Multimedia Tools and Applications (2020) 79:433–451
https://doi.org/10.1007/s11042-019-08126-7

An efficient lossy cartoon image compression method

Aljaž Jeromel[1] · Borut Žalik[1]

Received: 14 September 2018 / Revised: 18 July 2019 / Accepted: 13 August 2019 /
Published online: 29 August 2019

Abstract
This paper introduces a new lossy approach for compression of cartoon images. The image is firstly partitioned into regions of roughly the same colour. The chain codes are then determined of all regions. The sequence of the obtained chain code symbols is transformed with the Burrows-Wheeler Transform, Move-To-Front transform, and compressed with Run-Length Encoding. In the final step, an arithmetic encoder may be used to compress the obtained binary stream additionally. The proposed algorithm is asymmetric, meaning that the decompression does not reverse all the steps of the compression procedure. The experimental results have shown that the described method produces considerably better compression ratios than JPEG, JPEG2000, WebP, SPIHT, PNG, and two of the algorithms specialised in compression of cartoon images: the algorithm using quad-tree, and RS-LZ algorithm.

Keywords Cartoon images · Image compression · Chain codes · String transformations

1 Introduction

Image compression is a widely researched area with a huge amount of developed methods. According to [15], there are five major image categories: Monochromatic, greyscale, continuous-tone, discrete-tone, and cartoon images. Numerous methods intended for compressing the images from the first four categories have been proposed, some of them being very successful. On the other hand, although an extensive search through the available literature was done, no really efficient method has yet been developed for the compression of the cartoon images. Indeed, only a few cartoon image compression techniques have been reported until now. In 2006, Tsai et al. presented a quasi-lossless method for compressing cartoon images using quad-trees [20]. The method applies dithering if the image contains more than 256 colours. The method does not perform well when the image contains a lot of

This work was supported by the Slovenian Research Agency under Grants J2-8176 and P2-0041.

✉ Aljaž Jeromel
aljaz.jeromel@um.si

1 Faculty of Electrical Engineering and Computer Science, Koroška cesta 46, SI-2000 Maribor, Slovenia

small regions. Another lossless compression method, named RS-LZ, was developed by Li et al. [9]. The method uses the Freeman chain code in eight directions (F8) to represent the borders of the solid regions, and encodes pixels that are contained neither on the border nor inside of the solid region. Because of that, images containing a lot of small solid regions and images, where edge smoothing or JPEG-like compression has been applied before, are not compressed well. A promising research was done by Taylor in 2011 [18]. He introduced a lossy compression algorithm, which takes into account small colour differences between neighbouring pixels, and a quantization procedure for small details/noise. The algorithm owns a good compression ratio at the expense of losing some image information.

A new lossy method for cartoon image compression, named Chain Code Cartoon Compression (4C), is proposed in this paper. The main novelty of this work is fulfilling the gaps in cartoon image compression by developing a new approach, which would outperform the state-of-the-art general-purpose and domain-specific algorithms. The proposed method firstly divides the image into free-form regions having similar colours. The shapes of the regions are described with chain codes, which are then transformed with Burrows-Wheeler and Move-to-Front transforms. The obtained stream of chain code symbols is then encoded with Run-Length Encoding and arithmetic coding. As the algorithm is asymmetric, the decompression is faster than the compression. For cartoon images, the proposed method produces better compression ratios than the referenced algorithms. Images compressed with the proposed method are also more visually pleasing than those compressed with JPEG, JPEG2000, WebP, or SPIHT, with approximately the same structural similarity index (SSIM). In most cases, the same SSIM of two images means that they are of the same quality. PSNR is also considered of decompressed images obtained with the mentioned algorithms.

This paper contains 4 sections and an Appendix. The proposed compression algorithm and its steps are presented in Section 2. The third Section contains the results of the experiments, while conclusions are given in Section 4. The Appendix shows images used in the experiments.

2 The algorithm

The presented algorithm works in eight steps, shown in Fig. 1. The first three steps of the algorithm are lossy, and the other five are lossless. The losses are controlled by three user-defined parameters: $ctol$, msz, and $ptol$. Two of these parameters, $ctol$ and $ptol$, are thresholds for joining colours during the region detection and palette calculation, respectively. The third parameter, msz, defines the minimum size for the region to be allowed to stay independent after the region reduction step. Each region is described by chain codes in the fourth step. The chain codes of all regions are concatenated, and Burrows-Wheeler Transform (BWT), Move-to-Front Transform (MTF), and Run-Length Encoding (RLE) are applied to the resulting chain. In the final step, the arithmetic encoding is used. These steps are explained in detail in the following subsections.

2.1 Region detection

In the first step of the algorithm, the image is partitioned into connected regions of similar colour, based on the approach presented in [18]. The algorithm traverses the image in the raster scan order. Whenever a pixel not yet belonging to any region is encountered, the breadth-first traversal using the pixel's eight neighbourhood is performed from the starting

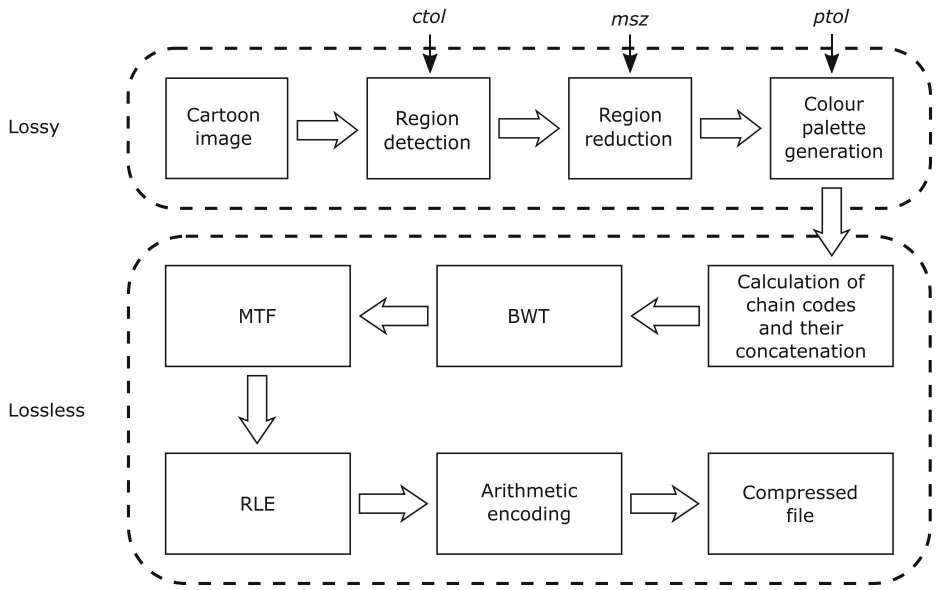


Fig. 1 Steps of the compression algorithm

pixel. Pixels whose colour differs from the region's average colour for less than the threshold $ctol$ are included into the considered region. The threshold for colour similarity is a user-defined parameter, by which the compression ratio and the quality of the compressed image are controlled. When deciding whether or not to add a considered pixel to the region, an additional test needs to be done. If the pixel is discovered by its diagonal neighbour, the other two pixels in their 2×2 sub-grid are checked as to whether their colours differ less than $ctol$. If they do, the considered pixel is rejected. This test is necessary to prevent the region interlacing, which causes the wrong reconstruction of the image. It should be noted that this test does not prevent region interlacing completely - however, it reduces it considerably. Region interlacing is explained in more detail in Section 2.9. When a considered pixel is accepted into a region, its RGB colour values modify the accumulated colour values of the region.

2.2 Region reduction

In this step, the regions smaller than the user-defined threshold msz are merged with neighbouring regions. The algorithm iterates through the list of regions and, when too small region is encountered, its neighbouring regions are determined. When the neighbouring regions are found, the most suitable of them is merged with the current region. Firstly, the region's luma colour component of YCbCr colour space is calculated. If the calculated value is lower than 100 (the value was determined experimentally), the region is considered as a part of the contour (the black border between the coloured regions). In this case, it is merged with the neighbouring region that has the most similar colour. Otherwise, the colours of the region's neighbours, larger than msz, are compared to the region's colour, and the largest region with the most similar colour is chosen for merging. However, if the region has no neighbours larger than msz, it is merged to the neighbouring region most similar in colour.

2.3 Generating palette

As a rule, cartoon images contain a small number of distinctly different colours. Thus, a colour palette is useful to make the data less redundant. Furthermore, because of colour tolerance in the first step and region merging in the second, some regions might get a slightly different colour, despite having the same colour originally. Because of that, a user-defined colour tolerance parameter $ptol$ is used to unify similar colours. In that way, the palette contains a small number of colours without visually impacting the image. The algorithm iterates through all regions and checks whether a similar enough colour has already been accepted. In that case, the colours are merged, and their average is weighted by the number of involved regions. Otherwise, the colour is considered as a new colour.

2.4 Determining chain codes

Determining the chain codes of the regions is the first step of the lossless part of the proposed algorithm. The Vertex Chain Code (VCC) [4] is used in our implementation, but any chain code could be used here [6, 16, 21]. An example of applying the Vertex Chain Code is given in Fig. 2. Vertex Chain Code moves along vertices that connect 4 neighbouring cells and encodes the number of pixels belonging to the region. Because it moves along the borders of the region, the only valid symbols in the output are 1, 2, and 3. There is, however, no need to establish the spatial relations between the regions, i. e. determining the holes in the regions. Namely, the holes in the case of raster images do not represent the hollow parts, but another region, filled with a specific colour.

The example of processing an image with the proposed algorithm up to this step is presented in Fig. 3, where $ctol = 50$ and $msz = 5$. Figure 3a shows the original image. In Fig. 3b, the first pixel in the raster scan order is used, and the breadth-first traversal is started from it to find all of the similar connected pixels. The result of filling the first region is given in Fig. 3c, where the next starting pixel for the breadth-first traversal is also marked. Figure 3d shows the image after the region detection step has been performed. The yellow pixel was absorbed by the orange region, because the difference in colour was less than $ctol$, and the orange region's average colour was modified accordingly. The result of the

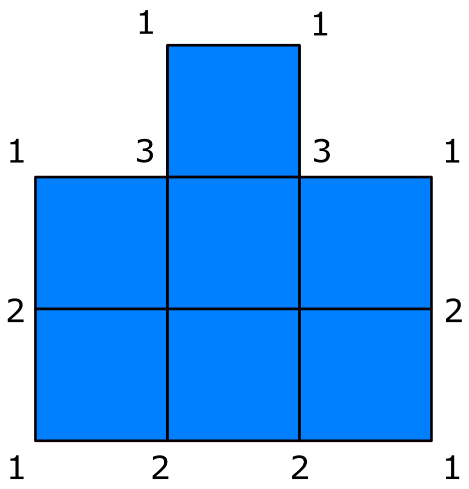


Fig. 2 Example of the Vertex Chain Code (VCC)

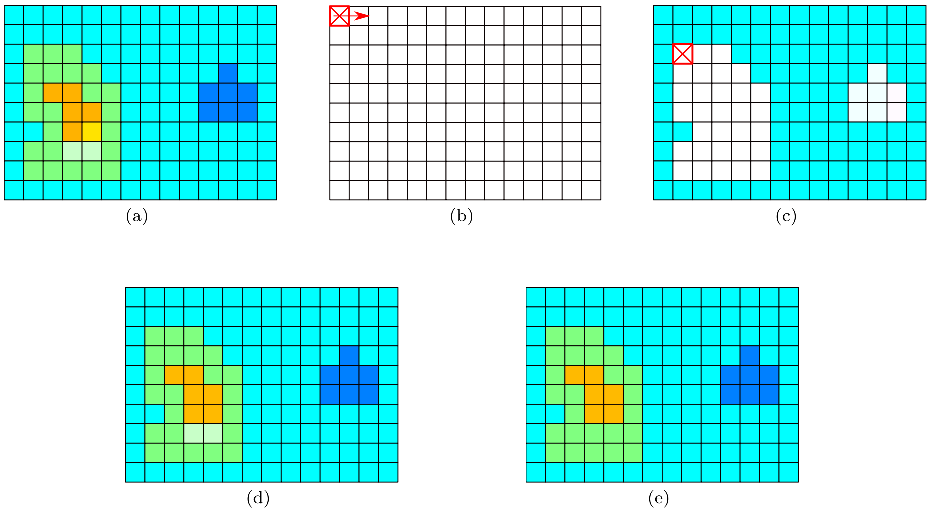


Fig. 3 Example of the lossy part of the proposed algorithm at work

region reduction step is shown in Fig. 3e. Because the light green region contained less than 5 pixels, it was merged with the surrounding green region.

The chain codes are then concatenated into one chain, which is more compressible than many short ones [10].

2.5 Chain code transformations

To achieve better compression, string transformation techniques are applied on the concatenated chain code symbols [22]. First, Burrows-Wheeler transform (BWT) [1] is used, followed by the Move-to-Front transform [5] and Run-Length Encoding (RLE). The BWT shuffles the string in a way that similar symbols are placed together, forming long runs of the same symbol. The MTF then transforms these runs into runs of zeros and RLE compresses these runs efficiently. The runs of 0 symbols are encoded as follows:

- Runs of 0-symbols shorter than threshold t_1 are encoded with the unary code [15].
- Runs of 0-symbols longer than t_1 and shorter than $2^{b_1} + t_1$ are binary encoded using b_1 bits.
- Runs of 0-symbols longer than $2^{b_1} + t_1$ and shorter than $2^{b_2} + t_1$ are encoded in binary code with b_2 bits.
- Runs of 0-symbols whose lengths are longer or equal to $2^{b_2} + t_1$ are split into runs that can be processed using the options above.

The values t_1, b_1, and b_2 are calculated by the programme on the stream of symbols produced by MTF. The programme tests each combination of values for t_1, b_1 and b_2, where the following relations apply:

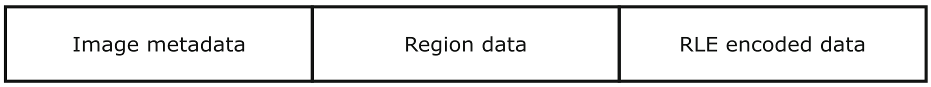
- $0 \leq t_1, b_1 < 8$,
- $1 \leq b_2 \leq 16$,
- $b_1 \leq b_2$,
- $t_1 \leq 2^{b_1}$.

The algorithm then chooses the values that produce the lowest number of bits on the transformed chain code symbols.

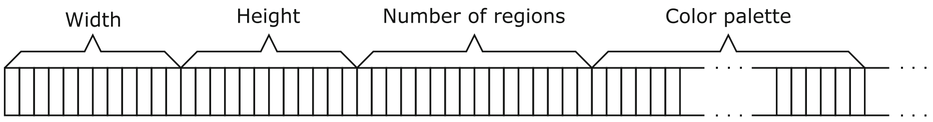
2.6 The storage format

To be able to reconstruct the image, certain parameters need to be written in the compressed file in addition to the region metadata that includes the coordinates, colours, and transformed chain code symbols. The structure of the compressed file is presented in Fig. 4. A more detailed description of the file structure is presented below:

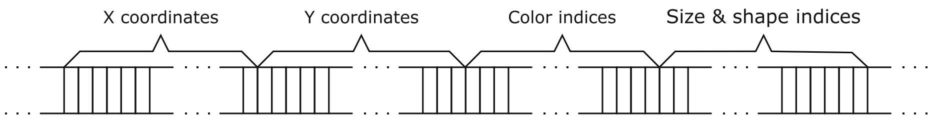
- 12 bits for the width of the image,
- 12 bits for the height of the image,
- 16 bits for the number of regions,
- 4 bits for the number of bits needed for the number of colours n_c,
- n_c bits for the number of colours,
- 24 bits for each colour's RGB components,
- l_x bits for the X coordinate of each region's starting pixel (explained in the continuation),
- l_y bits for the Y coordinate of each region's starting pixel (explained in the continuation),
- n_c bits for each region's colour index,



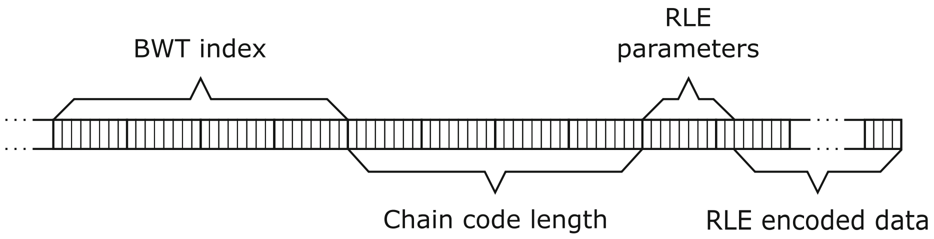
(a) Basic file structure



(b) Structure of the image metadata



(c) Detailed structure of region data



(d) Chain code data structure

Fig. 4 Structure of the compressed file

- variable length for size and shape indices of each region (explained in the continuation),
- 32 bits for BWT index,
- 32 bits for the length of the concatenated transformed chain code symbols,
- 3 bits for RLE threshold t_1,
- 3 bits for number of bits b_1,
- 4 bits for number of bits b_2,
- RLE encoded chain code symbols.

The bit length, l_x, for encoding x-coordinates, is determined using (1), where W is the width of the image.

$$l_x = \begin{cases} \left\lceil \log_2 \sqrt{W} \right\rceil, & \text{if } \lceil \log_2 x_i \rceil < \left\lceil \log_2 \sqrt{W} \right\rceil \\ \lceil \log_2 W \rceil, & \text{otherwise} \end{cases} \tag{1}$$

The bit length l_y for encoding y-coordinates is determined the same way from the height of the image H. The x_i and y_i values are obtained by subtracting the previous region's starting coordinates from the starting coordinates of the current region. The negative differences are mapped to positive numbers by adding the width of the image. Because the regions are sorted by their y-coordinate, the differences between y-coordinates are always positive. The differences between x-coordinates can be negative, in which case the width of the image is added to the difference. Because the region detection is applied in the raster-scan order, the first region always starts at the coordinates (0, 0), so its coordinates do not have to be stored.

The size and the shape indices of regions are an optional part of the compressed file, which is denoted using one bit. If the bit is 0, this part is skipped. Otherwise, for each region, one of the following codes is emitted:

- 0: the region is described by chain code symbols.
- 10: the region contains only one pixel.
- 11: the region contains two pixels. A 2-bit code representing shape of the region is emitted, in accordance with Fig. 5.

2.7 Arithmetic encoding

The file, organised as described in the previous subsection, is compressed by the binary arithmetic coder in the final step. The arithmetic coder is a compression procedure, where, instead of assigning a specific code to each character, the entire file is encoded into one number [15]. The arithmetic coder starts with an interval, which is then narrowed iteratively, depending on the observed character. In comparison to other well-known lossless data compression methods (i.e. Huffman coding), arithmetic coding compresses data more efficiently at the expense of the processing time. An extensive explanation of arithmetic coding is given in, for example, [2, 7, 14].

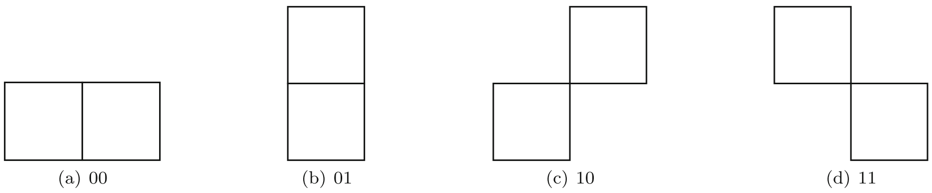


Fig. 5 Possible shapes of the region containing 2 pixels and their bit codes

The binary arithmetic encoder PAQ8L [11] was used in our case. The PAQ family of the encoders uses context mixing to determine the probabilities of zeros and ones. The PAQ7 and the newer versions use the artificial neural networks to achieve better compression, but, unfortunately, they require more processing time.

2.8 Decompression

During the decompression, lossless compression steps need to be applied in reversed order. Firstly, if the file was additionally compressed with an arithmetic encoder, the arithmetic decoding is applied. Then, the metadata from the file header is obtained. The rest of the file represents the RLE-encoded concatenated chain code symbols. The inverse operations of RLE, MTF and BWT are applied to reconstruct the concatenated codes of all regions. The algorithm iterates through all regions, reconstructs their contour, and fills the regions by the flood fill algorithm [19]. During the contour reconstruction, the pixels next to the chain code segments are marked as locked, so as not to be added mistakenly to another region. The decompression is much faster than compression, because it does not have to reverse the operations in the lossy part of the compression algorithm.

2.9 Region interlacing

During the region detection step, additional conditions are checked when performing the breadth-first traversal. In this subsection, the explanation is given why these conditions are necessary. Figure 6 shows the effects of region interlacing during compression and reconstruction of the image if no steps are taken to prevent it.

In Fig. 6a, the blue region has already been detected and saved, while the yellow region is currently being filled with pixels using the breadth-first traversal. Pixels that have not yet been checked, are white. The current pixel (the one where the red arrow starts) checks its bottom-left neighbour (where the arrow points to). Their 2×2 sub-grid is marked with the red border. If it is not checked whether the other pixels in this sub-grid differ less than $ctol$, the pixel would be added to the yellow region to get the situation in Fig. 6b after filling it. After that, the rest of the compression steps are performed upon the image.

During the reconstruction of the compressed image, the blue region is filled first, because it comes first in the raster scan order. The obtained situation is shown in Fig. 6c, where the blue region's border is marked with the violet colour and the locked pixels are crossed. Once a pixel has been locked, it cannot be modified any more. Figure 6d represents the situation after the yellow region has been reconstructed. Its border and locked pixels are marked with orange colour. The pixels marked with red dots have been assigned wrongfully to the yellow region, because they were not locked.

The solution, that would completely remove such scenarios, would be to check all the 2×2 sub-grids in the image and divide the problematic regions into multiple parts. But

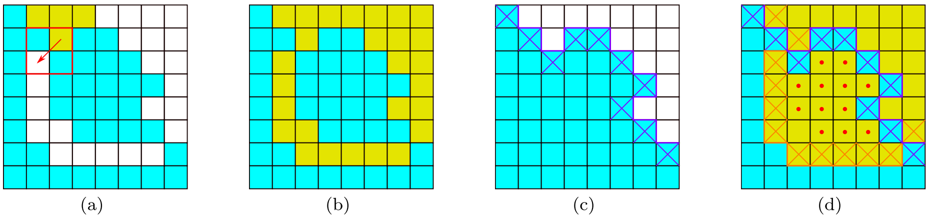


Fig. 6 Preventing wrong reconstruction of interlaced regions

such approach would increase the processing time of the compression significantly. Instead, the steps described in Section 2.1 are taken to reduce the frequency of region interlacing, but not completely remove it. When deciding whether or not to add a diagonally connected pixel to the current region, the 2×2 sub-grid is checked (Fig. 6a), and if the other two pixels (the blue pixels in Fig. 6a) differ less than $ctol$, the pixel is rejected.

3 Results

The results of the proposed 4C algorithm on a set of representative images, shown in Appendix A, are presented in this Section. The features of the algorithm are compared with the PNG [3], RS-LZ [9], WebP [12], SPIHT [13], the Quad-tree approach [20], as well as with the image compression standards JPEG [23] and JPEG2000 [17]. The compression efficiency and the time needed for compression and decompression were considered. However, as the referenced algorithms have very different characteristics, special care was given to the fair comparison of compression results:

– Comparison with lossless methods was done by applying only the lossy parts of 4C algorithm to images and saving them. The images were then loaded and compressed with the considered lossless methods. In this way, the efficiency of the lossless approaches was compared only to the lossless parts of the 4C algorithm. Some lossless methods could outperform the lossy algorithms in this way, because the redundancy, existing in the images, was reduced by the lossy parts of 4C algorithm.
– Comparison with lossy methods was done according to SSIM (Structural Similarity Index) [24], a metric for measuring the visual quality of images. The images were compressed firstly with 4C algorithm, and their SSIM was measured. The compression parameters for other lossy methods were then determined in a way that, after the compression, their SSIM values were as close as possible to SSIM obtained with the proposed 4C algorithm (see Table 4).

The parameters used to compress each of the testing images with the proposed 4C algorithm are given in Table 1. Different parameter values have been tested, and the ones that produced the most visually pleasing reconstruction and good compression ratio were chosen. The parameters depend on the thickness of the black borders and the amount of edge smoothing applied to the image. The following hints are proposed for setting the parameters:

– If the image has thin borders between regions, $msz \leq 10$.
– If the image has very smooth/blurred black edges, then $ctol \geq 50$.
– If the neighbouring colours do not differ considerably, $ctol \leq 25$ and $ptol \leq 25$.

For most cartoon images, $ctol = 50$, $msz = 20$, and $ptol = 100$ reproduce the original image very well.

As seen, the parameter values in the case of the image ”Ice cream” are much lower than the parameter values used for other images. The referenced image contains very thin black borders, and some very small important regions (see Appendix A). The compensation for that is the lowering of the msz parameter. The $ctol$ and $ptol$ parameters were lowered as well, not to treat different hues of the girl’s hair as one region, or merge the colours during the determination of the colour palette.

Table 1 Parameters, used to compress each of the test images with 4C algorithm

Image	Width [pixels]	Height [pixels]	$ctol$	$ptol$	msz
Bee	722	800	50	100	50
Chickens	2000	1562	45	50	3
Dino	768	720	50	100	50
Doctor	538	720	60	100	5
Ice cream	960	613	5	5	3
Racer	960	576	35	100	3
Seal	1412	1920	40	100	30
Shark	1920	1426	30	100	50
Teacher	1920	1371	60	100	50
Tiger	958	626	60	100	50
Turkey	518	720	35	100	50
Zebra	673	800	60	100	40

3.1 Processing time of the algorithm

The time, needed for compressing an image relies on its dimensions, as well as the user-selectable parameters. Larger colour tolerance generates a lower number of small regions and, consequently, requires less processing time. Colour tolerance for the palette has the same effect as colour tolerance for region detection. The larger value produces less colours and shorter processing time. On the other hand, higher msz value results in more iterations of the merging procedure, and requires more processing time. The CPU time needed for compression and decompression of each image is given in Table 2. The measurements were done on a personal computer with Intel Core i5-3570K CPU 3.40 GHz, 32 GB of RAM, running the 64 bit operating system Windows 10 Education.

As seen, image Ice cream requires significantly more processing time than other images (Bee, Racer, Tiger, Zebra) of similar size, due to the considerably lower values for $ctol$, msz, and $ptol$.

Table 2 Average compression and decompression times

Image	Compression time [s]	Decompression time [s]
Bee	0.5282	0.1147
Chickens	3.4604	0.8814
Dino	0.4712	0.1091
Doctor	0.4750	0.0708
Ice cream	1.7838	0.1739
Racer	0.6957	0.1257
Seal	2.3170	0.6179
Shark	2.5910	0.4715
Teacher	2.1034	0.5213
Tiger	0.5133	0.1097
Turkey	0.3238	0.0711
Zebra	0.4999	0.1024
Average	1.3136	0.2808

3.2 Compression efficiency

The parameters also affect the compression ratio. In general, larger values produce better compression ratio, since they result in less regions and colours. The compression efficiency for each image from Appendix A is shown in Table 3, where the best results are marked in bold. The quality of the images compressed with standards JPEG and JPEG2000 and algorithms WebP and SPIHT was chosen according to the SSIM. The SSIM values used for each image are given in Table 4, their PSNR [15] values in Table 5, and the parameters, controlling losses of those methods, are in Table 6.

As seen, the proposed 4C algorithm outperforms the lossless methods significantly on transformed images. The transformation, applied by the lossy steps of the proposed algorithm, makes the image more cartoon-like, which allows the lossless algorithms to achieve better compression than if they were applied on the original images. Out of the referenced algorithms, the Quad-tree approach produced the best results. The only image the Quad-tree method did not compress well was the image Ice cream, where very low values were used for $ctol$, msz and $ptol$. Because of that, the quad-tree contained a lot more nodes, which considerably worsened the compression.

The second most efficient algorithm out of the compared lossless methods was PNG, followed by RS-LZ. The RS-LZ's performance was worse than the performance of other lossless algorithms because of its definition of solid colour regions. It assumes the solid colour regions are made up of pixels that have the same colour as all eight of their neighbours. It then encodes the contour of the solid regions with chain codes. The pixels, not assigned to any solid region or its border that way, are encoded directly by storing RGB codes. Obviously, the problem for this algorithm are thin regions, where no pixel would be recognised as a member of the solid colour region.

The proposed 4C algorithm also produces considerably better compression than the compared lossy methods with practically the same SSIM. Among all of the compared methods, the standard JPEG produced the worst compression, while WebP was the most efficient general-purpose compression algorithm; it was even better than the special-purpose RS-LZ. In general, the lossless algorithms were more efficient than the lossy ones, because the lossless methods were run on transformed images, as explained at the beginning of this Section, which improved their compressibility substantially.

However, two quality metrics (see Tables 4 and 5) disagree substantially, and, therefore, it is a question of which is more reliable. Because PSNR is inversely proportional to MSE (Mean Squared Error), it perceives images with lower pixel difference as higher quality. Therefore, the manner in which the image information is lost is very important. The images compressed with JPEG, JPEG2000, WebP, and SPIHT show signs of blurring, which produces low MSE (and higher PSNR), while the proposed 4C algorithm introduces sharpening to the edges and reduces the number of colours, which result in higher MSE and lower PSNR. As discussed in [8], though the PSNR and the quality of reconstruction are correlated, this metric can be used reliably only when comparing the results of the same method.

Despite the high SSIM used in comparisons of the proposed method and the other lossy methods, the difference in reconstructed images can still be, in some cases, detected visually. In Fig. 7, the detail of Seal image is shown. The image contains relatively large areas of uniform or close to uniform colours, but the colour of the neighbouring areas differs a lot. Therefore, sharpening the edges of these areas is preferable in comparison to blurring. The effects of losses produced by the WebP, JPEG, JPEG2000 fall into the latter category, which visually degrades the cartoon image considerably, while the proposed algorithm sharpens

Table 3 Comparison of the compression efficiency for the test images (in bits per pixel)

Image	4C	4C+PAQ8L	Quad-tree	RS-LZ	PNG	JPEG	JPEG2000	WebP	SPIHT
Bee	0.0357	**0.0323**	0.1886	0.6035	0.4297	2.0369	1.4148	0.6912	1.1500
Chickens	0.0321	**0.0273**	0.0951	0.2477	0.2303	0.6822	0.4508	0.308	0.4350
Dino	0.0345	**0.0314**	0.1605	0.4638	0.3867	0.9868	0.7630	0.3931	0.6900
Doctor	0.1006	**0.0915**	0.2240	0.5079	0.4645	0.6932	0.9673	0.1906	0.5380
Ice cream	0.5258	**0.4515**	1.7977	1.0970	0.8370	1.1942	1.0022	0.4545	0.8030
Racer	0.1247	**0.1086**	0.2654	0.6534	0.4437	1.2754	0.9326	0.6828	0.8390
Seal	0.0191	**0.0169**	0.0825	0.2446	0.2529	0.1688	0.0886	0.0548	0.1235
Shark	0.0265	**0.0221**	0.1004	0.3068	0.2901	0.2702	0.2628	0.0818	0.1565
Teacher	0.0162	**0.0139**	0.0570	0.1730	0.1962	0.1551	0.0758	0.0607	0.1051
Tiger	0.0355	**0.0319**	0.1582	0.4510	0.3455	0.9557	0.5290	0.5554	0.5020
Turkey	0.0493	**0.0316**	0.2261	0.6320	0.5315	1.5843	1.1293	1.2234	0.9820
Zebra	0.0421	**0.0377**	0.1551	0.7989	0.5431	0.8966	1.4241	0.2315	0.6235
Average	0.0868	**0.0747**	0.2926	0.5150	0.4126	0.9083	0.7534	0.4107	0.5790

Table 4 SSIM values used for the comparison of images obtained with the proposed 4C method, standards JPEG and JPEG2000, and algorithms WebP and SPIHT

Image	4C	JPEG	JPEG2000	WebP	SPIHT
Bee	98.22%	98.33%	98.44%	98.22%	98.22%
Chickens	98.96%	98.84%	98.93%	98.97%	98.96%
Dino	97.13%	97.22%	97.22%	97.24%	97.13%
Doctor	97.26%	97.25%	97.08%	97.35%	97.26%
Ice cream	97.11%	97.15%	97.10%	97.11%	97.11%
Racer	98.08%	97.97%	98.10%	98.10%	98.08%
Seal	94.36%	94.89%	94.06%	95.17%	94.36%
Shark	97.42%	97.44%	97.65%	97.40%	97.42%
Teacher	93.98%	95.06%	93.94%	93.87%	93.98%
Tiger	97.15%	97.26%	97.30%	97.16%	97.15%
Turkey	98.44%	98.24%	98.42%	98.16%	98.44%
Zebra	96.41%	96.06%	96.56%	96.44%	96.41%
Average	97.04%	97.14%	97.07%	97.10%	97.04%

the edges. After decompression, the edge smoothing can be applied to the image to reduce the visibility of the sharpening in our case.

4 Conclusion

In this paper, a new method, named 4C (Chain Code Cartoon Compression), is presented for the lossy compression of cartoon images. It segments the image into free-form regions,

Table 5 PSNR values obtained with the proposed 4C method, standards JPEG and JPEG2000, and algorithms WebP and SPIHT

Image	4C	JPEG	JPEG2000	WebP	SPIHT
Bee	23.5553	43.3804	38.0609	38.4387	41.2454
Chickens	29.4900	40.6153	38.1208	37.1469	40.0882
Dino	25.0759	35.4936	33.0263	34.2003	35.6077
Doctor	23.0701	30.4367	30.7728	26.4377	32.4420
Ice cream	23.2215	32.2480	31.4631	29.8874	32.1415
Racer	24.8657	35.6742	32.8087	33.3063	35.9188
Seal	26.5575	29.6611	26.5015	28.3179	27.7666
Shark	30.8470	34.1215	35.0217	32.7214	35.1677
Teacher	26.3893	31.4729	27.8507	32.3562	31.7085
Tiger	23.9720	36.9331	31.7760	36.5994	36.1051
Turkey	25.1329	39.9792	36.8295	34.9539	38.5100
Zebra	21.7932	32.7884	31.7221	29.6004	33.9646
Average	25.3309	35.2337	32.8295	32.8305	35.0555

Table 6 Quality parameters used to compress test images with JPEG, JPEG2000, WebP, and SPIHT

	JPEG	JPEG2000	WebP		SPIHT
Image	Quality (0-100)	Quality (0-100)	Quality (0-100)	m	BPP
Bee	96	43	91	6	1.1500
Chickens	90	43	90	6	0.4350
Dino	83	38	77	6	0.6900
Doctor	65	35	10	6	0.5380
Ice cream	80	37	62	6	0.8030
Racer	87	38	90	6	0.8390
Seal	20	30	1	6	0.1235
Shark	47	40	29	6	0.1565
Teacher	19	33	15	6	0.1051
Tiger	85	38	91	6	0.5020
Turkey	93	42	100	6	0.9820
Zebra	70	36	20	6	0.6235

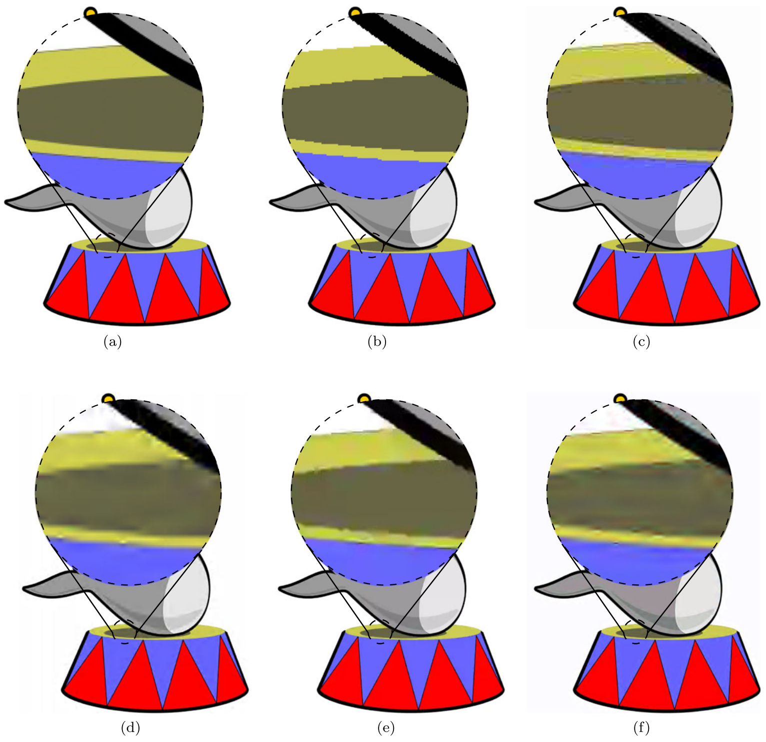


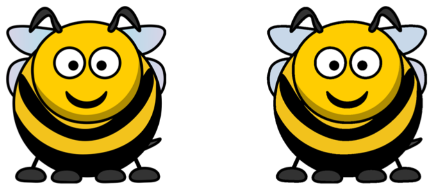
Fig. 7 Detail of the original image (**a**), reconstructed image compressed with the proposed algorithm (**b**), with the standard JPEG (**c**), with the standard JPEG2000 (**d**), with the algorithm WebP (**e**), and with the SPIHT algorithm (**f**)

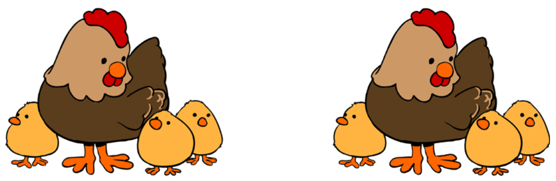
which are described with chain codes. The chain code symbols are then transformed using Burrows-Wheeler and Move-to-Front transforms and compressed with Run-Length Encoding, which may also be followed by an arithmetic encoder. Based on the experiments, the presented method outperforms JPEG, JPEG2000, WebP, and SPIHT in terms of compression ratio and visual degradation of the image. The lossless compression steps of the proposed algorithm also produce better results on the transformed images than the Quad-tree method, the PNG algorithm, and the RS-LZ algorithm.

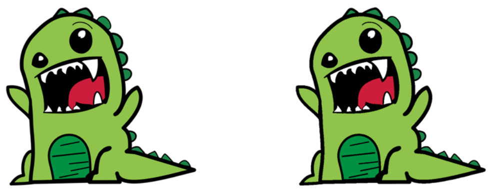
The parameters $ctol$, msz, and $ptol$, controlling the amount of losses, are determined experimentally at this stage. This aspect of the algorithm opens the new challenges to further research of how to determine the parameters automatically by analysing the images first. In addition, it would be reasonable to test the 4C method to compress I-frames on cartoon videos or animated movies compression.

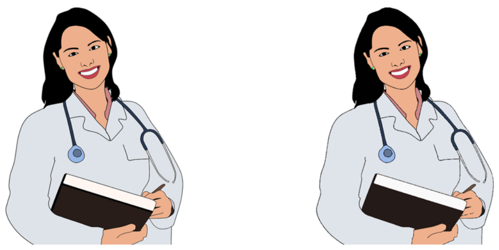
Appendix A

The original and the reconstructed cartoon images are given in this Appendix. The original image is on the left, while the reconstructed is on the right.

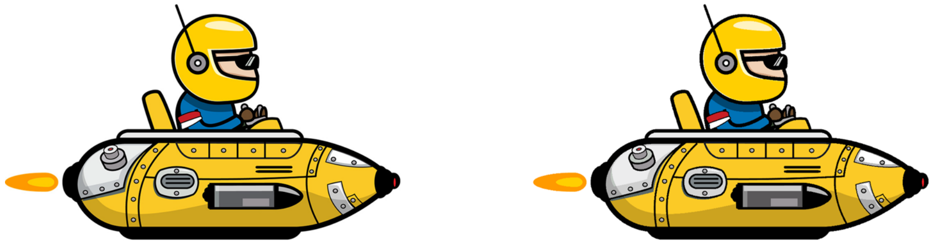

Bee

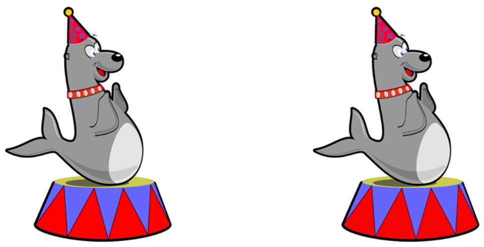

Chickens

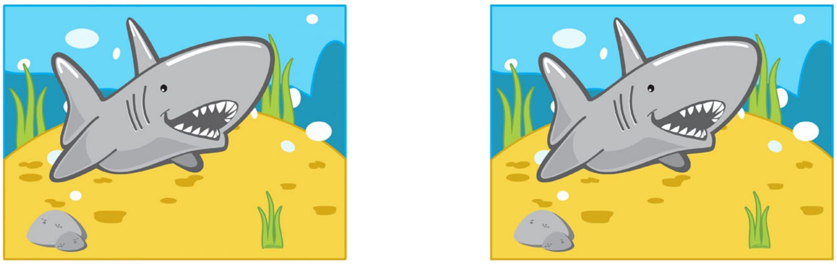

Dino

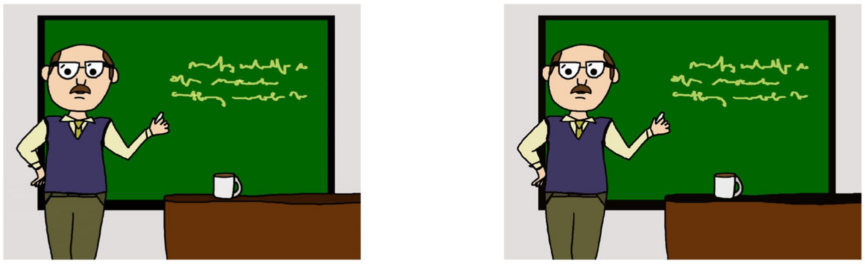

Doctor

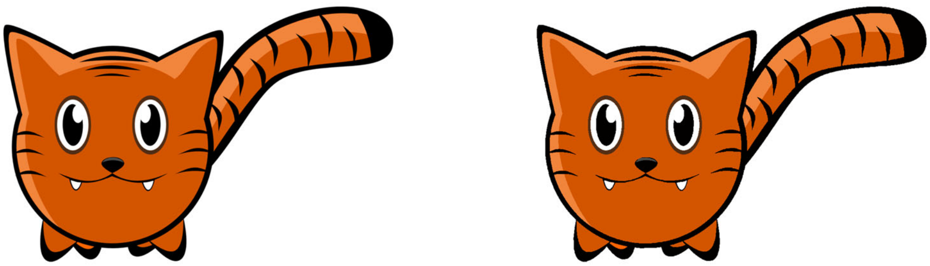
Ice Cream

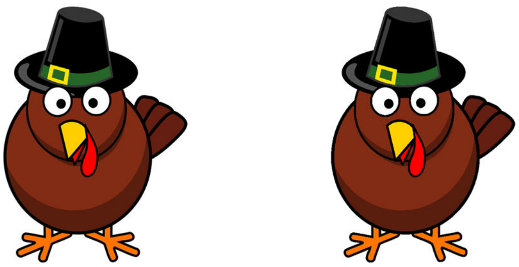

Racer


Seal


Shark


Teacher


Tiger


Turkey

Zebra

References

1. Adjeroh D, Bell T, Mukherjee A (2008) The Burrows-Wheeler transform: Data compression, suffix arrays and pattern matching. Springer, New York
2. Bodden E, Clasen M, Kneis J (2007) Arithmetic Coding revealed: A guided tour from theory to praxis Technical report. Sable Research Group, Montreal
3. Boutell T, Adler M, Bowler J et al (2003) Portable Network Graphics (PNG) Specification (Second Edition), W3C Recommendation. https://www.w3.org/TR/2003/REC-PNG-20031110/. Accessed: 15 June 2018
4. Bribiesca E (1999) A new chain code. Pattern Recogn 32:235–251
5. Elias P (1987) Interval and recency rank source coding: Two On-Line adaptive Variable-Length schemes. IEEE Trans Inf Theory 33:3–10
6. Freeman H (1961) On the encoding of arbitrary geometric configurations. IRE Trans Electron Comput EC10:260–268
7. Howard PG, Vitter JS (1992) Practical implementations of arithmetic coding. In: Storer JA (ed) Image and text compression. Kluwer Academic Publishers, Norwell, pp 85–112
8. Huynh-Thu Q, Ghanbari M (2008) Scope of validity of PSNR in image/video quality assessment. Electron Lett 44(13):800–801
9. Li ZL, Xia QX, Jiang LJ, Wang SZ (2009) Full Color Cartoon Image Lossless Compression based on Region Segment. In: 2009 World Congress on Computer Science and Information Engineering. IEEE, pp 545–548
10. López-Valdes HH, Sánchez-Cruz H, Mascorro-Pantoja MC (2016) Single chains to represent groups of objects. Digit Signal Process 51:73–81
11. Mahoney M Data Compression Programs. Web page, available at: http://mattmahoney.net/dc/
12. Rabbat R (2010) WebP, a new image format for the Web. Chromium Blog, web page, available at: https://blog.chromium.org/2010/09/webp-new-image-format-for-web.html Accessed: 10 June 2019
13. Said A, Pearlman WA (1996) A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans Circ Syst Video Technol 6(3):243–250
14. Said A (2002) Introduction to arithmetic coding - theory and practice. In: Sayood K (ed) Lossless compression handbook. Academic Press - Elsevier Inc., Cambridge Massachusetts, pp 101–152
15. Salomon D, Motta G (2010) Data compression: The complete reference, 5th edn. Springer, New York
16. Sánchez-Cruz H, Rodríguez-Dagnino M (2005) Compressing bi-level images by means of a 3-bit chain code. SPIE Opt Eng 44(9):1–8
17. Skodras A, Christopoulos C, Ebrahimi T (2001) The JPEG 2000 still image compression standard. IEEE Signal Proc Mag 18(5):36–58
18. Taylor T (2011) Compression of cartoon images. Masters thesis, Case Western Reserve University
19. Torbert S (2016) Applied computer science. Springer, New York, p 2
20. Tsai YC, Lee MS, Shen M, Kuo CCJ (2006) A Quad-Tree Decomposition Approach to Cartoon Image Compression. In: IEEE Workshop on Multimedia Signal Processing. IEEE, pp 456–460
21. Žalik B, Mongus D, Liu YK, Lukač N (2016) Unsigned Manhattan chain code. J Vis Commun Image Represent 38:186–194

22. Žalik B, Mongus D, Rizman Žalik K, Lukač N (2016) Chain code compression using string transformation techniques. Digit Signal Process 53:1–10
23. Wallace GK (1992) The JPEG still picture compression standard. IEEE Trans Consum Electron 38(1):xviii–xxxiv
24. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612

Aljaž Jeromel is a Technical Assistant of Computer Science at University of Maribor, Slovenia. He obtained B. Sc. In Computer Science in 2017. His areas of interest include data compression and computer graphics.

Borut Žalik is a Professor of Computer Science at University of Maribor, Slovenia. He obtained B.Sc. in Electrical Engineering in 1985, M.Sc. and Ph.D. in Computer Science in 1989 and 1993, respectively. He is the head of the Laboratory for Geometric Modelling and Multimedia Algorithms at Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests are in processing of multimedia data, data compression, and computational geometry. He published over 100 papers in refereed journals and almost 100 papers on scientific conferences. Currently, he is the Chair of Slovene Chapter of ACM.