A RUNLENGTH CODED LDPC SCHEME FOR INSERTION/DELETION CORRECTION IN MULTIMEDIA WATERMARKING

Bata Vasic[*] and Bane Vasic[+], *Fellow IEEE*
[*]Electronic Department, Faculty of Electronic Engineering, University of Nis , Serbia
[+]Electrical and Computer Engineering Department, University of Arizona, Tucson

Abstract

We describe a simple and effective coding scheme for insertion/deletion channels. It is based on runlength coding which converts a class of insertion/deletion channels that have infinite memory into memoryless channels, which are much easier to handle. Runlength coding is then combined with powerful error correction low-density parity-check (LDPC) codes designed for memoryless channels. We consider a novel applications of this technique in multimedia watermarking using quantization index modulation operating on the three dimensional mesh vertices. The runlength LDPC coding recovers the data hidden in the vertices removed by the process of mesh simplification.

I. INTRODUCTION

Synchronization errors occur in many data transmission and storage applications. Time misalignment between the transmitted and received data bits is difficult to handle, and even though such channels have been studied for decades, they are not characterized in terms of channel capacity, nor there exist a coding scheme that is believed to perform close to capacity. The main obstacle is that insertion/deletion channels have infinite memory.

The design of error correction systems that compensate for insertion and deletion errors is a reach area. In its seminal paper Levenshtein [1] provided a number theoretic construction of codes capable of correcting a single error (synchronization error or error in bit-value). Ulman's code [2] is a systematic form of Levenshtein code with no rate loss. Although a simple decoding algorithm for this class of codes exists [2], implementation is non-trivial for all but the smallest-length codes. Varshamov and Tenengol'ts [3] designed a code for asymmetric channels which can correct a single asymmetric error. Helberg code [4] is a generalization of Levensheein code providing multiple insertion/deletion capability. However, it has low rate and no systematic decoding algorithm.

A number of constructions is proposed that rely on the codes designed for substitution channels. Tonien and Safavi-Naini [5] constructed low rate multiple non-binary insertion/deletion correcting codes based on generalized Reed-Solomon codes and their subcodes. Dolgopolov [6] presented a non-binary linear codes to correct insertion/deletion errors, substitutions and erasures by purging the codewords from a linear substitution error correcting code. However, the alphabet

size must be larger than the codeword length. Ferreira *et al.* [7] use moment balancing templates to provide a single insertion/deletion correction capability to an arbitrary substitution error correcting code or a runlength limited code. However, these templates can only correct one error type per template codeword. Dolecek and Anantharam [8] presented a pruned first-order Reed-Muller code, i.e. a linear subcode which can correct multiple substitution errors in the presence of a single deletion or a single repetition error. They also constructed an expurgated array-based low-density parity-check (LDPC) code that enhances the code's substitution error capability in the presence of a single repetition error [9]. Landjev and Haralambiev [10] designed a multiple insertion/deletion correcting code by concatenation of the repetition code and the Levenshtein code.

In the past decade the research interest have shifted towards codes that support iterative decoding. Mitzenmacher [11] considered non-binary LDPC codes together with verification-based decoding algorithm. He proved that these codes are capacity approaching but the alphabet size is equal to the code length. Liu and Mitzenmacher [12] presented a code construction for a segmented deletion-only or insertion-only channel. Davey and MacKay [13] proposed watermark codes with a probabilistic decoding algorithm operating on a two-dimensional (2D) trellis for error-correction in insertion/deletion channels. Subsequently, marker codes based on a similar decoder was proposed by Ratzer [14]. These schemes, though observed to be powerful, rely on complex decoding algorithms which are not conducive to our applications.

One of the difficulties in addressing the problem of code construction is the fact that channels with synchronization errors have infinite memory, i.e. a synchronization error affects all subsequent symbols. In this paper, we introduce a coding scheme by which it is possible to transform certain channels with synchronization errors into memoryless channels. This naturally leads to a information encoding scheme by which conventional error-correcting codes may be used to compensate for synchronization errors.

We apply this concept to a secure watermarking scheme with guaranteed robustness to mesh optimization and simplification proposed in [15]. It combines sparse quantized index modulation (QIM) for data hiding with run-length modulated LDPC codes for recovering deleted watermark bits. The first step in the watermark recovery is detection of QIM bits, while the second step is deletion corection. A similar scheme was recently proposed by Coumou and Sharma [16] for audio signal watermarking. However, their method is based on the insertion/deletion/substitution correction scheme of Davey and MacKay [13] which combines marker codes for providing synchronization and LDPC codes for error correction. We provide simulation-based analysis of the watermark robustness subject to non-malicious transformations, namely mesh simplification. Mesh simplification deletes vertices, and the role of our codes is to mitigate this effect.

The rest of the paper is organized as follows. The second section gives relevant QIM and Sparse QIM notations and definitions as well as solution for increasing watermark robustness to affine transformations. Section III presents principles of insertions/deletions correcting three-dimensional (3D) watermark codes. Also, in this section we talk about Runlenght coding and LDPC coding. Finally, in fourth section we show numerical results of vertex deletion probability and error probability performance.

II. QUANTIZED INDEX MODULATION

We start in this section with a brief discussion of the embedding and detection in QIM and sparse QIM. Then we introduce quantization of polar and spherical coordinates. The *embedder* combines the *n*-dimensional vectors **u** and **x** and produces the watermarked sequence $\mathbf{y} \in \mathbf{R}^n$, where $\mathbf{u} \in \{0,1\}^n$ and $\mathbf{x} \in \mathbf{R}^n$ are the watermark sequence and the cover sequence, respectively. The difference **w=y-x** is the *watermarking displacement* signal and the embedder must keep the distortion $d(\mathbf{x},\mathbf{y})$ within a prescribed limit, i.e., $d(\mathbf{x},\mathbf{y}) \le nD$, where D is the maximum allowed distortion per dimension for every **x** and **u**. The distortion is typically defined as the simple Euclidian distance or the Hausdorff distance.

The QIM operates on independently on the elements u and x of the vectors **u** and **x**. To embed the bit $u \in \{0,1\}$, the QIM requires two uniform quantizers $\boldsymbol{Q}_0$ and $\boldsymbol{Q}_1$ defined as the mappings

$$\boldsymbol{Q}_u(x) = \Delta\left[\frac{1}{\Delta}\left(x - (-1)^u \frac{\Delta}{4}\right)\right] + (-1)^u \frac{\Delta}{4} \tag{1}$$

where, [ ] denotes the rounding operation, i.e. for a real x, $[x]$ is the integer closest to x. Thus, the quantization level of the "nominal" quantizer $\Delta\,[x/\Delta]$ is moved up or down by $\Delta/4$ depending on the value of u. Equivalently, the watermark bit u dithers the input x by the amount $\pm\Delta/4$. The watermark bit u determines the selection of a quantizer, so that $y = \boldsymbol{Q}_u(x)$. The minimum error produced by QIM is $\Delta/2$. Assuming uniform distribution of the quantization errors over the interval $[-\Delta/2, \Delta/2]$, the mean square error distortion is $\Delta^2/12$.

Ignoring the 3D-object content in the QIM may leads to serious degradation because small Euclidean error may be perceived by the Human Visual System (HVS) as a large distortion. Thus, we choose the vector **x** so that the application of QIM does not change visual quality [15]. Then, the QIM, spreads out the watermark bit over of the most favorable vertices of the cover signal **x.**

To increase the watermark robustness to affine transformation, spherical coordinates are used [17] and [15]. The default orientation and position of the cover 3D mesh is obtained by the following procedure. If V is a set of vertices in the 3D Euclidean space and an oriented edge from point u to point v is defined as the ordered pair (u,v), then F is a set of faces and 3D mesh is defined as a pair (V, F). The mass center of the vertices in the set V is calculated as

$$\mathbf{k}^c = \frac{1}{|V|}\sum_{v \in V} \mathbf{u}_v^C \tag{2}$$

where $\mathbf{u}_v^C = (x_v, y_v, z_v)$ is the position of the vertex **v** given in Cartesian coordinates, and $|V|$ denotes the size of the set V. Following [17] and [15], to ensure invariance to translation and rotation, the coordinate system is changed by translating the coordinate origin to the mass center $\mathbf{k}^c$, and by aligning the principal component vector with the z-axis. After the translation, the vertex positions in the new Cartesian coordinate system are $\mathbf{v}_v^C = \mathbf{u}_v^C - \mathbf{k}^c$ where $\mathbf{v}_v^C = (x_v', y_v', z_v')$. The 3D mesh is then rotated to align the principal component vector of the vertices with the z-axis, thus achieving robustness against rotation. Eigenvector that corresponds to the largest eigenvalue of the vertex coordinate covariance matrix is the principal component axis. Converting the Cartesian coordinates of the point **v** to spherical coordinates we achieve

robustness to uniform scaling and we have $r_v = \sqrt{x_v + y_v + z_v}$, $\theta_v = \arccos(z_v / r_v)$, $\varphi_v = \arctan(y_v / x_v)$, wherein $r_v \in [0,\infty)$, $\theta_v \in [0,\pi]$ and $\varphi_v \in [0,2\pi)$, are the radial distance, inclination angle, and the azimuth angle, respectively of the point **v**. Thus the vertex position in spherical coordinates are $\mathbf{v}_v^S = (r_v, \theta_v, \varphi_v)$. The original variant of the QIM operates only on the radial distance from the center of mass, i.e., moves a point **v** with the coordinates $(r_v, \theta_v, \varphi_v)$ to a point $(\boldsymbol{Q}_{u_v}(r_v), \theta_v, \varphi_v)$, where u_v is the bit hidden in the point *v*. The new center of the mass $\mathbf{k}^{c'}$ of the watermarked object has the following Cartesian coordinates: $x^{c'} = y^{c'} = \sum_{1 \le i \le n} \boldsymbol{Q}_{u_i}(r_i) \sin\varphi_v \sin\theta_v$ and $z^{c'} = \sum_{1 \le i \le n} \boldsymbol{Q}_{u_i}(r_i) \cos\varphi_v$. If a hidden binary sequence contains equal number of zeros and ones and if the vertex positions are uncorrelated with the quantization levels, then for a sufficiently large object $(x^{c'}, y^{c'}, z^{c'})$ converges to a zero vector**.** The above conditions are satisfied in practice, and the watermarked object is therefore invariant to affine transformations.

III. INSERTION/DELETION CORRECTING WATERMARK CODES

In this section we present the QIM watermarking employing LDPC codes for vertex deletion detection. The encoding proceeds as follows. After an optimizing probability of zeros and ones in Distribution transformer, watermark signal is encoded by binary LDPC code, and converted to *runs*. The next block of our system is *Synchronization Error Channel*, whereby watermarking process is completed and signal is prepared for transmission through the channel. Watermark retrieving process implies subprocesses of QIM and Runlenght bit detections as well as LDPC and Distribution decoders. The subsections of paper explain the essential blocks of the system.

We have already mentioned that channels with synchronization errors have infinite memory. Therefore, we use a coding scheme which enables to transform certain channels with synchronization errors into memoryless channels. The next subsection explains such encoding, called *runlength coding*.

A. Runlength coding

Transformation of channel with infinite memory to a memoryless channel, represents the bits at its input by runs of bits. The runlenghts are selected according to channel synchronization statistics and represent zeros and ones. We explain the main idea considering a case in which binary zeros are represented by runs of length *two*, and binary ones with runs of length *three* [18]. If the sequence of information bits is *b* =(0, 1, 1, 0, 1, 1), then the runlength encoded sequence *c* is initialized *c*=(11 000 111 00 111 000).

In our case the runlength sequence is embedded into selected vertices of a 3D object by using sparse-QIM: first, the watermark binary sequence is processed in the distribution transformer to optimize probability of zeros and ones, and encoded by a binary LDPC code. Then, conversion to runs and transmission through the channel are realized.

Symbols have different lengths, thus optimizing probability of symbols is necessary. The maximizing the mutual information, $I(\mathbf{X};\mathbf{Y})$, between the input alphabet **X** and the output

alphabet **Y**, over all input distributions of **X**. determines the Shannon capacity of memoryless channels, which is in discrete case obtained by maximizing the mutual information, $I(\mathbf{p})$, over all input probability distribution vectors $\mathbf{p}=(p(x))_{x\in X}$.

If the costs of transmission of different symbols are not equal, for the channel where symbol lengths are not equal, $c(x)$ as length of symbol x, can be viewed as the cost of transmission of the symbol x **.** If we define **c**, the transmission cost vector, $\mathbf{c}=(c(x))_{x\in X}$. In such channels, we use the notion of *unit-cost capacity* [19] defined as $C_{unit} = \max_{\mathbf{p}} I(\mathbf{p})/\mathbf{cp}^T$ The capacity is a function of the synchronization error probability as well as the size of the input alphabet **X**. A spectrum of transmission schemes can be obtained by changing the alphabet-size at the input. In general, for a channel C, with information alphabet **X**, we can calculate an associated unit-cost capacity $C(C; |\mathbf{X}|)$, and determine optimal capacity-achieving input probabilities. Subsequently, the data is encoded by using an error-correcting code and then, as described previously, the information symbols are encoded in runs of channel bits.

B. LDPC code decoding

Let C be an (n, k) LDPC code over the binary field GF(2). C is defined by the null space of H, an $m \times n$ parity-check matrix of C. H is the bi-adjacency matrix of G, which is a Tanner graph representation of C. G is a bipartite graph with two sets of nodes: n variable (bit) nodes $V = \{1, 2, \dots, n\}$ and m check nodes $C = \{1, 2, \dots, m\}$. A vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is a codeword if and only if $\mathbf{x}H^T = 0$, where H^T is the transpose of H. In this paper we consider $(d_v; d_c)$-regular LDPC, i.e., codes with Tanner graphs G in which all variable nodes have degree d_v, and all check nodes have degree d_c. Such codes have rate $R \geq 1-d_v/d_c$ [20].

The parity-check matrices of structured LDPC codes are arrays of permutation matrices obtained from Latin squares. A permutation matrix is a square binary matrix that has exactly one entry 1 in each row and each column and 0's elsewhere. Our codes (see [21] for details) make use of permutation matrices that have disjoint support. The runlength coding leads to a scheme in which conventional error-correcting codes may be used to compensate for insertion/deletion errors. In this section, based on our work in [18], we give a brief description of the system model synchronization-error channel and we consider and explain how information may be reliably transmitted through this channel.

Iterative decoding can be visualized as message passing on a Tanner graph [22]. The sum-product algorithm takes a priori information of the bit at position j, $\mu_j^{(0)}$ as the log-likelihood ratio of the i-th bit, $\log(P(x_i = 0|\, r_i)/P(x_i = 0|\, r_i))$, where $P(x_i = 0|\, r_i)$ is the a posteriori probability of the bit x_i being zero given the received symbol r_i. The messages passed from variable node j to check node c in the bipartite graph, $\lambda_{j,c}^{(0)}$, are initialized to $\mu_j^{(0)}$. In i-th iteration we update the messages to be passed from check node c to bit node j, $\Lambda_{c,j}^{(i)}$, as , and messages to be passed from bit node j to check node c, $\lambda_{j,c}^{(i)}$, according to $\lambda_{j,c}^{(i)} = \mu_j^{(0)} + \sum_{d\neq c} \Lambda_{d,j}^{(i)}$. The last step in iteration i is to compute updated log-likelihood ratios $\mu_j^{(i)}$ according to $\mu_j^{(i)} = \mu_j^{(0)} + \sum_c \Lambda_{c,j}^{(i)}$. For each bit j the estimation is made according to $\hat{x}_j = 1$ if $\mu_j^{(i)} < 0.$ and $\hat{x}_j = 0$ otherwise. The procedure halts when a valid codeword is generated or a maximum number of iteration has been reached.

We consider a channel with binary input and output alphabets. A bit may be inserted with a probability p_i, deleted with a probability p_d, or correctly transmitted with a probability $1-p_i-p_d$. In the context of 3D watermarking, the channel introduces only deletions ($p_i = 0$). We also assume that no two consecutive vertices are deleted in the process of simplification. This assumption is justified by the fact that the Ordered Statistics Vertex Extraction and Tracing Algorithm (OSVETA) [23] selects *stable* vertices, and two consecutive vertices are deleted extremely rarely. In other words, the space between two consecutive deleted vertices is sufficiently large, i.e. there are no *bursts* of synchronization errors. Separation of synchronization errors within a codeword may be also achieved by interleaving, i.e. by proper indexing of vertices. Consequently, we consider a variant of the channel described above in which the number of consecutive synchronization errors is restricted and operates on individual runs of binary sequences. Here, we define the term *run* in a binary sequences as an occurrence of k consecutive, identical symbols.

We denote a channel $\mathcal{C}$ with the parameter s_d, which is the maximum number of consecutive deletions as $(0, s_d)$. It is a special case of the channel we introduced in [18]. The probability of j consecutive deletions is $(p_d)^j$ for $j= 1, \ldots, s_d$. In each run of zeros or ones, only one of the $s_d + 1$ error events occurs, namely, (*i*) error-free transmission, or (*ii*) deletion of s_d bits, $j= 1, \ldots, s_d$. We do not consider errors in bit-values introduced by channel since we consider only common transformations. In general, these parameters may be chosen to match the behavior of the object in which we hide the watermark. Consider the channel $\mathcal{C}$ with parameters (p_d, s), having input **x**. As we have shown in [18] if all the runs in **x** have lengths greater than s, then the number of runs in any output **y** produced by $\mathcal{C}$ is equal to the number of runs in **x**. Summarizing, when transmitted through $\mathcal{C}$, each run of s_t bits, $s_t > s$, results in a run of j bits, $j = s_t - s, \ldots, s_t$. This observation leads naturally to an encoding scheme where symbols are encoded in runs of bits. To explain this, consider a channel with $s_d = 1$. At the receiver, the length of each run is counted in order to determine the output symbol. We note that at the receiver, the lengths of runs may change due to deletion. Nevertheless, a correct choice of runlengths (more precisely, by choosing runlengths greater than s) assigned to information symbols can lead to the i^{th} information symbol corresponding exactly to the i^t run of channel bits. As we noted in [18], an important consequence of such an encoding scheme is that transmission through the synchronization-error channel can now be represented as that of transmission through a memoryless channel. That is, any synchronization-error manifests as an error in the corresponding output symbol, and does not affect other symbols. Thus, classical error-correcting codes can be used to compensate for such errors.

The encoding scheme described above results in a discrete memoryless channel with the set of input symbols l_0 and l_0+1 (the runlengths corresponding to the input bits 0 and 1), the set of outputs of the channel correspond to the all the possible values of runlengths at the receiver, i.e. $\{l_0-1, l_0$ and $l_0+1\}$ and transitions that are possible only if output runlength is equal to the input or shorter by one. It is easy to see that this methodology is not restricted to encoding of binary-valued sequences, and can be easily extended to larger alphabets. For example, an input sequence is grouped into pairs of symbols (00, 01, 10 and 11), and then each pair is represented by a unique runlength (00 by runlength 2, 01 by runlengt 3 etc.). For example, an input sequence 0101001011 is parsed as 01, 01, 00, 10, 11 and then runlength encoded to obtain the sequence 000,111,00,1111,00000.

IV. NUMERICAL RESULTS

For all further computations we use results of vertex stability in relation with optimization process obtained by OSVETA [23] for the Naissa by Bata [24] 3D mesh model. For comparison and evaluation we have used 'Optimize' modifier from 3D Studio Max 2012 application [25].

Experimental tests of stability 1000 vertices, selected by OSVETA algorithm compared to the randomly allocated group of 1000 vertices showed the superiority of our approach compared to random selection of vertices. The results are summarized in Table 1.

	FT=0	FT=2	FT=4	FT=6	FT=8	FT=10
Total number of vertices	33465	25334	17472	12146	8588	5870
Random	0	276	495	654	760	845
OSVETA	0	3	21	44	77	156

TABLE 1 THE NUMBER OF VERTICES DELETED BY OPTIMIZATION OF VARIETY FACE THRESHOLD VALUES, USING RANDOM AND OSVETA SELECTION ALGORITHMS

The first row gives the total number of vertices remaining after simplification with a given Face Threshold (FT). The second and third row give the number of removed vertices out of 1000 vertices selected randomly and selected by the OSVETA. The nonzero numbers in the FT=0 column correspond to the total number of vertices in the original object. Probability of vertex deletion as a function of the face threshold is shown in Figure 1.

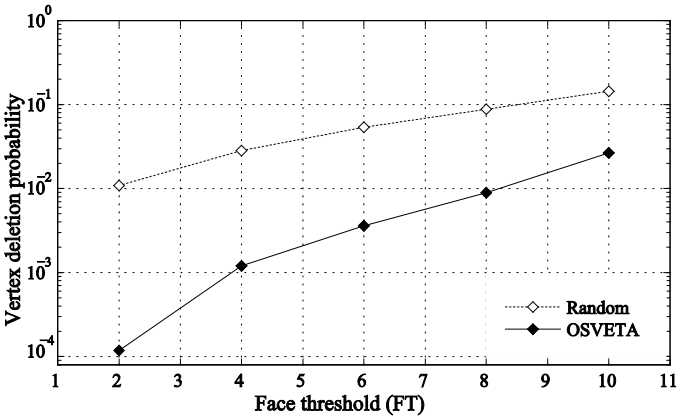

Figure 1. Probability of vertex deletion p_d as a function of the face threshold for the random selection of vertices (white markers), and vertices selected by OSVETA (black markers).

The important fact is that the OSVETA provides an extremely low probability of deleting two consecutive selected vertices. Actually, in 1000 selected vertices of all optimization levels we did not find any deleted pair of consecutive vertices. This justifying to the (p,1) deletion channel assumption.

A. Capacity Estimates

In order to determine bounds on the achievable rate of error correcting codes that may be used, we need to calculate the capacity of these channels. In this subsection, we investigate the capacity limits of channels described above.

For a channel $\mathcal{C}$, with information alphabet **X**, we can calculate an associated unit-cost capacity $C(\mathcal{C}; |\mathbf{X}|)$. These capacities constitute lower bounds on the capacity of $\mathcal{C}$. As the symbol costs (runlengths) are not uniform, the maximizing input distribution dictates that symbols with low transmission-cost have a higher probability than those with high cost. Figure 2. shows the optimal runlength probability distribution as a function of deletion probability p for |X|=2, and runelngth l_0 =2 and l_0+1 =3.

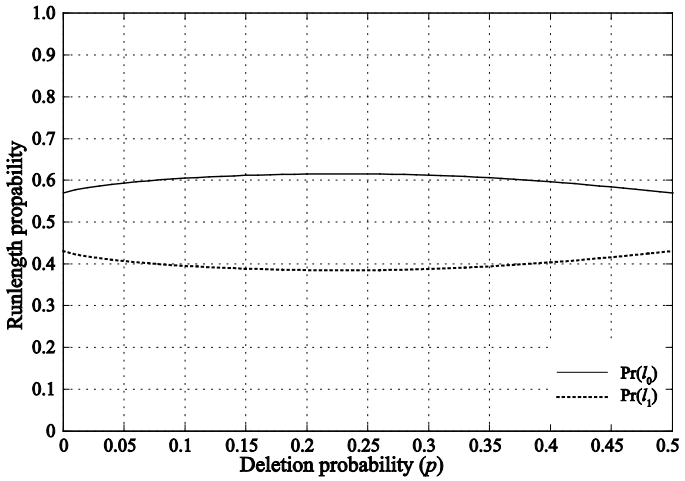

Figure 2. Optimal input runlength probabilities for the channel with |X|=2, and runlength l_0 =2 and l_0+1 =3. Capacity estimates for the Channel 2 ((0, p, 2; 2)).

B. Probability of Error Performance

In order to demonstrate the feasibility of implementation of our encoding methodology, we conducted experiments to simulate transmission of coded information through the channel (p, 1).

For our simulations, we chose synchronization-error probability (p) ranging from 0:01 to 0:05. Two codes were chosen as candidates for simulation. The codes were constructed by methods described in [26], and provide guarantees on error-correction capability under iterative error-correction decoding for BSC. The code parameters (code length, n, and the number of message bits, k) are as follows: n = 2212, k = 1899, *Reff* = 0.34 for Code 1, and n = 848, k = 661, *Reff* = 0.32 for Code 2, where R_{eff} denotes the effective rate, i.e. the number of information bits transmitted per channel use. For simplicity, in the simulations the equiprobable inputs are used.

Since the average cost of a bit transmission is 2.5 (symbols of duration 2 and 3 are used half of the time in average), there runlength coding introduces a rate loss of 2/5. The Effective rate is thus $R_{eff} = (2/5)R$ where R is the rate of the LDPC code.

On Figure 3. [15] the bit-error rates (BER) and frame-error (FER) rates are shown with respect to the vertex deletion probability parameter *p*. Sum-product algorithm was used for decoding the codes. For example to ensure that at most one in a million vertices is unrecoverable ($BER \leq 10^{-6}$) by an LDPC code (Code 1), the OSVETA has to provide a raw deletion probability of no more than $p=0.02$. From Figure 1 we see that this is readily achieved when face threshold is less than 6. The proposed scheme handles even higher FT if the maximum BER requirement is relaxed.

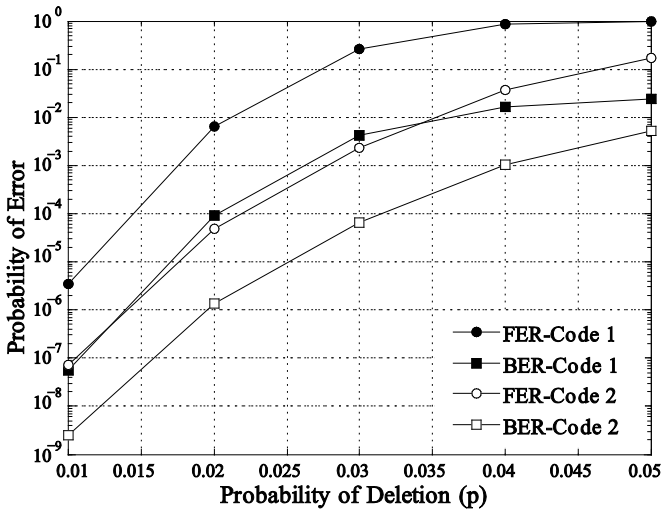

Figure 3. Codeword probability of error as a function of deletion probability *p* for codes of different lengths and rates.

We note that the input distribution is not optimized, i.e. the inputs were uniformly distributed. We also note that the use of distribution transformers introduce additional loss in the effective rate. However, a detailed discussion of this is beyond the scope of this analysis. We also note that although effective rates of codes for binary alphabet are low, by using codes over higher alphabets superior guarantees on maximum achievable rates may be obtained. This problem is left for future research.

V. CONCLUSIONS

A combination of QIM and error correction coding is an essence of the our 3D mesh watermarking method, which we have presented in this paper. Stabile vertices are selected by OSVETA algorithm, whereupon QIM is performed on spherical coordinates of these vertices. Runlength-encoded watermark bits are subjected to error correction code that ensures recovery of bits which are deleted by simplification. The runlength coding also makes the watermarking process equivalent to a memoryless channel and its conceptual simplicity allows using powerful codes that has developed for these channels.

Carefully designed low-density parity check codes give the strength of the proposed watermarking coding scheme. The sophisticated OSVETA algorithm provides a low probability of selected vertices deletion, whereas iterative decoding algorithm represents a computational simplicity in recovering of vertices, which are deleted by simplification process.

REFERENCES

[1] Levenshtein V.I., "Binary codes capable of correcting deletions. insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, 1966.

[2] Ullman J., "On the capabilities of codes to correct synchronization errors," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 95–105, Jan. 1967.

[3] Varshamov R.R. and Tenengol'ts G.M. "A code for correcting a single asymmetric errors," *Automation and Remote Control*, vol. 26, no. 2, pp. 286–290, 1965.

[4] Helberg A.S.J., Clarke W. A. and Ferreira H. C., "A class of dc free, synchronization error correcting codes," *IEEE Transactions on Magnetics*, vol. 29, no. 6, pp. 4048–4049, Nov. 1993.

[5] Tonien D. and Safavi-Naini R., "Construction of deletion correcting codes using generalized Reed-Solomon codes and their subcodes," *Designs, Codes and Cryptography*, vol. 42, pp. 227–237, 2007.

[6] Dolgopolov S., "Nonbinary codes that correct symbol insertions, drop-outs, and substitutions," *Problems of Information Transmission*, vol. 21, no. 1, pp. 26–29, 1985.

[7] Ferreira H. C., Abdel-Ghaffar K. A. S., Cheng L., Swart T. G. and Ouahada K., "Moment balancing templates: Constructions to add insertion/deletion correction capability to error correcting or constrained codes," *IEEE Transactions on Information Theory*, vol. 55, no. 8, pp. 3494–3500, Aug. 2009.

[8] Dolecek L. and Anantharam V., "Repetition error correcting sets: Explicit constructions and prefixing methods," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 2120–2146, Jan. 2010.

[9] Dolecek L. and Anantharam V., "Using Reed-Muller RM(1,m) codes over channels with synchronization and substitution errors," *IEEE Transactions on Information Theory*, vol.53, no.4, pp. 1430–1443, Apr. 2007.

[10] Landjev I. and Haralambiev K., "On multiple deletion codes," *Serdica Journal of Computing*, vol. 1, no. 1, pp. 13–26, 2007.

[11] Mitzenmacher M., "Polynomial time low-density parity-check codes with rates very close to the capacity of the *q*-ary random deletion channel for large *q*," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5496–5501, 2006.

[12] Liu Z. and Mitzenmacher M., "Codes for deletion and insertion channels with segmented errors," *In Proc. Int. Symp. Inform. Theory*, (ISIT), pp 846–850, 2007.

[13] Davey M. C. and Mackay D.J.C., "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[14] Ratzer E., "Marker codes for chanels with insertions and deletions", *Annals of Telecommunications*, 60:1-2, p. 29-44, 2005.

[15] Vasic B. and Vasic B., "Simplification Resilient LDPC-Coded Sparse-QIM Watermarking for 3D-Meshes," Submitted to *IEEE Transactions on Multimedia*, [Online]. Available: http://arxiv.org/abs/1204.2214

[16] Coumou D. and Sharma G. , "Insertion, deletion codes with feature-based embedding: a new paradigm for watermark synchronization with applications to speech watermarking," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 153-165, June 2008.

[17] Kalivas A., Tefas A. and Pitas I., "Watermarking of 3D Models Using Principal Component Analysis," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 5, pp. 676-679, Apr. 2003.

[18] Krishnan R. and Vasic B., "Coding for correcting insertions and deletions in bit-patterned media recording," *In Proc. IEEE Global Telecommunications Conference* (GLOBECOM '11), Dec. 2011, pp 1-5.

[19] Verdu S., "On Channel Capacity per Unit Cost," *IEEE. Transactions on Information Theory*, vol. 36, no. 5, pp. 1019–1030, Sep. 1990.

[20] Gallager R.G., "Low Density Parity Check Codes," Cambridge, MA, USA: M.I.T. Press, 1963.

[21] Nguyen D.V., Chilappagari S.K., Vasic B. and Marcellin M.W., "On the construction of structured LDPC codes free of small trapping sets," *IEEE. Transactions on Information Theory* (to appear).

[22] Vasic B. and Milenkovic O., "Combinatorial constructions of low-density parity-check codes for iterative decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1156-1176, June. 2004.

[23] Vasic B., "Ordered Statistics Vertex Extraction and Tracing Algorithm (OSVETA)," Technical report No 324 - 2012 - III 30., Nis, April, 2012, [Online]. Available: http://www.batavasic.com/research/OSVETA.pdf

[24] Serbian Film Center, Naissa Trophy 3D Model, Nis, August, 2011, [Online]. Available: http://www.batavasic.com/research/Naissa_by_Bata.zip

[25] Kauffman M., "Optimizing Your Autodesk® 3ds Max® Design Models for Project Newport", Autodesk University 2009, [Online] Available: http://au.autodesk.com/?nd=material&session_material_id=6296, pp:6.

[26] Vasic B., Milenkovic O. and McLaughlin S., "Scrambling for nonequiprobable signalling," *IEEE Electronics Letters*, vol. 32, no. 17, pp. 1551–1552, Aug. 1996.