Análisis de rendimiento en extracción del código de cadena **VCC realizado en el lenguaje de programación Java**

Carlos Alfonso Pinilla-Garzón[1]
Universidad de Cundinamarca
calfonsopinilla@ucundinamarca.edu.co

Cesar Yesid Barahona-Rodríguez[2]
Universidad de Cundinamarca
cbarahona@ucundinamarca.edu.co

DOI: https://doi.org/10.21158/23823399.v7.n0.2019.2458

Fecha de recepción: 09 de septiembre de 2019
Fecha de aprobación: 07 de diciembre de 2019

Cómo citar este artículo: Pinilla-Garzón, C. A.; Barahona-Rodríguez, C. Y. (2019). Análisis de rendimiento en extracción del código de cadena VCC realizado en el lenguaje de programación Java. *Revista Ontare, 7,* 27-48. **DOI:** https://doi.org/10.21158/23823399.v7.n0.2019.2458

[1] Estudiante del programada de Ingeniería de sistemas en la Universidad de Cundinamarca. **ORCID:** https://orcid.org/0000-0001-8748-1123
[2] Magister en Sistemas Computacionales por la Universidad Metropolitana de Educación Ciencia y Tecnología - UMECIT, Panamá. Especialista en Gestión de Proyectos por la Universidad Nacional Abierta y a Distancia- UNAD, Colombia. Ingeniero en Telecomunicaciones de la Universidad Militar Nueva Granada. **ORCID:** https://orcid.org/0000-0001-7673-7381

RESUMEN

Las imágenes digitales ofrecen características que se pueden obtener computacionalmente. En este trabajo se hace especial énfasis en la obtención del contorno de la imagen y el perímetro, características estas propias de una imagen. La metodología seleccionada para este propósito es la técnica VCC de los códigos de cadena, con la que se extrae el contorno, y la extracción del perímetro mediante la suma del número de celdas en el borde acorde con la conectividad. Para el análisis del rendimiento del algoritmo se toma como variable de medición el tiempo de la extracción de la cadena, con el objetivo de compararla con escenarios de prueba y llevar a cabo un análisis del algoritmo. A fin de probar el rendimiento del código se puso a prueba por medio de mediciones del tiempo computacional en el que tardaban en extraer el código de la cadena VCC, y al realizar las pruebas el descriptor demostró ser lo suficientemente rápido para la recolección de 25, 50 y 100 imágenes. El tiempo obtenido es significativamente diminuto para el ser humano, y se concluye que el algoritmo que implementa el código de cadena VCC es lo suficientemente eficaz al extraer una cantidad límite de imágenes digitales.

Palabras clave: imágenes digitales; código de cadena VVC; Java; contorno; perímetro; extracción de código.

Analysis of the performance in Vertex Chain

Code extraction executed in Java program-ming language

ABSTRACT

Digital images offer features that can be obtained computationally. This work specially empha-sizes on obtaining the contour and perimeter, which are specific characteristics of the image. For this purpose, we selected the Vertex Chain Code (VCC) technique as the methodology to extract the contour and the perimeter of the images by means of the sum of the number of cells in the borders, according to the connectivity. For the analysis of the algorithm performance, the time of extraction of the chain was taken as a measurement variable, so that it could be com-pared with test scenarios and then, carry out the analysis of the algorithm. The performance of the code was tested by measuring the computational time for extracting the code from the VCC, and when performing the tests, the descriptor proved to be fast enough for the collection of 25, 50, and 100 images. The time obtained was significantly tiny for the human being, and it is con-cluded that the algorithm that implements the Vertex Chain Code (VCC) is sufficiently effective to extract a limited amount of digital images.

Keywords: digital images; Vertex Chain Code (VCC); Java; contour; perimeter; code extraction.

Análise de desempenho na extração de código **de cadeia VCC realizado na linguagem de programação Java**

RESUMO

Um dos paradigmas mais utilizados na literatura para a implementação de um sistema EEG BCI são os potenciais evocados visuais em estado estacionário, que normalmente surgem no córtex occipital do cérebro. Para visualizar, extrair e classificá-los, são necessárias várias etapas. A metodologia do estudo foi dividida nas fases iniciais do projeto de um sistema BCI: aquisição, pré-processamento, extração e classificação. Neste estudo, foi realizada uma caracterização desses potenciais desde a aquisição utilizando o equipamento g. Nautilus com o padrão 10-20 da Universidade Antonio Nariño até a classificação dos dados usando os métodos matemáticos CCA e SED em diferentes janelas de tempo. Assim, como na implementação de um sistema BCI em tempo real, espera-se que o tempo de classificação seja o mais curto possível para a execução rápida de um comando, esse tipo de estudo permite identificar quais métodos são os mais válidos na classificação desses dados, bem como algumas variáveis a serem consideradas. Os resultados permitem identificar, então, uma melhor efetividade na classificação dos dados com o CCA do que com o SED, além do comportamento do sistema de acordo com as janelas de tempo.

Palavras-chave: imagens digitais; Código de cadeia VVC; Java; contorno perímetro; extração de código.

Analyse des performances d'extraction du code **VCC du langage de programmation Java**

RÉSUMÉ

Les images numériques offrent des fonctionnalités obtenues par la puissance de calcul des ordinateurs. Ce travail met l'accent sur l'obtention du contour de l'image et de son périmètre. Cette investigation utilise une méthodologie technique VCC des codes chaînes permettant l'extraction des contours et du périmètre en ajoutant le nombre de cellules en fonction de la connectivité. Pour analyser les performances de l'algorithme, le temps d'extraction de la chaîne est pris comme variable de mesure et compraré ensuite aux scénarios de test d'analyse algorithmique. Pour tester les performances du code et mesurer le temps de calcul nécessaire à son extraction de la chaîne VCC, l'utilisation du descripteur s'est avéré suffisamment rapide pour collecter 25, 50 et 100 images. Le temps d'extraction est ici minime, et l'algorithme mis en place pour le code de chaîne VCC est suffisamment efficace pour extraire une quantité importante d'images numériques.

Mots clés: images numériques; Code de chaîne VVC; Java; contour; périmètre; extraction de code.

1. Introducción

Las imágenes digitales son una gran fuente de información, ya que contienen bastantes características con las que se pueden trabajar. La binarización de las imágenes digitales será de gran relevancia para el procesamiento de las imágenes digitales 2D. Esto ayudará a validar dos estados en la imagen —blanco y negro—, y, con esto, obtener características a considerar, las cuales, en este trabajo, serán el contorno y el perímetro.

El contorno se extrae con la metodología del código de cadena VCC. Este se utiliza a fin de obtener la cadena de los vértices del contorno de una región digital. La metodología del código de cadena VCC nos ayuda a transformar regiones digitales bidimensionales en representaciones unidimensionales, lo que contribuye a disminuir el almacenamiento en el espacio de memoria y a que sea susceptible de trasmitir información de manera comprimida.

El perímetro de la imagen hace uso de un algoritmo que obtiene la cantidad de cambios de estados del píxel por píxel en la imagen binaria, y si se genera algún cambio se acumula en la cantidad de cambios. Cabe afirmar que en este algoritmo no se encuentran las capacidades de obtener perímetros diferentes, ya que, si hay perímetros internos y externos en la misma figura, será un acumulado total de ambos.

El principal enfoque del trabajo está en el código de cadena VCC, el cual se pondrá a prueba por medio de una medición de tiempo computacional, tomada en nanosegundos y convertida en segundos, de modo que se define su rendimiento en la extracción del código de cadena VCC en una cantidad límite de imágenes digitales. Sin embargo, antes de realizar las pruebas de rendimiento se realiza un escenario de pruebas para cada imagen; en este la imagen será clasificada según la cantidad de vértices, píxeles, diferentes figuras, formas, tamaños y posiciones, con el propósito de obtener similitudes y observar los cambios que se generan al obtener el tiempo y el código de cadena VCC.

2. Desarrollo del contenido

2.1 Imágenes binarias

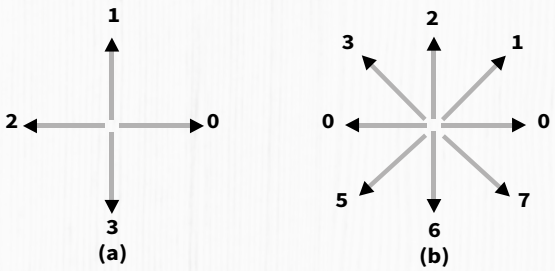
Según Morales (2013), una imagen digital se define matemáticamente como una función bidimensional f (x, y), donde x, y son coordenadas especiales en un plano, y f —en cualquier par de coordenadas— es la intensidad o nivel de gris de la imagen en esa coordenada.

2.2 Códigos de cadena

El código de cadena es un tipo de estructura de datos que permite representar el contorno de un objeto en una imagen binaria mediante una secuencia de segmentos conectados de forma consecutiva, de longitud y orientación específica, en la cual conecta píxeles adyacentes.

Según Kui y Zalik (2005), la conexión de los segmentos se lleva a cabo en entornos de 4-conecciones u 8-conecciones. Cuando se usa un entorno 4-conecciones se tienen cuatro orientaciones posibles para los segmentos, tal como se muestra en la figura 1a y se utilizan los números 0, 1, 2 y 3 a fin de especificar las orientaciones. Si se aplica un entorno 8-conecciones se cuenta con ocho orientaciones posibles, tal como se muestra en la figura 1b.

Figura 1. a) Código de cadena de Freeman 4 direccional; b) código de cadena de Freeman 8 direccional.



Fuente. Bribiesca, 1999.

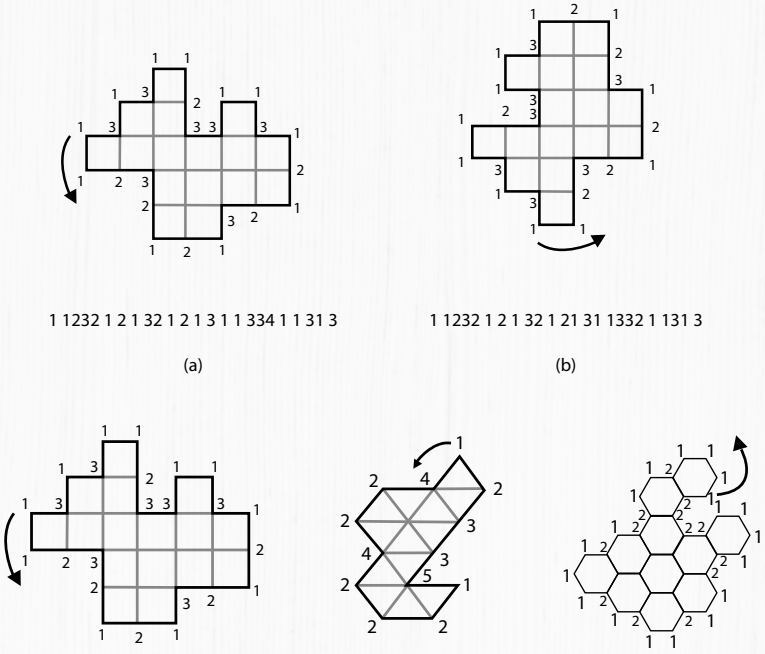
Código de cadena VCC.

El código de cadena tipo VCC se emplea con el propósito de representar formas en dos dimensiones compuestas por celdas regulares. Además, satisface los tres objetivos propuestos por Freeman: a) preservar fielmente la información de interés; b) permitir un almacenamiento comprimido; y c) facilitar procesos requeridos, como, por ejemplo, comparación o reconocimiento de forma.

El código de cadena tipo VCC tiene las siguientes características propuestas por Bribiesca (1999):

- Permitir una reconstrucción rápida y fiel.
- Es invariante la traslación y rotación de la forma, lo que facilita la comparación de objetos (Figura 2).
- El VCC muestra una enorme capacidad en la representación de formas compuestas por celdas regulares, bien sean triangulares, cuadradas o bien hexagonales (Figura 3). Al hacer uso de celdas cuadradas —píxeles— se cuenta con cuatro diferentes tipos de vértices. Los vértices tipo 1 son aquellos que tienen contacto con el borde, además, una celda regular —píxel— tiene contacto con dicho vértice. Los vértices tipo 2 son aquellos que tienen contacto con el borde, y además dos celdas regulares tienen contacto con este último vértice. Los tipos 3 son vértices que tienen contacto con el borde y además tres celdas regulares tienen contacto con este vértice. Finalmente, los vértices tipo 4 son aquellos vértices internos que tiene contacto con cuatro celdas regulares.
- Un conjunto de descriptores de forma puede extraerse de manera directa de la cadena, sin necesidad de ir a la representación matricial.
- El código de cadena VCC es una representación compacta de un objeto binario. Suministra una buena compresión de la descripción del contorno ya que cada elemento de la cadena se puede codificar solo con 3 bits.

Figura 3. Formas compuestas por distintos tipos de celdas

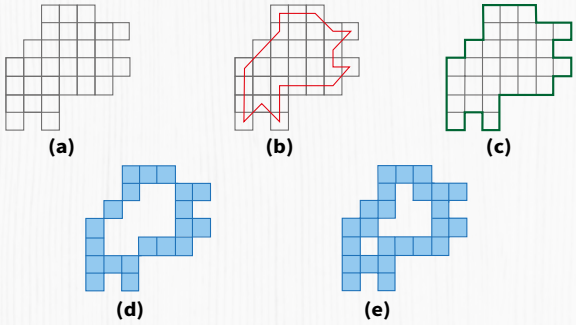


Fuente. Bribiesca, 1999.

2.3 Perímetro

El perímetro es uno de los descriptores geométricos básicos para una región, el cual ocasiona un valor cuantitativo a la frontera de la forma. Para medir el perímetro de una región digital en el espacio existen tres enfoques reportados en la literatura (Figura 4) (Morales, 2013): a) utilizando los lados frontera de las celdas en la región, b) visto como la suma de las distancias entre los centros de las celdas, y c) sumando el número de celdas en el borde acorde con la conectividad.

Figura 4. a) los tres enfoques para la medición del perímetro de una región digital en el espacio discreto; b) visto como la suma de las distancias entre los centros de las celdas; c) utilizando los lados frontera de las celdas en la región; d) y e) finalmente, sumando el número de celdas en el borde acorde con la conectividad usada

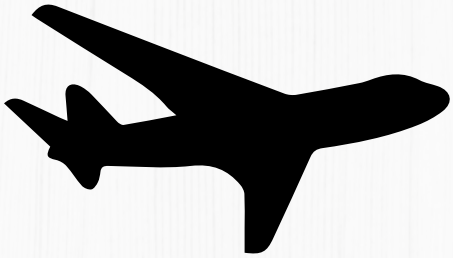


Fuente. Bribiesca, 1999.

3. Metodología

Para el buen funcionamiento del programa se debe contar con un escenario de imágenes prueba con las siguientes características: a) imágenes digitales que no contengan un fondo; b) la imagen debe tener solo dos valores en los píxeles, negro o blanco; c) la forma o figura, a la cual se le extraiga del código de cadena VCC, y el perímetro debe ser de color negro; d) el tamaño máximo de la imagen debe ser de 1024 pixeles de ancho y 720 pixeles de alto; y e) la imagen no puede contener agujeros o huecos internos.

Figura 5. Imagen de ejemplo que cumple las características propuestas para el buen funcionamiento de los algoritmos implementados



Fuente. Morales, 2013.

El objetivo que debe cumplir el programa es tomar el tiempo que tarda en obtener el código de cadena VCC de una cantidad límite de imágenes, además, obtendrá la cantidad de vértices y el perímetro de cada imagen.

El programa se desarrolla en los siguientes pasos:

- Cargar imagen por imagen del directorio en el que se encuentren las imágenes.
- Validar que la imagen se encuentre en el rango de las medidas.
- Evaluar los dos únicos valores que puede contener un píxel de la imagen, ya sea blanco o negro.
- Convertir la imagen en una matriz x, y de dos valores; será de color negro y blanco.
- Hallar la posición inicial de la imagen será el primer píxel negro que encuentre en la matriz x, y, con ello, el código de cadena VCC ya tendrá un punto de partida.
- Tomar el tiempo inicial en nanosegundos.
- Extraer el código de cadena VCC desde la posición inicial por medio de los movimientos que se definen por cuatro números: 0 será el movimiento al lado derecho, 1 será para abajo, 2 será para el lado izquierdo y 3 será para arriba; estos movimientos se definen conforme a la conectividad de celdas. A fin de finalizar la extracción del código se valida en cada movimiento si se encuentra en la posición inicial, con lo cual sabremos que ya recorrió todo el contorno de la imagen.
- Tomar el tiempo final en nanosegundos.
- Guardar en un archivo de texto el código de cadena VCC obtenido, con el nombre de la imagen y extensión txt.
- Retomar el tiempo inicial y el final, y realizar una diferencia de tiempos a fin de tomar el tiempo que tardó en obtener el código de cadena VCC, y convertir así ese tiempo en nanosegundos en segundos.

- Extraer el perímetro; acá recorrerá todos los píxeles y evaluará el cambio de color entre el píxel (x, y) y los píxeles (x+1, y) y (x, y+1), así irá acumulando el cambio de colores entre los píxeles.
- Después de obtener todos los datos se plasmarán en una tabla en la que se mostrará el tiempo total, el tiempo y el perímetro de cada imagen.

Figura 6. Esquema para obtener el VCC

Dirección derecha Movimiento 0	Z Z **1** Z x Z Z Z Z **Código = 3** **j++** **i--**	Z Z Z Z x **0** Z Z Z **Código = 1** **Movimiento = 1**	Z Z Z Z x **1** Z Z Z **Código = 2** **j++**
Dirección abajo Movimiento 1	Z Z Z Z x Z Z Z **1** **Código = 3** **j++** **i++**	Z Z Z Z x Z Z **0** 1 **Código = 1** **Movimiento = 2**	Z Z Z Z x Z Z **1** 1 **Código = 2** **i++**
Dirección izquierda Movimiento 2	Z Z Z Z x Z **1** Z Z **Código = 3** **j--** **i++**	Z Z Z **0** x Z Z Z Z **Código = 1** **Movimiento = 3**	Z Z Z **1** x Z Z Z Z **Código = 2** **j--**
Dirección arriba Movimiento 3	**1** Z Z Z x Z Z Z Z **Código = 3** **j--** **i--**	Z **0** Z Z x Z Z Z Z **Código = 1** **Movimiento = 0**	Z **1** Z Z x Z Z Z Z **Código = 2** **i--**

Nota: los 1 representan píxeles que pertenecen a la región, los 0 representan píxeles pertenecientes al fondo, x representa el centro de la máscara y el valor de las z no es relevante.

Fuente. Morales, 2013.

4. Resultados

Se realizaron varias pruebas en la obtención del código de cadena VCC, en las cuales las imágenes se categorizaron por los siguientes datos: cantidad de píxeles, vértices y perímetros. De ahí se tomaron los resultados que se presentan a continuación.

La tabla 1 muestra los resultados obtenidos en la categorización por píxeles. Las imágenes se dividen en grupos de imágenes en rangos de mil, diez mil, cien mil y un millón de píxeles.

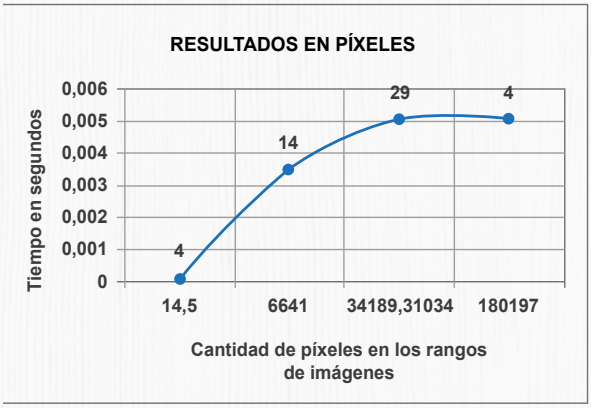
Tabla 1. Datos obtenidos de las imágenes categorizadas por cantidad de píxeles

Rangos de píxeles	Promedios		
	Imágenes	Cantidad	Tiempo
(0-1000)	4	14,5	0,0000967
(1001-10000)	14	6641	0,00350946
(10001-100000)	29	34189,3103	0,00508118
(100001-1000000)	4	180197	0,00507112
Total:	51	221041,81	0,01375846

Fuente. Elaboración propia.

Notamos que los últimos grupos tienden a un mismo tiempo y en un grupo es más elevado que el otro por una diferencia de 25 imágenes más. Esto sugiere que la cantidad de píxeles en una imagen puede retardar más el código.

Figura 7. Resultados gráficos de la Tabla 1



Fuente. Elaboración propia.

La tabla 2 muestra los resultados obtenidos por la categorización por perímetro, en la cual las imágenes se dividen por grupos de imágenes en rangos de mil y diez mil.

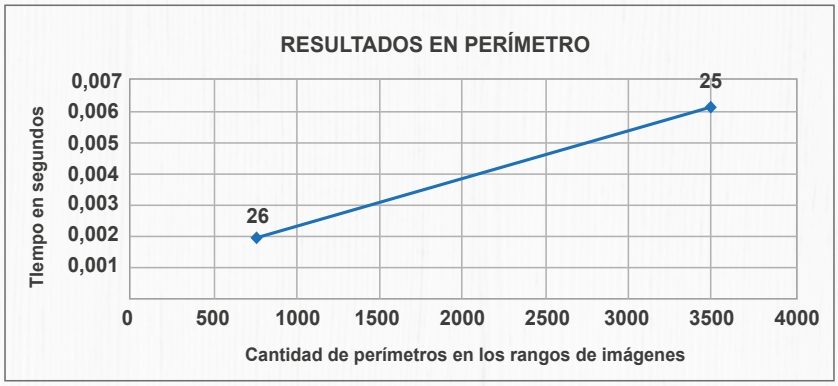
Tabla 2. Datos obtenidos de las imágenes categorizadas por cantidad de perímetros

Rangos de perímetros	Promedios		
	Imágenes	Cantidad	Tiempo
(0-1000)	26	656,730769	0,00193252
(1001-10000)	25	3691,42857	0,00616286
Total:	51	4348,15934	0,00809538

Fuente. Elaboración propia.

La figura 8 muestra el aumento del tiempo con respecto a la cantidad de perímetros en los rangos de imágenes. Notamos el aumento de tiempo, ya que su diferencia es el perímetro de las imágenes que están divididas en dos grupos similares.

Figura 8. Resultados gráficos de la Tabla 2



Fuente. Elaboración propia.

La tabla 3 muestra los resultados obtenidos por la categorización por vértices, en la cual las imágenes se dividen por grupos de imágenes en rangos de mil y diez mil.

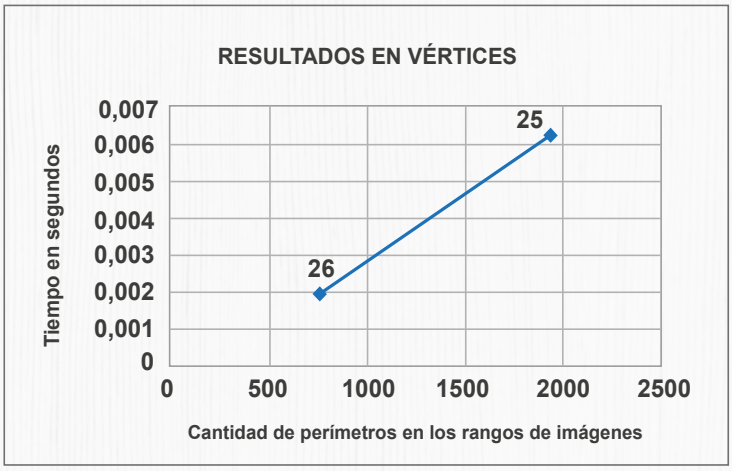
Tabla 3. Datos obtenidos de las imágenes categorizadas por cantidad de vértices

Rangos de vértices	Promedios		
	Imágenes	Cantidad	Tiempo
(0-1000)	26	656,307692	0,00206352
(1001-10000)	25	2065,2	0,0061745
Total:	51	2721,50769	0,00823802

Fuente. Elaboración propia.

La figura 9 muestra el aumento del tiempo con respecto a la cantidad de vértices en los rangos de imágenes. Notamos el aumento de tiempo, ya que su diferencia es el perímetro de las imágenes que están divididas en dos grupos similares.

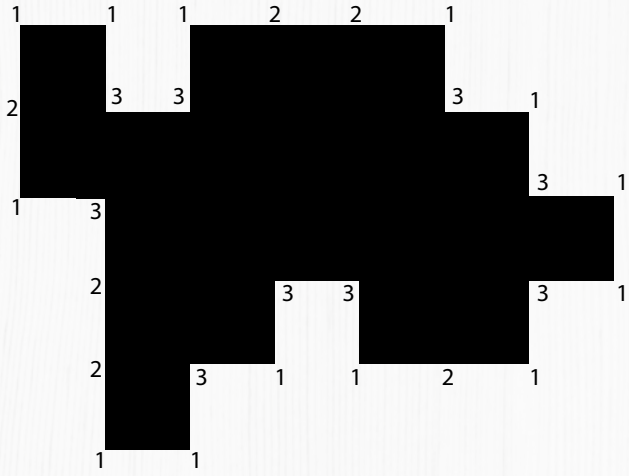
Figura 9. Resultados gráficos de la Tabla 3



Fuente. Elaboración propia.

A fin de verificar el funcionamiento de los algoritmos implementados, se presenta la figura 10 como una imagen de prueba. En los resultados mostrados después, la imagen de prueba se denomina «mancha.png».

Figura 10. Imagen binaria de prueba, con número de vértices en el contorno



Fuente. Morales, 2013.

A fin de realizar la obtención del código de cadena VCC, la imagen debe estar previamente binarizada, tal como se muestra en la figura 10. Con este el algoritmo se obtiene el código que se muestra en la figura 11, el cual se almacena en un archivo de texto.

Figura 11. Código de cadena VCC de la imagen binaria de prueba

```
1133122131311312133131122312
```

Fuente. Morales, 2013.

El algoritmo que obtiene el perímetro trabaja después de extraer el tiempo y el código de cadena VCC. Al finalizar esta extracción de información de la imagen procede a obtener el perímetro, en el que procesa la imagen binaria. En este caso será la imagen de prueba, en la cual su resultado es igual a 28, sumando todos los lados externos de los píxeles negros de la figura de prueba. El resultado se observa en la figura 12.

A fin de llevar acabo el experimento se tomaron unos conjuntos de imágenes binarias, formados en tres grupos, que contienen diferentes cantidades de imágenes. Para realizar el análisis de rendimiento del algoritmo se ejecuta el código con el grupo de imágenes, en el cual serán recorridas por su contorno de izquierda a derecha y luego de arriba a abajo.

Figura 12. Conjunto de imágenes binarias

Fuente. Morales, 2013.

Los resultados que obtenemos al trabajar tres diferentes grupos de imágenes se exponen en la figura 13, la figura 14 y la figura 15, los cuales presentan tiempos diferentes, que cambian en razón a la cantidad de imágenes y su complejidad; también se obtienen los códigos de cada imagen, los cuales se guardan en archivos de texto.

Figura 13. Resultados del conjunto de 25 imágenes

1. Tiempo: 0.001649489 s				
#	**Nombre**	**Tiempo(n's)**	**Vértice**	**Perímetro**
0	x.png	476177	1140	1140
1	mariposa2.jpg	191921	1992	1992
2	L.png	180620	988	988
3	circulo.png	70156	668	668
4	caramelo.png	139481	1974	1974
5	rectangulo.png	35254	738	738
6	mancha.png	1688	28	28
7	S.png	58398	1024	1024
8	triangulo.png	33075	1126	1126
9	punto.png	447	4	4
10	t.png	18956	736	736
11	paleta.png	19671	762	762
12	cuadro.png	14222	884	884
13	estrella.png	39154	1384	1384
14	fig_hueco2.png	1468	24	32
15	fig_hueco.png	1180	20	24
16	linea.png	19661	760	760
17	flecha.png	30368	1006	1006
18	cilindro,png	29662	968	968
19	casa.png	39791	1362	1362
20	avion.png	67942	2114	2114
21	T.png	15912	808	808
22	N.png	35699	1796	1796
23	hombre.jpg	30343	1254	1254
24	perro2.png	98243	3152	3152

Fuente. Elaboración propia.

Figura 14. Resultados del conjunto de 50 imágenes

1. Tiempo: 0.004456209 s				
#	**Nombre**	**Tiempo(n's)**	**Vértices**	**Perímetro**
0	gaviota_rot.jpg	1012939	3334	3334
1	x.png	272241	1140	1140
2	pato.jpg	279927	2386	2386
3	leon2.jpg	385128	4686	4694
4	leon2_rot.jpg	281885	4686	4694
5	mariposa2.jpg	92917	1992	1992
6	mujer.png	56364	1174	1174
7	gato.png	47277	1076	1080
8	leona.jpg	41327	968	968
9	L.png	42976	988	988
10	circulo.png	32277	668	668
11	caramelo.png	97406	1974	1974
12	rectangulo.png	32297	738	738
13	mancha.png	2841	28	28
14	S.png	50448	1024	1024
15	perro_rot.jpg	79905	1656	1656
16	triangulo.png	55525	1126	1126
17	punto.png	606	4	4
18	mujer_rot.png	99267	1174	1174
19	Flecha_rot.png	59232	1218	1218
20	liebre.jpg	21016	580	580
21	caballo.jpg	184114	4936	4948
22	t.png	21376	736	736
23	puerco.jpg	18630	700	700
24	jirafa.jpg	43127	1778	1778

Fuente. Elaboración propia.

Figura 15. Resultados del conjunto de 100 imágenes

1. Tiempo: 0.006189955 s				
#	**Nombre**	**Tiempo(n's)**	**Vértices**	**Perímetro**
0	gaviota_rot.jpg	1013334	3334	3334
1	x.png	119956	1140	1140
2	pato.jpg	162446	2386	2386
3	araña2.jpg	91451	1370	1370
4	leon2.jpg	213891	4686	4694
5	loro.jpg	46327	1734	1760
6	hombre_rot.jpg	32939	1254	1254
7	leon2_rot.jpg	161740	4686	4694
8	caballo_rot.jpg	163390	4936	4948
9	pinguino_rot.jpg	111306	3590	3590
10	cocodrilo_rot.jpg	125618	3488	3488
11	mariposa2.jpg	40264	1992	1992
12	gato_rot.png	42670	1076	1080
13	jirafa_rot.jpg	40351	1778	1778
14	mujer.png	53392	1174	1174
15	gato.png	40883	1076	1080
16	leona.jpg	21516	968	968
17	L.png	19806	988	988
18	circulo.png	20747	668	668
19	hoja3.jpg	21892	988	988
20	pinguino.jpg	121875	3590	3590
21	caramelo.png	84810	1974	1974
22	rectangulo.png	18843	738	738
23	mancha.png	1738	28	28
24	S.png	31980	1024	1024

Fuente. Elaboración propia.

5. Conclusiones

A fin de probar el rendimiento del código se puso a prueba por medio de mediciones del tiempo computacional en el que tardaban en extraer el código de la cadena VCC, con una cantidad N de imágenes digitales, en las cuales se dan tiempos diferentes, ya que las imágenes serán diferentes en varios aspectos, bien sea en su contorno o bien en el tamaño de la figura. Al realizar las pruebas el descriptor demostró ser lo suficientemente rápido para la recolección de 25, 50 y 100 imágenes. El tiempo obtenido es significativamente diminuto para el ser humano, y se concluye que el algoritmo que implementa el código de cadena VCC es suficientemente eficaz al extraer una cantidad límite de imágenes digitales.

Aunque el algoritmo que implementa el código de cadena VCC tiene sus limitaciones, ya que solo obtiene el código de la cadena de vértice del contorno externo, si se llegaran a realizar pruebas con imágenes con huecos solo tomaría el contorno exterior, ya que el algoritmo no cuenta con la extracción de contornos internos. Además, las imágenes deben ser binarias, dado que el algoritmo evalúa dos colores: blanco y negro.

Referencias

Bribiesca, E. (1999). A new chain code. *Pattern Recognition,* 32(2), 235-251. DOI: https://doi.org/10.1016/S0031-3203(98)00132-0

Kui, Y.; Zalik, B. (2005). An efficient chain code with Huffman coding. *Pattern Recognition*, 38(4), 553-557. DOI: https://doi.org/10.1016/j.patcog.2004.08.017

Morales, A. (2013). *Descripción y clasificación de formas 2D usando códigos de cadena tipo VCC* (Tesis de maestría). Instituto Tecnológico de León, México.