High-level COMPROMISE feature set (version V3.0)

1. Introduction

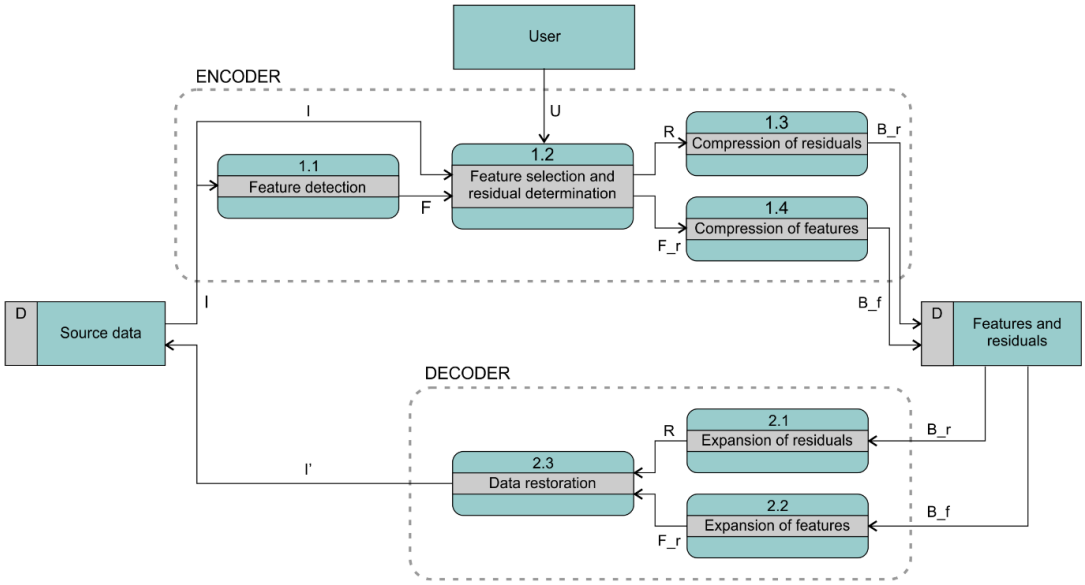

Fig. 1. Concept of the COMPROMISE data compression methodology

The main result, planned in COMPROMISE, is a **universal data compression methodology** with a **unified taxonomy of features.**

- The universal methodology means transformation of the input data stream into a representation, which enables utilization of a unique lossless data compression framework in the **lossless, near-lossless and lossy compression mode**. Here the latter two mean the lossless entropy coding preceeded with a near-lossless or lossy data preparation, respectively.
- The unified taxonomy of features means the design of a data representation, suitable for the **domain-independent** data compression and decompression.

The COMPROMISE methodology will be realised through the development of a new data compression paradigm, based on integration **of features and restoration methods.**

- The **features** introduce **prediction functions** for estimating the associated **patterns** of data primitives (samples). We do not assume overlaps between the patterns of different features. The predicted values are subtracted from the original input values to obtain the so-called **residuals**, expected to be better compressible than the originals.
- The **restoration** is a generalization of the traditional data expansion (decompression) method. The idea is to identify the data that may be omitted during the compression, and then still be restored with a sufficient quality (lossless, near-lossless, lossy) **from the context**.

The methodology will be validated in at least four pilot domains: digital audio, biomedical signals (both 1D), images (2D), and sparse voxel grids (3D). All three compression modes, lossy, near-lossless and lossless, are not necessarily equally important and need not be treated equally in all domains. For biomedical data, the lossless mode will absolutely dominate, but similarly, for example, when compressing vector fields, we will be primarily interested in the lossy mode. The methodology should therefore not constrain any existing data compression method by forcing adaptation to the proposed paradigm. In fact, the latter is designed to be so general that, in the extreme case, it will also allow the representation with features only or with residuals only.

2. Uncompressed input and output data streams

The data of interest in COMPROMISE is a limited (finite) **stream of samples**, organised in a clearly defined order.

A **sample** is an individual data item (data primitive) of the input stream I (Fig. 1) of the data compression pipeline, or of the output stream I' (Fig. 1) of the data expansion pipeline. The sample is, for example, a discrete point on a line in 1D, a pixel in 2D, or a voxel in 3D. It is specified by the location and value.

- The **sample location** identifies the sample within the stream uniquely, i.e. an array index i in 1D, a pair of matrix indices (i, j) in 2D, or a triple of indices (i, j, k) in 3D. It can be given explicitly for each individual sample or implicitly through a topology, established by a uniform or hierarchical subdivision of the input stream.
- The **sample value** (data attribute, e.g. the amplitude, heigth, etc.) is a sequence of bits at the sample location, structured according to the data type specification.
  - **Integers** in the positional notation are highly convenient for compression. If transformed to non-negative values, either by adding the absolute value of the minimum or by considering the sign separately, residuals with multiple leading zeros, that may be simply omitted, are derived often.
  - **Floating point numbers** in the positional notation can be considered in the same way as integers. However, the scientific notation, e.g. IEEE 754, appears slightly more complex. Here the replacement of an input value by a residual potentially lowers the exponent and shortens the mantissa by shifting its non-zero digits to the left. However, the lengths of both, the exponent and the mantissa must be stored, which usually requires more bits than the length of a number encoded positionally.
  - **Multiple attributes**, e.g. RGB colour components or a pair of stereo audio samples, may also be attached to a single data stream sample or, alternatively, streams of individual components may be considered as separate data streams.
  - **Samples without values** are also possible. This makes sense if we are interested into a geometric shape only, i.e. which samples do represent the region of interest. This corresponds to the complete-grid data stream of Boolean values 0 and 1, where the non-shape samples (e.g. 0-values) are omitted.

  Non-numerical sample values are currently **not considered** in COMPROMISE because of inability to simply derive a »residual« as the difference between the input value (symbol) and its prediction. Although the symbols can be numerically encoded e.g. by ASCII codes, it is hard to judge whether, for example, 'B' is better predicted by 'A' or 'C' than by 'V' or 'W'? The residuals should thus be designed on other principles, e.g. probabilies of good and bad guesses (predictions), which is, however, hardly compatible with the COMPROMISE paradigm. Furthermore, text, DNA and other symbol-based sequences usually require lossless compression only, which deviates from the idea of a universal data compression methodology.

The encoder's **input stream** $\boldsymbol{I}$ is a domain-dependent array of samples $s_{i,j,k}$. Similarly, the decoder's **output stream** $\boldsymbol{I'}$ is an array of samples $s'_{i,j,k}$. The data representation must be flexible enough to support both, a complete regular grid or a sparse arrangement of samples. Furthermore, it must be useful in 1D, 2D and 3D data domains at least.

2.1 Representation of a complete regular grid

A complete regular grid contains all the samples of a limited interval of an 1D signal, all the pixels of a raster image, or all the voxels of a limited voxel space. I is thus:

$$I \leftarrow \langle s_{i,j,k} \rangle,\ 0 \le i < resX,\ 0 \le j < resY,\ 0 \le k < resZ. \quad (1)$$

Here $s_{i,j,k}$ is the value of the sample at the location (i,j,k) in I, while $resX, resY$, and $resZ$ represent resolutions (the number of samples) in 3D Cartesian coordinate directions. In 1D domain, where $resY = resZ = 1$, we may omitt indices j and k (both 0 all the time).

$$I \leftarrow \langle s_0, s_1, \dots, s_{resX-1} \rangle. \quad (2)$$

Similarly, index $k = 0$ may be omitted in 2D domain, where $resZ = 1$.

$$I \leftarrow \begin{bmatrix} s_{0,0} & \cdots & s_{resX-1,0} \\ \vdots & \ddots & \vdots \\ s_{0,resY-1} & \cdots & s_{resX-1,resY-1} \end{bmatrix}. \quad (3)$$

The decoder's output stream I' is:

$$I' \leftarrow \langle s'_{i,j,k} \rangle,\ 0 \le i < resX,\ 0 \le j < resY,\ 0 \le k < resZ. \quad (4)$$

Equations (2) and (3), orginally used for the input stream I, can be adapted trivially to 1D and 2D simplifications of I'.

In the case of multiple attributes, samples of I and I' consist of $c > 1$ components $\langle {}^0 s_{i,j,k}, \dots, {}^{c-1} s_{i,j,k} \rangle$ and $\langle {}^0 s'_{i,j,k}, \dots, {}^{c-1} s'_{i,j,k} \rangle$, respectively.

2.2 Representation of sparsely arranged samples

A complete regular grid has a well-known advantage that, when the samples are stored sequentially in the raster scanning order, the locations of individual samples need not be stored. On the other hand, there are quite frequent situations when a huge amount of samples has unknown (not explicitly acquired), redundant (repeated in long runs or predictable trivially in some other manner), or irrelevant values (see the »Samples without values« paragraph in Section 2). In such cases, it often makes more sense to omit »non-interesting« samples, although some extra bits might be spent to store the locations of the interesting ones. The definitions (1) and (4) of I and I' should, thus, be replaced by (5) and (6):

$$I \leftarrow \langle (i,j,k,s_{i,j,k}) \rangle,\quad \{(i,j,k)\} \subseteq [0, resX-1] \times [0, resY-1] \times [0, resZ-1], \quad (5)$$

$$I' \leftarrow \langle (i,j,k,s'_{i,j,k}) \rangle,\quad \{(i,j,k)\} \subseteq [0, resX-1] \times [0, resY-1] \times [0, resZ-1]. \quad (6)$$

3 Features and residuals

*A **feature** is a piece of information that possesses high discriminative/predictive value for human interpretation or machine processing (e.g., computer vision, classification) of an input data stream I.*

As stressed in the Introduction, each feature f introduces some common rule/rules to predict the values of an internally (within f) specified pattern of samples from I. The predictions are then subtracted from the samples' values from I to obtain so-called residuals, which are expected to be better compressible than the original samples. The residuals are stored in the **stream of residuals $\boldsymbol{R}$**

(see Fig. 1), but some may also be omitted under certain conditions and then restored from the context automatically.

The **desription of the feature f** consists of:

a) $f.\boldsymbol{header}$**:** definitions of presence and structure of data in other two components.

b) $f.\boldsymbol{pattern}$**:** sequence of samples from I, affected by the feature. Note that we do not need the sample values $s_{i,j,k}$ here, but only their indices (i, j, k). Before the compression, each sample of $f.pattern$ is either:
   a) represented by the residual in R, obtained through utilization of $f.prediction$,
   b) omitted, because the context provides all information for satisfactory restoration, or
   c) coded directly in $f.prediction.control\_data$.

c) $f.\boldsymbol{prediction}$**:** unambiguous rules together with control data, which determine how the feature affects samples from $f.pattern$. The most interesting component is $f.prediction.function(sample\ s)$, which is used to compute the predicted (interpolated, approximated, or otherwise estimated) value for each sample $s \in f.pattern$. The residual $res(s) = s - f.prediction.function(s)$ is then computed and, if necessary, stored into R.

Two streams of features are handled in COMPROMISE (Fig. 1):

- The **feature stream $\boldsymbol{F}$** is the domain-dependent output stream of the feature detection module. The associated patterns may overlap. The overlaps can be so strong that individual features are redundant. This is best detected and resolved before the data compression step. Therefore, the subsequent step of feature selection reduces the stream F. Besides this, each feature type, used in the domain-dependent strem F, is provided with a **feature interpretation function**, which translates a feature of such type into the corresponding domain-independent representation of the uniform feature taxonomy. F and the feature interpretation functions are not considered further in this document.
- The **reduced feature stream $\boldsymbol{F_r}$**, obtained by feature selection, has the following properties.
  1. Each sample s of the input stream I, represented by $res(s) \in R$, must be addressed by at least one feature.

$$s \in I \ \wedge \ res(s) \in R \Longrightarrow s \in \bigcup_{f_i \in F_r} f_i.pattern \tag{7}$$

  2. $f_i.pattern$ may additionally contain samples, which are not represented by residuals in R, but omitted and restored, or stored directly in the control block of $f_i.prediction$.
  3. An individual feature contains sufficient information to expand and/or restore all the samples in its pattern independently from other features.
  4. The above statements imply that each sample s of the input stream I is addressed by at most one feature.

$$f_i.pattern \cap f_j.pattern = \emptyset, i \neq j \tag{8}$$

3.1 f.pattern

The feature component $f.pattern$ stores a sequence of samples (their locations i.e. indices) from I, which are addressed by f. As already stated in Section 2, samples' location may be given explicitly for each individual sample or implicitly through a topology. We distinguish three types of pattern:

- **Segment:** geometrically connected sequence of samples.
- **Region:** geometrically disconnected sequence of samples. List of segments.
- **Key samples:** region of segments with a single sample each. Local extrema of sample values are often chosen as the key samples.

A segment with $\|f.pattern\| > 1$ is represented by:

a) border: *interval* in 1D, one or more *loops* in 2D, one or more shells in 3D. The interval is simply a pair of key samples, while **chain codes** are typically used in 2D or 3D.

b) **box:** *interval* in 1D, *rectangle* in 2D, or *block* in 3D. Two vertices are needed for representation. In many cases, they can be derived from the topology indirectly, as the box often represents a cell in a uniform grid or a node in a uniform hierarchical subdivision (binary tree, quadtree, octree). This information is obtained from the F_r stream header.

c) **Key samples.**

3.2 Data headers

A broad general COMPROMISE methodology is designed to compress data of different types (integer, floating point numbers, multiple numerical attributes, or only the location information without associated data values) in all three data compression modes (lossless, near-lossless, lossy) in various data domains in 1D, 2D and 3D. This places a requirement on the data compression pipeline to have a rich and flexible repertoire of settings, which must also be available to the decoder. The default values of these settings are given in the **default configuration file**, which may be overwritten by the data grouped into four types of data headers:

a) File header: highest hierarchical level of all headers. The settings defined here are applicable to all lower levels, if not overwritten there.

b) $\boldsymbol{F_r}$ **stream header:** settings valid for all features in F_r. If some

c) $\boldsymbol{f.header}$: settings applicable only for $f.pattern$ and $f.prediction$. They overwrite the settings from F_r stream header and from the default configuration file.

d) $\boldsymbol{R}$ **stream header:** settings valid for all residuals in R.

Table 1. Examples of settings in different level headers

Setting	Value	Defined in
Type of all headers	Binary or some markup language	First bit of file header
Dimension	1D, 2D or 3D	Default (3D); File header
Topology	Uniform, Hierarchical, None (sparse data)	Default (Uniform); File header
Data compression	BASC, Interpolative, …	Default (3D); File header; F_r (all feature types or separately for individual types); R
Compression mode	Lossless, Near-lossless, Lossy	Default (Lossless); File header
Compression parameters	Method-dependent	Default (3D); File header; F_r (all feature types or separately for individual types); R
Data value type	Integer, Float, Multiple (n) ints, Multiple (n) floats, None	Default (Integer); R; F_r
Data value coding	Direct value (PCM), Direct (DPCM), Residual (PCM), Residual (DPCM)	Default (Direct value PCM for features, Residual PCM for residuals); R; F_r
Feature repertoire	Indices from the catalogue (to be done).	Default; File header
Restoration method	From the repertoire of prediction functions	Default, File header
…		

Speciall attention must be paid to settings of $f.pattern$ of individual features. They are expected to be usually defined in $f.header$, but can also be inherited from F_r stream header or even from the default configuration. Furthermore, the Topology setting and end eventual Data value type = None from Table 1 must be meaningfuly applied here, to treat the residuals and other data correctly. The pattern settins are given in Table 2.

Table 2. Settings of $f.pattern$ of an individual feature f

Setting	Value	Comment
Pattern type	Segment, Region, Key samples	
Region settings	0 – unique for whole region, 1 – separate for each individual segment in region	For Region only

Segment description	Border, Box, Key samples, Border/box + additional key samples in the interior	For each segment or inherited
Chain code method		For segments with more than 1 sample in 2D or 3D, when Segment desription = Border
Box coordinates		When Segment desription = Box
Border values	None, Key samples, Interpolation (+number and locations of samples)	when Segment desription = Border or Box
Interior included	Yes/No	when Segment desription = Border or Box
List of segments		For Pattern type = Region
List of key samples		Contains locations and/or values when necessary

There are quite some combinations of settings possible obviously, and some are expected to be used more frequently than others, so the introduction of some optional configuration files makes sense.

3.3 f.prediction

A feature introduces the relations between the sample locations and/or between the sample values in the pattern, enabling compact pattern encoding. Of course, different relations exist and, consequently, different feature types must be introduced. However, they all follow the same behavioural pattern according to the type of the accociated pattern of samples.

a) Samples without values. When only a geometric shape is important, i.e. which samples do represent the region of interest, the whole job is done by describing the pattern. There is nothing to predict.

b) Segment described with key samples. When the values of all the samples of the pattern are given, there is again nothing to predict.

c) Segment described with border/box. The values of the border samples may be given explicitly with key samples completely. However, it is more usual that only few samples on the border have values attached directly, while the others are obtained by **prediction**. The prediction is also used to compute values of the interior pixels (if Interior included is set to Yes). The predicted values are compared with the original ones to compute the residuals, which are either stored or omitted then.

d) Border/box + additional key samples in the interior. The same strategy as above is used, but the prediction is enhanced by additional key samples.

e) Regions. Each segment with clearly identified border is considered according to c) or d), while those without explicitly defined border are handled by b).

The prediction mentioned in c) is determined in $f.prediction.function$, and can belong to one of the following function types.

- **Interpolation function** – it interpolates key samples and/or values on the segment border/box. The goal is to find such interpolation that the residuals of the estimated sample values are "optimal" (with the lowest entropy already or best compressible to achieve the lowest entropy).
- **Approximation function** – it approximates samples of a given region in an "optimal" way (best-fitting curve/surface). Key samples and or the segment border/box are used to define the control points to be fitted.
- **Extrapolation function** – it predicts the value of the observed patterns by using the values from some predefined neighbourhood pattern.

Concrete members of each of these function classes are still to be defined and parameterised. The result will be the **catalogue** of feature specifications. For example, linear interpolation in one of the coordinate directions represents the most basic interpolation function. Radial basis functions (RBA) are enother example of interpolation (approximation?). The majority of known predictors from domain-dependent lossless algorithms (e.g. FLAC, JBIG…) are examples of the extrapolation

functions. The semantic meaning of each function is less important and usually too limited for use in a unified feature taxonomy, although the terms as gradient, trend, etc. may also be considered.

Use of masks can further improve the estimations obtained by the interpolation, approximation or extrapolation. The **mask** is a predefined pattern of a few sample locations in a considered segment, aimed to locally attract the graph of the considered prediction function. At each mask location, a limited repertoire of sample values is offered and the closest to the concrete sample value is chosen for calculation of the residual. Hovever, it is much more important, that the mask locations also decrease the residuals in adjacent locations within some local neighbourhood. As there is a low number of mask locations and also a low number of predefined values at these locations, the mask can be compactly represented by a few bits only.

Feature hierarchy must also be provided at least in the most basic form. From practical (e.g. visualization or playback) reasons, the input stream is often subdivided into a uniform grid of blocks or hierchicaly into 2, 4 or 8 uniform blocks at each hierarchical level. In each block, however, a »classical« segmentation into segments, regions and key samples might be useful. **Relation trees** represent a step forward by exposing hierarchical relationships among features/patterns (not only uniform blocks) at lower levels. In this context, **rhythm** may be defined, for example, to denote a sequence of segments, which are somehow (uniformly or non-uniformly) arranged in the considered spatial domain (1D, 2D, 3D…), such that the sample values within prescribed tolerances expose repetition, self-similarity, symmetry, trend, etc.

3.4 Data restoration

As stressed already, the restoration is a generalization of the traditional data expansion (decompression) method. The idea is to identify the data that may be omitted during the compression, and then still be restored with a sufficient quality (lossless, near-lossless, lossy) from the context. The samples to be restored can be produced in two ways:

a) in the interior of segments with the setting Interior included = No. If some sample from such a segment is represented by the residuum preferably, then it may be defined as a key sample.
b) Outside of any feature pattern. Equations (7) and (8) only require that all the samples, represented by residuals in *R,* must be present in at least (exactly indeed) one of the feature patterns, which are not allowed to overlap. However, the samples from *I* without the corresponding residuals in *R* need not be addressed by any feature.

The restoration method is defined in the default configuration file or in the compressed file header. In the case a), the *f.prediction.function* should be used instead, if defined.

4 Lossless, near-lossless and lossy data compression

In all three data compression modes, lossless entropy coding is used. This means that all errors, typical for lossy and near-lossless mode, are produced in pre-processing, i.e. feature detection, feature selection, and residual determination processes. Data restoration is an evident source of errors, but not the only one. Compression parameters (defined in the default configuration file or in the compressed file header) also define, which residuals may be further omitted (and then interpolated from the remaining ones) or they can introduce requantization with fewer bits for an individual residual.

In the **lossless** data compression mode, $I' = I$, i.e.

$$s'_{i,j,k} = s_{i,j,k},\ \ 0 \le i < resX,\ 0 \le j < resY,\ 0 \le k < resZ. \tag{9}$$

In the **near-lossless** mode, the error is locally controlled, i.e.

$$\left|s_{i,j,k} - s'_{i,j,k}\right| < \varepsilon, 0 \le i < resX,\ 0 \le j < resY,\ 0 \le k < resZ, \tag{10}$$

where ε is a user-defined local error threshold.

In the **lossy** mode, the error is globally controlled, i.e.

$$\frac{1}{n}\sum_{i=0}^{n-1}\left|s_{i,j,k} - s'_{i,j,k}\right| < \varepsilon, 0 \le i < resX, \qquad 0 \le j < resY, \qquad 0 \le k < resZ, \tag{11}$$

where ε is a user-defined cummulative error threshold, and $n = resX \cdot resY \cdot resZ$.

In the case of multiple attributes, the difference $\left|s_{i,j,k} - s'_{i,j,k}\right|$ in Eq. (8) and Eq. (9) can be computed as the Euclidean distance (Eq. 12), when all h components have nearly the same mean values E and nearly the same standard deviations SD In both, I and I'. Otherwise, the weighted Euclidean distance (Eq. 13) may be used, where w_h is the inverse variance ($\frac{1}{SD_h^2}$) of the h-th component of the samples in both, I and I'.

$$\left|s_{i,j,k} - s'_{i,j,k}\right| = \sqrt{\sum_{h=0}^{c-1}({}^{(h)}s_{i,j,k} - {}^{(h)}s'_{i,j,k})^2} \tag{12}$$

$$\left|s_{i,j,k} - s'_{i,j,k}\right| = \sqrt{\sum_{h=0}^{c-1} w_h \cdot ({}^{(h)}s_{i,j,k} - {}^{(h)}s'_{i,j,k})^2} \tag{13}$$

If the variances and eventually the mean values of different sample components and/or streams I and I' differ, the generalized formula (14) can be used.

$$\left|s_{i,j,k} - s'_{i,j,k}\right| = \sqrt{\sum_{h=0}^{c-1}(\frac{{}^{(h)}s_{i,j,k} - E_h}{SD_h} - \frac{{}^{(h)}s'_{i,j,k} - E'_h}{SD'_h})^2} \tag{14}$$

Note that the mean value E_h for the h-th component of the samples in both, I and I', were previously omitted in (13), as $(({}^{(h)}s_{i,j,k} - E_h) - ({}^{(h)}s'_{i,j,k} - E_h)) = ({}^{(h)}s_{i,j,k} - {}^{(h)}s'_{i,j,k})$

Finally, if the variances and mean values are not known (or the computation takes too much time), the equations (10) and (11) may be used separately for each attribute, resulting in (15) and (16), respectively. Note that the error tresholds ε_h for different attibutes may differ from each other.

$$\left|{}^{(h)}s_{i,j,k} - {}^{(h)}s'_{i,j,k}\right| < \varepsilon_h, 0 \le i < resX,\ 0 \le j < resY,\ 0 \le k < resZ, 0 \le h < c \tag{15}$$

$$\frac{1}{n}\left|{}^{(h)}s_{i,j,k} - {}^{(h)}s'_{i,j,k}\right| < \varepsilon_h, 0 \le i < resX, \qquad 0 \le j < resY, 0 \le k < resZ, 0 \le h < c \tag{16}$$