UNIVERSIDAD ABIERTA INTERAMERICANA
Facultad de Tecnología Informática

Carrera: Licenciatura en Matemática

TOPOLOGIA DIGITAL

Base para la visión artificial.

Autor: Jorge Alejandro Kamlofsky
Directores: Dra. Samira Abdel Masih – Ing. Néstor Balich

TESIS PRESENTADA PARA OPTAR AL TÍTULO DE LICENCIADO EN MATEMÁTICA

- Abril de 2011 -

Resumen

El presente trabajo contiene algunos resultados referentes al reconocimiento inteligente de imágenes digitales. Más específicamente, aplica y justifica matemáticamente un algoritmo que permite recorrer el borde de un objeto que se encuentra dentro de una imagen digital.

La disciplina indicada para el estudio matemático de imágenes digitales es la Topología Digital, la cual fue desarrollada por Azriel Rosenfeld en 1970, quien además planteó estrategias para el reconocimiento de formas que eventualmente pueden aparecer en una imagen digital.

El desarrollo de este trabajo se basa en los resultados obtenidos por Ulrich Eckhardt, Longin Latecki ([1] y [2]) y Azriel Rosenfeld ([3]). Este último autor fue quien ideó, en 1979, el algoritmo que se analiza en esta obra.

A los efectos de facilitar al lector la comprensión de los temas, éstos se ordenan y exponen de la siguiente manera:

- Capítulo 1: Se refiere al concepto y propiedades de una imagen digital.
- Capítulo 2: Se introducen las nociones básicas de la Topología Digital.
- Capítulo 3: Contiene la parte principal del trabajo, basada en los resultados obtenidos por Azriel Rosenfeld acerca de la Topología Digital. Allí se consideran los conceptos básicos de la topología usual en el plano y se los redefine en este nuevo ámbito: el de la Topología Digital. Se demuestran propiedades, teoremas, proposiciones y se incluyen gráficos ilustrativos.
- Capítulo 4: Se enuncia un algoritmo que recorre la frontera de un objeto dentro de una imagen digital: “el algoritmo BF4/BF8”.
- Capítulo 5: Se implementa “el algoritmo BF4/BF8” en una aplicación y se describe su código fuente.
- Capítulo 6: Se elaboran conclusiones.

En la actualidad existen numerosas aplicaciones basadas en el concepto que se presenta en este trabajo, como por ejemplo, en el diagnóstico por imágenes, en seguridad perimetral y automatismos. El límite de su alcance está sólo en nuestra imaginación.

Palabras claves:

Topología Digital, visión artificial, imágenes digitales, reconocimiento de patrones, curvas digitales, arcos digitales, Teorema de Jordan para curvas digitales, el plano digital.

Índice

Introducción

¿Dónde está la pelota?

En el CAETI (Centro de Altos Estudios en Tecnología Informática), centro de investigación dependiente de la Facultad de Tecnología Informática de la Universidad Abierta Interamericana, existe un grupo de investigación en robótica al cual pertenezco.
Dicho grupo posee la necesidad de innovación permanente que requiere esta disciplina. En particular, en este Centro se encuentra y reúne el grupo que participa en los campeonatos nacionales e internacionales de fútbol por robots que representa a dicha universidad.
Básicamente, un partido de fútbol jugado con robots se lleva a cabo de la siguiente manera:

1. Se presentan dos equipos en la cancha, con características físicas acotadas según las reglas del torneo.
2. Cada equipo de robots es manejado por un servidor que ordena sus movimientos por radio-frecuencia.
3. No existe control humano durante el juego. El mismo consiste en un partido de "programa contra programa".
4. En el centro de la cancha y a cierta altura se dispone de una cámara que filma los movimientos tanto de la pelota como de los robots participantes.
5. Esa filmación se distribuye a ambos servidores y en base a esa filmación deben actuar los programas.
6. La pelota es de color anaranjado, esférica y su medida es dato.
7. La luminosidad es constante y es dato.

Actualmente el programa de este equipo posee una estrategia de juego simple, pero efectiva.
Para el procesamiento de las imágenes provenientes desde la cancha, el equipo utiliza un algoritmo no propio. Se pretende desarrollar un algoritmo propio y se espera que sea más eficaz, de manera que permita así disponer de mayor cantidad de recursos informáticos para poder implementar en el futuro estrategias de juego más complejas.
El presente trabajo se limita a identificar la pelota como un elemento de la imagen con características propias, ubicar e informar su posición en la cancha, pretendiendo que esta tarea ocupe la menor cantidad de milésimas de segundo posible de modo de estar a la altura de las necesidades del juego.
Se descarta cualquier algoritmo que emplee lo que se denomina "fuerza bruta" para identificar a la pelota. En este caso, "fuerza bruta" consistiría en recorrer y traer a la memoria de una computadora todos los píxeles de una imagen digital y luego realizar los cálculos sobre la totalidad de ellos. Con esta metodología, el algoritmo se tornaría bastante ineficiente, ya que debería almacenar millones de pixeles, lo cual haría imposible su implementación en el juego, o bien requeriría gran cantidad de recursos informáticos.

Estrategias a utilizar para identificar a la pelota

Para localizar eficazmente la pelota se utiliza la llamada ***"Visión artificial por Reconocimiento de Patrones"*** (**ViPR** en sus siglas en inglés). Para ello, la mayoría de

la bibliografía y trabajos de investigación se basan en el concepto matemático de la "Topología Digital".
En este contexto se aplica el ***Teorema de curvas de Jordan en el plano digital***, que afirma que toda curva simple cerrada contenida en un conjunto conexo separa al conjunto en dos subconjuntos: su interior o agujero y su complemento o fondo. Este Teorema se lo presenta previa definición de otras cuestiones topológicas como vecinos, vecindades, adyacencias, caminos, conectividad, arcos y curvas. El Teorema de curvas de Jordan nos permitirá separar a un objeto del resto de la imagen digital en la que se encuentra inmerso.
Para lograr esto se tratará primeramente de identificar a la curva que representa el borde o frontera del objeto (que en nuestro caso es la pelota), utilizando un algoritmo presentado en el mismo trabajo de Rosenfeld [3]: "el algoritmo BF4/BF8". Este permitirá localizar los puntos de un conjunto conexo (la pelota) que limitan con su complemento y los del complemento que limitan con el conjunto, conformando así la frontera del mismo.
La tarea siguiente consistirá en definir qué tipo de objeto tiene a esa curva por borde, analizando las propiedades geométricas del mismo. Estas se pueden comparar con ciertos patrones previamente recopilados, identificando de este modo al objeto en cuestión.
Los trabajos en el área de Topología Digital realizados por el Dr. Ulrich Eckhardt y Longin Latecki, en particular el trabajo denominado "Topologías para los espacios digitales Z^2 y Z^3" [1] marcan una clara guía acerca del estado del arte en esta materia. En particular, en su resumen inicial dice:

"Mostramos que hay sólo dos topologías en Z^2 y cinco topologías en Z^3 cuyos conjuntos son conexos en el sentido intuitivo. Las dos topologías en Z^2 son bien conocidas (por ejemplo, una la presenta D. Marcus, F Wyse en [4], y la segunda E. Khalimsky et al. en [5] y encuentran aplicaciones en gráficos de computadoras y visión artificial (por ejemplo A. Rosenfeld [3], y T. Y. Kong et al [6]). Dos de las cinco topologías de Z^3 son producto de las topologías conocidas para Z^1 y Z^2. Las restantes tres también se generan desde las dos topologías de Z^2".

Relevancia del tema

En la actualidad existen resultados prometedores de aplicaciones basadas en el concepto de Topología Digital. El desarrollo de esta tecnología puede abrir una nueva ventana en los procesos de automatización. Con sólo tener una cámara, podremos contar autos sin tener sensores, podremos seguir un objeto o detectar fuego sin necesidad de disponer de sensores de humo ni de temperatura. Será posible, además, crear alarmas perimetrales que diferencien personas de animales, a fin de utilizarlas en seguridad aeroportuaria, seguridad vial, reconocimiento facial, dactiloscopia, diagnóstico médico por imágenes, defensa, etc.
Como se puede ver, esta línea de trabajo tiene mucho por desarrollar, tanto desde el punto de vista matemático de la Topología Digital como también en el desarrollo tecnológico del concepto ViPR y sus aplicaciones.

Acerca de nuestro problema y su correspondiente programa informático

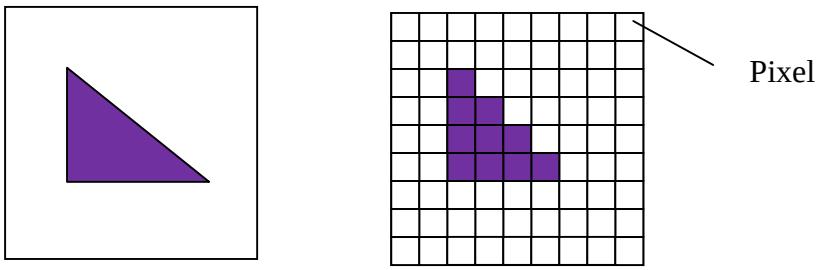
Las características o patrones del objeto a buscar son simples y son dato: Una pelotita color anaranjada de cierto diámetro. Es decir, en la imagen digital se debe buscar un objeto circular, color anaranjado, de cierta área (o cantidad de pixeles según la definición que es dato) que es único. Sin embargo, bien podemos aplicar los métodos, funciones y clases de este programa y los conceptos de este trabajo en caso de tener que buscar otro objeto con otras características o patrones.

Capítulo 1

Conceptos y propiedades básicas de una imagen digital

Capítulo 1: Conceptos y propiedades básicas de una imagen digital

En nuestra vida cotidiana observamos innumerables imágenes bidimensionales. Por ejemplo, la pintura hecha por un artista, una imagen natural capturada por una cámara, un telescopio o un microscopio. Todas ellas representan una variación continua de sombras y tonos. Por esta razón, imágenes de este tipo reciben el nombre de ***imágenes continuas o analógicas***.
Para poder almacenar este tipo de imágenes en una computadora es necesario "digitalizarlas". Este proceso consiste en dividir la imagen analógica en retículas o celdas a las que se les asigna un determinado color. Cada una de estas celdas recibe el nombre de "pixel" (contracción del inglés de las palabras **Pic**ture **el**ement).



(a) *(b)*

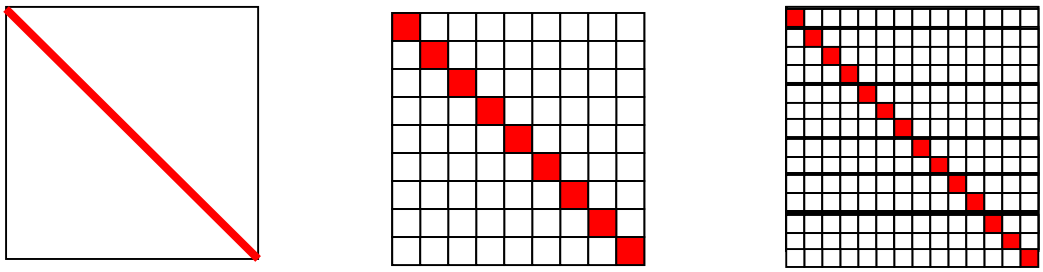
Figura 1.1: (a) Imagen analógica de un triángulo. (b) Imagen digitalizada del triángulo.

Un pixel es sólo una unidad de división sin un tamaño real concreto. No podemos afirmar si un pixel mide 1 cm. o 1 km. de lado. Es sólo una medida de división en celdas. De este modo, podemos hablar de una imagen que tenga 200 x 100 pixeles (o sea, que consta de 200 filas y 100 columnas) sin saber qué tamaño real tiene. Lo único que sabemos es que la hemos dividido en 20.000 celdas. Sin embargo, cuando a esa imagen le asignamos una resolución, entonces sí sabremos qué tamaño tiene la misma y cuánto mide cada pixel. Por ejemplo, si una imagen tiene una resolución de 10 píxeles por pulgada, significa que en cada pulgada (2.54 cm.) habrá 10 celdas. En este caso, cada pixel equivaldrá a un cuadrado de 0.254 cm. de lado. Si además se especifica que la imagen es de 200 x 100 píxeles, entonces podemos calcular su tamaño real de la siguiente manera:

Alto: 200 x 0.254 cm. = 50.8 cm. Ancho: 100 x 0.254 cm. = 25.4 cm.

Por otro lado, si dijéramos que una imagen tiene una resolución de 1 pixel por pulgada, lo que sabríamos ahora es que esa celda mide 2.54 cm. de lado.

Como vemos, la resolución de una imagen se mide en píxeles por pulgada. Cuanto mayor sea la resolución, es decir, cuanto más píxeles haya en cada pulgada, más calidad tendrá la imagen pero, desafortunadamente, ocupará más espacio en la memoria del ordenador.



(a) *(b)* *(c)*

Figura 1.2: (a) Imagen analógica de un segmento de recta. (b) Imagen digitalizada del segmento de recta, de 9x9 píxeles. (c) Imagen digitalizada del segmento de recta, de 14x14 píxeles. Obsérvese que la imagen (c) tiene mayor resolución que la imagen (b).

Cada pixel o celda de una imagen digital se identifica mediante un par ordenado de números naturales. Por ejemplo, el pixel (1, 2) es la celda ubicada en la columna 1 (contada de izquierda a derecha) y en la fila 2 (contada de arriba hacia abajo) de la imagen digital.

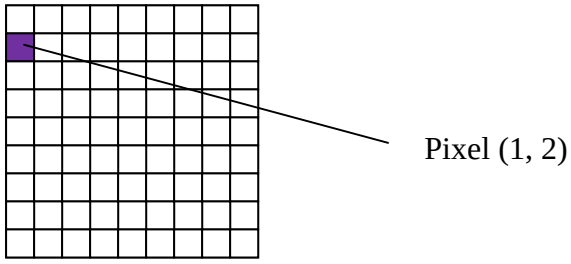


Figura 1.3: Identificación de los píxeles de una imagen digital.

Mediante el proceso de digitalización se logra transformar imágenes continuas en imágenes fácilmente manipulables y controlables tanto para nosotros como, fundamentalmente, para el ordenador.

Podríamos afirmar entonces, que una ***imagen digital es un conjunto finito de píxeles a los que se les asigna un determinado color***. Sin embargo, para analizar sus propiedades topológicas (como por ejemplo conectividad, frontera o vecindad) necesitaremos dar la definición matemática de la misma, y es la siguiente:

Definición 1.1: Imagen Digital

Una imagen digital es una función f: $A \subset \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{Z}^n$, *con n = 1,3 ó 4, que asigna a cada elemento o pixel (i, j)* $\in A$ *un único elemento f(i, j)* $\in \mathbb{Z}^n$ *que representa el color asociado a dicho pixel.*

Por ejemplo, si una imagen digital es de 4 x 3 pixeles, es decir, si consta de 4 filas y 3 columnas, entonces su dominio es A = {1, 2, 3} x {1, 2, 3, 4}. Por otro lado el valor de "n", referente al espacio de llegada $\mathbb{Z}^n$, queda determinado según el modo con que se confecciona la imagen digital, el cual pueden ser:

■ ***Modo Blanco y Negro:***

En este caso el color asociado a cada pixel sólo puede ser blanco o negro. Por convención, al color negro se le asigna el valor "0" mientras que al blanco el valor "1". En Blanco y Negro podemos definir a una imagen digital como sigue:

$$f:\ A \subset \mathbb{N} \times \mathbb{N} \rightarrow \{0\ ,\ 1\}$$

Ejemplo 1.1

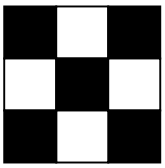
Dada la imagen digital en Blanco y Negro de 3 x 3 píxeles definida por

$$f(\ i,\ j) = \begin{cases} 1 & \text{si } (i,j) = (1,2) \text{ ó } (2,1) \text{ ó } (2,3) \text{ ó } (3,2) \\ 0 & \text{en otro caso} \end{cases}$$

Representar gráficamente dicha imagen.

Resolución:

En este caso la imagen digital es una función f: $\{1,2,3\} \times \{1,2,3\} \rightarrow \{0\ ,\ 1\}$ y corresponde al siguiente gráfico:



■ ***Modo en Escala de Grises:***

Para este modo cada pixel puede adoptar distintas gamas de grises, las cuales pueden tomar valores enteros comprendidos entre 0 y 255. En este caso, al color negro se le asigna el valor "0" mientras que al blanco el valor "255". Por ejemplo, un pixel de color gris al 20 % (es decir, 20% negro y 80% blanco) le corresponde el valor 204, mientras que a uno de color gris al 50 % le corresponde el valor 128.
En Escala de Grises se define una imagen digital de la siguiente manera:

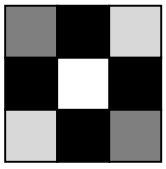
$$f:\ A \subset \mathbb{N} \times \mathbb{N} \rightarrow \{0\ ,\ 1,\ 2\ ,...,\ 255\}$$

Ejemplo 1.2

Dada la imagen digital en Escala de Grises de 3 x 3 píxeles definida por

f(1,1) = 128	f(2,1) = 0	f(3,1) = 204
f(1,2) = 0	f(2,2) = 255	f(3,2) = 0
f(1,3) = 204	f(2,3) = 0	f(3,3) = 128

Su representación gráfica es la siguiente:



■ ***Modo RGB color:***

Es el modo estándar para la representación de imágenes en pantallas. Su nombre proviene de la unión de las primeras letras de las palabras inglesas **R**ed, **G**reen y **B**lue. En este modo, cada pixel puede tomar un color que resulta de la combinación de las distintas tonalidades de rojo, verde y azul. De esta manera, a cada pixel se le asocia un vector de 3 componentes **(R, G, B)**, en las que cada una de ellas puede tomar valores enteros comprendidos entre 0 y 255. Las componentes **R**, **G** y **B** representan la cantidad de rojo, verde y azul respectivamente.
En modo RGB una imagen digital queda definida del siguiente modo:

$$f:\ A \subset \mathbb{N} \times \mathbb{N} \rightarrow \{0,1,...,255\} \times \{0,1,...,255\} \times \{0,1,...,255\}$$

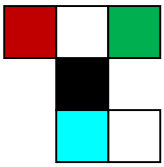
Por ejemplo, a un pixel de color rojo al 100% le corresponde el vector (255,0, 0), a uno de color verde al 100% le corresponde el vector (0,255, 0), mientras que a uno de color turquesa el vector (0, 255,255). En RGB, todas las componentes en 0 forman el negro, mientras que todas las componentes en 255 forman el blanco. Combinando los distintos valores de rojo, verde y azul, un pixel puede adoptar 256^3 = 16.777.216 colores diferentes.

Ejemplo 1.3

Dada la imagen digital en modo RGB color definida por

f(1,1) = (255,0,0)	f(2,2) = (0,0,0)	f(3,1) = (0,255,0)
f(2,1) = (255,255,255)	f(2,3) = (0,255,255)	f(3,3) = (255,255,255)

En este caso, $f:\ A \subset \mathbb{N} \times \mathbb{N} \rightarrow \{0,1,...,255\} \times \{0,1,...,255\} \times \{0,1,...,255\}$ donde A = { (1,1), (2,1), (2,2) , (2,3), (3,1), (3,3) }. Su representación gráfica es:



■ ***Modo CYMK:***

Es utilizado para impresión y su nombre proviene de la unión de las letras asociadas a las palabras **C**yan, **Y**ellow, **M**agenta y Bla**K**. En este modo, cada pixel puede tomar un color que resulta de la combinación de las distintas tonalidades de celeste, amarillo, fucsia y negro, las cuales pueden variar entre los valores 0 y 255. Así, a cada pixel se le asocia un vector de 4 componentes en las que cada una de ellas puede tomar valores enteros comprendidos entre 0 y 255. La primera componente representa la tonalidad de celeste, la segunda de amarillo mientras que la tercera y cuarta corresponden a las del fucsia y negro respectivamente.
En CYMK una imagen digital queda definida así:

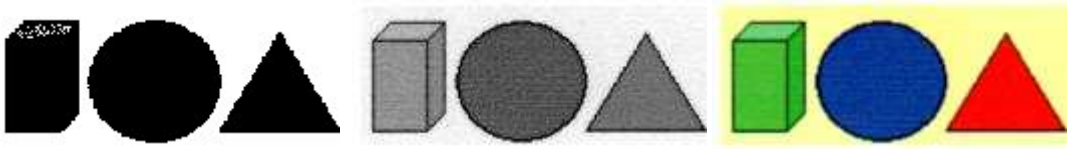
$$f:\ A \subset \mathbb{N} \times \mathbb{N} \rightarrow \{0,1,...,255\} \times \{0,1,...,255\} \times \{0,1,...,255\} \times \{0,1,...,255\}$$

En este modo, todas las componentes en 0 forman el blanco, mientras que todas en 255 hacen el negro.

El siguiente ejemplo ilustra de qué manera queda almacenada una imagen digital en la memoria del ordenador.

Ejemplo 1.4

Considere las tres imágenes de 177 x 75 pixeles, almacenadas en modo Blanco y Negro, Escala de Grises y en modo RGB color respectivamente:



(a) *(b)* *(c)*

Figura 1.4: (a) Imagen digital en Blanco y Negro. (b) Imagen digital en Escala de Grises. (c) Imagen digital en modo RGB color.

Cada una de estas imágenes tiene una resolución de 96 píxeles por pulgada, con lo cual cada pixel mide 0.2645 mm. y, de este modo, el tamaño de la imagen es:

Alto: 177 x 0.2645 mm. = 46.8 mm.
Ancho: 75 x 0.2645 mm. = 19.8 mm.

Tomemos ahora, de cada imagen, un cuadrado de 1cm de lado perteneciente al extremo superior izquierdo. En este caso, en cada centímetro habrá $\frac{96}{2.54} = 37.8 \cong 38$ píxeles, con lo cual cada imagen de 1 cm. de lado es de 38 x 38 píxeles.
Cada una de estas porciones de imágenes es almacenada en el ordenador mediante una matriz o arreglo bidimensional de 38 filas por 38 columnas.

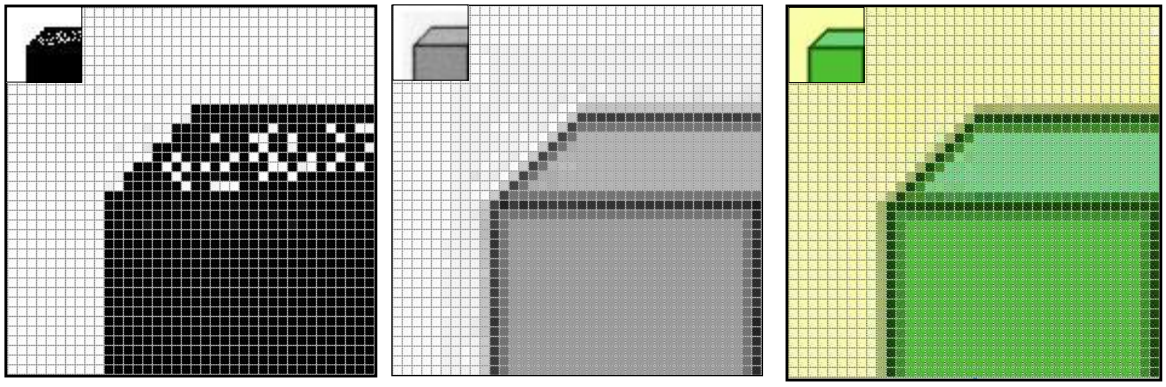


Figura 1.5: Porciones de imágenes de 1 cm. de lado extraídas de las imágenes en Blanco y Negro, en Escala de Grises y en modo RGB color.

Para la porción de imagen en **modo Blanco y Negro**, la matriz almacena el valor “0”, si el pixel correspondiente es de color negro, o el valor “1”, en caso de que éste sea de color blanco, según se muestra en la siguiente figura:

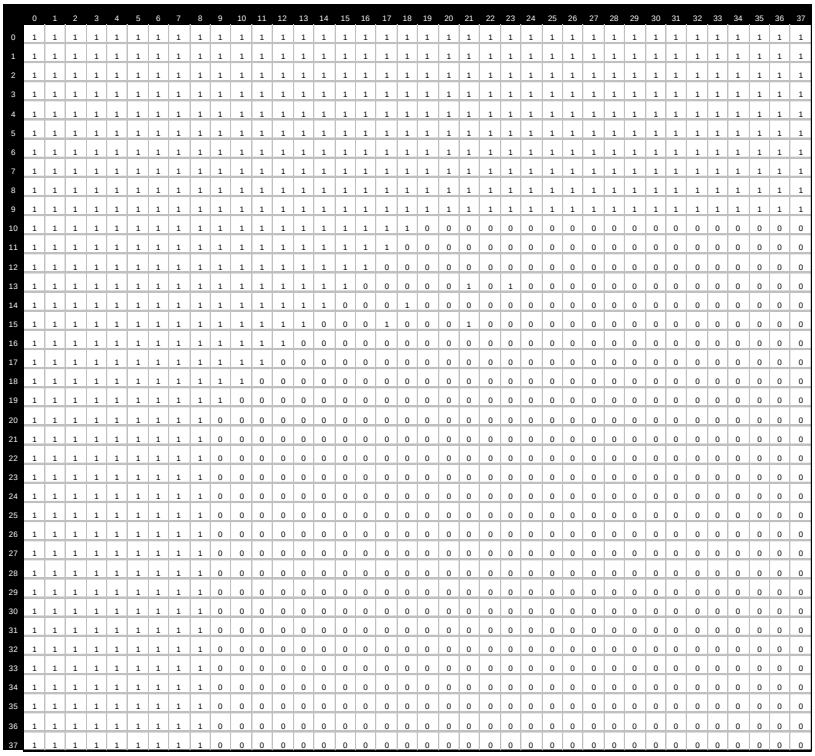


Figura 1.6: Porción de imagen de 1 cm. de lado en modo Blanco y Negro, almacenada mediante una matriz que contiene sólo unos o ceros.

En cambio, en **modo Escala de Grises**, la matriz almacena valores enteros comprendidos entre “0”, si el pixel correspondiente es de color negro, y “255”, en caso de que éste sea de color blanco.
En la siguiente figura se ha resaltado en tono gris los píxeles o celdas que corresponden al borde del objeto.

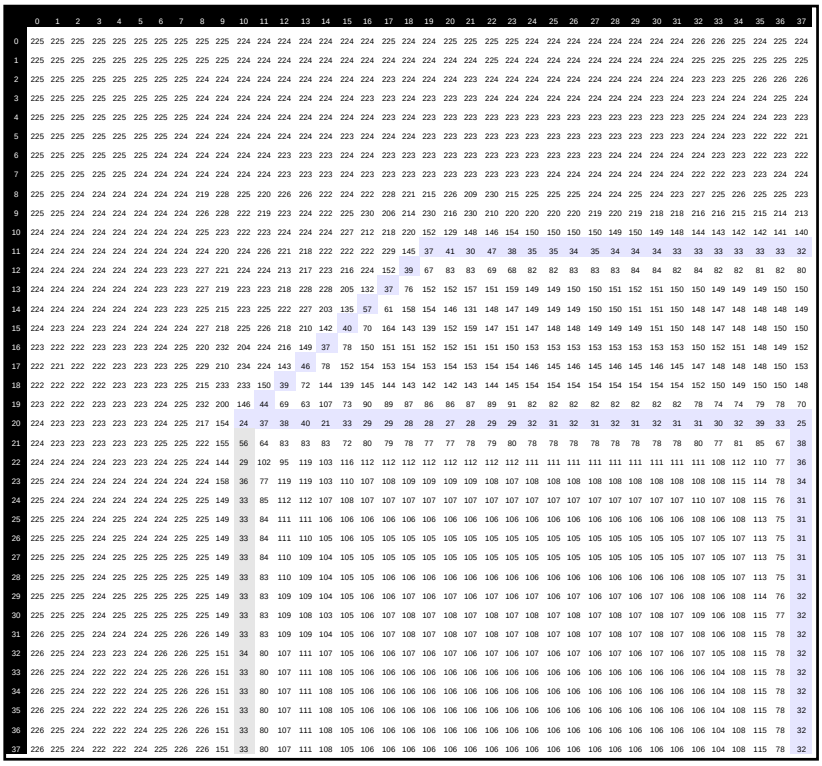


Figura 1.7: Porción de imagen de 1 cm. de lado en modo Escala de Grises, almacenada mediante una matriz que contiene valores comprendidos entre 0 y 255.

La matriz asociada al **modo RGB color** es similar a la anterior, con la diferencia de que cada elemento de la matriz contiene un vector de tres componentes (Rojo, Verde, Azul). En este caso, cada componente toma valores enteros comprendidos entre 0 y 255.

Del ejemplo anterior podemos deducir que:

Definición 1.2: Matriz imagen de una imagen digital

Una imagen digital de m x n píxeles es almacenada en el ordenador mediante una matriz ***A*** *m x n. En este caso, el coeficiente* A_{ij} *representa el color asociado al pixel ubicado en la fila "i" y columna "j" para* $1 \leq i \leq m$, $1 \leq j \leq n$. *Según el modo con que se confeccione la imagen, dicho coeficiente puede ser un número, una terna o bien una cuaterna ordenada de enteros comprendidos entre 0 y 255.*

La matriz ***A*** *recibe el nombre de* ***matriz imagen*** *o* ***matriz asociada a la imagen digital****.*

Segmentación de imágenes digitales

En numerosas ocasiones se necesita descomponer una imagen digital en subconjuntos o regiones a los efectos de que el ordenador pueda identificar a los objetos que componen dicha imagen. Esto se puede lograr mediante el proceso de "segmentación", cuya definición es la siguiente:

Definición 1.3: Segmentación de una imagen digital

La segmentación de una imagen digital S es el proceso que consiste en descomponer dicha imagen en subconjuntos o regiones disjuntas dos a dos. En este caso, la imagen segmentada S se compone de una colección de subconjuntos no vacíos $S_1, S_2, \ldots, S_m$ *de forma tal que :*

$$S = \bigcup_{i=1}^{m} S_i \quad con \quad S_i \cap S_j = \varnothing \quad si \quad i \neq j$$

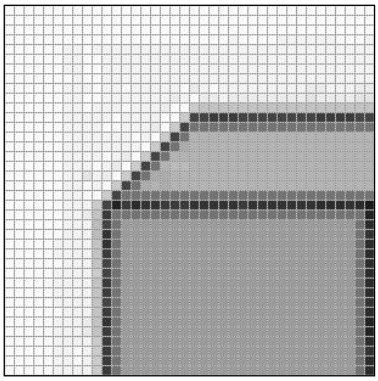
Un caso especial e importante de segmentación ocurre cuando m = 2, es decir,

$$S = S_1 \cup S_2 \quad \text{con} \quad S_1 \cap S_2 = \varnothing$$

De este modo, S se descompone en un conjunto S_1 y su complemento. Esta situación se aplica, por ejemplo, cuando de una imagen se desea separar el objeto principal (representado por el conjunto S_1) de su complemento o fondo (representado por el conjunto S_2)
Uno de los métodos que se utilizan para segmentar una imagen es el "Método de Umbralización", que consiste en reasignar a cada pixel de una imagen digital, un nuevo valor según supere o no cierto umbral. Explicaremos el funcionamiento de este método mediante el siguiente ejemplo.

Ejemplo 1.5

Dada la siguiente imagen de 38 x 38 píxeles en Escala de Grises:



a) Aplicar el Método de Umbralización para separar el objeto (porción de paralelepípedo rectangular) del fondo.
b) Aplicar al objeto el Método de Umbralización a fin de separar el objeto de sus bordes.

<u>Resolución</u>

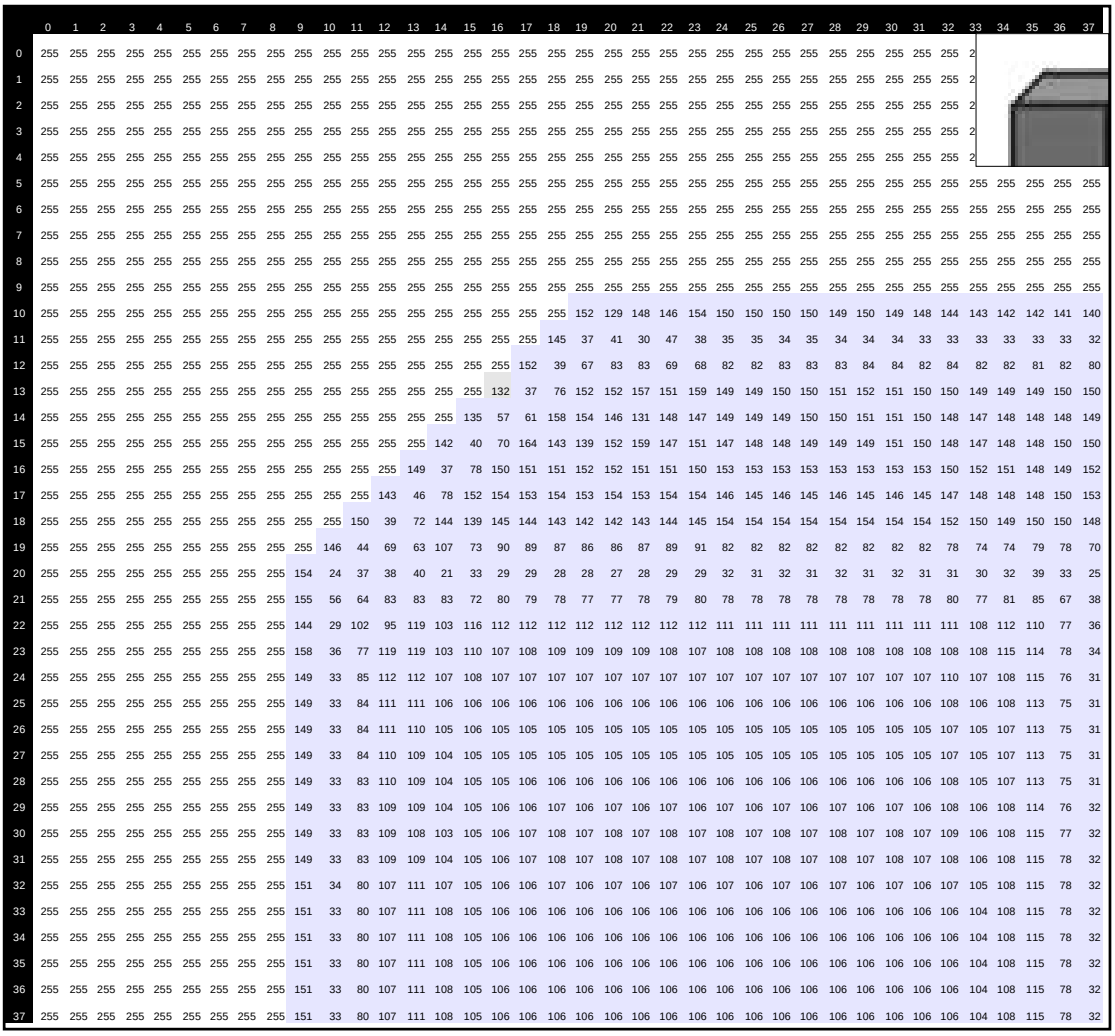
a) Aquí debemos decidir si un pixel pertenece al objeto o a su complemento. Si observamos la matriz imagen en Escala de Grises que se muestra a continuación:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
0	225	225	225	225	225	225	225	225	225	225	224	224	224	224	224	224	224	225	224	224	225	225	225	225	224	224	224	224	224	224	224	224	226	226	225	224	225	224
1	225	225	225	225	225	225	225	225	225	225	224	224	224	224	224	224	224	224	224	224	224	224	225	224	224	224	224	224	224	224	224	224	225	225	225	225	225	225
2	225	225	225	225	225	225	225	225	224	224	224	224	224	224	224	224	224	223	224	224	224	223	224	224	224	224	224	224	224	224	224	223	223	225	226	226	226	226
3	225	225	225	225	225	225	225	225	224	224	224	224	224	224	224	224	223	223	224	223	223	223	224	224	224	224	224	224	224	224	223	224	223	224	224	224	225	224
4	225	225	225	225	225	225	225	225	224	224	224	224	224	224	224	224	224	223	224	223	223	223	223	223	223	223	223	223	223	223	223	223	225	224	224	224	223	223
5	225	225	225	225	225	225	225	224	224	224	224	224	224	224	224	223	224	224	224	223	223	223	223	223	223	223	223	223	223	223	223	223	224	224	223	222	222	221
6	225	225	225	225	225	225	224	224	224	224	224	224	223	223	223	224	224	223	223	223	223	223	223	223	223	223	223	223	224	224	224	224	224	223	223	222	223	222
7	225	225	225	225	225	224	224	224	224	224	224	224	223	223	223	224	223	223	223	223	223	223	223	223	223	224	224	224	224	224	224	224	222	222	223	223	224	224
8	225	225	224	224	224	224	224	224	219	228	225	220	226	226	222	224	222	228	221	215	226	209	230	215	225	225	225	224	224	225	224	223	227	225	226	225	225	223
9	225	225	224	224	224	224	224	224	226	228	222	219	223	224	222	225	230	206	214	230	216	230	210	220	220	220	220	219	220	219	218	218	216	216	215	215	214	213
10	224	224	224	224	224	224	224	224	225	223	222	223	224	224	224	227	212	218	220	152	129	148	146	154	150	150	150	150	149	150	149	148	144	143	142	142	141	140
11	224	224	224	224	224	224	224	224	224	220	224	226	221	218	222	222	222	229	145	37	41	30	47	38	35	35	34	35	34	34	34	33	33	33	33	33	33	32
12	224	224	224	224	224	224	223	223	227	221	224	224	213	217	223	216	224	152	39	67	83	83	69	68	82	82	83	83	83	84	84	82	84	82	82	81	82	80
13	224	224	224	224	224	224	223	223	227	219	223	223	218	228	228	205	132	37	76	152	152	157	151	159	149	149	150	150	151	152	151	150	150	149	149	149	150	150
14	224	224	224	224	224	224	223	223	225	215	223	225	222	227	203	135	57	61	158	154	146	131	148	147	149	149	149	150	150	151	151	150	148	147	148	148	148	149
15	224	223	224	223	224	224	224	224	227	218	225	226	218	210	142	40	70	164	143	139	152	159	147	151	147	148	148	149	149	149	151	150	148	147	148	148	150	150
16	223	222	222	223	223	223	224	225	220	232	204	224	216	149	37	78	150	151	151	152	152	151	151	150	153	153	153	153	153	153	153	153	150	152	151	148	149	152
17	222	221	222	222	223	223	223	225	229	210	234	224	143	46	78	152	154	153	154	153	154	153	154	154	146	145	146	145	146	145	146	145	147	148	148	148	150	153
18	222	222	222	222	223	223	223	225	215	233	233	150	39	72	144	139	145	144	143	142	142	143	144	145	154	154	154	154	154	154	154	152	152	150	149	150	150	148
19	223	222	222	223	223	223	224	225	232	200	146	44	69	63	107	73	90	89	87	86	86	87	89	91	82	82	82	82	82	82	82	82	78	74	74	79	78	70
20	224	223	223	223	223	223	224	225	217	154	24	37	38	40	21	33	29	29	28	28	27	28	29	29	32	31	32	31	32	31	32	31	31	30	32	39	33	25
21	224	223	223	223	223	223	225	225	222	155	56	64	83	83	83	72	80	79	78	77	77	78	79	80	78	78	78	78	78	78	78	78	80	77	81	85	67	38
22	224	224	224	224	223	223	224	225	224	144	29	102	95	119	103	116	112	112	112	112	112	112	112	112	111	111	111	111	111	111	111	111	111	108	112	110	77	36
23	225	224	224	224	224	224	224	224	224	158	36	77	119	119	103	110	107	108	109	109	109	109	108	107	108	108	108	108	108	108	108	108	108	108	115	114	78	34
24	225	224	224	224	224	224	224	225	225	149	33	85	112	112	107	108	107	107	107	107	107	107	107	107	107	107	107	107	107	107	107	110	107	108	115	76	31
25	225	225	224	224	225	224	224	225	225	149	33	84	111	111	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	108	106	108	113	75	31
26	225	225	225	224	225	224	224	225	225	149	33	84	111	110	105	106	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	107	105	107	113	75	31
27	225	225	225	224	225	225	225	225	225	149	33	84	110	109	104	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	107	105	107	113	75	31
28	225	225	225	224	225	225	225	225	225	149	33	83	110	109	104	105	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	108	105	107	113	75	31
29	225	225	225	224	225	225	225	225	225	149	33	83	109	109	104	105	106	106	107	106	107	106	107	106	107	106	107	106	107	106	107	106	108	106	108	114	76	32
30	225	225	225	224	225	225	225	225	225	149	33	83	109	108	103	105	106	107	108	107	108	107	108	107	108	107	108	107	108	107	108	107	109	106	108	115	77	32
31	226	225	225	224	224	224	225	226	226	149	33	83	109	109	104	105	106	107	108	107	108	107	108	107	108	107	108	107	108	107	108	107	108	106	108	115	78	32
32	226	225	224	223	223	224	226	226	225	151	34	80	107	111	107	105	106	106	107	106	107	106	107	106	107	106	107	106	107	106	107	105	108	106	108	115	78	32
33	226	225	224	222	222	224	225	226	226	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	108	115	78	32
34	226	225	224	222	222	224	225	226	226	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	108	115	78	32
35	226	225	224	222	222	224	225	226	226	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	108	115	78	32
36	226	225	224	222	222	224	225	226	226	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	108	115	78	32
37	226	225	224	222	222	224	225	226	226	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	108	115	78	32

Figura 1.8: Imagen almacenada mediante una matriz que contiene valores comprendidos entre 0 y 255.

Podemos comprobar que la mayoría de los píxeles que no pertenecen al objeto tienen un valor superior a 200. Por lo tanto, tomaremos al valor 200 como umbral que permita descomponer la imagen en dos partes: el objeto y su complemento.

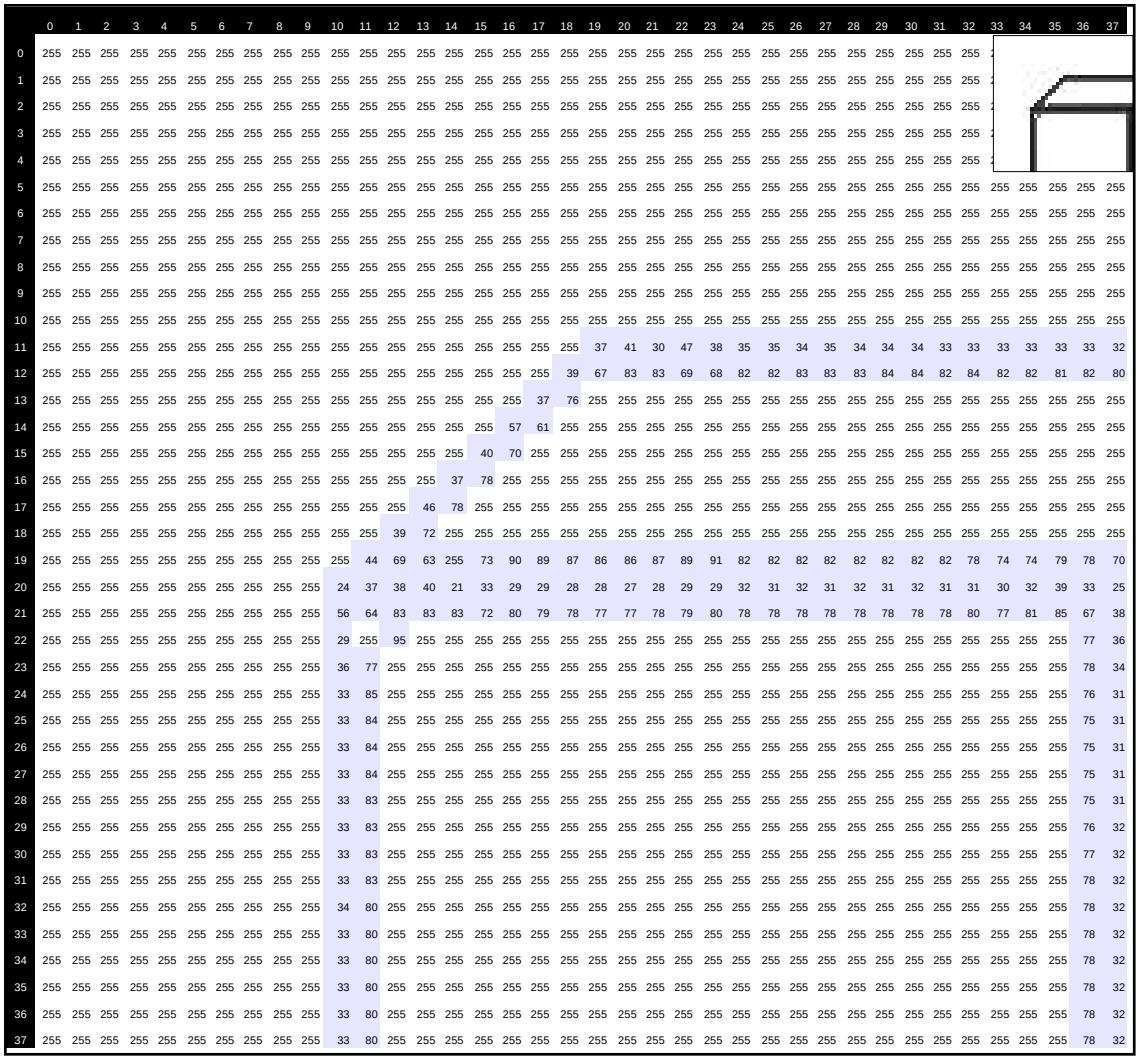
Mediante un programa informático, se asigna el valor del color blanco (255) a los píxeles cuyo valor superen 200. De este modo, con esta nueva matriz imagen, los píxeles que contengan el valor 255 pertenecerán al fondo, mientras que los restantes pertenecerán al objeto en sí.
La siguiente figura muestra la nueva matriz, que corresponde a la imagen segmentada:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2				
1	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2				
2	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2				
3	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2				
4	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2				
5	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
6	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
7	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
10	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	152	129	148	146	154	150	150	150	150	149	150	149	148	144	143	142	142	141	140
11	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	145	37	41	30	47	38	35	35	34	35	34	34	34	33	33	33	33	33	33	32
12	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	152	39	67	83	83	69	68	82	82	83	83	83	84	84	82	84	82	82	81	82	80
13	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	132	37	76	152	152	157	151	159	149	149	150	150	151	152	151	150	150	149	149	149	150	150
14	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	135	57	61	158	154	146	131	148	147	149	149	149	150	150	151	151	150	148	147	148	148	148	149
15	255	255	255	255	255	255	255	255	255	255	255	255	255	255	142	40	70	164	143	139	152	159	147	151	147	148	148	149	149	149	151	150	148	147	148	148	150	150
16	255	255	255	255	255	255	255	255	255	255	255	255	255	149	37	78	150	151	151	152	152	151	151	150	153	153	153	153	153	153	153	153	150	152	151	148	149	152
17	255	255	255	255	255	255	255	255	255	255	255	255	143	46	78	152	154	153	154	153	154	153	154	154	146	145	146	145	146	145	146	145	147	148	148	148	150	153
18	255	255	255	255	255	255	255	255	255	255	255	150	39	72	144	139	145	144	143	142	142	143	144	145	154	154	154	154	154	154	154	154	152	150	149	150	150	148
19	255	255	255	255	255	255	255	255	255	255	146	44	69	63	107	73	90	89	87	86	86	87	89	91	82	82	82	82	82	82	82	78	74	74	79	78	70	
20	255	255	255	255	255	255	255	255	255	154	24	37	38	40	21	33	29	29	28	28	27	28	29	29	32	31	32	31	32	31	32	31	31	30	32	39	33	25
21	255	255	255	255	255	255	255	255	255	155	56	64	83	83	83	72	80	79	78	77	77	78	79	80	78	78	78	78	78	78	78	78	80	77	81	85	67	38
22	255	255	255	255	255	255	255	255	255	144	29	102	95	119	103	116	112	112	112	112	112	112	112	112	111	111	111	111	111	111	111	111	111	108	112	110	77	36
23	255	255	255	255	255	255	255	255	255	158	36	77	119	119	103	110	107	108	109	109	109	109	108	107	108	108	108	108	108	108	108	108	108	108	115	114	78	34
24	255	255	255	255	255	255	255	255	255	149	33	85	112	112	107	108	107	107	107	107	107	107	107	107	107	107	107	107	107	107	107	107	110	107	108	115	76	31
25	255	255	255	255	255	255	255	255	255	149	33	84	111	111	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	108	106	108	113	75	31
26	255	255	255	255	255	255	255	255	255	149	33	84	111	110	105	106	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	107	105	107	113	75	31
27	255	255	255	255	255	255	255	255	255	149	33	84	110	109	104	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	105	107	105	107	113	75	31
28	255	255	255	255	255	255	255	255	255	149	33	83	110	109	104	105	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	108	105	107	113	75	31
29	255	255	255	255	255	255	255	255	255	149	33	83	109	109	104	105	106	106	107	106	107	106	107	106	107	106	107	106	107	106	107	106	108	106	108	114	76	32
30	255	255	255	255	255	255	255	255	255	149	33	83	109	108	103	105	106	107	108	107	108	107	108	107	108	107	108	107	108	107	108	107	109	106	108	115	77	32
31	255	255	255	255	255	255	255	255	255	149	33	83	109	109	104	105	106	107	108	107	108	107	108	107	108	107	108	107	108	107	108	107	108	106	108	115	78	32
32	255	255	255	255	255	255	255	255	255	151	34	80	107	111	107	105	106	106	107	106	107	106	107	106	107	106	107	106	107	106	107	106	107	105	108	115	78	32
33	255	255	255	255	255	255	255	255	255	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	115	78	32
34	255	255	255	255	255	255	255	255	255	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	115	78	32
35	255	255	255	255	255	255	255	255	255	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	115	78	32
36	255	255	255	255	255	255	255	255	255	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	115	78	32
37	255	255	255	255	255	255	255	255	255	151	33	80	107	111	108	105	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	106	104	108	115	78	32

Figura 1.9: Matriz asociada a la imagen segmentada, obtenida al separar el objeto de su fondo.

b) Luego de separar al objeto del fondo de la imagen, se puede lograr la separación del objeto de sus bordes. Observando la matriz imagen segmentada del apartado anterior, se puede comprobar que los píxeles que pertenecen al borde del objeto tienen un valor inferior a 100. Entonces, podemos tomar al valor 100 como umbral que permita separar al objeto de su borde. Luego, se asigna el valor del color blanco (255) a los píxeles que no pertenecen al borde del objeto, es decir, a aquellos cuyo valor superen 100.
La matriz resultante de la nueva imagen segmentada se muestra a continuación.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255					
1	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255					
2	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255					
3	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255					
4	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255					
5	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
6	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
7	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
10	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
11	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	37	41	30	47	38	35	35	34	35	34	34	34	33	33	33	33	33	33	32
12	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	39	67	83	83	69	68	82	82	83	83	83	84	84	82	84	82	82	81	82	80	
13	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	37	76	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
14	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	57	61	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
15	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	40	70	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
16	255	255	255	255	255	255	255	255	255	255	255	255	255	255	37	78	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
17	255	255	255	255	255	255	255	255	255	255	255	255	255	46	78	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
18	255	255	255	255	255	255	255	255	255	255	255	255	39	72	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
19	255	255	255	255	255	255	255	255	255	255	255	44	69	63	255	73	90	89	87	86	86	87	89	91	82	82	82	82	82	82	82	82	78	74	74	79	78	70
20	255	255	255	255	255	255	255	255	255	255	24	37	38	40	21	33	29	29	28	28	27	28	29	29	32	31	32	31	32	31	32	31	31	30	32	39	33	25
21	255	255	255	255	255	255	255	255	255	255	56	64	83	83	83	72	80	79	78	77	77	78	79	80	78	78	78	78	78	78	78	80	77	81	85	67	38	
22	255	255	255	255	255	255	255	255	255	255	29	255	95	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	77	36
23	255	255	255	255	255	255	255	255	255	255	36	77	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	34
24	255	255	255	255	255	255	255	255	255	255	33	85	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	76	31
25	255	255	255	255	255	255	255	255	255	255	33	84	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	75	31
26	255	255	255	255	255	255	255	255	255	255	33	84	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	75	31
27	255	255	255	255	255	255	255	255	255	255	33	84	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	75	31
28	255	255	255	255	255	255	255	255	255	255	33	83	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	75	31
29	255	255	255	255	255	255	255	255	255	255	33	83	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	76	32
30	255	255	255	255	255	255	255	255	255	255	33	83	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	77	32
31	255	255	255	255	255	255	255	255	255	255	33	83	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	32
32	255	255	255	255	255	255	255	255	255	255	34	80	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	32
33	255	255	255	255	255	255	255	255	255	255	33	80	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	32
34	255	255	255	255	255	255	255	255	255	255	33	80	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	32
35	255	255	255	255	255	255	255	255	255	255	33	80	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	32
36	255	255	255	255	255	255	255	255	255	255	33	80	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	32
37	255	255	255	255	255	255	255	255	255	255	33	80	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	78	32

Figura 1.10: Matriz asociada a la imagen nuevamente segmentada, obtenida al separar el objeto de su borde.

De este modo, los píxeles que tienen valor 255 pertenecen al fondo o al interior del objeto, mientras que los restantes pertenecen a los bordes del objeto.

En el siguiente capítulo introduciremos las nociones básicas de Topología Digital, las cuales serán necesarias para dotar a una imagen digital de una estructura que permita su estudio a través de conceptos topológicos como frontera, conectividad, adyacencia o vecindad.

Capítulo 2

Nociones de Topología Digital

Capítulo 2: Nociones de Topología Digital

El procesamiento digital de imágenes [9] es una disciplina de rápido crecimiento con múltiples aplicaciones en negocios (lectura de documentos), industrias (ensamblado automático y control), medicina (radiología, hematología), ciencias naturales como meteorología, geología y muchos campos más [10].
Consiste en analizar una imagen digital utilizando la Matemática y la Tecnología Informática: dada una imagen, se trata de identificar los objetos que la componen, estudiar sus propiedades y relaciones existentes entre ellos. Por ejemplo, una página impresa consta de caracteres sobre un fondo; una mancha de sangre en un microscopio se compone de células de sangre sobre un fondo; una placa de tórax muestra el corazón, pulmones, costillas; una imagen satelital de la Tierra está constituida por distintas tonalidades que representan diversas regiones del planeta.
Para identificar los objetos que forman una imagen es necesario descomponer a la misma en regiones o subconjuntos, separándolos del fondo. Como vimos en el capitulo anterior, este proceso recibe el nombre de “segmentación” y uno de los métodos que se aplican para lograrlo es el “Método de Umbralización”.
Una vez que la imagen ha sido segmentada en subconjuntos, el paso siguiente consiste en establecer propiedades y relaciones entre estos. Algunas propiedades geométricas pueden obtenerse fácilmente. Por ejemplo, si la imagen digital segmentada es en Escala de Grises y cada subconjunto tiene una tonalidad de gris diferente, el área aproximada de un subconjunto se calcula sumando las áreas de los píxeles que componen dicho subconjunto.

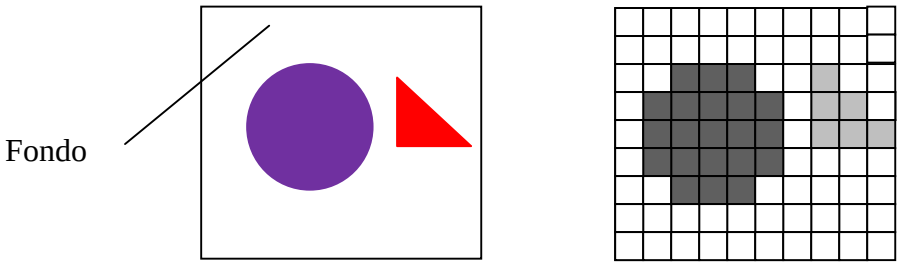


Figura 2.1: (a) Imagen analógica que contiene un círculo y un triángulo. (b) Imagen digitalizada y segmentada, en Escala de Grises .Ésta ha sido descompuesta en 3 subconjuntos: círculo, triángulo y fondo. Observar que el círculo consta de 21 píxeles mientras que el triángulo está formado por 6 píxeles. En este caso, las áreas aproximadas de cada objeto se calculan así:
Para el círculo: 21 d^2. Para el triángulo: 6 d^2, donde d = longitud del lado de un píxel.

Sin embargo, otras propiedades son netamente topológicas, y por esta razón se necesitan ciertos conceptos como “vecindad”, “adyacencia”, “frontera” o “conectividad”.
Las propiedades topológicas de un subconjunto de una imagen digital son útiles por varias razones ya que, una vez que el subconjunto fue identificado, suele suceder que se necesite segmentarlo en regiones conexas. Por ejemplo, si la imagen digitalizada de una página impresa ha sido segmentada en dos subconjuntos: texto y fondo, en ocasiones se necesita identificar cada carácter que compone el texto. Esto se logra segmentando el texto en subconjuntos formados por sus componentes conexas.

(a) *(b)*

Figura 2.2: (a) Imagen analógica de una página impresa. (b) Imagen digitalizada y segmentada, en Escala Blanco y Negro. Ésta ha sido descompuesta en 2 subconjuntos: texto y fondo. Segmentando el texto en sus componentes conexas, podremos identificar a cada una de las letras.

Puede ocurrir también que se precise recorrer la frontera de esos subconjuntos o regiones, ya que esto proporciona una codificación compacta de la forma de la región. O bien, puede que se necesite "reducir" las regiones a "esqueletos", sin cambiar sus propiedades de conectividad, ya que esto también produce representaciones compactas dentro del ordenador.
Existen numerosos algoritmos que permiten segmentar un subconjunto en sus componentes conexas, recorrer su frontera o reducirlo.
Sin embargo, para verificar que estos algoritmos funcionen correctamente, es necesario analizar algunas propiedades topológicas del subconjunto. Este es el motivo por el cual surge la Topología Digital, definición que damos a continuación:

Definición 2.1: Topología Digital

La Topología Digital es la disciplina de la Matemática creada para estudiar las propiedades topológicas de una imagen digital.

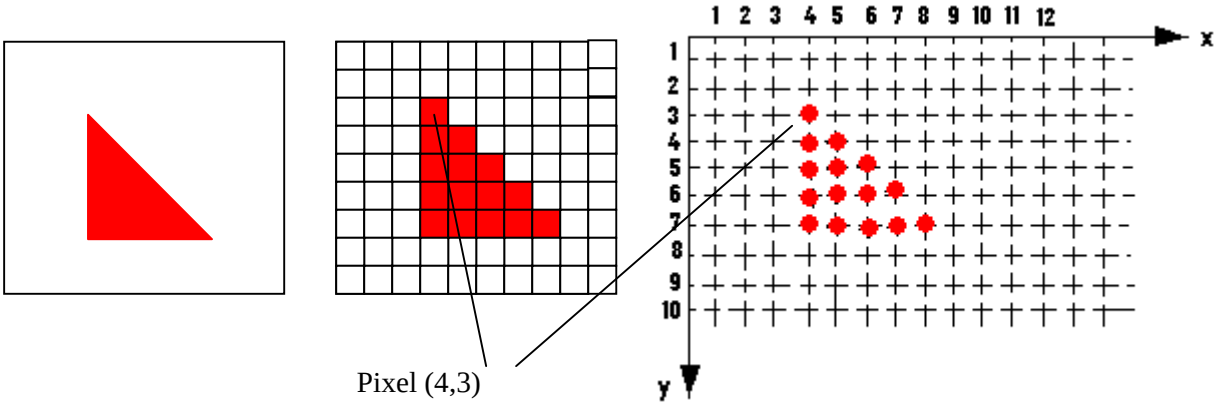
El plano digital y su topología

Cuando se estudia topológicamente una imagen digital, no se precisa saber cuál es el color asociado a cada pixel. Por esta razón identificaremos a una imagen digital, que es una función *f:* $A \subset \mathbb{N} \times \mathbb{N} \to \mathbb{Z}^n$, con su dominio A.
De este modo, la imagen digital será visualizada como un subconjunto finito de $\mathbb{N} \times \mathbb{N}$ o mejor aún, de $\mathbb{Z} \times \mathbb{Z}$. A este último conjunto lo llamaremos “el plano digital”.

Definición 2.2: El plano digital

El plano digital $\mathbb{Z}^2$ *es el conjunto formado por los pares ordenados de números enteros. Es decir,*

$$\mathbb{Z}^2 = \{ (m, n) : m , n \in \mathbb{Z} \}$$



(a) *(b)* *(c)*

Figura 2.3: (a) Imagen analógica de un triángulo. (b) Imagen digitalizada del triángulo. (c) Representación de la imagen digital en el plano digital.

Así, a una imagen digital la consideraremos inmersa en el plano digital $\mathbb{Z}^2$ y, para estudiar sus propiedades topológicas, deberemos dotar a $\mathbb{Z}^2$ de una topología. Antes de continuar, introduciremos las siguientes notaciones.

Notaciones básicas

■ *Si* $P = (p_1,p_2)$, $Q = (q_1,q_2) \in \mathbb{Z}^2$ la distancia euclídea entre P y Q es

$$d(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

■ *Si* $P = (p_1,p_2) \in \mathbb{R}^2$ y $\varepsilon > 0$, el disco de centro P y radio ε es

$$B(P, \varepsilon) = \{X = (x, y) \in \mathbb{R}^2 : d(P, X) < \varepsilon \}$$

■ Si A es un subconjunto de $\mathbb{R}^2$ entonces

$\overline{A}$ = complemento de A

Int(A) = interior de A.

Fr(A) = frontera de A.

Estamos "tentados" en asignar a $\mathbb{Z}^2$ la topología relativa a la usual de $\mathbb{R}^2$. Pero la siguiente proposición nos muestra que esta topología carece de interés, como veremos a continuación.

Proposición 2.1

Sea $\tau_{\mathbb{R}^2}$ la topología usual de $\mathbb{R}^2$ y $\tau_{\mathbb{Z}^2}$ la topología relativa de $\mathbb{Z}^2$. Entonces

a) $\tau_{\mathbb{Z}^2}$ es la topología discreta de $\mathbb{Z}^2$, es decir, $\tau_{\mathbb{Z}^2} = \{ A : A \subseteq \mathbb{Z}^2 \}$

b) Si $A \subseteq \mathbb{Z}^2$ entonces $Fr(A) = \varnothing$.

Demostración

a) Sea $P \in \mathbb{Z}^2$ y $0 < \varepsilon < ½$. Entonces $\{P\} = B(P, \varepsilon) \cap \mathbb{Z}^2$. Luego, todo punto del plano digital es abierto en $\mathbb{Z}^2$, con lo cual cualquier subconjunto $A \subseteq \mathbb{Z}^2$ es también abierto, pues $A = \bigcup_{P \in A} \{P\}$.

b) Del inciso a) sabemos que todo subconjunto de $\mathbb{Z}^2$ es abierto en $\mathbb{Z}^2$.
Luego, si $A \subseteq \mathbb{Z}^2$ se tiene $Int(A) = A$, $Int(\overline{A}) = \overline{A}$, de donde resulta

$Fr(A) = \overline{Int(A)} \cap \overline{Int(\overline{A})} = \overline{A} \cap \overline{\overline{A}} = \overline{A} \cap A = \varnothing$. ■

Como podemos comprobar, no es conveniente asignar a $\mathbb{Z}^2$ la topología relativa, pues resultaría imposible identificar y recorrer la frontera de un conjunto ya que ésta es siempre vacía. Para solucionar esto, D. Marcus y F. Wyse [4] introdujeron una nueva topología, llamada la ***Topología de Marcus-Wyse***, que es la que utilizaremos en este trabajo. Para definirla, necesitaremos previamente los siguientes conceptos:

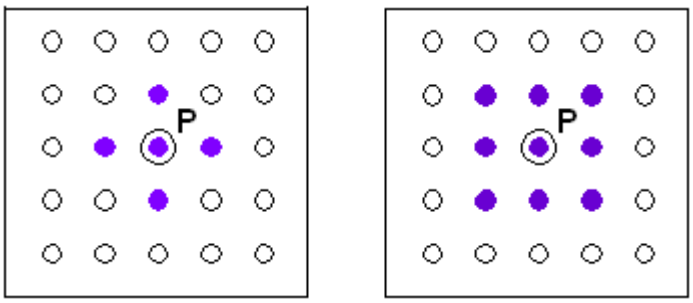
Definición 2.3: Vecinos 4N y 8N de un punto.

Dado un punto $P = (x, y) \in \mathbb{Z}^2$ entonces

a) Los vecinos 4N de P son (x, y) , $(x \pm 1, y)$ y $(x, y \pm 1)$.

b) Los vecinos 8N de P son (x, y) , $(x \pm 1, y)$, $(x, y \pm 1)$,$(x+1, y \pm 1)$ y $(x-1, y \pm 1)$.

c) Si $Q \in \mathbb{Z}^2$ diremos que Q ***es adyacente 4N (u 8N) a P*** si P es vecino 4N (u 8N) de Q.



(a) *(b)*

Figura 2.4: (a) Vecinos 4N de P. (b) Vecinos 8N de P.

Fácilmente se comprueba que la relación *"es vecino de"* es reflexiva, simétrica pero no es transitiva.
El conjunto de puntos que son vecinos 4N y 8N de un punto $P \in \mathbb{Z}^2$ recibe el nombre de "vecindad 4N" y "vecindad 8N" respectivamente. Podríamos definir este concepto en términos de la distancia euclídea, como sigue:

Definición 2.4: Vecindad 4N y 8N de un punto.

Dado un punto $P = (x, y) \in \mathbb{Z}^2$ *entonces*

a) La vecindad 4N de P es $\{Q \in \mathbb{Z}^2 : d(P,Q) \leq 1\}$.

b) La vecindad 8N de P es $\{Q \in \mathbb{Z}^2 : d(P,Q) \leq \sqrt{2}\}$.

Observemos que en la topología de $\mathbb{Z}^2$ relativa a la usual del plano, tanto el punto P como el conjunto formado por sus vecinos 4N, son los abiertos más pequeños que contienen a P.

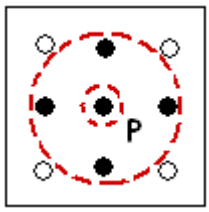


Figura 2.5: Los abiertos más pequeños que contienen a P, en la topología relativa de $\mathbb{Z}^2$.

Quizás este hecho haya inspirado a D. Marcus y F. Wyse a definir una topología para $\mathbb{Z}^2$ en términos de una base que contenga a esos conjuntos. Dicha base es la siguiente:

Definición 2.5: Base para una topología en $\mathbb{Z}^2$

Sea $P = (x, y) \in \mathbb{Z}^2$ *y U(P) el conjunto definido por*

$$U(P) = \begin{cases} \{P\} & \text{si } x + y \text{ es impar} \\ \{Q : Q \text{ es vecino 4N de } P\} & \text{si } x + y \text{ es par} \end{cases}$$

El conjunto $\mathbf{B} = \{U(P) : P \in \mathbb{Z}^2\}$ *recibe el nombre de* <u>*Base para una topología de*</u> $\mathbb{Z}^2$

Ahora bien, debemos probar que **B** es efectivamente una base, y para ello hay que verificar las siguientes condiciones:
1. $\forall\ (x, y) \in \mathbb{Z}^2\ \exists\ W \in \mathbf{B} : (x, y) \in W$.
2. Si $(x, y) \in W_1 \cap W_2$ con $W_1, W_2 \in \mathbf{B} \Rightarrow \exists\ W_3 \in \mathbf{B} : (x, y) \in W_3 \subset W_1 \cap W_2$.

La siguiente proposición lo demuestra:

Proposición 2.2

El conjunto $\mathbf{B}$ es una base para una topología de $\mathbb{Z}^2$.

Demostración

1. Sea $P = (x, y) \in \mathbb{Z}^2$ entonces tomamos $W = U(P)$.
De este modo, $W \in \mathbf{B} \wedge (x, y) \in W$.

2. Sean $W_1, W_2 \in \mathbf{B} \wedge (x, y) \in W_1 \bigcap W_2$.

Para evitar casos triviales, supondremos que $W_1 \neq W_2$.
Entonces $W_1 = U(P_1)$ y $W_2 = U(P_2)$ con $P_1 = (x_1,y_1) \neq P_2 = (x_2,y_2)$.
Se presentan los siguientes casos:

a) Si $x_1 + y_1 \wedge x_2 + y_2$ son impares, entonces
$W_1 = \{P_1\}$, $W_2 = \{P_2\}$. Luego $W_1 \bigcap W_2 = \varnothing$, con lo cual no hay nada que probar.

b) Si $x_1 + y_1$ es impar $\wedge$ $x_2 + y_2$ es par, entonces
$W_1 = \{P_1\}$, $W_2 = \{P_2 , (x_2 \pm 1, y_2) , (x_2, y_2 \pm 1)\}$.

b.1) Si P_2 es vecino 4N de P_1 entonces $W_1 \bigcap W_2 = \{P_1\}$; tomando $W_3 = W_1 \bigcap W_2 = \{P_1\}$ se verifica que $(x, y) = P_1 \in W_3 \subset W_1 \bigcap W_2$.

b.2) Si P_2 no es vecino 4N de P_1 entonces $W_1 \bigcap W_2 = \varnothing$, con lo cual no hay nada que demostrar.

c) Si $x_1 + y_1 \wedge x_2 + y_2$ son pares, entonces

En este caso $P_2 \notin W_1$ pues si estuviera en dicho conjunto, $P_2 = (x_1 \pm 1, y_1)$ ó $P_2 = (x_1, y_1 \pm 1)$ lo cual es absurdo pues la suma de las componentes de P_2 es par.
De manera similar se prueba que $P_1 \notin W_2$. Por lo tanto,
$W_1 \bigcap W_2 \subset \{(x_1 \pm 1, y_1), (x_1, y_1 \pm 1), (x_2 \pm 1, y_2), (x_2, y_2 \pm 1)\}$. Los puntos de este conjunto verifican que la suma de sus componentes es impar.
Luego, si $(x, y) \in W_1 \bigcap W_2$, tomamos $W_3 = \{(x, y)\}$, de donde se obtiene que $(x, y) \in W_3 \subset W_1 \bigcap W_2$. ■

Ahora sí estamos en condiciones de presentar la ***Topología de Marcus-Wyse*** para $\mathbb{Z}^2$. La llamaremos "***la topología digital de*** $\mathbb{Z}^2$ y es la siguiente:

Definición 2.6: La topología digital τ ***de*** $\mathbb{Z}^2$

La topología digital τ *de* $\mathbb{Z}^2$ *es la topología generada por la base*

$$\mathbf{B} = \left\{ U(P) : P \in \mathbb{Z}^2 \right\}$$

De este modo, los miembros de τ *se escriben como unión de elementos de* $\mathbf{B}$.

Observación:

Cabe preguntarnos porqué los elementos de la base **B**, es decir, los conjuntos U(P) con $P \in \mathbb{Z}^2$ se definen de manera distinta, según sea par o impar la suma de las componentes del punto P. ¿No podríamos definir a U(P) como el conjunto formado por el punto P o bien como la vecindad 4N de P? Si lo hiciéramos de este modo, esto no resultaría conveniente pues:

Si U(P) = {P} $\forall P \in \mathbb{Z}^2$ entonces cualquier punto $P \in \mathbb{Z}^2$ es abierto.

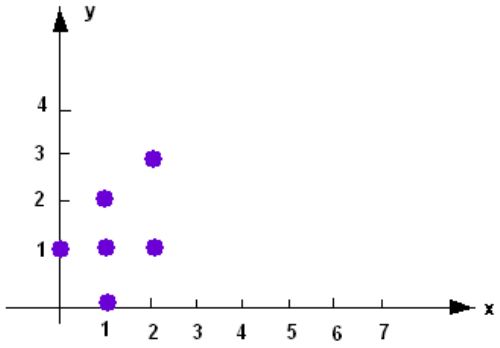
Por otro lado, si U(P) = { Q : Q es vecino 4N de P} $\forall P \in \mathbb{Z}^2$ entonces cualquier punto $P = (x, y) \in \mathbb{Z}^2$ es también es abierto pues {P}= U(P) $\bigcap$ U(P_1) $\bigcap$ U(P_2) donde $P_1 = (x+1, y)$, $P_2 = (x-1, y)$.

En ambos casos la topología digital τ sería la topología discreta que, como vimos, carece de interés.

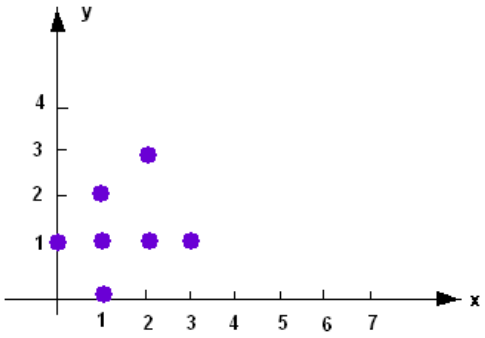
En los siguientes ejemplos se mostrarán algunos conjuntos abiertos de la topología digital y, además, se ilustrará cómo calcular el interior y la frontera de los mismos.

Ejemplo 2.1

El conjunto S = { (0,1), (1,0), (1,1) , (1, 2) , (2,1) , (2,3) } es abierto en la topología digital τ pues S = U(1,1) $\cup$ U(2,3). Su representación en el plano digital es la siguiente:



En cambio, el conjunto T que se muestra a continuación no es abierto pues no existe $B \in$ **B** de modo tal que $(3,1) \in B \subset T$, ya que el único elemento de la base **B** que contiene a (3,1) es U(3,1).



Ejemplo 2.2

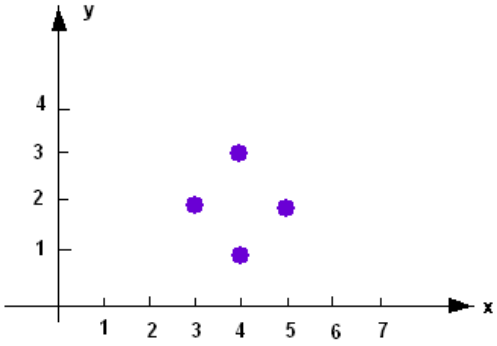
Dado el conjunto S = { (4,1), (3,2), (5,2) ,(4,3) }

a) Probar que S es abierto en la topología τ.

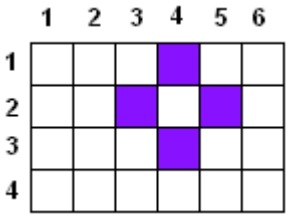
b) ¿Qué imagen digital de 4 x 6 píxeles representa el conjunto S?

Resolución

a) S es abierto pues S = U (4,1) ⋃ U(3,2) ⋃U(5,2) ⋃U(4,3). Su representación en el plano digital es



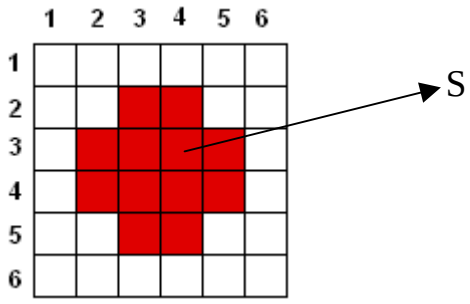
b) La imagen digital de 4 x 6 píxeles que representa el conjunto S es



Al identificar una imagen digital con su dominio, no es posible determinar cuál es el color asociado a cada pixel.

Ejemplo 2.3

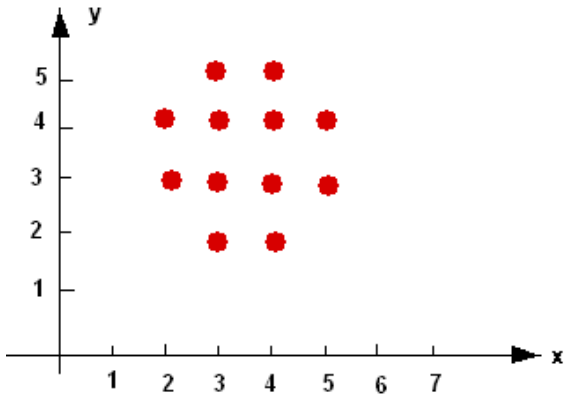
Sea S la imagen digitalizada que muestra la figura:



a) Representar S en el plano digital.
b) Hallar Int(S).
c) Calcular Fr(S).

Resolución:

a) En este caso, S = {(2,2), (3,2), (4,2), (2,3), (2,4), (3,4), (4,4)} y su representación en el plano digital es:



b) Observamos que (3,2) ∈ Int(S) porque existe B = U(3,2) ∈ **B** tal que (2,3) ∈ B ⊂ S. Por otro lado, el punto (4,2)∉ Int(S) pues no existe B∈ **B** de modo tal que (4,2) ∈ B ⊂ S, ya que el único elemento de la base **B** que contiene a (4,2) es U(4,2).

Haciendo un análisis similar para los demás puntos de S resulta:

$$Int(S) = \{(3,2), (2,3), (3,3), (4,3), (3,4), (4,4), (5,4), (4,5)\}$$

c) Recordemos que $Fr(S) = \overline{Int(S)} \cap \overline{Int\left(\overline{S}\right)}$.

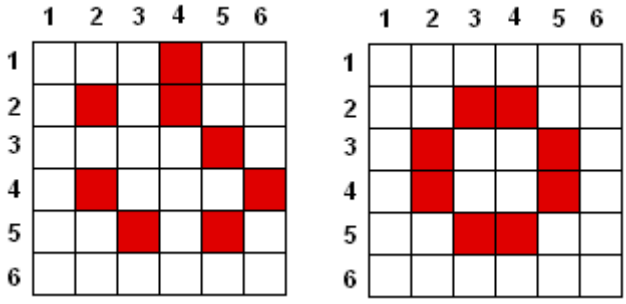
En el inciso anterior probamos que (3, 2) ∈ Int(S), con lo cual (3, 2) ∉ Fr(S) .

Por otro lado, (4,2)∉ Int(S). Pero además (4,2) ∉ $Int\left(\overline{S}\right)$ porque U(4,2) tampoco está contenido en $\overline{S}$. Por lo tanto (4,2) ∈ Fr(S). Un razonamiento similar para los demás puntos de S y sus vecinos 4N nos conduce a concluir que:

$$Fr(S) = \{(4, 2), (5,3), (2, 4), (3,5), (4, 1), (2, 2), (6, 4), (5,5)\}$$

Observación

Si dibujáramos la imagen digital asociada a Fr(S) del ejemplo anterior, obtendríamos el gráfico de la Figura 2. 6 a):



a) *b)*

Figura 2.6: a) Frontera topológica del conjunto S. b) Borde de S, según nuestra idea intuitiva.

Obviamente esto no es lo que intuitivamente esperábamos, pues debería ser "el borde" de S, según el concepto que tenemos en la topología usual de $\mathbb{R}^2$ acerca de la "frontera de un conjunto". Esta "falencia" puede ser corregida si introducimos en la topología digital una nueva definición de frontera de un conjunto, a saber:

La frontera de S es el conjunto de puntos de S que tienen vecinos 4N en $\overline{S}$.

Más adelante, en el Capítulo 4, analizaremos con más profundidad este concepto. En el siguiente Capítulo describiremos las propiedades topológicas de una imagen digital.

Capítulo 3

Propiedades topológicas de una imagen digital

Capítulo 3: Propiedades topológicas de una imagen digital

Como mencionamos en el capítulo anterior, para procesar digitalmente una imagen es preciso segmentarla; de este modo se podrán identificar a los objetos o subconjuntos que componen dicha imagen, separándolos de su fondo.
El paso siguiente consiste en analizar las propiedades topológicas de dichos subconjuntos, y por esta razón consideramos a la imagen digital inmersa en $\mathbb{Z}^2$, a quien le asignamos la topología digital τ.
En este capítulo describiremos algunas de estas propiedades topológicas, que serán necesarias para justificar "el algoritmo BF4/BF8", el cual permitirá recorrer la frontera de un objeto perteneciente a una imagen digital.
En general, las imágenes digitales provienen de imágenes analógicas hechas sobre un papel que, usualmente, es rectangular. Por esta razón y sin pérdida de generalidad, de aquí en adelante supondremos que una imagen digital es un conjunto rectangular de pares ordenados de números enteros.
En términos más precisos, se tiene la siguiente definición.

Definición 3.1: Imagen digital Π y su frontera

Una imagen digital Π *de M x N píxeles es un subconjunto finito de* $\mathbb{Z}^2$, *de modo tal que*

$$\Pi = \{ (x, y) \in \mathbb{Z}^2 : 1 \leq x \leq N , 1 \leq y \leq M \}$$

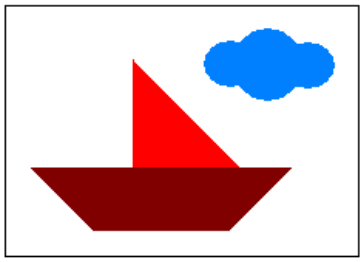
En este caso, el borde de Π, *que la simbolizaremos por Borde* (Π), *es*

$$Borde\ (\Pi) = \{(x, y) \in \Pi : x = 1 \vee x = N \vee y = 1 \vee y = M\}.$$

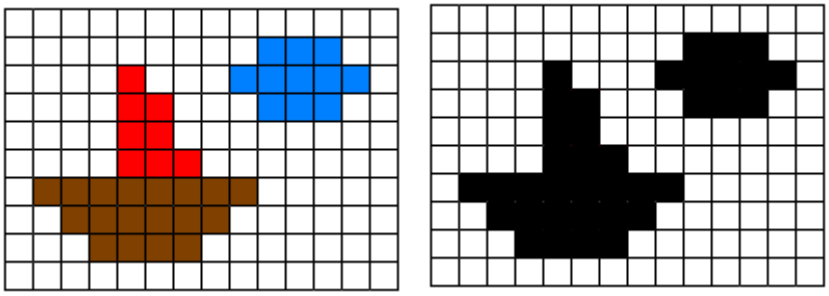
Desde el punto de vista computacional, la imagen Π es almacenada en el ordenador mediante su correspondiente matriz imagen (confrontar la Definición 1.2 de la página 13).
Comenzaremos trabajando con una imagen digital Π segmentada en dos subconjuntos: un conjunto S, formado por los objetos que componen la imagen, y su complemento $\overline{S}$, que representa el fondo. Supondremos que la segmentación fue hecha en modo Blanco y Negro: los píxeles del conjunto S son de color negro, mientras que los del fondo son blancos. De este modo, en el plano digital, los pares ordenados que provienen del conjunto S los representaremos mediante círculos de color negro y a los del complemento $\overline{S}$, de color blanco.
Aclararemos estos conceptos mediante un ejemplo.

Ejemplo 3.1

La siguiente figura muestra una imagen analógica:



Su imagen digitalizada, de 10 x 14 píxeles y luego segmentada en modo Blanco y Negro, se muestra a continuación:



(a) *(b)*

Figura 3.1: (a) Imagen digitalizada. (b) Imagen segmentada en Blanco y Negro.

En este caso, el conjunto S está formado por los píxeles negros y representan los objetos que componen la imagen, mientras que su complemento es el fondo blanco de la misma. La representación de la imagen segmentada en $\mathbb{Z}^2$ es Π, según se muestra en la siguiente figura:

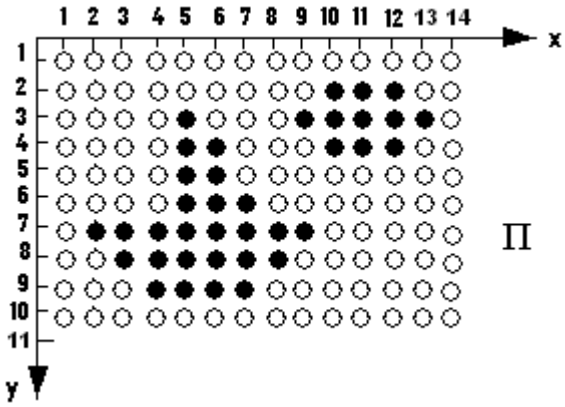


Figura 3.2: Representación de Π *en el plano digital.*

Se observa que el borde de Π es $\{(x, y) \in \Pi : x = 1 \vee x = 14 \vee y = 1 \vee y = 14\}$.

Desde el punto de vista computacional, no es posible trabajar dentro del ordenador con el plano digital $\mathbb{Z}^2$ ya que éste es un conjunto infinito. Por esta razón nos limitaremos exclusivamente al estudio topológico de la imagen Π y, para ello, le asignaremos la topología relativa a τ de $\mathbb{Z}^2$. Más precisamente,

Definición 3.2: La topología τ_Π de Π

La topología τ_Π de Π es la topología generada por la base

$$\mathbf{B}_\Pi = \{\ U(P) \cap \Pi : P \in \mathbb{Z}^2 \}$$

De este modo, los miembros de τ_Π se escriben como unión de elementos de $\mathbf{B}_\Pi$.

Formularemos a continuación el concepto de conectividad para subconjuntos de una imagen digital Π.
Para evitar casos especiales, supondremos de ahora en adelante que cualquier subconjunto $S \subset \Pi$ no interseca el borde de Π. Es decir, $S \cap \text{Borde}(\Pi) = \varnothing$

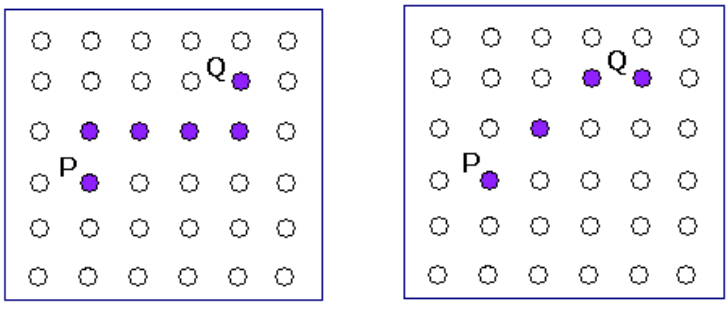
Caminos y conectividad

Definición 3.3: Caminos o trayectorias

Sean P y Q dos puntos de la imagen Π. Un "camino o trayectoria de P a Q" es una sucesión finita de puntos $P = P_0, P_1, ..., P_n = Q$ pertenecientes a Π tal que P_i es "vecino" de P_{i-1} para $1 \le i \le n$.

Observación

La definición anterior abarca en realidad dos definiciones, pues "vecino" puede significar "vecino 4N" o bien "vecino 8N". En estos casos, se dirá que es un ***"camino 4N"*** o ***"camino 8N"*** respectivamente. En general, los caminos 4N son también caminos 8N.



(a) *(b)*

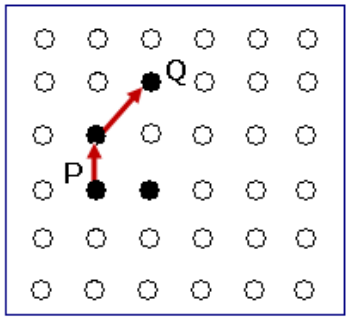
Figura 3.3: (a) Camino 4N de P a Q. (b) Camino 8N de P a Q.

Definición 3.4: Conectividad de dos puntos.

Dos puntos P y Q "están conectados en S" si existe un camino desde P hasta Q que se compone totalmente de puntos de S.
Según sea un camino 4N o un camino 8N, diremos que "P y Q tienen conectividad 4N en S" o bien "P y Q tienen conectividad 8N en S".

Ejemplo 3.2

Sea S el conjunto formado por los círculos negros de la imagen Π que muestra la figura:



Entonces P y Q tienen conectividad 8N pero no tienen conectividad 4N en S.

La siguiente proposición es importante para introducir el concepto de "componentes conexas de un conjunto".

Proposición 3.1

Dado S un subconjunto no vacío de Π , se define la siguiente relación "$\sim$ " en S:

$P \sim Q$ si y sólo si P y Q están conectados en S.

Entonces dicha relación es de equivalencia.

Demostración

Debemos verificar la propiedad reflexiva, simétrica y transitiva.

Reflexiva:

$P \sim P$ pues $P_0 = P$, $P_1 = P$ es un camino de P a P en S.

Simétrica:

Si $P \sim Q$ entonces $\exists\ P = P_0, P_1, ..., P_n = Q$ un camino de P a Q tal que $P_i \in S$ $\forall\ i = 0,1,...,n$. Luego $Q = P_n, P_{n-1}, ..., P_0 = P$ es un camino de *Q* a *P* en *S*. Por lo tanto, $Q \sim P$.

Transitiva:

Si $P \sim Q$ y $Q \sim R$ entonces
$\exists\ P = P_0, P_1, ..., P_n = Q$ un camino de P a Q tal que $P_i \in S\ \ \forall i = 0,1,...,n$

Por otro lado, $\exists \; Q = P_n, P_{n+1}, ..., P_{n+m} = R$ un camino de Q a R tal que $P_j \in S$ $\forall j = n, n+1, ..., n+m$. Entonces $P = P_0, P_1, ..., P_n, P_{n+1}, ..., P_{n+m} = R$ es un camino de *P* a *R* en S. En consecuencia, $P \sim R$. ■

Según A. Rosenfeld [3], las clases de equivalencia definidas por esta relación reciben el nombre de "componentes conexas". Más precisamente,

Definición 3.5: Componentes conexas de un conjunto.

Sea S un subconjunto no vacío de Π *y "~ " la relación de equivalencia definida en S mediante*

P ~ Q si y sólo si P y Q están conectados en S.

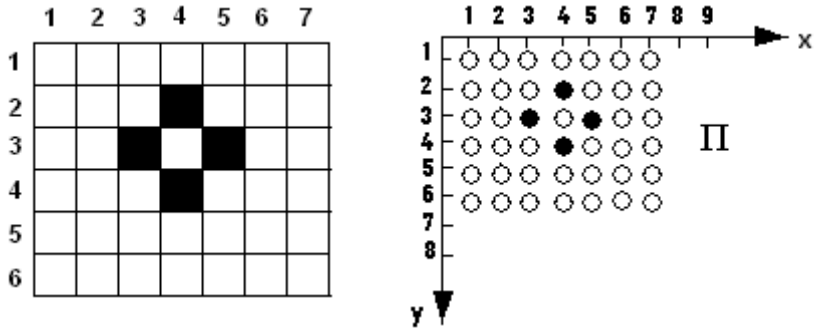
Entonces las clases de equivalencia de S reciben el nombre de ***componentes conexas de S***

Observación

Nuevamente, la definición anterior abarca en realidad dos definiciones, pues "conectividad" puede significar "conectividad 4N" o bien "conectividad 8N". En estos casos, se hablará de ***"componentes 4N conexas"*** o bien ***"componentes 8N conexas"*** respectivamente. Por otro lado, las componentes 4N pueden no coincidir con las componentes 8N de un conjunto, como lo ilustra el siguiente ejemplo:

Ejemplo 3.3

Sea Π la imagen digital de 6 x 7 píxeles y S = {(4,2), (3,3), (5,3), (4,4)} un subconjunto que se muestra continuación:



(a) *(b)*

Figura 3.4: (a) Imagen digital de 6 x 7 píxeles. (b) Su representación Π *en el plano digital.*

Entonces

Componentes 4N de S:
Ningún par de puntos de S tiene conectividad 4N. Por lo tanto S tiene 4 componentes 4N conexas, a saber: {(4,2)}, {(3,3)}, {(5,3)}, {(4,4)}

Componentes 8N de S:
En este caso, cualquier par de puntos de S tiene conectividad 8N. Luego S tiene una sola componente 8N conexa, que es S.

Cuando el conjunto S tiene sólo una componente conexa, se dice que es conexo. Para ser más precisos, podemos establecer la siguiente definición:

Definición 3.6: Conjunto conexo.

Si S es un subconjunto no vacío de Π *entonces*

S es "4N conexo" si tiene una sola componente 4N conexa.
S es "8N conexo" si tiene una sola componente 8N conexa.

Obviamente, si un conjunto es 4N conexo entonces también es 8N conexo. Sin embargo la recíproca no es cierta, como veremos a continuación.

Ejemplo 3.4

En el Ejemplo 3.3, S es 8N conexo pero no es 4N conexo.
Por otro lado, $\overline{S}$ también es 8N conexo pero tampoco es 4N conexo. Sus componentes 4N conexas son dos: $\Pi - (S \cup \{(4,3)\})$ y $\{(4,3)\}$.

La Definición 3.6 fue formulada por A. Rosenfeld [3] y es más bien una definición de "arco conexión". Sin embargo, coincide con el concepto topológico que se tiene de conjunto conexo, siempre y cuando éste sea 4N conexo. Esta afirmación la demostraremos en breve, pero previamente recordaremos la definición topológica de conjunto conexo.

Definición 3.7: Conjunto topológicamente conexo.

Sea S un subconjunto no vacío de Π *. Diremos que*

S es topológicamente conexo si lo es en la topología relativa a τ_Π *de* Π*. Es decir ,*

Si $S = A \cup B$ *con A, B abiertos en S tales que* $A \cap B = \varnothing \Rightarrow A = \varnothing$ *ó* $B = \varnothing$

Proposición 3.2

Sea S un subconjunto no vacío de Π . Entonces
S es topológicamente conexo si y sólo si es 4N conexo.

Demostración

$\Rightarrow$) Supongamos por el absurdo que S no es 4N conexo. Entonces, $\exists$ P, Q$\in$ S tales que P y Q no tienen conectividad 4N en S. Consideremos los conjuntos

$\mathbf{A} = \{Z \in S : Z \wedge P$ tienen conectividad 4N en S $\}$
$\mathbf{B} = \overline{\mathbf{A}} = \{Z \in S : Z \wedge P$ no tienen tiene conectividad 4N en S $\}$

Por lo tanto,
$S = \mathbf{A} \cup \mathbf{B}$. Además, $\mathbf{A} \neq \varnothing$ y $\mathbf{B} \neq \varnothing$ pues $P \in \mathbf{A}$ y $Q \in \mathbf{B}$.

Por otro lado, el conjunto

$$\{U(P) \cap \Pi \cap S : P \in \mathbb{Z}^2\} = \{U(P) \cap S : P \in \mathbb{Z}^2\}$$

Es una base para el subespacio topológico S, pues éste hereda la topología relativa a τ_Π de Π. Basándonos en este hecho, probaremos que **A** y **B** son abiertos en S.

<u>**A** es abierto en S:</u>

Sea $Z \in \mathbf{A}$; entonces $\exists\ P = P_0, P_1, ..., P_n = Z$ un camino 4N de P a Z tal que $P_i \in S$ $\forall i = 0,1,...,n$. Por otra parte, $U(Z) \cap S \subset \mathbf{A}$ pues
Si $W \in U(Z) \cap S$ entones W es vecino 4N de Z; luego $P = P_0, P_1, ..., P_n = Z$, $P_{n+1} = W$ es un camino 4N en S, que conecta P con W. De aquí resulta que $W \wedge P$ tienen conectividad 4N en S y en consecuencia $W \in \mathbf{A}$. Por lo tanto $Z \in U(Z) \cap S \subset \mathbf{A}$.

<u>**B** es abierto en S</u>

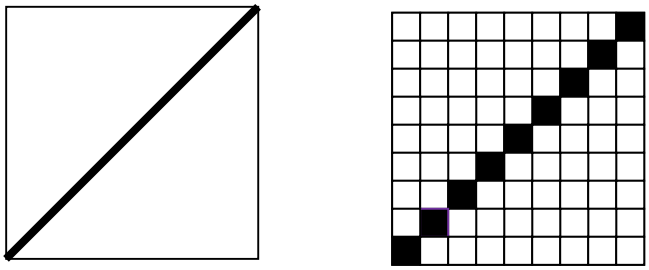
Sea $Z \in \mathbf{B}$; probemos que $U(Z) \cap S \subset \mathbf{B}$.
Si $W \in U(Z) \cap S$ entones $W \wedge P$ no tienen conectividad 4N en S pues si lo tuvieran, existiría $P = P_0, P_1, ..., P_n = W$ un camino 4N en S, de P a W. Luego, la secuencia de puntos $P = P_0, P_1, ..., P_n = W$, $P_{n+1} = Z$ sería un camino 4N en S, que conecta P con Z, lo cual es un absurdo pues $Z \wedge P$ no tienen conectividad 4N en S. En consecuencia $W \in \mathbf{B}$ y de este modo $Z \in U(Z) \cap S \subset \mathbf{B}$.

De este modo $S = \mathbf{A} \cup \mathbf{B}$ con **A**, **B** abiertos en S, disjuntos y no vacíos. Pero esto es una contradicción ya que S es topológicamente conexo.

$\Leftarrow$) Sea $S = \mathbf{A} \cup \mathbf{B}$ con **A**, **B** abiertos en S, disjuntos y no vacíos. Debemos probar que $\mathbf{A} = \varnothing$ ó $\mathbf{B} = \varnothing$.
Supongamos por el absurdo que $\mathbf{A} \neq \varnothing$ y $\mathbf{B} \neq \varnothing$. Entonces, $\exists P, Q \in S$ tales que $P \in \mathbf{A}$ y $Q \in \mathbf{B}$.
Como P y Q están 4N conectados en S, $\exists\ P = P_0, P_1, ..., P_n = Q$ un camino 4N de P a Q tal que $P_i \in S\ \ \forall i = 0,1,...,n$.
Por otro lado, como **A** es abierto en S, se tiene que $P_0 = P \in U(P) \cap S \subset \mathbf{A}$. Luego, al ser P_1 vecino 4N de P, resulta que $P_1 \in U(P) \cap S$ y, en consecuencia, $P_1 \in \mathbf{A}$.
Un argumento análogo demuestra que $P_2 \in \mathbf{A}$, …, $P_n = Q \in \mathbf{A}$.
De este modo, $Q \in \mathbf{A} \cap \mathbf{B}$, lo cual es un absurdo pues **A** y **B** son disjuntos. ■

¿Por qué definir dos tipos de conectividad?

Cabe preguntarnos porqué para un subconjunto de una imagen digital se definen dos tipos de conexiones: "conectividad 4N" y "conectividad 8N". ¿No bastaría con definir sólo una de ellas? Pues bien, la Topología Digital trata de trasladar adecuadamente las propiedades topológicas del plano continuo ($\mathbb{R}^2$) al discreto ($\mathbb{Z}^2$) de manera tal que, en la mayoría de los casos, éstas se preserven. A modo de ejemplo, observemos la siguiente figura:



(a) *(b)*

Figura 3.5: (a) Imagen continua . (b) Imagen digitalizada

La figura de la izquierda representa una imagen continua que contiene un segmento de recta. Si consideramos que el conjunto S es el segmento de recta, en la topología usual de $\mathbb{R}^2$ esta imagen tiene:

Componentes conexas de S: Hay una sola y es S, pues el segmento de recta es un conjunto arco conexo.

Componentes conexas de $\overline{\mathbf{S}}$***:*** Tenemos dos: la porción de imagen que está por arriba de del segmento de recta y la que se encuentra debajo de él.

De este modo, en el plano continuo, la imagen continua tiene una componente conexa negra y dos blancas.

Por otro lado, la figura de la derecha muestra la correspondiente imagen digitalizada. En este caso, el conjunto S está representado por los píxeles negros. Así,

Componentes 4N conexas de S: Hay nueve, pues cada píxel de S es una componente 4N conexa.

Componentes 4N conexas de $\overline{\mathbf{S}}$***:*** Tenemos 2: el conjunto de píxeles blancos que están por encima de los negros y los píxeles blancos que se encuentran debajo de los negros.

Por lo tanto, considerando las componentes 4N conexas, la imagen digital tiene nueve componentes negras y dos blancas. Por otro lado,

Componentes 8N conexas de S: Hay una sola, pues todos píxeles de S tienen conectividad 8N en S.

Componentes 8N conexas de $\overline{\mathbf{S}}$***:*** Hay también una sola, ya que todos los píxeles blancos tienen conectividad 8N en $\overline{\mathbf{S}}$.

En consecuencia, considerando las componentes 8N conexas, habrá una componente negra y una blanca,

En ambos casos, tanto para conectividad 4N como para conectividad 8N, la propiedad topológica de "conexión" no se preserva al pasar del plano continuo al discreto.
¿Habrá alguna manera de corregir esta situación?
Una forma de resolver este problema, y otros de índole similar, fue dada ingeniosamente por A. Rosenfeld [3] y es la siguiente:

El tipo de conectividad que se debe considerar para los píxeles negros debe ser diferente a la de los píxeles blancos. Así, si para el conjunto S se considera conectividad 4N, para $\overline{\mathbf{S}}$ *deberemos tomar conectividad 8N. O bien si para S se considera conectividad 8N, para* $\overline{\mathbf{S}}$ *deberemos tomar conectividad 4N.*

Volviendo a nuestro ejemplo, si tomamos conectividad 8N para los píxeles negros y conectividad 4N para los blancos, entonces la imagen digital tiene las mismas componentes conexas que la imagen original, es decir, presenta una componente conexa negra y dos componentes blancas.
Esta idea formulada por A. Rosenfeld constituye la base y fundamento de la Topología Digital y la enunciaremos formalmente a continuación:

Definición 3.8: Tipos de conectividad para un subconjunto y su complemento.

Sea S un subconjunto no vacío de Π . *Entonces*

Si consideramos conectividad 4N para S, en $\overline{S}$ *deberemos tomar conectividad 8N.*

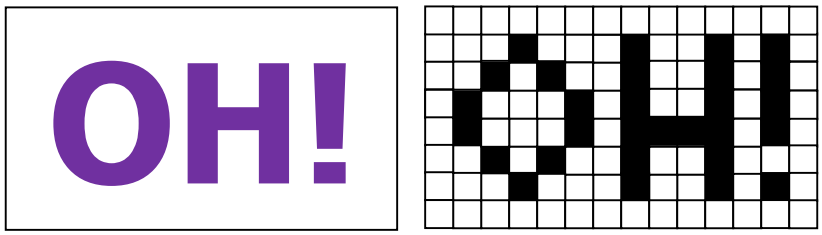
En cambio,

Si consideramos conectividad 8N para S, en $\overline{S}$ *deberemos tomar conectividad 4N.*

Ejemplo 3.5

La siguiente figura muestra una página impresa y su correspondiente imagen digitalizada de 8 x 14 píxeles:

.



(a) *(b)*

Figura 3.6: (a) Imagen de una página impresa. (b) Imagen digitalizada

La representación de Π en el plano digital es la siguiente:

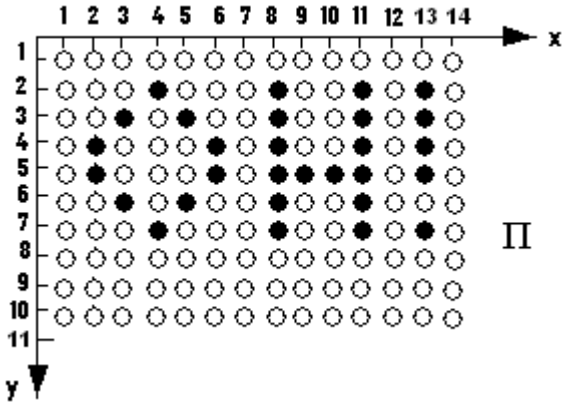


Figura 3.7: Representación de Π *en el plano digital.*

Si el conjunto S es el texto de la página impresa, es decir, el conjunto de píxeles negros de la imagen digital, calcularemos las componentes conexas de S y de $\overline{S}$, considerando conectividad 8N para S y conectividad 4N para $\overline{S}$:

Componentes 8N conexas de S: Son cuatro, a saber:

La letra "O": {(4,2), (3,3), (5,3), (2,4), (6,4), (2,5) , (6, 5), (3,6), ((5,6), (4,7)}.

La letra "H": {(8,2), (8,3), (8,4), (8,5), (8,6), (8,7), (9,5), (10,5), {(11,2), (11,3), (11,4), (11,5), (11,6), (11,7)}.

El símbolo "❙": {(13,2), (13,3), (13,4), (13,5)}.

El punto "•": {(13,7)}

Componentes 4N conexas de $\overline{S}$: Hay dos:

Los píxeles blancos ubicados dentro de la letra "O": {(4,3), (3,4), (4,4), (5,4),(3,5), (4,5), (5,5), (4,6)}

El conjunto $\overline{S}$ – {(4,3), (3,4), (4,4), (5,4),(3,5),(4,5), (5,5), (4,6)}.

Observación

El ejemplo anterior ilustra cómo, al hallar las componentes conexas de la imagen digital de un texto impreso, logramos identificar cada carácter del mismo.

Fondo y agujeros

En esta sección nos detendremos a explicar algunos conceptos vinculados con las componentes 4N u 8N conexas del complemento de un subconjunto de una imagen digital. Primeramente demostraremos la siguiente proposición:

Proposición 3.3

Sea S un subconjunto no vacío de una imagen digital Π de M x N píxeles. Entonces existe una única componente (4N u 8N) conexa de $\overline{S}$ que contiene al borde de Π.

Demostración

Recordemos que Borde (Π) = $\{(x, y) \in \Pi : x = 1 \vee x = N \vee y = 1 \vee y = M\}$.

Además, como $S \cap$ Borde (Π) = $\varnothing$, resulta Borde (Π) $\subset \overline{S}$.

Por otro lado, consideremos $S_1, S_2, \ldots, S_n$ las componentes (4N u 8N) conexas de $\overline{S}$. Entonces

$\overline{S} = S_1 \cup S_2 \cup \ldots \cup S_n$ con $S_i \cap S_j = \varnothing$ si $i \neq j$

Por lo tanto,

Borde (Π) $\subset S_1 \cup S_2 \cup \ldots \cup S_n$

De este modo, si P $\in$ Borde (Π) $\Rightarrow$ P$\in S_i$ para algún i. Pero entonces Borde (Π) $\subset S_i$, pues si existiera Q$\in$ Borde (Π) tal que Q$\notin S_i$ entonces Q$\in S_j$ para algún $j \neq i$.

Por otro lado, como Borde (Π) es un rectángulo en $\mathbb{Z}^2$, P y Q tienen conectividad 4N en $\overline{S}$ (y por ende, conectividad 8N). Por lo tanto $P \in S_j$ y en consecuencia $P \in S_i \cap S_j$ lo cual es una contradicción pues las componentes conexas son disjuntas.
De este modo, Borde (Π) $\subset S_i$ y, obviamente, S_i es la única componente conexa que contiene a Borde (Π).■

La Proposición anterior da lugar para introducir las siguientes definiciones.

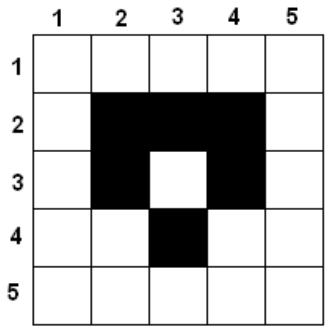
Definición 3.9: Fondo y agujeros de un conjunto

Sea S un subconjunto no vacío de una imagen digital Π. *Entonces*
El ***"Fondo de S"*** *es la única componente conexa de* $\overline{S}$ *que contiene a Borde* (Π).
Las demás componentes de $\overline{S}$, *si existen, reciben el nombre de* ***"agujeros de S"***.

El concepto de "fondo" y "agujero" de un conjunto depende del tipo de conectividad que se considere para dicho conjunto. El siguiente ejemplo ilustra esta afirmación.

Ejemplo 3.6

Sea Π la imagen digital de 5 x 5 píxeles y S = {(2,2), (3,2), (4,2), (2,3), (4,3), (3,4)} el conjunto de píxeles negros que muestra la figura



a) Hallar el fondo y los posibles agujeros de S, si se considera conectividad 8N para S.

b) Determinar el fondo y los posibles agujeros de S, tomando conectividad 4N para S.

Resolución

a) En este caso, como consideramos conectividad 8N para S, deberemos tomar conectividad 4N para su complemento. Entonces:

Componentes 4N conexas de $\overline{S}$*:* Hay dos: {(3,3)} y $\overline{S}$ - {(3,3)}. Por lo tanto

Fondo de S: $\overline{S}$ - {(3,3)}

Agujero de S: {(3,3)}

b) Aquí deberemos tomar conectividad 8N para $\overline{S}$**.** En este caso,

Componentes 8N conexas de $\overline{S}$***:*** Sólo hay una y es $\overline{S}$**,** pues éste es 8N conexo. Luego,

Fondo de S: $\overline{S}$

Agujero de S: no tiene.

Como sabemos, en la topología usual de $\mathbb{R}^2$ los conjuntos que no tienen agujeros reciben el nombre de "simplemente conexos". Esta idea puede trasladarse al plano digital de la siguiente manera:

Definición 3.10: Conjunto simplemente conexo

Sea S un subconjunto no vacío de una imagen digital Π. *Entonces S es* ***simplemente conexo*** *si no tiene agujeros.*

Ejemplo 3.7

En el ejemplo anterior, si consideramos conectividad 4N para S, éste es simplemente conexo pues no tiene agujeros.

Observación

De la Definición 3.9, sabemos que si S no tiene agujeros entonces $\overline{S}$ tiene una única componente conexa, que es el "fondo de S". Por lo tanto podemos afirmar que:
S es simplemente conexo $\Leftrightarrow$ $\overline{S}$ es conexo.

Arcos y Curvas

En el procesamiento de imágenes digitales, un método comúnmente utilizado para analizar la forma que tienen los objetos de una imagen digital, consiste en reducir los conjuntos "gruesos" a conjuntos "delgados". Por ejemplo, si un objeto es "alargado" o "simplemente conexo", se trata de reducirlo a un "arco". O bien, si tiene un "agujero", se trata de reducirlo a una "curva cerrada". Este proceso de "adelgazamiento", que expondremos más adelante, requiere de conceptos como arcos y curvas digitales. Por esta razón veremos este tema a continuación.

Definición 3.11: Arco

Un subconjunto S de una imagen digital Π *es un arco si es conexo y todos, salvo dos de sus puntos (sus "extremos") tienen exactamente dos vecinos en S, mientras que los dos extremos tienen exactamente uno. Más precisamente,*
Un arco S es un camino $P_0, P_1, \ldots, P_n$ *formado por puntos distintos y tal que* P_i *es vecino de* $P_j \Leftrightarrow j = i \pm 1$.

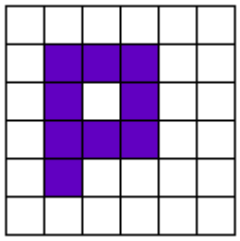
Es fácil ver que un arco puede ser considerado como un camino abierto que no se cruza ni se toca a sí mismo.

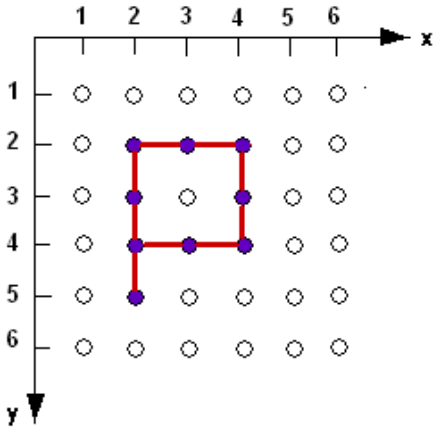
La definición de arco involucra la palabra “vecino”, lo cual puede significar “vecino 4N” o “vecino 8N”. De este modo hablaremos, cuando sea necesario especificar, de un “arco 4N” o de un “arco 8N”.
Para evitar casos especiales, supondremos que un arco siempre tiene al menos dos puntos. Mostraremos a continuación algunos ejemplos que aclararán estos conceptos.

Ejemplo 3.8

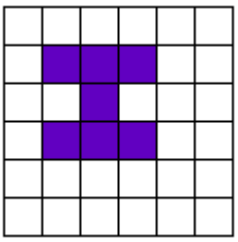
Para cada una de las siguientes imágenes digitales de 6 x 6 píxeles, indicar si el subconjunto S es un arco 4N, 8N o ninguno de los dos.

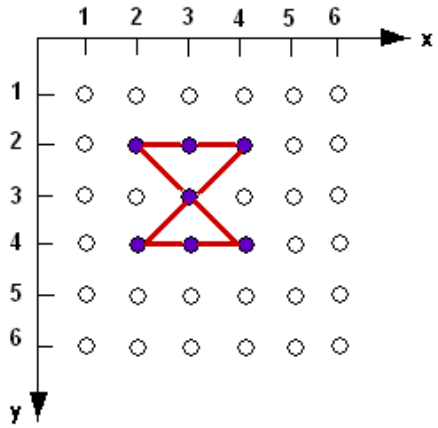
a) S = {(2,5), (2,4), (2,3), (2,2), (3,2), (4,2), (4,3), (4,4), (3,4)}



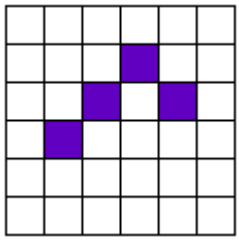


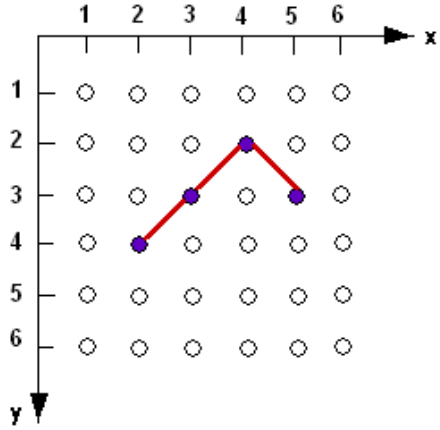
b) S = {(2,2), (3,2), (4,2), (3,3), (2,4), (3,4), (4,4)}



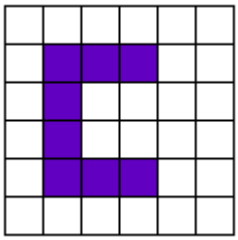


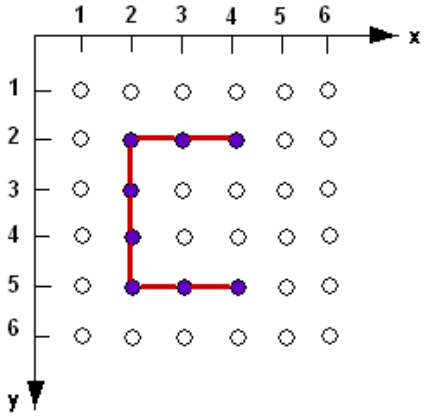
c) S = {(2,4), (3,3), (4,2), (5,3)}





d) S = {(4,5), (3,5), (2,5), (2,4), (2,3), (2,2), (3,2), (4,2)}





Resolución

a) S no es un arco 4N pues (2,4) tiene tres vecinos 4N. Por la misma razón tampoco es un arco 8N.

b) S no es un arco 4N ya que hay cuatro puntos que tienen exactamente un vecino 4N. Ellos son: (2,2), (4,2), (4,4) y (2,4). Tampoco es un arco 8N pues por ejemplo (3,3) tiene seis vecinos 8N: (2,2), (3,2), (4,2), (2,4), (3,4), y (4,4) .

c) S no es arco 4N pues ningún par de puntos son vecinos 4N. Sin embargo, S es un arco 8N.

d) Claramente S es un arco 4N pero no es un arco 8N pues por ejemplo (2, 3) tiene tres vecinos 8N: (2, 2), (3,2) y (2, 4).

Podemos observar gráficamente que un arco es siempre simplemente conexo. Este hecho lo demostraremos, pero previamente enunciaremos las siguientes proposiciones.

Proposición 3.4

Sea P = (x, y) un punto del plano digital Π. *Consideremos el conjunto*

$$N(P) = \{Q \in \Pi : Q \text{ es vecino 8N de } P\}$$

Entonces

a) N(P) – {P} es 4N conexo.

b) Si Q es un vecino 8N de P entonces N(P) – { P, Q } es 4N conexo.

Demostración

Es trivial y por esta razón la omitimos. ■

Notemos que la proposición anterior deja de ser cierta si en lugar de N(P) consideramos el conjunto formado por los vecinos 4N de P.

Proposición 3.5

Sea S un subconjunto de una imagen digital Π *tal que S = {a, b} donde a y b son vecinos 8N. Entonces* $\overline{S}$ *es 4N conexo.*

Demostración

Sean P, Q $\in \overline{S} = \Pi - \{ a , b \}$; debemos probar que P y Q tienen conectividad 4N en $\overline{S}$. Como Π es 4N conexo, $\exists$ $P_0 = P, P_1, \ldots, P_n = Q$ un camino 4N en Π. Sin pérdida de generalidad, podemos suponer que $P_i \neq P_j \ \forall \ i \neq j$.
Se presentan los siguientes casos:

1) Si $P_i \notin \{a, b\} \ \forall \ i = 0,1,...,n$ entonces $P_0 = P, P_1, \ldots, P_n = Q$ es un camino 4N en $\Pi - \{ a , b \}$ y, en consecuencia, P y Q tienen conectividad 4N en $\overline{S}$.

2) Si P_i = "a" para algún i entonces, como P_{i-1} y P_{i+1} son vecinos 4N de "a", también son vecinos 8N de "a". De aquí resulta que $P_{i-1}, P_{i+1} \in N(a) - \{a\}$. Por la Proposición 3.4 este conjunto es 4N conexo, con lo cual $\exists$ $R_1 = P_{i-1}, \ldots, R_k = P_{i+1}$ un camino 4N en N(a) – {a}. Por lo tanto $P_0 = P, \ldots, P_{i-1} = R_1, \ldots, R_k = P_{i+1}, \ldots, P_n = Q$ es un camino 4N en Π - {a}.
Para simplificar la notación, llamaremos a este camino $Q_0 = P, Q_1, \ldots, Q_{n+k-2} = Q$.

2.1) Si $Q_i \neq$ "b" $\forall \ i = 0,1,...,n+k-2$ entonces $Q_0 = P, Q_1, \ldots, Q_{n+k-2} = Q$ es un camino 4N en $\Pi - \{ a , b \}$ y, por lo tanto, P y Q tienen conectividad 4N en $\overline{S}$.

2.2) Si Q_i = "b" para algún i entonces, como Q_{i-1} y Q_{i+1} son vecinos 4N de "b", resulta que $Q_{i-1}, Q_{i+1} \in N(b) - \{a, b\}$. Por la Proposición 3.4 este conjunto es también 4N conexo, con lo cual $\exists$ $T_1 = Q_{i-1}, \ldots, T_m = Q_{i+1}$ un camino 4N en N(b) – { a, b }.
De este modo, $Q_0 = P, \ldots, Q_{i-1} = T_1, \ldots, T_m = Q_{i+1}, \ldots, Q_n = Q$ es un camino 4N en Π - {a, b} y, en consecuencia, P y Q tienen conectividad 4N en $\overline{S}$.

3) Si P_i = "b" para algún i, la demostración es similar al caso 2). ■

Basándonos en estas dos proposiciones, probaremos a continuación que un arco siempre es simplemente conexo.

Proposición 3.6

Si $S \subset \Pi$ *es un arco entonces S es simplemente conexo.*

Demostración

Debemos probar que $\overline{S}$ es conexo (confrontar la Observación de la página 39). Sin pérdida de generalidad, debido a que la demostración es similar, trabajaremos con la conectividad 8N para S, es decir, supondremos que S es un arco 8N. De este modo, deberemos considerar conectividad 4N para $\overline{S}$, o sea, probaremos que $\overline{S}$ es 4N conexo.

Aplicaremos el Principio de Inducción sobre n, la cantidad de puntos que tiene el conjunto S.

Para n = 2

En este caso, S = {a , b} con a, b vecinos 8N. De la Proposición 3.5 resulta que $\overline{S}$ es 4N conexo y en consecuencia, S es simplemente conexo.

Supongamos, por hipótesis inductiva, que todo arco de "n" puntos que no interseca el borde de Π es simplemente conexo.

Veamos para n+1

Sea S un arco formado por (n+1) puntos. Consideremos S_1, el arco que se obtiene de eliminar uno de los puntos extremos de S, que llamaremos "a". Entonces $S = S_1 \cup \{a\}$.
Sean $P, Q \in \overline{S}$; debemos probar que P y Q tienen conectividad 4N en $\overline{S}$.
Por la Ley de Morgan tenemos que $\overline{S} = \overline{S_1 \cup \{a\}} = \overline{S_1} \cap \overline{\{a\}}$. En consecuencia, $P, Q \in \overline{S_1} \cap \overline{\{a\}}$. De aquí resulta que $P, Q \in \overline{S_1}$, que es 4N conexo por hipótesis inductiva. Por lo tanto $\exists\ P_0 = P, P_1, \ldots, P_n = Q$ un camino 4N en $\overline{S_1}$. Podemos suponer, sin pérdida de generalidad, que $P_i \neq P_j\ \forall\ i \neq j$.
Se presentan los siguientes casos:

1) Si $P_i \notin \{a\}\ \forall\ i = 0,1,...,n$ entonces $P_0 = P, P_1, \ldots, P_n = Q$ es un camino 4N en $\overline{S_1} \cap \overline{\{a\}}$ y, en consecuencia, P y Q tienen conectividad 4N en $\overline{S}$.

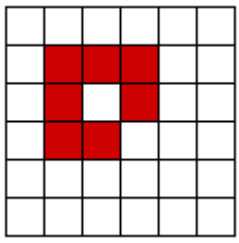
2) Si $P_i \in \{a\}$ para algún i entonces, como P_{i-1} y P_{i+1} son vecinos 4N de "a", también son vecinos 8N de "a". De aquí resulta que $P_{i-1}, P_{i+1} \in N(a) - \{a, b\}$, donde "b" es el único vecino 8N de "a" que está en S_1.
Por la Proposición 3.4 este conjunto es 4N conexo, con lo cual $\exists\ R_1 = P_{i-1}, \ldots, R_k = P_{i+1}$ un camino 4N en $N(a) - \{a, b\} \subset \overline{S_1} \cap \overline{\{a\}}$.
Por lo tanto $P_0 = P, \ldots, P_{i-1} = R_1, \ldots, R_k = P_{i+1}, \ldots, P_n = Q$ es un camino 4N en $\overline{S_1} \cap \overline{\{a\}}$. Luego P y Q tienen conectividad 4N en $\overline{S}$.■

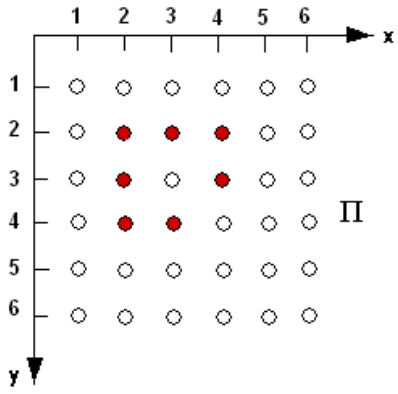
Observación

La proposición anterior deja de ser cierta si utilizamos conectividad 4 N, tanto para S como para su complemento, como lo muestra el siguiente ejemplo.

Ejemplo 3.9

Sea Π la imagen digital de 6 x 6 píxeles y S el subconjunto definido por S = {(2,2), (2,3), (2,4), (3,2), (3,4), (4,2), (4,3)}, según se indica en la siguiente figura:





Si aplicamos conectividad 4N para S y $\overline{S}$, se puede observar que S es un arco 4N pero $\overline{S}$ no es 4N conexo, pues tiene un agujero (que es el conjunto {(3, 3)}). En consecuencia, S no es simplemente conexo. Sin embargo, si consideramos conectividad 8N para $\overline{S}$, éste resulta 8N conexo y por lo tanto S es simplemente conexo.

Definición 3.12: Curva

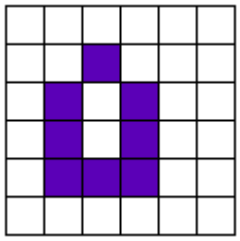
Un subconjunto S de una imagen digital Π *es una curva si es conexo y todos sus puntos tienen exactamente dos vecinos en S. Más precisamente,*
Una curva S es un camino $P_0, P_1, \ldots, P_n$ *formado por puntos distintos y tal que* P_i *es vecino de* $P_j \Leftrightarrow ((j = i \pm 1) \vee (i = 0 \wedge j = n) \vee (j = 0 \wedge i = n))$.

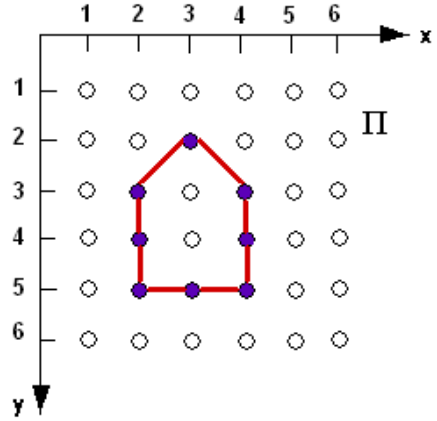
Podríamos decir entonces que una curva es un arco cerrado o dicho de otro modo, es un camino que se cruza o toca a sí mismo sólo una vez.
Cuando sea necesario especificar, hablaremos de una "curva 4N" o de una "curva 8N".según sus puntos tengan "vecindad 4N" o "vecindad 8N".
Para evitar casos especiales, supondremos que una ***curva 4N tiene al menos 8 puntos*** y que ***una curva 8N tiene al menos 4 puntos***. Veamos algunos ejemplos.

Ejemplo 3.10

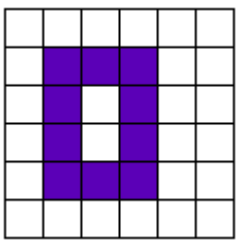
Para cada una de las siguientes imágenes digitales de 6 x 6 píxeles, indicar si el subconjunto S es una curva 4N, 8N o ninguna de las dos.

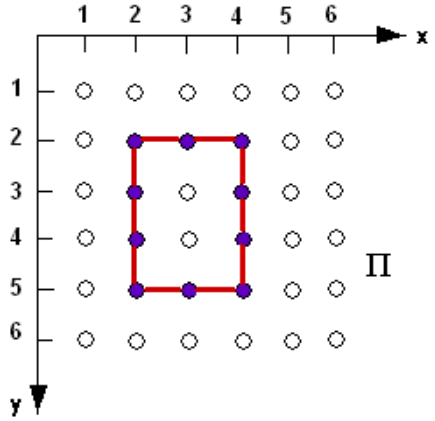
a) S = {(2,3), (2,4), (2,5), (3,5), (4,5), (4,4), (4,3), (3,2)}





b) S = {(2,2), (2,3), (2,4), (2,5), (3,5), (4,5), (4,4), (4,3), (4, 2), (3,2)}





Resolución

a) S no es una curva 4N ya que (3,2) no tiene ningún vecino 4N. Tampoco es una curva 8N porque por ejemplo (4, 4) tiene tres vecinos 8N: (4, 3), (4, 5) y (3, 5).

b) En este caso S es una curva 4N pero no es 8N ya que (4, 4) tiene tres vecinos 8N: (4, 3), (4, 5) y (3, 5).

Un resultado bastante curioso es que una curva no puede ser 4N y 8N a la vez. Esto se demuestra en la siguiente proposición.

Proposición 3.7:

Una curva $S \subset \Pi$ no puede ser 4N y además 8N.

Demostración

Supongamos por el absurdo que S es una curva 4N y 8N.
Luego, si $P \in S$, P debe tener exactamente dos vecinos 4N y dos vecinos 8N. Por lo tanto, las dos únicas posibilidades en cuanto a la disposición de P y sus vecinos 8N son las siguientes:

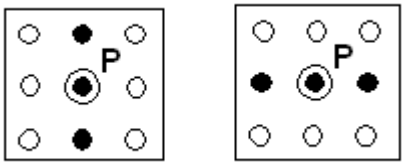


Figura 3.8: Posible disposición de un punto P de la curva y sus correspondientes vecinos 8N. Los círculos negros representan puntos de la curva S, mientras que los blancos pertenecen a su complemento.

De este modo, todos los puntos de S deberían estar alineados horizontal o verticalmente, lo cual es imposible ya que los extremos de S sólo tendrían un vecino en S. ■

En la topología usual de $\mathbb{R}^2$ un resultado fundamental, tanto desde el punto de vista teórico como práctico, es "El Teorema de la curva de Jordan". Dicho teorema afirma un hecho que intuitivamente es bastante evidente, como es que una curva simple cerrada C en el plano divide a éste en dos regiones: una acotada, llamada "el interior de C" y otra no acotada, llamada "el exterior de C", siendo C la frontera común de ambas.

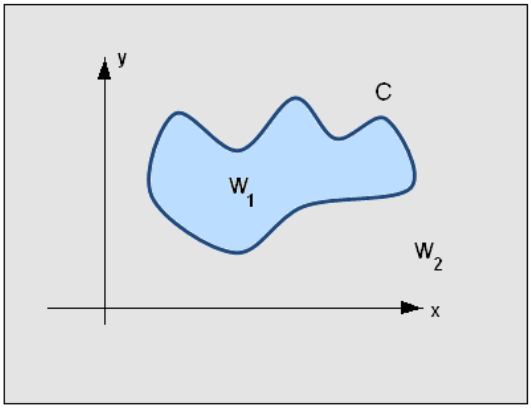


Figura 3.9: El Teorema de la curva de Jordan. La curva C divide al plano en dos regiones: su interior W_1 y su exterior W_2. Tanto W_1 como W_2 tienen por frontera a la curva C.

¿Podrá este teorema extenderse al plano digital, para curvas digitales? Afortunadamente la respuesta es afirmativa (para una curva 4N con al menos ocho puntos o una curva 8N con al menos cuatro puntos), y la enunciamos a continuación.

Teorema 3.8: Teorema de la curva de Jordan para curvas digitales.

Sea S una curva 4N en el plano digital $\mathbb{Z}^2$. *Entonces* $\mathbb{Z}^2 - S$ *tiene exactamente dos componentes 8N conexas. Una de ellas es acotada, llamada "el interior de S" y la otra es no acotada, llamada "el exterior de S".*

De manera similar, si S es una curva 8N en el plano digital $\mathbb{Z}^2$ *entonces* $\mathbb{Z}^2 - S$ *tiene exactamente dos componentes 4N conexas.*

Demostración

Se omite pues se encuentra completamente desarrollada en [6]. ■

La siguiente Proposición es una consecuencia inmediata del Teorema anterior.

Proposición 3.9:

Si $S \subset \Pi$ *es una curva, entonces S tiene exactamente un agujero.*

Demostración

Por el Teorema de la curva de Jordan para curvas digitales, sabemos que

$$\mathbb{Z}^2 - S = W_1 \cup W_2$$

Donde W_1 es el interior de S y W_2 su exterior.
Por otro lado,

$\overline{S} = \Pi - S = (\mathbb{Z}^2 - S) \cap \Pi = (W_1 \cup W_2) \cap \Pi = (W_1 \cap \Pi) \cup (W_2 \cap \Pi) =$
$= W_1 \cup (W_2 \cap \Pi)$.

De este modo, W_1 y $(W_2 \cap \Pi)$ son las dos únicas componentes conexas de $\overline{S}$ (4N u 8N, según sea el caso). Sólo una de ellas, $(W_2 \cap \Pi)$, contiene al borde de Π. En consecuencia, W_1 es el único agujero de $\overline{S}$. ■

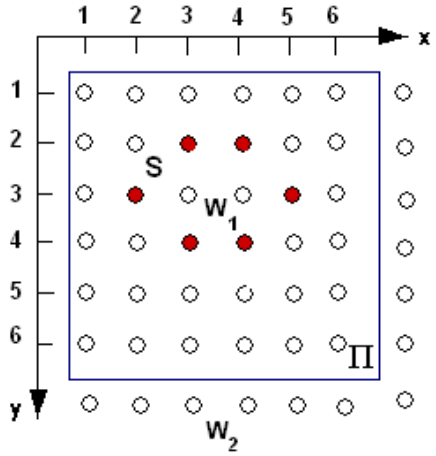
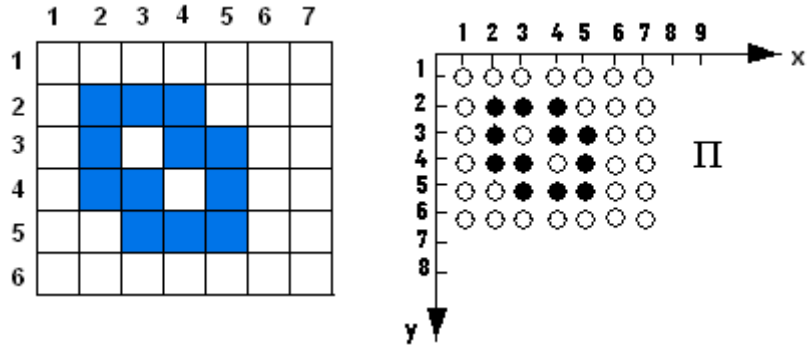


Figura 3.10: El Teorema de la curva de Jordan para curvas digitales. La curva S divide el plano digital en dos componentes: su interior W_1 *y su exterior* W_2. *Este último contiene al borde de* Π.

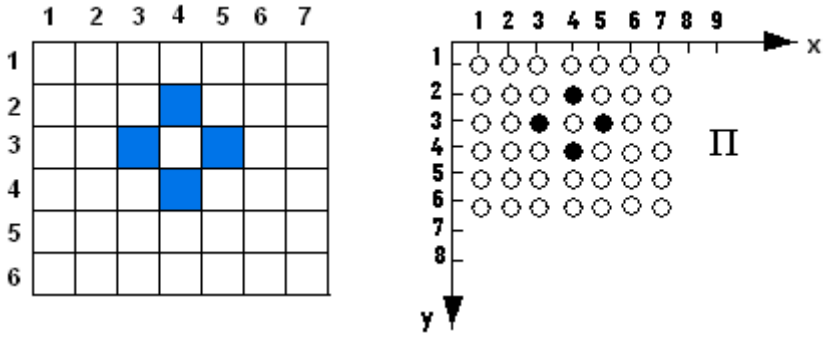
La Proposición anterior puede dejar de ser cierta si consideramos el mismo tipo de conectividad, tanto para la curva como para su complemento. O también si se considera una curva 8N con sólo tres puntos. Los siguientes ejemplos ilustran este hecho.

Ejemplo 3.11

Sea la siguiente curva S = {(2,2), (3,2), (4,2), (4,3), (5,3), (5,4), (5,5), (4,5), (3,5), (3,4), (2,4), (2,3)}, proveniente de una imagen digital de 6 x 7 píxeles:

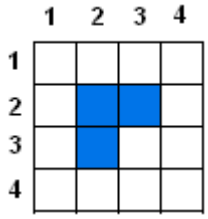


Si consideramos conectividad 4N tanto para S como para $\overline{\mathbf{S}}$, podemos observar que S tiene dos agujeros: {(3, 3)} y {(4,4)}. De este modo, no se verifica la Proposición 3.9.
Por otro lado, si tomamos la curva S = {(4,2), (3,3), (4,4), (5,3)} y consideramos conectividad 8N para S y su complemento, vemos que S no tiene agujeros, con lo cual la Proposición 3.9 también deja de ser cierta.



Ejemplo 3.12

El conjunto S = {(2,3), (2,2), (3,2)} es una curva 8N formada por tres puntos. Sin embargo, no se verifica la Proposición 3.9 ya que S no tiene agujeros.



Observación

El Teorema de la curva de Jordan para curvas digitales afirma que una curva S separa a su complemento $\overline{\mathbf{S}}$ en dos componentes: una interior, llamada "agujero" y otra exterior, llamada "fondo". Ambas componentes tienen conectividad 4N u 8N, según se considere para S conectividad 8N o 4N respectivamente. Pero dichas componentes ¿serán

topológicamente conexas? Bajo ciertas hipótesis, podemos asegurar que sí lo son. Esta afirmación nos lleva a enunciar la siguiente Proposición.

Proposición 3.10:

Sea $S \subset \Pi$ *una curva 8N . Entonces su fondo y agujero son topológicamente conexos.*

Demostración

Por el Teorema de la curva de Jordan para curvas digitales, sabemos que

$$\overline{\mathbf{S}} = \Pi - S = W_1 \cup W_2$$

Donde W_1 es el interior o agujero de S y W_2 es su exterior o fondo.
Como S es una curva 8N, las componentes W_1 y W_2 son 4N conexas. Por lo tanto, de la Proposición 3.2 se deduce que ambas componentes son topológicamente conexas. ■

Resulta geométricamente obvio que todo punto de una curva tiene por vecinos a algún punto de su agujero y de su fondo. Para demostrarlo, necesitaremos previamente la siguiente proposición.

Proposición 3.11:

Sea ***W*** *un subconjunto de una imagen digital* Π *y* ***A*** *una componente conexa de* ***W****.*
Sea $P_0, P_1, \ldots, P_n$ *un camino en* ***W****. Entonces*

Si $P_k \in \mathbf{A}$ *para algún* $0 \le k \le n \Rightarrow P_i \in \mathbf{A} \quad \forall\, i = 0,\ldots, n$

Demostración

Sin pérdida de generalidad, podemos suponer que k = 0. Recordemos que si $P_0 \in \mathbf{A}$ entonces $\mathbf{A} = \{Q \in \mathbf{W} : P_0 \wedge Q$ están conectados en **W**$\}$
Por otro lado, sea i tal que $1 \le i \le n$; entonces $P_0, P_1, \ldots, P_i$ es un camino en **W** de P_0 a P_i. Luego P_0 y P_i están conectados en **W** y, en consecuencia, $P_i \in \mathbf{A}$. ■

Basándonos en esta proposición, probaremos a continuación la afirmación mencionada anteriormente.

Proposición 3.12:

Todo punto de una curva $S \subset \Pi$ *es adyacente (en el sentido de conectividad de* $\overline{\mathbf{S}}$*) a las dos componentes de* $\overline{\mathbf{S}}$*.*

Demostración

Sea $P \in S$ y W_1, W_2 las componentes conexas de $\overline{\mathbf{S}}$, es decir, $\overline{\mathbf{S}} = W_1 \cup W_2$. Debemos probar que P es vecino de algún punto de W_1 y de algún punto de W_2
Como S – {P} es un arco, éste es simplemente conexo, por la Proposición 3.6.
Por lo tanto, $\overline{\mathbf{S}-\mathbf{\{P\}}}$ es conexo (en el sentido de conectividad de $\overline{\mathbf{S}}$). Por la ley de Morgan se tiene que

$\overline{\mathbf{S}-\{\mathbf{P}\}} = \overline{\mathbf{S}\cap\overline{\{\mathbf{P}\}}} = \overline{\mathbf{S}} \cup \overline{\overline{\{\mathbf{P}\}}} = \overline{\mathbf{S}} \cup \{\mathbf{P}\} = W_1 \cup W_2 \cup \{P\}$.

Es decir, $W_1 \cup W_2 \cup \{P\}$ es conexo. Luego, dado $Q \in W_1$, existe un camino $P_0 = P, P_1,\ldots,P_n = Q$ en $W_1 \cup W_2 \cup \{P\}$ que une P con Q. Sin pérdida de generalidad, podemos suponer que los puntos de este camino son distintos. En consecuencia, $P_1,\ldots,P_n = Q$ es un camino en $W_1 \cup W_2 = \overline{\mathbf{S}}$. Como $Q \in W_1$ y ésta es una componente conexa de $\overline{\mathbf{S}}$, de la Proposición 3.11 resulta que $P_i \in W_1 \ \forall\, i = 1,\ldots, n$. En particular, $P_1 \in W_1$, y además es vecino de P.

Por otro lado, dado $R \in W_2$, existe un camino $Q_0 = P, Q_1,\ldots,Q_n = R$ en $W_1 \cup W_2 \cup \{P\}$ que une P con R. Nuevamente, suponiendo que los puntos de este camino son distintos, se prueba de manera similar que $Q_1 \in W_2$, que además es vecino de P. ■

Afinamiento

Como mencionamos en secciones anteriores, un método muy utilizado en el procesamiento de imágenes para analizar la forma que tienen los objetos de una imagen digital, consiste en reducir los conjuntos "gruesos" a conjuntos “delgados”. Este proceso de "adelgazamiento" recibe el nombre de “afinamiento” y su definición la formulamos a continuación.

Definición 3.13: Afinamiento de un conjunto

El afinamiento de un subconjunto S de una imagen digital Π *, es el proceso que consiste en eliminar los puntos de S, sin cambiar las propiedades de conectividad o simple conectividad de S, ni de su complemento* $\overline{\mathbf{S}}$.

La clase de puntos que se pueden eliminar con seguridad se caracterizan por la proposición que enunciaremos a continuación. Pero previamente aclararemos algunas notaciones y expresiones que figuran en la misma.

Algunas notaciones

■ Recordemos que, si $P = (x, y) \in \Pi$, el conjunto N(P) es

$N(P) = \{Q \in \Pi : Q \text{ es vecino 8N de } P\} = \{ (x, y), (x, y \pm 1), (x \pm 1, y), (x \pm 1, y \pm 1)\}$

■ Cuando decimos “las componentes conexas en el sentido de S o en el sentido de $\overline{\mathbf{S}}$”, nos estamos refiriendo al tipo de conectividad asignada a S y $\overline{\mathbf{S}}$ respectivamente. Por ejemplo, si consideramos conectividad 4N para S, entonces para $\overline{\mathbf{S}}$ deberemos tomar conectividad 8N. En este caso, si $\boldsymbol{A} \subset S$ y $\boldsymbol{B} \subset \overline{\mathbf{S}}$ entonces

Las ***“componentes conexas de A en el sentido de S”*** significa ***“las componentes 4N conexas de A”***. Mientras que las ***“componentes conexas de B en el sentido de*** $\overline{\mathbf{S}}$***”*** significa *“**las componentes 8N conexas de B”***.

Proposición 3.13:

Sea S un subconjunto de una imagen digital Π *. Las siguientes propiedades del punto* $P \in S$ *son equivalentes:*

a) $S \cap N(P)$ *tiene la misma cantidad de componentes (en el sentido de S) que* $S \cap [N(P) - \{P\}]$.

b) $\overline{\mathbf{S}} \cap N(P)$ *tiene la misma cantidad de componentes (en el sentido de* $\overline{\mathbf{S}}$*) que* $[\overline{\mathbf{S}} \cap N(P)] \cup \{P\}$.

c) $S \cap [N(P) - \{P\}]$ *tiene exactamente una componente adyacente a P. ("componente" y "adyacente" en el sentido de S)*

d) $\overline{\mathbf{S}} \cap N(P)$ *tiene exactamente una componente adyacente a P. ("componente" y "adyacente" en el sentido de* $\overline{\mathbf{S}}$*)*

e) $S - \{ P \}$ *tiene la misma cantidad de componentes (en el sentido de S) que S, y* $\overline{\mathbf{S}} \cup \{P\}$ *tiene la misma cantidad de componentes (en el sentido de* $\overline{\mathbf{S}}$ *) que* $\overline{\mathbf{S}}$.

Demostración

Consideraremos conectividad 8N para S y, por lo tanto, tomaremos conectividad 4N para $\overline{\mathbf{S}}$. Sólo probaremos la siguiente implicación:

c) $\Rightarrow$ d)

Supongamos por el absurdo que $\overline{\mathbf{S}} \cap N(P)$ tiene al menos dos componentes 4N conexas que son adyacentes a P. Sean W_1 y W_2 dichas componentes. Entonces, $\exists P_1 \in W_1$ y $P_2 \in W_2$ tales que P_1 y P_2 son vecinos 4N de P. Además, P_1 y P_2 no están 4N conectados en $\overline{\mathbf{S}} \cap N(P)$ pues pertenecen a diferentes componentes.

Por otro lado, sea C la curva 4N que va de P_1 a P_1, pasando por todos los puntos de $N(P) - \{P\}$.

Es decir, C: $Q_0 = P_1, \ldots, Q_k = P_2, Q_{k+1}, \ldots, Q_n = P_1$, con $Q_i \in N(P) - \{ P \}$ $\forall$ i = 0,…, n.

Por lo tanto, como P_1 y P_2 no están 4N conectados en $\overline{\mathbf{S}} \cap N(P)$, resulta

$$\{Q_0 = P_1, \ldots, Q_k = P_2\} \cap S \neq \varnothing \ \wedge \ \{Q_k = P_2, Q_{k+1}, .., Q_n = P_1\} \cap S \neq \varnothing$$

Entonces $\exists$ $\mathbf{x} \in \{Q_0 = P_1, \ldots, Q_k = P_2\} \cap S$ $\wedge$ $\exists$ $\mathbf{y} \in \{Q_k = P_2, Q_{k+1}, \ldots, Q_n = P_1\} \cap S$.

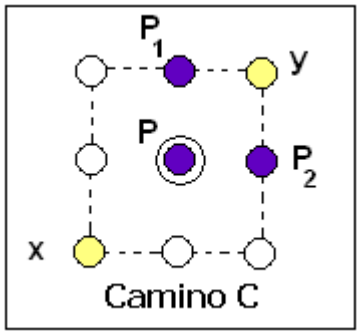


Figura 3.11: Posible disposición de los puntos P_1 *,* P_2 *y de los puntos* ***x*** *e* ***y****.*

Luego **x**, **y** $\in$ $S \cap [N(P) - \{P\}]$. Además, **x** e **y** son vecinos 8N de P y no están 8N conectados en $S \cap [N(P) - \{P\}]$, pues cualquier camino 8N que los una, necesariamente debe pasar por P_1 o por P_2.
En consecuencia, **x** e **y** pertenecen a diferentes componentes 8N conexas de $S \cap [N(P) - \{P\}]$ que son adyacentes a P, lo cual es una contradicción. ■

Los puntos que verifican las propiedades de la proposición anterior juegan un papel relevante en el proceso de afinamiento de un conjunto. Por esta razón reciben un nombre especial, que citamos a continuación.

Definición 3.14: Punto Simple

Sea S un subconjunto de una imagen digital Π ; *un punto* $P \in S$ *es "simple" si verifica las propiedades de la Proposición 3.13.*

Algunos puntos son fáciles de reconocer si son simples o no, como veremos a continuación.

Proposición 3.14:

Sea S un subconjunto de una imagen digital Π *y* $P \in S$. *Entonces:*

a) Si P es un punto" aislado"(no tiene otros vecinos en S) entonces P no es simple.

b) Si P es un punto "interior" (tiene todos sus vecinos 8N en S) entonces P no es simple.

c) Si P es un punto "final" (tiene exactamente un vecino en S) entonces P es simple.

d) Si S es simplemente conexo y P es simple entonces S – { P } es simplemente conexo.

e) Si S es conexo y P es simple entonces S – {P} es también conexo.

Demostración

a) Si P es un punto aislado, entonces $S \cap N(P) = \{P\} \ \wedge \ S \cap [N(P) - \{P\}] = \varnothing$.
Por lo tanto, no se verifica la Proposición 3.13 a) y, en consecuencia, P no es simple.

b) En este caso, como $N(P) \subset S$, resulta $\overline{\mathbf{S}} \cap N(P) = \varnothing \ \wedge \ [\overline{\mathbf{S}} \cap N(P)] \cup \{P\} = \{P\}$.
Luego, al no verificarse la Proposición 3.13 b), P no es simple.

c) Sea P_1 el único vecino de P que está en S. Por lo tanto,
$S \cap N(P) = \{P, P_1\} \wedge \ S \cap [N(P) - \{P\}] = \{P_1\}$. Ambos conjuntos tienen una sola componente (en el sentido de S) y por lo tanto, de la Proposición 3.13 a), P resulta simple.

d) Como S es simplemente conexo, resulta $\overline{\mathbf{S}}$ conexo. Por otro lado, como P es simple, de la Proposición 3.13 e) se tiene que $\overline{\mathbf{S}} \cup \{P\}$ es también conexo. En consecuencia, S – {P} es simplemente conexo pues $\overline{\mathbf{S} - \mathbf{\{P\}}} = \overline{\mathbf{S}} \cap \overline{\overline{\mathbf{\{P\}}}} = \overline{\mathbf{S}} \cup \mathbf{\{P\}} = \overline{\mathbf{S}} \cup \mathbf{\{P\}}$

e) De la Proposición 3.13 e), S-{P} tiene una sola componente conexa, pues S es conexo. En consecuencia, S – {P} es también conexo. ■

Cabe preguntarnos en qué tipo de conjuntos podemos encontrar siempre puntos simples. La respuesta está dada en la siguiente Proposición:

Proposición 3.15:

Sea S un subconjunto simplemente conexo con más de dos puntos. Entonces S tiene al menos dos puntos simples.

Demostración

Se aplica el Principio de Inducción sobre la cantidad de puntos que tiene el subconjunto S. Los detalles de la demostración serán omitidos. ■

Veremos a continuación cómo se lleva a cabo el proceso de "afinado" o "adelgazamiento" de ciertos conjuntos.

Afinamiento de conjuntos simplemente conexos

Dado un subconjunto $S \subset \Pi$ simplemente conexo, quisiéramos reducirlo a un conjunto que sea lo "más delgado posible", pero sin que pierda la propiedad topológica de "simple conexión". Pues bien, sabemos que los arcos son los conjuntos simplemente conexos más delgados. ¿Podríamos adelgazar al conjunto, hasta convertirlo en un arco?

Por un lado, de la Proposición 3.15 podemos asegurar que un conjunto simplemente conexo (con más de dos puntos) tiene al menos dos puntos simples. Por otro lado, de la Proposición 3.14 d) sabemos que, si $P \in S$ es un punto simple, $S - \{P\}$ sigue siendo simplemente conexo. De este modo, podríamos eliminar a dicho punto del conjunto y repetir este proceso de eliminación de puntos simples, pero ¿hasta cuándo? ¿Cómo darnos cuenta en qué momento debemos detener el proceso, si es que logramos convertir al conjunto S en un arco? Afortunadamente podemos dar una respuesta, ya que existe una manera de identificar un arco en términos de puntos simples, y es la siguiente.

Teorema 3.16:

Un subconjunto $S \subset \Pi$ es un arco $\Leftrightarrow$ es simplemente conexo y tiene exactamente dos puntos simples.

Demostración

Supongamos que S es el arco formado por los puntos $P_0, P_1, \ldots, P_n$. Sólo probaremos la siguiente implicación, para el caso en que S sea un arco 4N:

$\Rightarrow$)

De la Proposición 3.6, sabemos que S es simplemente conexo.

Como P_0 y P_n tienen exactamente un vecino en S, éstos son puntos finales. En consecuencia, de la Proposición 3.14 c), resulta que ambos puntos son simples.

Por otro lado, si $1 \leq i \leq n-1$, P_{i-1} y P_{i+1} son los dos únicos vecinos 4N de P_i que están en S. Luego,

$S \cap N(P_i) = \{ P_{i-1}, P_i, P_{i+1} \}$ tiene una sola componente 4N conexa.
$S \cap [N(P_i) - \{ P_i \}] = \{ P_{i-1\ i}, P_{i+1} \}$ tiene dos componentes 4N conexas.

Por lo tanto, de la Proposición 3.13 a) se tiene que P_i no es simple. ■

Del Teorema anterior podemos afirmar que debemos detener el proceso de eliminación de puntos simples cuando sólo le queden al conjunto dos de ellos, pues así lo habremos reducido a un arco.
Esto lo podemos lograr si eliminamos todos los puntos simples del conjunto, salvo aquéllos que son "puntos finales", debido a que éstos son los candidatos a ser los dos puntos simples del arco.
De lo anteriormente expuesto podemos elaborar el siguiente algoritmo:

Algoritmo para afinar un conjunto simplemente conexo

Sea S un subconjunto simplemente conexo de una imagen digital de M x N píxeles. Consideramos conectividad 4N para S. Entonces

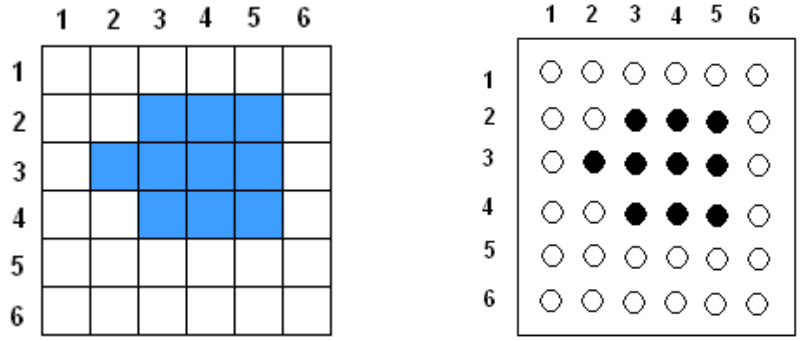
```
Para i = 1 hasta N  hacer
    Para j = 1 hasta M  hacer
        P = (i, j)
        Si P∈  S entonces
            Si P es un "punto final", no eliminarlo.
              Sino
                Si  S ∩ N(P)  tiene la misma cantidad de componentes 4N conexas que
                     S ∩ [N(P) – {P}] entonces eliminar P pues es un punto simple.
                Fin Si
            Fin Si
        Fin Si
    Fin Para
Fin Para
```

En caso de considerar conectividad 8N para S, el algoritmo es similar. El siguiente ejemplo ayudará a comprender cómo se lleva a cabo el afinamiento para este tipo de conjuntos.

Ejemplo 3.13

Sea el siguiente conjunto simplemente conexo perteneciente a una imagen digital de 6 x 6 píxeles definido por:

S = {(3,2), (4,2), (5,2), (2,3), (3,3), (4,3), (5,3), (3,4), (4,4), (5,4)}



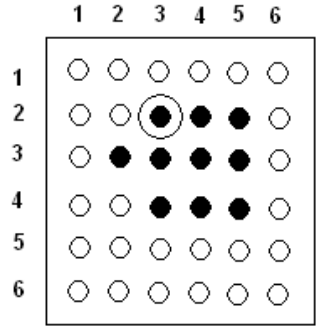
Considerando para S conectividad 4N, aplicar el Método de Afinamiento a fin de reducirlo a un arco.

Resolución:

Comenzaremos recorriendo el conjunto S por filas, a partir de la primera columna.
El primer punto que encontramos es P = (3, 2). Este no es un punto final, de modo que calculamos:

S ⋂ N(P) = {(3,2), (4,2), (2,3), (3,3), (4,3)} tiene una componente 4N conexa.
S ⋂ [N(P) – {P}] = {(4,2), (2,3), (3,3), (4,3)} tiene una componente 4N conexa.

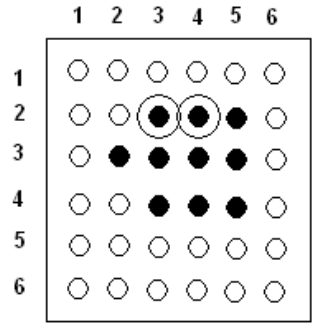
Luego P es simple, con lo cual procedemos a eliminarlo. Marcaremos sobre él un círculo, para indicar que ha sido borrado, según se muestra en la figura:



Análisis para P = (4, 2)

S ⋂ N(P) = {(4,2), (5,2), (3,3), (4,3), (5,3)} tiene una componente 4N conexa.
S ⋂ [N(P) – {P}] = {(5,2), (3,3), (4,3), (5,3)} tiene una componente 4N conexa.

En consecuencia, P también es simple y lo eliminamos:



Análisis para P = (5,2)

S ⋂ N(P) = {(5,2), (4,3),(5,3)} tiene una componente 4N conexa.
S ⋂ [N(P) – {P}] = {(5,2), (4,3),(5,3)} tiene una componente 4N conexa.

Luego P también es simple y lo eliminamos.

Análisis para P = (2,3)

Este es un punto final, pues tiene sólo un vecino 4N en el nuevo conjunto, que es (3,3). Por lo tanto no lo debemos eliminar.

Análisis para P = (3,3)

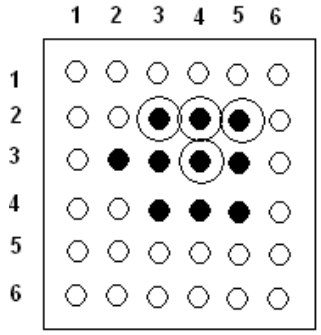
S ⋂ N(P) = {(3,3), (2,3), (3,4), (4,3), (4,4)} tiene una componente 4N conexa.
S ⋂ [N(P) – {P}] = {(2,3), (3,4), (4,3), (4,4)} tiene dos componentes 4N conexas.

Por lo tanto P no es simple y por esta razón no lo eliminamos.

Análisis para P = (4,3)

S ⋂ N(P) = {(4,3), (3,3), (3,4), (4,4), (5,3), (5,4)} tiene una componente 4N conexa.
S ⋂ [N(P) – {P}] = {(3,3), (3,4), (4,4), (5,3), (5,4)} tiene una componente 4N conexa.

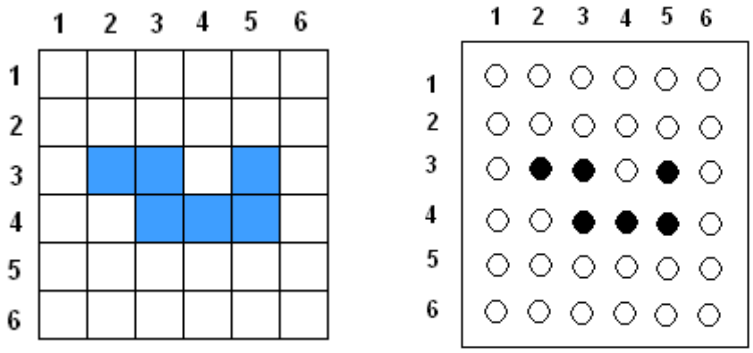
Luego P es simple y lo eliminamos.



Análisis para P = (5,3)

Es un punto final, ya que tiene (5,4) es el único vecino 4N que tiene en el nuevo conjunto. Por lo tanto no lo eliminamos.

Se puede verificar que los restantes puntos (3,4), (4,4) y (5,3) no son simples. De este modo, hemos reducido el conjunto S al siguiente arco:



Afinamiento de conjuntos conexos con un solo agujero

Dado un subconjunto $S \subset \Pi$ conexo y con un solo agujero, quisiéramos también transformarlo en un conjunto que sea lo "más delgado posible", sin que pierda la propiedad de "conexión" y que, además, siga teniendo un solo agujero. Sabemos que las curvas son los conjuntos más delgados que son conexos y que tienen sólo un agujero. ¿Podríamos adelgazar al conjunto, hasta reducirlo a una curva?

Nuevamente, valiéndonos de la Proposición 3.14 e), sabemos que si $P \in S$ es un punto simple, entonces S – {P} sigue siendo conexo. Por otro lado, de la Proposición 3.13 e), $\overline{\mathbf{S}-\mathbf{\{P\}}} = \overline{\mathbf{S}} \cup \mathbf{\{P\}}$ tiene la misma cantidad de componentes conexas que $\overline{\mathbf{S}}$, es decir, S – {P} sigue teniendo un solo agujero. Ahora, si eliminamos todos los puntos simples, ¿lograremos convertir al conjunto S en una curva? Felizmente podemos dar una respuesta, ya que existe una manera de identificar a una curva en términos de puntos simples, y es la que se enuncia en el siguiente Teorema.

Teorema 3.17:

Un subconjunto $S \subset \Pi$ es una curva $\Leftrightarrow$ es conexo, tiene exactamente un agujero y no tiene puntos simples.

Demostración

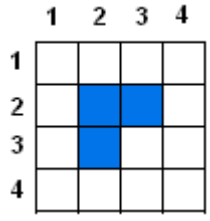
Sólo probaremos la siguiente implicación, para el caso en que S sea una curva 4N:

$\Rightarrow$)

Por definición de curva, S es un conjunto conexo. Por otro lado, de la Proposición 3.9 resulta que S tiene exactamente un agujero. La demostración de que la curva no tiene puntos simples es similar a la del Teorema anterior, pues cada punto de la misma tiene exactamente dos vecinos 4N que están en S. ■

Observación

El teorema anterior puede dejar de ser cierto si se considera una curva 8N con sólo tres puntos. Por ejemplo, el conjunto S = {(2,3), (2,2), (3,2)} es una curva 8N y fácilmente se puede comprobar que todos sus puntos son simples.



Del Teorema anterior podemos afirmar que debemos eliminar todos los puntos simples del conjunto, pues así lo habremos reducido a una curva.
Lo anteriormente expuesto nos lleva a elaborar el siguiente algoritmo:

Algoritmo para afinar un conjunto conexo y sin agujeros

Sea S un subconjunto simplemente conexo de una imagen digital de M x N píxeles. Consideramos conectividad 4N para S. Entonces

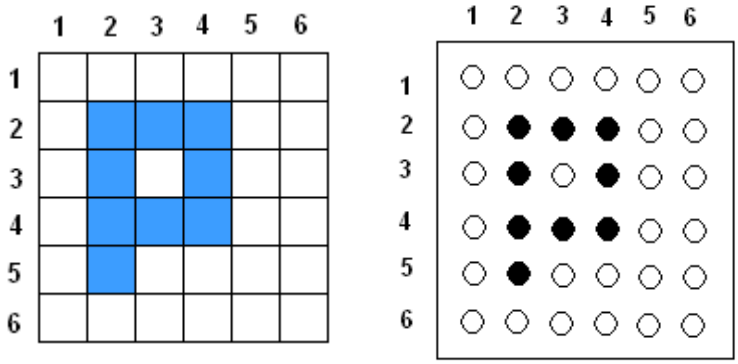
```
Para i = 1 hasta N  hacer
    Para j = 1 hasta M  hacer
        P = (i, j)
        Si P∈ S entonces
            Si  S ⋂ N(P)  tiene la misma cantidad de componentes 4N conexas que
                S ⋂ [N(P) – {P}] entonces eliminar P pues es un punto simple.
            Fin Si
        Fin Si
    Fin Para
Fin Para
```

En caso de considerar conectividad 8N para S, el algoritmo es similar. Finalizaremos este capítulo con un ejemplo que mostrará la manera de afinar este tipo de conjuntos.

Ejemplo 3.14

Considere el siguiente conjunto conexo y con un agujero, perteneciente a una imagen digital de 6 x 6 píxeles definido por:

S = {(2,5), (2,4), (2,3), (2,2), (3,2), (4,2), (4,3), (4,4), (3,4)}



Considerando para S conectividad 4N, aplicar el Método de Afinamiento a fin de reducirlo a una curva.

Resolución:

Comenzaremos recorriendo el conjunto S por filas, a partir de la primera columna.
El primer punto que encontramos es P = (2, 2). Calculamos:

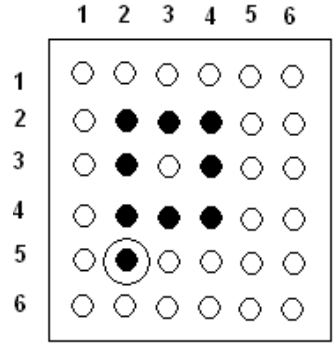
$S \cap N(P) = \{(2,2) , (3,2) , (2,3)\}$ tiene una componente 4N conexa.
$S \cap [N(P) - \{P\}] = \{(3,2) , (2,3)\}$ tiene dos componentes 4N conexas.

Luego P no es punto simple y por lo tanto no lo eliminamos.

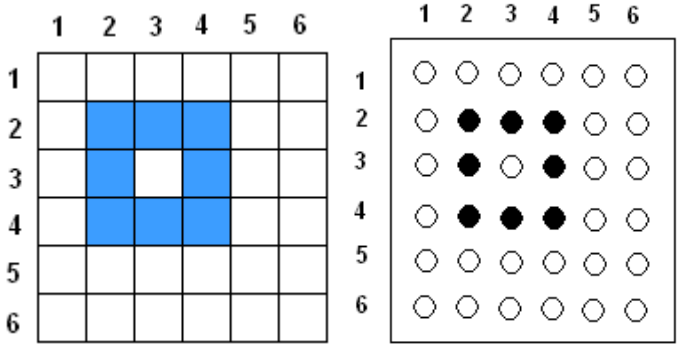
Repitiendo el análisis para los puntos (3, 2), (4,2), (2, 3), (4, 3), (2, 4), (3, 4) y (4,4), se puede verificar que ninguno de ellos es simple, razón por la cual no los eliminamos.
Finalmente, si P = (2, 5) entonces

$S \cap N(P) = \{(2,5) , (2,4) , (3,4)\}$ tiene una componente 4N conexa.
$S \cap [N(P) - \{P\}] = \{(2,4) , (3,4)\}$ tiene una componentes 4N conexa.

Luego P es punto simple y en consecuencia lo eliminamos.



De este modo, hemos reducido el conjunto S la siguiente curva:



En el capítulo siguiente, basados en los resultados desarrollados en este trabajo, describiremos el Algoritmo BF4/BF8, el cual permitirá identificar y recorrer la frontera de un subconjunto de una imagen digital.

Capítulo 4

El algoritmo BF4 / BF8

Capítulo 4: El algoritmo BF4 / BF8

Durante el procesamiento de imágenes digitales, existen situaciones en las cuales un objeto puede ser reconocido a través de su borde. Por ejemplo, en el fútbol de robots, para localizar y mover la pelota sólo se necesita identificar y recorrer el borde de la misma.

En este capítulo mostramos un algoritmo que permitirá recorrer el borde de un objeto. También probaremos que este algoritmo es válido, gracias a los resultados teóricos del capítulo anterior. Explicaremos, además, cómo reconstruir el objeto a partir de su borde. Pero primeramente debemos definir el concepto de "borde" de un subconjunto de una imagen digital. Como vimos en el Ejemplo 2.3 del Capítulo 2 (pág. 26), la "frontera" de un subconjunto, según la topología de la imagen digital Π, puede no coincidir con la noción intuitiva de "borde". Por esta razón introduciremos la siguiente definición:

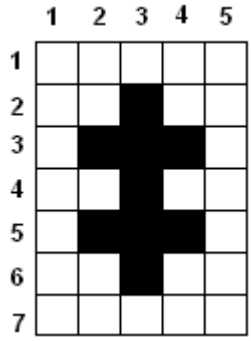
Definición 4.1: Borde de un subconjunto de una imagen digital.

Dado un subconjunto S de una imagen digital Π*, el "borde de S", que lo simbolizaremos por Borde(S), es el conjunto de puntos de S que tienen vecinos 4N en* $\overline{\mathbf{S}}$.

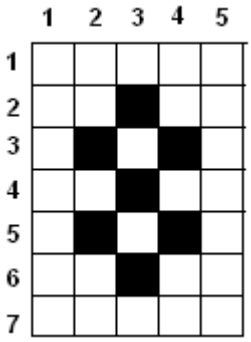
También se podría definir una frontera o borde más grueso, compuesto por los puntos que tienen vecindad 8N en $\overline{\mathbf{S}}$. Pero la definición basada en vecindad 4N es la generalmente utilizada. El siguiente ejemplo ilustra este concepto.

Ejemplo 4.1

Sea S el conjunto de píxeles negros de la imagen digital que muestra la figura:



Se observa que (3,3) y (3, 5) son los únicos píxeles de S que no tienen vecindad 4N con $\overline{\mathbf{S}}$.Por lo tanto, Borde(S) es el conjunto de píxeles negros que se indican a continuación:



El borde de *S* consta, en general, de muchas partes, ya que *S* puede tener varias componentes conexas, y cada una de ellas puede poseer o no agujeros. Por esta razón, determinaremos la frontera de cada componente de S con respecto a cada componente de $\overline{\mathbf{S}}$. Previamente necesitaremos la siguiente definición:

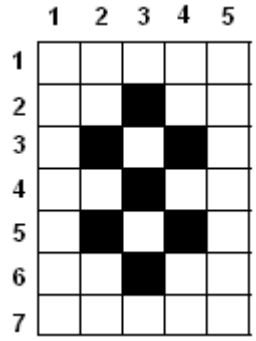
Definición 4.2: Borde de una componente con respecto a una componente de su complemento.

Sea C una componente conexa de S y D una componente conexa de $\overline{\mathbf{S}}$. *El borde de C con respecto a D es el conjunto definido por*

$$C_D = \{P \in C : P \text{ tiene al menos un vecino 4N en } D\}.$$

Ejemplo 4.2

Sea S el conjunto de píxeles negros que figura en la siguiente imagen digital:



En este caso, si consideramos conectividad 8N para S y conectividad 4N para su complemento, tenemos:

Componentes 8N conexas de S: Hay una pues S es 8N conexo. Es decir, S = C.

Componentes 4N conexas de $\overline{\mathbf{S}}$*:* Hay tres: D_1 =Fondo (S) , D_2 = {(3,3)} , D_3= {(3,5)}

Luego, $\mathbf{C_{D_1}}$ = {(3,2), (2,3), (4,3), (3,4)}, $\mathbf{C_{D_2}}$ = {(3,4), (2,5), (4,5), (3,6)}.

Presentaremos a continuación el Algoritmo BF4, que permitirá hallar el borde C_D.

El Algoritmo BF4

Este algoritmo considera conectividad 4N para C y conectividad 8N para D. Supondremos que C tiene más de un punto y que como dato inicial se nos da un par ordenado de puntos (P_0, Q_0) donde $P_0 \in C$, $Q_0 \in D$ y P_0 es vecino 4N de Q_0.
El algoritmo especifica cómo obtener un nuevo par (P_{i+1}, Q_{i+1}) a partir de (P_i, Q_i), con $P_{i+1} \in C$, $Q_{i+1} \in D$ y P_{i+1} vecino 4N de Q_{i+1}. De este modo se visitarán todos los puntos del borde C_D.
Explicaremos más detalladamente su funcionamiento:

Funcionamiento del Algoritmo BF4

Inicializamos el valor i = 0 y consideramos los datos iniciales $P_0 \in C$ y $Q_0 \in D$ con P_0 vecino 4N de Q_0.

Tomamos los vecinos 8N de P_i, ordenados en sentido horario y comenzando con Q_i.

A estos vecinos los llamaremos $R_{i\,1} = Q_i$, R_{12},…., $R_{i\,8}$.

Sea $j = \min\{k : R_{i\,k} \in C \wedge R_{i\,k}$ es vecino 4N de Pi$\}$

Consideremos $R_{i\,j}$.

Si $R_{i\,j-1} \in D$ entonces definir

$$\begin{cases} P_{i+1} = R_{i\,j} \\ Q_{i+1} = R_{i\,j-1} \end{cases}$$

Sino (es decir, si $R_{i\,j-1} \notin D$) definir

$$\begin{cases} P_{i+1} = R_{i\,j-1} \\ Q_{i+1} = R_{i\,j-2} \end{cases}$$

Incrementar en 1 el valor de i.

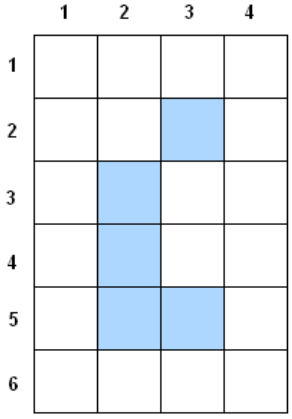
Repetimos el proceso hasta que $P_i = P_0$ y $Q_i = Q_0$.

De este modo, $C_D = \{P_0, P_1, P_2, \ldots.\}$

El siguiente ejemplo ilustrará cómo aplicar el Algoritmo BF4.

Ejemplo 4.3

Sea S el conjunto de píxeles celestes de la siguiente imagen digital:



Considerando conectividad 4N para S y conectividad 8N para $\overline{\mathbf{S}}$, tenemos que S tiene dos componentes 4N conexas, que son:

$$C = \{(2,3), (2,4), (2,5), (3,5)\} \quad \text{y} \quad C_1 = \{(3,2)\}$$

Por otro lado, $\overline{\mathbf{S}}$ tiene una sola componente 8N conexa, es decir, $\overline{\mathbf{S}}$ = Fondo(S) = D.

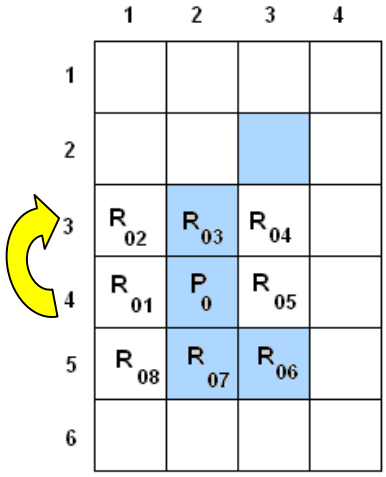
Aplicar el Algoritmo BF4 para determinar C_D, tomando como dato inicial (P_0, Q_0) donde $P_0 = (2,4)$ y $Q_0 = (1,4)$

Resolución

Primer Paso

En este caso, $P_0 = (2,4)$ y $Q_0 = (1,4)$. Tomamos los vecinos 8N de P_0, ordenados en sentido horario y comenzando con Q_0.

A estos vecinos los llamamos $R_{01} = Q_0$, R_{02},…., R_{08}, según se muestra en la siguiente figura:



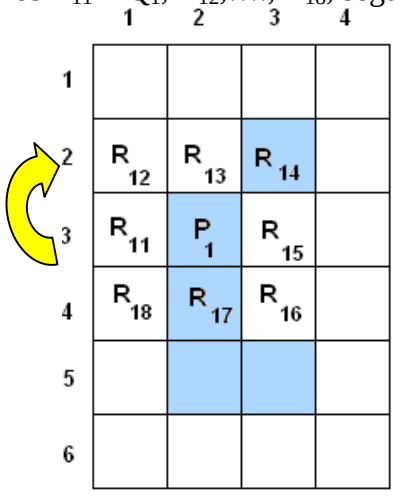
Observamos que R_{03} es el primero de estos vecinos 8N que está en C y es vecino 4N de P_0.Como $R_{02} \in$ D, tomamos

$$\begin{cases} P_1 = R_{03} = (2, 3) \\ Q_1 = R_{02} = (1, 3) \end{cases}$$

Como $P_1 \neq P_0$ y $Q_1 \neq Q_0$, el proceso continúa.

Segundo Paso

Ahora consideramos, $P_1 = (2,3)$ y $Q_1 = (1,3)$. Tomamos los vecinos 8N de P_1, ordenados en sentido horario y comenzando con Q_1.
A estos vecinos los llamamos $R_{11} = Q_1$, R_{12},…., R_{18}, según se ve en la siguiente figura:



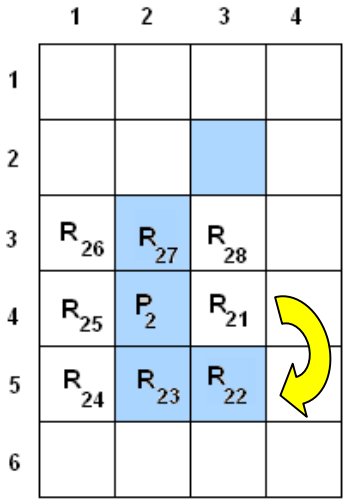
En este caso, R_{17} es el primero de estos vecinos 8N que está en C y es vecino 4N de P_1.Como $R_{16} \in$ D, tomamos

$$\begin{cases} P_2 = R_{17} = (2, 4) \\ Q_2 = R_{16} = (3, 4) \end{cases}$$

Como $P_2 = P_0$ pero $Q_2 \neq Q_0$, el proceso continúa.

Tercer Paso

Tomamos $P_2 = (2,4)$ y $Q_2 = (3,4)$. Consideramos los vecinos 8N de P_2, ordenados en sentido horario y comenzando con Q_2.
A estos vecinos los llamamos $R_{21} = Q_1$, R_{22},…., R_{28}, según se observa en la figura:



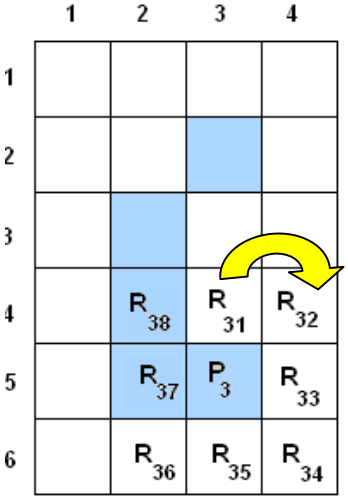
Ahora, R_{23} es el primero de estos vecinos 8N que está en C y es vecino 4N de P_2.Como $R_{22} \notin D$, tomamos

$$\begin{cases} P_3 = R_{22} = (3, 5) \\ Q_3 = R_{21} = (3, 4) \end{cases}$$

Como $P_3 \neq P_0$ y $Q_3 \neq Q_0$, el proceso continúa.

Cuarto paso

Tomamos $P_3 = (3,5)$ y $Q_3 = (3,4)$. Consideramos los vecinos 8N de P_3, ordenados en sentido horario y comenzando con Q_3.
A estos vecinos los llamamos $R_{31} = Q_3$, R_{32},…., R_{38}, según se ve en la figura:



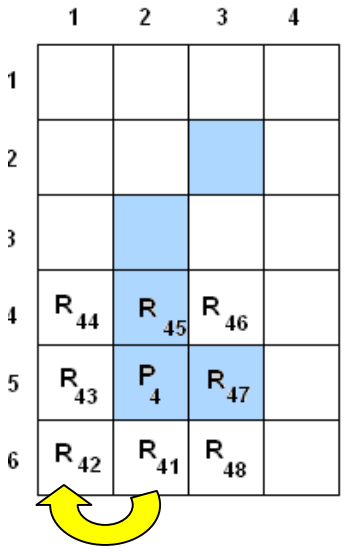
Ahora, R_{37} es el primero de estos vecinos 8N que está en C y es vecino 4N de P_3.Como $R_{36} \in D$, tomamos

$$\begin{cases} P_4 = R_{37} = (2, 5) \\ Q_4 = R_{36} = (2, 6) \end{cases}$$

Como $P_4 \neq P_0$ y $Q_4 \neq Q_0$, el proceso continúa.

Quinto Paso

Tomamos $P_4 = (2,5)$ y $Q_4 = (2,6)$. Consideramos los vecinos 8N de P_3, ordenados en sentido horario y comenzando con Q_4.
A estos vecinos los llamamos $R_{41} = Q_4$, R_{42},…., R_{48}, según se muestra en la siguiente figura:

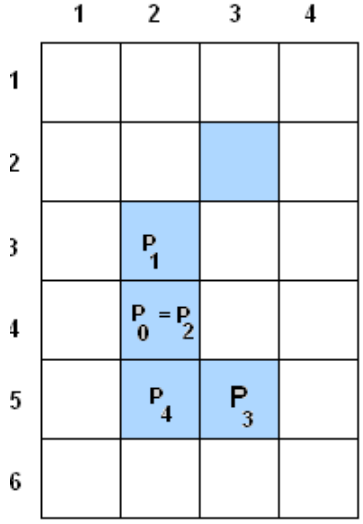


Ahora, R_{45} es el primero de estos vecinos 8N que está en C y es vecino 4N de P_4.Como $R_{44} \in D$, tomamos

$$\begin{cases} P_5 = R_{45} = (2, 4) \\ Q_5 = R_{44} = (1, 4) \end{cases}$$

Como $P_5 = P_0$ y $Q_5 = Q_0$, el proceso se detiene.

Luego, el conjunto $\{P_0, P_1, P_2, P_3, P_4\}$ es ciertamente C_D, como era de esperar.



En el ejemplo anterior se comprobó que el Algoritmo BF4 visita y obtiene todos los puntos de C_D. Pero ¿cómo podemos probar analíticamente que la secuencia de puntos $\{P_0, P_1,\dots\}$ es efectivamente el conjunto C_D? La siguiente Proposición lo demuestra.

Proposición 4.1:

Sea S un subconjunto de una imagen digital Π *. Consideramos conectividad 4N para S mientras que, para* $\overline{\mathbf{S}}$*, conectividad* 8N*. Sea C una componente 4N conexa de S con al menos dos puntos, D una componente 8N conexa de* $\overline{\mathbf{S}}$ *y* C_D *es el conjunto definido por*

$$C_{\mathbf{D}} = \{P \in C : P \text{ tiene al menos un vecino 4N en } D\}$$

Sea $P_0 \in$ C, $Q_0 \in$ D *con* P_0 *vecino 4N de* Q_0*. Si* $\{P_0, P_1,\ldots\} \wedge \{Q_0, Q_1,\ldots\}$ *es la secuencia de puntos obtenidos mediante el Algoritmo BF4 entonces*

a) $\{P_0, P_1,\ldots\} \subset C_{\mathbf{D}} \wedge \{Q_0, Q_1,\ldots.\} \subset D$

b) P_i *es vecino 4N de* Q_i *para* $i = 0, 1,\ldots.$

c) $\{P_0, P_1,\ldots\} = C_{\mathbf{D}}$

Demostración

Probaremos a) y b) aplicando el Principio de Inducción sobre "n", la cantidad de puntos obtenidos mediante el Algoritmo BF4.

Para n = 1

Tomamos los vecinos 8N de P_0, ordenados en sentido horario y comenzando con Q_0.
A éstos los llamamos $R_{01} = Q_0, R_{02},\ldots., R_{08}$.
Observemos que R_{0k} es vecino 4N de P_0 si y sólo si "k" es impar. Sea

$$j = \min\{k : R_{0\,k} \in \mathbf{C} \wedge R_{0\,k} \text{ es vecino 4N de } P_0\}$$

Notemos que R_{0j} existe, pues **C** es 4N conexo y tiene más de un punto. Advertimos, además, que "j" es impar.

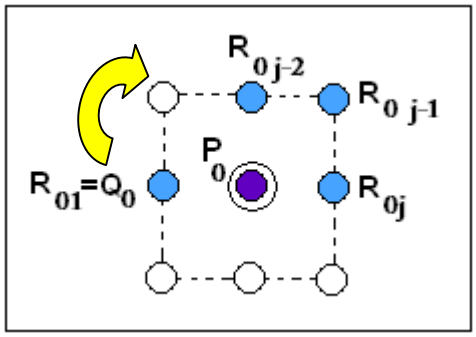


Figura 4.1: Posible disposición de los puntos $R_{0\,1}$ *,* $R_{0\,j-2}$ *,* $R_{0\,j-1}$ *y* $R_{0\,j}$ *.*

Se presentan dos casos:

1) Si $R_{0\,j-1} \in$ **D** entonces definimos

$$\begin{cases} P_1 = R_{0\,j} \\ Q_1 = R_{0\,j-1} \end{cases}$$

Obviamente, $Q_1 \in$ **D**, $P_1 \in$ $\mathbf{C_D}$ $\wedge$ P_1 es vecino 4N de Q_1

2) Si $R_{0\,j-1} \notin$ **D** entonces

Como j-2 es impar, $R_{0\,j-2}$ es vecino 4N de P_0.

Luego podemos asegurar que $R_{0\,j-2} \notin \mathbf{S}$, pues si perteneciera a este conjunto, al estar 4N conectado con P_0, debería estar en **C.** Pero esto es imposible ya que R_{0j} es el primero de los vecinos 4N de P_0 que está en **C**.

En consecuencia, $R_{0\,j-2} \in \overline{\mathbf{S}}$. Más aún, $R_{0\,j-2} \in \mathbf{D}$ ya que $Q_0 = R_{01}, R_{03},\ldots,R_{0\,j-2}$ es un camino 8N en $\overline{\mathbf{S}}$ que conecta Q_0 con $R_{0\,j-2}$.

Por otro lado, $R_{0\,j-1} \notin \overline{\mathbf{S}}$ pues si lo estuviera, estaría conectado con Q_0 a través del camino 8N en $\overline{\mathbf{S}}$ definido por $Q_0 = R_{01}, R_{03},\ldots,R_{0\,j-2}, R_{0\,j-1}$, pero esto no es posible debido a que $R_{0\,j-1} \notin \mathbf{D}$.

Por lo tanto, $R_{0\,j-1} \in \mathbf{S}$. Pero también $R_{0\,j-1} \in \mathbf{C}$ pues está 4N conectado con P_0 mediante el camino $P_0, R_{0j}, R_{0\,j-1}$. De este modo, definimos

$$\begin{cases} P_1 = R_{0\,j-1} \\ Q_1 = R_{0\,j-2} \end{cases}$$

Claramente $Q_1 \in \mathbf{D}$, $P_1 \in \mathbf{C_D} \wedge P_1$ es vecino 4N de Q_1.

Supongamos, por hipótesis inductiva, que $\{P_0, P_1,\ldots P_n\} \subset \mathbf{C_D} \wedge \{Q_0, Q_1,\ldots Q_n\} \subset \mathbf{D}$ y que además P_i es vecino 4N de Q_i para i = 0, 1,…,n.

Veamos para n+1

La verificación de que $Q_{n+1} \in \mathbf{D}$, $P_{n+1} \in \mathbf{C_D} \wedge P_{n+1}$ es vecino 4N de Q_{n+1} es exactamente igual al caso n = 1.

Probemos ahora el inciso c)

Podemos afirmar que el funcionamiento del Algoritmo BF4 no se ve afectado si todos los puntos de $\overline{\mathbf{S}}$, excepto aquellos de **D** y del **Fondo(S),** se transfieren de $\overline{\mathbf{S}}$ a **S**, por lo que, sin pérdida de generalidad, podemos suponer que **C** tiene a lo sumo un agujero.
De este modo se presentan dos situaciones:

1) Si **C** es simplemente conexo

En este caso $\mathbf{D} = \overline{\mathbf{S}}$. Aplicaremos el Principio de Inducción sobre “n”, es decir, sobre la cantidad de puntos que tiene **C**.

Para n = 2

El conjunto **C** está formado por dos puntos, o sea, **C** = {**a**, **b**}
Como **C** es 4N conexo, **a** y **b** son vecinos 4N. Los demás vecinos 4N de **a** y **b** están en $\overline{\mathbf{S}}$, pues si estuvieran en **S**, pertenecerían a **C** (ya que están 4N conectados con **a** y **b**), lo cual es imposible pues **C** tiene sólo dos puntos.
Por lo tanto, $\mathbf{C_D} = \mathbf{C} = \{\mathbf{a}, \mathbf{b}\}$
Tomando P_0 = **a**, el Algoritmo BF4 permite obtener $P_1 \in \mathbf{C_D}$. En consecuencia, $\{P_0, P_1\} \subset \mathbf{C_D} = \{\mathbf{a}, \mathbf{b}\}$, de donde resulta que $\{P_0, P_1\} = \mathbf{C_D}$.

Supongamos, por hipótesis inductiva, que si **C** es una componente simplemente conexa y tiene “n” puntos, entonces $\mathbf{C_D}$ es el conjunto formado por los puntos $\{P_0, P_1,\ldots\}$ que resultan de aplicar el Algoritmo BF4 al conjunto **C**.

Veamos para n+1

Si **C** tiene (n+1) puntos, como **C** es simplemente conexo, de la Proposición 3.15 sabemos que **C** tiene un punto simple **P**. Luego podemos escribir

$\mathbf{C} = \mathbf{B} \cup \{\mathbf{P}\}$
Donde $\mathbf{B} = \mathbf{C} - \{\mathbf{P}\}$

Si definimos $\mathbf{B}_{\mathbf{D}} = \{b \in \mathbf{B}$: b tiene al menos un vecino 4N en $\mathbf{D}\}$
Entonces

$$\mathbf{C_D} = \begin{cases} B_D & \text{si P no tiene vecinos 4N en D} \\ B_D \cup \{P\} & \text{si P tiene algún vecino 4N en D} \end{cases}$$

Se presentan los siguientes casos:

1.a) Si **P** tiene algún vecino 4N en **D** entonces aplicamos al conjunto **C** el primer paso del Algoritmo BF4, tomando como par inicial a (P_0, Q_0) donde $P_0 = \mathbf{P}$. De este modo obtenemos el par (P_1, Q_1) con $P_1 \in \mathbf{C} - \{\mathbf{P}\} = \mathbf{B}$. A continuación aplicamos nuevamente el algoritmo al conjunto **B**, tomando ahora como par inicial a (P_1, Q_1). De este modo se obtienen los puntos $\{P_1, P_2, \ldots\}$.
Por otro lado, de la Proposición 3.14 d) y e), **B** sigue siendo conexo y simplemente conexo. Como además tiene "n" puntos, de la hipótesis inductiva resulta que $\{P_1, P_2, \ldots\} = \mathbf{B}_{\mathbf{D}}$. Por lo tanto,

$$\mathbf{C_D} = \mathbf{B_D} \cup \{\mathbf{P}\} = \{P_1, P_2, \ldots\} \cup \{P_0\} = \{P_0, P_1, P_2, \ldots\}$$

1.b) Si **P** no tiene vecinos 4N en **D** entonces aplicamos a **B** el Algoritmo BF4 comenzando con un par inicial (P_0, Q_0) donde $P_0 \in \mathbf{B}$. De este modo obtenemos los puntos $\{P_0, P_1, \ldots\}$.Por hipótesis inductiva resulta que $\mathbf{B_D} = \{P_0, P_1, P_2, \ldots\}$.Luego,

$$\mathbf{C_D} = \mathbf{B_D} = \{P_0, P_1, P_2, \ldots\}.$$

2) Si **C** tiene exactamente un agujero

En este caso $\overline{\mathbf{S}}$ tiene dos componentes: el **Fondo(S)** y el agujero. Llamamos **D** a cualquiera de estas componentes. Se presentan dos casos:

2.a) **C** es una curva

De la proposición 3.12 sabemos que $\mathbf{C} = \mathbf{C_D}$
Tomemos $\mathbf{P} \in \mathbf{C}$, entonces

$$\mathbf{C} = \mathbf{B} \cup \{\mathbf{P}\}$$

Donde $\mathbf{B} = \mathbf{C} - \{\mathbf{P}\}$. Luego $\mathbf{C_D} = \mathbf{B_D} \cup \{\mathbf{P}\} = \mathbf{B} \cup \{\mathbf{P}\}$

Aplicamos al conjunto **C** el primer paso del Algoritmo BF4, tomando como par inicial a (P_0, Q_0) donde $P_0 = \mathbf{P}$. De este modo obtenemos el par (P_1, Q_1) con $P_1 \in \mathbf{C} - \{\mathbf{P}\} = \mathbf{B}$.
A continuación aplicamos nuevamente el algoritmo al conjunto **B**, tomando ahora como par inicial a (P_1, Q_1). De este modo se obtienen los puntos $\{P_1, P_2, \ldots\}$.
Por otro lado, como **B** es un arco, de la Proposición 3.6 sabemos que es simplemente conexo. En consecuencia, de lo recientemente demostrado en la situación 1), resulta $\{P_1, P_2, \ldots\} = \mathbf{B_D}$.
Por lo tanto,

$$\mathbf{C_D} = \mathbf{B_D} \cup \{\mathbf{P}\} = \{P_1, P_2, \ldots\} \cup \{P_0\} = \{P_0, P_1, P_2, \ldots\}$$

2.b) **C** no es una curva

Para este caso aplicaremos el Principio de Inducción sobre "n", la cantidad de puntos simples que tiene **C**.

Para n = 1

Sea **P** el único punto simple de **C**. Nuevamente expresamos a este conjunto de la siguiente manera:

$$\mathbf{C} = \mathbf{B} \cup \{\mathbf{P}\}$$

Donde **B = C – {P}**
En este caso,

$$\mathbf{C_D} = \begin{cases} B_D & \text{si P no tiene vecinos 4N en D} \\ B_D \cup \{P\} & \text{si P tiene algún vecino 4N en D} \end{cases}$$

Observemos que, por las Proposiciones 3.14 e) y 3.13 e), **B** sigue siendo conexo y con un solo agujero. Por otro lado, del Teorema 3.17 resulta que **B** es una curva.

2.b.1) Si **P** no tiene vecinos 4N en **D** entonces, al aplicar a la curva **B** el Algoritmo BF4, la secuencia de puntos obtenida verifica, por el inciso 2.a), que $\mathbf{B_D} = \{P_0, P_1, P_2, \ldots\}$. Por lo tanto,

$$\mathbf{C_D} = \mathbf{B_D} = \{P_0, P_1, P_2, \ldots\}.$$

2.b.2) Si **P** tiene algún vecino 4N en **D**, la demostración es similar a la efectuada en el caso 2.a).

Supongamos, por hipótesis inductiva, que si **C** es una componente conexa con un único agujero y tiene "n" puntos simples, entonces $\mathbf{C_D} = \{P_0, P_1, \ldots\}$ donde $P_0, P_1, \ldots$ es la secuencia de puntos obtenida al aplicar el Algoritmo BF4 al conjunto **C**.

Veamos para n+1

Si **C** tiene (n+1) puntos, tomemos un punto **P** simple. Nuevamente expresamos a **C** de la forma

$$\mathbf{C} = \mathbf{B} \cup \{\mathbf{P}\}$$

Donde **B = C – {P}**

El resultado sigue de aplicar la hipótesis inductiva al conjunto **B**, siguiendo un razonamiento similar a lo demostrado en los casos 1.a) y 1.b). ■

En la próxima sección enunciaremos el Algoritmo BF8, que también permitirá hallar el borde C_D.

El Algoritmo BF8

Este algoritmo, muy similar al BF4, considera conectividad 8N para C y conectividad 4N para D. Aquí simplemente R_{ij} es el primero de los puntos $R_{i\ 1}$, R_{12},…., $R_{i\ 8}$ que pertenece a C. De este modo se define

$$\begin{cases} P_{i+1} = R_{i\,j} \\ Q_{i+1} = R_{i\,j-1} \end{cases}$$

El detalle de su funcionamiento se indica a continuación:

Funcionamiento del Algoritmo BF8

Inicializamos el valor i = 0 y consideramos los datos iniciales $P_0 \in C$ y $Q_0 \in D$ con P_0 vecino 4N de Q_0.

Tomamos los vecinos 8N de P_i, ordenados en sentido horario y comenzando con Q_i.

A estos vecinos los llamaremos $R_{i\,1} = Q_i$, R_{12},…., $R_{i\,8}$.

Sea $j = \min\{k : R_{i\,k} \in C\}$

Consideremos $R_{i\,j}$ y definimos

$$\begin{cases} P_{i+1} = R_{i\,j} \\ Q_{i+1} = R_{i\,j-1} \end{cases}$$

Incrementar en 1 el valor de i.

Repetimos el proceso hasta que $P_i = P_0$ y $Q_i = Q_0$.

De este modo, $C_D = \{P_0, P_1, P_2, \ldots\}$

La demostración de que efectivamente $C_D = \{P_0, P_1, P_2, \ldots\}$ es muy similar a la efectuada en la Proposición anterior.

¿Cómo es almacenado el borde de un conjunto dentro del ordenador?

Una de las grandes ventajas que ofrece el Algoritmo BF4/BF8 es que permite almacenar el borde de un conjunto ocupando un espacio muy reducido en la memoria del ordenador.
Más precisamente, dado que los sucesivos P_i obtenidos por BF4 o BF8 son vecinos 8N, podemos especificar el borde de **C** con respecto a **D** dando la posición inicial de P_0 y una cadena de números de 3 bits (que en sistema decimal representan valores enteros comprendidos entre "0" y "7") que indican la posición de P_{i+1} con respecto a P_i
Uno de los códigos más utilizados para construir dicha cadena es el llamado ***"Código Cadena"***, en el cual el código "k", asociado al punto P_{i+1}, indica girar un ángulo de 45 k grados alrededor de P_i y en sentido anti-horario, como lo ilustra la siguiente figura:

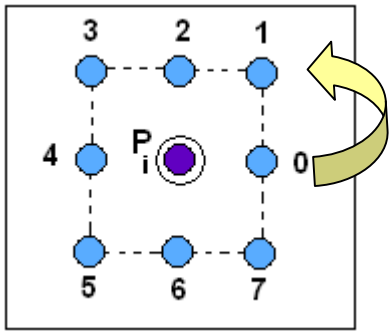


Figura 4.2: Código Cadena para determinar la posición de P_{i+1} con respecto a P_i.

Por ejemplo, si P_{i+1} tiene el código "3", significará que éste está ubicado en la siguiente posición con respecto a P_i:

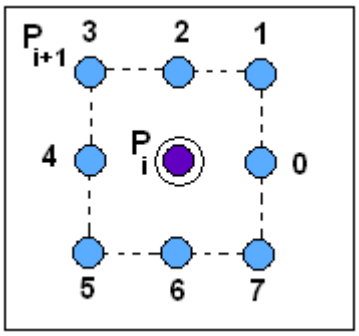


Figura 4.3: Posición de P_{i+1} con respecto a P_i, en caso de tener asociado el código cadena 3.

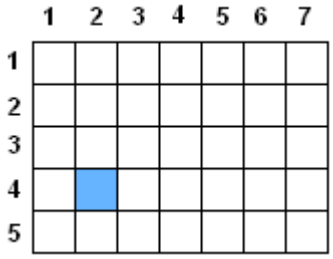
Obviamente, el código cadena de Q_{i+1} será "4". El siguiente ejemplo ilustrará aún más este concepto.

Ejemplo 4.4

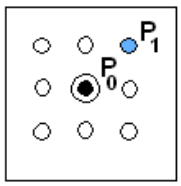
Al aplicar el Algoritmo BF4 a una componente C de una imagen digital de 5 x 7 píxeles, se obtuvo su borde C_D. La información del mismo fue almacenada en el ordenador mediante el punto inicial P_0= (2,4) y el Código Cadena (1, 1, 7, 7, 4, 4, 4). Reconstruir C_D a partir de estos datos.

Resolución

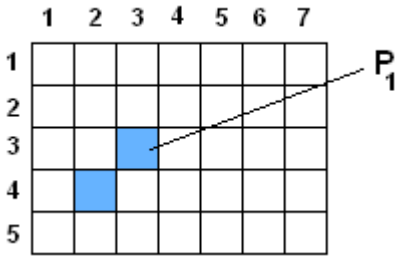
Como el Código Cadena consta de siete elementos, el borde C_D es el conjunto formado por los puntos P_0, P_1, P_2,….,P_7, los cuales fueron obtenidos al implementar el Algoritmo BF4. De este modo, P_1 tiene asociado el código "1", P_2 el código "1", P_3 el código "7" y así sucesivamente.
Primeramente, ubicamos en la imagen digital el píxel P_0 = (2,4) según indica la figura:



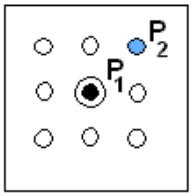
Como P_1 tiene código1, éste se encuentra en la siguiente posición relativa a P_0:



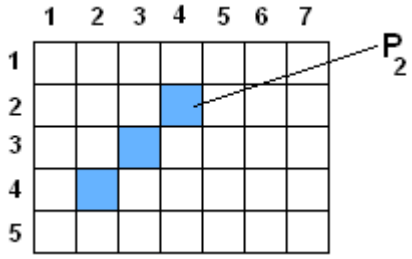
Luego, P_1 queda representado dentro de la imagen digital de la siguiente manera:



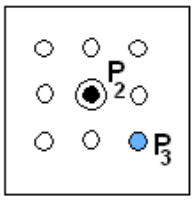
Ahora, como P_2 tiene código1, éste se encuentra en la siguiente posición relativa a P_1:



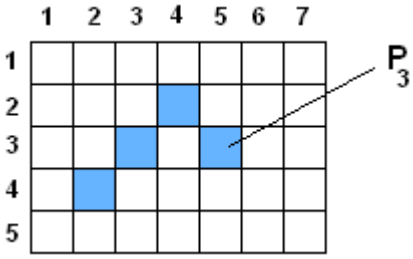
Por lo tanto, la representación de P_2 dentro de la imagen digital es:



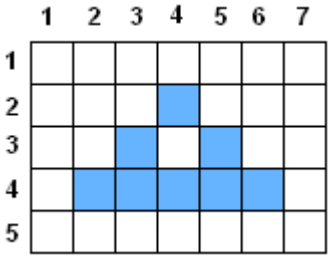
Como P_3 tiene código7, éste se encuentra en la siguiente posición relativa a P_2:



Y su representación es:



Aplicando la misma metodología para los puntos restantes, resulta que el borde C_D es

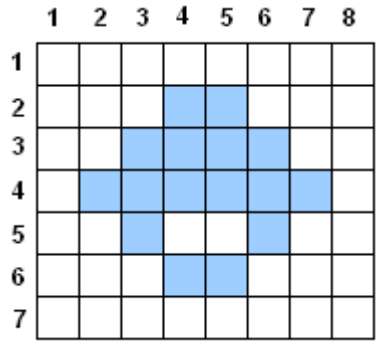


¿Cómo reconstruir un conjunto a partir de su borde?

Para reconstruir *C* a partir de su borde, necesitamos conocer el par de puntos inicial (P_0, Q_0) (proveniente de aplicar el Algoritmo BF4/BF8) y el Código Cadena de cada borde C_D. De este modo se podrán obtener fácilmente los puntos de C_D así como una banda en *D* adyacente a C_D, para cada *D*. Una vez logrado esto, es fácil "colorear" el interior de *C*. Notemos que si no hubiéramos marcado los puntos D que son adyacentes a C, no hubiera sido fácil decidir de qué lado de la curva encontramos el interior o el fondo de la imagen. El siguiente ejemplo mostrará cómo hacerlo.

Ejemplo 4.5

Sea S el conjunto de píxeles celestes de la siguiente imagen digital:



Considerando conectividad 8N para S y conectividad 4N para $\overline{\mathbf{S}}$, tenemos que S es 8N conexo. Luego, S tiene una única componente. Es decir,

S = C.

Por otro lado, $\overline{\mathbf{S}}$ tiene dos componentes 4N conexas:

$$D1 = \{(4, 5), (5, 5)\} \wedge \quad D2 = \Pi - S - D1 = \text{Fondo}(S)$$

En este caso, $\text{Borde}(S) = C_{D1} \cup C_{D2}$

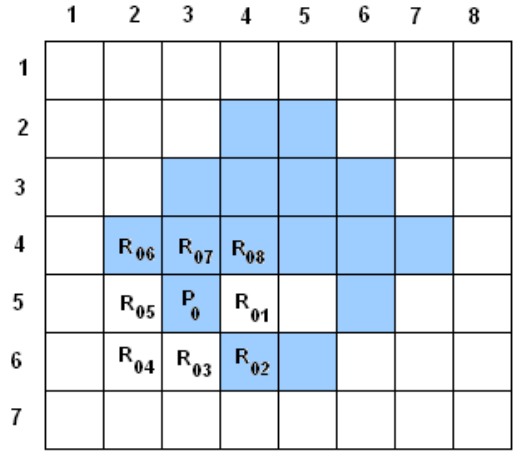
a) Aplicar el Algoritmo BF8 para determinar C_{D1}, tomando como dato inicial (P_0, Q_0) donde $P_0 = (3,5)$ y $Q_0 = (4,5)$. Determinar, además, el Código Cadena asociado a C_{D1}.

b) Aplicar el Algoritmo BF8 para determinar C_{D2}, tomando como dato inicial (P_0, Q_0) donde $P_0 = (2,4)$ y $Q_0 = (1,4)$. Determinar, además, el Código Cadena asociado a C_{D2}.

Resolución

Primer Paso

Tomamos los vecinos 8N de P_0, ordenados en sentido horario y comenzando con Q_0.
A estos vecinos los llamamos $R_{01} = Q_0$, R_{02},…., R_{08}, según se muestra en la siguiente figura:



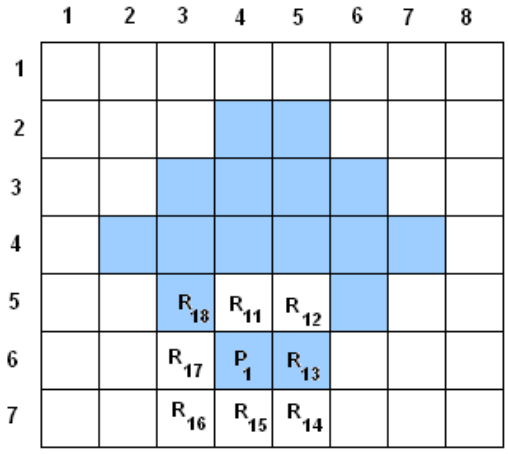
Observamos que R_{02} es el primero de estos vecinos 8N que está en C. Por lo tanto tomamos

$$\begin{cases} P_1 = R_{02} = (4, 6) \\ Q_1 = R_{01} = (4, 5) \end{cases}$$

En este caso, el Código Cadena de P_1 es "7", mientras que el de Q_1 es "0" (notemos que "0" es el número que le sigue al "7")
Como $P_1 \neq P_0$ y $Q_1 = Q_0$, el proceso continúa.

Segundo Paso

Ahora consideramos, $P_1 = (4,6)$ y $Q_1 = (4,5)$. Tomamos los vecinos 8N de P_1, ordenados en sentido horario y comenzando con Q_1.
A estos vecinos los llamamos $R_{11} = Q_1$, R_{12},…., R_{18}, según se ve en la siguiente figura:



En este caso, R_{13} es el primero de estos vecinos 8N que está en C. Luego tomamos

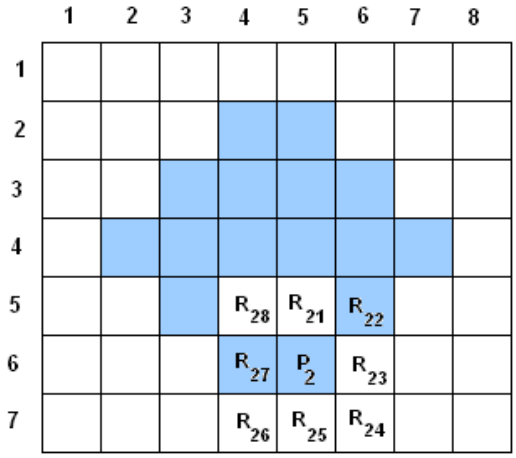
$$\begin{cases} P_2 = R_{13} = (5, 6) \\ Q_2 = R_{12} = (5, 5) \end{cases}$$

En este caso, el Código Cadena de P_2 es "0", mientras que el de Q_2 es "1" (pues "1" es el número que le sigue al "0")

Como $P_2 \neq P_0$ y $Q_2 \neq Q_0$, el proceso continúa.

Tercer Paso

Tomamos $P_2 = (5,6)$ y $Q_2 = (5,5)$. Consideramos los vecinos 8N de P_2, ordenados en sentido horario y comenzando con Q_2.
A estos vecinos los llamamos $R_{21} = Q_1$, R_{22},…., R_{28}, según se observa en la figura:



Ahora, R_{22} es el primero de estos vecinos 8N que está en C. Por lo tanto

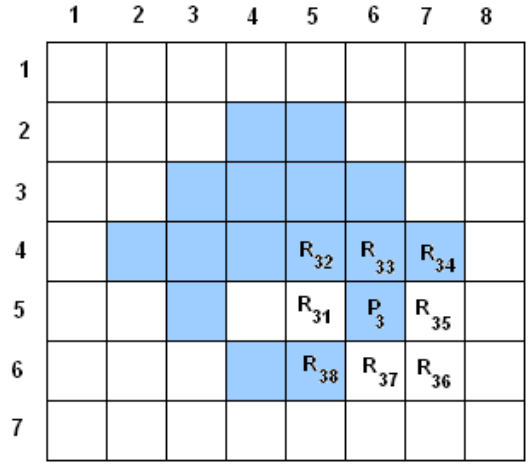
$$\begin{cases} P_3 = R_{22} = (6, 5) \\ Q_3 = R_{21} = (5, 5) \end{cases}$$

El Código Cadena de P_3 es entonces "1"

Como $P_3 \neq P_0$ y $Q_3 \neq Q_0$, el proceso continúa.

Cuarto paso

Tomamos $P_3 = (6,5)$ y $Q_3 = (5,5)$. Consideramos los vecinos 8N de P_3, ordenados en sentido horario y comenzando con Q_3.
A estos vecinos los llamamos $R_{31} = Q_3$, R_{32},…., R_{38}, según se ve en la figura:



Ahora, R_{32} es el primero de estos vecinos 8N que está en C. En consecuencia

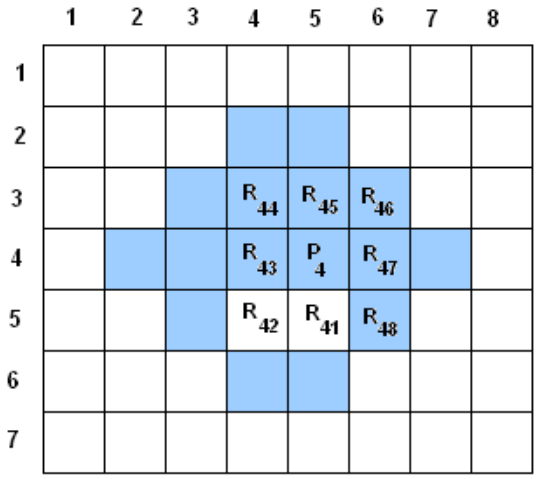
$$\begin{cases} P_4 = R_{32} = (5, 4) \\ Q_4 = R_{31} = (5, 5) \end{cases}$$

En este caso, el Código Cadena de P_4 es "3".

Como $P_4 \neq P_0$ y $Q_4 \neq Q_0$, el proceso continúa.

Quinto Paso

Tomamos $P_4 = (5,4)$ y $Q_4 = (5,5)$. Consideramos los vecinos 8N de P_3, ordenados en sentido horario y comenzando con Q_4.
A estos vecinos los llamamos $R_{41} = Q_4$, R_{42},…., R_{48}, según se muestra en la siguiente figura:



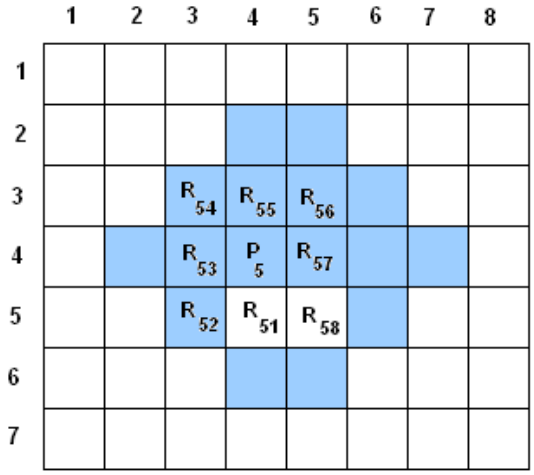
Ahora, R_{43} es el primero de estos vecinos 8N que está en C. Por lo tanto

$$\begin{cases} P_5 = R_{43} = (4, 4) \\ Q_5 = R_{42} = (4, 5) \end{cases}$$

El Código Cadena de P_5 es "4"
Como $P_5 \neq P_0$ y $Q_5 = Q_0$, el proceso continúa.

Sexto Paso

Tomamos $P_5 = (4,4)$ y $Q_5 = (4,5)$. Consideramos los vecinos 8N de P_3, ordenados en sentido horario y comenzando con Q_4.
A estos vecinos los llamamos $R_{51} = Q_5$, R_{52},…., R_{58}, según se muestra en la siguiente figura:



Ahora, R_{52} es el primero de estos vecinos 8N que está en C. Luego

$$\begin{cases} P_6 = R_{52} = (3, 5) \\ Q_6 = R_{51} = (4, 5) \end{cases}$$

Como $P_{6=}\ P_0$ y $Q_{6\ =}\ Q_0$, el proceso se detiene.

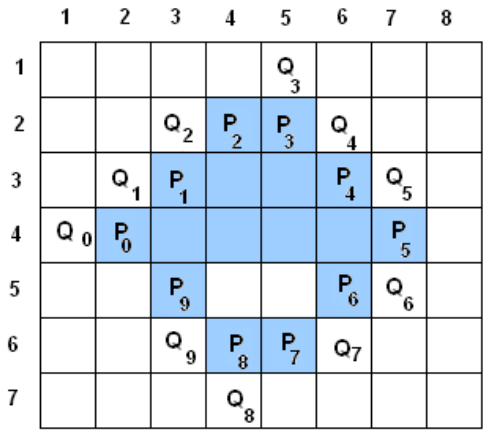
De esta manera, el borde de C con respecto a D1 es

$$C_{D1} = \{P_0, P_1, P_2, P_3, P_4, P_5\}$$

El mismo quedará almacenado en la memoria del ordenador a través del par inicial (P_0, Q_0) donde $P_0 = (3,5)$, $Q_0 = (4,5)$ y el Código Cadena (7, 0, 1, 3, 4).

Observación:
Es muy fácil obtener el Código Cadena de $\{Q_1, Q_2, Q_3, Q_4, Q_5\}$. Este es {0, 1, 2, 4, 5} ya que el código de Q_i es el consecutivo de P_i.

b) El procedimiento es similar al del inciso a). La secuencia de puntos resultantes se muestra a continuación:



En la memoria del ordenador, C_{D2} quedará almacenado a través del par inicial $P_0 = (2,4)$, $Q_0 = (1,4)$ y el Código Cadena (1, 1, 0, 7, 7, 5, 5, 4, 3).

En el siguiente ejemplo mostraremos cómo reconstruir el conjunto S a partir de su borde, tal como lo llevaría a cabo el ordenador.

Ejemplo 4.6

En el ejemplo anterior, reconstruya el conjunto S a partir de su borde.

Resolución

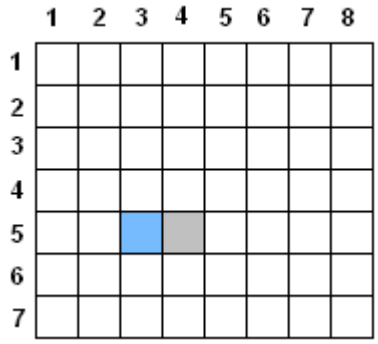
Sabemos que

$$\text{Borde}(S) = C_{D1} \cup C_{D2}$$

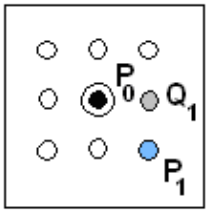
Recordemos que el ordenador almacena, para cada C_D, el par inicial (P_0, Q_0) y su correspondiente Código Cadena. Lo que haremos primeramente es dibujar C_{D1} y C_{D2}, junto con los puntos de D1 y D2 que son adyacentes a C. Es decir, representaremos también los puntos $Q_0, Q_1,\ldots$que se obtuvieron mediante el Algoritmo BF8.

Representación de C_{D1}

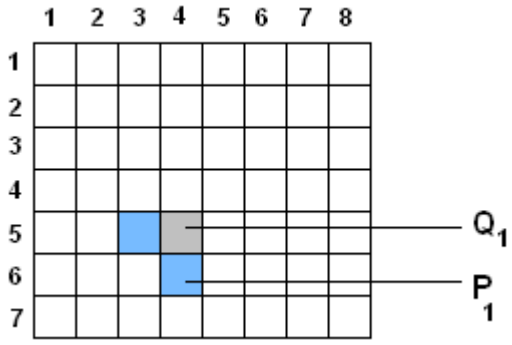
En este caso $P_0 = (3,5)$, $Q_0 = (4,5)$ y su correspondiente Código Cadena es (7, 0, 1, 3, 4). Como el Código Cadena consta de cinco elementos, el borde C_{D1} es el conjunto formado por los puntos $P_0, P_1, P_2,\ldots,P_5,$. De este modo, P_1 tiene asociado el código "7", P_2 el código "0" y así sucesivamente.
Primeramente, ubicamos en la imagen digital los pares $P_0 = (3,5)$ y $Q_0 = (4,5)$. Los píxeles de C_{D1} los representaremos de color celeste, mientras que los de D1, de color gris, según indica la figura:



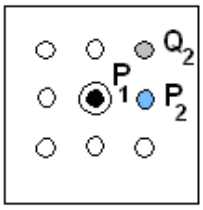
Como P_1 tiene código "7", Q_1 tendrá código "0" (pues es el consecutivo de "7"). Luego, éstos se encuentran en la siguiente posición relativa a P_0:



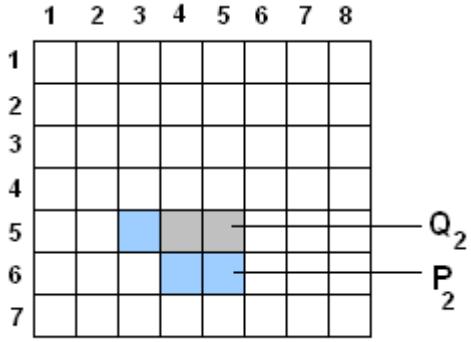
Luego, P_1 y Q_1 quedan representados dentro de la imagen digital de la siguiente manera (en este caso Q_1 coincide con Q_0):



Ahora, como P_2 tiene código “0”, Q_2 tiene código “1”. De este modo, se encuentran en la siguiente posición relativa a P_1:



Por lo tanto, su representación dentro de la imagen digital es:



Aplicando el mismo procedimiento para los puntos restantes, se llega a que C_{D1} es el siguiente conjunto de píxeles celestes:

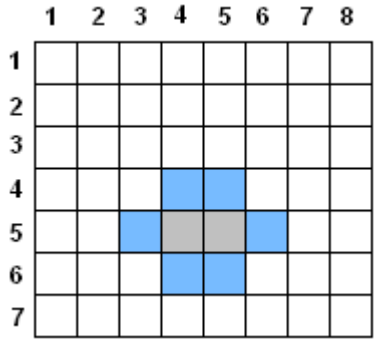


Figura 4.4: Reconstrucción de C_{D1} a partir de (P_0, Q_0) y su Código Cadena.

Representación de C_{D2}

Aquí tenemos $P_0 = (2,4)$, $Q_0 = (1,4)$ y el Código Cadena es (1, 1, 0, 7, 7, 5, 5, 4, 3). Siguiendo la misma técnica que el caso anterior, resulta que C_{D2} es el siguiente conjunto de píxeles celestes:

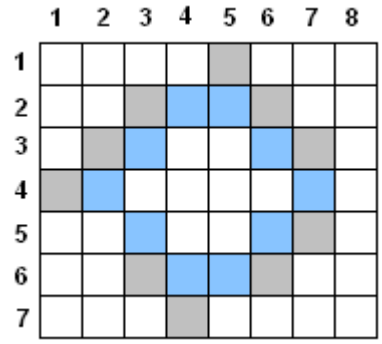
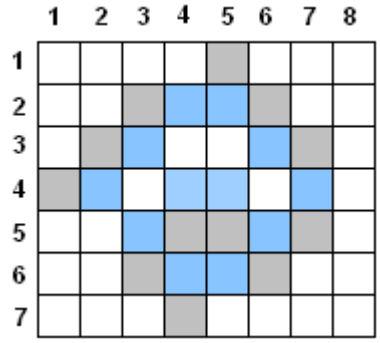
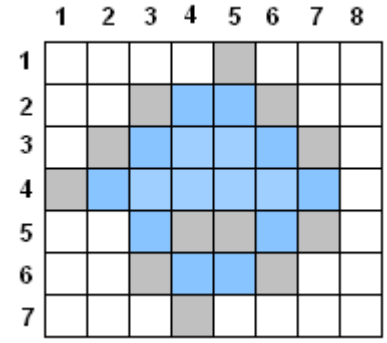


Figura 4.5: Reconstrucción de C_{D2} a partir de (P_0, Q_0) y su Código Cadena.

Uniendo los gráficos de las Figuras 4.4 y 4.5 se obtiene finalmente el borde de S:



Por último, para obtener el conjunto S, simplemente se “colorea” de celeste interior de C:



Observemos que si no hubiésemos pintado de gris los píxeles de D1 y D2 que son adyacentes a C_{D1} y C_{D2}, no hubiéramos podido distinguir el interior de S del fondo de la imagen.

En el capítulo siguiente haremos una breve descripción de la implementación del Algoritmo BF4/BF8 en una aplicación específica.

Capítulo 5

Descripción de la aplicación

Capítulo 5: Descripción de la aplicación

En este capítulo se ha desarrollado una aplicación que implementa el Algoritmo BF8. Mediante el mismo, lograda una primera separación por segmentación, se puede recorrer el borde de cada objeto. De este modo es posible seleccionar y separar, dentro de una imagen digital, un objeto con características definidas con anterioridad. En este caso particular, se pretende encontrar un objeto con tonalidades anaranjadas, que posea una geometría circular y un área que es dato.
La aplicación se desarrolló dentro del paradigma de "Programación Orientada a Objetos", según las actuales tendencias tecnológicas. Más específicamente, se utilizó el lenguaje **Python**, que resultó ser el más conveniente por la condición del problema planteado: no es un único objeto el que se desea analizar, sino que la cantidad de objetos a analizar es desconocida.
En "Programación Orientada a Objetos" normalmente existe un programa principal que contiene instrucciones simples, funciones, creación de objetos, uso de sus métodos, etc. Pero la parte más compleja de este paradigma radica en la definición y programación de las clases.

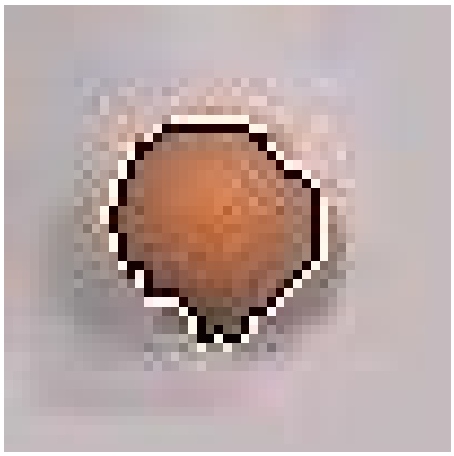


Figura 5.1: Imagen aumentada del objeto que analiza esta aplicación: una pelota de color anaranjado.

Funcionamiento del programa principal

El programa principal se encuentra en el archivo llamado "Principal.py", y ejecuta las siguientes acciones:

- Recibe el nombre del archivo con la foto donde se encuentra la pelotita.
- Presenta en pantalla la ubicación del centro de la pelotita y otros datos relevantes.
- Genera un archivo imagen "Bordemarcado.jpg" que es la misma imagen con el borde de la pelotita pintada en blanco y negro.
- Genera un archivo de texto "Puntosdelborde.txt" que contiene los puntos del borde de la pelotita.

Para llevar a cabo lo anteriormente mencionado se procede así:

■ Se crea un objeto *matriz*, a partir del nombre del archivo con la foto.
Matriz.crear toma la imagen y devuelve el ancho y alto de la misma.

Matriz.valor trae una matriz de ancho x alto, con 1 y 0 según el resultado de la segmentación.

■ Se recorre la matriz hasta encontrar un 1. Cuando lo encuentra, se crea un objeto *conjunto*.

Conjunto.contar trae la cantidad de pixeles segmentados.

Conjunto.marcarborde recorre el borde del conjunto:

- ■ Graba un archivo imagen con la frontera del objeto marcada en blanco y negro.
- ■ Graba un archivo de texto con los puntos del borde del objeto.
- ■ Devuelve por pantalla la ubicación del centro, el área calculada, valores mínimos y máximos, un coeficiente de circularidad y demás información de interés.

Una vez creado el programa con este tipo de estructuras, es posible decidir si el objeto encontrado es el buscado o no.

El diagrama de clases

En esta sección mostramos el diagrama de clases que se utilizarán en este programa.

El código fuente

Principal.py

```
import os
import pygame
import Image
from pygame import *
from pygame.locals import *
from clasematrix import Matrix
from claseconjunto import Conjunto

archivo="web-shot1.jpg"
matriz=Matrix()
matriz.crear(archivo)
print "Ancho: "+str(matriz.ancho)+" x Alto: "+str(matriz.alto)
laMatriz=matriz.valor
altoMatriz=matriz.alto
anchoMatriz=matriz.ancho
laOtraMatriz=laMatriz
=0
x=0
i=0
a=0
for y in range(0, altoMatriz):
   if i==1:
      break
   else:
      for x in range(0, anchoMatriz):
         if i==1:
            break
         else:
            if laOtraMatriz[y][x]==1:
               conjunto=Conjunto()
               conjunto.contar(laMatriz,altoMatriz,anchoMatriz)
               print "Area en píxeles contados: "+str(conjunto.numero)
               numero=conjunto.numero
               a=x
               b=y
               conjunto.marcarBorde(archivo,laOtraMatriz,altoMatriz,anchoMatriz,a,b)
               centro=conjunto.centro
               radio=conjunto.radio
               area=conjunto.area
               laOtraMatriz=conjunto.valor
               if area==0:
                  i=1
                  print "Objeto no encontrado"
               else:
                  if area>numero:
                     circularidad=float((area-(area-numero))/area)
                  else:
                     circularidad=float((area-(numero-area))/area)
                  a=x
                  b=y
                  i=1
                  break
            elif laOtraMatriz[y][x]==0:
               numero=0
               radio=0
               area=0
               circularidad=0
```

```
                centro=0
print "CARACTERISTICAS DEL OBJETO:"
print "Coeficiente de circularidad: "+str(circularidad)
print "Coordenadas del Centro: "+str(centro)
print "Radio: "+str(radio)
print "Area calculada: "+str(area)
print conjunto.contenido
```

Clasematrix.py:

```
import os
import pygame
import Image
import math

class Matrix(object):
    """La clase Matrix carga una imagen y devuelve una matriz binaria segmentada"""
    def __init__(self):
        self.__valor=0
        self.__alto=0
        self.__ancho=0
    def getValor(self):
        return self.__valor
    def getAlto(self):
        return self.__alto
    def getAncho(self):
        return self.__ancho
    def setValor(self,valor):
        self.__valor=valor
    def setAlto(self,alto):
        self.__alto=alto
    def setAncho(self,ancho):
        self.__ancho=ancho
    valor=property(getValor,setValor)
    alto=property(getAlto,setAlto)
    ancho=property(getAncho,setAncho)

    def crear(self,archivo):
        """Toma el archivo de imagen y retorna los valores de ancho y alto"""
        """Con ancho y alto, segmenta y pasa 1 y 0 a una matriz segmentada"""
        imagen=Image.open(archivo)
        alto = imagen.size[0] #toma el Alto
        ancho = imagen.size[1] #toma el ancho
        self.__ancho=ancho
        self.__alto=alto
        print str(alto)+" x "+str(ancho)
        x=0
        y=0
        # Aquí se introducen los valores RGB que segmentan por el color anaranjado.
        Rmin=92
        Rmax=224
        Gmin=68
        Gmax=128
        Bmin=50
        Bmax=101
        valor=[]
        for y in range(alto):
            a=[0]*ancho
            valor.append(a)
        for y in range(0, alto):
            for x in range(0, ancho):
                R= imagen.getpixel((x,y))[0]
```

```
            G= imagen.getpixel((x,y))[1]
            B= imagen.getpixel((x,y))[2]
            if ((R<Rmax and R>=Rmin) and (G<Gmax and G>=Gmin) and (B<=Bmax and B>Bmin)):
               valor[y][x]=1
            else:
               valor[y][x]=0
      print "Lectura de imagen terminada"
      self.__valor=valor
      return self.__valor
      return self.__ancho
      return self.__alto
```

Claseconjunto.py:

```
import os
import pygame
import Image
import math
from clasematrix import Matrix

class Conjunto(object):
   """Divide a la matriz en conjuntos segmentados conexos"""
   def __init__(self):
      self.__area=1
      self.__alto=0
      self.__centro=0
      self.__ancho=0
      self.__circularidad=0
      self.__numero=0
      self.__valor=0
      self.__contenido=0
      self.__centro=0
      self.__radio=0
   def getArea(self):
      return self.__area
   def getAlto(self):
      return self.__alto
   def getAncho(self):
      return self.__ancho
   def getCircularidad(self):
      return self.__circularidad
   def getNumero(self):
      return self.__numero
   def getContenido(self):
      return self.__contenido
   def getValor(self):
      return self.__valor
   def getCentro(self):
      return self.__centro
   def getRadio(self):
      return self.__radio
   def setArea(self,area):
      self.__valor=area
   def setAlto(self,alto):
      self.__alto=alto
   def setAncho(self,ancho):
      self.__ancho=ancho
   def setCircularidad(self,circularidad):
      self.__circularidad=circularidad
   def setNumero(self,numero):
      self.__numero=numero
   def setContenido(self,contenido):
```

```
        self.__contenido=contenido
    def setValor(self,valor):
        self.__valor=valor
    def setCentro(self,centro):
        self.__centro=centro
    def setRadio(self,radio):
        self.__radio=radio
    area=property(getArea,setArea)
    alto=property(getAlto,setAlto)
    ancho=property(getAncho,setAncho)
    numero=property(getNumero,setNumero)
    circularidad=property(getCircularidad,setCircularidad)
    contenido=property(getContenido,setContenido)
    valor=property(getValor,setValor)
    centro=property(getCentro,setCentro)
    radio=property(getRadio,setRadio)

    def contar(self,laMatriz,altoMatriz,anchoMatriz):
        """Cuenta la cantidad de píxeles que cumplen con la segmentación"""
        x=0
        y=0
        numero=0
        a=0
        for y in range(0, altoMatriz):
            for x in range(0, anchoMatriz):
                if laMatriz[x][y]==0:
                    a=a+1
                elif laMatriz[x][y]==1:
                    numero=numero+1
        self.__numero=numero
        return self.__numero

    def marcarBorde(self,archivo,laOtraMatriz,altoMatriz,anchoMatriz,a,b):
        """Se pone una marca a los elementos que pertenecen al borde"""
        imagen=Image.open(archivo)
        MaxX=0
        MinX=anchoMatriz
        MaxY=0
        MinY=anchoMatriz
        # BF4/8 TIENE UN "DO-WHILE", ES DECIR, EJECUTA UNA VEZ Y LUEGO PREGUNTA
CONDICION.
        # POR LO TANTO, PRIMERO DADOS Po Y Qo OBTENGO P1 Y Q1 (DO).
        # MAS ADELANTE ENTRAREMOS AL "WHILE CON P1 Y Q1".
        #V8No son los vecinos 8N del punto Pn=a en cualquier orden.
        V8No=[[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0]]
        #V8N son los vecinos 8N del punto ordenados a partir del punto Qn=b
        V8N=[[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0]]
        p=[0]*2
        q=[0]*2
        a1=[0]*2
        b1=[0]*2
        i=1
        p=[a,b]
        q=[a,b-1]
        P=[p]
        Q=[q]
        laOtraMatriz[P[0][0]][P[0][1]]=2
        laOtraMatriz[Q[0][0]][Q[0][1]]=-1
        Po=[p]
        Qo=[q]
        A=[a1]
        B=[b1]
```

```
Base=[[-1,0],[-1,-1],[0,-1],[1,-1],[1,0],[1,1],[0,1],[-1,1]]
k=0
pos=0
laOtraMatriz[P[0][0]][P[0][1]]=2
laOtraMatriz[Q[0][0]][Q[0][1]]=-1
for j in range(0, 8):
    for i in range (0,2):
        V8No[j][i]=Base[j][i]+P[0][i]
B[0][0]=Q[0][0]-P[0][0]
B[0][1]=Q[0][1]-P[0][1]
for j in range(0, 8):
    if((B[0][0]==Base[j][0]) and (B[0][1]==Base[j][1])):
        break
pos=j
for j in range(0, 8):
    if(pos+j<8):
        for i in range (0,2):
            V8N[j][i]=V8No[pos+j][i]
    else:
        for i in range (0,2):
            V8N[j][i]=V8No[pos+j-8][i]
k=0
for j in range(0, 8):
    if k==1:
        break
    for i in range (0,2):
        P[0][0]=V8N[j][0]
        P[0][1]=V8N[j][1]
        if(laOtraMatriz[P[0][0]][P[0][1]]==1):
            Q[0][0]=V8N[j-1][0]
            Q[0][1]=V8N[j-1][1]
            k=1
            break
# Grabo las coordenadas de los puntos del borde en un archivo de texto.
archivo=open('puntosdelborde.txt','w')
linea="["+str(P[0][0])+","+str(P[0][1])+"]"+"\n"
archivo.writelines(linea)
# PINTO DE NEGRO EL PRIMER PUNTO DEL BORDE INTERIOR Y DE BLANCO EL
  EXTERIOR
imagen.putpixel((P[0][0],P[0][1]),(0,0,0))
imagen.putpixel((Q[0][0],Q[0][1]),(255,255,255))
while((laOtraMatriz[P[0][0]][P[0][1]]==1)and(laOtraMatriz[Q[0][0]][Q[0][1]]==0) or
(laOtraMatriz[Q[0][0]][Q[0][1]]==-1)):
    k=0
    pos=0
    if P[0][0]>MaxX:
        MaxX=P[0][0]
    else:
        n=1
    if P[0][0]<MinX:
        MinX=P[0][0]
    else:
        n=1
    if P[0][1]>MaxY:
        MaxY=P[0][1]
    else:
        n=1
    if P[0][1]<MinY:
        MinY=P[0][1]
    else:
        n=1
    # Grabo con 2 a P y -1 a Q
```

```
    laOtraMatriz[P[0][0]][P[0][1]]=2
    laOtraMatriz[Q[0][0]][Q[0][1]]=-1
    for j in range(0, 8):
        for i in range (0,2):
            V8No[j][i]=Base[j][i]+P[0][i]
    B[0][0]=Q[0][0]-P[0][0]
    B[0][1]=Q[0][1]-P[0][1]
    for j in range(0, 8):
        if((B[0][0]==Base[j][0]) and (B[0][1]==Base[j][1])):
            break
    pos=j
    for j in range(0, 8):
        if(pos+j<8):
            for i in range (0,2):
                V8N[j][i]=V8No[pos+j][i]
        else:
            for i in range (0,2):
                V8N[j][i]=V8No[pos+j-8][i]
    k=0
    for j in range(0, 8):
        if k==1:
            break
        for i in range (0,2):
            P[0][0]=V8N[j][0]
            P[0][1]=V8N[j][1]
            if(laOtraMatriz[P[0][0]][P[0][1]]==1):
                Q[0][0]=V8N[j-1][0]
                Q[0][1]=V8N[j-1][1]
                # PINTO DE NEGRO EL PRIMER PUNTO DEL BORDE INTERIOR Y DE BLANCO
                  EL EXTERIOR
                imagen.putpixel((P[0][0],P[0][1]),(0,0,0))
                imagen.putpixel((Q[0][0],Q[0][1]),(255,255,255))
                k=1
                break
    # Grabo las coordenadas de los puntos del borde en un archivo de texto.
    linea="["+str(P[0][0])+","+str(P[0][1])+"]"+"\n"
    archivo.writelines(linea)
imagen.save("bordemarcado.jpg")
centro=[(MaxX+MinX)/2,(MaxY+MinY)/2]
radio=float((((MaxX-((MaxX+MinX)/2))+((MaxX+MinX)/2)-MinX+(MaxY-
            ((MaxY+MinY)/2))+((MaxY+MinY)/2)-MinY))/4)
area=3.141*(radio**2)
self.__radio=radio
self.__area=area
self.__centro=centro
self.__valor=laOtraMatriz
self.__contenido="Máx en X: "+str(MaxX)+" - Min en X: "+str(MinX)+" - Máx en Y:
                  "+str(MaxY)+" - Min en Y: "+str(MinY)
return self.__contenido
return self.__valor
return self.__centro
return self.__area
return self.__radio
```

Capítulo 6

Conclusiones

Capítulo 6: Conclusiones

En el marco actual de constante desarrollo tecnológico, se presentan siempre nuevas oportunidades. Entre ellas se encuentran los sistemas que implementan ViPR (Visión Artificial por Reconocimiento de Patrones). Sin embargo, no se han popularizado y no participan aún de nuestra vida cotidiana. La topología Digital como soporte teórico constituye un recurso fundamental para esta nueva tecnología en desarrollo.

El presente trabajo está basado en un paper que tiene más de 30 años y una vigencia absoluta, siendo referenciado en los trabajos de topología digital más recientes. A pesar de su antigüedad, la implementación del algoritmo BF4/BF8, utilizando lenguajes de tecnologías abiertas y programación orientada a objetos, dan un paso muy importante para el desarrollo de aplicaciones ViPR. Dicho algoritmo es totalmente novedoso, ya que permite obtener de una manera eficiente los puntos del borde de un objeto inmerso en una imagen digital. Luego faltará aplicar geometría afín y computacional para tomar el objeto, rotarlo y trasladarlo al origen de coordenadas, conocer geométricamente qué características se pueden obtener desde ese conjunto de puntos y luego compararlas con bases de datos que contengan características similares a la de los objetos a analizar.

Es importante destacar, además, que el análisis de una imagen digital no necesariamente se limita a una foto, sino que puede provenir de sensores infra-rojo, imágenes de radar, rayos X, resonancia magnética y demás.
El concepto topológico más recurrentemente analizado desde diversos puntos de vista en este trabajo es el de “conjunto conexo”, y tiene su razón lógica: un objeto real es un conjunto topológicamente conexo. Si se logra dividirlo, deja de ser un objeto, y se transforma en una separación del mismo. Para separar a un objeto conexo del fondo de una imagen, es necesario recorrer su borde, y para ello se ha usado el algoritmo BF4 / BF8.

Con el tema desarrollado en este trabajo nos encontramos con oportunidades muy favorables para avanzar en esta línea, a saber:

- En las aplicaciones que utilizan técnicas de aprendizaje automático (basados en probabilidad y estadísticas) pues el alto poder de cómputo y cámaras de alta definición están al alcance de cualquier interesado.
- En los diversos lenguajes de programación relacionados con el tratamiento de imágenes.
- En el Área de Telecomunicaciones, ya que facilitan notoriamente la obtención, colaboración e intercambio del conocimiento entre diversos ámbitos académicos.

Los conceptos expuestos en esta obra son meramente introductorios. El desarrollo de este enfoque es muy prometedor y se presenta paradigmático. Se desea continuar en esta línea de investigación, que promete una gran variedad de aplicaciones en múltiples campos. En este sentido, se podrán hacer más eficientes y seguros los procesos y tareas cotidianas que apuntan a una mejor calidad de vida.
Con este trabajo se espera, además, que entusiasme y atraiga a matemáticos al procesamiento automático de imágenes, a la robótica y a la inteligencia artificial, donde hay numerosas áreas de trabajo por desarrollar y un futuro promisorio.

Referencias

[1] Ulrich Eckhardt – Latecki, Topologies for the Digital Spaces Z^2 and Z^3, Universidad de Hamburgo, 2001.

[2] Ulrich Eckhardt – Latecki, Digital Topology, Universidad de Hamburgo, 1995.

[3] A. Rosenfeld, Digital Topology, Universidad de Columbia, 1979 The American Mathematical Monthly, Vol. 86, No. 8 pp. 621-630, 1979.

[4] D. Marcus - F. Wyse et al., Solution to Problem 5712, monthly 77, (1970) 1119.

[5] E. Khalimsky et al, Topología y sus aplicaciones 36, pp. 1-17, 1990.

[6] T. Y. Kong et al, A topological aproach to digital topology, Amer. Math. Monthly 98, pp. 901-917, 1991.

[7] J. Munkres, "Topología" 2° edición, editorial Pearson Educación Madrid 2002.

[8] M. Cardenas, El teorema de la curva de Jordan para el plano digital, Universidad de Los Andes, 2005.

[9] A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press, New York, 1976.

[10] A. Rosenfeld, ed., Digital Picture Analysis, Springer, Berlin, 1976.