EFFICIENT REPRESENTATION OF SEGMENTATION CONTOURS USING CHAIN CODES

Francesco Verdoja, Marco Grangetto

Università degli Studi di Torino, Dept. of Computer Science

ABSTRACT

Segmentation is one of the most important low-level tasks in image processing as it enables many higher level computer vision tasks like object recognition and tracking. Segmentation can also be exploited for image compression using recent graph-based algorithms, provided that the corresponding contours can be represented efficiently. Transmission of borders is also key to distributed computer vision. In this paper we propose a new chain code tailored to compress segmentation contours. Based on the widely known 3OT, our algorithm is able to encode regions avoiding borders it has already coded once and without the need of any starting point information for each region. We tested our method against three other state of the art chain codes over the BSDS500 dataset, and we demonstrated that the proposed chain code achieves the highest compression ratio, resulting on average in over 27% bit-per-pixel saving.

Index Terms— Superpixel, chain codes, 3OT, compression

1. INTRODUCTION

Image segmentation is the process of partitioning an image into distinct semantically meaningful regions. It serves as foundation for many high-level computer vision tasks, such as scene understanding [1] and object recognition [2]. Moreover, if detected region contours in images are compressed efficiently as side information, they might enable advanced image/video coding approaches based on shape-adaptive graph transform encoders [3, 4] and motion predictors of arbitrarily shaped pixel-blocks [5]. Lastly, efficiently coded contours can be used, at a much lower coding cost than compressed video, in the context of distributed computer vision, to perform computation intensive object detection or activity recognition [6].

To compress borders, chain code techniques are widely used as they preserve information and bring considerable data reduction. They also allow various shape features to be evaluated directly from this representation; edge smoothing and shape comparison are also easily computed [7]. The ability of chain codes to describe regions by mean of their border shape is demonstrated to be the most efficient way to deal with this task; in [8, 9] it is shown that algorithms using chain codes achieve higher compression rate than JBIG [10], the ISO/IEC standard for compression of bi-level images.

The context of segmentation region borders, however, presents one characteristic that standard chain codes are not tailored to: since all image pixels must be assigned to a region, all borders are shared between two regions. It follows that, if one chain code per border is used, all edges will be encoded twice, resulting in an higher number of symbols. Moreover, every chain code needs an edge coordinate, from where the code sequence is started.

In this work, we propose an algorithm able to produce chain codes to encode efficiently borders of segmentation regions exploiting the following properties: 1. every border is visited and encoded only once; 2. the starting coordinate of the chain code is not needed as it is known implicitly; 3. the distribution of the chain code symbols is likely to be highly skewed so as to be amenable to efficient entropy coding.

This work has been partially supported by Sisvel Technology Ph.D. scholarship.

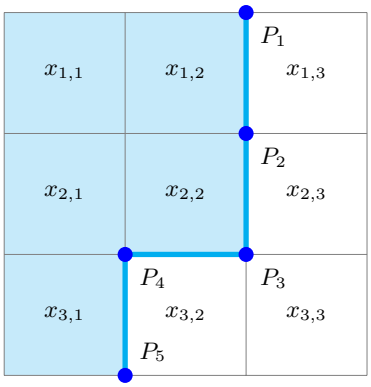


Fig. 1. A 3×3 image segmented into two regions; the active crack-edges are outlined in blue.

The paper is organized as follows: in Section 2 we will first review the state of the art for standard chain codes; then, we will present our approach in Section 3; lastly, in Section 4 we'll compare our performance with those of other techniques over a standard segmentation dataset, to show that our approach is able to achieve significant gains over classical chain codes.

2. CHAIN CODES

Chain code algorithms encode binary regions by describing their contours using sequences of symbols from an alphabet. The contour map of a binary input image I is represented by the so called horizontal and vertical *crack-edges*. They are the contour line segments that separate two adjacent pixels: if the pixels belong to two different regions, the crack-edge is called *active*; otherwise, if they belong to the same region, the crack-edge is called *inactive*. The two ends of an active crack-edge are called vertices and are denoted as P_k and P_{k+1}. Chain code algorithms encode active crack-edges by virtue of the position of their surrounding vertices. Figure 1 shows an example of a 3×3 sample image containing two regions: $r_1 = \{x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}, x_{3,1}\}$ and $r_2 = \{x_{1,3}, x_{2,3}, x_{3,2}, x_{3,3}\}$. The contour map separating the two regions is represented using a vertex vector $\Gamma = [P_1 P_2 P_3 P_4 P_5]$. The chain code algorithm translates the vector of consecutive vertices Γ into a vector of chain code symbols $\Sigma = [S_1 S_2 S_3 S_4 S_5]$ by encoding a vertex P_{k+2} according to the previous two vertices P_k and P_{k+1}. It has to be noted that for the first two vertices P_1 and P_2, some convention has to be used.

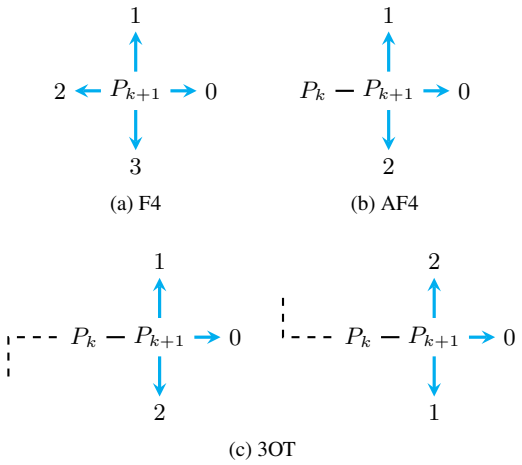


Fig. 2. Graphical representation of different chain codes

The decoding process then takes Σ and, applying an inverse translation, computes Γ. It then reconstruct the binary image I by filling the regions enclosed by the crack-edges in Γ. Since all vertices are encoded according to their relative position to P_1, the absolute position of the the latter has usually to be provided somehow as side information to the decoder. Σ is then further compressed with entropy coding techniques, e.g. Huffman, adaptive arithmetic coding, or context tree coding [11, 12].

2.1. Freeman chain codes (F4 and AF4)

One of the first algorithms developed is the Freeman chain code [13]. In a 4-connectivity context, the algorithm F4 assigns a code from a 4-symbol alphabet $\{0, 1, 2, 3\}$ to P_{k+2} based on its relative position from P_{k+1}, according to the scheme presented in Figure 2a.

Since one of the four directions is the one where P_k is, and $P_{k+2} \neq P_k$, we know that just three symbols should be enough to discriminate the remaining three directions. Differential Freeman (AF4) [11] uses the scheme illustrated in Figure 2b, where the symbol "0" is used for "go forward", "1" for "turn left" and "2" for "turn right" according to the direction of the segment connecting P_k and P_{k+1}.

2.2. Three OrThogonal symbol chain code (3OT)

The 3OT algorithm [14] uses a 3-symbol differential approach similar to AF4, but exploits one extra information: it keeps track of the last time that there has been a change in direction. Then, "0" still means "go forward", but now "1" means "turn accordingly to the direction you were facing before the previous turn" while "2" means "turn the opposite way to the one you were facing before the previous turn". As can be seen in Figure 2c, when the previous direction is facing upward, turning upward again is coded as "1", while turning downward is coded as "2"; viceversa, when the previous direction is facing downward, it's turning upward that is coded as "2", while turning downward is coded as "1".

3OT has been reported as one of the better performing chain codes in the state of the art [8, 9].

Fig. 3. Image segmented into 150 regions with borders shown in red

3. THE PROPOSED TECHNIQUE

In this section we'll present an algorithm to encode a segmentation of an image using a chain code to describe the borders of the segmented regions. The framework proposed might be used in conjunction with any standard chain code; in this study we work with 3OT as base chain code, given its aforementioned qualities. From now on, we'll refer to our approach as *Segmentation-3OT* (S-3OT). S-3OT uses the same alphabet of 3OT with an added symbol (i.e. "3"), the meaning of this symbol is going to be explained in detail here.

Let's start by defining an n regions segmentation of an image $I = \{x_i\}_{i=1}^{N}$ with N pixels as a partition $R = \{r_i\}_{i=1}^{n}$ of the pixels of I; more precisely, the segmented regions must satisfy the following constraints:

$$\begin{aligned} &\forall x \in I, \quad \exists r \in R \,|\, x \in r \;; \\ &\forall r \in R, \quad \nexists r' \in R - \{r\} \;|\, r \cap r' \neq \emptyset \;. \end{aligned} \tag{1}$$

For each region $r_i \in R$, we call $\Gamma_i = [P_k^i]_{k=1}^{m}$ the vector containing all m vertices of the active crack-edges of r_i, sorted clockwise, starting from a vertex determined accordingly to some convention. Please note that all crack-edges touching the image border are considered active and are included in Γ_i. Also, note that since the region is closed, $P_m^i = P_1^i$.

In Figure 3 a possible segmentation of a sample image is shown; given one segmentation, the red borders represent the information we need to encode in a symbol sequence Σ. From the region borders, obtained by Σ, the decoder can then assign to each closed region a different label to reconstruct the segmentation.

One approach might be to encode the whole border grid at once; chain codes follow a single path along a border and therefore it would be necessary to keep track of all the crossings that in turn could require a significant coding overhead. Another approach might be to encode the borders region by region: to this end, one might apply a standard chain code to each region border. By doing so, however, one would encode each crack-edge twice, as each border is always shared between two regions. A possible countermeasure to the previous issue is to use some convention to decide which of the two regions "owns" a specific crack-edge, e.g. all crack-edges are owned by the left-most or top-most region. Then, when we are encoding one region we would skip the crack-edges not owned by that region; also this approach requires some coding overhead to signal the offset to jump to the next owned coordinate of the edge. Lastly, to encode regions by chain codes we need to specify a starting position in some way.

S-3OT uses a hybrid approach that borrows some ideas from both the approaches we have just discussed: it proceeds region by region, but it keeps track of the path it has already covered once,

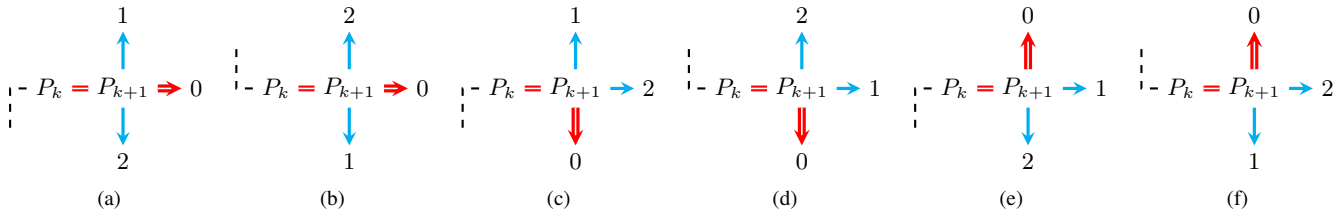


Fig. 4. Graphical representation of the chain codes assigned by S-3OT to P^i_{k+2} according to Rule 2; the crack-edges marked with double red lines are lying on the known borders.

avoiding to encode it twice. S-3OT has been developed with a few desired properties in mind:

Property 1. The decoding process should require no information other than the sequence Σ and the sizes of the image. No offsets or starting positions are used for each chain code.

Property 2. The decoder will go through Σ in the same order as the encoder; for this reason, when processing a region, information on the previously encoded regions is available and should be exploited.

Property 3. The chain code symbols must be selected so as that their probability distribution is likely to be highly skewed to favor the following entropy coding.

We'll proceed here in explaining S-3OT algorithm. S-3OT maintains a vector $\overline{\Gamma}$ of vertices which have been already encoded. $\overline{\Gamma}$ is initialized with all vertices lying on the image canvas starting from the top left most vertex of the image and going clockwise around the image border. $\overline{\Gamma}$ is going to be used as "context" during the encoding to adhere to the aforementioned Property 2. S-3OT also maintains a set $\overline{R}$ which contains the regions still to be encoded; initially $\overline{R} = R$. Then, until $\overline{R} = \emptyset$, the algorithm selects the region $r_i \in \overline{R}$ containing the pixel $\overline{x}$ in the most top-left position among the regions still in $\overline{R}$; it then encode $\Gamma_i = [P^i_k]_{k=1}^m$ using the vertex in the top-left corner of $\overline{x}$ as P^i_1 and then enumerating the vertices clockwise. Using this convention, no starting point coordinates has to be transmitted (Property 1). Also, P^i_1 and P^i_2 don't need to be encoded, as their position is always known by the way P^i_1 is selected: they will always lay on the top crack-edge of $\overline{x}$. In other words, we are sure that the left and top crack-edge of $\overline{x}$ have already been coded, otherwise $\overline{x}$ wouldn't be the selected pixel.

Let's call $\pi(\Gamma, [P_k P_{k+1}])$ a function that, given a vector of vertices Γ and two consecutive vertices P_k and P_{k+1}, returns the vertex P_{k+2}, if $[P_k P_{k+1}] \in \Gamma$. Also, let's call $3OT([P_k P_{k+1} P_{k+2}])$ the function that returns the symbol that 3OT returns for the vertex sequence $[P_k P_{k+1} P_{k+2}]$.

Then, from Γ_i, the chain code Σ_i is constructed according to the following rules to determine the symbol $S^i_{k+2} \in \{0, 1, 2, 3\}$ to be assigned to the vertex P^i_{k+2}, given P^i_k and P^i_{k+1}. We'll call $\overline{P}_{k+2}$ the vertex returned by $\pi(\overline{\Gamma}, [P^i_k P^i_{k+1}])$, i.e. the next vertex on the known border after P^i_k and P^i_{k+1}.

Rule 1 (Follow the border)**.** This rule is applied when we are on a known border, more precisely when $[P^i_k P^i_{k+1}] \in \overline{\Gamma} \ \wedge \ P^i_{k+2} = \overline{P}_{k+2}$ .When this condition is met, $S^i_{k+2} = 0$. Please note that in this context "0" is used even if the border is changing direction.

Rule 2 (Leave the border)**.** When leaving the known border, just two directions have to be discriminated, since out of the four possible, one is where P^i_k is, and the other is where the known border would continue. Moreover, the symbol "0" can't be used, as it would be interpreted according to Rule 1 by the decoder. We'll then use symbols "1" and "2" to discriminate between the two possible directions. More precisely when $[P^i_k P^i_{k+1}] \in \overline{\Gamma} \ \wedge \ P^i_{k+2} \neq \overline{P}_{k+2}$, S-3OT assigns a symbol according to the following:

$$S^i_{k+2} = \begin{cases} S^{3OT}_{k+2} & \text{if } S^{3OT}_{k+2} \neq 0\,, \\ 3OT([P^i_k P^i_{k+1} \overline{P}_{k+2}]) & \text{otherwise;} \end{cases} \tag{2}$$

where $S^{3OT}_{k+2} = 3OT([P^i_k P^i_{k+1} P^i_{k+2}])$. A graphical representation of this rule is given in Figure 4. It can be noticed there that when the known border is proceeding straight (Figure 4a and Figure 4b), the symbols assigned to the other directions are the same 3OT would have used. In all other cases, the known border is not straight. In this cases, if P^i_{k+2} is straight ahead of P^i_k and P^i_{k+1}, we use the symbol that 3OT would have assigned to the direction occupied by the known border; instead, the other direction maintains the corresponding 3OT symbol. To give one example, in the case presented in Figure 4c, if the direction to follow is upward the symbol "1" is used, which is the one 3OT would have used. "Going downward" following the known border is encoded as "0", according to Rule 1. "Going straight" is encoded as "2", as it is the symbol 3OT would have assigned to the direction where the known border is, i.e. downward. In other words, according to this rule, if the direction of the known border is not straight, the 3OT code for its direction and to signal to go straight are swapped.

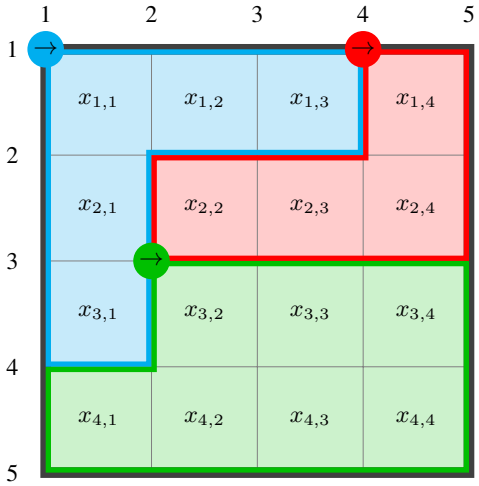
Rule 3 (Not on the border)**.** When P^i_{k+2} is not on the known border, i.e. when $[P^i_k P^i_{k+1}] \notin \overline{\Gamma}$, S-3OT uses the classical 3OT code, then $S^i_{k+2} = S^{3OT}_{k+2}$.

Rule 4 (Follow until the end)**.** Lastly, if for any $k \in [1, m-3]$ it happens that $[P^i_j]_{j=k}^m \in \overline{\Gamma}$, the symbol "3" is appended to the chain code, and the encoding of r_i ends. This symbol signals to the decoder that from P^i_{k+2} onward it just has to follow the known borders until the starting point is reached again.

After the computation of Σ_i is terminated, either by going through all Γ_i or by the special symbol "3", $\overline{\Gamma}$ is recomputed including also all vertices in Γ_i, r_i is removed from $\overline{R}$, and Σ_i is appended to the end of Σ. S-3OT then proceeds selecting the next region to be encoded until $\overline{R} = \emptyset$.

Figure 5 presents a simple example of application of S-3OT. It can be noted that the sequence produced by S-3OT is considerably shorter than the ones computed by standard chain codes.

The algorithm just presented produces chain codes which are strictly shorter than those produced by classical chain codes—which use exactly one symbol for each vertex. S-3OT does not encode P_1 and P_2, resulting in one less symbol for each region, then it is also




```
F4    = 000322332111 0332221001 000332222101
AF4   = 000220102200 0202002201 000202000221
3OT   = 000220101200 0202002201 000202000221
S-3OT = 00220101200 0020023 3
```

Fig. 5. A 4×4 image segmented into three regions; the active crack-edges and the starting positions are outlined in the color of the region. Below the image are the corresponding chain codes.

able to terminate the code with the symbol "3", gaining possibly many more symbols.

Note that P_1^i will always be in $\overline{\Gamma}$, if r_i is not a completely contained regions, i.e. a region which lies entirely inside another region without being adjacent to any other third region. This allows S-3OT to operate without the need of starting coordinates (Property 1). In the case of a completely contained region, one solution might be to split the containing region into two regions to be merged again while decoding. Also, thanks to the way P_1^i is selected we always know that the last turn was upward and that the first movement will go right, this allow us to avoid encoding P_1^i and P_2^i.

4. EXPERIMENTAL VALIDATION

To objectively assess the quality of chain codes produced by S-3OT, we have performed extensive tests over the 500 images in the BSDS500 dataset [15], which has become the standard for evaluating segmentation algorithms. All images in the dataset have a resolution of 481×321. We have performed three scenarios, varying the number of segmentation regions: we used the SLIC algorithm [16] to produce first 30, then 150 and finally 600 regions for all the 500 images in the dataset. In all scenarios completely contained regions have been removed. Then, we produced the chain codes of the segmentation contours using F4, AF4, 3OT and S-3OT. For our comparative analysis we have run standard chain codes region by region, using the same convention adopted by S-3OT to avoid the need for starting coordinates (i.e. always select the top left most not yet encoded pixel as $\overline{x}$); as already discussed standard chain codes do not exploit the presence of common border between any two segmented regions.

For performance evaluation we calculated the first order entropy of each chain code sequence to get an estimate of the coding rate measured in bit per symbol (bps). In Table 1, the average performance over the 500 images in the dataset over the three scenarios are reported in terms of length of the chain code sequence, bit per symbol estimate and image compression rate expressed in bit per pixels (bpp). In Table 2 the average frequencies of each symbol for each chain code are reported as well; this table confirms that S-3OT complies well with Property 3. In can be noted that, although the added symbol to the 3OT alphabet weighs a little bit on the bps scores of S-3OT, the smaller number of symbols and the higher asymmetry in the symbol frequencies compensate for that, letting the overall number of bits (and corresponding compression rate) be the best with respect to all other techniques with a gain of 31%, 28% and 23%, for the cases with 30, 150 and 600 regions, respectively. This gain is clearly explained with the S-3OT capacity to efficiently encode already known borders, either using symbol "0" or "3".

As a side note, among the classical chain codes, our tests also confirm the better performance of 3OT over F4 and AF4. This results are consistent with those reported in other studies [8, 9].

Table 1. Average results over the BSDS500 dataset

		30 regions	150 regions	600 regions
length	F4, AF4, 3OT	23020,576	32110,896	49478,628
	S-3OT	**16316,310**	**23562,800**	**37468,274**
	gain over 3OT	29,12%	26,62%	24,27%
bps	F4	1,996	1,998	2,000
	AF4	1,550	1,560	1,568
	3OT	1,307	1,303	**1,305**
	S-3OT	**1,259**	**1,280**	1,330
	gain over 3OT	3,66%	1,73%	-1,85%
bpp	F4	0,298	0,416	0,641
	AF4	0,231	0,325	0,502
	3OT	0,195	0,271	0,418
	S-3OT	**0,134**	**0,196**	**0,323**
	gain over 3OT	31,48%	27,79%	22,86%

Table 2. Average symbol frequencies over the BSDS500 dataset

	0	1	2	3
F4	0,2553	0,2447	0,2553	0,2447
AF4	0,4182	0,2778	0,3040	-
3OT	0,4182	0,5051	0,0767	-
S-3OT	0,5806	0,3523	0,0555	0,0116

5. CONCLUSIONS

We proposed a framework to encode image segmentation contours using a chain code able to exploits the characteristics of the domain. The proposed approach produces strictly shorter sequences than classical chain codes and, although it requires one extra symbol, we demonstrated how it is able to outperform the other chain codes thanks to its highly skewed symbol frequencies and its shorter sequence length. We tested our approach on over 1500 images, proving a bit per pixel gain of over 27% compared with classical 3OT. Future work might be oriented in finding a proper context-based entropy coding to further compress S-3OT symbol sequence.

6. REFERENCES

[1] M. Pawan Kumar and Daphne Koller, "Efficiently selecting regions for scene understanding," in *Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 3217–3224, IEEE.

[2] Yong Jae Lee and Kristen Grauman, "Object-graphs for context-aware visual category discovery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 346–358, 2012.

[3] Wei Hu, Gene Cheung, Antonio Ortega, and Oscar C. Au, "Multiresolution Graph Fourier Transform for Compression of Piecewise Smooth Images," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 419–433, Jan. 2015.

[4] Giulia Fracastoro, Francesco Verdoja, Marco Grangetto, and Enrico Magli, "Superpixel-driven Graph Transform for Image Compression," in *2015 IEEE International Conference on Image Processing (ICIP)*, Quebec City, Canada, Sept. 2015, pp. 2631–2635, IEEE.

[5] I. Daribo, D. Florencio, and G. Cheung, "Arbitrarily Shaped Motion Prediction for Depth Video Compression Using Arithmetic Edge Coding," *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4696–4708, Nov. 2014.

[6] Daniel Weinland, Remi Ronfard, and Edmond Boyer, "A Survey of Vision-based Methods for Action Representation, Segmentation and Recognition," *Comput. Vis. Image Underst.*, vol. 115, no. 2, pp. 224–241, Feb. 2011.

[7] Martin D. Levine, *Vision in man and machine*, McGraw-Hill series in electrical engineering. McGraw-Hill, New York, 1985.

[8] Hermilo Snchez-Cruz, Ernesto Bribiesca, and Ramn M. Rodrguez-Dagnino, "Efficiency of chain codes to represent binary objects," *Pattern Recognition*, vol. 40, no. 6, pp. 1660–1674, June 2007.

[9] Ionut Schiopu and Ioan Tabus, "Lossless contour compression using chain-code representations and context tree coding," in *Workshop on Information Theoretic Methods in Science and Engineering (WITMSE), Tokyo, Japan*, 2013, pp. 6–13.

[10] *JBIG, ISO/IEC 11544 (ITU-T T.82)*, 2001.

[11] Yong Kui Liu and Borut alik, "An efficient chain code with Huffman coding," *Pattern Recognition*, vol. 38, no. 4, pp. 553–557, Apr. 2005.

[12] Ionut Schiopu and Ioan Tabus, "Anchor points coding for depth map compression," in *2014 IEEE International Conference on Image Processing (ICIP)*, Paris, France, Oct. 2014, pp. 5626–5630.

[13] Herbert Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 2, pp. 260–268, June 1961.

[14] Hermilo Snchez-Cruz and Ramn M. Rodrguez-Dagnino, "Compressing bilevel images by means of a three-bit chain code," *Optical Engineering*, vol. 44, no. 9, pp. 097004, Sept. 2005.

[15] Pablo Arbelaez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik, "Contour Detection and Hierarchical Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[16] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.