Aproximación de Formas de Objetos Digitales por Polígonos

Jorge Kamlofsky[1], María Lorena Bergamini[1]

[1] CAETI - Universidad Abierta Interamericana
Av. Montes de Oca 725 – Buenos Aires – Argentina
{Jorge.Kamlofsky, Maria.Bergamini}@uai.edu.ar

Resumen. En este trabajo se presenta un avance en las investigaciones que pretenden obtener algoritmos para realizar el seguimiento de un objeto en movimiento. Se propone un algoritmo de simplificación de curvas de borde mediante transformación de dichas curvas a polígonos: se logran aproximaciones adecuadas de las curvas y además se permite ignorar efectos de ruido causados por la digitalización. El grado de aproximación se controla con un parámetro ε, que permite manejar el balance entre el error de la aproximación, y la complejidad del polígono resultante.

I. Introducción.

La visión artificial consiste en identificar objetos dentro de las imágenes digitales reconociendo en ellos sus patrones característicos. La topología digital ha demostrado ser un marco teórico adecuado para el análisis de objetos dentro de imágenes digitales, desde su concepción a inicios de la década de 1970 cuando se presentaron dos topologías sobre el plano [1,2] que son las únicas topologías posibles en dicho espacio [3]. Tratar a ciertos píxeles que conforman un objeto dentro de una imagen como un conjunto, y aplicarles propiedades topológicas es un enfoque simple que ha demostrado ser altamente efectivo cuando se pretende interpretar automáticamente el contenido de una imagen.

Es sabido que actualmente existe una gran cantidad de aplicaciones hechas por empresas líderes mundiales que usan visión artificial basada en topología digital para la eficiente identificación de objetos. Es posible identificar y seguir en tiempo real objetos en movimiento. Dos estrategias son generalmente usadas para tal fin: análisis de los bordes del objeto y análisis de sus esqueletos. En ambos casos, sólo se analiza una pequeña cantidad de puntos, de modo que ese conjunto de puntos obtenidos mantiene ciertas propiedades topológicas respecto del objeto original.

Una primera propiedad de interés es la conexión: un objeto dentro de una imagen es topológicamente conexo. Los puntos del objeto que son adyacentes al complemento constituyen el borde del objeto. Dicho borde puede verse como una curva digital simple cerrada. Este subconjunto es de gran interés en el análisis de objetos dentro de imágenes, ya que muchas características del objeto (conexión, convexidad, dimensiones, agujeros, etc.) pueden estudiarse por medio de su borde. En [4] se le dio el marco topológico adecuado a las imágenes digitales, donde además se presentó un algoritmo para seguimiento de bordes llamado BF4/8 y otro para afinamiento de formas a sus esqueletos.

Dependiendo de las características del objeto, puede ser útil para su reconocimiento el análisis de sus bordes o bien simplificar la imagen, identificando las partes convexas. En el primer caso, el análisis de bordes es inmediato, se trabaja con la curva borde resultante de aplicar el algoritmo BF4/8. En el segundo caso, se divide el objeto en partes convexas, identificándolas con vértices de un grafo. Las aristas del mismo indican las adyacencias entre las partes convexas [5]. Tratar con polígonos convexos nos asegura robustez algorítmica y baja complejidad computacional [6].

Siendo la digitalización una transformación de un objeto del dominio real infinito al codominio discreto (finito), esa pérdida de información se refleja en la aparición de ruido. En las imágenes digitales, ese ruido tiene notoria visibilidad en las fronteras de los objetos donde aparecen formas de dientes de serrucho, dificultando su estudio, ya que el análisis de bordes por geometría diferencial requiere el análisis de longitud, rectitud, curvatura y convexidad [7, 8], características que se ven afectadas por las irregularidades del borde, producto del proceso de digitalización. El ruido en los bordes ha sido tratado de distintas formas entre otros en [9, 10].

Para facilitar la identificación de objetos dentro de las imágenes digitales, se suele concentrar esfuerzos en identificar sus patrones [11], independientemente de la posición, orientación y tamaño del objeto analizado. Aquí se nos presenta el problema de que las transformaciones afines no son invariantes en una digitalización debido al ruido y al nivel de resolución de la imagen [9].

El presente trabajo propone un método para aproximar el borde de un objeto por una sucesión de segmentos rectos. La aproximación se hace teniendo en cuenta una tolerancia del error cometido al remplazar puntos del borde por segmentos rectos. Como consecuencia de la aproximación, el borde se suaviza, eliminándose la incidencia del ruido en la forma de la imagen digital.

Otra ventaja de este enfoque es que el borde se reduce ahora a un conjunto de puntos que son los vértices del polígono aproximante, lo que brinda una enorme ganancia en capacidad de procesamiento, cuestión elemental para lograr seguimiento de objetos en tiempo real.

II. DEFINICIONES.

Una *imagen digital* es una función $f: \pi \subset Z^2 \rightarrow Z^n$, con $n \in N$, que asigna a cada pixel $(x,y) \in \pi$ un valor $f(x,y) \in Z^n$ que representa el color de dicho pixel [12]. El valor n queda determinado según el modo de confección de la imagen digital.

Se denomina *segmentación de una imagen digital* a la descomposición del dominio de la misma en subconjuntos o regiones disjuntas dos a dos [4,12], de acuerdo a los valores de f. La imagen segmentada se compone de una colección de subconjuntos no vacíos $S_1, S_2, \ldots, S_m$ de forma tal que $\pi = \cup_{1 \le i \le m} S_i$ con $S_i \cap S_j = \varnothing$ si $i \neq j$. El proceso de segmentación de una imagen nos permite separar los objetos de interés del fondo de la imagen.

Dado un punto $P = (x,y) \in Z^2$, los *vecinos 4N* de P son $(x\pm1, y)$, $(x, y\pm1)$. Los *vecinos 8N* de P son $(x\pm1, y)$, $(x, y\pm1)$, $(x+1, y\pm1)$, $(x-1, y\pm1)$. Si $Q \in Z^2$ se dice que Q es *adyacente 4N* (*8N*) a P si P es vecino 4N (8N) de Q.

El conjunto de puntos que son vecinos 4N y 8N de un punto $P \in Z^2$ recibe el nombre de *vecindad 4N* y *vecindad 8N* respectivamente.

Sean P y Q puntos del dominio de la imagen digital. Un *camino 4N* (*8N*) de P a Q es una secuencia de puntos $P = P_0, P_1, \ldots, P_n = Q$, tal que P_i es vecino 4N (8N) de P_{i+1}, para i = 0, 1, …, n-1. Un subconjunto S se dice 4- (8-) *conexo* si para cada par de puntos P y Q de S, existe un camino 4N (8N) de P a Q consistente en puntos de S.

Sea $S \subset \pi$, $S \neq \varnothing$. El *fondo de S* es la única componente conexa de S^c que contiene al borde de π. Las demás componentes de S^c, si existen, reciben el nombre de *agujeros de S*.

Sea C una componente conexa de S y D una componente conexa de S^c. El *borde de C respecto a D* es el conjunto definido por $C_D = \{P \in C / P$ tiene al menos un vecino 4N en $D\}$.

Un subconjunto S de una imagen digital es un *arco 4N* (*8N*) si es conexo y todos, salvo dos de sus puntos (sus extremos) tienen exactamente dos vecinos 4N (8N) en S. Más precisamente, un arco S es un camino $P_0, P_1, \ldots, P_n$ formado por puntos distintos, tal que para $0 < i < n$, los únicos vecinos 4N (8N) de P_i son el punto anterior P_{i-1} y del punto siguiente P_{i+1}.

Un subconjunto S de una imagen digital es una *curva 4N* (*8N*) si es conexo y todos sus puntos tienen exactamente dos vecinos 4N (8N) en S. Más precisamente, una curva S es un camino $P_0, P_1, \ldots, P_n$ formado por puntos distintos tal que para $0 < i < n$, los únicos vecinos 4N (8N) de P_i son P_{i-1} y P_{i+1}, mientras P_1 y P_n son los únicos vecinos 4N (8N) de P_0; y P_0 y P_{n-1} son los únicos vecinos 4N (8N) de P_n. Se puede decir entonces que una curva es un arco cerrado.

El borde de un objeto digital S se puede obtener mediante el algoritmo BF4/8 creado por A. Rozenfeld y presentado en [4]. Una implementación de ese algoritmo puede observarse en [12]. El algoritmo construye una curva formada por puntos de S que son vecinos 4N/8N de algún punto de S^c.

Tal curva de borde puede representarse como un conjunto de pares ordenados que dan las coordenadas enteras de los puntos en la curva. Sin embargo, también puede almacenarse, con menos requerimientos de memoria, en la forma de código cadena. El código cadena de una curva consiste en las coordenadas del punto inicial, seguido de una lista de códigos de movimientos para pasar al siguiente punto. El código de movimiento, cuando se usa vecindad 8N es un entero k entre 0 y 7, indicando que el vecino siguiente se obtiene con un desplazamiento en la dirección $k\pi/4$. Cada punto de la curva, entonces, puede codificarse con sólo 3 bits. Usando vecindad 4N, el código de movimiento es un número k entre 0 y 3, indicando un desplazamiento en la dirección $k\pi/2$. Así, cada punto de la curva, puede codificarse con sólo 2 bits.

III. APROXIMACIÓN DE FIGURAS DIGITALES 2D POR POLÍGONOS.

Como se mencionó en la introducción, el borde de un objeto permite estudiar las características de éste, a fin de identificarlo. La curva de borde es un conjunto de puntos, y en general, no es suave, no sólo por su naturaleza discreta, sino también por la presencia de ruido en la imagen.

A fin de suavizar y simplificar la curva de borde, se propone un algoritmo para aproximar una curva digital por un polígono.

El ancho de un conjunto de puntos se define como la menor distancia entre dos rectas paralelas que encierran a los puntos. Un conjunto de puntos alineados tiene ancho cero. Basándonos en estas afirmaciones, se aproximará con un segmento de recta a un conjunto de puntos con ancho menor que cierta tolerancia.

Dada la lista de puntos de borde $P_0, P_1, \ldots, P_n$, el algoritmo propuesto particiona la lista en sublistas de puntos consecutivos $\{P_k, P_{k+1}, \ldots, P_{k+m}\}$. Los puntos en cada sublista se considera que están aproximadamente alineados, esto es, el ancho del conjunto de puntos en cada sublista es menor que la tolerancia especificada.

Esta partición se hace secuencialmente, se van analizando uno a uno los puntos del borde y se examina si puede pertenecer al lado actual, o es un punto en un lado nuevo del polígono, que comenzaría en el punto anterior.

Con tolerancias muy pequeñas, se generan lados de pocos puntos y polígonos de muchos vértices. En el caso límite de tolerancia cero, los vértices del polígono son casi todos los puntos de la curva. Con tolerancias más grandes se obtienen lados que abarcan muchos puntos, y por lo tanto, polígonos de menos lados.

El paso central del algoritmo consiste en analizar si un punto puede ser admitido en el lado que se está construyendo. Este paso se debe realizar para cada punto de la curva. Para mejorar la eficiencia, se realiza un preprocesamiento para disminuir la cantidad de puntos de la curva. Este procedimiento consiste en detectar sucesiones de puntos que ya están alineados (conjunto de ancho cero) y eliminar los interiores, para quedarse con el primero y el último. Para ello se puede aplicar la caracterización del código cadena para digitalización de rectas [8].

Sin embargo, para hacer este paso de bajo costo computacional, sólo se consideran segmentos digitales con una pendiente $k\ \pi/4$ (considerando que el borde es una curva 8N), con k = 0,1, …, 7, que se caracterizan porque su código cadena tiene un solo código (k) que se repite.

El algoritmo que se propone en este trabajo para aproximar y simplificar curvas de borde usando polígonos se detalla en la tabla 1.

Tabla 1. Algoritmo para aproximar curvas digitales

Entrada: una curva borde de un objeto representada mediante su código cadena, $P_0, c_1,\ldots c_n$, y una tolerancia ε
Salida: el conjunto de vértices del polígono que aproxima el objeto.
Inicialización:
PP = PrimerPoligonalizacion($P_0, c_1,\ldots, c_n$)
lados = {}
L = {P_0} *% comienza un lado con el primer punto.*
$V_0 = P_0$ *% primer vértice del polígono*
k = 0
n = cantidad de puntos del borde luego de poligonalización inicial
Para i = 1 : n
Si AdmitirPuntoEnLado(L, PP_i),
L = L + {PP_i}
Si no
lados = lados + {L}
L = {PP_{i-1} , PP_i} *% comienza un nuevo lado*
k = *k* + 1
$V_k = PP_{i-1}$ *% un nuevo vértice del polígono*
Fin si
Fin para

A. Procedimiento PrimerPoligonalizacion($P_0, c_1,\ldots, c_n$)

Este procedimiento recorre el código cadena de la curva, y si encuentra repeticiones consecutivas de código k, correspondientes a segmentos de recta de pendiente $k\ \pi/4$ (en curvas 8N) o pendiente $k\ \pi/2$ (en curvas 4N), calcula los puntos extremos de tal segmento, y los almacena en la lista de vértices del primer polígono.

En el caso más general, los segmentos digitales de cualquier pendiente pueden detectarse identificando regularidades en el código cadena [8]. Así, la reducción de puntos en el borde puede ser mayor, aún sin pérdida de información. Sin embargo, la implementación de búsqueda de tales regularidades complejiza el procedimiento *PrimerPoligonizacion*, resultando finalmente el algoritmo de aproximación con similar costo computacional.

La primer poligonalización es una simplificación de la curva de borde, eliminando puntos que están alineados. Equivale a una aproximación con tolerancia cero. Mediante esta transformación se pretende obtener disminución del costo computacional en la etapa principal del algoritmo, sin perder información relevante acerca de las características del borde.

En la tabla 2 se detalla este procedimiento para curvas 8N.

Tabla 2. Primer poligonalización de curvas digitales.

PrimerPoligonalizacion
Entrada: una curva 8N representada mediante su código cadena $C = P_0, c_1,\ldots c_n$; y n = cantidad de puntos de C.
Salida: un conjunto de vértices PP de un polígono que representa la curva C.
Inicialización:
PP = {P_0}
$V_{ant} = P_0$
j = 1
Para i = 1 : n
Si $c_i = c_{i+1}$
j = j + 1 *% cuenta repeticiones de un código*
Si no
$V_{act} = V_{ant}$ +j*redondea(cos(c_i*π/4), sen(c_i*π/4))
PP = PP + {V_{act}} *% agrega un nuevo vértice*
$V_{ant} = V_{act}$, *j = 1*
Fin si
Fin para

Para curvas 4N, el prepocesamiento es similar. La única modificación es cálculo de V_{act}.

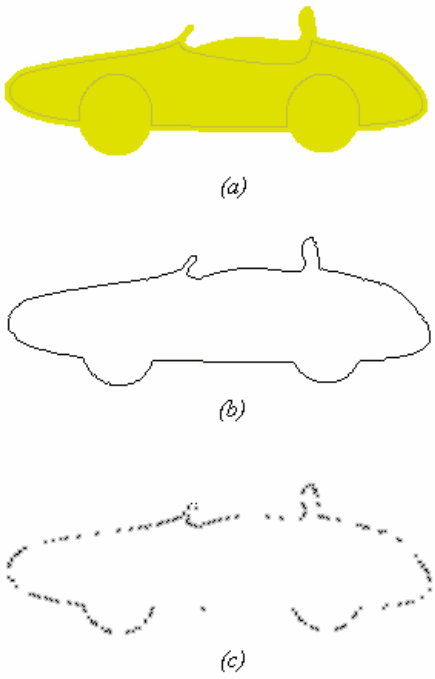

Fig. 1. (a) Forma analizada. (b) Borde completo. (c) Vértices de primer polígono.

Obsérvese la Fig. 1. En (a) se presenta el objeto a analizar, el dibujo de un automóvil con 14971 píxeles. En (b) se muestra el borde obtenido mediante BF8 (usando la implementación del mismo presentada en [12]). La cantidad de puntos del borde es de 648 píxeles. En (c) se muestra el conjunto de vértices PP. La cantidad total de puntos

resultantes es de 212 píxeles, que representa el 32% de los puntos del borde del objeto.

B. Procedimiento AdmitirPuntoEnLado(L,PP$_i$)

AdmitirPuntoEnLado es un procedimiento que recibe la lista de puntos en el lado *L* que se está construyendo y el nuevo candidato PP_i, y mide el ancho del conjunto de puntos formado por los puntos en *L* más PP_i. Si el ancho es menor que la tolerancia, devuelve "Verdadero", en caso contrario devuelve "Falso". El ancho del conjunto se calcula como el ancho de su envoltura convexa. Entonces, cualquier algoritmo para calcular envoltura convexa puede ser aplicado [8].

Sin embargo, como este procedimiento debe obtener sucesivamente la cáscara convexa de conjuntos de puntos que difieren en el agregado de un punto en cada instancia, (mientras el ancho se mantenga menor que la tolerancia), se aplica un algoritmo incremental. Sea S_m el conjunto de los primeros *m* puntos y sea $conv(S_m)$ su envoltura convexa. Si el siguiente punto P_{m+1} está dentro de ella, $conv(S_{m+1})=conv(S_m)$. En otro caso, se eliminan de $conv(S_m)$ los vértices que el punto P_{m+1} puede *ver*, y se agrega P_{m+1} para obtener $conv(S_{m+1})$.

Una vez obtenida la envoltura convexa, se calcula su ancho [8] para compararlo con la tolerancia especificada.

En la Fig. 2 se muestra el resultado de aplicar el algoritmo de aproximación a un arco 8N. En la parte (a), se observa el conjunto de puntos obtenido al aplicar el procedimiento de primer poligonalización, resultando en un conjunto de 9 puntos. En la poligonalización se han eliminado puntos entre P_0 y P_1, P_2 y P_3, P_4 y P_5, P_6 y P_7. En la parte (b) de la figura se muestra la envoltura convexa de los primeros 5 puntos. El ancho de la misma es w_1, que es menor que la tolerancia ε. La envoltura convexa que incluye los puntos de P_0 a P_5 no verifica que su ancho sea menor que la tolerancia. El conjunto de puntos entre P_4 y P_8 tiene ancho w_2, también menor que la tolerancia. En la parte (c) se muestran los segmentos resultantes de aproximar P_0-P_4 y P_4-P_8, identificando tres vértices de la poligonal que aproxima la curva 8N original, como se muestra en Fig. 2.(d).

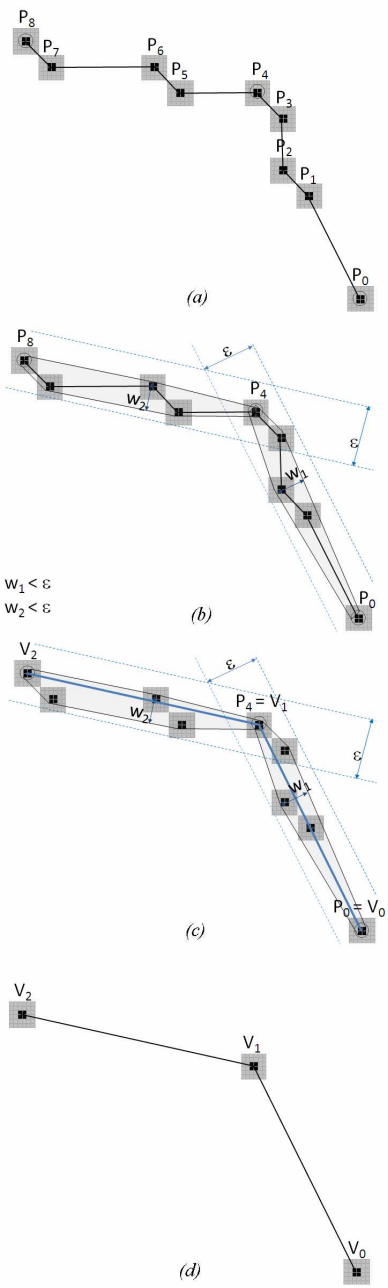

Fig. 2. (a) Primer poligonalización de una curva. (b) Envolturas convexas de P_0 a P_4 y de P_4 a P_8. (c) Vértices V_0, V_1, V_2 resultantes de la poligonalización. (d) Poligonal resultante.

IV. EJEMPLO DE APLICACIÓN DEL ALGORITMO

En la Fig. 3 se muestra el resultado de aplicar el algoritmo a la imagen original mostrada en la Fig. 1, con tolerancias $\varepsilon_1 = 2$ y $\varepsilon_2 = 5$.

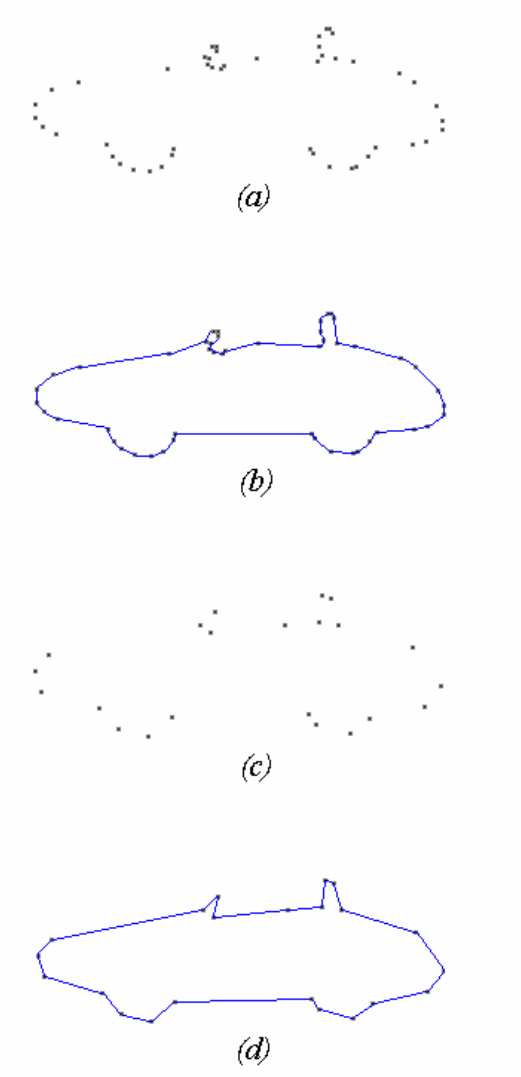

Fig. 3. (a) Vértices del polígono resultante de aproximar la curva de borde de la Fig. 1 con tolerancia ε = 2 píxeles. (b) Borde aproximado por segmentos, con ε = 2 píxeles. (c) Vértices del polígono resultante de aproximar la curva de borde con ε = 5 píxeles. (d) Borde aproximado por segmentos, con ε = 5píxeles.

En el ejemplo puede destacarse que la aproximación de curvas con poligonales usando el algoritmo propuesto con tolerancia ε = 2 píxeles se obtuvo una frontera simplificada consistente de 48 píxeles (Fig. 3.a), y la forma del objeto mantiene las características relevantes de la figura original (Fig. 3.b). Elevando la tolerancia a ε = 5 píxeles, el borde se reduce a 22 píxeles (Fig. 3.c). Obviamente, la calidad de la aproximación (Fig. 3.d) disminuye.

V. Conclusiones

El método propuesto para la simplificación del borde de un objeto digital permite representar al objeto con una cantidad reducida de puntos, manteniendo las características principales de la forma del objeto. Además, mediante la poligonalización de curvas de borde se logra eliminar efectos adversos de la digitalización.

El uso de envolturas convexas permite calcular el ancho de un conjunto de puntos de manera eficiente.

El control de la tolerancia queda en manos del usuario del algoritmo, quien podrá definirla según los requerimientos de la aplicación. Así, con baja tolerancia pueden obtenerse resultados de alta calidad, mientras que con tolerancias mayores el usuario gana en velocidad de procesamiento.

Creemos que la herramienta aquí presentada se destacará en el proceso de reconocimiento de objetos en movimientos, donde los requerimientos de eficiencia y flexibilidad son de importancia relevante.

Referencias

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

[2] E. Khalimsky et al., Digital Topology and its applications 36, pp. 1-17, 1990.

[3] U. Eckhardt – L. Latecki, Topologies for the Digital Spaces Z^2 and Z^3, Academic Press, Computer Vision and Image understanding 90, pp. 295-312, 2003.

[4] A. Rozenfeld, Digital Topology, The American Mathematical Monthly, 86(8) pp. 621 - 630, 1979.

[5] H. Liu, W. Liu, L. Latecki, Convex Shape Decomposition, IEEE 78-1-4244-6985-7/10, 2010

[6] M. De Berg et al. Computational geometry, ISBN 3-540-61270-X Springer-Verlag Berlin Heidelberg New York, 1997.

[7] A. Rosenfeld, E. Johnston, Angle Detection on digital curves, IEEE transactions on computers, 1973.

[8] U. Eckhardt, Digital Lines and Digital Convexity, Lecture Notes in Computer Science, Vol 2243, pp. 209-228, 2001.

[9] K. Kishimoto, Characterizing Digital Convexity and Straightness in Terms of ‘‘Length’’ and ‘‘Total Absolute Curvature’’, Computer vision and image understanding, vol. 63 (2), pp. 326–333, 1996.

[10] A. Gross, L. Latecki, Digital Geometric Invariance and Shape Representation, IEEE 0-8186-7190-4/95, 1995.

[11] M. Munich et al., Sifting through features with VIPR, IEEE 1070-9932/06, 2006.

[12] J. Kamlofsky, Topología Digital Base para la Visión Artificial, http://imgbiblio.vaneduc.edu.ar/fulltext/files/TC099930.pdf, 2011.