Optimal context quantization in lossless compression of image data sequences

Forchhammer, Søren; Wu, X.; Andersen, Jakob Dahl

Published in:
I E E E Transactions on Image Processing

Link to article, DOI:
10.1109/TIP.2003.822613

Publication date:
2004

Document Version
Publisher's PDF, also known as Version of record

Optimal Context Quantization in Lossless Compression of Image Data Sequences

Søren Forchhammer, Xiaolin Wu, and Jakob Dahl Andersen

***Abstract*—In image compression context-based entropy coding is commonly used. A critical issue to the performance of context-based image coding is how to resolve the conflict of a desire for large templates to model high-order statistic dependency of the pixels and the problem of context dilution due to insufficient sample statistics of a given input image. We consider the problem of finding the optimal quantizer Q that quantizes the K-dimensional causal context $C_t = (X_{t-t_1}, X_{t-t_2}, \ldots, X_{t-t_K})$ of a source symbol X_t into one of a set of conditioning states. The optimality of context quantization is defined to be the minimum static or minimum adaptive code length of given a data set. For a binary source alphabet an optimal context quantizer can be computed exactly by a fast dynamic programming algorithm. Faster approximation solutions are also proposed. In case of m-ary source alphabet a random variable can be decomposed into a sequence of binary decisions, each of which is coded using optimal context quantization designed for the corresponding binary random variable. This optimized coding scheme is applied to digital maps and α-plane sequences. The proposed optimal context quantization technique can also be used to establish a lower bound on the achievable code length, and hence is a useful tool to evaluate the performance of existing heuristic context quantizers.**

***Index Terms*—Adaptive code length, context modeling, context quantization, dynamic programming, entropy coding.**

I. Introduction

THIS paper considers issues of designing efficient entropy codes for image compression. A key problem in sequential source coding of a discrete random sequence $X_0, X_1, X_2, \ldots$ is the estimation of conditional probabilities $P(X_t|X^{t-1})$, where X^{t-1} denotes $X_0, X_1, \ldots, X_{t-1}$, the prefix of X_t. The estimation is based on a model. Given a class of models, the number of parameters must be carefully selected. Algorithm *Context* [1] dynamically selects a variable-order subset of the past samples, X^{t-1}, called the context, C_t. The algorithm organizes the contexts in a tree and it can be shown to be universal. Many practical source coders choose *a priori* a model with fixed complexity, based on domain knowledge such as correlation structure and typical data length, and estimate only the model parameters. For instance, the JBIG standard for binary image compression uses the contexts of a fixed size template. The actual coding is implemented by sequentially applying arithmetic coding based on the estimated conditional probabilities. Estimating the conditional probabilities $P(X_t|C_t)$ directly using count statistics from past samples can face severe context dilution problems if the number of symbols in the context is large, or if the symbol alphabet is large. To avoid this problem, the state-of-the-art lossless image compression algorithm CALIC [2] and the JPEG 2000 entropy coding algorithm EBCOT [3] quantize the context, C_t into a relatively small number of conditioning states, and estimate $P(X_t|Q(C_t))$ instead, where Q is a context quantizer. Also a state-of-the-art compression scheme, PWC (Piecewise-constant image model) [4] for palette and graphic images, applies heuristic context quantization. These heuristic context quantizers have produced some of the best performing image compression algorithms, despite the fact that they are not strictly universal. A pivotal issue for these source coders is the design of the context quantizer Q.

Optimal context quantization for minimum code length was formulated and treated in [2], [5]. Off-line optimal context quantizer design algorithms were proposed to minimize the static code length for binary representations. These algorithms attempt to find a context quantizer Q with a given number of conditioning states that minimizes the conditional entropy $H(Y_t|Q(C_t))$, for a particular bit Y_t in the binary representation of X_t. Only quantizers Q of a linear structure, imposed by a projection, were considered in that work. The globally optimal context quantizer for minimum conditional entropy in the original high-dimensional context space was presented in [6] given the conditional probability density functions. It turns out that this problem is one of optimal vector quantization design with respect to the Kullback-Leibler distance. Such a vector quantizer was called minimum conditional entropy context quantizer (MCECQ). For a given data set, designing the context quantizer by minimizing the adaptive code length was also introduced in [7] and [8]. Encouraged by the success of the 'heuristic quantizers' we focus on this more general class of context quantizers, although they are not strictly universal as e.g., the tree structured algorithm Context [1]. In this paper we study the issues in estimating the conditional probabilities and applying the context quantization (CQ) techniques to a given data set, as well as fast dynamic programming algorithms to design optimal context quantizers. The fast algorithms are based on (decomposition in) binary decisions. The complexity of the algorithms is also analyzed. In order to verify the efficacy of optimal context quantization experiments were carried out applying the proposed CQ schemes to digital maps and the α-plane sequences of MPEG4. The focus of these experiments

Manuscript received January 21, 2003; revised September 15, 2003. This work was carried out while the second author was a Visiting Professor at the Technical University of Denmark.

S. Forchhammer and J. D. Andersen are with the Research Center COM, Technical University of Denmark, DK-2800 Lyngby, Denmark.

X. Wu is with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada L8G 4M2 (e-mail: sf@com.dtu.dk; xwu@mail.ece.mcmaster.ca; jda@com.dtu.dk).

Digital Object Identifier 10.1109/TIP.2003.822613

is to optimize entropy coding in the spatial domain. In Section II, the theory of MCECQ [6], [8] is introduced. Section III examines the problems involved when real data are used, and proposes fast solutions based on dynamic programming for binary variables. Section IV presents CQ design algorithms, suitable binary decomposition to facilitate CQ design, and a CQ-based compression scheme. Some experimental results are given in Section V.

II. Context Quantization

This section introduces two recent concepts of context quantization based on minimizing the conditional entropy [6], [8] and minimizing the adaptive code length [7], [8], respectively. In [8] the results were formulated for an abstract context alphabet. Here they are summarized for a finite set of contexts. Let X be a discrete random variable, and let the random variable C be a jointly distributed random vector drawn from a finite set. The elements, $\mathbf{c}$, of the set specifies the context. Given a positive integer M, we wish to find the quantizer $Q : C \rightarrow \{1, 2, \ldots, M\}$ that minimizes the cost function in question.

A. Minimum Conditional Entropy Context Quantization

The Minimum Conditional Entropy Context Quantization (MCECQ) [6], [8] was introduced based on minimization of the conditional entropy $H(X|Q(C))$ given the probability density functions $P(X|C)$ and the number of quantized contexts, M. Clearly, $H(X|Q(C)) \geq H(X|C)$ by the convexity of H. Thus in this setting the aim is to minimize the increase in conditional entropy for a given limited M. The quantization regions $A_m = \{\mathbf{c} : Q(\mathbf{c}) = m\}$, $m = 1, \ldots, M$, of an (optimal) minimum conditional entropy context quantizer are generally quite complex, and may not even be convex or connected. However, their associated sets of probability density functions $B_m = \{P_{X|C}(\cdot|\mathbf{c}) : \mathbf{c} \in A_m\}$ are simple convex sets in the probability simplex for X, owing to the necessary condition for optimal Q[8].

The solution is especially simple for a binary random variable, Y, as the probability simplex is one-dimensional. In this case, the quantization regions B_m are simple intervals. If the random variable Z is defined as $P_{Y|C}(1|C)$ (the posterior probability that $Y = 1$ as a function of C), then the conditional entropy $H(Y|Q(C))$ of the optimal context quantizer can be expressed by

$$H(Y|Q(C)) = \sum_{m=1}^{M} P\{Z \in [q_{m-1}, q_m)\} H(Y|Z \in [q_{m-1}, q_m)) \tag{1}$$

for some set of thresholds $\{q_m\}$ dividing the unit interval into M contigous quantization intervals B_m. Therefore the optimal MCECQ can be found by searching over the set of interval end points, $\{q_m\}$. This is a scalar quantization problem, which can be solved exactly using dynamic programming. Thus for a binary variable, Y, the problem of optimal MCECQ design is reduced to one of scalar quantization, regardless of the dimensionality of the context space. Once the scalar problem is solved, the optimal MCECQ cells A_m are given by

$$A_m = \{\mathbf{c} : P_{Y|C}(1|\mathbf{c}) \in [q_{m-1}, q_m)\}. \tag{2}$$

In [9] the minimum conditional entropy criterion above was applied to compute optimal context quantizers for the application of bi-level image compression (tested using the leave-one-out approach [9]). The authors showed that the cost function (1) satisfies the so-called 'concave Monge property', which is a strong monotonicity that can be exploited to greatly reduce the search domain of dynamic programming. As a result, the optimization of $\{q_m\}$ in (1) may be solved in $O(NM)$, where N is the number of raw, ie. unquantized contexts before quantization.

B. Context Quantization by Minimum Code Length

In MCECQ it is assumed that the probability density functions $P(X|C)$ are known. In that case it would be more efficient (compression wise) to use $P(X|C)$ directly in the coding rather than applying context quantization. In practice, however, a training set or the data sequence to be coded is used to estimate $P(X|C)$. In this case instead of using MCECQ designed for an assumed $P(X|C)$, the context quantizer is designed to minimize the actual adaptive code length of the given data set, x^T [8]. This principle is described below.

Let I be the size of the finite alphabet of x. Let n_i denote the counts of the symbol i in context $\mathbf{c}$. A simple estimator of the probability that the next occurrence is i in context $\mathbf{c}$ is given by

$$\hat{p}(i|\mathbf{c}) = \frac{n_i + \delta}{\sum_{i=0}^{I-1} n_i + I\delta} \tag{3}$$

where δ is a parameter of the estimator. The ideal adaptive code length for the data set x^T is

$$L(x^T|C) = \sum_{t=1}^{T} -\log \hat{p}_t(x_t|\mathbf{c}_t) \tag{4}$$

where $\mathbf{c}_t$ is the context of x_t and $\hat{p}_t(x_t|\mathbf{c}_t)$ is given by sequentially updating the counters (3) based on the causal part, x^{t-1}, of the data.

Let $N_i(\mathbf{c})$ be the occurrence count for symbol i in context $\mathbf{c}$ over the whole data set. For a given sequence x^T, the total code length $L(x^T|Q(\mathbf{c}))$ out of the adaptive context-based arithmetic coding may be computed based on the set of counts $\{N_i\}(\mathbf{c})$ over the contexts $\mathbf{c}$, because the order of the symbol appearrance within a context class does not change the ideal adaptive code length. For a given context quantizer, with quantization regions $A_m = \{\mathbf{c} : Q(\mathbf{c}) = m\}$, $m = 1, \ldots, M$, the counts after quantization are given by

$$N_i(m) = \sum_{\mathbf{c} \in A_m} N_i(\mathbf{c}). \tag{5}$$

Let L_m denote the ideal adaptive code length of all symbols whose contexts fall into CQ cell A_m. Let T_m be the total number of occurences of symbols whose contexts fall in A_m.

$$L_m = \sum_{k=0}^{T_m - 1} \log(k + I\delta) - \sum_{i=0}^{I-1} \sum_{j=0}^{N_i(m)-1} \log(j + \delta) \tag{6}$$

where the last term only sums over values of i for which $N_i(m) > 0$. The context quantizer $Q(\mathbf{c})$ minimizing the total code length is determined by

$$\min_{M,Q(\mathbf{c})} \sum_{m=1}^{M} L_m = \min_{M,Q(\mathbf{c})} L(x^T|Q(\mathbf{c})). \tag{7}$$

This defines the optimal context quantizer $(M, Q(\mathbf{c}))$ for the given data set, x^T, with respect to the adaptive code length, (when the cost of the context quantizer itself is neglected). If more than one value of M attains the minimum in (7) the smallest of these is chosen. The optimization problem may be defined based on the set of counts $\{N_i\}(\mathbf{c})$. Therefore the dimensionality is the size of the alphabet, regardless of the order as well as the dimensionality of the contexts, $\mathbf{c}$.

III. Empirical Context Quantization – Binary Case

The practical significance of the optimal CQ algorithms presented is that they offer constructive means of optimizing source codes for minimum code length via high-order context modeling. Measuring the code length of the same data set that is used for the CQ optimization establishes a lower bound on the achievable code length.

In the following development our focus is on designing and implementing CQ for actual coding and the scope in the rest of the paper is restricted to CQ for coding binary decisions. Non-binary variables, x, may be decomposed into a sequence of binary decisions, y, and coded using the proposed method. The estimator (3), with $I = 2$, is often the basis of the adaptive probability estimation for coding binary data with $\delta \in [0, 1]$.

One approach to CQ design is first to estimate the probabilities of $P_{Y|C}$ using (3) and the probabilities of the contexts $P(C)$ for a large training data set. Thereafter the dynamic programming MCECQ design algorithm is applied to compute the context quantizer which minimizes the conditional entropy (1) for a given M. This MCECQ solution could also be characterized as a minimum static code length solution (when the costs of the context quantizer and the probability parameters are neglected). The conditional entropy $H(Y|Q(\mathbf{c}))$(1) in MCECQ is a non-increasing function in M. But the actual adaptive code length (achievable description length) of a given input sequence is not, due to the impact of model cost. Thus context quantizers of different resolutions (M) should be evaluated by the corresponding adaptive code lengths on the data set to determine the value of M yielding an overall mimimum. Using the adaptive code length as cost function, the optimal value of M is determined within the CQ calculation.

A. Minimum Code Length Context Quantization

This section presents a solution to CQ design for minimum adaptive code length of a given binary data set. We refer to the procedure as Minimum (adaptive) Code Length Context Quantization (MCLCQ). Lets consider adaptive context based coding of a binary sequence y^T. Let 0 and 1 be the labels of the binary variable Y. The probability estimate (3) is sequentially updated on the fly for each of the binary input symbols. The quantizer cells of the optimal solution to (7) may be defined by the set of counts, $\{N_0(\mathbf{c}), N_1(\mathbf{c})\}$, for all the contexts. The counts are discrete as opposed to the probability simplex. We have not been able to prove or specify conditions ensuring convexity of the quantization partition when using the adaptive code length.

In our implementation, we restrict ourselves to quantizer cells specified by contigous intervals $A_m = \{\mathbf{c} : \hat{P}_{Y|C}(1|\mathbf{c}) \in [q_{m-1}, q_m)\}$. Under this restriction the solution may be found by dynamic programming as for the MCECQ. The probability estimates $\hat{P}(Y|C)$ that are used for *ordering* the contexts prior to the dynamic programming procedure are obtained from the final counts on the data set, using (3) with $\delta = 0$.

In calculating the adaptive code length for potential quantization classes, the probability estimator (3) is used with δ set to the value used in the coding procedure we are optimizing for. The adaptive code lengths used in the dynamic programming are calculated based on the counts, $(N_0(\mathbf{c}), N_1(\mathbf{c}))$, for each possible context cell (quantizer interval). Since the dynamic programming algorithm uses the actual adaptive code length for a given finite sequence as the cost function (for fixed δ), it can automatically decide the optimum number of coding contexts M. This is simply done by increasing the number of context quantizer cells in the bottom-up dynamic programming process, until reaching the point when the actual code length starts to increase.

If the size of the training set is normalized to be of the length of an input sequence to be coded, then the dynamic programming algorithm can automatically decide the optimal number, M, of coding contexts for this input size and the δ value chosen. The MCLCQ procedure above may also be used for on-line context quantization. In this case δ is set to a small value in the initial sorting of $\hat{P}(Y|C)$.

B. Complexity of Context Quantization – Binary Case

Let $L_m(N_0, N_1)$ denote the ideal adaptive code length of context quantizer cell A_m given the counts $(N_0(\mathbf{c}), N_1(\mathbf{c})) = (N_0, N_1)$ as in the binary case of (6). Then we have

$$L_m(N_0, N_1) = -\log \frac{\prod_{j=0}^{N_0-1}(\delta + j) \prod_{j=0}^{N_1-1}(\delta + j)}{\prod_{j=0}^{N_0+N_1-1}(2\delta + j)} \tag{8}$$

where $\prod_{j=0}^{N_i-1}(\delta + j) \equiv 1$ if $N_i = 0$. These adaptive code lengths (8) can be computed in $O(1)$ time independent of the interval length by a fast algorithm proposed by [10]. The idea is to use table look-up to compute the adaptive code length for small values of the counts, and use Stirlings approximation for the products in (8) for large values when such an approximation yields high precision. With the fast adaptive code length computation technique, one can precompute and store the adaptive code lengths for all possible quantizer intervals. This preprocess takes $O(N^2)$ time, where N is the number of distinct unquantized raw contexts. Aided by the intermediate results of the preprocess (adaptive code lengths of all possible quantizer intervals), the dynamic programming algorithm can be completed in $O(MN^2)$ time. Unfortunately, we have verified that the cost function of adaptive code length does not satisfy the "concave Monge property" (also known as quadratic inequality). Only for the MCECQ version of the problem the "concave Monge property" is satisfied and the computational complexity of designing an M-level optimal

context quantizer can be reduced from $O(MN^2)$ to $O(MN)$ [9] by applying a technique called matrix search to quantizer design as shown by Wu [11]. However, since MCLCQ is an off-line design process, the $O(MN^2)$ complexity to achieve the minimum adaptive code length is acceptable. If this complexity is deemed too high, one can settle for the minimum static code length (MCECQ) in $O(MN)$ time. A good compromise is to design the context quantizer for the minimum static code length, but still use adaptive arithmetic coding in all coding contexts to adapt to changing source statistics. In our experiments, adaptive arithmetic coding using a context quantizer designed for the minimum static code length obtains code lengths that are only one percent longer than if the context quantizer is designed for the minimum adaptive code length.

Further speed up of the dynamic programming algorithm is achieved by merging all the raw contexts that have the same counts. This can reduce the number of initial contexts subject to quantization significantly. This will not affect the optimal solution because those contexts would be merged anyway by the CQ scheme above. The estimator (3) is optimal if the events in a context are independent and the prior distribution initially is beta distributed with nuissance parameter δ. In this view all the contexts of the same counts have the same posterior distribution of the parameter $\hat{p}(0|\mathbf{c})$, which also suggests that they should be quantized into the same context cell.

For an analysis of worst-case complexity, we develop a bound for the maximum number (N') of contexts having distinct counters $(N_0(\mathbf{c}), N_1(\mathbf{c}))$. Given a training set of size T, the case leading to the maximum value of N' is considered: All contexts have distinct counters and the sum of the counts is as small as possible. This case is characterized by having j distinct sets of counters for contexts with $j-1$ occurences. For a given T, the maximum value of N' is obtained by having no duplicate set of counters and 'filling up' the sets starting with the small values of j. Let J' be the minimum value of J satisfying

$$\sum_{j=2}^{J}(j-1)j \geq T. \tag{9}$$

For the given value of T

$$\sum_{j=2}^{J'} j > N' \tag{10}$$

gives a bound on the number of contexts with distinct counters. Combining (9) and (10) shows that the number of contexts, N', with distinct counters is $O(T^{2/3})$. Inserting in the complexity expression above gives $O(MT^{4/3})$ for MCLCQ in the binary case and $O(MT^{2/3})$ for MCECQ optimization in the binary case. Reading the data and contexts of the training set is of course $O(T)$. This shows that the CQ optimization itself using dynamic programming for the binary case is quite manageable.

Additional reduction in the computations of the dynamic programming training on data set may be achieved by merging all contexts having identical probability estimate, $\hat{p}(0|\mathbf{c})$. The effect of this depends somewhat on the parameter of δ chosen for the estimate.

C. Quantizer Mapping

Besides the complexity of CQ optimization there is the resource issue of handling and representing an arbitrary quantizer mapping. In two-pass coding, the representation of the quantizer as a preamble may also challenge the efficiency.

The simplest way of implementing an arbitrary quantizer mapping Q is to use a look-up table. But since $|C|$, the number of all possible raw contexts, is very large for high-order contexts, building a huge table of $|C|$ entries for Q is clearly impractical. However, one can use hashing techniques to avoid excessive memory use of the $Q(\mathbf{c})$ table by exploiting the fact that the actual number of different raw contexts appearing in an input sequence is much smaller than $|C|$.

Another approach for reducing the high space complexity of the mapping Q is to modify the structure and partition a high-order context C into two (or more) subcontexts C_1 and C_2 of lower order, and then use multiple tables of smaller sizes to implement the context quantizer in some product form. One such technique proposed by Wu *et al.* [12] is based on the assumption of a Bayesian-type conditional probability estimate, given the subcontext values, $\mathbf{c}_1$ and $\mathbf{c}_2$:

$$\hat{P}_{Y|C_1C_2}(0|\mathbf{c}_1\mathbf{c}_2) = \frac{\hat{P}_{C_1|Y}(\mathbf{c}_1|0)\hat{P}_{C_2|Y}(\mathbf{c}_2|0)\hat{P}(0)}{\sum_{x=0,1}\hat{P}_{C_1|Y}(\mathbf{c}_1|x)\hat{P}_{C_2|Y}(\mathbf{c}_2|x)\hat{P}(x)} \tag{11}$$

where $\mathbf{c}_1$ and $\mathbf{c}_2$ denote the context values of C_1 and C_2, respectively. In adaptive arithmetic coding, a quantized version of the Bayesian estimate may then be used

$$\hat{P}_{Y|C_1C_2}(0|\mathbf{c}_1\mathbf{c}_2) = \frac{\hat{P}(Q_1(\mathbf{c}_1)|0)\hat{P}(Q_2(\mathbf{c}_2)|0)\hat{P}(0)}{\sum_{x=0,1}\hat{P}(Q_1(\mathbf{c}_1)|x)\hat{P}(Q_2(\mathbf{c}_2)|x)\hat{P}(x)} \tag{12}$$

where Q_1 and Q_2 are MCECQ quantizers of disjoint contexts C_1 and C_2. This technique reduces the quantization table from a size of $|C_1| \cdot |C_2|$ into two tables of total size $|C_1| + |C_2|$.

IV. CQ-Based Coding Scheme

For use in a coding scheme, the context quantization may be calculated based on the counts obtained from a training set and thereafter fixed, or it may adaptively be calculated based on counts from a causal part of the data set. In the latter case, the encoder and decoder must apply the same context quantization algorithm to the causal data. Finally CQ may be used in a two-pass manner: collecting statistics, performing CQ and coding the context mapping as a preamble before coding the data set itself. The two-pass coding raises the question of coding the context mapping which we do not consider here instead focussing on the first solutions.

We shall only apply context quantization to binary random variables. Non-binary variables X_t are decomposed into a sequence of binary decisions when being coded. A simple choice is to code the bits of the binary representation of each value x_t.

A. Binary Decomposition

In general, any binary decomposition of the values x_t may be used. We shall use the approach of ordering the possible values and let each decision code whether the value is the next in order until the actual value has been coded. We consider two ways of ordering the values: 1) Based on an order of individual pixel values within the context. 2) Dynamic ordering based on the adaptively estimated likelihood of the possible values. As an example of the first type of ordering, PWC [4] starts by coding the binary decision whether the current pixel x_t is equal to the previous pixel, x_{t-1}. This is done by coding a so called edge-map. The dynamic ordering is elaborated below.

B. Two-Level Context Based Prediction

The current nonbinary value x_t may be decomposed in order of decreasing estimated conditional binary probabilities. Such a binary decomposition is context specific and data dependent. The decomposition is used as well in the CQ design as in the actual coding. Let z^T denote the training data and z_t the t'th sample of z^T. The training data is traversed once, and statistics are gathered for the unquantized contexts, $\mathbf{c}$. Based on the statistics for the causal part z^{t-1} the values are ordered by $\hat{p}(z_t|\mathbf{c})$, which are estimated using (3). Thereafter the values are considered one by one in order of decreasing probability until the actual value is found. Statistics is gathered for each of these binary decisions. After the statistics for the training data have been gathered, CQ is applied to the contexts, $\mathbf{c}$, for each of these binary decisions. In our tests on image sequences the previous image is used as the training data, z^T, for the current image, x^T. The conditional probabilities $\hat{p}(x_t|\mathbf{c})$ used for the binary decomposition in the actual coding are based on the statistics of the previous image, z^T, and the causal part of the current image, x^{t-1}.

Very likely new contexts or new values in a given context will appear in the data being coded. These cases are handled separately using an escape technique. The new contexts are thereafter included without quantization.

C. CQ Coding Scheme

Based on a combination of the techniques above, we present a CQ coding scheme for image sequence data. The coding scheme is specified by 1) binary decomposition, 2) context quantization, and 3) adaptive coding. In the coding step the statistic counts are reset to zero before each image. Thus the probability estimates of the binary variables $\hat{p}(x_t = i|Q(\mathbf{c}))$ are based on the statistics of the causal part, x^{t-1} of the current image. Based on these estimates the data may be coded using arithmetic coding.

The scheme is aimed at sequences of computer generated image data as maps, graphics, α-planes etc. Single images as individual maps may be tiled and the scheme applied to a sequence of these tiles. Each image or tile in the sequence is referred to as a frame.

The coding of each pixel value is decomposed in binary decisions, supplemented by m-ary decisions for ease of implementation. The coding of x_t is completed when its value is defined by an affirmative answer to a question. The context, $\mathbf{c} = (x_{t-t_1}, \ldots, x_{t-t_K})$, is defined by a template.

The basic version is presented first: Is the current pixel value x_t equal to

1) the west pixel value, x_{t-1}? (B)
2) the most probable value of the context pixels, $x_{t-t_1}, \ldots, x_{t-t_K}$, (besides that of the west pixel)? (B)
3) one of the remaining context pixel values?
4) the most probable (remaining) symbol value seen in the context, $\mathbf{c}$? (B)

where (B) marks binary decisions. If the value has not been coded by affirmative answer to one of the questions above, the actual value among the remaining colors is coded using the full context, $\mathbf{c}$. These values are referred to as escape values. Binary CQ based on (3) – (7) is applied directly to the binary decisions above. For questions 2 and 4, binary CQ based on (3) – (7) with two-level contexts is applied as described in Section IV-B. An individual CQ is carried out for each of the binary questions. The last question may be repeated (three times is default in the basic version). In the nonbinary question 3, only one context is used.

The basic version uses one template defining unquantized contexts, $\mathbf{c}$. An extended version is introduced below which may apply a different (smaller) template ($\mathbf{c}'$) for questions 4 and 5. In this extended version, the last coding of the escape values above using the full context is replaced by the (nonbinary) questions: Is the current pixel value, x_t equal to

5) a remaining symbol value seen in the context, $\mathbf{c}'$?
6) a remaining symbol value seen in the causal part of the current frame, x^{t-1}, or the previous frame, z^T?

If the value has not been coded by the decisions above, it is coded out of the remaining symbol values of the alphabet. Question 5 is coded using the unquantized context, $\mathbf{c}'$. Question 6 simply assumes a uniform distribution over the remaining values.

A few comments to this scheme may be in place. The first question is always asked. Once the value has been determined no more questions are asked. One or more of the questions after the first may be void depending on the unquantized context and the values having appeared in it. The questions 3, 5 and 6 are coded as nonbinary; the actual value is coded or an escape symbol tells to proceed to the next question. Contexts not present in the data set, on which the CQ is based, are included without quantization when they appear.

The decomposition scheme above is partly inspired by PWC [4] and RAPP [13], which are efficient coders for palettized and graphic images. In principle PWC starts with the same question (coding the edge map) and RAPP [13] codes if the value is equal to one of the values in the (4 pixel) template.

The PWC algorithm uses a binary decomposition of which a few details are given: To code the first binary decision, PWC uses a 10-pixel context. A heuristic, predefined context quantization is applied to map the template pixels onto an edge map. An edge between two 4-neighboring pixels records whether or not they are identical. The edge map is defined by the edges between each of the four causal 4-neighbors of the current pixel and all their causal 4-neighbors. Thus the quantized context is defined by nine binary edge values. If the first decision does

not suffice to determine the pixel value, the coding process continues with binary questions on whether the current pixel value is identical to one of the three other causal 8-neighbors. Further details are given in [4]. If initial guesses fail, PWC may resort to coding a prediction error.

V. Experimental Results

The CQ coding schemes of the previous section were applied to digital maps and α-plane sequences. These test data are not binary so the coding includes the decomposition in binary decisions. First the CQ scheme is applied as an analysis of the performance of some existing context quantization schemes. The results of the CQ algorithms are measured by ideal code lengths, i.e., calculating the adaptive code length based on estimates of conditional probabilities.

A. Maps

The context quantization was applied to street maps. First we focus on measuring the performance of coding the first question (i.e., is the current pixel identical to the west pixel?) in the CQ based coding scheme in comparison with existing methods applying a heuristic context quantization. A street map of Copenhagen [14], [15] having a resolution of 723 by 546 pixels with 12 colors was used. Table I gives the results for (optimal template sizes) coding with a 5 pixel template directly compared with context quantization by relative pixel patterns [13]–[15] (9 pixels), edge patterns (as in PWC, 10 pixels), and a combination of 4 unquantized pixels and 3 pixels quantized by relative pixel patterns. In comparison, Table II gives the optimal adaptive code length obtained by training the CQ on the very same data set. MCLCQ and MCECQ gave the same code lengths within the accuracy given in the table. MCLCQ automatically finds the number of contexts to use. For MCECQ we searched for the number of contexts yielding the shortest code length. These CQ based code lengths can not be realized as they require that the decoder knows the exact context quantization. The proposed optimal context quantizer technique may here be used to establish a lower bound on the achievable (adaptive) code length, and hence is a useful tool to evaluate the performance of existing heuristic context quantizers. If a coder e.g., is restricted to a template size of $M = 5$ pixels, it is seen that even optimal context quantization (0.316 bpp) provides little gain compared to template coding (0.327 bpp) for this map. For larger templates, the results in Table II suggest that improvement is possible over the heuristic methods of Table I. For this map the first question accounts for most of the code length (0.305 bpp out of 0.381 bpp for the 7 pixel combination of unquantized and relative pixels patterns). Table II also gives the number of contexts appearing in the data set as well as the number of context classes after quantization. The total number of distinct contexts, N, increases with increasing template size. The number of contexts with distinct counts, N' is much smaller decreasing after a maximum for a template size of nine. The optimal number of classes, M, after CQ was 32 at the most in this experiment (Table II).

Browsing maps on the internet, the previous map may be used to train the CQ. The maps may also be tiled and requested sequentially. The tiles of maps already received may be used for coding the next tile. Tests were conducted on four street maps [7]. For three of the maps, MCLCQ in the basic version reduced the code length of PWC by 7–22%. For one of the images the results were basically the same.

TABLE I
Copenhagen Map. Ideal Code Lengths (Bits per Pixel) for Coding the First (West Pixel) Question Using Different Context Definitions. (The Optimal Template Size for Each Method is Also Given)

TEMPLATE (5 PIXEL)	REL. PATT. (9 PIXEL)	EDGE PATT. (10 PIXEL)	COMBINED (4+3 PIXEL)
0.327	0.322	0.331	0.305

TABLE II
Copenhagen Map. Ideal Code Lengths (Bits per Pixel) Calculated for the First (West Pixel) Question Using MCLCQ or MCECQ as a Function of the Template Size. (MCLCQ and MCECQ Yield the Same Code Lengths.) the Number of Different Contexts Appearing in all and the Number of Context Classes After CQ by Counts, MCLCQ and MCECQ are Also Given

TEMP. SIZE	WEST PIX (BPP)	CONTEXTS (N)	COUNT CQ (N')	MCLCQ (M)	MCECQ (M)
1	0.539	12	12	10	10
2	0.431	100	94	22	22
3	0.397	388	261	24	25
4	0.352	1064	488	27	27
5	0.316	2446	744	29	28
6	0.290	5025	1000	31	30
7	0.266	8279	1111	32	31
8	0.248	12505	1199	31	31
9	0.230	17728	1244	32	30
10	0.210	23751	1200	32	31
11	0.192	30257	1178	30	30
12	0.175	37114	1117	31	30

B. α-Planes

Experiments were conducted coding MPEG4 α-plane sequences. MPEG4 is an object-based video compression standard. MPEG4 coding using video objects, may associate an α-plane (see Fig. 1) with each frame of a video object. The α-plane specifies, on a pixel basis, how the video objects in the foreground should be blended into the background. An initial application area could be in storage of high quality object based video.

The adaptive CQ schemes, including the binary decomposition, were applied to (lossless) coding of three MPEG4 α-plane sequences Logo, Weather and Rain. Each frame has a resolution of 720 by 486 pixels which are traversed row by row. The sequences are coded frame by frame without using motion compensation. The context quantizer is designed based on the previous frame. The results of the Logo sequence (children, layer 2) are depicted in Figs. 2–4. The first frame is kept out of the results, coding the remaining 299 frames having a total of 104 Mpixels. Fig. 2 plots the total ideal code length for the first binary decision (the west pixel) as a function of the template size for the Logo sequence. This decision is fully coded based on binary CQ. For a template size of 9, the average number of different contexts per frame was 2579 over the sequence. Mapping contexts with the same counts reduced the average number of different contexts to 179. The optimal CQ algorithms reduced this to as little as 11. The best results were obtained finding the template pixels using a greedy search. In each pass of the data set, the best one out of the neighboring pixels of the template of size n is

Fig. 1. An α-plane image from test sequence Logo. Left) Full frame having 21 different pixel values. Right) Enlargement of part of the frame.

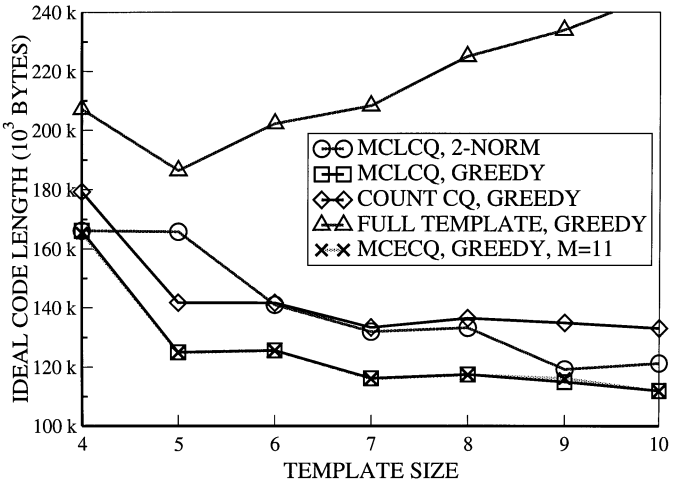

Fig. 2. Ideal code lengths as a function of the template size for the first question for the Logo sequence. The template pixels were found by greedy search or 2-norm distance.

chosen for the template of size $n+1$. Fig. 2 also depicts the result using templates which grow in size by incrementally adding the next nearest causal neighbor (in 2-norm distance to the pixel to be coded).

For the first question applying the edge map quantization heuristic for a 10 pixel template (as in PWC) gives an ideal code length of 359 808 bytes. This is significantly reduced to 121 237 bytes applying the MCLCQ to the same template pixels.

Fig. 3 as a function of the template size. MCLCQ using template pixels determined by greedy search gives the best result, eg. 318 780 bytes for a 9 pixel template. The MCECQ results using a greedy template are almost as good requiring 321 024 bytes in comparison. For this MCECQ result we searched for the number of contexts ($M = 24$) yielding the shortest ideal code length. The CQ results are better than what is obtained by quantization by counts and much better than the result without context quantization. It may be noticed that the CQ algorithms are robust for increasing template size whereas the performance of the full context version deteriorates above 5 template pixels. This demonstrates the effectiveness of CQ as a technique of combating the context dilution problem.

The CQ schemes were also compared with the best methods in the literature for lossless image coding, including palettized images. Tables III and IV give the results for the three MPEG4 α-sequences: Logo, Weather and Rain (destruction, layer 8). As for Logo the first frame is kept out. For the Weather sequence the remaining 299 frames were coded. For the Rain sequence the remaining 168 frames were coded. Table III shows that PWC, 2D-PPM and RAPP all outperform CALIC

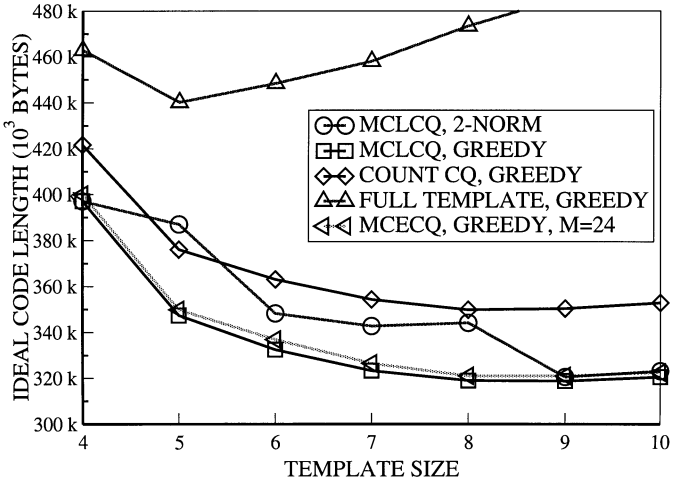

Fig. 3. Ideal code lengths for the Logo sequence as a function of the template size. The template pixels were found by greedy search or 2-norm distance.

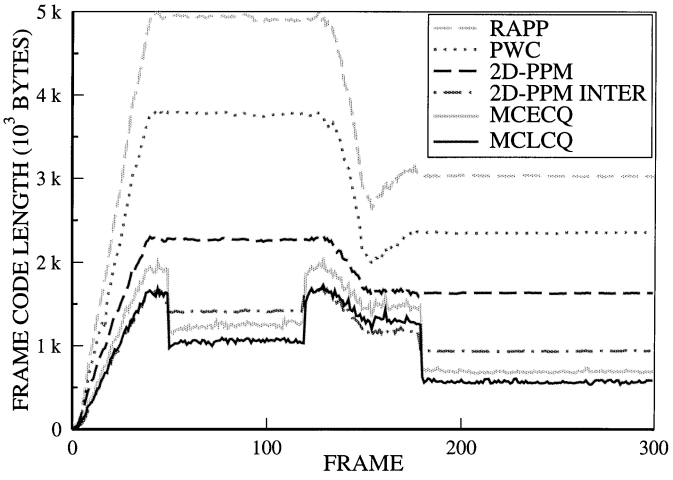

Fig. 4. Code lengths, frame by frame, for the Logo sequence comparing MCLCQ and MCECQ with RAPP, PWC and 2D-PPM.

TABLE III
TOTAL CODE LENGTHS (BYTES) FOR α-PLANE SEQUENCES CODED INDIVIDUALLY

SEQUENCE	PWC	2D-PPM	RAPP	CALIC
LOGO	820,484	535,730	1,067,216	2,616,770
WEATHER	1,050,212	1,262,516	883,104	1,151,512
RAIN	3,863,470	4,159,430	4,209,170	3,799,541
TOTAL	5,734,166	5,957,676	6,159,490	7,566,823

TABLE IV
TOTAL IDEAL CODE LENGTHS (BYTES) FOR α-PLANE SEQUENCES CODED USING STATISTICS OF THE PREVIOUS FRAME

SEQUENCE	MCLCQ	PWC	2D-PPM
LOGO	274,785	792,129	336,873
WEATHER	778,455	786,798	914,984
RAIN	3,277,285	3,678,917	3,430,093
TOTAL	4,330,525	5,257,844	4,681,950

[2], which serves as a benchmark for lossless coding of natural images. This indicates that these alpha sequences display more similarity with graphics and palettized images than natural images. The 2D-PPM scheme [16] combines classic PPM (prediction by partial matching) coding with a template defined

in the two dimensional image plane. The optimal choices of (max.) template size used with the 2D-PPM were 5 pixels for Logo, 2 pixels for both Weather and Rain. PWC and 2D-PPM gave the best results in Table III. The best MCLCQ (Table IV) significantly outperforms all of the methods in Table III. The MCLCQ coding scheme uses statistics (but not specific pixel values) from the previous frame, whereas PWC and the other techniques just code individual frames. In order to also make use of the previous frame PWC and 2D-PPM were modified such that the statistics were initialized by that of the previous frame. (Actually these code lengths were estimated by taking the code length of two consecutive frames and subtracting the code length of the first frame. Therefore the header size was also removed from these results.) Table IV gives the results. These interframe results for PWC and 2D-PPM gave code lengths which were 21% and 8% longer, respectively, than the MCLCQ code length for the test set (Table IV). MCLCQ used a 12 pixel template combined with a 9 pixel template for escapes and asking question 4 five times for Logo. MCLCQ used a 4 pixel template combined with a 1 pixel template for escapes for Weather and a 3 pixel template combined with a 1 pixel template for escape for Rain. For both of these sequences the template pixels are chosen by 2-norm distance and only asking question 4 once. The optimal choices of (max.) template size for 2D-PPM were 8 pixels for Logo, 4 pixels for Weather and 2 pixels for Rain.

Fig. 4 depicts the results frame by frame for the best settings of the extended MCLCQ and the basic MCECQ compared with 2D-PPM (with and without initialization), PWC and RAPP for the Logo sequence. The two low plateaus of the CQ results reflect parts with very high frame to frame correlation. But also outside these parts of the sequence, the MCLCQ scheme outperforms the other algorithms. Only the interframe 2D-PPM comes close in performance. For this sequence the average number of decisions per pixel were only 1.03 binary and 0.01 nonbinary decisions (in the basic version).

The Logo sequence is lossless coded very efficiently with MCLCQ using as little as 0.021 bpp. Rain is considerably more difficult to code requiring 0.45 bpp. Table V gives the results of applying MCLCQ, PWC and 2D-PPM to quantized versions of the Weather and Rain sequences. The quantizer is a slightly modified version of a Lloyd scalar quantizer [17] quantizing to 16 levels. MCLCQ used a 4 pixel template combined with a 3 pixel template for escapes for Weather and a 7 pixel template combined with a 4 pixel template for Rain. Question 4 was asked twice in both cases. For 2D-PPM, the optimal (max.) template sizes of 3 for Weather and 4 for Rain were used.

Including some of the techniques of PWC and 2D-PPM in MCLCQ would probably lead to even better results. PWC resorts to prediction of values which are different from the nearest neighbors. 2D-PPM decreases the context size until the context has been seen before. Another candidate for improvement is due to the fact that MCLCQ sets the counters used for the actual coding to zero before each frame. The PWC and 2D-PPM results were improved by carrying over the statistics from the previous frame (Table IV). The optimal choice is probably to initialize the estimates eg. using a down scaling of the counters from the previous frame. Finally for the α-planes, the temporal redundancy could also be exploited at pixel level. From an applications point of view issues as transmission errors, packetization, random access should also be addressed depending on the application.

TABLE V
TOTAL IDEAL CODE LENGTHS (BYTES) FOR QUANTIZED α-PLANE SEQUENCES CODED USING STATISTICS OF THE PREVIOUS FRAME.

SEQUENCE	MCLCQ	PWC	2D-PPM
WEATHER	354,437	409,374	412,254
RAIN	821,168	1,092,836	865,883
TOTAL	1,175,605	1,502,210	1,278,137

The emphasis of this work has been to demonstrate efficient coding based on the new CQ design. The techniques could also be applied within the entropy coding in lossy coding schemes.

VI. CONCLUSIONS

New techniques for optimal context quantizer design for minimum static and minimum adaptive code length calculated for a given data set were presented. The techniques were developed for optimizing contexts in which a binary random variable may be coded by adaptive arithmetic coding. Non-binary data can also be coded by these techniques if they are decomposed into a binary representation. Fast CQ design algorithms based on dynamic programming were developed and analyzed, especially with respect to the adaptive code length (the case of MCLCQ). In a specific application, the new CQ design algorithms were used to evaluate the performance of existing heuristic context quantizers used to compress digital maps. Also, in conjunction with a binary decomposition based on the (estimated) likelihood of the possible values of the input random variable, a coding scheme was developed for image (sequence) data. This approach may be used for tiles of an image as well, for example, in browsing of digital maps. Good results were demonstrated by the ideal code lengths calculated for MPEG4 α-plane sequences, surpassing those of current state-of-the-art lossless techniques such as PWC and CALIC.

REFERENCES

[1] J. Rissanen, "A universal data compression system," *IEEE Trans. Inform. Theory*, vol. 29, pp. 656–664, Sept. 1983.
[2] X. Wu, "Lossless compression of continuous-tone images via context selection and quantization," *IEEE Trans. Image Processing*, vol. 6, pp. 656–664, May 1997.
[3] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, pp. 1158–1170, July 2000.
[4] P. J. Ausbeck Jr., "The piecewice-constant image model," *Proc. IEEE*, vol. 88, pp. 1779–1789, Nov. 2000.
[5] X. Wu, "Context quantization with fisher discriminant for adaptive embedded wavelet image coding," in *Proc. Data Comp. Conf.*, Mar. 1999, pp. 102–111.
[6] X. Wu, P. A. Chou, and X. Xue, "Minimum conditional entropy context quantization," in *Proc. Int. Symp. Inform. Theory*, 2000, pp. 43–43.
[7] S. Forchhammer, X. Wu, and J. D. Andersen, "Lossless image data sequence compression using optimal context quantization," in *Proc. Data Comp. Conf.*, Mar. 2001, pp. 53–62.
[8] X. Wu, P. A. Chou, and S. Forchhammer, "Minimum conditional entropy context quantization," IEEE Trans. Inform. Theory, submitted for publication.
[9] D Greene, F. Yao, and T. Zhang, "A linear algorithm for optimal context clustering with application to bi-level image coding," in *Proc. Int. Conf. Image Proc. 1998*.
[10] B. Martins and S. Forchhammer, "Tree coding of bilevel images," *IEEE Trans. Image Processing*, vol. 7, pp. 517–528, Apr. 1998.

[11] X. Wu, "Optimal quantization by matrix-searching," *J. Algorithms*, vol. 12, no. 4, pp. 663–673, Dec. 1991.
[12] X. Wu, J. Wen, and W.-H. Wong, "Conditional entropy coding of VQ indexes for image compression," *IEEE Trans. Image Processing*, vol. 8, pp. 1005–1013, Aug. 1999.
[13] V. Ratnaker, "RAPP: Lossless image compression with runs of adaptive pixel patterns," in *32nd Asilomar Conf. on Signals, Systems and Comp.*, Nov. 1998.
[14] S. Forchhammer and O. R. Jensen, "Content layer progressive coding of digital maps," in *Proc. Data Comp. Conf.*, Mar. 2000, pp. 233–242.
[15] ——, "Content layer progressive coding of digital maps," *IEEE Trans. Image Processing*, vol. 11, pp. 1349–1356, Dec. 2002.
[16] S. Forchhammer and J. M. Salinas, "Progressive coding of palette images and digital maps," in *Proc. Data Comp. Conf.*, Mar. 2002, pp. 362–371.
[17] S. M. Aghito and S. Forchhammer, "Context based coding of quantized alpha planes for video objects," in *Proc. IEEE MMSP*, Dec. 2002, pp. 101–104.

Søren Forchhammer received the M.S. degree in engineering and the Ph.D. degree from the Technical University of Denmark, Lyngby, in 1984 and 1988, respectively.

Currently, he is an Associate Professor at Research Center COM at the Technical University of Denmark, where is has been employed since 1988. His main interests include source coding, data compression, video coding, information theory, and image communications.

Xiaolin Wu recieved the B.Sc. degree from Wuhan University, China, in 1982, and the Ph.D. degree from the University of Calgary, Calgary, AB, Canada, in 1988, both in computer science.

He is currently a Professor with the Department of Electrical and Computer Engineering, McMaster University, ON, Canada, and a Research Professor of computer science, Polytechnic University, Brooklyn, NY, and holds the NSERC-DALSA research chair in Digital Cinema. His research interests include visual computing and communications, data compression, and signal quantization. He has published over 100 research papers in these fields.

Jakob Dahl Andersen was born in 1962. He received the M.Sc.E.E. degree from the Technical University of Denmark (DTU) in 1988.

Since 1989 he has been employed at the Department of Telecommunication/COM Center at DTU. He has participated in several projects with the European Space Agency (ESA). His research interests include error-correcting codes and image compression.