An Approach to Code Generation for Trellis-Coded Quantization Based on Geometrically Uniform Codebooks

Lorenzo Cappellari
<lorenzo.cappellari@dei.unipd.it>

Department of Information Engineering
University of Padova, Italy

Workshop “Toni Mian” - Padova, 26 Ottobre 2007
What is Trellis-Coded Quantization (TCQ)?

Past & Present of TCQ

TCQ Design with a Maximum Distance Criterion

Experiments & Comparisons

Conclusions
Source Coding

Max Distortion

Typical Set
Source coding under a squared-error constraint is a sphere covering problem.

Proofs of asymptotic limits involve random reconstruction codebooks [Shannon’s theory, 1948].

Feasible, high-dimensional vector quantizers rely on structured reconstruction codebooks:
- lattice quantization
- trellis-coded quantization

At high rates, the decision regions of an optimal vector quantizer are all asymptotically congruent with each other [Gersho’s conjecture, 1979].
Trellis-Coded Quantization

TCQ uses a one-dimensional codebook and a shift-register

- the codebook is partitioned into sub-codebooks
- using a binary convolutional code, each state transition of the shift-register is labeled with one of the sub-codebooks
- given a random realization, the corresponding optimum path on the trellis diagram is found (using the Viterbi algorithm)

If the codebook/sub-codebooks are sufficiently regular (e.g. lattices), the decision regions of the equivalent reconstruction codebook are all congruent with each other [Forney, 1991]
TCQ has first appeared [Marcellin et al., 1990] as the natural counterpart, in the source coding domain, of trellis-coded modulation (TCM) techniques [Ungerboeck, 1982].

- TCQ uses the same “good” convolutional codes (empirically designed) used in TCM.

- In origin, focus was on codebook/sub-codebooks optimization.
  - at low rates, the codebook can be optimized using iterative algorithms.
  - in entropy-constrained systems, the rate can be taken into account as well during the optimization [Marcellin, 1994].
In any case, the size $M$ of the initial codebook puts (at rates close to $\log_2 M$ bit/sample) a non-negligible limit on the achievable performance [Pearlman et al., 1980].

Recent approaches investigate the opportunity to use larger codebooks:

- more random, non-stationary labels can be used on the trellis diagram [Anderson et al., 2005]
- it seems that no performance gains are obtained from changing the very simple shift-register transitions [Eriksson et al., 2007]
Our research on TCQ aimed to:

- investigate the high rate performance of regular ("geometrically-uniform") TCQ structures
- optimize the convolutional codes rather than the actual codebook/sub-codebooks structure

Using (entropy-coded) regular TCQ structures:

- structure of the sub-codebooks $\rightarrow$ coarse-grain tessellation of the space
- convolutional code $\rightarrow$ fine-grain tessellation (i.e. final performance)
- entropy coding $\rightarrow$ takes eventually care of unused codewords (i.e. the ones outside the typical set)
Sphere Covering with Regular TCQ

Coarse Region
(sub-codebooks structure)

Typical Set
Considering that each codeword of the convolutional code corresponds to a different point (reconstruction codeword) in each coarse region, the following strategy was devised:

- use maximum-Hamming distance binary codes
- assign more distant points (in Euclidean space) to more distant binary codewords using a proper sub-codebook labeling
Sphere Covering with Regular TCQ

Maximized Distances

Coarse Region
(sub-codebooks structure)

Typical Set
Asymptotic gain over EC-USQ (in dB) for the partition $\mathbb{Z}/4\mathbb{Z}$ (1000 coded samples).

<table>
<thead>
<tr>
<th>states</th>
<th>Ungerboeck codes</th>
<th>distance-optimal codes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>code</td>
<td>granular gain</td>
</tr>
<tr>
<td>64</td>
<td>[103 024]</td>
<td>1.282 ± .0012</td>
</tr>
<tr>
<td>128</td>
<td>[235 126]</td>
<td>1.332 ± .0010</td>
</tr>
<tr>
<td>1024</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Experiments & Comparisons

\begin{align*}
\gamma_g(S) \text{ (dB)}
\end{align*}

\begin{align*}
\gamma_g(S) \text{ (dB)} & = [561 \ 753] \\
\gamma_g(S) \text{ (dB)} & = [133 \ 171] \\
\gamma_g(S) \text{ (dB)} & = [23 \ 35]
\end{align*}

Dimensionality of S

\begin{align*}
10^1 & \quad 10^2 & \quad 10^3
\end{align*}
Asymptotic gain over EC-USQ (in dB) for the partition $\mathbb{Z}^2/2\mathbb{Z}^2$ (1000 coded samples).

<table>
<thead>
<tr>
<th>states</th>
<th>Ungerboeck codes</th>
<th>Granular gain</th>
<th>distance-optimal codes</th>
<th>Granular gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[5 2]</td>
<td>0.981 ± 0.0018</td>
<td>[5 7]</td>
<td>0.980 ± 0.0019</td>
</tr>
<tr>
<td>8</td>
<td>[13 04]</td>
<td>1.074 ± 0.0017</td>
<td>[13 17]</td>
<td>1.074 ± 0.0017</td>
</tr>
<tr>
<td>16</td>
<td>[23 04]</td>
<td>1.124 ± 0.0016</td>
<td>[23 35]</td>
<td>1.137 ± 0.0015</td>
</tr>
<tr>
<td>32</td>
<td>[45 10]</td>
<td>1.171 ± 0.0015</td>
<td>[53 75]</td>
<td>1.191 ± 0.0015</td>
</tr>
<tr>
<td>64</td>
<td>[103 024]</td>
<td>1.225 ± 0.0014</td>
<td>[133 171]</td>
<td>1.243 ± 0.0014</td>
</tr>
<tr>
<td>128</td>
<td>[235 126]</td>
<td>1.272 ± 0.0014</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>256</td>
<td>[515 362]</td>
<td>1.303 ± 0.0013</td>
<td>[561 753]</td>
<td>1.311 ± 0.0013</td>
</tr>
<tr>
<td>1024</td>
<td>–</td>
<td>–</td>
<td>[2335 3661]</td>
<td>1.351 ± 0.0013</td>
</tr>
</tbody>
</table>
Experiments & Comparisons

Dimensionality of $S$

$\gamma_g(S)$ (dB)

- [561 753]
- [133 171]
- [53 75]
- [23 35]
Signal-to-quantization noise ratio using optimized alphabet $\mathbb{Z}/4\mathbb{Z}$-TCQ ($M$ states, 1000 uniformly distributed input samples).

<table>
<thead>
<tr>
<th>rate bit/sample</th>
<th>$M$</th>
<th>SQNR (dB)</th>
<th>traditional codes</th>
<th>proposed codes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16</td>
<td>6.405 ± .0029</td>
<td>6.412 ± .0029</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>6.496 ± .0028</td>
<td>6.519 ± .0027</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6.579 ± .0027</td>
<td>6.593 ± .0026</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>12.802 ± .0023</td>
<td>12.812 ± .0023</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>12.915 ± .0021</td>
<td>12.936 ± .0021</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>13.012 ± .0019</td>
<td>13.022 ± .0019</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>19.024 ± .0019</td>
<td>19.033 ± .0019</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>19.142 ± .0017</td>
<td>19.157 ± .0017</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>19.236 ± .0015</td>
<td>19.243 ± .0015</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

- Previous approaches: alphabet optimization/larger alphabets
- Our objective: optimize the convolutional codes (at high rates)
- Proposed technique: straightforward idea of taking advantage of distance-optimal codes
- Results: rediscovering of some “empirical” Ungerboeck codes/slight improvements over other ones