Suitability of chain codes for near-lossless cartoon image compression

Aljaž Jeromel*
Supervised by: Borut Žalik†

University of Maribor
Faculty of Electrical Engineering and Computer Science
Maribor / Slovenia

Abstract

The suitability of different chain codes for compression of cartoon images is considered in this paper. Cartoon images differ importantly from other raster images, as they are produced by a human and not captured by a camera. They are characterised by large regions of pixels with the same colour, and, consequently, the number of visually distinct colours is very low. Surprisingly, only a few methods have been proposed for compressing cartoon images.

In this paper, we estimate the suitability of known chain codes for region representation, including Freeman Chain Code in Four and Eight Directions (F4, F8), Vertex Chain Code (VCC), Three-Orthogonal Chain Code (3OT), and a new variation of the VCC, Modified Vertex Chain Code (M_VCC). An evaluation was done of the compression efficiency.

Keywords: Cartoon images, Image compression, Chain codes, String transformations

1 Introduction

Data compression is one of the oldest topics in Computer Science. By removing redundancies in data, less data are needed to be maintained. However, different kinds of data (e.g. text, images, video) have different properties, and, because of that, each can be compressed the most efficiently only by specialised data compressors. Furthermore, different specimens of the same kind of data may differ substantially, and may not be suitable for compression using the same approach. For example, according to Salomon and Motta [1], images are divided into five categories: Bi-level images, grayscale images, continuous tone images, which are typically obtained from a camera, discrete tone images, which are usually created by a computer and contain distinct borders between regions of different colours, and cartoon images, which are similar to the discrete tone images, but differ from them by even more evident borders between the different colours and a very low amount of colours used. An example of a cartoon image is shown in Figure 1.

Figure 1: Cartoon image

In a recent article [2], an efficient method for compressing the cartoon images was proposed. The method takes advantage of the low number of visually distinct regions in a cartoon image by segmenting it and encoding the border of each region using the chain code. This paper considers the efficiency of representing the regions' borders in regard to the different chain codes, including Freeman Chain Code in Four and Eight Directions (F4, F8), Vertex Chain Code (VCC), Three-Orthogonal Chain Code (3OT), and a new variation of the VCC, Modified Vertex Chain Code, M_VCC.

This paper is structured as follows. The related work in cartoon image compression is presented in Section 2. In Section 3, the most frequently used chain codes are presented, together with the new variation of VCC. The results are given in Section 4, and the conclusions are presented in Section 5.

2 Related work

Despite that Image Compression is a very well-investigated field, most of the research has been done on continuous tone images, as they are the most common of the aforementioned five types of images. Only a few techniques specialised in cartoon images have been developed, and only two of them use chain codes for compression.

The research in this field began in 2006, when Tsai et al. developed a cartoon image compression method utilising the quad-tree [3], where the colour palette is limited to 256 colours. Thus, dithering is applied to the image if it

*aljaz.jeromel@um.si
†borut.zalik@um.si

contains more colours, which allows its compression efficiency to be slightly higher. After determining the colour palette, the image is divided into blocks of 32×32 pixels, which are then represented using the quad-tree. The output file is compressed additionally with the LZW algorithm [1].

In 2009, Zhe-Lin et al. proposed another method for compressing cartoon images, named RS-LZ, that employed simple segmentation to extract regions of uniform colour [4]. Such regions were described using F8. The leftover pixels are encoded using their RGB colour values, and the resulting data stream is processed with a ZIP encoder [1]. The main disadvantage of this method is the combination of F8 chain code and the process of determining which pixels belong to the solid region. For a pixel to be assigned to a region, it has to be of the same colour as all of its eight neighbours, resulting in some pixels being classified wrongly as leftovers and encoded separately. This, unfortunately, lowers the compression efficiency.

An important contribution was made by Taylor in 2011 [5]. He proposed the breadth-first search using a colour tolerance to identify the regions of uniform colour, while simultaneously removing some of the possible noise. The noise could be the consequence of edge smoothing, or previous lossy compression with, for example, JPEG. In the next step, regions containing a very low amount of pixels are merged with larger regions, to reduce noise further and improve the efficiency of the compression. Regions are then encoded using the position of the first pixel encountered in the raster scan order, the width and height of the region, its colour, and the pixel bit mask. The latter is encoded with Run-Length Encoding (RLE). At the end, the output stream is encoded additionally with Huffman coding [1].

The latest developed method, which forms the framework for this work, is the Chain Code Cartoon Compression algorithm (4C) [2]. The algorithm consists of four steps. In the first step, the image is partitioned into regions of uniform colour using the breadth-first search, similar to Taylor's suggestion [5]. Too small regions are merged with their most similar neighbours. In the third step, the chain codes of all regions are determined, and concatenated to improve the compression efficiency [6]. In the final step, the chain codes are transformed using the string transformation algorithms, the Burrows-Wheeler Transform (BWT) [7], Move-To-Front transform (MTF) [1], and Run-Length Encoding (RLE) [1], and written to the output stream, along with the image metadata. Optionally, the output stream can be compressed additionally by a binary arithmetic coder, e.g. PAQ8L [8]. Because of the colour tolerance in the first step, and the region merging in the second, the 4C algorithm is lossy. However, by setting the parameters in the aforementioned steps adequately, the difference between the source and the compressed image can hardly be distinguished. An example is shown in Figure 2.

The deformity of the images was tested using the Structural Similarity Index (SSIM) [9] and Peak signal to noise ratio [1]. The Table containing the SSIM and PSNR results for the test images is available in the article presenting the 4C algorithm [2].

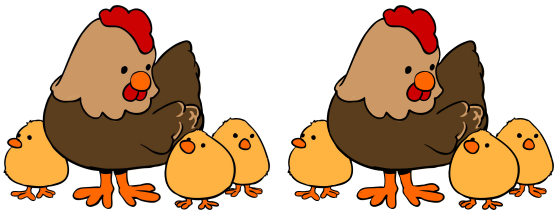


Figure 2: Original cartoon image (left) and image compressed with the 4C algorithm (right)

3 Chain codes

The most popular chain codes are presented in this Section. The idea of using simple instructions for representing the boundary of the rasterized object is very old, and was proposed by Freeman in 1961 [10]. He proposed two chain codes to represent the boundary pixels, the eight-directional F8 and the four-directional F4. Since then, some other chain codes have been developed. In 1999, Bribiesca proposed the Vertex Chain Code, VCC, which consists of only three symbols [11]. Another chain code popular in data compression is the 3OT chain code, proposed in 2005 by Sánchez-Cruz et. al [12], which also contains only three symbols.

Though very efficient at representing the contour of an object, most chain codes allow pixels of the same object to be connected in four directions only. It is more efficient, however, to allow pixels to be connected in eight directions, which is allowed by, for example, the F8 chain code. Since more data than just chain code are needed to reconstruct a region, having a lower number of regions results in better compression. Therefore, a chain code which can represent diagonally connected components should be more efficient at compressing cartoon images. The only popular chain code that is able to represent the diagonally connected components is F8, but its downside is the number of symbols, i.e. F8 consists of eight different symbols, which means that each symbol is encoded with three bits. Thus, a new chain code, based on VCC, is proposed, which can also represent the diagonally connected components. The chain codes compared in this paper are described in more detail in the following Subsections.

3.1 Freeman Chain Code in Four Directions

The first considered chain code is F4. It consists of four symbols, which represent the direction of movement which are shown in Figure 3.

With this chain code, the contour of an object is tracked by moving from pixel to pixel, or by tracking the edges

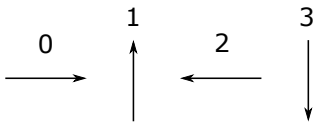


Figure 3: Symbols of the F4 chain code

of border pixels. An example of encoding a shape by the F4 chain code, while moving in the clockwise direction, is shown in Figure 4. Starting from the topmost pixel (the starting vertex marked in red), the code is 0303303222112101.

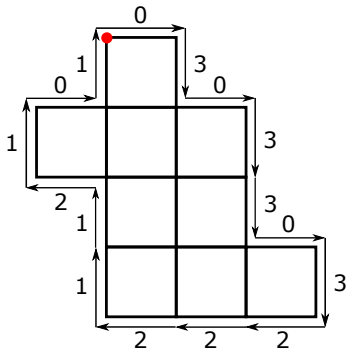


Figure 4: An example of encoding a shape using the F4 chain code

3.2 Freeman Chain Code in Eight Directions

Tracking the contour of the object using the F8 chain code is done by moving along the bordering pixels in contrast to moving along pixels' edges. The symbols of the F8 chain code are shown in Figure 5.

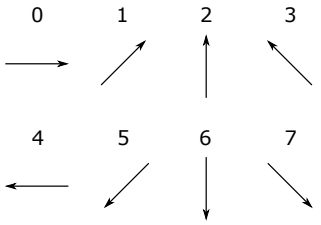


Figure 5: Symbols of the F8 chain code

An example of tracking the contour of an object with the F8 chain code while moving clockwise is shown in Figure 6. Starting from the topmost pixel (marked in red), the code is 76744231.

3.3 Vertex Chain Code

The Vertex Chain Code, VCC, was designed by Bribiesca in 1999 [11]. To determine the Vertex Chain Code representation of a shape, the algorithm tracks its border by moving from one of the pixel's corners (vertex) to the next.

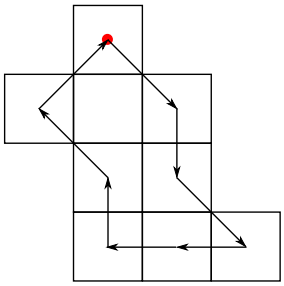


Figure 6: An example of encoding a shape using the F8 chain code

The chain code contains only three symbols, which represent the number of neighbouring pixels belonging to the shape, as shown in Figure 7.

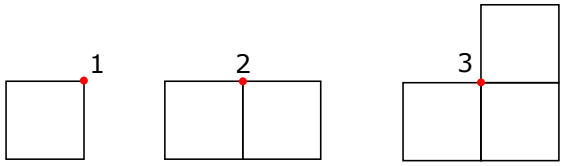


Figure 7: Symbols of the VCC

Because of the low number of different symbols, VCC is suitable for data compression. An example of encoding the contour of an object with VCC is shown in Figure 8. Reading the chain code clockwise from the topmost pixel (the starting vertex is marked in red) yields the code 11312311221231 13.

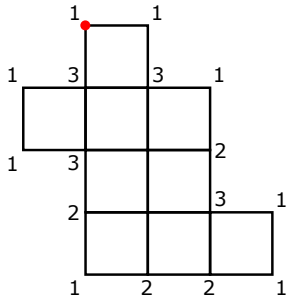


Figure 8: An example of encoding an object with VCC

3.4 Three-Orthogonal Chain Code

Designed in 2005 by Sánchez-Cruz et. al, the 3OT chain code is one of the newest of the popular chain codes [12]. The same as with VCC, it consists of only three symbols, which makes it very suitable for compression. To encode a shape with the 3OT chain code, the contour is tracked by the edges of the bordering pixels. The track of two directions must be kept; the current and the previous direction. The symbol 0 is output when there is no change to the current direction. The symbol 1 is generated, when the current direction changes and becomes equal to the previous

direction, and the symbol 2 is issued when the current direction changes and becomes the opposite of the previous direction. The symbols of the 3OT chain code are shown in Figure 9, and an example is shown in Figure 10. The encoding is done in the clockwise direction, and the 3OT code starting in the topmost pixel is 1211011200201121.

For better clarification on how the 3OT chain code works, certain pixel edges are marked in Figure 10. The encoding begins at the vertex, marked with the red dot. As the encoding is started at the top left vertex, the previous direction can be set to 'up'. Tracking the contour in the clockwise direction, we move right using edge A, and set the current direction to 'right'. The edge A cannot be encoded yet, as only two directions are known. Next, the edge B is tracked in the downward direction. The edge can be encoded as both the current and the previous direction are set. As the direction over edge B is different from the current direction, it is then compared to the previous direction. As they are opposite, the edge B is encoded with symbol '2', the previous direction is set to 'right', and the current direction is set to 'down'. Next, the edge C is encoded. The direction over C is 'right', which is not equal to the current direction ('down'). However, it is equal to the previous direction, so C is encoded using symbol '1', the previous direction is set to 'down', and the current direction is set to 'right'. The next edge is D, and the direction over it is 'down', which is equal to the previous direction, so again, the symbol '1' is output, the previous direction is set to 'right', and the current direction is set to 'down'. Next, the edge E is encoded. The direction over it is 'down', which is the same as the current direction. Therefore, the symbol '0' is output. However, in this case, the current and the previous directions are not updated. The rest of the object is then encoded following the same procedure. In the end, the code for the edge A is inserted at the beginning of the chain code.

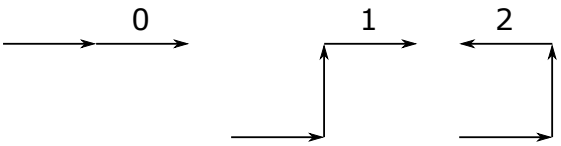


Figure 9: Symbols of the 3OT chain code

3.5 Modified Vertex Chain Code

When compressing cartoon images by means of segmentation, it cannot be guaranteed that pixels of the same colour are not connected diagonally. Objects where bordering pixels are connected diagonally cannot be represented by most of the traditional chain codes without breaking them into multiple parts. To represent such an object with, for example, VCC, the object has to be divided into components that have pixels connected in four directions only. This, however, means more metadata need to be encoded. Therefore, it is desirable to represent such shapes with a single chain.

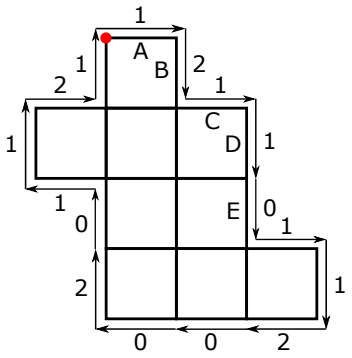


Figure 10: Sample encoding of an object with the 3OT chain code

The only chain code that could represent such a shape without breaking it into two or more parts, is F8. Therefore, a new chain code based on the Vertex Chain Code is proposed in the paper. The difference between the new chain code, Modified Vertex Chain Code (M_VCC), and the original VCC is the behaviour when diagonally connected pixels are encountered. In that case, the symbol '3' is output instead of '1'. An example of this is shown in Figure 11. An example of a different number of regions to encode when using M_VCC and VCC is shown in Figure 12.

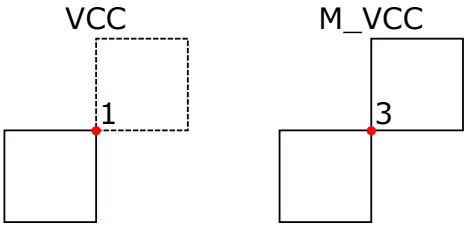


Figure 11: The difference between VCC and M_VCC: When encoding the vertex between two diagonally connected pixels, symbol '3' is output instead of symbol '1'

Encoding diagonally connected parts of the region with a single chain code allows us to reduce the size of the compressed file, as less starting coordinates need to be encoded to reconstruct the chain codes. An example of the

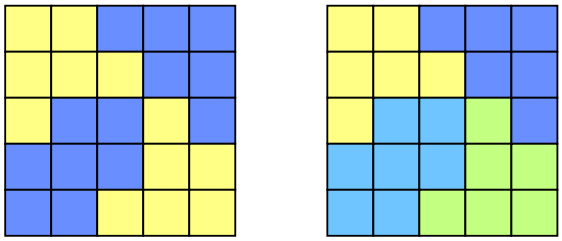


Figure 12: The difference between the number of regions when M_VCC and VCC are used.

difference in encoding a diagonally connected object with VCC and M_VCC is shown in Figure 13, where the encoding is done in the clockwise direction. The obtained chain code strings are 121121 and 12121212 for VCC, and 12112321212123 for M_VCC. The starting vertices are marked with red and blue dots.

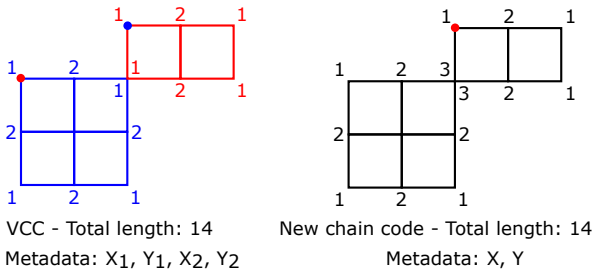


Figure 13: Comparison of VCC and M_VCC representation of a diagonally connected object

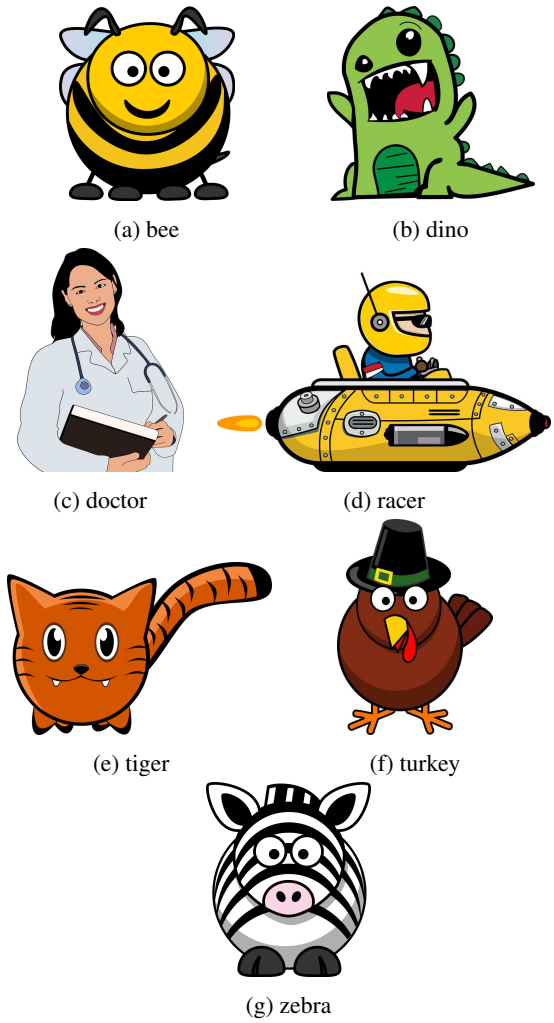
4 Results

The results of encoding the test images with the 4C algorithm [2] using chain codes F4, F8, VCC, 3OT, and M_VCC, are presented in this Section. The images used are the benchmark images for the 4C algorithm [2], which are obtained from the Creative Commons image search, and are free of copyright. They are shown in Figure 14. The compression efficiencies of the tested chain codes are given in Table 1, where the best result for each image is marked in bold. The values are in bits per pixel (bpp) [1]. The compression ratios (ratio between the size of the compressed file and the uncompressed file) [1] are given in Table 2.

Table 1: Compression efficiencies for different chain codes

Image	F4	F8	VCC	3OT	M_VCC
bee	0.7296	0.0515	0.6957	0.7331	**0.0402**
dino	0.8619	0.0460	0.8383	0.8622	**0.0404**
doctor	3.7795	0.1633	3.7271	3.7685	**0.1584**
racer	6.5949	0.1537	6.5236	6.5769	**0.1488**
tiger	0.6983	0.0470	0.6636	0.6972	**0.0410**
turkey	1.0355	0.0709	0.9928	1.0291	**0.0615**
zebra	0.8406	0.0602	0.7934	0.8426	**0.0505**
Average	2.0286	0.0811	1.9855	2.0247	**0.0737**

As seen, the chain codes capable of representing the diagonal pixel connectivity, F8 and M_VCC, outperform the other chain codes by a significant margin. The reason for that is their ability to represent multiple diagonally connected regions as one. The metadata (coordinates, colour) for only one of those regions needs to be, because of that, encoded and stored in the compressed file. This lowers the total compressed file size significantly. Between them,



(a) bee (b) dino (c) doctor (d) racer (e) tiger (f) turkey (g) zebra

Figure 14: Test images

Table 2: Compression ratios for different chain codes

Image	F4	F8	VCC	3OT	M_VCC
bee	0.0304	0.0021	0.0290	0.0305	**0.0017**
dino	0.0359	0.0019	0.0349	0.0359	**0.0017**
doctor	0.1573	0.0068	0.1551	0.1568	**0.0066**
racer	0.2748	0.0064	0.2718	0.2740	**0.0062**
tiger	0.0297	0.0020	0.0276	0.0290	**0.0017**
turkey	0.1294	0.0089	0.1241	0.1286	**0.0077**
zebra	0.0350	0.0025	0.0330	0.0351	**0.0021**
Average	0.0989	0.0044	0.0965	0.0986	**0.0040**

M_VCC proved to be more efficient in every test case, due to its lower number of different symbols. Out of the less efficient chain codes, the Vertex Chain Code, proved to be somewhat more efficient at encoding cartoon images than the 3OT and the F4 chain codes.

5 Conclusions

In this paper, the efficiency of different chain codes was compared for compression of cartoon images with the 4C algorithm. The chain codes considered are F4, F8, VCC and 3OT, as well as a new variation of the Vertex Chain Code, M_VCC. Images were encoded most efficiently by using the new chain code M_VCC, followed tightly by the F8 chain code. The reason for that lies in the ability to represent diagonally connected objects as one, as it reduces the amount of metadata that need to be encoded in the compressed file.

References

[1] David Salomon and Giovanni Motta. *Data Compression: The Complete Reference, 5th edition*. Springer, New York, 2010.

[2] Aljaž Jeromel and Borut Žalik. An efficient lossy cartoon image compression method. *Multimedia Tools and Applications*, 79:433–451, 2020.

[3] Yi-Chen Tsai, Ming-Sui Lee, Meiyin Shen, and C.-C. Jay Kuo. A quad-tree decomposition approach to cartoon image compression. In *IEEE Workshop on Multimedia Signal Processing*, pages 456–460. IEEE, 2006.

[4] Li Zhe-Lin, Xia Qin-Xiang, Jiang Li-Jun, and Wang Shi-Zi. Full color cartoon image lossless compression based on region segment. In *2009 World Congress on Computer Science and Information Engineering*, pages 545–548. IEEE, 2009.

[5] Ty Taylor. Compression of cartoon images. Master's thesis, Case Western Reserve University, 2011.

[6] Hiram H. López Valdes, Hermilo Sánchez-Cruz, and Magdalena C. Mascorro-Pantoja. Single chains to represent groups of objects. *Digital Signal Processing*, 51:73–81, 2016.

[7] Donald Adjeroh, Tim Bell, and Amar Mukherjee. *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*. Springer, New York, 2008.

[8] Matt Mahoney. Data compression programs. Web page. Available at: http://mattmahoney.net/dc/.

[9] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

[10] Herbert Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, 10:260–268, 1961.

[11] Ernesto Bribiesca. A new chain code. *Pattern recognition*, 32:235–251, 1999.

[12] Hermilo Sánchez-Cruz and Ramon M. Rodríguez-Dagnino. Compressing bi-level images by means of a 3-bit chain code. *SPIE Opticl Engineering*, 44:1–8, 2005.