The inconvenient properties of the YUV transform is causing me more problems than anticipated.

# 1 Summary

1. Summary
2. The Inverse Problem
   (a) I don't know how to solve the inverse problem if I use linear approximation for quantization.
   (b) Inverse problem without linear approximation for quantization is solvable with $(256 \times 256)$ Cholesky factorization. However, the indexing is awkward, so it would take some time to code.
3. Accounting for Quantization in the Dual Update
   (a) I can handle quantization in the dual update, but...
   (b) This will lead to suboptimal solutions if quantization is not also handled in the primal updates.
4. Accounting for Quantization in the Primal Update
   (a) I don't have good solution here.
   (b) Approximate solutions also have problems.

Everywhere I look there's another problem to solve. I'm starting to lean towards doing my convolutions in YUV domain to dodge all of these issues. That way, the conversion to JPEG coefficients is just a projection. I'm not sure exactly how that effects normalization of dictionaries, the model, or the dictionary learning process, but that seems like an easier problem to deal with than these. The math doesn't really care whether I'm working in RGB or YUV.

# 2 The Inverse Problem

Here is the fundamental problem.

Let $\boldsymbol{B}$ be the linear operator consisting of three to four operations: compute the YUV transform, downsample the U and V channels, compute an $8 \times 8$ block norm-preserving DCT. If using a linear approximation of quantization, I would then zero out some of the higher frequencies.

The problem I need to solve is this:

$$\min_{\boldsymbol{v}} \frac{\mu}{2} \|\boldsymbol{v} - \boldsymbol{x}\| + \frac{\rho}{2} \|\boldsymbol{B}\boldsymbol{v} - \boldsymbol{y}\|_2^2 \tag{1}$$

(My notes have more complicated expressions in place of $\boldsymbol{x}$ and $\boldsymbol{y}$, but how those variables are calculated isn't relevant to this specific problem.)

This problem is equivalent to solving for $\boldsymbol{v}$ in this equation:

$$(\mu \mathbf{I} + \rho \boldsymbol{B}^T \boldsymbol{B})\boldsymbol{v} = \boldsymbol{x} + \boldsymbol{B}^T \boldsymbol{y} \tag{2}$$

When thought of as a matrix and without the linear quantization approximation $\boldsymbol{B}$ is of size $(256 \times 768)$. With use of Woodbury matrix identity, I could compute the Cholesky on a $(256 \times 256)$ matrix. However, this would prevent me from using the linear quantization approximation.

There is also an equivalent inverse problem:

$$(\mu \boldsymbol{C} + \rho \boldsymbol{S})\boldsymbol{v} = \boldsymbol{C}(\boldsymbol{x} + \boldsymbol{B}^T \boldsymbol{y}) \tag{3}$$

where $\boldsymbol{C}$ is an operator that only acts across channels, and $\boldsymbol{S}$ is like $\boldsymbol{B}^T \boldsymbol{B}$ only with the YUV transform and its transpose removed. Put another way, $\boldsymbol{C}$ acts across channels, and $\boldsymbol{S}$ acts across space. We can think of this matrix $\mu \boldsymbol{C} + \rho \boldsymbol{S}$ as the sum of a sparsely banded matrix and a block diagonal matrix in which all diagonal blocks are identical. There is lots of structure here, but I'm at a loss on how to exploit it. It would be amazing if the inverse could be a sum/composite of spatial and channel functions, but I am skeptical that the inverse has a form that holds such properties and have no idea how I'd find that form if it did exist.

# 3 Accounting for Quantization in the Dual Update

One solution to the quantization problem is to approximate it with a linear operator that just zeros out wherever the JPEG compression algorithm zeroed out frequencies. Such a solution poses challenges to the inverse problem in the previous section, but makes the dual update very straightforward (simply modify $\boldsymbol{B}$ to include the linear approximation for quantization). However, if the linear approximation is not used, quantization needs to be accounted for in the dual update.

In ADMM, the dual update is a gradient ascent step in the augmented Lagrangian:

$$\mathrm{L}_\rho(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\eta}) = f(\boldsymbol{x}, \boldsymbol{v}) + \boldsymbol{\eta}^T(\boldsymbol{B}\boldsymbol{v} + \boldsymbol{c}) + \frac{\rho}{2}\|\boldsymbol{B}\boldsymbol{v} - \boldsymbol{c}\|_2^2 \tag{4}$$

which produces the update function:

$$\frac{\boldsymbol{\eta}^{(k+1)}}{\rho} = \frac{\boldsymbol{\eta}^{(k)}}{\rho} + \boldsymbol{B}\boldsymbol{v} - \boldsymbol{c} \tag{5}$$

where $\boldsymbol{c}$ is entirely unrelated to the $\boldsymbol{C}$ from the previous section, $\boldsymbol{B}\boldsymbol{v} - \boldsymbol{c} = 0$ is the constraint, and $\rho$ is the step size for dual gradient ascent.

However, in my case, my constraint is $\boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v}) - \boldsymbol{Q}(\boldsymbol{c}) = 0$, where $\boldsymbol{Q}$ is the quantization operator.

The simplest approach to incorporate quantization into the dual update is this:

$$\frac{\boldsymbol{\eta}^{(k+1)}}{\rho} = \frac{\boldsymbol{\eta}^{(k)}}{\rho} + \boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v}) - \boldsymbol{Q}(\boldsymbol{c}) \tag{6}$$

This approach produces a quantized dual variable. There is a nice property here that if the constraint is met, the dual variable stays constant (so if the algorithm finds the minimum, this update equation won't move away from it.)

I see two drawbacks to this approach:

1. The discrete nature could lead to oscillation instead of convergence.
2. If quantization is not incorporated into the primal update, the primal update will be suboptimal. The algorithm will likely converge towards a suboptimal, feasible solution.

The first item can be addressed by removing the quantize nature of the dual updates.

$$\begin{aligned} \Delta_{\boldsymbol{\eta}} &= \arg\min_{\boldsymbol{\epsilon}} \|\boldsymbol{\epsilon}\|_2^2 \\ &\text{subject to } \boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v} - \boldsymbol{\epsilon}) - \boldsymbol{Q}(\boldsymbol{c}) = 0 \end{aligned} \tag{7}$$

$$\frac{\boldsymbol{\eta}^{(k+1)}}{\rho} = \frac{\boldsymbol{\eta}^{(k)}}{\rho} + \Delta_{\boldsymbol{\eta}} \tag{8}$$

Overrelaxation can be added without too much headache as well. (Overrelaxation in ADMM is similar to Nesterov momentum in that it "looks forward" for gradients, but unlike Nesterov momentum, overrelaxation doesn't keep a cummulative velocity term.)

$$\begin{aligned} \Delta_{\boldsymbol{\eta}} &= \arg\min_{\boldsymbol{\epsilon}} \|\boldsymbol{\epsilon}\|_2^2 \\ &\text{subject to } \boldsymbol{Q}\left(\frac{(\alpha-1)\boldsymbol{B}\boldsymbol{v}^{(k)} + \boldsymbol{B}\boldsymbol{v}^{(k+1)}}{\alpha} - \boldsymbol{\epsilon}\right) - \boldsymbol{Q}(\boldsymbol{c}) = 0 \end{aligned} \tag{9}$$

$$\frac{\boldsymbol{\eta}^{(k+1)}}{\rho} = \frac{\boldsymbol{\eta}^{(k)}}{\rho} + \alpha\Delta_{\boldsymbol{\eta}} \tag{10}$$

However, the second drawback cannot be easily addressed with the dual update. The problem needs to be fixed during the primal update.

# 4 Accounting for Quantization in the primal update

The problem that needs to be solved:

$$\boldsymbol{v} = \arg\min_{\boldsymbol{v}} \frac{\mu}{2}\|\boldsymbol{v} - \boldsymbol{x}\|_2^2 + \frac{\rho}{2}\|\boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v}) - \boldsymbol{Q}(\boldsymbol{c}) + \frac{\boldsymbol{\eta}}{\rho}\|_2^2 \tag{11}$$

The approach that I think makes most sense here is to estimate $\boldsymbol{v}$ ignoring quantization, and then try to modify the result in a way that does not change the quantization of $\boldsymbol{v}$.

That is,

$$\boldsymbol{v} = \boldsymbol{v}_{\text{est}} + \boldsymbol{B}^{-1}\boldsymbol{\epsilon} \tag{12}$$

where

$$\boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v}_{\text{est}} + \boldsymbol{\epsilon}) = \boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v}_{\text{est}}) \tag{13}$$

and $\boldsymbol{B}^{-1}$ the a right-side inverse (the one you'd use to reconstruct a signal from JPEG coefficients. That is, inverse DCT, upsample UV channels, and convert from YUV to RGB.)

However, I do not know how to get $\boldsymbol{\epsilon}$ in a tractable way, since $\boldsymbol{B}\boldsymbol{v}$ and $\boldsymbol{v}$ are in diferent domains.

I thought about doing this:

$$\begin{aligned} \boldsymbol{\epsilon} = \arg\min_{\boldsymbol{\epsilon}} &\|\boldsymbol{B}(\boldsymbol{v}_{\text{est}} - \boldsymbol{x}) + \boldsymbol{\epsilon}\|_2^2 \\ \text{subject to } &\boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v}_{\text{est}} + \boldsymbol{\epsilon}) - \boldsymbol{Q}(\boldsymbol{v}_{\text{est}}) = 0 \end{aligned} \tag{14}$$

However, substituting

$$\|\boldsymbol{B}(\boldsymbol{v}_{\text{est}} - \boldsymbol{x}) + \boldsymbol{\epsilon}\|_2^2 \tag{15}$$

for

$$\|\boldsymbol{v}_{\text{est}} - \boldsymbol{x} + \boldsymbol{B}^{-1}\boldsymbol{\epsilon}\|_2^2 \tag{16}$$

is mathematically dubious. Using eigenvalues, you can bound one norm using the other, but a $\boldsymbol{\epsilon}$ that increases one norm could decrease the other. (I find it helpful to think of the unit balls: this approach is using tilted elipsiod unit ball instead of a spherical unit ball, and hoping the two line up well enough that it doesnt matter.)

I had one last idea. Let $\boldsymbol{W}$ be the RGB to YUV transform followed by downsampling the UV channels, and $\boldsymbol{W}^{-1}$ be upsampling the UV channels followed by the YUV to RGB transform (again, only a right-side inverse). I thought about doing this:

$$\begin{aligned} \boldsymbol{\epsilon} = \arg\min_{\boldsymbol{\epsilon}} &\|\boldsymbol{W}^{-1}\boldsymbol{B}(\boldsymbol{v}_{\text{est}} - \boldsymbol{x}) + \boldsymbol{W}^{-1}\boldsymbol{\epsilon}\|_2^2 \\ \text{subject to } &\boldsymbol{Q}(\boldsymbol{B}\boldsymbol{v}_{\text{est}} + \boldsymbol{\epsilon}) - \boldsymbol{Q}(\boldsymbol{v}_{\text{est}}) = 0 \end{aligned} \tag{17}$$

The substitution sill suffers some of the same problems as before, but I think this is the closest elipsoid I can get to the sphere without resorting to an iterative or combinatoric approach. I haven't solved it, but it should have a closed-form solution. It's a quadratic function, and it's constraints are just holding it in a 4-d rectangular prism.