Successive Lossy Compression for Laplacian Sources

Youngsuk Park and Hyeji Kim

June 11, 2014

Abstract

In this project report, we present three practical schemes of lossy compression for Laplacian source and L-1 distortion: The first scheme generalizes the successive refinement scheme. The second scheme modifies the first scheme by using only ternary alphabet. The third scheme encodes only the indices of maximum and minimum component. The whole of suggested schemes have a rateless property as well as are practical in terms of low complexity and applicability in real image data. Moreover, these also work for a Gaussian source.

1 Introduction

The lossy compression of continuous alphabet has been a practically valuable research field given the fact that most of multimedia sources are analog. Especially, a practical compression scheme for Laplacian source is significant because Laplacian distribution is widely adapted as the model for the DFT coefficients of correlation of image pixels or amplitude of voice.

With respect to discrete alphabet source, not only it is well developed in a sense that theoretical rate distortion theorem [1] was solved, but several practical schemes of Trellis bases quantizer [2], LDPC ensemble [3] and polar code [4] [5] have been developed. But, the lossy compression for continuous alphabets is relatively less exploited. Previously, the scheme using extremes and the Sparse linear regression scheme [6] were suggested for a Gaussian source. For a Laplacian source, MCMC based compressor [7] in high distortion and expansion coding compressor [8] in low distortion were suggested.

In this paper, we present some schemes satisfying the following viewpoints

- L-1 distortion criterion,
- high distortion regime working also in low distortion range,
- rateless and sequential property,
- low complexity and storage.

The rest of paper is organized as follows. In Section 2, the problem setup of lossy compression for Laplacian source is described. In Section 3, the successive refinement scheme is reviewed [9]. In Section 4, we introduce a generalized suceessive refinement scheme with continuous alphabets, along with a similar scheme except only for using discrete alphabet

in Section 5. In Section 6, we use the scheme only encoding the indices of the maximum and minimum component. In Section 7 and 8, three schemes in 4, 5, 6 are applied to compress real image data and a Gaussian source.

2 Problem Setup

Our problem setup follows the setting of lossy compression, where the source and the distortion measure are as follows.

Laplaician distribution Source sequence $\mathbf{x} = \{x_1, \cdots, x_n\}$ where x_i is i.i.d. $Lap(\lambda)$ given by

$$f_{X_i}(x_i) = \frac{1}{2\lambda} e^{-|x_i|/\lambda}$$

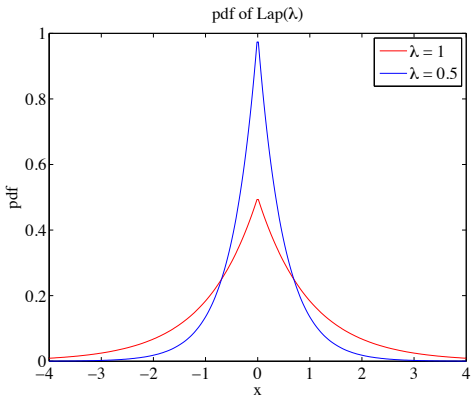

Figure 1: **PDF of Laplacian(λ)**

L-1 distortion The distortion measure is L-1 norm given by

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_1$$

2.1 Rate distortion function

The fundamental limit of lossy compression [1] of i.i.d Laplacian source for L-1 distortion is well characterized by rate distortion function given by

$$\begin{aligned} R(D) &= \min_{p(\hat{x}|x):\ E[d(x,\hat{x})] \le D} I(X; \hat{X}) \\ &= \begin{cases} \log(\lambda/D) & \text{for } D \le \lambda \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

This rate distortion function is achieved with $\hat{X}$ distributed according to

$$f_{\hat{X}}(\hat{x}) = \frac{D^2}{\lambda^2}\delta(\hat{x}) + (1 - \frac{D^2}{\lambda^2})\frac{1}{2\lambda}e^{-|\hat{x}|/\lambda}$$

as shown in Figure 2 and joint distribution with X is as shown in Figure 3.

We let $LapMixer(\lambda, D)$ denote this distribution.

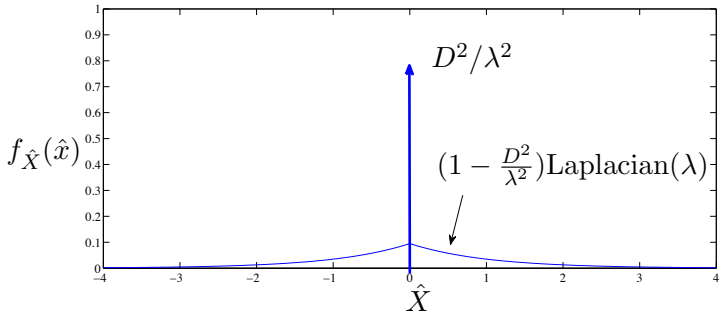

Figure 2: **Marginal distribution of $\hat{X} \sim LapMixer(\lambda, D)$**

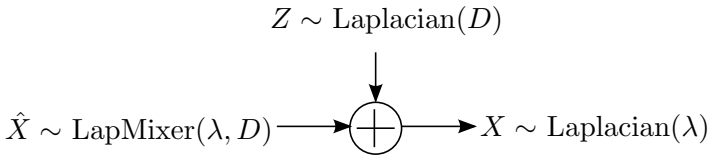

Figure 3: **Joint distribution of $\hat{X}$ and X**

3 Background: Successive Refinement

Theorem 1 The successive refinement rate-distortion region $\mathcal{R}(D_0, D_1)$ is the set of rate pair (R_1, R_2) such that, for a distortion pair (D_0, D_1) with $D_0 \leq D_1$

$$\begin{aligned} R_1 &\geq R(D_1) \\ R_1 + R_2 &\geq R(D_0) \end{aligned}$$

where $R(D) = \min_{p(\hat{x}|x):\mathbf{E}(d(X,\hat{X}))\leq D} I(X;\hat{X})$ is the rate-distortion function for a single descritpion.

Definition 1 If $(R_1, R_2) = (R(D_1), R(D_0) - R(D_1))$ is actually achievable for all $D_0 \leq D_1$ and there is no loss of optimality in describing the source successively by a coarse description and a refinement of it. Such a source is referred to as a **successively refinable source**.

Note that Laplacian source is a successively refinable source and thus all of points in $R(D)$ curve are achievable for sufficiently large n. However, one of the main problem of this successive refinement scheme is the codebook size which is exponential on the source size n. For example, in order to achieve target a (R, D) through K successive refinement steps with a rate increment $\Delta R = R/K$, the codebook size $Ke^{nR/K}$ is required. Another problem would be the computational complexity of joint typicality encoding.

4 Generalized Successive Refinement Scheme with Continuous Alphabets

Generalized successive refinement scheme follows the basic idea of successive refinement scheme 3 in a sense that both of them exploit several iterative stages to achieve target distortion D sequentially. Here, we generalize it by setting the iteration number $K_n = n/\log n$ varying with n so that the codebook size is polynomial on n, *i.e.,* $K_n e^{nR/K_n} = \frac{n^2 R}{\log n}$. On the top of that, L-1 norm minimizing criterion is used for encoding instead of joint typicality criterion in the encoder.

4.1 Scheme

Notation For an i.i.d $Lap(\lambda)$ source $\mathbf{x}$, fix the number of iteration K and the subcodebook size M. For each iteration stage $k \in [1:K]$, define $\Delta R = \frac{\log M}{n}$, $R_k = k \times \frac{\log M}{n}$, and $D_k = \lambda e^{-R_k}$.

Codebook generation: For each stage $k \in [1:K]$, fix $p(\hat{x}) = LapMixer(D_{k-1}, D_k)$ Randomly and independently generate M sequences $\hat{\mathbf{x}}(m)$, $m \in [1:M]$, each according to $p(\hat{\mathbf{x}}) = \prod_{i=1}^{n} p_{\hat{X}}(x_i)$. The generated sequences, so called sub-codeword $\hat{\mathbf{x}}(m)$, constitute the subcodebook $\mathcal{C}^{(k)} = \{\hat{\mathbf{x}}^{(k)}(1), \cdots, \hat{\mathbf{x}}^{(k)}(M)\}$ with probability

$$p(\mathcal{C}^{(k)}) = \prod_{m=1}^{M} \prod_{i=1}^{n} p_{\hat{X}}(\hat{x}_i(m)) \tag{1}$$

These randomly generated K subcodebooks consists of the codebook $\mathcal{C} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(K)}\}$ and are revealed to both encoder and decoder.

Encoding At a stage k, choose an index $m^{(k)} = \text{argmin}_{m \in [1:M]} \|\mathbf{x}^{(k)} - \hat{\mathbf{x}}^{(k)}(m)\|_1$. And send such $m^{(k)}$. For the next stage $k+1$, set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{x}^{(k)}(m^{(k)})$ and repeat it up to the last K iteration.

Decoding At a stage k receiving $m^{(k)}$, the decoder chooses the reproduction sequence $\hat{\mathbf{x}}^{(k)}(m^{(k)})$ and then recover a sequence $\hat{\mathbf{x}}^{(1:k-1)} + \hat{\mathbf{x}}^{(k)}(m^{(k)})$ sequentially where $\hat{\mathbf{x}}^{(1:k-1)} = \sum_{j=1}^{k-1} \hat{\mathbf{x}}^{(j)}(m^{(j)})$ is the recovered sequence up to the previous $k-1$ stage. Consequently, at the last stage K, the decoder reproduces the sequence $\sum_{k=1}^{K} \hat{\mathbf{x}}^{(k)}(m^{(k)})$

Analysis The total rate for K iterative stage is $R = R_K = K\frac{\log M}{n}$. The size of codebook is $K \times Mn$, the computational complexity is $o(K_n \times M_n \times n)$. For instance, if set by

$L(n) = \frac{n}{\log n}$ and $M = n$, then the scheme requires $\frac{n^3}{\log n}$ codebook size and $o(\frac{n^3}{\log n})$ computational complexity.

4.2 Performance Analysis

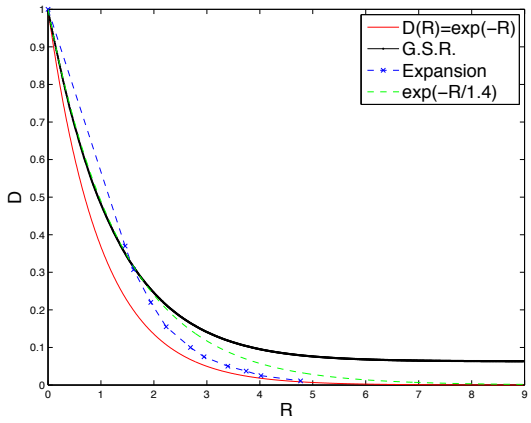

Figure 4: **Achievable rate distortion pairs using generalized successive refinement.** In this numerical result, we set $\lambda = 1$, $n = 10^3$, and $M = 10^3$ and averaged over 10 simulation. $R(D)$(red-solid) is the rate distortion function; G.S.R(black-solid) is achievable rate using our scheme 4.1; Expansion(blue-dashed) is achievable rate using expansion coding and time-sharing; and $e^{-R/1.4}$(green-dashed) is drawn for reference.

As in Figure 4, our scheme performs well in low rate. But it is not better than expansion coding in high rate. The degree of decaying becomes slower if referred to as the line $e^{-\frac{R}{1.4}}$. Note that, it converges to about 0.05 distortion point slightly above the 0 distortion point.

4.3 Suggested Improvement Points

4.3.1 fitting to right successive sources

For a successive refinement scheme in section 3 for Laplacian sources, we know that it can achieve the rate distortion function $R(D)$ of all rates for a sufficiently large n. This is theoretically guaranteed because, for each stage, it uses huge amount of subcodebook $e^{n\Delta R}$ where $\Delta R \geq I(X;X)$, $X \sim Lap(D_{k-1})$, and $\hat{X} \sim LapMixer(D_{k-1}, D_k)$. In addition, it uses joint typicality encoding in order to find a sequence satisfying Laplacian distribution with target distortion. But, the encoder in our scheme 4.1 simply chooses a message such that minimizing L-1 norm distortion on each stage instead of joint typicality encoding. Consequently, the normalized empirical distribution of a recovered sequence $\mathbf{x}^{(1:k)}$ is not guaranteed as a Laplacian source any more.

In Figure 4, the achievable curve decays like $e^{-R/1.4}$ at low rate but stop to follow it at high rate, which implies that $\mathbf{x}^{(1:k)}$ deviates from the expected $Lap(D_{k-1})$ as iterative stages go on. Moreover, we supposed that the normalized empirical distribution of

$\mathbf{x}^{(1:k)}$ might have a steeper decaying tail with more zeros than $Lap(D_{k-1})$. As in Figure 5, the normalized histograms at each rates turns out to have a similar distribution of $Lap(1.4 \times D_{k-1}) = \frac{1}{2 \times 1.4 \times D_{k-1}} e^{-R/(1.4 \times D_{k-1})}$ rather than $Lap(D_{k-1})$

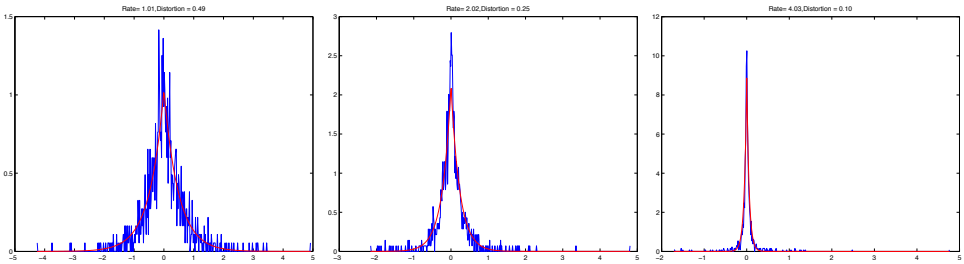

Figure 5: **Histograms of recovered sequences $\mathbf{x}^{(1:k)}$ at stage k.** Each represents the histogram at stage $k = 150$(left), $k = 300$(middle), 600(right) having the point $R_k = 1$, $R_k = 2$, and $R_k = 4$. Blue line indicates normalized histogram and red line indicates $Lap(1.4 \times D_{k-1})$

From this observation, we introduce a fitting parameter c. And changed the target curve $D(R) = \lambda e^{-R}$ into $D^c(R) = \lambda e^{-R/c}$. In result, we experimentally found that the fitting parameter $c \approx 1.4$ have the best performance within $1 \leq c \leq 3$ range and this value does not depend on the source parameter λ nor source size n.

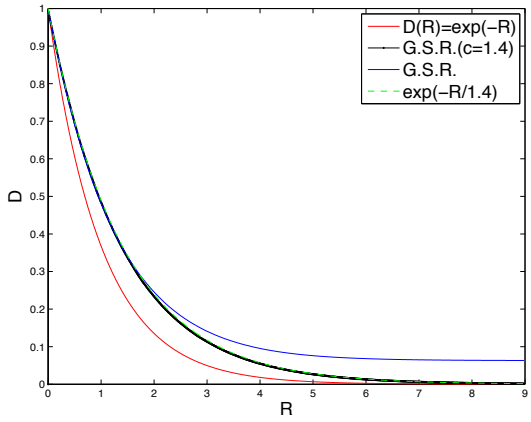

Figure 6: **Achievable rate distortion pairs using generalized successive refinement with fitting.** $R(D)$(red-solid) is the rate distortion function; G.S.R with $c = 1.4$ (black-solid) is achievable rate using the scheme with fitting 4.3.1; G.S.R(blue-solid) is achievable rate using the scheme without fitting 4.1; and $e^{-R/1.4}$(green-dashed) is drawn for reference

Analysis In Figure 7, our suggested scheme improves the performance by completely overlapping the fitted $D^c(R) = \lambda e^{-R/c}$ curve in high rate as well, resulting in converge to $D = 0$.

4.3.2 reducing the storage through permutation

Our scheme generating each subcodebook reduces the storage to be polynomial on n and the total storage $S^{\text{orig}}(K, M, n, |\mathcal{X}|) = K \times \log(|\mathcal{X}|^{Mn})$ nats is required. We can further reduce the storage by utilizing the fact that the big portion of each subcodebook is composed of zeros.

Codebook generation Firstly, for the k stage, generate the first sub-codeword $\hat{\mathbf{x}}^{(k)}(1)$ according to $LapMixer(D_{k-1}, D_k)$ as usual. Secondly, the rest of sub-codewords are generated by permutating $\hat{\mathbf{x}}^{(k)}(1)$ randomly. In other words, for a n by n random permutation matrix $P \sim unif[1 : \binom{n}{n(1-(D_k/D_{k-1})^2)} \times n(1-(D_k/D_{k-1})^2)!]$, use $\hat{\mathbf{x}}^{(k)}(m) = P \times \hat{\mathbf{x}}^{(k)}(1)$ for $m \in [2 : M]$.

Analysis The required storage for each permutated sub-codeword $\hat{\mathbf{x}}^{(k)}(m)$ for $m \in [2 : n]$ is given by

$$\begin{aligned}
&\log\left(\binom{n}{n(1-(D_k/D_{k-1})^2} \times n(1-(D_k/D_{k-1})^2)!\right) \\
&\quad = \log\left(\frac{n!}{n(D_k/D_{k-1})^2! \times n(1-(D_k/D_{k-1})^2)!} \times n(1-(D_k/D_{k-1})^2)!\right) \\
&\quad = \log\left(\frac{n!}{n(D_k/D_{k-1})^2!}\right) \\
&\quad = \log\left(\frac{n!}{ne^{-\frac{2\log M}{n}}!}\right)
\end{aligned}$$

Therefore, the required total storage is roughly $S^{\text{imprv}}(K, M, n, |\mathcal{X}|) = K \times \Bigg(\log|\mathcal{X}|^n + (M-1)\log\left(\frac{n!}{ne^{-\frac{2\log M}{n}}!}\right)\Bigg)$. With our setting $M = n$, the storage improvements can be indicated as the $\text{ratio}(n, M, |\mathcal{X}|) = S^{\text{orig}}(K, M, n, |\mathcal{X}|)/S^{\text{imprv}}(k, M, n, |\mathcal{X}|)$ as shown in 1.

Table 1: Storage saving via permutation codebook

n	M	$\lvert\mathcal{X}\rvert$	$\text{ratio}(M, n, \mathcal{X})$
10^3	10^3	30	36.5
10^3	10^3	50	41.8
10^4	10^4	50	230
10^5	10^5	50	1455

Figure 7 shows that performance of permutation code is very close to that of the previous scheme 4.3.1 in low rate.

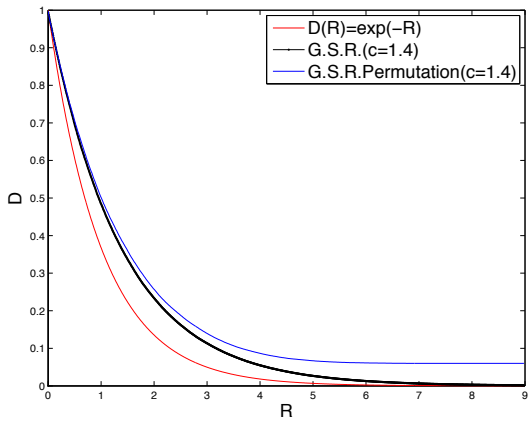

Figure 7: **Achievable rate distortion pairs using generalized successive refinement with fitting and permutation codebook.** $R(D)$(red-solid) is the rate distortion function; G.S.R with $c = 1.4$ (black-solid) is achievable rate using our scheme 4.3.1 with fitting; and G.S.R Permutation with $c = 1.4$(blue) is achievable rate using our scheme 4.3.2 with fitting and permutation codebook.

5 Generalized Successive Refinement Scheme with Ternary Alphabets

Even though the codebook size is significantly reduced through the scheme 4.1, 4.3.1, and 4.3.2 compared with the existing scheme 3, it still requires a huge storage for a continuous codebook. Thus, we suggest a more practical scheme only using finite alphabets in a codebook.

Remember that $LapMixer(D_{k-1}, D_k)$ in Figure 3 is used for generating a subcodebook in the section 4. And for large source size n, the rate increment $\Delta R = \log(M)/n$ between stages would be small enough to be told $D_k/D_{k-1} = e^{\Delta R} = e^{-\frac{\log M}{n}} \approx 1$. This means that $(D_k/D_{k-1})^2 \approx 1$ portion of the subcodebook consists of zero and the remaining negligible $1-(D_k/D_{k-1})^2 \approx 0$ portion of the subcodebook consists of alphabets following $Lap(D_{k-1})$. In result, we can expect that discretizing Laplacian portion into 1 bit alphabet would have only a negligible effect on additional distortion.

5.1 Choose a good discrete alphabet

Here, we wanted to select 1 bit symmetric alphabet in addition to '0' alphabet which behaves like $LapMixer$ in terms of L-1 norm preservation. Thus, we define a PMF $BinaryMixer(\lambda, D)$ given by

$$p_{\hat{X}}(\hat{x}) = \begin{cases} (D/\lambda)^2 & \text{if } \hat{x} = 0 \\ \frac{1}{2}(1 - D^2/\lambda^2) & \text{if } \hat{x} = \lambda \text{ or } -\lambda \end{cases}$$

Note that, for n i.i.d $BinaryMixer(\lambda, D)$ source $\mathbf{x}_B$ and $LapMixer(\lambda, D)$ source $\mathbf{x}_L$, both of L-1 distortion are close, *i.e.*, $\frac{1}{n}\|\mathbf{x}_B\|_1 \approx \frac{1}{n}\|\mathbf{x}_L\|_1 \approx (1-(D/\lambda)^2)\lambda$.

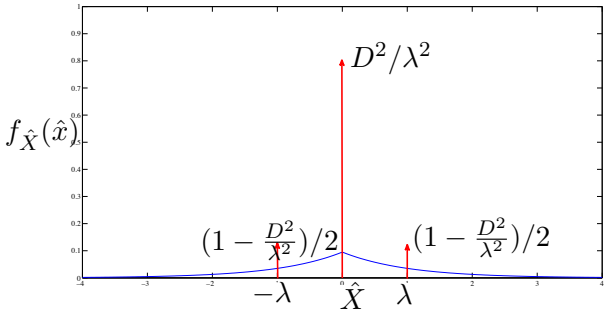

Figure 8: **BinaryMixer(λ, D)**

5.2 Scheme

Notation For an i.i.d $Lap(\lambda)$ source $\mathbf{x}$, fix the number of iteration K and the subcodebook size M. For each iteration stage $k \in [1:K]$, define $\Delta R = \frac{\log M}{n}$, $R_k = k \times \frac{\log M}{n}$, and $D_k = \lambda e^{-R_k}$.

Codebook generation: For each stage $k \in [1:K]$, fix $p(\hat{x}) = BinaryMixer(D_{k-1}, D_k)$ instead of $LapMixer(D_{k-1}, D_k)$.

The rest of part of Codebook generation, Encoding, Decoding rule are the same as in the scheme 4.1

5.3 Performance Analysis

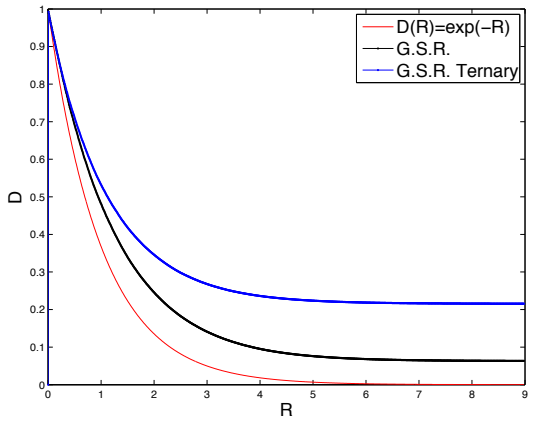

Figure 9: **Achievable rate distortion pairs using generalized successive refinement with ternary alphabet.** $R(D)$(red-solid) is the rate distortion function; G.S.R(black-solid) is achievable rate using the scheme with ternary alphabet in 5.2; and G.S.R(blue-solid) is achievable rate using the scheme with continuous alphabet in 4.1

As in Figure 9, this suggested scheme performs well in the rage of low rate. Also, the performance of this ternary discretization is close to that of continuous alphabet scheme 4.1 in low rate. But the gap between two schemes becomes bigger as rate increases. In addition, this scheme saturates at 0.2 unable to converge on $D = 0$ for a high rate.

Considering the fact that the scheme is reachable on the target distortion $D = 0$ can be one of the significant basis of a good lossy compressor, the next approach improving this saturation issue is necessary.

5.4 Suggested Improvement Points

5.4.1 fitting to right successive sources

We supposed that, like in the scheme 4.3.1, this saturation could have been originated from the fact that the normalized histograms after k iteration stage would be far way from the expected $Lap(D_{k-1})$ distribution. Thus, we introduce a fitting parameter c again. And changed the target curve $D(R) = \lambda e^{-R}$ into $D^c(R) = \lambda e^{-R/c}$.

Analysis In result, we experimentally found that there is trade off between performance in low rate and performance in high rate depending on c.

- for a large $c > 2.5$, it works well in high rate and does not saturate.
- for a small $c < 2$, it works well in low rate but saturates.

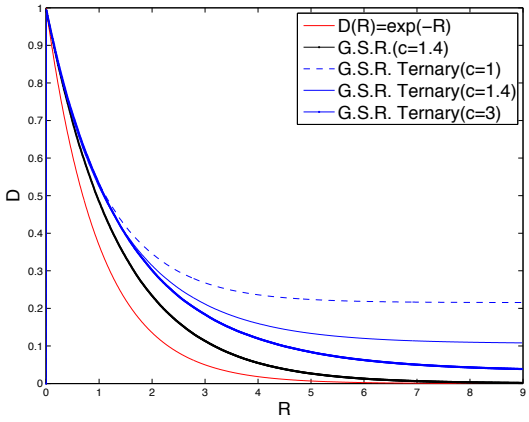

Figure 10: **Achievable rate distortion pairs using generalized successive refinement with ternary alphabet and fitting.** $R(D)$(red-solid) is the rate distortion function; G.S.R with $c = 1.4$ (black-solid) is achievable rate using the scheme with fitting in 4.3.1; and G.S.R Ternary's(blue) are achievable rate using the scheme with ternary alphabet and fitting in 5.4.1.

5.4.2 reducing the storage through permutation

The idea and formula for storage calculation are the same as in 4.3.2. Table 2 shows roughly 3 times storage saving in case of Ternary alphabet compared with 30 alphabet. See Figure 11 for the performance.

Table 2: Storage saving via permutation codebook

n	M	$\lvert\mathcal{X}\rvert$	ratio$(M, n, \mathcal{X})$
10^3	10^3	3	12.1
10^3	10^3	30	36.5
10^5	10^5	3	413
10^5	10^5	30	1282

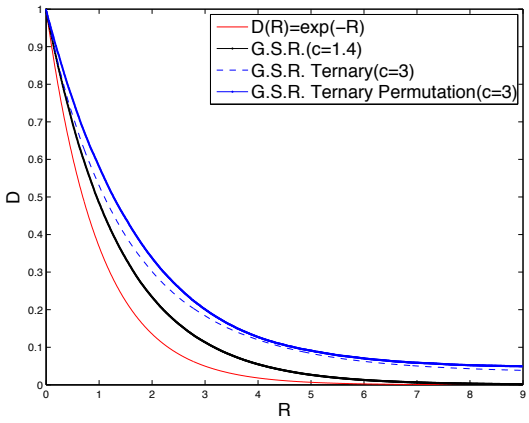

Figure 11: **Achievable rate distortion pairs using generalized successive refinement with fitting, permutation codebook, and ternary alphabet.** $R(D)$(red-solid) is the rate distortion function; G.S.R with $c = 1.4$ (black-solid) is achievable rate using our scheme 4.3.1 with fitting; G.S.R Ternary with $c = 3$(blue dashed) is achievable rate using our scheme 5.4.1; and G.S.R Permutation with $c = 3$(blue) is achievable rate using our scheme 5.4.2 with fitting and permutation codebook.

6 Sending the max index and the min index

The two schemes in 4 and 5 are based on random codebook generation approach. The codebook of sequences must be shared between the encoder and the decoder. We suggest another scheme that requires a smaller codebook size.

6.1 Scheme

Codebook generation A set of nonnegative numbers $\{\alpha_1, \alpha_2, \cdots, \alpha_K\}$ is known to the decoder. Each α_k is the median of kth largest number in $x^n \sim$ i.i.d. $Lap(\lambda)$.

Encoding At a stage k, the encoder sends the index of the maximum value and the index of minimum value of x, i.e. $m^{(k)} = (m_1^{(k)}, m_2^{(k)})$, where

$$m_1^{(k)} = \underset{i\in[1:n]}{\operatorname{argmax}}\, x_i^{(k)}, m_2^{(k)} = \underset{i\in[1:n]}{\operatorname{argmin}}\, x_i^{(k)}$$

For the next stage $k+1$, set $x^{(k+1)}$ according to

$$x_i^{(k+1)} = \begin{cases} x_i^{(k)} - \alpha_k & \text{if } i = m_1^{(k)}, \\ x_i^{(k)} + \alpha_k & \text{if } i = m_2^{(k)}, \\ x_i^{(k)} & \text{otherwise.} \end{cases}$$

Decoding At a stage k receiving $m^{(k)} = (m_1^{(k)}, m_2^{(k)})$, the decoder chooses the reproduction sequence $\hat{x}^{(k)}(m^{(k)})$ according to

$$\hat{x}_i^{(k)}(m^{(k)}) = \begin{cases} \alpha_k & \text{if } i = m_1^{(k)}, \\ -\alpha_k & \text{if } i = m_2^{(k)}, \\ 0 & \text{otherwise.} \end{cases}$$

Analysis The rate for each step is not a constant. At step k, message $(m_1^{(k)}, m_2^{(k)})$ is uniform among $n - 2(k-1)$ indices, i.e., indices that were not received upto the previous step.

$$R_1 = 2\log\binom{n}{2}/n, \quad R_2 = R_1 + 2\log\binom{n-2}{2}/n, \quad \ldots, \quad R_k = \sum_{i=1}^{k} 2\log\binom{n-2i+2}{2}/n$$

6.2 Performance Analysis

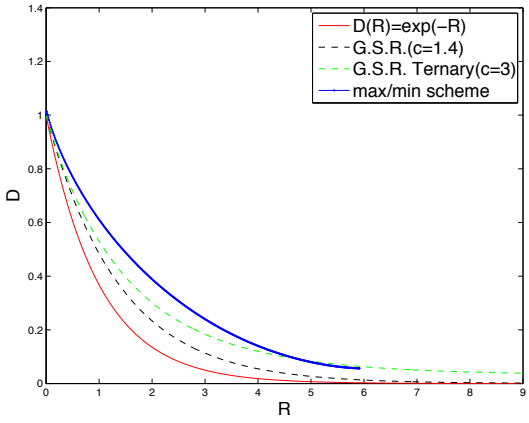

Figure 12: **Achievable rate distortion pairs using max/min scheme**

The performance of deterministic scheme is worse than the random coding scheme. There are couple of facts we believe to be the reason. This scheme estimates the kth largest and smallest index by the median of kth largest and smallest index. When k is small, the kth largest and smallest index does not vary much, so the scheme works well. As k becomes large, it varies much, so approximating it by the median does not perform well. We think the scheme might perform better for a larger n.

This scheme has much smaller complexity. Thus, it is not surprising that this scheme works worse than the others. Also, we think the takeaway message is that exploiting randomness is good.

7 Application of our Schemes to Image Data

We test our scheme on an image data. We construct the source sequence $\mathbf{x}$ as follows:

1. Download Lena 512 by 512 image, and resize the image to a 44 by 44 image (shown in Figure 13).
2. Take FFT of the 44 by 44 image.
3. Take the imaginary part of FFT coefficients to construct $\mathbf{y}$(shown in right side of Figure 14).
4. Scale $\mathbf{y}$ by the L-1 norm of $\mathbf{y}$ and obtain $\mathbf{x}$, i.e., $\mathbf{x} = \mathbf{y}/\|\mathbf{y}\|$.

Figure 13: **Lena image 44 by 44**

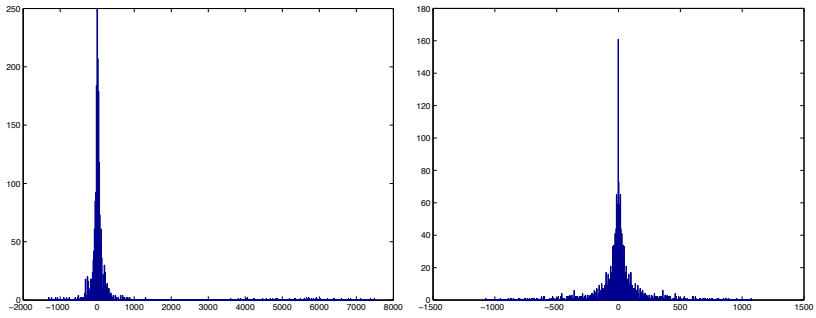

Figure 14: **FFT coefficients: Real part(left) and Imaginary part(right)**

7.1 Performance Analysis

As shown in Figure 15, the achievable rate distortion curve is close to the curve for iid simulated Laplacian source. It works slightly better than the iid source. Although we do not exploit the correlation of data, the achievable rate distortion curve is similar to the one for iid Laplacian source.

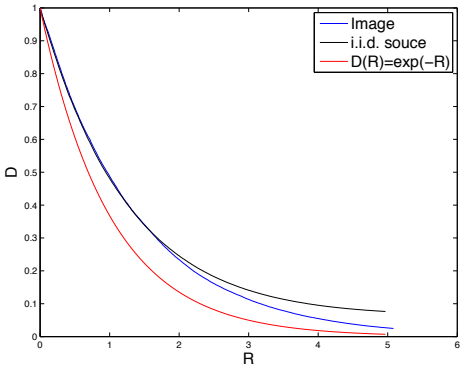

Figure 15: **Rate distortion region for real image vs. i.i.d source $\sim$ Laplacian**(1)

7.2 Comments

Our result does not guarantee that our scheme will work nicely on the real image data. More experiments need to be done on various images. We also applied our scheme only on the imaginary part of FFT coefficients. That was because the real part of FFT coefficients did not follow Laplacian distribution closely. We think there would be a way to deal with this. We would like to do more work in this direction.

8 Universality: for a Gaussian Source

We apply schemes to an i.i.d. Gaussian sequence. Rate distortion function for Gaussian source with L-1 norm is unknown, so we plot Shannon lower bound instead.

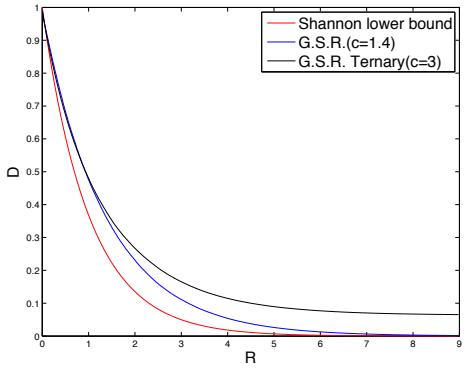

Figure 16: **Rate distortion regions of our scheme on Gaussian i.i.d. source**

9 Conclusion

We conclude that the generalized successive refinement with continuous alphabets along with fitting in 4.3.1 has the best performance among our schemes and even better than expansion coding scheme in low rate. Also, the generalized successive refinement with ternary alphabets along with fitting and permutation codebook in 5.4.2 requires the lowest complexity and storage among our schemes. Since the expansion coding scheme performs well in high rate, we can use time sharing in order to attain optimality of both schemes. But, in this case, we cannot but lose rateless property for high rate.

Since our scheme works well for Gaussian source, we have a hope for our schemes to be universal. The further works about whether any sources having the symmetric and exponential tail distribution are applicable or not can be worked on.

Due to the fact that real data including images might contain many large components, which makes real data deviate from Laplacian distribution, our schemes are sometimes not efficient to be applied into those data. Thus, we would like to check if the picture of nature would be a better applicable source or the scheme becomes efficient after filtering high components.

10 Acknowledgement

We greatly thank Albert No.

References

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York: Wiley, 1991.

[2] A. J. Viterbi and J. K. Omura, "Trellis encoding of memoryless discrete time sources with a fidelity criterion," *IEEE Trans. on Information Theory*, vol. 20, no. 3, pp. pp. 325–332, 1974.

[3] Y. Mastungaga and H. Yamamoto, "A coding theorem for lossy data compression by ldpc codes," *IEEE Trans. on Information Theory*, vol. 49, no. 9, pp. pp. 2225–2229, 2003.

[4] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. on Information Theory*, vol. 55, pp. pp. 3051–3073, Jul. 2009.

[5] S. B. Korada and R. L. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Trans. on Information Theory*, vol. 56, pp. pp. 1751–1768, Apr. 2010.

[6] R. Venkataramanan, T. Sarkar, and S. Tatikonda, "Lossy compression via sparse linear regression: Computationally efficient encoding and decoding," *IEEE Trans. on Information Theory*, vol. abs/1212.1707, 2012.

[7] S. B. Korada and R. L. Urbanke, "An mcmc approach to universal lossy compression of analog sources," *IEEE Trans. on Signal Processing*, vol. 60, no. 10, pp. pp. 5230–5240, 2012.

[8] O. K. H si and S. Vishwanath, "Lossy compression of exponential and laplacian sources using expansion coding," *IEEE Trans. on Information Theory*, vol. abs/1308.2338, 2013.

[9] A. El Gamal and Y. H. Kim, *Network Information Theory*. Cambridge University Press, 1st ed., 2011.