1 System Design and Implementation

In the course of this project, we began by implementing a C program to perform simple Vector Quantization, and adapted it to an implementation of Channel-Optimized Vector Quantization by modifying the optimality conditions as was outlined in the Description section. We then wrote a C implementation of the I→J system without a channel. To interface with images, we wrote a Python wrapper program. We also used GNU Octave, MathWorks MATLAB, and Python to produce data and create plots.

In the subsequent sections we will detail the design challenges that were faced along the way and how they were addressed.

1.1 Conditions of Optimality

Expanding for the optimality expressions given in [Description], we get that the optimal reconstruction of X, given fixed encoders and received codebook indices k, l is given by

$$x_{(k,l)} = \frac{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} E[X|X \in R_i, Y \in S_j]P(k,l|i,j)}{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} P(X \in R_i, Y \in S_j)P(k,l|i,j)} \tag{1}$$

$$= \frac{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} P(k,l|i,j)\int_{x\in R_i} x\frac{f_X(x)}{P(X\in R_i, Y\in S_j)}dx}{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} P(X \in R_i, Y \in S_j)P(k,l|i,j)} \tag{2}$$

Let $\mathcal{T}$ be a finite training set of ordered pairs drawn from the joint distribution for X and Y. Letting M_{ij} be the number of training vectors in $R_i \times S_j$ we can thus approximate the integral and probabilities as follows:

$$x_{(k,l)} = \frac{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} P(k,l|i,j)\sum_{x\in R_i} x\frac{1}{M_{ij}}}{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} P(X \in R_i, Y \in S_j)P(k,l|i,j)} \tag{3}$$

By symmetry, the optimal reconstruction for Y is given by

$$y_{(k,l)} = \frac{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} P(k,l|i,j)\sum_{y\in S_j} y\frac{1}{M_{ij}}}{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} P(X \in R_i, Y \in S_j)P(k,l|i,j)} \tag{4}$$

Similarly, we write out the nearest neighbour conditions; the idea is the same as in COVQ. The optimal encoding partitions for X given fixed partitions in Y, and fixed decoding points $\{(x_{(k,l)}, y_{(k,l)})\}$ are given by:

$$R_i = \{x|d(x,i) \le d(x,h), \forall h \in \{1,...,N_X\}\} \tag{5}$$

and symmetrically, the optimal encoding partitions for Y given fixed partitions in X, and fixed decoding points $\{(x_{(k,l)}, y_{(k,l)})\}$ are given by::

$$S_j = \{y|d(y,j) \le d(y,h), \forall h \in \{1,...,N_Y\}\} \tag{6}$$

where $d(x,i)$ is given by

$$d(x,i) = E[Y^2|X=x] + \sum_{j=1}^{N_Y}\sum_{k=1}^{N_X}\sum_{l=1}^{N_Y}((x-x_{(k,l)})^2 - 2y_{(k,l)}E[Y|X=x, Y\in S_j] + y_{(k,l)}^2)P(Y\in S_j|X=x)P(k,l|i,j) \tag{7}$$

and $d(y,j)$ is given by

$$d(y,j) = E[X^2|Y=y] + \sum_{i=1}^{N_X}\sum_{k=1}^{N_X}\sum_{l=1}^{N_Y}((y-y_{(k,l)})^2 - 2x_{(k,l)}E[X|Y=y, X\in R_i] + x_{(k,l)}^2)P(X\in R_i|Y=y)P(k,l|i,j) \tag{8}$$

Here, we note how the nearest neighbour condition for X depends on the Y encoding regions $\{S_j\}$, and vice versa. We therefore adapt the Lloyd iteration into three stages. This poses a design issue with respect to (a) order of iteration and (b) initialization of codebook. The former we will address at the end of this section, the latter in the next section.

Focusing on $d(x,i)$, since $d(y,j)$ is analogous, we write out the expectations:

$$d(x,i) = \int_{-\infty}^{\infty} y^2 f_{Y|X}(y|X=x)dy + \sum_{j=1}^{N_Y}\sum_{k=1}^{N_X}\sum_{l=1}^{N_Y}((x-x_{(k,l)})^2 - 2y_{(k,l)}\int_{-\infty}^{\infty} y^2\frac{f_{Y|X}(y|X=x)}{P(Y\in S_j)}dy + y_{(k,l)}^2)\int_{S_j} y f_{Y|X}(y|X=x)dy\cdot P(k,l|i,j) \tag{9}$$

The part of this expression that poses a new problem is $f_{Y|X}(y|X=x)$. In order to approximate this with a finite training set, we can first quantize the training values with two one-dimensional uniform quantizers. This is formalized below.

Define $q_X(x):\mathbb{R}\to\{x_1,\ldots,x_{L_X}\}$ and $q_Y(y):\mathbb{R}\to\{y_1,\ldots,y_{L_Y}\}$ to be the L_X- and L_Y-level uniform quantizers for the sources X and Y respectively, where the $x_i - x_{i-1}$ is constant for all $i\in\{2,\ldots,L_X\}$, and $y_i - y_{i-1}$ constant for all $i\in\{2,\ldots,L_Y\}$

$$q_X(x) = x_i \in \{x_1,\ldots,x_{L_X}\} \iff x\in\left[\frac{x_{i-1}+x_i}{2}, \frac{x_i+x_{i+1}}{2}\right) \tag{10}$$

$$q_Y(y) = y_i \in \{y_1,\ldots,y_{L_Y}\} \iff y\in\left[\frac{y_{i-1}+y_i}{2}, \frac{y_i+y_{i+1}}{2}\right) \tag{11}$$

with the convention that $x_0 = y_0 = -\infty$, and $x_{L_X+1} = y_{L_Y+1} = +\infty$.

For simplicity of design, we attach this uniform quantizer to the beginning of our system. That is, we essentially transform the sources to have state spaces of size L_X, L_Y before performing the I→J quantization. (Modified block diagram)

We can now explicity rewrite our approximated conditions in computable terms via a uniformly quantized training set. First, we introduce some terms to simplify notation. Let $m(x_g, y_h)$ be the number of training pairs that quantize to the pair (x_g, y_h):

$$m(x_g, y_h) = |\{(x, y) \in \mathcal{T} : q_X(x) = x_g, q_Y(y) = y_h\}| \tag{12}$$
$$\forall g \in \{1, \ldots, L_X\}, \forall h \in \{1, \ldots, L_Y\}$$

Let $I_X(x_g) \in \{1, \ldots, N_X\}$, $I_Y(y_h)$ be the index of 'closest' X codevector, $I_Y(y_h) \in \{1, \ldots, N_Y\}$ be the index of 'closest' Y codevector:

$$I_X(x_g) = \underset{i \in \{1,\ldots,N_X\}}{\arg\min}\ d_X(x_g, i) \tag{13}$$
$$I_Y(y_h) = \underset{j \in \{1,\ldots,N_Y\}}{\arg\min}\ d_Y(y_h, j) \tag{14}$$

Notice that these are simply the nearest neighbour conditions for the uniformly quantized training set.

Let $M(x_g), M(y_h)$ be the number of training pairs that uniformly quantize to pairs of the form $(x_g, *)$, $(*, y_h)$ respectively:

$$M(x_g) = \sum_{h=1}^{L_Y} m(x_g, y_h) \tag{15}$$
$$M(y_h) = \sum_{g=1}^{L_X} m(x_g, y_h) \tag{16}$$

let $M_j(x_g)$ be the number of training pairs that uniformly quantize to pairs of the form (x_g, y_h) with y_h in the encoding region S_j, and $M_i(y_h)$ be the number of training pairs that uniformly quantize to pairs of the form (x_g, y_h) with x_g in the encoding region R_i:

$$M_j(x_g) = \sum_{y_h : I_Y(y_h) = j} m(x_g, y_h) \tag{17}$$
$$M_i(y_h) = \sum_{x_g : I_X(x_g) = i} m(x_g, y_h) \tag{18}$$

let $S_j(x_g), S_i(y_h)$ be given by:

$$S_j(x_g) = \sum_{y_h : I_Y(y_h) = j} y_h \cdot m(x_g, y_h) \tag{19}$$
$$S_i(y_h) = \sum_{x_g : I_X(x_g) = i} x_g \cdot m(x_g, y_h) \tag{20}$$
$$\tag{21}$$

and $T(x_g), T(y_h)$ be given by:

$$T(x_g) = \sum_{y_h=1}^{L_Y} y_h^2 \cdot m(x_g, y_h) \tag{22}$$
$$T(y_h) = \sum_{x_g=1}^{L_X} x_g^2 \cdot m(x_g, y_h) \tag{23}$$

We now rewrite d_X, d_Y in the above terms. For shorthand, let $q_X(x) = x_g \in \{x_1, \ldots, x_{L_X}\}$ and $q_Y(y) = y_h \in \{y_1, \ldots, y_{L_Y}\}$

$$d_X(x,i) = \frac{1}{M(x_g)}\left(T(x_g) + \sum_{j=1}^{N_Y}\sum_{k=1}^{N_X}\sum_{l=1}^{N_Y}\left(\left((x - x_{(k,l)})^2 + y_{(k,l)}^2\right)M_j(x_g) - 2y_{(k,l)}S_j(x_g)\right)P(k,l|i,j)\right) \quad (24)$$

$$d_Y(y,j) = \frac{1}{M(y_h)}\left(T(y_h) + \sum_{i=1}^{N_X}\sum_{k=1}^{N_X}\sum_{l=1}^{N_Y}\left(\left((y - y_{(k,l)})^2 + x_{(k,l)}^2\right)M_i(y_h) - 2x_{(k,l)}S_j(y_h)\right)P(k,l|i,j)\right) \quad (25)$$

(24) and (25) are now easily computable for given pair (x,i) or (y,j), so the two nearest neighbour optimization steps reduce to performing a search over codebook indices i, j respectively, for the solution to equations (13), (14) respectively.

We also note that, besides aiding notation, the terms $M_j(x_g), S_j(x_g), T(x_g)$, and their counterparts $M_i(y_h), S_i(y_h), T(y_h)$ can be computed initially, independent of x_g, y_h respectively, reducing the computational complexity of the nearest neighbour lookup.

We now write out the centroid condition in a similar manner. We reiterate that the encoding partitions $\{R_i\}_{i=1}^{N_X}, \{S_j\}_{j=1}^{N_Y}$ are held constant in the application of this condition.

Let $M_{(i,j)}$ be the number of training vectors that belong to $R_i \times S_j$:

$$M_{(i,j)} = \sum_{\substack{x_g : I_X(x_g) = i \\ y_h : I_Y(y_h) = j}} m(x_g, y_h) \quad (26)$$

and let $S^X_{(i,j)}$, $S^Y_{(i,j)}$ be given by:

$$S^X_{(i,j)} = \sum_{\substack{x_g : I_X(x_g) = i \\ y_h : I_Y(y_h) = j}} x_g \cdot m(x_g, y_h) \quad (27)$$

$$S^Y_{(i,j)} = \sum_{\substack{x' : I_X(x') = i \\ y' : I_Y(y') = j}} y_h \cdot m(x_g, y_h) \quad (28)$$

With the above notation, the reconstructions that minimize end-to-end distortion are:

$$x_{(k,l)} = \frac{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} S^X_{(i,j)} P(k,l|i,j)}{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} M_{(i,j)} P(k,l|i,j)} \quad (29)$$

$$y_{(k,l)} = \frac{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} S^Y_{(i,j)} P(k,l|i,j)}{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} M_{(i,j)} P(k,l|i,j)} \quad (30)$$

Equations (29), (30) are now easily computable when our encoders are fixed.

Adding the uniform quantizer to the system presents new parameters. These are namely the choice of $\{x_1, y_1, x_{L_X}, y_{L_Y}, L_X, L_Y\}$. Together, these parameters control the granularity of quantization, and position

of the quantizer. For our simulations, we chose to (a) center the quantizer about the sample mean of $\mathcal{T}$, (b) pick $\{x_1, y_1, x_{L_X}, y_{L_Y}\}$ such that all of $\mathcal{T}$ lies within the square defined by the four terms. With these decisions, the added complexity is simply how fine the quantizer ought to be.

For a given training set $\mathcal{T}$, we want fine enough quantization so to not distort the equations (24) and (25), but coarse enough so that the conditional probabilities are well-defined for most bins. It is clearly the case that with a larger training set $\mathcal{T}$, we can use finer quantizations while maintaining well-defined bins. For our purposes, we iterate through values of L_X, L_Y, choosing the result with lowest average distortion.

It is also important to note that the computational complexity depends primarily on the numbers L_X, L_Y, not the size of $\mathcal{T}$, as was the case in the I→I system.

As was mentioned in this section, the fact that there are now three necessary conditions for optimality necessitates a decision in the order of their application in the quantizer training stage. We chose to give equal opportunities to each condition, beginning with one of the encoders. There are clearly other possible rotation schemes that might provide an advantage. These are not explored in this paper.

To summarize this section, we present the procedure which we dub the 'Channel Optimized Stage' of our algorithm. Assuming we are given a codebook of size $N_X \cdot N_Y$, and the encoding mappings I_X, I_Y, we apply our modified Lloyd iteration to obtain a better set of encoding mappings and codebook.

1. Store the previous distortion value, D_{avg}.
2. Update the X encoder; for each $x_g \in \{1, \ldots, L_X\}$, let
$$I_X(x_g) = \underset{i \in \{1,\ldots,N_X\}}{\arg\min} \; d_X(x_g, i) \tag{31}$$
3. Update the Y encoder; for each $y_h \in \{1, \ldots, L_Y\}$, let
$$I_Y(y_h) = \underset{i \in \{1,\ldots,N_Y\}}{\arg\min} \; d_Y(y_h, j) \tag{32}$$
4. Update the codebook; for each $k \in \{1, \ldots, N_X\}$, $l \in \{1, \ldots, N_Y\}$ let
$$x_{(k,l)} = \frac{\sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} S^X_{(i,j)} P(k,l|i,j)}{\sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} M_{(i,j)} P(k,l|i,j)} \tag{33}$$
$$y_{(k,l)} = \frac{\sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} S^Y_{(i,j)} P(k,l|i,j)}{\sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} M_{(i,j)} P(k,l|i,j)} \tag{34}$$
5. Repeat until the average distortion changes less than a desired positive threshold, δ; repeat if
$$\frac{(D_{avg} - D^*_{avg})}{D_{avg}} > \delta \tag{35}$$

where D_{avg} is the distortion of the previous iteration, and the new distortion D^*_{avg} is given by

$$D^*_{avg} = \frac{1}{M} \sum_{g=1}^{L_X} \sum_{h=1}^{L_Y} \left(M(x_g) \cdot d_X(x_g, I_X(x_g)) + M(y_h) \cdot d_Y(y_h, I_Y(y_h)) \right) \tag{36}$$

1.2 Codebook Initialization

Since each nearest neighbour condition depends on the encoding regions of the other source and the codebook pairs $(x_{(k,l)}, y_{(k,l)})$ of the decoder, and the joint centroid conditions depend on the encoding regions for both sources, it is necessary to appropriately find either (a) initial encoding regions $\{R_i\}, \{S_j\}$ for the encoders or (b) initial encoding regions for one of the sources and initial reconstruction points $\{(x_{(k,l)}, y_{(k,l)})\}$ for the decoder. We decided to do both, exploring a couple strategies.

To make finding initial encoding regions simpler, we took a heuristic approach to firstly initialize a codebook of the desired size, $\{(x_{(k,l)}, y_{(k,l)})\}$, $k = 1 \dots N_X$, $l = 1 \dots N_Y$, then apply suitable conditions to find initial encoding regions for *both* sources, and finally begin the main iteration.

To find our initial codebook, our inspiration came from the Linde-Buzo-Gray (LBG) Splitting Algorithm. Recall that the LBG algorithm initializes the codebook to a single codevector, usually placed at the 'center of mass' of the source distribution with respect to a distortion measure. We proceed in the same way, initially letting $N_X = N_Y = 1$, with initial codebook pair $(x_{(1,1)}, y_{(1,1)})$ given by:

$$x_{(1,1)} = \frac{1}{M} \sum_{g=1}^{L_X} \sum_{h=1}^{L_Y} x_g m(x_g, y_h) \tag{37}$$

$$y_{(1,1)} = \frac{1}{M} \sum_{g=1}^{L_X} \sum_{h=1}^{L_Y} y_h m(x_g, y_h) \tag{38}$$

which are respectively the expected values of the uniformly quantized training set for X and Y. Trivially, our encoders are given by:

$$I_X(x) = 1 \tag{39}$$

$$I_Y(y) = 1 \tag{40}$$

We then define the average distortion from this choice of codebook as the expected value of the sum of the squared error distortion between the quantized training vectors in $\mathcal{T}$ and the codebook:

$$D^*_{avg} = \frac{1}{M} \sum_{g=1}^{L_X} \sum_{h=1}^{L_Y} ((x_g - x_{(1,1)})^2 + (y_h - y_{(1,1)})^2) \cdot m(x_g, y_h) \tag{41}$$

We then split in the X and apply a modified Lloyd iteration described as follows.

1. Split in X: let $N_X = 2N_X$.
2. Remember previous distortion; let $D_{avg} = D^*_{avg}$
3. Update the X encoder; for each $g \in \{1, \dots, L_X\}$ let

$$I_X(x_g) = \underset{i \in \{1,\dots,N_X\}}{\arg\min} \; d_X(x_g, i) \tag{42}$$

where $d_X(x_g, i)$ is given as in (24), however with zero probability of channel error:

$$d_X(x_g, i) = \frac{1}{M(x_g)} \left(T(x_g) + \sum_{j=1}^{N_Y} \left(\left((x_g - x_{(i,j)})^2 + y^2_{(i,j)} \right) M_j(x_g) - 2 y_{(i,j)} S_j(x_g) \right) \right) \tag{43}$$

4. Update the X codebook values; for each $k \in \{1, \ldots, N_X\}$, $l \in \{1, \ldots, N_Y\}$, let

$$x_{(i,j)} = \frac{\sum_{\substack{x_g : I_X(x_g)=i \\ y_h : I_Y(y_h)=j}} x_g \cdot m(x_g, y_h)}{\sum_{\substack{x_g : I_X(x_g)=i \\ y_h : I_Y(y_h)=j}} m(x_g, y_h)} \tag{44}$$

   Notice that this is the expected value of the partition (i, j).

5. Update the Y encoder; for each $h \in \{1, \ldots, L_Y\}$, let

$$I_Y(y_h) = \underset{j \in \{1,\ldots,N_Y\}}{\arg\min} \; d_Y(y_h, j) \tag{45}$$

   where $d_Y(y_h, j)$ is defined in the same way as $d_X(x_g, i)$ in (43), i.e. we let the channel error probability of (25) be zero.

6. Update the Y codebook values; for each $k \in \{1, \ldots, N_X\}$, $l \in \{1, \ldots, N_Y\}$, let

$$y_{(i,j)} = \frac{\sum_{\substack{x_g : I_X(x_g)=i \\ y_h : I_Y(y_h)=j}} y_h \cdot m(x_g, y_h)}{\sum_{\substack{x_g : I_X(x_g)=i \\ y_h : I_Y(y_h)=j}} m(x_g, y_h)} \tag{46}$$

7. Define the average distortion. let

$$D^*_{avg} = \frac{1}{M} \sum_{h=1}^{L_Y} d_Y(y_h, I_Y(y_h)) \left(\sum_{g=1}^{L_X} m(x_g, y_h) \right) \tag{47}$$

8. If $\frac{(D_{avg} - D^*_{avg})}{D_{avg}} > \delta$ then return to step 2,
   otherwise if N_X is less than desired, go to next split (step 1),
   otherwise finish.

After convergence of the iteration for our final value of N_X, we then apply the symmetric splitting algorithm in Y until we reach our desired codebook size N_Y in the Y. This yields an initial codebook optimized with respect to the dependence between X and Y. It is worth noting that the properties of the channel do not enter this stage at all; this was a conscious choice that allows for the channel to not be defined for codebooks of size less than $N_X \cdot N_Y$.

The above routine we dub the 'Initialization Stage', we follow this with the 'Channel Optimization Stage' which was described in the previous section.

1.3 Codeword Assignment

Thus far, our algorithm optimizes the mappings I_X, I_Y from source pairs onto codebook pairs, and the codebook $\{(x_{(k,l)}, y_{(k,l)})\}$ subject to the minimization of our distortion functions d_X, d_Y. Since this optimization does not ensure a globally optimal solution, it is possible for the iteration to converge to a sub-optimal configuration. Hence, it "doesn't hurt" to find other ways to optimize the system. We introduce the mappings b_X, b_Y as the .

Define the channel index sets of X and Y respectively to be $\{1, \ldots, N_X\}, \{1, \ldots, N_Y\}$; for convenience, these are numbered in the same way as the encoding index sets. We require that there be an invertible mapping (i.e. a permutation) between encoding indices for each source and their respective channel index set. We call these mappings $b_X(i), b_Y(j)$ respectively. We also denote their inverses $b_X^{-1}(i'), b_Y^{-1}(j')$, where

we use the convention that "primed" indices are channel indices.

The optimization problem alluded to above is that of minimizing the average end-to-end distortion by changing the mappings b_X, b_Y for a fixed encoders, I_X, I_Y, and fixed codebook $\{(x_{(k,l)}, y_{(k,l)})\}$, $k = 1, \ldots, N_X$, $l = 1, \ldots, N_Y$.

Following the construction of **??** for the point-to-point case, we adapt the proof to our system. Expressing the objective function in terms of our index mappings and our quantizers gives:

$$D(I_X, I_Y, b_X, b_Y) = E\left[(X - x_{(K,L)})^2 + (Y - y_{(K,L)})^2; b_X, b_Y\right] \tag{48}$$

$$= \sum_{i,j} P(I_X(X) = i, I_Y(Y) = j) E\left[(X - x_{(K,L)})^2 + (Y - y_{(K,L)})^2 | I_X(X) = i, I_Y(Y) = j; b_X, b_Y\right] \tag{49}$$

$$= \sum_{i,j} p(i,j) \sum_{k,l} p(b(k), b(l) | b(i), b(j)) E\left[((X - x_{(k,l)})^2 + (Y - y_{(k,l)})^2)) | I_X(X) = i, I_Y(Y) = j; b_X, b_Y\right] \tag{50}$$

$$= \sum_{i,j} p(i,j) \sum_{k,l} p(b_X^{-1}(k), b_Y^{-1}(l) | b_X(i), b_Y(j)) \iint_{\substack{x: I_X(x) = i \\ y: I_Y(y) = j}} (x - x_{(k,l)})^2 + (y - y_{(k,l)})^2 dx dy \tag{51}$$

$$= blah \tag{52}$$

$$D_C(b_X, b_Y) = \sum_{i,j} \sum_{k,l} P(b_X^{-1}(k), b_Y^{-1}(l) | b_X(i), b_Y(j)) \cdot \left((x_{(i,j)} - x_{(k,l)})^2 + (y_{(i,j)} - y_{(k,l)})^2\right) \tag{53}$$

The minimization of this function is an NP hard problem. We use the probabilistic method of Simulated Annealing to find a good approximation to the globally optimal solution.

Simulated Annealing works by iteratively choosing 'neighbour states' at random from the 'state space', and probabilistically deciding whether or not to transition to that state based on how long we've been searching and whether or not the new state is 'lower energy' than the current state.

In our case, the functions b_X, b_Y define the state space, and we define a 'neighbour state' of a given (b_X, b_Y) to be a pair (b'_X, b'_Y) such that $b_X(i) \neq b'_X(i)$ for at most two values of i and $b_Y(j) \neq b'_Y(j)$ for at most two values of j. The 'energy' function that we are trying to minimize is (53).

We now describe the algorithm in terms of our functions. We require a few parameters that together control the amount of time spent searching the state space. Define:

$$\{T_i = 10, T_f = 0.00025, R = 0.8, \phi_{max} = 5, \psi_{max} = 200\} \tag{54}$$

In words, these are respectively: the initial 'temperature', the final 'temperature', the 'cooling rate', the number of energy drops until we lower the temperature, the number rejected new states until we lower the temperature. These values are taken from **??**. We stop our search when our timing variable the 'temperature' equals T_f.

1. Initialize b_X, b_Y to the identity maps:

$$b_X(i) = i, \quad \forall i \in \{1, \ldots, N_X\} \tag{55}$$

$$b_Y(j) = j, \quad \forall j \in \{1, \ldots, N_Y\} \tag{56}$$

2. Initialize temperature T; let $T = T_i$.

3. Initialize counters; let $\phi = 0$, $\psi = 0$

4. Initialize 'best' channel index maps; let

$$b_{X_{best}} = b_X \tag{57}$$
$$b_{Y_{best}} = b_Y \tag{58}$$

5. At random, choose $i_1, i_2 \in \{1, \ldots, N_X\}$, choose $j_1, j_2 \in \{1, \ldots, N_Y\}$. Define b'_X, b'_Y as follows:

$$b'_X(i) = \begin{cases} b_X(i_1) & \text{if } i = i_2 \\ b_X(i_2) & \text{if } i = i_1 \\ b_X(i) & \text{otherwise} \end{cases} \tag{59}$$

$$b'_Y(i) = \begin{cases} b_Y(j_1) & \text{if } j = j_2 \\ b_Y(j_2) & \text{if } j = j_1 \\ b_Y(j) & \text{otherwise} \end{cases} \tag{60}$$

   (a) If $D_C(b'_X, b'_Y) < D_C(b_X, b_Y)$, accept the change; let

$$b_X = b'_X \tag{61}$$
$$b_Y = b'_Y \tag{62}$$

   (b) If $D_C(b'_X, b'_Y) \geq D_C(b_X, b_Y)$, accept the change with probability $\exp(\frac{-(D_C(b'_X, b'_Y) - D_C(b_X, b_Y))}{T})$.

   (c) Otherwise, keep the current state.

6. If $D_C(b_X, b_Y) < D_C(b_{X_{best}}, b_{Y_{best}})$, store this 'best' state; let

$$b_{X_{best}} = b_X \tag{63}$$
$$b_{Y_{best}} = b_Y \tag{64}$$

7. (a) If we accepted the change, increment ϕ; let $\phi = \phi + 1$

   (b) Otherwise, increment ψ; let $\psi = \psi + 1$

8. (a) If $\psi = \psi_{max}$, reset ψ, lower temperature; let

$$\psi = 0 \tag{65}$$
$$T = R \cdot T \tag{66}$$

   (b) Otherwise, if $\phi = \phi_{max}$, reset ϕ, lower temperature; let

$$\phi = 0 \tag{67}$$
$$T = R \cdot T \tag{68}$$

   (c) Otherwise, leave temperature unchanged.

9. (a) If $T > T_f$, go back to step 5

   (b) Otherwise, restore best state; let

$$b_X = b_{X_{best}} \tag{69}$$
$$b_Y = b_{Y_{best}} \tag{70}$$

(a) left

(b) right

Figure 1: An example of a stereo image.

1.4 Image Coding

Since our system is suited for two correlated sources, we decided to apply it to the compression of two images that ought to (intuitively) share spatial correlation. The class of images we focused on was the class of stereoscopic images - that is, pairs of images of the same setting taken from two different angles. We explore the effect of transform coding on pairs of images from this class as input to our system.

Transform coding is often used in practice for efficient image coding **??**. The concept behind transform coding is to perform an *orthogonal* linear transform on large blocks of the source scalars prior to scalar quantization, then inverse transformed on the decoding side. This is illustrated in (Figure??). The reason this is done is to pairwise *de-correlate* the components of the input vector such that the components of the output vector of the transform are (close to) pairwise uncorrelated **??**.

Of the known computable orthogonal linear transforms, the Discrete Cosine Transform (DCT or DCT-II) has been found to be the most effective **??**, and it is widely used in image and video compression. A 2-D DCT is often used on blocks of 8×8 pixels of the source image before lossy quantization, e.g. JPEG, JPEG-2000. We give the 2-D DCT for 8×8 pixel blocks $\{x_{i,j}\}, i = 0 \dots 7, j = 0 \dots 7$ below:

$$\left\{ X_{i',j'} = \sum_{i=0}^{7} \sum_{j=0}^{7} x_{i,j} \cos\left[\frac{\pi}{8}\left(j + \frac{1}{2}\right) j'\right] \cos\left[\frac{\pi}{8}\left(i + \frac{1}{2}\right) i'\right] \right\} \quad i' = 0 \dots 7, j' = 0 \dots 7 \tag{71}$$

The database of freely licenced stereo images we used was found at **??**. An example of the type of image we used is shown in figure 1.

An important question in transform coding is that of bit allocation. We wish to know how many bits, i.e. how many codevectors, to assign to each DCT coefficient to optimally recover the source. In the single source case, this problem was studied in **??**. A simple greedy algorithm optimally allocates bits to the DCT coefficients. The algorithm assumes that the $(0, 0)$ coefficient (the DC coefficient) is Gaussian, while the other 63 coefficients are Laplacian **??**. An example of how this algorithm allocates bits to the images of figure 1 is given in figure 2.

The DCT computations and image manipulation was coded in Python using Pillow Imaging library and numpy.

$$\begin{bmatrix} 7 & 7 & 5 & 4 & 0 & 0 & 0 & 0 \\ 7 & 5 & 4 & 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a) left

$$\begin{bmatrix} 7 & 6 & 5 & 3 & 0 & 0 & 0 & 0 \\ 7 & 4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 6 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) right

Figure 2: Greedy independent 64 bit allocation for the stereo images of figure 1

$$\begin{bmatrix} 0.443 & -0.01 & -0.093 & -0.011 & -0.061 & 0.003 & -0.017 & 0.062 \\ 0.416 & 0.006 & -0.011 & 0.014 & -0.031 & -0.039 & 0.003 & -0.045 \\ 0.498 & -0.061 & -0.01 & 0.074 & -0.034 & -0.017 & 0.008 & 0.054 \\ 0.438 & 0.027 & -0.008 & 0.074 & -0.016 & 0.022 & 0.006 & -0.004 \\ 0.449 & 0.075 & -0.019 & 0.017 & 0.014 & 0.063 & -0.029 & 0.007 \\ 0.632 & 0.028 & -0.034 & -0.018 & 0.033 & -0.037 & 0.032 & 0.061 \\ 0.765 & 0.177 & -0.032 & -0.063 & -0.033 & 0.076 & -0.001 & 0.089 \\ 0.679 & 0.148 & 0.028 & 0.04 & 0.046 & -0.013 & 0.046 & 0.034 \end{bmatrix}$$

Figure 3: DCT coefficient correlation matrix of the stereo images in figure 1

$$\begin{bmatrix} 0.654 & 0.134 & 0.054 & 0.087 & 0.077 & 0.124 & 0.119 & 0.144 \\ 0.371 & 0.072 & 0.043 & 0.059 & 0.05 & 0.063 & 0.053 & 0.048 \\ 0.273 & 0.067 & 0.065 & 0.06 & 0.055 & 0.05 & 0.062 & 0.059 \\ 0.23 & 0.056 & 0.064 & 0.053 & 0.046 & 0.045 & 0.047 & 0.039 \\ 0.242 & 0.053 & 0.05 & 0.056 & 0.047 & 0.062 & 0.058 & 0.036 \\ 0.213 & 0.064 & 0.064 & 0.05 & 0.061 & 0.052 & 0.045 & 0.059 \\ 0.238 & 0.068 & 0.07 & 0.056 & 0.048 & 0.046 & 0.042 & 0.054 \\ 0.233 & 0.081 & 0.074 & 0.066 & 0.046 & 0.05 & 0.039 & 0.049 \end{bmatrix}$$

Figure 4: Average DCT coefficient correlation matrix of the dataset of 39 pairs of stereo images

The bit allocation problem for the two source case was not studied. We have thus omitted bit allocation from our simulations.

1.5 Design Summary

To recap, and to provide an overall picture, we will now give an overview of the system design. The following describes our process for images.

1. Convert pixel data of stereo images to 64 scalar pair sets of 2-D DCT coefficients;
2. Find good bit allocation using the optimal bit allocation greedy algorithm for independent sources;
3. Perform 'Initialization Stage' quantization with splitting;
4. Perform 'Channel-Optimization Stage' quantization until convergence;
5. Perform simulated annealing to find approximately optimal channel codeword mappings b_X, b_Y;
6. Return to step 4 until satisfactory distortion is achieved.