Introduction to and Instructions for the EECS 651 Term Project

Projects typically:

- Are concerned with lossy, as opposed to lossless, coding. Thus they deal with rate and distortion, as opposed to just rate. However, lossy coding projects often deal with variable-length coding, in which case they have a substantial lossless coding component.
- Will focus on some particular source or coding technique or fidelity criterion or analysis or complexity contraint or some other specified aspect or constraint.
- Have a strong experimental, e.g. computer implementation component. But purely theoretical projects are also possible.
- Will be related to previous work found in books and/or papers.
- Keep it simple. They work on the simplest version of a source coding system or issue that permits something interesting to be investigated.

Projects must:

- Have a well specified focus and goal.
- Have a strong connection to the material presented in the course. For example, in so far as possible, the project should be described using the terminology and concepts introduced in the course (as opposed to the terminology and concepts particular to some paper or book). Connections between the subject of the project and the course should be recognized. Analysis methods developed in the class should be used whenever possible. Comparisons of the system being considered in the project to those studied in class should be made.
- Suggest, analyze, and/or experiment with alternative parameters, or analysis methods. Questions include: Why is the system designed the way it is? Why does it work as well or as poorly as it does? Could it be improved? Is the performance of the system explanable? Does the performance match analytical predictions? Does the system performance relate to other systems in some way that can be explained? Try to have an interesting "story" to tell -- issues, hypothesized approach, positive and negative results.
- Critically assess the design and performance of the system being studied. Be critical, judgmental, quantitative. Do "engineering" where possible. It is more important that the experiments and assessments be done well, than it is for the system to have outstanding performance. But you don't want to be testing or analyzing a naive system.

Projects may:

- Use software developed elsewhere. The project is not intended as a test of your programming skills. Rather it should make use of your engineering/experimentation/analysis/interpretation skills. Your project report will describe what software you wrote and what prewritten software you used.
- Closely follow old methods. You need not propose something new. There is always plenty of room for experimentation and analysis, even with the simplest and oldest of methods or issues.

Groups

- Group projects are strongly encouraged. Groups of size 2, 3 and 4 are appropriate. If you are looking for other students in the class with similar or complementary interests, send email to me and I'll forward it to the class.

Deliverables

- A project proposal: Due Wednesday, April 4, in class, before the lecture begins.

  This should spell out in as much detail as possible the scope of your project (overview, objectives, goals, source, coding technique, fidelity critieria, experiments, analysis methods, benchmarks, software and source material to be obtained or created, etc.) This is not the sort of thing you do the night before the due date. And you should not consider that you start your project after the proposal is graded. Rather your project begins now and you should invest much time in your proposal, so that your project will go as smoothly as possible.

- An oral presentation: At a date and time to be determined. Groups of 3 will be given something like 25 minutes, and groups of larger or smaller sizes will have a little more or less time.

- A written report: One per group. Due on the last day of finals, by 4PM, Thursday, April 26.

  The report should be written for students in the class. That is, it presumes the students understand the class material, but have not seen the new material that you have found or developed. They should be able to learn your topic from your report. The description should be sufficiently detailed that they can repeat what you have done.

- Each group member will individually submit a confidential summary of the contributions of each member of the group, inlcuding his/herself.

Feedback

- To get feedback and suggestions, each group must meet with me three times:

  (1) by the end of Friday, March 30, to discuss preliminary plans. (A signup sheet will be posted on my door.)

  (2) after the proposal is graded

  (3) after the oral report.

Grading will be based on

- The proposal.
- The final project report, as well as the oral presentation.
- Factors include: Quantity, quality, degree of difficulty of the work. The analytical or experimental results. The quality of your critical judgments and analyses. The depth of understanding displayed. Innovativeness. Connectedness to the course.
- The project counts 35% of the overall class grade. The proposal counts 5%, and the oral and written portions count 30%.
- If all members of a group contribute roughly equally to the project, they will all receive the same project grade. If some group members contribute significantly more than others, then adjustments may be made.

1. Some Potential Real-World Sources

a. Speech

CELP is the most widely used high performance type of speech coder. There are many variations of CELP and many parameters to experiment with.

We have some files of speech samples. One can easily create more. It should be possible to get standard speech coders to experiment with or modify.

b. Audio/Music

Audio coders typically use transform or subband coding. It is particularly interesting how modern audio coders use perceptual models of hearing to produce decoded reproductions with low SNR, e.g. 10 to 20 dB, that sound just like the original.

We have a files of music, more can be obtained. It is possible to get audio coding software to experiment with.

c. Images

Image coding makes for very nice projects because it is easy to see the results. Wavelet coding methods seems to be the current best combination of performance and complexity. JPEG 2000 is a recent wavelet based image coding standard.

We have access to lots of images. We have access to JPEG source code. There are many image coders to be found at various websites. An image coding project could be oriented to some particular type of image, e.g. Xray images.

There is room for a number of projects related to JPEG or JPEG 2000, involving a number of people.

d. Video

Video coding continues to be a major frontier area of compression. Modern standardized methods use motion compensated prediction and transform coding of the prediction error. MPEG and H.263 are two very well known video coding standards. A new standard, H.264, makes significant improvements over those two. There are also wavelet based video coders. Generally, a video coding project will deal with a relatively small number of frames of a video sequence in order to avoid having to deal with huge amounts of data.

We have access to several video sequences, and more can be obtained on the web. Software implementing standard video coders is available.

There is room for a number of projects related to the new standard, involving a number of people.

3. Random Process Source models

Many projects will need a random process source model for design purposes, or they may analyze a coding technique on a tractable model Some projects might focus on source modeling itself; i.e. trying to match source models to sources in order that codes or theory developed for the model will accurately predict the performance of codes on actual sources. Sample models include:

Markov, autoreregressive (AR), moving average (MA), ARMA, and hidden Markov. Gaussian and Laplacian densities are widely used because of their reasonableness and tractability.

4. Fidelity criteria

If you're doing quantization, you'll have to choose one. Or a project could focus on fidelity criteria themselves. For example, what image or speech fidelity criteria most accurately reflect human perception. The big advances in high fidelity compression of music (e.g. MP3) is based on models of human hearing. Related to this is the development of "perceptually" based codes; i.e. codes that are designed to "sound good" or look good" but which need not have good SNR. For example, how should a transform code be optimized for good rate vs. perception, as oposed to rate vs. SNR.

5. Types of codes

Many projects will focus on some particular coding technique and how to tune it up (choose its parameters) or to modify it to improve performance or reduce complexity. In class we will consider (at least briefly) a number of additional types of codes beyond those we have already discussed.

For example, here are some types of structured vector quantizers: transform, multistage, tree-structured, lattice, scalar-vector, block-constrained, polar, gain-shape, hierarchical table lookup, tree, trellis, forward adaptive, backward adaptive.

6. Experiments

Most projects will probably involve a computer implementation and testing of some source coding method. A source, code, fidelity criteria and design methodology will have to be selected.

7. Analysis

The goal of source coding theory is to predict how well a given code will work on a given source, how well the best codes can work, and how to design good codes. Most projects should include some analysis of code performance, possibly even estimating the fundamental limits of the specific coding technique on the given source. One can also do nonformulaic analysis, wherein one critically assesses the results of experiments by comparing to appropriate benchmarks. However, wherever possible each project should attempt to use some quantitative theory in the design and/or analysis

8. Complexity

The principal issue in code design is performance at a reasonable complexity. So most projects will make an effort to quantify the complexity of the method and, where possible, to minimize complexity . Some projects could focus on optimizing a method for the best performance/complexity tradeoffs.

9. Robustness

One might wish to consider the robustness of a coding technique to variations in the source properties or to errors in transmission of the bits from encoder to decoder.

12. Critical Assessments: Whatever you do, your project report should be filled with critical assessments and judgments. That is, you don't simply implement something and show it to us. Instead you leave us with insights, conclusions and take away messages.

13. Facilities

CAEN computers will probably be commonly used, but any computer you can get your hands on is OK. Beware that for some projects, PC's and Mac's may not have sufficient power, although this is much less true than it was a few years ago.

We do have programs that implement the generalized Lloyd VQ design algorithm, and also encoding and decoding algorithms. Another program implements the standard TSVQ design. (And there are TSVQ encoding and decoding programs.) Some standard speech, audio, image, video and lossless coding algorithms are available or can be found on the web. It is possible for you to build upon these, or use them as benchmarks.

14. Some additional potential project foci:

Noisy channel quantization: Design and/or analyze lossy source codes that must operate in the presence of bit errors. For example, model the channel as binary symmetric channel. Or combine source and channel coding, or source coding and modulation.

Noisy source quantization: Suppose source X is corrupted by noise before it reaches your encoder. Design the encoder/decoder to get the best rate/distortion performance, where distortion

is measured between the decoded output and the original uncorrupted source output X.

Progressive/embedded/layered coding: Suppose the encoding must be done so that the encoded bits can be grouped into layers so that a rough approximation can be decoded from the first layer, a better approximation from the first two layers, and so on. Many common coding techniques can be made progressive, for example with transform coding, one could send the low frequency coefficients first. Sometimes the encoder can be structured to produce the layers one at a time in order of significance. Sometimes the encoder determines all layers at the same time. General question: Does layering limit performance? If so, by how much?

Multiple description source coding: Here the encoder produces two or more encoded versions of the source data such that if only one is received, a satisfactory approximation can be decoded. However, if two or more are received, then better approximations will be produced. This is useful, for example, when source encoder output is packetized for internet transmission, and the internet delivers some but not all packets. How can one get the best possible performance from the multiple encoded versions while maintaining satisfactory approximations from just a subset?

Distributed encoded: Suppose two highly correlated sources at different physical locations are each to be source encoded. It is possible to exploit their correlation to reduce rate and/or distortion, even though the encoders at each site see only the output of one source.

Complexity constraints: Source code design with special complexity constraints, such as that the decoder be especially simple, or that the encoder be especially simple. How does the performance of a quantizer relate to its complexity?

Continuous-time sources: Real-world continuous-time (or continuous-space) sources are not strictly bandlimited. So there is not usually an obvious sampling rate to choose. How do source coding performance (in terms of MSE and rate in bits/sec) and complexity (in terms of storage and op's per second) depend on the sampling rate? For example, with coding techniques like DPCM or transform coding, how will the design and performance depend on sampling rate?

Simultaneous quantization and estimation/detection: Design quantizers with the goal of being able to estimate or detect something (like the presence or absence of some pattern) from the decoded data that one could estimate/detect from the original source data in addition to or instead of trying to minmize decoder distortion.

Proposal Guidelines

Proposals should contain a detailed outline of what you propose to do. A clearly goal statement.

If your project focuses on a particular type of source code, make sure you should clearly specify its structure.

For example, if you are planning to design or implement some coding technique, then you should clearly specify the code strucure using words, formulas, and/or block diagrams. You will also indicate what are the parameters that are left unspecified now, but will be chosen later. For example, if you were implementing transform coding you might show block diagram of the encoder and decoder and indicate that the parameters to be specified later were the dimension k, transform T, and scalar quantizers for the transform coefficients, where the latter are specified by thresholds levels, binary codewords and levels. Also indicate what variations you plan to consider on the code structure, for example, fixed and variable-rate coding, two-dimensional quantizers instead of scalar quantizers.

You should indicate how performance will be measured. What source model or data you plan to design the code for.

How do you plan to design the code, i.e. choose the parameters. If you have an algorithm in mind, describe it. If it's based on training data, what training data will you use.

Indicate what programs you will write yourself, what existing software you will use,

What results you hope to present. For example, results might include plots of performance vs. various choices of parameters or various changes to the structure.

What high level questions you plan to ask?

What theory you plan to use. e.g. what theoretical calculations from the course you might be able to use.

What theory or design methods you hope to develop. For example, do you need to develop optimality conditions for some code structure for which there are none.

What critical assessments you plan to make.

You are not bound to do what you propose. But your proposal will be judged on the soundness of what it proposes.

A proposal from a 3 person group might typically be 5 to 15 pages or so, double spaced, including figures.