TPC Data Compression

Jens Berger [a] Ulrich Frankenfeld [b] Volker Lindenstruth [c]
Patrick Plamper [d] Dieter Röhrich [b] Erich Schäfer [e]
Markus W. Schulz [c] Timm M. Steinbeck [c] Reinhard Stock [a]
Kolja Sulimma [d] Anders Vestbø [b] Arne Wiebalck [c,*]

[a] *Department of Physics, University of Frankfurt, Germany*

[b] *Department of Physics, University of Bergen, Norway*

[c] *Kirchhoff-Institute for Physics, University of Heidelberg, Germany*

[d] *Institute for Computer Science, University of Frankfurt, Germany*

[e] *Max-Planck-Institute for Physics, Munich, Germany*

Abstract

In the collisions of ultra-relativistic heavy ions in fixed-target and collider experiments, multiplicities of several ten thousand charged particles are generated. The main devices for tracking and particle identification are large-volume tracking detectors (TPCs) producing raw event sizes in excess of 100 MBytes per event. With increasing data rates, storage becomes the main limiting factor in such experiments and, therefore, it is essential to represent the data in a way that is as concise as possible. In this paper, we present several compression schemes, such as entropy encoding, modified vector quantization, and data modeling techniques applied on real data from the CERN SPS experiment NA49 and on simulated data from the future CERN LHC experiment ALICE.

Key words: Data Compression, Vector Quantization, Online Tracking

*

University of Heidelberg
Kirchhoff-Institute for Physics, Technical Computer Science
Arne Wiebalck
Schröderstr. 90
D-69120 Heidelberg
Phone: 00 49 6221 54 4362
Fax : 00 49 6221 54 4345

Email address: `wiebalck@kip.uni-heidelberg.de` (Arne Wiebalck).

Preprint submitted to Elsevier Science *21 January 2002*

1 Introduction

In relativistic heavy-ion collisions at AGS, SPS, RHIC and LHC energies, charged particle multiplicities range from several hundred to approximately 60000. It has been proven that large-volume tracking devices that provide three-dimensional spatial information, such as Time Projection Chambers (TPCs) [1,2], have the capacity to handle such high multiplicities and high-track densities (e.g., EOS [3], NA49 [4], STAR [5], ALICE [6]).

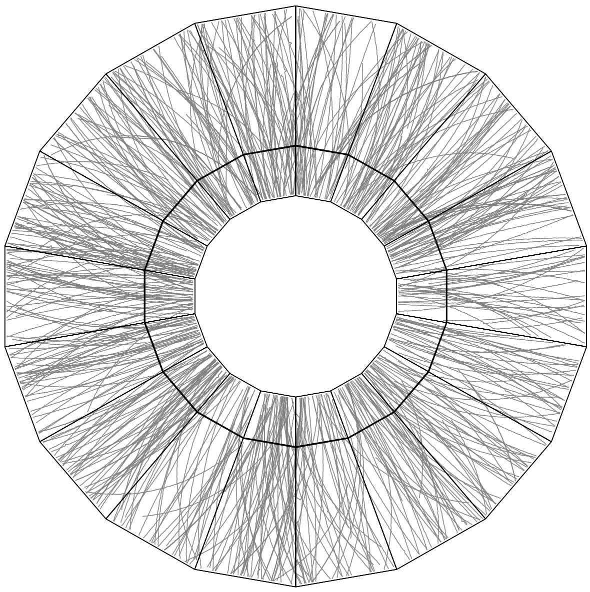

Fig. 1. Event display of a full event, generated with the ALICE fast simulator using the HIJING parametrization ($dN_{ch}/dy = 8000$). Only the projection of the tracks in the η-range from 0 to 0.1 is shown.

The aforementioned detectors are operating at event rates of up to a few hundred Hertz. The increasing integration density of particle detector readout electronics is driving up the number of channels producing raw data. The expected data rate of the ALICE TPC of about 13 GBytes/sec after zero suppression may serve as an example for the resulting data rates. Therefore, advanced data compression techniques are becoming increasingly important. Two data sets, real data taken by the NA49 experiment at the SPS and simulated data for the ALICE experiment at the LHC, will be discussed.

2 The Experiments

2.1 NA49

The current CERN SPS experiment, NA49, focuses on collisions of ^{207}Pb-projectiles with an incident energy of 160 GeV per nucleon with a lead target. The main detectors are four large-volume tracking detectors: The two Vertex Time Projection Chambers (VTPCs) inside two 1.5 Tesla super-conducting magnets, and the two large Main Time Projection Chambers (MTPCs) positioned in a field-free region downstream of the magnets on both sides of the beam [4]. These detectors record the trajectories of charged particles from which the particle momenta are determined. In a central Pb-Pb collision, approximately 1600 charged particles are produced of which approximately 1300 are detected in the TPCs.

The tracking detector's sensitive volume of about 36 m^3 is read out by approximately 182000 8 bit ADC-channels times 512 time bins representing $9 \cdot 10^7$ volume elements. The 182000 pads are arranged orthogonal to the beam direction in 234 rows (72 rows per VTPC and 90 rows per MTPC). Since the occupancy in the TPC for a central Pb-Pb collision is about 10%, most of the data will be noise centered around 0 after pedestal subtraction.

A typical pad signal is shown in Fig. 2. After shaping by the readout electronics, the signal shows a good Gaussian symmetrization with a FWHM of about three time bins [4].

The compressed raw data volume remaining after zero suppression is about 10 MBytes per event [7], resulting in 10 TBytes of data for a typical run of 10^6 events.

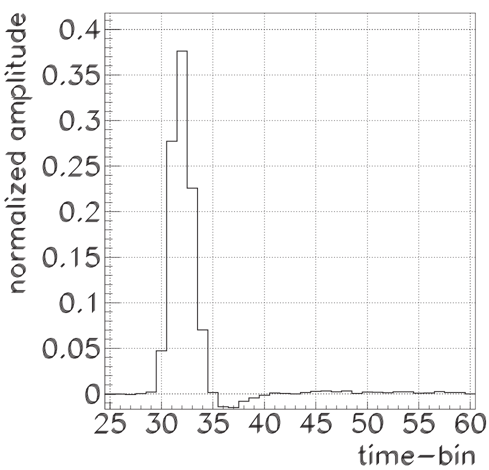

Fig. 2. Time response of the NA49 electronics.

ALICE (A Large Ion Collider Experiment) [8] at the Large Hadron Collider (LHC) to be built at CERN will start to take data in 2006. Lead ions will collide here at energies of up to 2.76 TeV per nucleon. The main subdetector in ALICE for tracking and particle identification is the cylindrical Time Projection Chamber [6]. The expected maximum multiplicity during Pb-Pb mode of $dN_{ch}/dy = 8000$ will result in up to 20000 charged primary and secondary tracks in this detector. The occupancy will range between 15% at the outermost and 40% at the innermost radius.

The number of ALICE TPC readout channels will be 570000, and the number of time bins for each pad is 512. Taking into account the 10-bit ADC dynamic range, the event size directly at the detector readout exceeds 350 MBytes. Before compression algorithms as discussed in this paper are applied, the amount of data is reduced in three steps as follows.
The ADC conversion gain is typically chosen such that σ_{noise} corresponds to 1 ADC count [6]. The TPC's dynamic range corresponds to 10 bits. The resulting *relative* accuracy increases with the ADC values and is not required for the upper part of the dynamic range. Compressing the ADC values nonlinearly from 10 to 8 bit leads to a *constant relative* accuracy over the whole dynamic range and reduction of the event size down to 290 MBytes. Fig. 3 shows a plot of the used conversion table [11].
Since it is problematic to resolve individual tracks that have a low p_t and cross the TPC under small angles relative to the beam pipe, a 45-degree cone is cut out of the data, resulting in the rejection of all particles which are not in the geometrical acceptance of the outer detectors, such as the Transition Radiation Detector (TRD) or the Time of Flight Detector (TOF). This cutout reduces the data volume further by 40%.

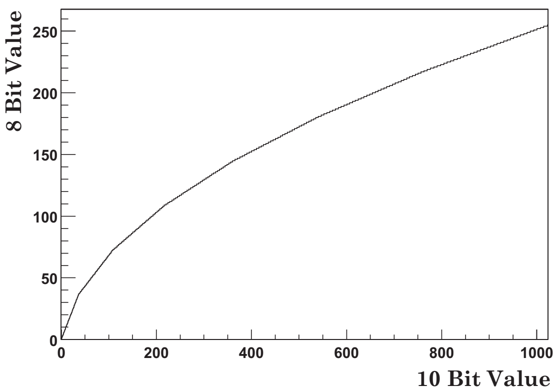

Fig. 3. Plot of the 10-to-8 bit conversion table.

The 10 to 8 bit conversion, the 45-degree cone cutout, and finally zero suppression (i.e., pedestal subtraction, one-dimensional cluster-finding in time direction and run-length encoding), reduce the raw event size of 350 MBytes to about 66 MBytes $\pm$ 15%, including 10% coding overhead.

At the maximum readout frequency of 200 Hertz, a resulting bandwidth of 13 GBytes/sec has to be handled. The volume of real data to be archived is expected to be about 2.7 PBytes per year. Considering the estimated magnetic tape costs in the first year of design luminosity of 0.6 EURO/GByte [9], the costs for taping will be approximately 1600 kEURO per year.

3 Zero Suppression and Data Coding

3.1 Zero Suppression

The most important compression technique for TPC data is to detect the hits and discard the noise in between by replacing it with zeroes. This so-called zero suppression is performed in the front-end electronics by pedestal subtraction (a threshold operation) and one-dimensional hit-finding in time direction. The long zero sequences can then be compressed by run-length encoding [10]. The idea of run-length encoding (RLE) is to replace a sequence of identical symbols by an escape character or tag, normally the symbol itself, and the length of the sequence. In TPC data, the only datum for which we expect a significant number of longer sequences is zero, and, therefore, RLE is only performed on zeroes. This is equivalent to storing only the hits and their positions. The thresholding and hit-finding are lossy techniques, which could lead to a loss of small clusters or tails of clusters, while run-length encoding is a lossless technique.

3.1.1 Zero Suppression in the NA49 TPC

In the NA49 experiment, the detection of hits is done by digital signal processors. Any two successive time bins above a threshold of typically three times the sigma of the noise are called a hit. The position of each cluster spread over four time bins is found by center-of-gravity formation over the charge distributions. A typical charge cluster is shown in Fig. 4. Applying this technique, a compression factor of 10 is achieved in agreement with the occupancy of the TPC.

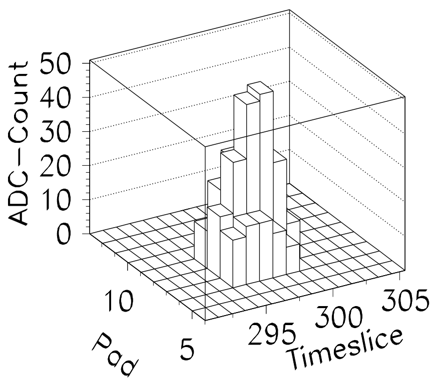

Fig. 4. Two-dimensional charge cluster after zero suppression.

3.1.2 Zero Suppression in the ALICE TPC

Zero suppression for ALICE TPC data is done by the ALTRO (ALice Tpc ReadOut) ASIC. This custom CMOS chip contains the circuitry to perform tail cancellation, pedestal subtraction, zero suppression, formatting and buffering [6]:

"The basic pulse detection scheme is based on the rejection of samples with a value smaller than a constant decision level (threshold). When a sample is found above the threshold, it is considered to be the start of a pulse. To reduce the noise sensitivity, a glitch filter checks for a consecutive number of samples above the threshold, confirming the existence of a real pulse. The minimum sequence of samples above the threshold that defines a pulse can vary from one to three. In order to keep enough information for further feature extraction, the complete pulse shape must be recorded. Therefore, a sequence of samples (pre-samples) before the signal overcoming the threshold and a sequence of samples (post-samples) after the signal returning below the threshold can be recorded. The number of pre- and post-samples can vary independently in the range between zero and four."[6]

3.2 Coding of Raw Data

In the remaining document, we will assume that the detection of hits consisting of one-dimensional cluster sequences and the zero suppression has already been performed and we will focus on how to represent sequences of hits. We use the following abbreviations:

x_i the position of hit i.
l_i the length of hit i.
$z_i = x_i - (x_{i-1} + l_{i-1})$ is the number of zeros immediately preceding hit i.
$A_i(0)$,..., $A_i(l_i - 1)$ the ADC values of hit i.
$p(A)$ the probability of ADC value A in the data stream.

These abbreviations will be used without the indices when discussing an arbitrary hit.

3.2.1 NA49

Run-Length Encoding

The problem with the data format introduced in Section 3.1 is that single zeroes must be encoded using two bytes in order to distinguish a single one from a sequence of zeroes. Depending on the hit detection parameters, single zeroes can be quite common within hit data. It is more efficient to use a rare value, such as 254 ($p(254) \approx 1/250000$), as the tag to bypass this coding weakness. If one encounters a 254 in the input data, it can either be changed to 255 or represented by a special sequence as 254,0. A hit in this data format looks like: $254, z-1, A[0], \ldots, A[l-1]$. This data format reduces the size of a full multiplicity event to about 95% relative to the original NA49 data format. [1]

Storage of Hit Position and Length

Another way to avoid the repeating of zeroes within hits is to do zero suppression implicitly by storing the hit position and length followed by the hit data. The hit can include all possible values, including zeroes. The hit position can be given relatively to the previous hit to keep the numbers small. Hit distances and hit lengths in NA49 can be as large as 512 and need 2 bytes. But as long zero sequences do not occur very often and long hits are extremely rare, it is better to use only one byte for this information. Zero sequences longer than 255 can be represented using a dummy hit of length 0 and long hits can be represented by inserting 0 zeroes. A hit in this format looks like: $z, l, A[0], \ldots, A[l-1]$. This data format reduces the data size also to 95%.

If the compression is performed in hardware, it is easier and faster to store the length of the hit after the hit data. In this case, the input data can be processed exactly in the order of appearance and the output data is a linear data stream too. To read data in this format, one has to start at the end and proceed backwards. A hardware implementation of such a cluster finder has been made for the STAR experiment [12].

3.2.2 ALICE

Before applying compression techniques to simulated TPC data, the format of the stored data is adapted. A threshold of 10 ADC counts is applied to the 8 bit data. This threshold is intentionally set rather high for the generation of the reference data in order to avoid artificially high compression ratios. In addition to the data above the threshold, a pre- and post-history of one time bin is saved for each cluster. The position of such a one-dimensional cluster and its length are saved using 9 and 4 bits, respectively. A 9-bit cluster length covers the whole range of possible time bins for the start bin. Another approach could be to store only the differences between the start bins of adjacent hits, but, as mentioned below, separate Huffman trees for the data and header information

[1] Compression ratios given in this document are calculated as $\frac{\textit{Compressed Size}}{\textit{Original Size}} \cdot 100$ [%].

are built and, thus, the difference in compression should be negligible. A 4-bit cluster length would mean a tradeoff between the needed space for encoding the length and the distribution of the length of one-dimensional clusters in the sample data. This format necessitates the splitting of clusters longer than 15 time bins. If the prehistory of a hit directly follows the post-history of the preceding hit, these two hits are merged to one cluster and the header information is saved only once. Fig. 5 illustrates this approach. Due to the threshold, this preparation reduces the data size to about 88% relative to the zero suppressed run-length encoded 8 bit raw event size. This data format corresponds to the storage of hit position and length as discussed already for NA49.

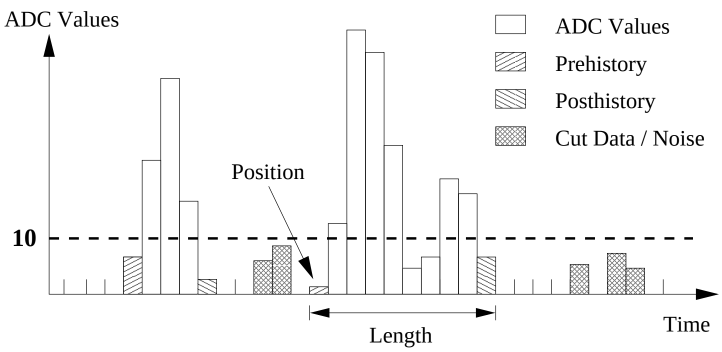

Fig. 5. Illustration of the used data format. In general, hit information is stored using the position of the prehistory bin and the length including the posthistory bin, e.g. for the left hit in the figure. If the prehistory bin of a hit is the immediate successor of the posthistory bin of the preceding hit, the two hits are regarded as one large hit, e.g. the large hit in illustrated in the figure above. In that case the header information is stored only once, although two successive bins are below the threshold.

4 Lossless Data Compression

4.1 Entropy Coding by Variable Length Coding

The improvements of the aforementioned data formats primarily concern the performance and complexity of the compression code, whereas the improvement in data size is rather small.

One crucial observation is that the ADC values are not equally probable, which can mean further reduction in data size. Small ADC values occur often in the data stream but larger ones are rare (see Fig. 6). The distribution is approximately exponential. The expected size of the data can be reduced

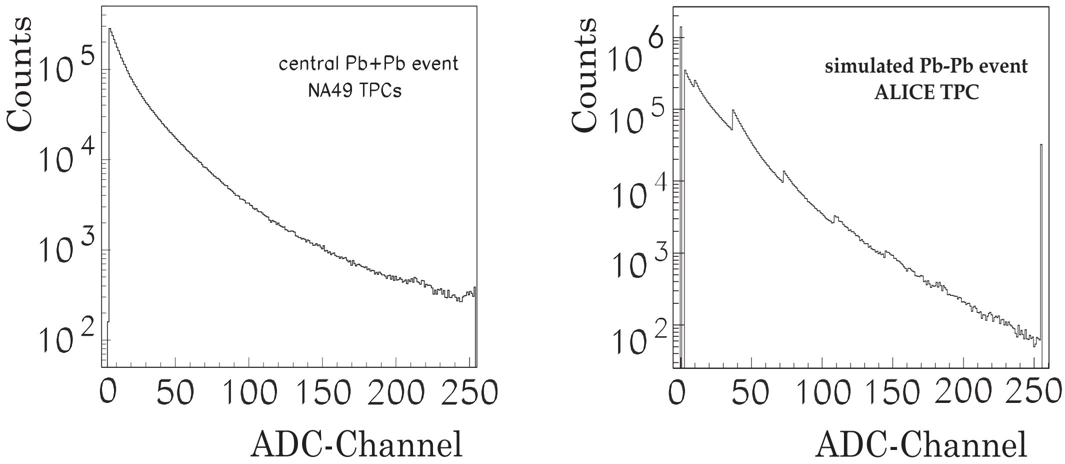

Fig. 6. Left: Distribution of ADC values for a central NA49 Pb-Pb collision. Right: Distribution of ADC values for a simulated ALICE Pb-Pb event. There are two steps in the distribution: the first, at an ADC value of 10, is due to the applied threshold; all other steps are due to the 10 to 8 bit conversion.

if short words are used for frequent values and longer ones for rare values. Another advantage of this technique is that one is no longer restricted to 256 symbols, as one would be with bytes, or powers of two. One can choose as many codes as necessary. This means if one enhances the ADC resolution to 9 bits, the size of the data is no longer doubled, but is increased by 1/8 in the worst case scenario.

There is a theoretical lower limit on the average word size that can be achieved with this strategy. This lower limit is called entropy of the data source and can be computed as:

$$E = -\sum_{A \in \Omega} p(A) \log p(A),$$

where Ω is the set of all possible words that are output by the data source. It can be shown that this limit is tight for stochastic data sources [13]. The difference between the number of bits used to represent a single character and its information entropy is the potential for entropy encoding techniques. The entropy of the simulated ALICE TPC data for example is about 4.96 bits/character, promising compression ratios of 62% in the optimal case. There exist compression techniques that approach this theoretical lower limit, for example, Arithmetic Coding [14]. The idea of Arithmetic Coding is to assign to every symbol in the input stream an interval between 0 and 1, with a size according to the occurrence probability of this symbol. For the first symbol in the input stream, the interval from 0 to 1 is shrunk to the range assigned to this symbol. The new boundaries form a new interval which is shrunk by the next input symbol and its corresponding interval. Thus, the interval gets smaller and smaller and the input stream of symbols is replaced by a single floating point number representing the encoded message. The major drawback of Arithmetic Coding is the number of operations needed for encoding a single symbol. Other algorithms achieve similar results without such intensive computing.

4.1.1 Huffman Coding

An alternative to Arithmetic Encoding is applying Huffman Codes [15], which are easy to implement and achieve good compression results without the need for extensive processing power. The basic idea of assigning short codes to frequent symbols and longer ones to rare symbols is realized by assigning each input symbol to a leaf of a binary tree, the so-called Huffman Tree. Each branch of this tree is either assigned the 0 or the 1 bit, and the path from the root node to the leaf defines the code used for this symbol. To encode data using Huffman Codes, one uses a table containing the bit sequences of the codes for each symbol and their length. The encoder simply concatenates the codes for the various symbols. Decoding a data stream requires more intensive computing since the Huffman tree has to be traversed, but there are sophisticated algorithms to expedite this task [16].

There is also a number of modifications, such as adaptive Huffman Coding, that build and modify a code tree during run-time. For TPC data, this is not necessary as the distribution of the ADC values is very well known in advance. It is sufficient to measure the distribution from time to time, for example when the experiment parameters have changed, and to create a new static code table from that information.

It is important to note that the average size of an event will be reduced by this approach, but there can be events that are larger than before compression. This has to be considered when designing the readout system. Using Huffman Coding on NA49 data, the relative event size can be reduced to 67% and 42% for Pb-Pb and p-p collisions, respectively. For the simulated ALICE data, separate Huffman trees have been built for the ADC values and the header information, i.e., position and length. This approach reduces the data size of the sample to about 65%.

A TPC is obviously not a stochastic data source, as adjacent ADC values are highly correlated. Therefore, it is possible to compress the data to a lower bit rate than the entropy of the ADC values by using this sequence correlation.

4.1.2 Differentiation

This approach is common for lossless graphics compression. It exploits the fact that, especially for graphics, adjacent symbols are similar. In this case, the distribution of the derivative of the input has a lower entropy than the data itself. This is not true for the TPC data of the NA49 or the ALICE experiment. Here, the sampling rate is well matched to the bandwidth of the signal, and, therefore, the slope of the hits is usually quite steep. A hit in this data format looks like: $z, l, A[0], A[1]-A[0], \ldots, A[l-1]-A[l-2]$. The NA49 data size can be reduced to 69% for Pb-Pb and to 54% for p-p events if the differentiated data is subsequently entropy-encoded. For the simulated ALICE events, the data was differentiated and the result encoded using Huffman Coding. Fig. 7 shows the distribution of the differentiated signal. With this approach, the data volume could be reduced to 73%.

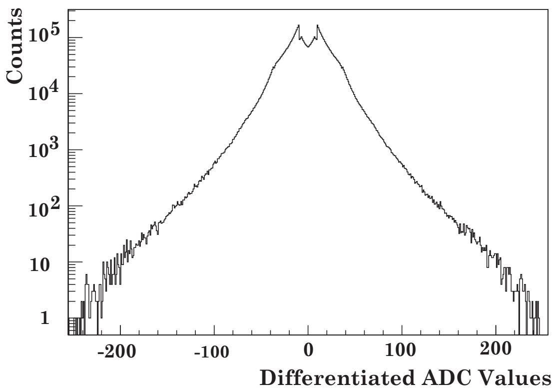

Fig. 7. Distribution of the ADC values in the differentiated signal for ALICE TPC data.

4.1.3 Code Table Coding

A more sophisticated approach, similar to differentiation, is predictive encoding. The idea is to guess the next value of the data stream based on the values previously sent and then send only the difference between this guess and the actual value. The receiver uses the same function as the sender to calculate the guess and to reconstruct the data. If the guess is often close to the actual value, the entropy of the difference is small. Differentiation is a special case wherein the data is always guessed to stay constant.

The problem is to derive a function to guess the next value. One way is to take sample data and build a table with the best guess for each combination of preceding values. Obviously, this is only possible if a small number of preceding bytes is considered (usually not more than two). But with this approach, a different code table can be used depending on the preceding values. The implementation is very fast and easy, but requires a lot of memory.

This approach is feasible because good results are achieved, even if the table depends only on the previous value. In fact, the results did not get better if we made the table dependent on the two preceding values. A hit in this format looks like: $z, l, f(A[0], A[1]), \ldots, f(A[l-2], A[l-1])$. A reduction to 61% for Pb-Pb and 40% for p-p has been measured for NA49 data. The simulated ALICE TPC data could be reduced to about 71%.

4.1.4 Cluster Prototypes

The shapes of the hits before sampling are very similar and depend on only a few parameters. To extract these parameters from the data and store only those parameters and the deviation from what a hit with these parameters is supposed to look like was proposed by H. Beker and M. Schindler in [17].

There are various characteristics for which the hits can be parametrized. Besides the width, the maximum, the integral charge, the position of the maximum, or a combination of these, are possible choices.

These approaches suffer from two effects. The first one is that the hits are usually very steep and short. They reach their maximum within one or two time buckets and decrease at a similar slope. As the position of the hit relative to the sampling clock is random, the same hit can produce quite different data depending on its position. In particular this effect introduces a noise proportional to the slope. This increases the entropy of the data. The second effect is that the clusters are very short, typically two bytes for proton and six bytes for lead data. This means that the overhead used to store the parameters is a significant amount of data.

Cluster prototyping is not a suitable compression technique for NA49 or ALICE TPC data, but it might get good results for experiments with larger hits or higher sampling frequencies compared to the hit length. In particular, the fact that the compression results get better with higher sampling frequencies compensates part of the increased data rate and might therefore affect the decision for a particular frequency.

5 Lossy vs. Lossless Signal Processing in TPCs

Prior to discussion of possible lossy compression techniques, the sources of distortion of the ideal signal shall be outlined first. The main tasks of TPC subdetectors in heavy-ion experiments are reliable particle identification by energy loss measurement and momentum determination, requiring track reconstruction in a high multiplicity environment. Several aspects complicate the task of an exact event reconstruction.

Prior to entering the TPC's sensitive volume interactions with the surrounding material may change the original particle track by multiple scattering and secondary particle production. For this reason, the material budget of the TPC has to be kept as low as possible. The energy loss by ionization inside the volume is distributed statistically following approximately the Bethe-Bloch formula. A more precise model for energy loss is given by W. W. Allison and J. H. Cobb [18]. During the drift to the end-caps of the TPC, the produced electron clouds are broadened by diffusion, which is determined by the transverse diffusion constant D_T. Although this effect is minimized by the magnetic field in drift direction forcing the electrons on a helix around drift direction, diffusion influences the position resolution of the reconstructed space points. Exponential fluctuations in the gas amplification of single electrons at the readout chambers enhance the delocalization caused by diffusion by a factor $\sqrt{2}$ [6]. The image charge induced on the ALICE TPC readout pads is processed by the front-end electronics, consisting of a charge sensitive preamplifier/shaper circuit, a 10-bit ADC, and a digital circuitry performing tail cancellation, pedestal subtraction and zero suppression. At this stage, several sources contribute to the distortion of the original signal, for example, the non-linearity of the preamplifier, and the differential and integral non-linearities of the ADC. Tail cancellation, pedestal subtraction and zero suppression are all lossy techniques manipulating the original raw data, but not the physics information. The conversion of the 10 bit data to 8 bit is a data volume reduction with information loss, but as indicated in chapter 2.2, this conversion leads to a constant relative accuracy for the single ADC values. The last component in this data processing chain is the computer performing lossless or lossy data compression to handle the data stream.

In order to discuss the question of whether to use lossy compression techniques or not, the readout as a whole system should be regarded as outlined here. Although attempts are made to keep all of the aforementioned effects at a minimum, some loss of information during readout is inevitable. The answer to the question of applicability of lossy techniques will be provided by the impact that lossy compression schemes have on the physical observables, and by weighing the tradeoff between these impacts and the costs for data storage.

6 Lossy Data Compression

There is some gain in using lossless techniques, such as simple Huffman Encoding. Better results can be achieved when very small noise-like changes of the data are tolerated. This leads to the referenced lossy compression schemes. One kind of lossy compression, the zero suppression together with a run-length encoder, is already in use in the NA49 readout system and will also be used for ALICE. The following sections focus on the further reduction of the remaining data.

Quantization of the data can be seen as a method of lossy compression. One quantization step takes place already at the input ADC, where the continuous analogue waveform is sampled to produce digital values of limited resolution (e.g., 8 bit in the NA49 experiment). However, given the adjustment of the sampling frequency to the preamplifier's cutoff frequency and the ADC resolution to the inherent noise (lossy analogue signal processing), these losses are small. Scalar quantization is used, meaning that every single voltage is rendered as one digital number and there is no dependency between a sample and its neighbors. A very simple approach to lossy compression would be to quantize the digital data further, say from 8 bit down to 7 bit, by discarding the least significant bit from every data byte. Seen alone, this would merely reduce the data to 87.5% of its size.

6.1 Vector Quantization

Vector Quantization [19] is another, but more sophisticated, type of quantization. Here, statistical dependencies between successive data samples are exploited. Instead of quantizing data samples independently, several samples are grouped together to form a vector of data samples. This vector is then compared to entries in a codebook of vectors and the index of the best matching vector from this codebook is stored, wherein the optimum would be the shortest distance using Euclidian metrics, or any other metric system that is best suited for the application. To be optimal, the codebook has to be trained on the statistical properties of typical input data. This is usually done before the codebook is used with an algorithm known as the modified LBG-Algorithm [20]. The adapted codebook then stays unmodified throughout the actual quantization.

6.2 Residual Encoding

Though this approach offers very low bit rate, it is obvious that there is almost no possibility for the vector quantizer to change behavior. Since the codebook is preproduced, only the given vectors are available to represent the output data. Even if the codebook was well adapted to the data, sometimes large quantization errors could occur, see Fig. 8. To prevent this, we need to store

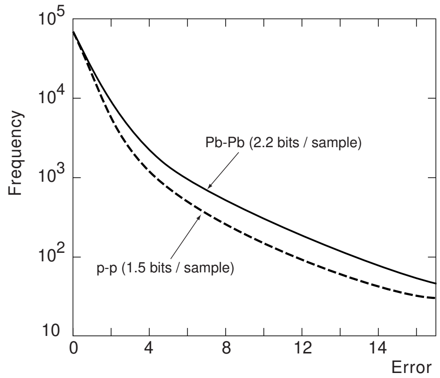

Fig. 8. Error distribution of vector quantization for NA49 data. The error is measured as the sum of the differences of individual timebins betweeen data and code vector. The curves are fitted to the corresponding histograms of errors for Pb-Pb and p-p data, respectively.

the differences between the input data and the selected codebook entry, the so-called residuals. These residuals are then quantized and entropy encoded to achieve an even lower bit rate. A quantization of the residuals is especially effective since the distribution of errors between codebook entry and input data is very steep. If the codebook is sufficiently trained, the vector quantizer will find codevectors with small deviations from the data vectors, so the residual encoder will mostly get values around zero. When these values are quantized, even more values are mapped to zero. Here, arithmetic compression is the method of choice. Huffman Coding would merely reduce the zero entries to a single bit despite their high probability. The arithmetic coder has no such restrictions; the zeroes are mapped to much less than a bit, normally less than 1/4 of a bit.

6.3 Experimental Results

6.3.1 NA49

The algorithms described were applied to a set of TPC events from the NA49 Pb-Pb collisions as well as to some p-p data. The compression results are summarized in Table 1. Fig. 9 illustrates the contributions for lossless vector

Type of Encoder	Entropy [bit/sample]	Rel. Event Size in %
Zero Suppressed Raw Data	8	100
Huffman	5.8	73
Differentiation	5.5	69
Code Table Coding	4.9	61
RVQ3 Lossless	4.8	60
VQ3 Contribution	2.3	
Residual Value Contribution	2.5	
RVQ3 Lossy (error 1 ADC value)	3.8	48
VQ3 Contribution	2.3	
Residual Value Contribution	1.5	
VQ3 Lossy	2.3	29

Table 1: Compression performance of algorithms for NA49 Pb-Pb data. The entropy is given as the average number of bits used to encode a sample.

quantization on NA49 Pb-Pb data. Compression factors for p-p data, which have a much lower occupancy than heavy-ion data, are higher by 30%-40%. In our experiments, we used a vector quantizer of length three (VQ3), so that the majority of hits, which have a length of up to six time bins, are modeled by two vectors. The size of the codebook is 256 entries. This leads to a data rate of 2 bits/sample for the vector quantizer alone. The algorithm RVQ3 is a vector quantizer using vectors of length three with quantized residuals. Allowing an absolute error of one in the residual quantization, the change in

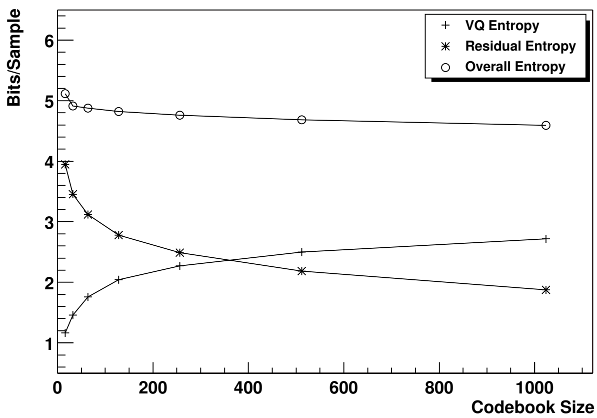

Fig. 9. Entropy contributions for lossless vector quantization on NA49 data.

the number of clusters was less than 10^{-4}, and no change in the number of tracks was observed.

6.3.2 ALICE

For simulated ALICE TPC data, a slightly different approach has been chosen. The sizes of accepted deviations between a data vector and its corresponding codebook entry have to be adjusted to the effects these changes have on the measured physical quantities.

Since changes of the ADC values directly affect the total charge and pulse shape and thus the position of a one-dimensional cluster, the mapping function reflects these two aspects. Therefore, for successful mapping, a pair of data and codebook vectors must fulfill two conditions:

$$\frac{\left|\sum_{i=1}^{dim} C_{Q_i} - \sum_{i=1}^{dim} D_{Q_i}\right|}{\sum_{i=1}^{dim} D_{Q_i}} \leq thresh$$

$$\forall i \quad \frac{|C_{Q_i} - D_{Q_i}|}{D_{Q_i}} \leq thresh$$

In these relations, C_{Q_i} and D_{Q_i} denote the ADC values in a codebook vector and a data vector, respectively, while *thresh* is the value of the threshold to adjust. The first condition ensures the total relative charge difference, i.e., the difference in the sums of the ADC values of data and code vector, to be under the threshold. This condition is relevant for dE/dx. The second condition ensures that the shape of a pulse, and its centroid, is not significantly affected. Therefore, the relative differences between the components of the data and code vectors have to be smaller than the specified threshold. For simplification, the chosen threshold for both conditions is the same. Additionally, choosing different values can either affect the centroid of a cluster (one ADC value holds all of the total charge difference) or the maximum tolerated charge difference can never be reached. The higher the allowed threshold, the better the mapping probability and subsequent compression, but, simultaneously, the data may become more impaired.

The cost for encoding data using vector quantization is higher than for Huffman due to the more complicated mapping, but it can be significantly reduced using sophisticated search algorithms. Decoding is done by a simple lookup, which is considerably faster than Huffman decoding.

In order to find a suitable set of the parameters, vector dimension, and codebook size, these parameters have been tested in several combinations as shown in Table 2. For all parameter sets, the threshold has been varied from 0% to 60%. Fig. 10 illustrates how the compression depends on the tolerated deviation for a subset of the combinations in Table 2.

Vector Dimension \ Code Book Size	32	64	128	256	512	1024
2	x	x	x	x		
3		x	x	x	x	
4			x	x	x	x

Table 2: Parameter Sets tested for Vector quantization on simulated ALICE data.

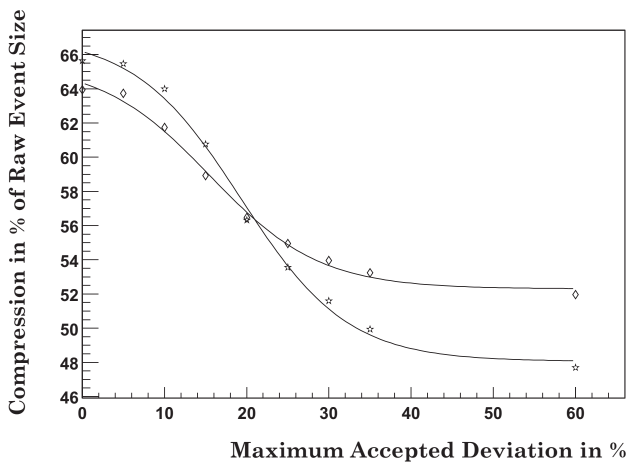

Fig. 10. Compression ratio versus maximum accepted charge difference for a codebook of dimension 2 and with 64 ($\star$) and 128 ($\diamond$) entries.

With small deviations, the larger codebook achieves better compression. This is due to the better probability of finding a matching entry from the larger set of vectors. The larger the accepted difference, the better the compression with the smaller codebook. This is because a smaller codebook requires less bits for storing the index of an entry and, with an increasing threshold, more samples are actually mapped to vectors. The achievable compression ratio with this technique is about 50%.

Fig. 11 shows the effects on momentum resolution and dE/dx resolution in the sample data. The achieved value for relative momentum resolution of 2.65% is slightly above the value for σ_{p_t}/p_t expected in [6], but this is due to the fact that instead of a Gaussian fit, the RMS is used here. This is done as the data does not have a pure Gaussian distribution. Using a Gaussian fit, values for σ_{p_t}/p_t comparable with the ones in [6] are achieved. In the error range, the momentum resolution coincides with the one of the untreated data.

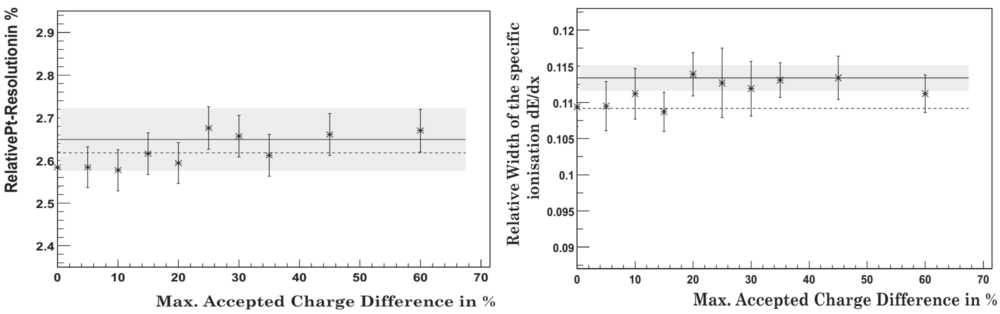

Fig. 11. Left: Effects on momentum resolution. Relative width of p_t resolution averaged over all tracks in a high multiplicity event. Right: Effects on dE/dx resolution: Relative width of the specific ionization. Both plots are for a codebook with 64 entries and a vector dimension of 2. The values for the reference data with the applied threshold (- - -) (see Section 3.2.2), original data (—) and its error (grey area) are also shown.

Also shown in Fig. 11 is the influence of the lossy compression on the relative width of the specific ionization dE/dx. The resolution of about 11% is compatible with the one in [6]. The resolution of the data with the threshold applied is compatible with the resolution of the original data. The resolution of the data with the threshold applied is slightly better due to the fact that the analysis software drops clusters with more than 30 maxima. The used data format tends to split larger clusters in smaller ones and the corresponding space points are thus not lost. This plot also shows no significant influence of the vector quantization.

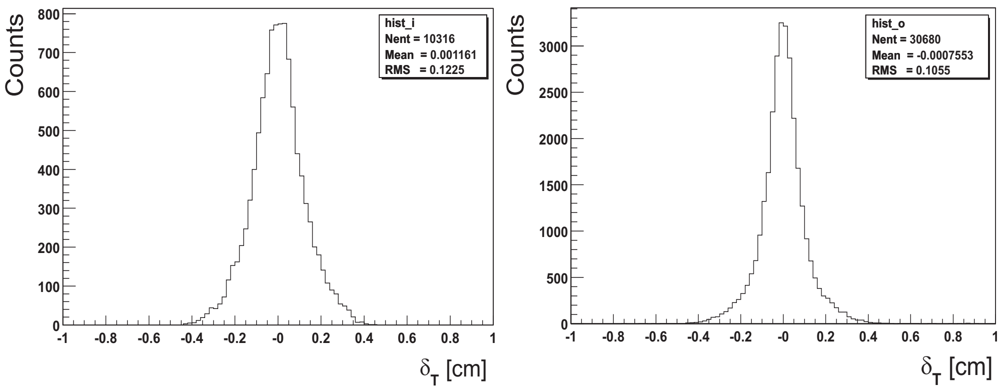

Fig. 12. Distribution of residuals for an inner (left) and an outer (right) sector of the original data.

Another quite sensitive physical variable should be the space point resolution since, by changing the ADC values, the centroid of the corresponding cluster is changed directly. In order to evaluate effects on the space point resolution, transverse and longitudinal residuals, i.e., the distances of space points from the assigned track are measured.

Fig. 12 shows the distribution of the transverse residuals for an inner and outer chamber in the original data. Since the impact of multiple scattering

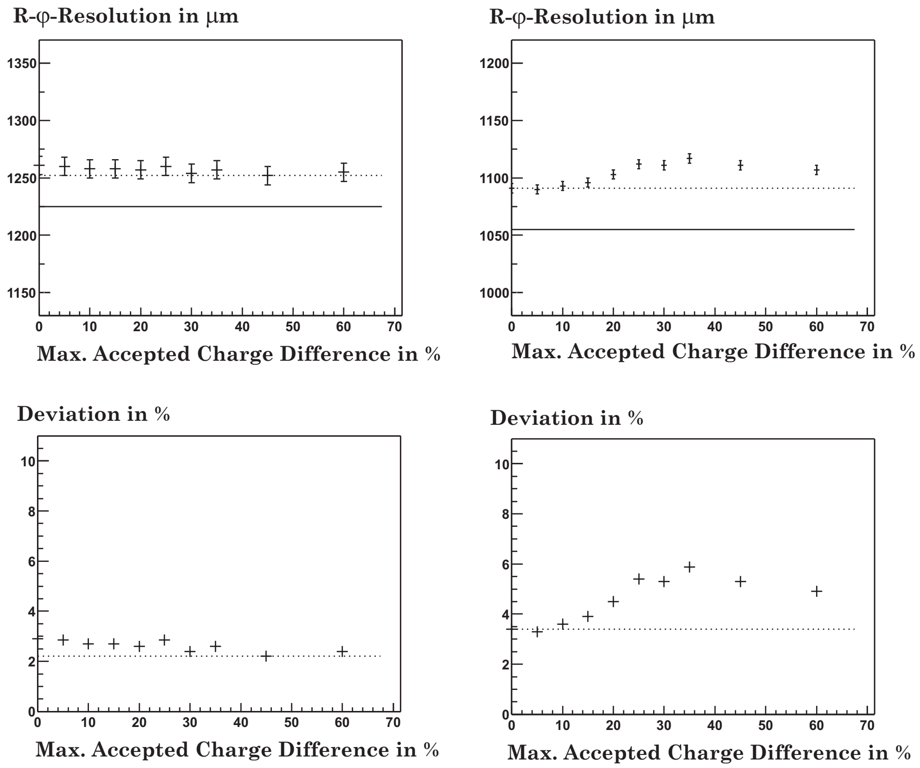

Fig. 13. R-ϕ-resolution versus accepted charge difference. Left: inner chamber. Right: outer chamber. The upper plots show absolute resolution whereas the lower ones shows relative deviation from the resolution in the original data. The straight line denotes the value of the original data, the dotted one the data with the threshold.

on position resolution is smaller for tracks with high momentum, only tracks with a momentum greater than 1 GeV/c are taken into account here. Due to different pad sizes in the outer chamber, the distribution is not normal. On this account, RMS is used again instead of a Gaussian fit. The resolution gained from Fig. 12 is about 1225 μm for an inner and 1055 μm for an outer sector. After applying the threshold mentioned above, the vector quantizer space resolution deteriorates by 30 μm (inner sector) and 40 μm (outer sector), see Fig. 13. For the inner chamber, the deterioration of resolution is mainly due to the applied threshold and not to the vector quantizer, whereas in the outer chamber, starting at a charge deviation of 20%, the influence of the lossy compression is clear to see, although this effect is quite small. So space point resolution is a physical variable sensitive to small changes of ADC values. Resolution in drift direction is not shown here because it is not significantly affected by the threshold, and the effects of vector quantization are comparable to the ones in pad direction. The reason that all impacts on the physical observables are small is shown in Fig. 14; the average accepted error is much smaller than the maximum tolerated error.

Table 3 summarizes the compression performance of the algorithms applied on simulated ALICE TPC data. These compression ratios imply that hit level

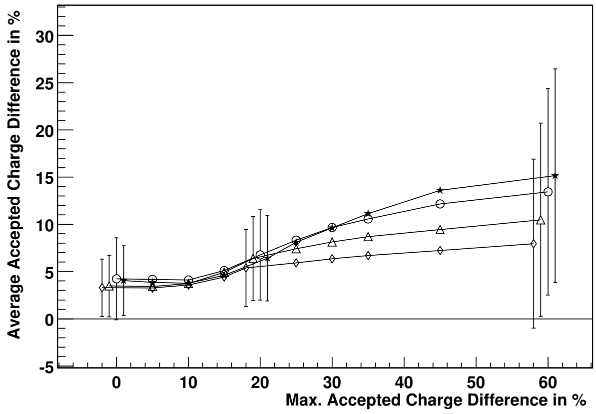

Fig. 14. Average charge difference for different codebook sizes (64($\star$), 128($\circ$), 256($\triangle$) and 512($\diamond$) entries) and a vector dimension of three. For clarity, the dispersion is not shown for every data point, but only at 0%, 20% and 60%, and the errorbars have been displaced against each other. The reason for an average error of $\approx 4\%$ at a maximum accepted error of 0% is due to the fact that one ADC value as the deviation is always accepted since the noise of the front-end electronics is that size.

compression of high occupancy ALICE TPC data is limited to about 50%.

Type of Encoder	Entropy [bit/sample]	Rel. Event Size in %
Zero Suppressed Raw Event Size	8	100
Used Data Format	7	88
UNIX gzip (LZ77)	6.4	80
UNIX compress (LZW)	7.4	92
UNIX pack (static Huffman Coding)	6.4	80
UNIX compact (adaptive Huffman Coding)	6.4	80
Arithmetic Encoding	6.4	80
Huffman Coding (differentiated data, multiple code trees)	5.8	73
Code Table Coding	5.7	71
Huffman Coding (ADC values, multiple code trees)	5.2	65
Vector Quantization	5.1 - 3.8	64-48

Table 3: Compression performance of algorithms on ALICE simulated data. The entropy is given as the average number of bits used to encode a sample. For the test of Arithmetic Coding an implementation from [14] has been used.

7 Hardware Realization

The application of bandwidth reduction techniques in a readout system reduces the needed average bandwidth for the following stages in the system, including interconnect, by a factor roughly proportional to the compression ratio. Therefore, it is beneficial to implement at least the zero suppression and run-length encoding very close to the ADCs on the detector. For the most commonly used algorithms, such as run-length encoding and Huffman Coding, the hardware implementation is greatly simplified by the fact that there is a fixed number of operations that is applied to each data element and the operations are simple, such as table lookups and shifts.

7.1 Zero Suppression and Run-Length Encoding

For this task, several hardware implementations already exist. Most of them provide additional functionality, such as pedestal subtraction, non-linear gain adjustment and possibly digital filters for tail cancellation. For small-scale experiments, the implementation is mostly FPGA-based because of the initial cost that is involved in the development and verification of an ASIC. However, for experiments with several thousand channels, a decision must be made as to the more expensive but flexible FPGA, and the cheaper, lower power ASIC. The STAR cluster-finder ASIC [12] and the ALTRO chip for ALICE [21] are realized as ASICs. One of the authors, Kolja Sulimma, has implemented similar functionality in several types of Xilinx-based FPGAs performing at rates sufficient for the discussed experiments.

7.2 Huffman Encoding

Huffman Coding and decoding in hardware is used in most of the current high-speed tape drives. The process of using a Huffman coder with a static code table is straight forward. It consists of a table lookup, a shift, and a subsequent logical OR that produces a window of the codestream in a register. The register has to be fragmented into words of the size needed by the following transport or processing steps.

We have produced a VHDL-based implementation that has been verified and benchmarked using various Xilinx FPGAs [22]. The achieved rates vary between 115 and 205 MSamples/sec depending on the chosen device. The implementation requires about 10.000 equivalent gates.

The hardware implementation of vector quantization is more complicated due to the search algorithms that have to be performed. This has been an active field of research for the last ten years, especially in the video compression community. These efforts have resulted in hardware implementations, such as the VAMPIRE chip [23].

8 Summary and Outlook

In current heavy-ion physics experiments, the TPC subdetectors produce by far the largest amount of data. The NA49 experiment at the CERN SPS has a typical compressed TPC event size of 10 MBytes, STAR at RHIC has to handle events between 15 and 20 MBytes. The future heavy-ion experiment ALICE, at the planned CERN LHC, will produce typical event sizes of about 65 MBytes. To realize the amount of data to be handled, these event sizes have to be combined with the trigger rates and the duration of the active period of the experiments. The event rate at NA49 is approximately 10 Hertz, STAR runs at 50 to 100 Hertz, and ALICE will record events with rates up to 200 Hertz. The duration of these experiments spans several months each year. For ALICE, the data collected in one year will be approximately 2.7 PByte.

We have investigated commonly used methods of data compression above the canonical baseline methods, such as zero suppression, in order to reduce the amount of TPC data that has to be written to permanent storage: lossless algorithms, such as Huffman Coding, Differentiation or Code Table Coding, as well as lossy Vector Quantization. For the lossy Vector Quantization a detailed study of the impact on the physical data has been made using simulated data of the ALICE TPC. It has been shown that the impact on the physical observables is measurable, but small. The spacepoint resolution emerged to be the most sensitive quantity. For some of the algorithms hardware based implementations have been realized and tested. As shown, all these traditional methods can compress zero suppressed TPC raw data only by factors up to 3.

The techniques investigated in this paper are all based on local modeling of the data since they work on the scale of ADC samples and clusters. However, the parameters that enter the physics analysis have a more global nature. These are especially the track parameters and the particles' integrated charge depositions. Thus, compression methods that exploit models of a higher abstraction level are expected to yield better results. The most promising strategy so far is the online track reconstruction. This technique has been experimentally applied at NA49 and the first results indicate that compression rates in the order of a factor of 10, with only a small impact on data quality, can be achieved [24].

This kind of compression must allow for a subsequent second pass of calibration and distortion corrections, track and vertex finding, fitting and dE/dx analysis. Since the aim of track finding is not to extract physics information, but merely to build a data model that will be used to collect clusters and to code cluster information efficiently, any inefficiencies in track finding, e.g., due to an unprecise track model, will result in an inefficient compression, but not in a loss of clusters. No relevant data are lost.

Summarizing we can state that hit-level data compression schemes can compress TPC raw data by a factor 2 to 3 depending on the inherent structure of the data and the ability of the various techniques to exploit these data characteristics. In order to improve the compression ratios substantially, higher level data abstractions, e.g. cluster and track modeling, have to be considered. Online tracking seems to be a promising option that we will address in future research.

References

[1] D. R. Nygren, LBL Int. Report, Feb. 1974.

[2] D. R. Nygren, Phys. Scripta 23 (1981): 584.

[3] G. Rai et al., IEEE Trans. Nucl. Sci. 37 (1990): 56.

[4] S. Afanasiev et al. (NA49 Collaboration), *The NA49 Large Acceptance Hadron Detector*, Nucl. Instrum. Meth. A430 (1999): 21.

[5] K. H. Ackermann et al. (STAR Collaboration), *The STAR Time Projection Chamber*, Nucl. Phys. A661 (1999): 681c-685c.

[6] The ALICE Collaboration, *Technical Design Report for the Time Projection Chamber (TPC)*, CERN/LHCC 2000-001 (2000).

[7] W. Rauch, IEEE Trans. Nucl. Sci. 41, No. 1 (1994).

[8] The ALICE Collaboration, *Technical Proposal for a Large Ion Collider Experiment at the CERN LHC*, CERN/LHCC 95-71 (1995).

[9] S. Bethke et. al., *Report of the Steering Group of the LHC Computing Review*, CERN/RRB-D 2001-3.

[10] B. Buchanan, Handbook of Data Communications and Network, Boston/ Dordrecht/London: Kluwer Academic Publishers, 1999.

[11] M. Ivanov, GSI Darmstadt. Private Communication.

[12] M. Botlo, M. J. LeVine, R. A. Scheetz, M. W. Schulz, P. Short, J. Woods and D. Crosetto, *The STAR Cluster-Finder ASIC*, IEEE Trans. Nucl. Sci. 45 (1998): 1809.

[13] C. E. Shannon, Bell System Technical Journal, 27 (1948): 379-423.

[14] M. Nelson and J. L. Gailly, The Data Compression Book, New York: M&T Books, 1996.

[15] D. A. Huffman, *A Method for the Construction of Minimum Redundancy Codes*, Proceedings of the IRE, Vol. 40 (9) (1952): 1098-1001, 1952.

[16] B. Hoff, *A High-Speed Static Huffman Decoder*, Dr. Dobb's Journal, No. 271, (Nov. 1997): 56

[17] H. Beker and M. Schindler, *Data Compression on Zero Suppressed High Energy Physics Data*, Alice Internal Note, INT-1996-03.

[18] W. W. Allison and J. H. Cobb, *Relativistic Charged Particle Identification by Energy Loss*, Ann. Rev. Nucl. Part. Sci., 30 (1980): 253.

[19] R. M. Gray, *Vector Quantization*, IEEE ASSP Magazine (Apr. 1984): 4ff.

[20] Y. Linde, A. Buzo and R. M. Gray, *An Algorithm for Vector Quantizer Design*, IEEE Transactions on Communications, Vol.28 (Jan. 1980): 84ff.

[21] L. Musa, ALICE Internal Note, INT-ALICE 99-50.

[22] S. Schössler and T. Jahnke, *Project Work "Huffman Coder" for "Hardware Design using FPGAs"*, Department of Computer Science, University of Frankfurt,http://www.sulimma.de/prak/ss00/projekte/huffman/Huffman.html, 2000.

[23] J. E. Fowler, K. C. Adkins, S. B. Bibyk, and S. C. Ahalt, *Real-Time Video Compression Using Differential Vector Quantization,*, IEEE Transactions on Circuits and Systems for Video Technology, (Feb. 1995).

[24] J. Berger, *Messung von Lepton-Paaren aus Meson-Zerfällen in den Hadronexperimenten NA49 und STAR*, Diploma Thesis, University of Frankfurt 1998.

[25] J. S. Lange *et al.*, *The STAR Level-3 Trigger System,* Nucl. Instrum. Meth. A, 453 (2000): 397.

List of Figures

List of Tables

Type of Encoder	Entropy [bit/sample]	Rel. Event Size in %
Zero Suppressed Raw Data	8	100
Huffman	5.8	73
Differentiation	5.5	69
Code Table Coding	4.9	61
RVQ3 Lossless	4.8	60
VQ3 Contribution	2.3	
Residual Value Contribution	2.5	
RVQ3 Lossy (error 1 ADC value)	3.8	48
VQ3 Contribution	2.3	
Residual Value Contribution	1.5	
VQ3 Lossy	2.3	29

Table 1: Compression performance of algorithms for NA49 Pb-Pb data. The entropy is given as the average number of bits used to encode a sample.

Vector Dimension “ Code Book Size	32	64	128	256	512	1024
2	x	x	x	x		
3		x	x	x	x	
4			x	x	x	x

Table 2: Parameter Sets tested for Vector quantization on simulated ALICE data.

Type of Encoder	Entropy [bit/sample]	Rel. Event Size in %
Zero Suppressed Raw Event Size	8	100
Used Data Format	7	88
UNIX gzip (LZ77)	6.4	80
UNIX compress (LZW)	7.4	92
UNIX pack (static Huffman Coding)	6.4	80
UNIX compact (adaptive Huffman Coding)	6.4	80
Arithmetic Encoding	6.4	80
Huffman Coding (differentiated data, multiple code trees)	5.8	73
Code Table Coding	5.7	71
Huffman Coding (ADC values, multiple code trees)	5.2	65
Vector Quantization	5.1 - 3.8	64-48

Table 3: Compression performance of algorithms on ALICE simulated data. The entropy is given as the average number of bits used to encode a sample. For the test of Arithmetic Coding an implementation from [14] has been used.