejos no-propios, veamos cómo el complejo no-propio de un toro que se mostró en la figura E.2, puede describirse como una lista de celdas.

Índice	N_{lin}	Lineas
P	4	$+L_2, +L_1, -L_2, -L_1$

La lista parcial de los 0-bloques consisten en este caso de una fila. El valor de N_{lin} indica el número de los "plugs"de los 1-bloques de celdas (líneas) incidentes a P y sus índices con signo. El signo negativo de un índice de una línea indica que P es el punto final de la línea con orientación.

Etiqueta	Punto		Región	
	Inicial	Final	Derecha	Izquierda
L_1	P	P	$-F$	$+F$
L_2	P	P	$-F$	$+F$

La lista parcial de los 1−bloques de celdas contiene para cada 1−bloque de celdas con orientación su punto inicial y su punto final, que en este caso es siempre el mismo. También contiene índice con signo de dos 2−bloques de celdas (regiones) incidentes que difieren únicamente por sus signos. Un signo negativo indica que la orientación inducida de los 1−bloques en los 2−bloques es −1.

Etiqueta	NCI	Apuntador	Lista Encadenada
F	8	Z→	$P \rightarrow +L_1 \rightarrow P \rightarrow +L_2 \rightarrow P \rightarrow -L_1 \rightarrow P \rightarrow -L_2$

La lista parcial de los 2−bloques celdas consisten en este caso de una sola fila. Los índices de los bloques de celdas en la frontera del 2−bloque de celdas se nos muestra como una lista encadenada en la última columna. El valor N_{CI} es el número de elementos de esa lista. El orden de los elementos en la lista corresponde a la orientación de los 2−bloques de celda. El apuntador Z nos da el inicio de la lista.

E.5. Lista de Celdas 3 dimensional

Una lista de celdas 3D describe un bloque complejo 3D de acuerdo con la Definición E.1. Sus bloques de celdas corresponden a los bloques de una partición de una imagen 3D. La partición es especificada por algún procedimiento de segmentación que asigna a todos los voxeles de un segmento la equiteta de esos segmentos. El interior de un segmento es un 3-bloque y se le llama **volumen**.

Un 2-bloque F_m es el interior de la intersección de las fronteras de dos volúmenes adyacentes V_i Y V_k en una de las fronteras,

$$F_m = Int(\delta V_i \cap \delta V_k, \delta V_i) = Int(\delta V_i \cap \delta V_k, \delta V_k)$$

Un 2-bloque F_{ik} incidente al volumen V_i se llama **cara** de V_i. Cada cara es incidente a dos volúmenes.

De manera similar, un 1-bloque L_j es el interior de la intersección de las fronteras de tres o más caras adyacentes F_i (i=1,2,3, ...) en una de las fronteras, se le conoce como **línea**. Un 1-bloque es incidente a tres o más caras y a tres o más volúmenes.

Un 0-bloque es una intersección de las fronteras de tres o más líneas y es llamado **punto de bifurcación**. Un 1-bloque inicia y termina en un 0-bloque.

De acuerdo con la Definición C.8 podemos llamar al bloque subcomplejo compuesto de todos los bloques de celdas incidentes a un bloque de celdas b dado, excepto b, la **estructura incidente de b**. En un espacio 2D la estructura de incidencia de una celda propia es una secuencia cíclica isomorfa a una esfera 1-dimensional. Se puede representar como una lista encadenada o como un arreglo de índices de los bloques de celdas. En el caso de 3D la estructura de incidencia de un 0-bloque o un 3-bloque es isomorfo a una esfera 2-dimensional la cual en el caso general se puede representar sólo por una lista de celdas 2D o una estructura equivalente. Por lo tanto, se requiere una lista de celdas

2D para representar la estructura de incidencias de cada punto de bifurcación y cada volumen de un bloque complejo 3D. Sin embargo, hay casos en los que las estructuras de incidencia son isomorfas a esferas 2-dimensional. Una 2-esfera combinatoria es isomorfa a la superficie de un poliedro convexo y, por lo tanto, puede representarse como una lista de caras poligonales. El conjunto Cl^* de un 2-bloque de celda que corresponde a una cara de un poliedro puede representarse como una secuencia cíclica de 0-bloques y 1-bloques. Por lo tanto, una lista de caras y el conjunto de la correspondiente secuencia cíclica describen completamente la estructura de incidencia de un 3-bloque. De manera similar, una lista de líneas incidentes y el conjunto de la correspondiente secuencia cíclica describen el conjunto SON^* de cada línea (una secuencia cíclica de caras y volúmenes alternantes) incidente a un punto de bifurcación describen completamente la estructura de incidencia del punto de bifurcación. Esta es nuestra base teórica de las listas de celdas.

Definimos para cada cara (2-bloque) f su orientación de acuerdo con los conceptos dados anteriormente, mientras se especifica un orden cíclico de las líneas y los puntos en su frontera. Una línea también está orientada, esto significa que tiene un punto de bifurcación inicial y uno final. Una línea en la frontera de f puede ser orientada coherente o anti-coherente a f. En el último caso, la línea es orientada en contra del orden cíclico de la frontera de f. El orden cíclico de la frontera de f define de manera única la función de ligado coherente por pares de f con relación a las celdas que la ligan, por lo tanto, definiendo su orientación. Cada cara f tiene una orientación inducida en cada dos volúmenes incidentes a ella. Un volumen es orientado si su función de ligado coherente por pares con relación a sus caras es definida. Esta es la lista de signos asignados a cada cara orientada que liga el volumen. Un volumen V se dice que se encuentra en el lado positivo de una cara f incidente a V si f tiene un signo positivo en su función de ligado coherente por pares. Como en el caso 2D, hacemos una distinción entre una lista de celdas 3D geométrica y una topológica, esta última no contiene coordenadas.

Consideremos el ejemplo de la Figura E.5, este es un complejo que contiene sólo dos cubos. Una cara de cada cubo es la intersección de sus fronteras, mientras que la unión de las otras cinco caras de cada cubo es considerada como una cara individual.

La lista de celdas de la imagen 3D de la Figura E.5 se muestra a continuación:

Índice	N_{SON}	Líneas
P_1	2	$+L_1, -L_2$
P_2	2	$-L_1, +L_2$

La lista parcial de puntos es similar a la de una lista de celdas 2D. Se ha omitido la información geométrica en este caso, esto es las coordenadas.

Índice	Inicial	Final	N_{SON}	Apuntador	Lista Encadenada
L_1	P_1	P_2	5	Z_1	$-F_1 \to V_1 \to -F_2 \to V_2 \to +F_3 \to V_0$
L_2	P_2	P_1	5	Z_2	$-F_1 \to V_1 \to -F_2 \to V_2 \to +F_3 \to V_0$

Aquí N_{SON} denota el número de bloques celdas en el conjunto $SON*$ de la línea L_i, esto es el número de bloques de celdas ligadas por L_i. Estos bloques de celdas son listadas

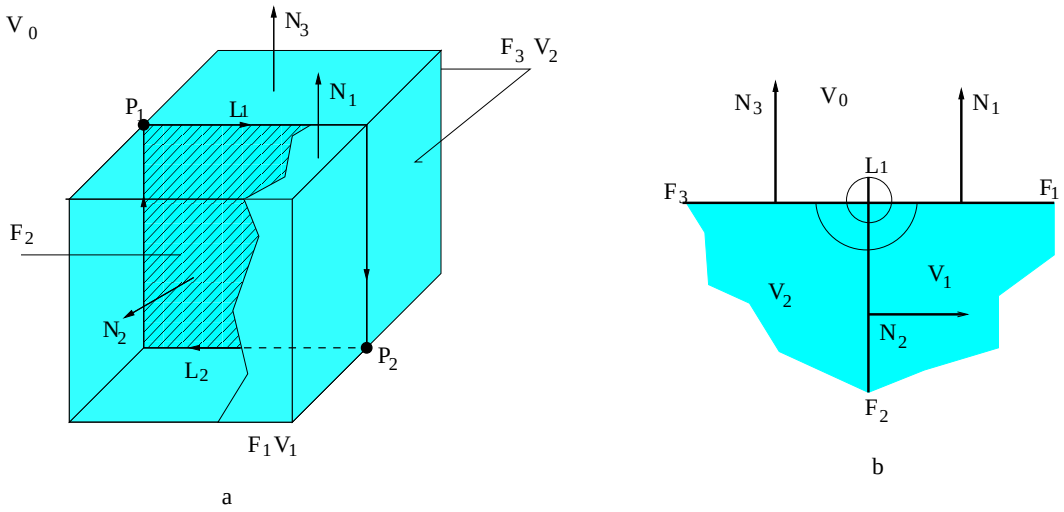

Figura E.5: Un bloque complejo simple (a) y su sección cruzada perpendicular a L_1(b).

en la lista encadenada en la última columna. El orden de la secuencia corresponde a una rotación derecha en L_i. Un signo negativo antes de una etiqueta de una cara indica que es orientada contraria a la dirección de rotación. El índice V_0 representa el volumen afuera de los cubos. El apuntador Z_i apunta al principio de la lista encadenada.

Índice	+Vol	-Vol	N_{Cl}	Apuntador	Lista Encadenada
F_1	V_0	V_1	4	Z_3	$P_1 \rightarrow +L_2 \rightarrow P_2 \rightarrow +L_2$
F_2	V_1	V_2	4	Z_4	$P_1 \rightarrow +L_2 \rightarrow P_2 \rightarrow +L_2$
F_3	V_0	V_2	4	Z_3	$P_1 \rightarrow -L_1 \rightarrow P_2 \rightarrow -L_2$

La lista de caras contiene, para cada cara F_i, el índice de dos volúmenes ligados por F_i. El volumen denotado por $+Vol$ cae en la dirección de la normal positiva de F_i. El resto de la lista es similar a la lista de celdas 2D.

Índice	N_{Cl}	Caras
V_1	2	$+F_1, -F_2$
V_0	2	$+F_2, +F_3$

La lista parcial de volúmenes contiene, para cada volumen V_i, el número de las caras incidentes N_{Cl} las cuales están listadas en la última columna. El signo negativo antes del índice de una cara en la fila de V_i indica que la normal ala cara esta apuntando afuera de V_i.

Bibliografía

[1] Rosenfeld A. Connectivity in digital pictures. *Journal of the ACM*, 17(1):146–160, 1970.

[2] Rosenfeld A. A note on perimeter and diameter in digital pictures. *Information and Control*, 24(4):384 – 388, 1974.

[3] Huang W. Et All. A novel algorithm for image perimeter computation based on non-symmetry anti-packing representation. *Journal of Shanghai University (English Edition)*, 12(6):524 – 530, 2008.

[4] Koplowitz J. Design of perimeter estimators for digitized planar shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):611–622, 1989.

[5] V. Kovalevsky. Finite topology as applied to image analysis computer vision. *Computer Vision, Graphics, and Image Processing*, 46(1):141 – 161, 1989.

[6] V. Kovalevsky. Digital geometry based on the topology of abstract cell complexes. *Discrete Geometry for Computer Imagery*, pages 259 – 284, 1993.

[7] V. Kovalevsky. *Digital and Image Geometry, Advanced Lectures [based on a winter school held at Dagstuhl Castle, Germany in December 2000].* Springer-Verlag, London, UK, 2001.

[8] V. Kovalevsky. *Proceedings of the 10th international conference on Combinatorial Image Analysis*. Springer-Verlag, London, UK, 2004.

[9] V. Kovalevsky. Axiomatic digital topology. *Journal of Mathematical Imaging and Vision*, 25(1-2):41 –58, 2006.

[10] V. Kovalevsky. *Geometry of Locally Finite Spaces, Computer Agreeable Topology and Algorithms for Computer Imagery.* Editing House Dr. Baerbel Kovalevski, Berlin, Germany, 2008.

[11] B. Mendelson. *Introduction to Topology.* Dover Books on Mathematics, USA, tercera edition, 1990.

[12] C. Prieto. *Topología Básica.* Fondo de Cultura Económica, México, segunda edition, 2013.

[13] T. Y. Rosenfeld, A. & Kong. *Topological Algorithms for digital Image Processing.* Elsevier Science B.V, Amsterdam, The Netherlands, 1996.

Índice alfabético