Czech Technical University
Center for Machine Perception

RESEARCH REPORT

No. CTU-CMP-1997-K335-1997-CMP-140

Adaptive non-linear Predictor for Lossless Image Compression
Minimal Thesis

Jaroslav Fojtík

fojtik@vision.felk.cvut.cz

November 24, 1997

Acknowledgement:
This research was supported by the Czech Ministry of Education grant VS96049, the Grant Agency of the Czech Republic, grants 102/95/1378, 102/97/0480, 102/97/0855 and European Union grant Copernicus CP941068.

This publication can be obtained via anonymous ftp from
ftp://cmp.felk.cvut.cz/pub/cvl/articles/fojtik/compress.minimum.gz

Czech Technical University, Faculty of Electrical Engineering
Department of Control Engineering, Center for Machine Perception,
121 35 Prague 2, Karlovo náměstí 13, Czech Republic
FAX +42 2 290159, phone +42 2 24357465, http://cmp.felk.cvut.cz

Preface

This work is submitted to fulfil the requirements set upon the finishing of the first period of author's doctoral studies at the Czech Technical University, Prague, Faculty of Electrical Engineering, Department of Control Engineering. This work has been carried out under the supervision of Doc. Ing. Václav Hlaváč, CSc. and Doc. Ing. Bohuslav Kirchmann, CSc.

Abstract

Selection of the data compression depends on a compression ratio, which an user wish to achieve. If the compression ratio should increase, a data model must more better approximate the data structure.

This work aim at clearly defining the image compression task and tries to find limitation of tested models. It also comments related approaches and topics in the literature and proposes the new method for lossless image compression.

The new method lossless image compression method performs very well and results achieved can overtop other methods that we are aware of. We developed further the Slessinger's idea to represent an image as residuals of a special local predictor. The predictor configurations in a binary image are grouped into couples that differ in a representative point only. Only residuals that correspond to the less frequent predictor from the couple are stored. An optimal predictor is based on the frequency of predictor configuration in the image. Two main extensions are proposed:

1. The method is generalized for gray-level image or images with even more degrees of freedom.

2. The method that works with adaptive estimator is proposed.

The resulting *FH-Adapt* algorithm performs very well and results can compete with most current algorithms for the same purpose. The predictor can learn automatically from the compressed image and copes even with complicated fine textures. We are able to estimate the influence of the cells in a neighbourhood and use only significant ones for compression.

The plans for the further research aiming towards the PhD. thesis are sketched at the end of this report.

Contents

1 Introduction **4**
- 1.1 Image compression . . . 4
- 1.2 Aim of this report . . . 4
- 1.3 Organization of this document . . . 5

2 Lossless image compression, state of the art **6**
- 2.1 Well known compression methods . . . 6
- 2.2 Main ideas of the coding . . . 8
- 2.3 Binary Schlesinger's predictor . . . 8

3 Our results obtained so far **12**
- 3.1 Our approach . . . 12
- 3.2 Generalization for gray-level images . . . 12
- 3.3 Encoding residuals . . . 13
  - 3.3.1 Code with flexible length . . . 13
  - 3.3.2 Combination of the Huffman code and code with logarithmical growth 14
- 3.4 Adaptive compression . . . 15
- 3.5 Color images with palette function . . . 18
- 3.6 Summarize of advantages and disadvantages . . . 20

4 Experiments **22**
- 4.1 Gray level images . . . 22
- 4.2 Color images with palette function . . . 24
- 4.3 Computational time aspects . . . 25

5 Limitations of previous approach and future work **27**
- 5.1 Texture classification . . . 27
- 5.2 Improvement of the compression performance . . . 27

6 Thesis plan **29**

7 Conclusions **30**

A Disk File format **33**
- A.1 Uncompressed data . . . 34
- A.2 Compressed data . . . 34

1 Introduction

The work with the data needs a storage space. If anybody wish to have more and more data then he either needs a bigger storage space or store data into smaller space. Second possibility is called data compression.

Shannon's information theory showed, that the amount of the information in the data is much smaller than the size of original data. This idea is fundamental principle of data compression.

The amount of real information is typically very small in graphics data and it is a useful task to seek for the better and better data models.

1.1 Image compression

Compression algorithms try to find redundant data and represent them in a more compact way. Each compression algorithm consists of model description and transformed data. The compression is as good as model describes encoded data (how much information is included in the model).

The main difference between compression of strings and images is the more complex context in the case of images. The stronger is the structure of the available context the more radical can the compression be.

1.2 Aim of this report

The minimal thesis closes one phase of the PhD. candidate research at the Czech Technical University. The aim is to:

- Demonstrate that the author has a clear view of his research area and
- has a reasonable formulation of the task potentially leading to the PhD. thesis.

We have chosen the image compression as a thesis topic.

We show the reader the state of the art of the currently used compression algorithms. Consequently, a list of currently used algorithms is included into this report.

Next we present a new data model for compression of images in this document. This model has *two unusual features*. First, the non-linear predictor that was first in its basic form proposed by M. Schlesinger is used. We were introduced to these ideas in the "Theoretical methods of pattern recognition" course read eng. Prof. Schlesinger at the CTU Prague in summer 1996. Generalizations of the Schlesinger's predictor are proposed. Second, the results of compression are well for available set of test images.

We aim to develop further a new image compression model in the PhD. research. This model should reflect better natural structure of raster image data. This implies research of most common attributes of images and use results to modify data model. We aim to test experimentally the developed method.

The thesis plan is outlined in section*6. Thesis plan.* We would appreciate any comments and suggestion to it.

1.3 Organization of this document

Our comments to previous work related to image compression is in section 2. In the section 3. we propose a new algorithm. In the section 4, results of this algorithm are shown. Problems, which should be solved and hints how to use the local predictor for solving another problems are shown in the section 5. The plans of the future work are in the section 6.

2 Lossless image compression, state of the art

The basic principle used in lossless compression of symbols is to code more frequent symbols using shorter code words, cf. Huffman coding [Ani89]. The simplest operation is concatenation of symbols that results in shorter string of symbols. The context of a current symbol in a string has one degree of freedom (1-DOF) as only relation to preceding and following symbols is taken into account.

The context of a binary image has two degrees of freedom (2-DOF), i.e. in vertical and horizontal direction. The gray-level image context has 3-DOF, the 3rd DOF allows to change intensity values. A colour (e.g. RGB) image context has 4-DOF. The fourth direction points across color image planes.

Stronger context exists in the image data. Each dependence among data could decrease amount of information per unit of information (Byte, kByte). Each additional knowledge about data structure allows us to create a better data model. Models used today for an image compression do not fully reflect the two dimensional context of images.

Most of hitherto used algorithms [1] for image compression use only a simple 1-DOF data model. These are for example PCX - only RLC coding, GIF - method LZW see [Mel96] and new PNG both directional differences + LZH. See [JW94] for detailed description of the supported compression methods.

Under development are some newest algorithms. But most of them do not have significantly better model than well known 1-DOF compression algorithms for example ARJ, ZIP, AIN. These are used in future text for comparison with tested 2D models. On other hand there exist very good algorithms based on segmentation of domains and separately encoding see [RA96], but they have an enormous time consumption.

2.1 Well known compression methods

PCX

This approach is only 1-DOF modified RLC (Run Length Coding) method. See [JW94] for the better explanation. Bit planes are stored separately for each row. If there are 8 or 24 bit planes then data are stored in 8bit strips for each row. Each strip is compressed separately. Let we call an actual processed data unit $[Data]$ with a size (Byte). If $[Data]$ repeats n times and ($n > 1$ or $[Data] > 191$) then two values are stored [n+190] and [Data] to the output stream. The singular items $[Data] < 191$ are stored normally.
Advantages : This approach is very simple. Results could be good for line art with a small number of lines.
Disadvantages : Compression performance is very bad and data model has included unclear expectations. (The probability of numbers 0-191 is lesser than this for numbers 192-255.)

[1] In the next section is included more detailed description of algorithms.

LZH

This method uses a floating history window. The encoded alphabet has n symbols. Method adds new symbols for indexing into a history. If a substring starting on an actual position is found in th history, then the actual symbol is replaced by the history index and the size of a substring. All symbols are now encoded using Huffman (or another prefix) code. See also [Mel96].
Advantages :LZW seems to be a high performance algorithm for 1D data and its modifications are used in industry these days (for example Stacker v.4.0 and PkZip).
Disadvantages :This algorithm has only a 1D data model. For images should exist a better data model which reflects better its natural structure.

IBM Q80 coder (Lossless JPEG)

First of all a differentions of the image the function is computed. These differentions are stored in the fixed prefix code. This code was created after analysis of large set of the test images. See [ISO94] for more information.
Advantages : JPEG has implemented and well known only a lossy part of the compression. It was a good idea to allow a lossless one.
Disadvantages : This algorithm is reported to be an obsolete. A static data model used inside is not a good solution.

PNG (Portable Network Graphics)

The plain description is included in [Pet97]. An algorithm consists from two steps. In first step the differention of image function $\Delta_x(x,y) = f(x-1,y) - f(x,y)$ is computed and a matrix which contains only diferentions is created. (The diferentions $\Delta_y(x,y) = f(x,y-1) - f(x,y)$ or Δ_{xy} can be also used in this method). This matrix is encoded using the LZW method in the second step. Both parts may be connected by a pipe and these can run in parallel on Unix-like systems.
Advantages : I expect that this is the today's hopeful method. The source code of the algorithm is freely available and a compression performance is not bad. *Disadvantages* : The second step is not specialized for 2D image data. PNG is based on the expectance that the size of the domains with limited 1 st derivation is smaller than the amount of domains with greater 1 st derivation. But 2D context of images contains more redundancies.

MLP (Multi Level Progressive)

This is an example of the pyramidal compression. See [Fry93] for detailed description. First of all, small disjunctive boxes are constructed. Some characteristic value is computed for each box (for example average, median etc.). All pinnacles of small pyramids are used for further computation. The computation stops when a top single value is reached. Residuals are computed as differences between higher level and actual level in each level . All residuals are then encoded.
Advantages : This method reflect better two dimensional structure of the data.

Disadvantages : Computation of pyramids is very time/space expensive and results are not very good.

2.2 Main ideas of the coding

Ideas used in known compression methods are described in this section. This thoughts were obtained after previous methods were analyzed. These ideas are mentioned only in the simplest form. Each idea can lead to a group of compression algorithms. Some algorithms combine more basic ideas together.

A The neighbourhood elements have approximately same value.

B The probability of characters and/or symbols is not the same.

C Some subpatterns can be repeated more times.

D The derivation of the image function ($df(x, y)/dx$ or $df(x, y)/dy$) is limited.

E There are approximately homogeneous regions in an image.

F Any characteristics is common for some area in the grid.

The statement (A) leads directly to the RLC code. This code is not appropriate for compressing of images, see Table 8. Combination of (B) and (C) leads to the LZH method with a floating window, which is the todays most effective compression method for 1D data.

The statement (D) can be used if there is some physical dependence in a data. Combination of (D) and (B) leads to the IBM Q80 coder (lossless JPEG). Recent good common algorithms are based on (D) followed by combination (B) and (C).

The statement (F) leads to the pyramidal compression approach. The basic concept is described in [HŠ92]. More complicated approaches are mentioned in the [Fry93].

The basic truth is that a design consists of 3 basic steps [Mar82]. An idea at the beginning, an algorithmization and implementation at last. But idea is sometimes simple and creating algorithm or implementation is too difficult. Simple implementation is needed for common use of the new algorithm.

Effective compression algorithm must have all these three items easy to understand and reproduce. All requirements are contradictory and a compromise must be found. We focus how to achieve best compression ratio in the first theoretical step.

2.3 Binary Schlesinger's predictor

Our compression method builds on the representation of a binary image proposed by M. Schlesinger from Ukraine [Sch89]. We shall abbreviate it as *FH-2 -DOF* in the sequel. We guess that this method is almost unknown in the image analysis community. Thus we explain it first. We use slightly adapted notation to be able to describe newly proposed methods in the same formalism.

The *FH-2-DOF* method assumes a binary image f as an input. Its support is $T=\{(x, y) : 0 \leq x \leq M;\ 0 \leq y \leq N\}$. The image is the mapping function $f(x, y) \rightarrow \{0, 1\}$. Without

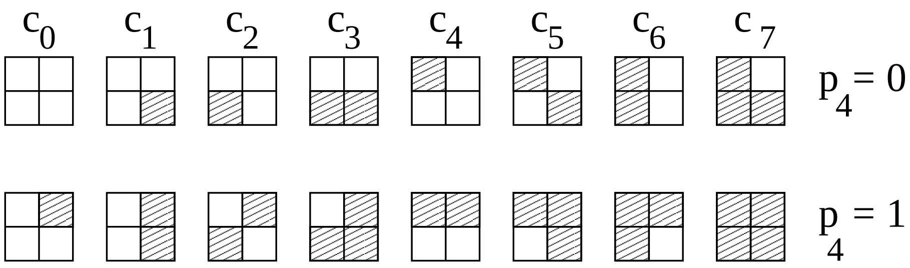

Figure 1: All possible values of the probes. Configurations in the same column differ in the upper right pixel only.

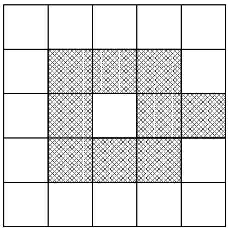

Figure 2: Hypothetical input binary image

loss of a generality, we assume that the left column and the bottom line of the image have value zero[2], i.e. $f(0, y) = f(x, 0) = 0$.

The *FH-2-DOF* method traverses the image f by the probe of 2×2 cells. Its representative cell (x, y) is the upper right corner of the probe. The value of the actual (estimated) cell is $p_4 = f(x, y)$. Let values of the probe be placed in the particular position in the probe be called $p_1 = f(x, y-1)$; $p_2 = f(x-1, y-1)$; $p_3 = f(x-1, y)$; see Fig. 3.

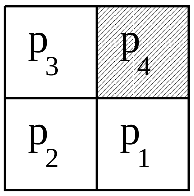

Figure 3: 2×2 testing probe.

Schlesinger designed the predictor (estimator, probe) e in the following way. All 16 possible combinations of cells within a 2×2 probe are considered, see Fig. 1. Probes are arranged into two rows in such a way that probes in one column differ only in the upper right bit.

The method consists of two inverse phases, compression and decompression. For all methods proposed in this paper, the compression phase consists of:

A An optimal predictor design, that is based on the frequency analysis of the probes configurations in the particular image.

B Data reduction that writes predictor residuals into a sparse matrix.

C The reduced data are optimally coded, and together with the predictor description, create the compressed image.

[2]We assume that the leftmost column and bottommost row in the input image is zero for simplicity. It could be nonzero but must be known. Thus some other compression technique should be used for left most column and bottom most row, e.g. RLC coding.

	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
$p_4 = 0$	0	1	1	2	1	0	0	2
$p_4 = 1$	1	2	0	1	2	1	2	0

Table 1: Frequency table for the hypothetical image

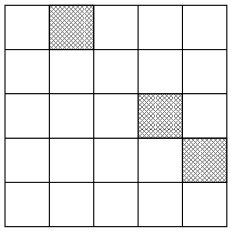

Figure 5: The residuals calculated from Fig. 2

The optimal predictor uses the information about *frequency analysis* of probe configurations in the image. The frequency analysis needs one pass through image and can be performed in parallel. We shall explain algorithms using a small hypothetical input image, see Fig. 2. The frequency of configurations is stored in a table with two rows and eight columns. See Tab. 1, where is an example of the frequency table calculated from the image in Fig. 2.

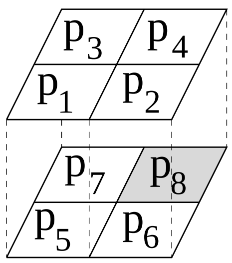

Figure 4: 3-DOF predictor

The predictor $\hat{p}_4 = e(p_1, p_2, p_3)$ estimates the value of the image in the representative point. Let define a penalty function g

$$\begin{aligned} g(\hat{p}_4, p_4) &= 0 \quad \text{for } \hat{p}_4 = p_4 \\ &= 1 \quad \text{for } \hat{p}_4 \neq p_4 \end{aligned} \tag{1}$$

Maxims in each column of the frequency table are found and the corresponding probes are marked. One bit is needed to distinguish between two probes in one column. The stored bit value signs more common probe in each column. One byte is enough to store the information about all eight maxims (8 columns); e.g. 11001110 for the frequency table in Tab. 1. The principal Schlesinger's idea is that only positions of less frequent configurations in the image are remembered.

The construction of an optimal predictor can be formulated as an optimization task that minimizes sum of penalties for all positions of the probe in the image

$$\sum_{x=1}^{M} \sum_{y=1}^{N} g(e(p_1, p_2, p_3), p_4) \to \min \tag{2}$$

The image is traversed in the bottom-up manner and from the right to left[3]. In each probe position in the image, it is checked if more frequent or less frequent configuration appeared.

[3] The manner could be changed, but appropriate side rows (columns) must have special properties.

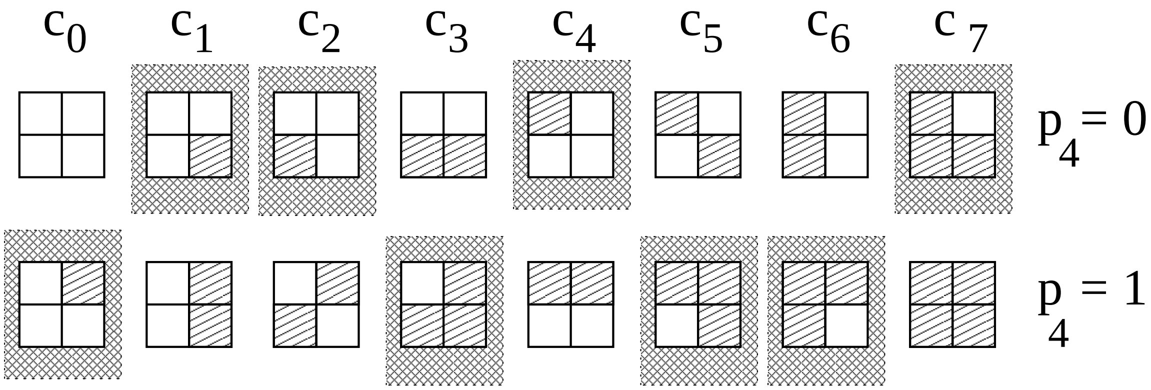

Figure 6: Schlesinger's corner configurations

In the former case, the value 0 is written to the representative cell. In the latter situation, the value 1 is stored that corresponds to the residual of the predictor. The algorithm continues over all support and produces a residual image. In this image is the number of pixels with value 1 decreased. This reduced image is typically as sparse as most of the information was captured by the predictor. The test image Fig. 2 is transformed to Fig. 5. The description of the predictor (one byte) belongs to the compressed image.

The *decompression* algorithm has at hand description of the predictor (one byte) and a reduced image with residuals. The left most column and bottom most row are known.

The probe is put to bottom left position first. Its values p_1, p_2, and p_3 are known. This three values uniquely determine one of the eight columns of the predictor. Which of the couple of probes is determined by the value of the stored residual. The new decompressed value p_4 is set. It can overwrite existing residual on the position p_4.

The probe traverses the image linewise to the right and upward until the whole original image is restored.

Let us note that Schlesinger codes reduced image as list of x, y coordinates of residuals. He even uses suboptimal coding for special arrangement of probes see Fig. 6. In such case, the standard set operation can be performed on reduced images without a need to decompress it.

3 Our results obtained so far

3.1 Our approach

One of sources of redundancy in raster images is the fact that neighbouring cells [4] are likely to have the same intensity (see formula (A) in the list of basic ideas in the section 2). That implies that the local context of the current cell is important. An attempt is made to examine the neighbourhood of the current cell using a small probe. We shall start from a simple probe, the 2×2 window for the case of binary images see Fig. 3.

We shall develop further Schlesinger's idea of predictive coding. The new value in the current cell is estimated using a *predictor* function. The predicted value is encompassed by the probe. In the compression phase, an image is systematically traversed by the probe. The estimated value is compared with the actual value of the cell. Where there is a discrepancy between estimated and the actual value the *residual* is stored on x, y coordinates. It a predictor is good then a precise match will be in very many cells and amount of the information written into into compressed image is small. The description of the predictor, positions of residuals in the image are stored together onto target file.

3.2 Generalization for gray-level images

The gray-level image can be treated as stacked from bit planes see [Dav69] for better explanation. We generalized the predictor to cope with the current bit plane and the closest higher bit plane, see Fig. 4. The context encompassed by the volumetric probe has 3-DOF. We name the algorithm *FH-3-DOF*. This is the abbreviation. Letters F and H are initials of authors surnames Fojtík and Hlaváč.

In the FH-3-DOF predictor, cells from one neighbouring bit-plane were taken. Number of dimensions can be extended to other bit-planes in the RGB components in colour image. Of course, the constraint on the predictor support must be always fulfilled. It says that all probe elements without actual on must belong to the already reconstructed part of the image. Otherwise the causal decompression were not possible. You will probably need non-causal decoding process, which will be very complicated and we cannot design it now.

The definition of the optimization task is similar to the previous case[5] and is valid for each bit plane (The plane above the bottom plane is assumed to be all 0):

$$\sum_{x=1}^{M} \sum_{y=1}^{N} g(e(p_1, p_2, p_3, p_4, p_5, p_6, p_7), p_8) \rightarrow \min \tag{3}$$

The frequency analysis of probe configurations is analogical to the previous case. Here 2^7 bits = 16 bytes are needed to store the frequency table information. The image of the bit-plane is reduced and residuals are stored in the same way as in *FH-2-DOF* case. This requires second pass through original image data. The result is a sparse matrix with small number of residuals.

[4] The concept *cell* is used instead of common pixel for 2D images. The reason is that the cell will be used for higher dimensions.

[5] The predicted cell should have the highest index to avoid problems in further computation.

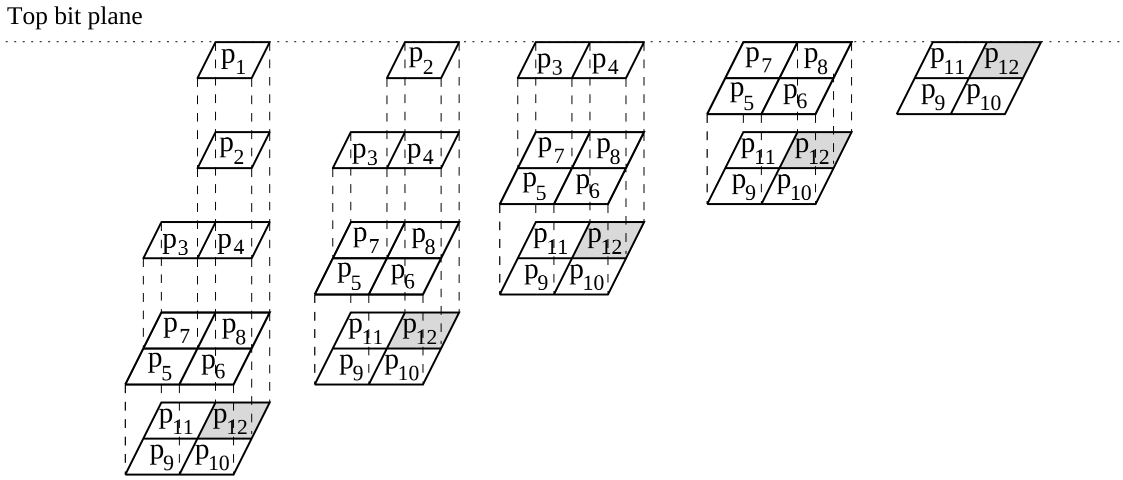

Figure 7: Modifications of probes for encode planes

The development of the shape of the probe, when we are coding several planes, is illustrated on Fig. 7. The right probe is used for processing a top plane. One probe to the left is used for coding the bit plane below the top plane and so on.

3.3 Encoding residuals

To achieve high compression ratio the matrix with residuals should be effectively coded. Consider Fig. 5 where an example of residuals is shown. The intention is to code residuals using line scans and distances on them between residuals. The counting starts from a hypothetical starting point positioned left from the first row. Counting ends in the hypothetical end point positioned right from last row[6]. The distances $l_1, \ldots, l_k$ obtained represent uniquely residuals.

$$\sum_{i=1}^{k} l_i \;=\; M\,N + 1 \tag{4}$$

This equation is always valid and can be used for error checking.

3.3.1 Code with flexible length

The question of effective coding distances $l_1, \ldots, l_k$ is important. The Huffman coding is not suitable because the maximal length $l_{max} = M\,N+1$ is a big number. The time consumption of computing Huffman table is also important. The best algorithms can do it in the time[7] $o(n * log(n))$, where best complexity of sorting items is $o(n * log(n))$ and complexity of computing the Huffman table is $o(n)$. The number $o((M\,N+1) * log(M\,N+1))$ could be very high.

[6] In our implementation counting can also finish in the last point in the last row.

[7] $o()$ is the symbol of the function, which computes complexity and n is the number of elements.

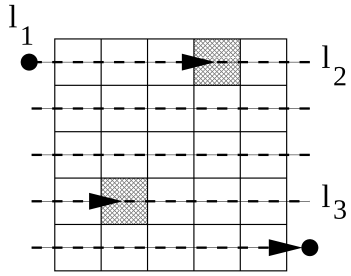

Figure 8: Lengths to be coded.

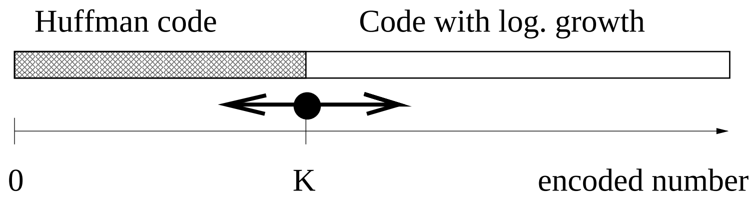

Figure 9: Coding lengths.

Consequently a code for representation of a integer numbers must be used. There are three available types of this code Fibonacci, Elias and " ω " according to [Mel96]. " ω " code is not advisable for our purpose. This code grows very quickly with the increasing of the encoded number. And we need to encode large values very often. Computing of Fibonacci code is not so simple. The Elias code is better for our purpose. This code is composed from two parts, first uses unary and second binary code. Experiments was shown that some first small numbers (1,2,3 etc) are most significant. Thus a new code was derived from the Elias code. We call this code a code with logarithmical growth see 2'nd column in Tab. 2. The length of this code grows logarithmically with the encoded number l_i.

3.3.2 Combination of the Huffman code and code with logarithmical growth

We have chosen the combination of the Huffman coding and the prefix code with logarithmical growth for better compression ratios. The used code with logarithmical growth is shown in Tab. 2. The combination of the Huffman code and the prefix code with logarithmical growth is illustrated in Fig. 9. The number K is the highest number coded by the Huffman code. If $K \neq 0$ then a modified code placed in the column $round(log_2(K))$ is chosen as a modification of the above mentioned code with logarithmical growth.

There are three options:

A $K = 0$, i.e. only the code with logarithmical growth is used (basic variant).

B K is fixed. Experiments have shown that best values are typically $K = 32$ or $K = 64$.

C The K is set optimally according to data to be compressed. This option is described next.

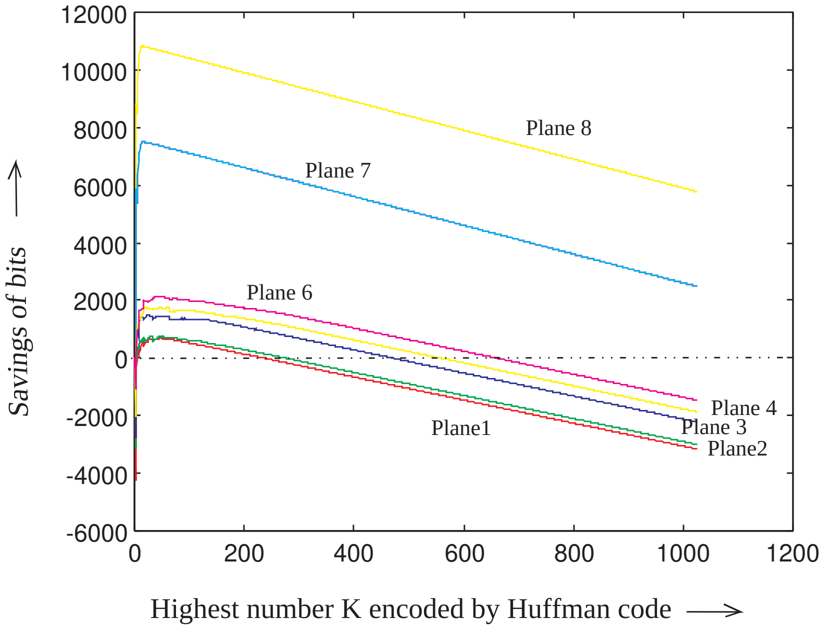

Figure 10: Changing of the overall size in dependence of a K value

The length L_1 of the Huffman code table is an increasing function

$$L_1 = 4 + (n+1)\,\log_2 S \qquad [bits], \tag{5}$$

where S is the size of Huffman code space (that corresponds to the longest coded symbol). $(n+1)$ is number of entries in the table[8]. The $\log_2 S$ term gives the length of the table entry in bits. The length of the coded residuals, roughly speaking[9], monothonically decreases. The difference between these two monotonic function is our function. We are looking for its maximum that corresponds to optimal K.

The exact dependence between the number K and the length of encoded residuals is illustrated in Fig. 10. Eight curves, one for each bit plane, are depicted.

3.4 Adaptive compression

The two previous predictors had a fixed number of probe cells (3 in the *FH-2DOF* case and 7 in the *FH-3-DOF* case) in a fixed arrangement. The desire is to exploit more complex context in the predictor. The support set of the predictor can have more cells in more dimensions[10]. Such predictor allows to encode more complicated structures in the image as, for instance, some fine texture.

The size of the predictor support is limited from the practical point of view by the size of the frequency table. It grows exponentially with the number of cells k in the predictor support set. Its size is 2^k. Say, for 20 cells, the frequency table size is $2^{20} = 1$Mbit. This is

[8] n entries are used for numbers and 1 is used for the prefix.
[9] The small perturbation are caused namely by the changes of the exponent, e.g. from 2^3 to 2^4.
[10] Don't forget to the previous constraint for choosing cells

Number	Basic Code	1 st modification	2 nd modification	3 rd modification
1	00	00	000	0000
2	01	01	001	0001
3	100	1000	010	0010
4	101	1001	011	0011
5	11000	1010	10000	0100
6	11001	1011	10001	0101
7	11010	110000	10010	0110
8	11011	110001	10011	0111
9	1110000	110010	10100	100000
10	1110001	110011	10101	100001
11	1110010	110100	10110	100010
12	1110011	110101	10111	100011
13	1110100	110110	1100000	100100
14	1110101	110111	1100001	100101
15	1110110	11100000	1100010	100110
16	1110111	11100001	1100011	100111
17	111100000	11100011	1100100	101000

Table 2: Code with logarithmical growth

too much. Of course, not all predictor cells bear the same amount of information. The trick we propose is to through away some cells from a relatively large probe adaptively. This is done according to the content of the particular image.

Predictor probes are constructed in the similar manner as before. They have more cells, e.g. 20, and could span n-DOF. Assume the RGB colour image. If we compress one of the colour components of the RGB colour image, we have 3-DOF due to the gray-level image plus 2-DOF if the context in the same bit-plane in other two colour components is taken into the account. Probes are again arranged in couples that differ only in the value of the actual (estimated) cell.

The frequency analysis comes next. The result of it is the frequency table with the size 2^{k+1}. k is the number of support cells of the predictor[11]. This table is usually too big. The aim is to select only those i cells, $i < k$, that are the most informative and thus reduce the length of the frequency table to 2^i. The selection step does not need to pass through original image data any more and thus is independent of an image size.

We may look at cells of the predictor as at features common in statistical pattern recognition. Features contribute to the estimate of the representative point of the probe. Recall that k is the number of probe feature points (cells).

After the frequency analysis, it is known how often a particular predictor probe occurs in the image. Moreover, the features can be ordered according to their importance for estimation of the representative cell. We found an algorithm that allows to dig out influences

[11] An actual cell doesn't belong to the set of k elements, because it has special properties.

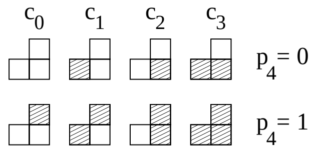

Figure 11: Extracted feature No.3

of individual features from the frequency table. Input data are located inside the frequency table.

Let us explain a decoupling on the specific example with $k = 3$. All configurations of the probe were shown in Fig. 1. Assume for a moment that we like to decouple influence of the feature No. 3 (the top left corner of the probe). If we omit this feature couples of probe will merge as is depicted in Fig. 11. The frequency of the new probe configuration is the sum of frequencies of its two constituting probe configurations, which differs with the value of removed cell only.

A good representation of the frequency table allows very effective bit operations. Recall that the actual cell has the highest index. Let the frequency table be stored in a vector $\vec{\mathbf{q}}$ with 2^{k+1} elements. First 2^k elements of the vector $\vec{\mathbf{q}} = (q_0, \ldots, q_k)$ correspond to frequencies of the upper row of probe configurations.

Let e_{all} be the number of incidences of all less frequent configurations of the probes. Let b be an index to the first element of the second half of the frequency table, $b = 2^k$ and c is 2 power the position of the removed (examined) feature[12].

$$e_{all} = \sum_{i=0}^{b-1} \min(q_i, q_{i+b}) \tag{6}$$

Let us define the *estimativness* e_c of the feature indexed by c.

$$e_c = e_{all} - \frac{1}{2} \sum_{i=0}^{b-1} \min\left(q_i + q_{i \oplus c}, q_{i \oplus b} + q_{i \oplus c \oplus b}\right) \tag{7}$$

The factor $\frac{1}{2}$ in front of the sum comes the fact that the estimativness of the feature c is in fact scanned twice. The estimativness e_c tells how increases a number of residuals (fails of the predictor), when feature c would be omitted. The operator $\oplus$ means logical eXclusive OR operation in this content.

The feature with lowest estimativness is ruled out from the probe. The same algorithms can operate iteratively and throw away second less informative feature, etc.

At last, we have to tell when to stop iterations. The number of residuals e_{all} increases monotonically if we rule out features gradually. The number of entries in the frequency table decreases monotonically and thus its length is decreased. The difference between these two functions has an extreme that we look for. If the increase of the length of the coded residuals (given by the estimativness e) is bigger than decrease of the table length or the table is empty then the algorithm stops.

[12]In the first equation the operator $+$ does same operation as the operator $\oplus$

3.5 Color images with palette function

This section discuses about a lossless compression of pseudo colour images (images with a palette) see also [Ska93] for description of this colour representation. Our predictor and a lot of other predictors are based on the assumption about the continuity of the image function. But many types of palette functions can break this feature.

Let us assume the palette image is mapping $color = palette(f(x, y))$. It is usually created from the true color image (RGB camera, colour scan) or a hand painting program. The algorithm that quantized the original true colour image caused discontinuities in index function $f(x, y)$. This effect decreased the compression ratio.

Above mentioned problem can be solved some ways. The simplest way was described in [Fry93]. This way consists of adding brightness index to each colour item and simply sort all palette items according to this brightness.

Our idea is to re-establishes the continuity of the index function $f(x, y)$. We change both the index function $f'(x, y)$ and the $palette'$ function in the way that the resulting value of the image function for all x, y remains the same:

$$palette(f(x, y)) = palette'(f'(x, y)). \tag{8}$$

Each multi-level image can be treated as stacked bit planes. Our method minimizes the number of value changes in the current bit plane by rearranging bit planes below it. We start from the most significant bit plane. Then the algorithm proceeds to bit planes that are below it.

First of all, the table of neighbourhood relations of all colors (indices) is computed and stored in a table (triangular due to symmetricity of the relation). All image is traversed and information of adjacent pixels are stored into the table. Let Fig. 4 be the table of neighbourhood relation of pixel values which belongs to the Fig. 3. This table has same number of rows and columns and this number is equal to number of colors in the image. Number of the colors must be power of two[13]. The table is symmetrical along the diagonal axis.

For illustration let us assume an actual pixel $x_{i,j}$ in the position i, j with value $p_4 = f(x_{i,j})$. Values of the top pixel $p_1 = f(x_{i+1,j})$ and right pixel $p_3 = f(x_{i,j+1})$ are needed for counting neighbourhoods. Having values p_1, p_3, p_4 algorithm increments these cells in the table: $[p_1, p_4]$; $[p_3, p_4]$ and also $[p_4, p_1]$; $[p_4, p_3]$. Let $S_{i,j}$ is called element this table on position i, j [14].

When the table is ready then we can proceed to a next step. We want to split this table S into two groups G_1 and G_2. Groups are made in the first step according to the value of the brightness. The value of the element k is defined as index of the table of neighbourhood relations to its group for each colour.

The quality of each index can be calculated as a number of indices in the own group minus a number of indices of the other group.

[13] If not then some columns and or rows must be zero.

[14] Table must be symmetrical along diagonal axis and thus $S_{i,j} = S_{j,i}$

4 4 4 4 6 6 6 6 6 6 1 0 0 0 0 4 1 0 4 4 6 6 4 4 6 6 6 4 4 4 6 6 6 6 6 1 1 0
6 4 4 4 6 6 6 6 6 6 1 0 0 0 0 4 1 0 4 4 4 4 6 6 6 6 6 6 4 4 6 6 6 6 6 0 0 0
6 4 4 4 6 6 0 4 6 6 1 6 6 0 4 6 4 0 4 6 6 4 4 4 6 6 6 4 4 6 6 6 6 6 6 6 1 1
4 4 4 4 6 6 4 4 6 4 1 4 4 0 1 6 1 0 4 6 6 6 4 4 6 6 6 4 4 4 6 6 6 6 6 4 1 1
4 6 4 4 4 1 0 4 6 6 6 6 6 4 1 6 4 0 4 6 6 6 6 4 6 6 6 6 6 4 6 6 6 6 6 6 6 1
6 6 6 4 6 6 0 4 6 6 6 6 6 4 1 6 4 0 4 6 6 6 4 4 4 6 6 6 6 4 7 6 6 6 6 6 4 1
4 4 6 4 1 0 4 6 6 6 6 6 6 6 6 4 4 1 6 6 6 6 6 6 6 6 6 6 4 4 6 6 6 6 6 6 6 6
6 4 6 4 1 1 0 6 6 6 6 6 6 6 1 4 1 0 4 6 6 6 4 6 6 6 6 4 6 4 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 6 4 1 0 4 1 6 4 4 4 6 6 6 6 6
6 6 6 6 4 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 1 4 6 6 6 6 6 6 6 6
6 6 6 3 3 3 6 6 6 4 6 6 6 6 6 6 6 6 6 6 6 6 6 0 4 0 0 0 0 0 0 4 1 1 0 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 0 0 4 0 0 0 0 6 0 1 6 6 6 6
6 6 6 6 3 3 3 3 3 6 4 6 6 6 6 6 6 6 6 6 6 6 4 0 0 0 0 0 0 0 0 0 0 1 0 6 6 6
6 6 6 6 3 3 3 6 6 6 4 6 6 6 6 6 6 6 6 6 6 6 4 4 3 0 0 0 0 0 0 0 3 3 1 4 6 6
6 6 6 6 3 3 3 3 3 3 4 4 6 6 6 6 6 6 6 6 6 4 1 0 0 0 7 0 0 0 0 0 0 0 0 0 0 6
6 6 6 6 3 3 3 3 3 3 4 4 6 6 6 6 6 6 6 6 6 4 1 0 0 0 0 0 0 0 0 0 0 0 0 1 4 6
6 6 3 3 3 3 3 3 3 3 0 4 6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 6 6
6 6 3 3 3 3 3 3 3 3 3 4 6 6 6 6 6 6 6 6 6 0 3 0 0 0 2 0 0 0 0 0 0 0 0 0 4 4 6
3 3 3 3 3 3 3 3 3 0 0 3 6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 6
6 3 3 3 3 3 3 3 3 3 0 3 6 6 6 6 6 6 6 6 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 6
3 3 3 3 3 3 3 3 0 0 0 4 6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 6
3 3 3 3 3 3 3 3 3 0 0 3 4 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 6
3 3 3 3 3 3 0 0 0 3 4 4 4 4 4 6 6 6 6 6 6 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 4
3 3 3 3 3 3 3 0 0 0 4 4 6 6 6 6 6 6 6 6 6 0 1 0 0 0 0 0 0 0 0 0 0 0 0 4 4 1
3 3 3 3 0 0 0 0 0 4 4 4 4 4 4 4 1 0 0 0 1 4 0 4 3 0 0 0 0 0 0 0 3 3 1 0 1 4
3 3 3 3 3 0 0 0 0 3 4 4 4 4 4 4 1 0 0 4 6 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 6
4 4 3 3 3 3 4 4 4 4 4 4 4 4 4 4 1 0 0 0 0 0 1 0 1 6 4 1 1 1 0 1 6 1 0 4 4 4
4 3 3 3 3 0 0 0 4 4 4 4 4 4 4 4 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 3 3 1 0 0 4 4
6 4 3 4 4 6 6 4 4 4 4 4 4 4 4 4 1 0 0 0 0 0 0 1 0 0 4 1 1 4 6 6 6 4 4 4 4 4
4 4 4 3 3 3 4 4 4 4 4 4 4 4 4 4 1 0 0 0 0 0 1 1 0 0 4 4 1 1 4 4 0 0 4 4 4 4
1 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 0 1 1 6 6 6 6 6 6 4 4 4 4 4
6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 0 0 4 6 6 6 6 6 6 4 4 4 4 4
1 0 6 6 6 6 6 6 6 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 1 1 1 6 6 6 6 6 4 4 4 4 4 4
1 4 6 6 6 6 6 6 4 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 1 1 4 6 6 6 6 6 6 4 4 4 4 4
0 0 0 6 6 6 6 6 6 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 1 1 0 0 0 6 6 6 4 4 4 4 4 4
0 0 4 6 6 6 6 6 6 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 1 1 1 0 6 6 6 6 4 4 4 4 4 4
1 1 4 4 6 6 6 6 6 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 1 1 1 1 4 6 6 6 6 6 4 4 4 4
0 1 1 4 6 6 6 6 6 4 4 4 4 4 4 4 1 0 0 0 0 0 0 0 1 1 0 0 1 6 6 6 6 4 4 4 4 4
4 4 4 4 4 4 6 6 6 6 1 0 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 6 6 6 4 1 1 1
1 4 6 6 6 6 6 6 6 4 4 4 4 4 4 1 1 1 0 0 0 0 0 1 1 1 1 1 4 4 4 6 6 6 4 4 4 4
6 6 1 1 1 4 6 6 6 6 6 1 0 0 0 0 1 4 4 4 4 6 6 6 4 4 4 6 4 0 6 6 6 6 6 6 1 0
6 4 0 0 0 1 6 6 6 6 4 0 0 0 0 0 1 1 4 4 4 4 6 6 6 4 4 4 6 6 6 6 6 6 6 0 0 0
6 6 6 4 4 4 6 6 6 6 6 1 1 0 0 0 4 6 6 6 6 6 4 6 4 6 6 6 6 4 6 6 6 6 6 6 1 0
6 4 4 4 1 6 6 6 6 6 6 1 1 0 0 0 1 4 4 6 6 6 4 4 4 6 6 4 6 0 4 6 6 6 6 6 1 1
6 6 6 4 1 1 1 6 6 6 6 6 1 0 1 1 4 6 6 6 4 4 6 6 6 6 6 6 6 4 1 6 6 6 6 6 6 6
6 4 6 4 4 1 6 6 6 6 6 1 0 0 0 1 4 6 6 6 6 4 4 6 6 6 6 6 6 4 4 6 6 6 6 6 4 0
6 6 6 6 0 0 6 6 6 6 6 6 6 6 1 1 4 6 4 6 6 6 6 6 6 6 6 4 4 0 0 6 6 6 6 6 6 6
4 6 6 4 1 0 4 6 6 6 6 6 6 1 1 1 4 6 6 6 6 4 6 6 6 6 4 6 6 4 1 6 6 6 6 6 6 6
7 7 3 3 3 7 7 7 6 6 6 6 6 6 6 6 4 6 6 6 6 6 6 4 6 4 1 1 6 6 6 6 6 6 6 6 6 6
6 6 4 4 1 1 6 6 6 6 6 6 6 6 6 1 4 4 4 6 6 6 6 6 6 4 4 4 1 0 4 6 6 6 6 6 6 6

Table 3: Testing image for palette color preprocessing

	1	2	3	4	5	6	7	8
1	898	270	5	85	169	0	135	5
2	270	199	0	16	200	0	121	1
3	5	0	0	0	0	0	0	1
4	85	16	0	291	46	0	57	0
5	169	200	0	46	752	0	584	3
6	0	0	0	0	0	0	0	0
7	135	121	0	57	584	0	1736	12
8	5	1	1	0	3	0	12	0

Table 4: Table of neighbourhood relations of pixel values

If the element k belongs into the group G_1:

$$w_k = \sum_{j \in G_1; j \neq i} S_{i,j} - \sum_{j \notin G_1; j \neq i} S_{i,j} \tag{9}$$

If the element k belongs to the group G_2 then the equation is modified:

$$w_k = \sum_{j \in G_2; j \neq i} S_{i,j} - \sum_{j \notin G_2; j \neq i} S_{i,j} \tag{10}$$

Let us define the function of the quality of each group:

$$q(W_1) = \sum_{x \in G_1} w_x \quad q(W_2) = \sum_{x \in G_2} w_x \tag{11}$$

And the overall quality of splitting into two groups (optimization criterion):

$$q = q(W_1) + g(W_2) \tag{12}$$

The global optimization criterion is the the sum of qualities of all indices in both groups. The worst index in each group with the lowest value w_k is found. The worst indices are swapped between groups. This is repeated until the optimization criterion decreases. Then algorithm undo last moving and stops. The division into two groups in the actual bit plane is done.

The algorithm proceeds to less significant bit planes. The task is to split both groups into four groups now. Previous group $G1$ is divided into two groups G_{11} and G_{12} and the group G_2 is divided to G_{21} and G_{22}. Groups $G_{11}G_{12}G_{21}G_{22}$ are obtained. Elements may be moved only between G_{11} and G_{12} and between G_{21} and G_{22}. The approach is similar than this described above.

This algorithm creates an optimal code for above described compression scheme. This code minimizes entropy in a highest bit plane and maximizes entropy in a lowest bit plane. The method does not depend only on the FH predictor and it may be used in another compression schemes.

3.6 Summarize of advantages and disadvantages

Let us summarize pros and contr of the method. The *advantages* are:

1. The method is fully automatic. No interventions from the user are expected.
2. Method could be quick because simple operations are used only. E.g. SHL, XOR, ADD, and selection from the table.
3. The compression results are on the top of current methods.
4. The method does not upper limit of the compression performance.
5. Substantial parts of the compression and decompression can run in parallel.

6. The predictor that constitutes the core of the method is extensible to more degrees of the freedom.

7. When the resolution of the image increases, the compression ratio does not degrade. This happen in the case of pyramidal algorithms, say MLP see [Fry93].

The *disadvantages* of the method are:

1. The compression needs at least two passes through image data. When the Huffman coding is used even three passes are needed. This passes are fully sequential.

2. The decompression algorithm is very sensitive to errors in data. This is the same with other methods.

Figure 12: Lena

4 Experiments

The goal of experiments was to (1) test properties of the proposed method, (2) compare it with the results of others, and (3) find possibilities of further improvements.

4.1 Gray level images

Let us show the behaviour of the proposed *FH-3-DOF* compression on the standard gray-level testing image Lena, see Fig. 12 of the size 256×256×8 bits. Selected bit-planes of the original and compressed image are shown in Fig. 13. Notice that the compression is more effective on bit-planes corresponding to more significant bits as there are usually less changes. Notice how well the residuals (from the predictor) express the significant features in the image. The mentioned experiment is quantified in tabular form in Tab. 5. Notice how number of residuals increases from the most significant bit-plane to the least significant bit-plane. This result is obvious as in the least significant bit-plane there is much randomness. It is difficult to represent randomness by the estimator. The estimativness gives a clear insight into the predictor. The feature No. 6 and feature No. 4 from the neighbouring bit plane bear the highest amount of information.

Bit-plane	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
2-DOF residuals	2010	5477	10177	15067	21865	28595	31771	32214
3-DOF residuals	empty	3841	5935	9421	15045	21690	28622	31272
estimativness e_1	empty	25	160	218	233	91	125	320
estimativness e_2	empty	616	2165	3108	4930	4919	2521	457
estimativness e_3	empty	52	180	219	233	91	128	352
estimativness e_4	empty	1892	4101	5543	6611	6600	2813	447
estimativness e_5	0	14	123	143	92	26	112	361
estimativness e_6	1457	2097	3477	4169	5807	5031	2312	467
estimativness e_7	0	23	118	136	170	85	71	220

Table 5: Quantitative description of the FH-2-DOF and FH-3-DOF methods

The proposed adaptive compression method *FH-Adapt* was compared on several images from the standard JPEG image set [AT94]. Methods and results were taken: SEGMENT from [RA96], Portable Video Research Group at Stanford PVRG-JPEG from [CAJ96], Rounding transform from [JCP96], and Logic coding from [CAJ96].

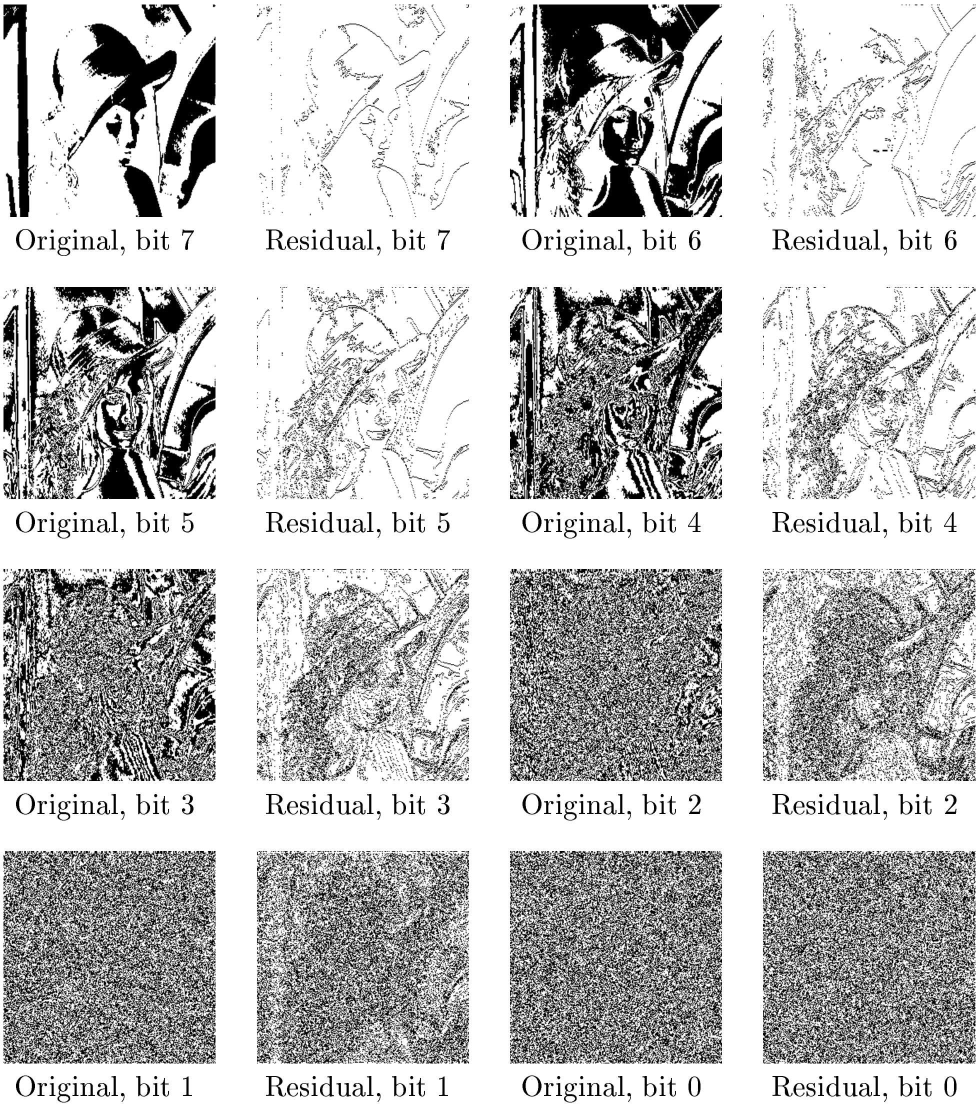

Figure 13: Comparison between bit-planes of the original image and bit-planes storing residuals.

Method/Image	Lena	Baboon	Boats
PVRG-JPEG	22.7-29.4	6.7-13.7	24.9-33.1
Logic	26.1	12.6	29.3
Segment	*54.6*	*36.3*	empty
Rounding Tr.	37.6	empty	empty
PNG	20.7	*41.7*	33.7
FH-3-DOF	**34.0**	**19.1**	**41.2**

Table 6: Compression coefficients compared with results of others

The quantitative measure used for comparison of different methods is *compression efficiency CE* [CAJ96]

$$CE = \frac{\text{total input bytes} - \text{total output bytes}}{\text{total input bytes}} \cdot 100\% \tag{13}$$

The table Tab. 6 shows how our method compares with others. The value CE shows that our method performs well. It was gotten over by Segment and PNG in the case of an image of a monkey (Baboon). The reason is that the predictor cannot cope with the complicated structure of monkey's hairs. Segment performs also better on Lena. Unfortunately we were not able to experiment with Segment because its code has not been available for us.

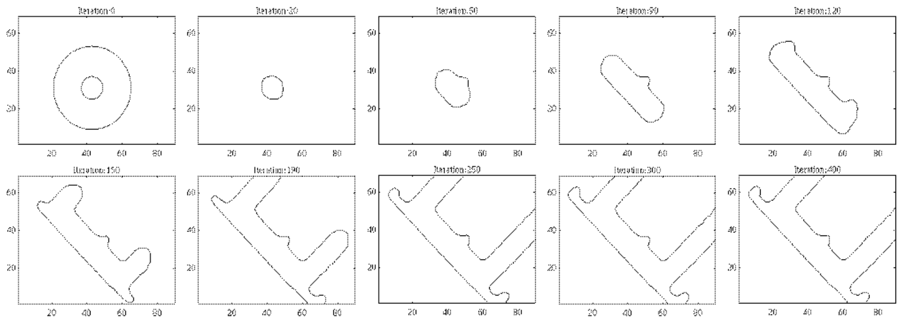

Figure 14: Image of the scanned technical drawing

The Table 8 shows the comparison of our method with Lena and a technical drawing, see Fig. 14. This is an image which is perfectly suited for our compression method as can be seen from results.

The last Table 7 shows what is the influence of three methods of coding residuals and three different predictors. This allows us to understand, what individual improvement brings to us. The values in the table are the length of the compressed 512 $\times$ 512 Lena in bytes. The PNG compression is very similar to the IBM Q80 coder [ISO94]. IBM Q80 is obsolete and its implementation is inaccessible for us.

4.2 Color images with palette function

The approach how to cope with a non-continuous image function was described in the chapter 3.5. This situation occurs mainly at color images with palette function. For experiments

Method/Coding	Exp. code only	Fixed Huffman	Moving Huffman
FH-2-DOF	203000	200758	200599
FH-3-DOF	178655	176021	175922
FH-Adapt.	178130	175541	175432

Table 7: Comparison of proposed 3 predictors and 3 types of coding techniques

Type of data	Lena	CE Lena	Drawing	CE Drawing
Uncompressed	262656	0	106940	0
ZIP	225007	15.62	10224	90.44
ARJ	228389	14.35	10380	90.29
PCX	274256	-2.85	38624	63.88
GIF	278771	-4.54	17290	83.83
FH-Adapt.	**175434**	**34.21**	**7297**	**93,18**

Table 8: Comparison with typical compressions

was used test image *Descent*, see Fig. 15 of the size 640×480×8 bits with a palette function.

Original image　　Gray sorted by Y　　Optimal code

Figure 15: Comparison between 3 modifications of color image.

4.3 Computational time aspects

Time tests was measured on RAM disk to avoid randomness in approaching to the data carrier. Compression the image Lena 512 × 512 × 8Bits using $FH - 3DOF$ method spends 6.2s. Decompression is approximately 2x faster - in Lena case it runs 2.7s. Loading/Storing a data without compression spends 0.1s. All timing tests was done on the machine Pentium 120MHz and with the RAM disk.

Name of image	Original image	FH	Gray+FH	Optimal+FH
descent	65078	24384	27211	21930

Table 9: Sizes of compressed images with palette

p_4	p_3	p_2
p_5	p_9	p_1
p_6	p_7	p_8

Figure 16: Probe for Texture Classification

5 Limitations of previous approach and future work

We describe here ways of possible improvement the compression ratio, compression performance and possible another use of developed algorithm. I would like to investigate these below listed ideas in the future work.

5.1 Texture classification

The estimativness is now used for the construction of an optimal predictor now. But a value of the estimativness could be used for another purpose. Matrix of estimativness describes something about a type of an image content. It can tell us something about the type of the texture inside the image.

The probe in the Fig.3 is causal and is well suited for construction of the causal predictor. The centric probe as is shown in Fig.16 should be better for texture classification.

5.2 Improvement of the compression performance

The compression performance may be improved by modification of the algorithm (for our set of test images). We guess that only minor improvement is possible without big changes of the algorithm.

First tested approach is to use Gray code for encoding the image function $f(x, y)$. See Fig.18 which shows the case of the simple 1D function. Three lines on the bottom site of this image show changes of 3 bits from the number $f(x, y)$. This approach is based on the idea that number of changed bits is smaller than this number with a normal binary code. Simple table can be used for the conversion between Gray and binary code. Translating data through the table is very fast and simple. This idea seems to be correct, but improvement is

Plane	Binary	Gray
4	3	3
3	5	3
2	8	5

Figure 17: Number of changes for binary and gray code

only from 1% to 2%. This is caused by the fact, that the adaptiveness part of the non-linear predictor eliminates some changes.

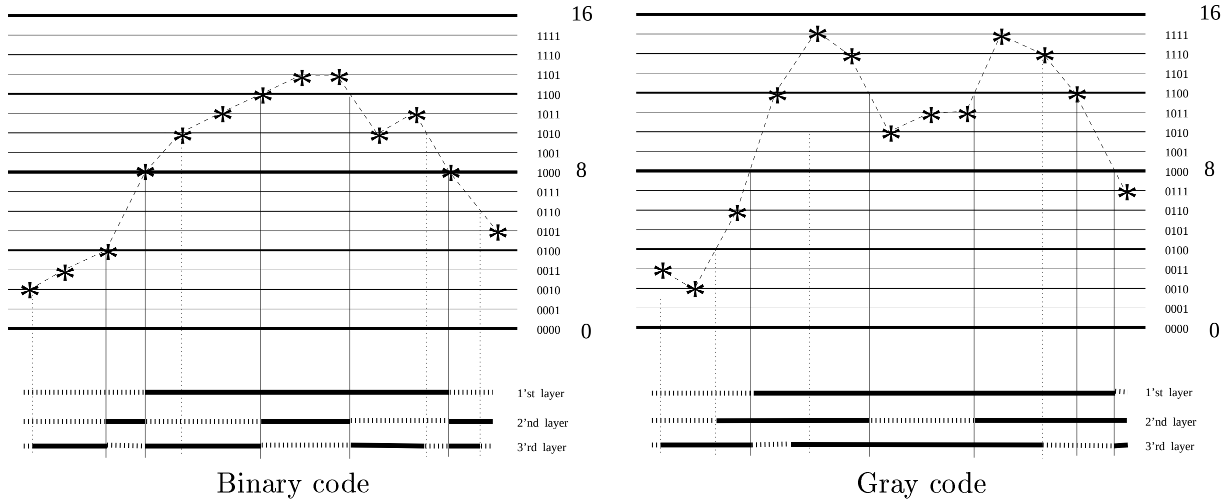

Figure 18: Approach of encoding gray level images

The method proposed in this report uses only a semiadaptive data model. This means that predictor goes through the data and computes an optimal data model. The model is used for generation output data after finishing this phase. We could design a fully adaptive data model, which creates a predictor description during passing through image data, see Fig.19. This model needs only one pass through data on a path compression and decompression sides. The decoding algorithm must also dynamically build a data model.

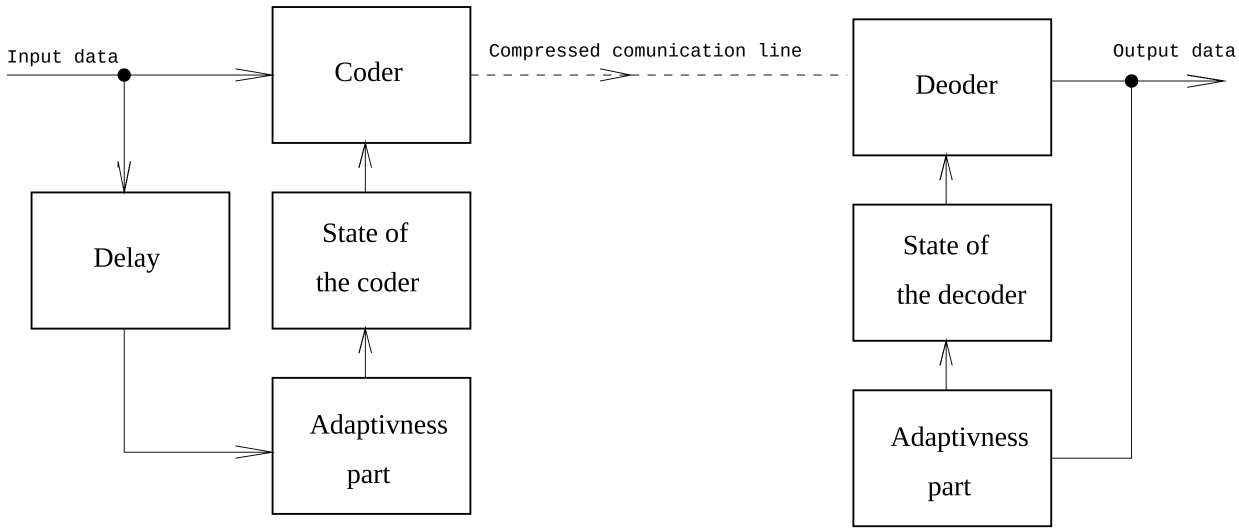

Figure 19: Schematic illustration of the fully adaptive data model

6 Thesis plan

I like to do the following in my research leading to dissertation thesis:

A To improve my knowledges about lossless compression algorithms of others. This item includes studying of available articles.

B To improve the our own new compression method. This task includes (a) increasing compression ratio, (b) decreasing computational time, and (c) find out limitations of this approach.

C Do experiments with more degrees of freedom of the probe (e.g. color images).

D To adapt the method in such a way that it could be usable in applications.

More particular description of these topics is the following:

I shall seek for more effective algorithms.

I like to clarify a mathematical description of the algorithm used in the theoretical part.

I intend to improve the time consumption of experimental compression program in the experimental part. I shall test compression algorithm on a large set of images and I shall find most common features in the images.

The implementation of the experimental algorithm should be polished and will be prepared to integration into graphical programs.

7 Conclusions

We have proposed a new lossless image compression method that performs in many respects very well and could be compared with other known methods. Basic idea of the method is a bit unusual, i.e. using predictors that predict differences in one bit only. This idea comes from M.I. Schlesinger [Sch89] and we learned about it during his series of lectures in Prague in Spring 1996.

We propose three significant extensions. First extension is to have a predictor with more degrees of freedom. Second one is to use adaptive predictors according to statistics of the particular image to be compressed. The method is namely suitable for images with strong structure as e.g. technical drawings, maps, etc. Third we extend the method such way that is abble to cope with images with pallette.

The reader might want to test the method himself. If he wants to do so see the www page
http://cmp.felk.cvut.cz/˜fojtik/ for the actual version of the compression algorithm.

References

[Ani89] Jain Anil, K. *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[AT94] R.B. Arps and T.K. Truong. Comparision of international standards for lossless still image compression. *Proceedings of the IEEE*, 82(6):889–899, June 1994.

[CAJ96] A.K. Chaudhary, J. Augustine, and J. Jacob. Lossless compression of images using logic minimization. In P. Delogne, editor, *Proceedings of the International Conference on Image Processing*, volume II, pages 77–80, Lausanne, Switzerland, September 1996. IEEE Signal Processing Society, Louvain-La-Neuve, Belgium.

[Dav69] Spencer David, R. Bit plane encoding of continuous tone pictures. In Fox Jerome, editor, *Proceedings of the symposium on Computer processing in communications*, volume XIX, pages 101–120, Brooklyn, N. Y., April 1969. Polytechnic press of the polytechnit institute of Brooklyn.

[Fry93] Michael Frydrych. Textové informační systémy. Master's thesis, Karlova Univerzita MFF, 1993.

[HŠ92] V. Hlaváč and M. Šonka. *Počítačové vidění.* Grada Praha, 1992.

[ISO94] ISO/IEC 10918-1: Information technology-digital compression and coding of continuous-tone still images: Requirements and guidelines, 1994.

[JCP96] H.Y. Jung, T.Y. Choi, and R. Prost. Rounding transform for lossless image coding. In P. Delogne, editor, *Proceedings of the International Conference on Image Processing*, volume II, pages 65–68, Lausanne, Switzerland, September 1996. IEEE Signal Processing Society, Louvain-La-Neuve, Belgium.

[JW94] D.Murray James and vanRyper William. *Encyclopedia of Graphics File Formats.* O'Reilly & Associates, Inc., Sebastopol, California, USA, 1994.

[Mar82] D Marr. *Vision – A Computational Investigation into the Human Representation and Processing of Visual Information.* W.H. Freeman and Co., San Francisco, 1982.

[Mel96] Bořivoj Melichar. *Textové informační systémy (in czech) (Text information systems).* FEL ČVUT, 1996.

[Pet97] Kolář Petr. Nový grafický formát PNG (in czech). *Softwarové noviny*, (2):75–79, February 1997.

[RA96] K. Ratakonda and N. Ahuja. Segmentation based reversible image compression. In P. Delogne, editor, *Proceedings of the International Conference on Image Processing*, volume II, pages 81–84, Lausanne, Switzerland, September 1996. IEEE Signal Processing Society, Louvain-La-Neuve, Belgium.

[Sch89] M.I. Schlesinger. *Matematiceskie sredstva obrabotki izobrazenij, in Russian, (Mathematic tools for image processing).* Naukova Dumka, Kiev, Ukraine, 1989.

[Ska93] Václav Skala. *Světlo barvy a berevné systémy v počítačové grafice (in Czech).* Academia, Praha, 1993.

[15]

[15] This is not complete list of all literature which I have read yet. This is a list of articles which closely relates to main topic of this minimal thesis.

A Disk File format

The compressed data should be stored on some carrier in a reasonable format. The structure of well known graphics formats are detailed described in [JW94].
Such format would support several features:

- Unambiguous identification.
- Both compressed and uncompressed modes.
- Good expansibility in future.
- Do not store useless information (short overall length).
- Format should not be very complicated.
- Error detection in data.

Today's best formats have short identifier in the header. This identifier is intended for identification of the file format and a version of this format. The header is followed by description of individual objects. Each object descriptor describes only the type of an object and its size. The object descriptor is directly followed by data of the object. It is a good idea to use one directory object which allows hierarchical storing of data. Such object envelopes included objects below it.

Type of data	Identification	Extension	Hierarchical data
GIF	$1/256^3$	Yes	No
PCX	$1/256^2$	No	No
TIFF	$1/253^2$	Yes	No
FTG	$1/245^4$	Yes	Yes

Table 10: Description of todays well known data formats

We cannot modify any of well known currently used image formats for our purposes. Thus we decided to create a new format. This format can be used for storing any bitmap. We will call this format FTG.

Construction of a new graphical format is now under development. It is not ours primary task. On the other hand the format is needed for reasonable work with image compression techniques. The description of currently used data format follows.

For the reasonable manipulation with compressed image data we developed the file type identifier. It must be present on the beginning of each file.

```
typedef struct
    {
    char Identif[4];
    WORD version;
    } ImgHeader;
```

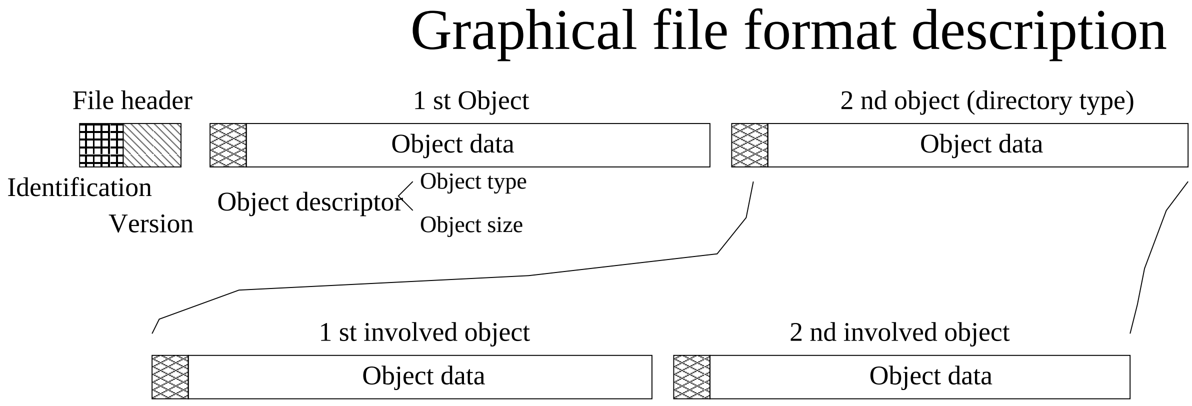

Figure 20: Structure of file format

Each file has a 4 byte long identifier. This item is used for correct autodetection of this file format.

version is a version number of this file. Hi byte contains a major release number This basic specification discuses about version 1.0. Lo byte contains a minor release number.

A header if followed by set of objects. Each object must have small header to for identification its type. The descriptor of the object is listed here:

```
typedef struct
    {
    BYTE type;
    DWORD size;
    } ImgObject;
```

A.1 Uncompressed data

If the data are uncompressed, then they are stored simply row by row. In this release, data are not stored into separate bit planes. No extra bytes are inserted at the end of each row.

A.2 Compressed data

Data are stored in separated bit planes in this situation. All data part of disk is modified to the bitstream.
At the beginning of each bit plane following information is stored :

Compression algorithm does internal autodetection of compression results. If these results seems to be bad (compressed size of the bit plane is greater than uncompressed) then all plane is stored uncompressed after *compressed* bit is reset.

type	bit size	description
compressed	1	Is this plane compressed?
probe	16	Actual shape of the probe
histogram	$2^{probesize}$	decompression data
maskDx	2	mask for RLC compression of the first column
maskDy	2	mask for RLC compression of the first row
IsHuffman	1	Is used Huffman code?
HufItems	3	Count of items coded using huffman.
ItLength	2	Length of each entry (in bits).
HufData	HufItems*ItLength	Length of Huffman table.
main data	?	Main image data for each plane.