

For the second code :

The encoded sequence is 001011001000. If the first bit is in error, then the sequence is 101011001000. Parsing the sequence into codewords, we have 10 10 11 00 10 00 and the decoded sequence therefore is $a_1 a_1 a_2 a_3 a_1$ and a_2 . Therefore only one symbol is in error.

(d) For the first code, the encoded sequence is 1010001011. If the third bit is in error, then sequence is 1000001011 and this is parsed as 1 000 0010 1 1 which is decoded as a_2, a_3, a_4, a_2, a_2 . Therefore 3 symbols are in error plus one is deleted.

For the second code : The encoded sequence is 001011001000. If the third bit is in error, then the sequence becomes 000011001000 and this decodes into $a_2 a_2 a_3 a_2 a_1 a_2$. One character is received in error, namely the second one, before the decoder gets back on track.

We see that in both cases the second code gets back on track faster.

2. JPEG quality levels

To calculate the MSE, you have to write out the image as a jpeg, and then read it back in and calculate the MSE. The MATLAB code is:

```
% For aerial4.tif

aer=imread('aerial4.tif','tif');
imwrite(aer,'aerial4.jpg','jpg');
aerjpg=imread('aerial4.jpg','jpg');
aerd=double(aer);
aerjpgd=double(aerjpg);
mse(aerd,aerjpgd)

% For xray.tif

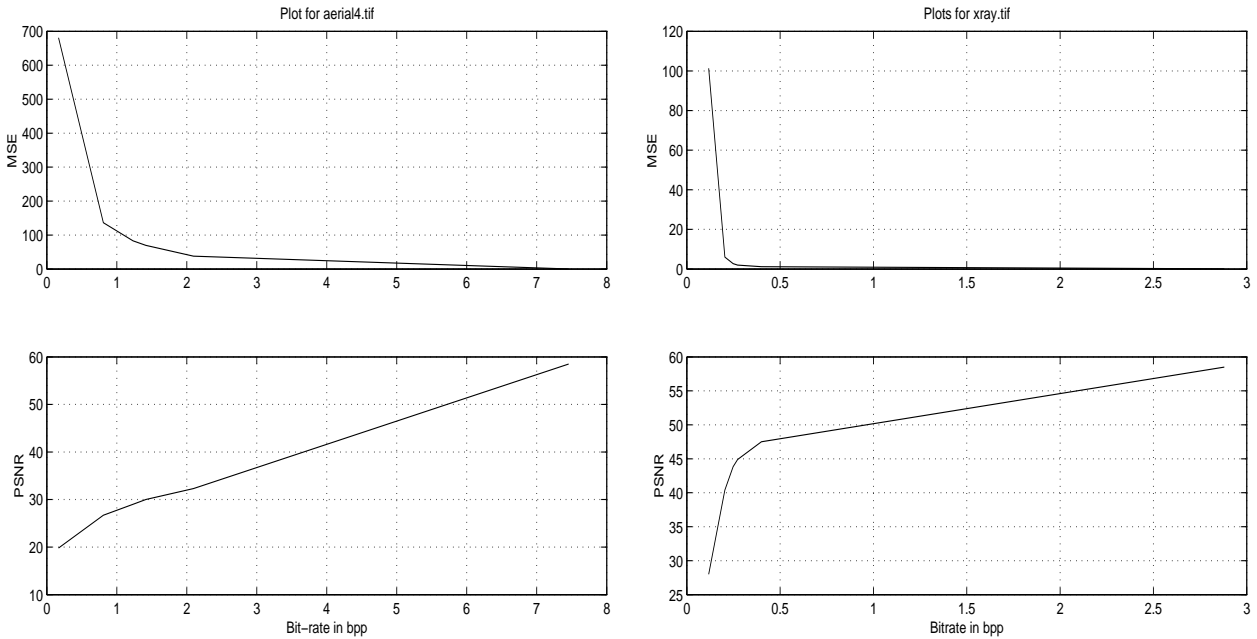
xray=imread('xray.tif','tif');
imwrite(xray,'xray.jpg','jpg');
xrayd=double(xray);
xrayjpg=imread('xray.jpg','jpg');
xrayjpgd=double(xrayjpg);
mse(xrayd,xrayjpgd)
```

We get an MSE of 37.9 for the image 'aerial4.tif' and an MSE of 1.166 for the image 'xray.tif'. To calculate the bit-rate, we use `ls -l` on a Unix system to find out the file size in bytes. Then the formula to be used is

$$\text{Bit - rate} = \frac{\text{Size} \times 8}{\#\text{pixels}}$$

to get the bit rate in bits per pixel (bpp).

Applying this formula yields bit-rates of 2.0973 bpp for 'aerial4.tif' and 0.4016 bpp for 'xray.tif'. Using different quality levels, the plots of MSE versus bitrate and PSNR versus bitrate are below. It is clear that xray is more compressible. Seeing the bpp values of the two images, as well as the visual quality and the MSE results, one can say that 'xray.tif' is much more compressible than 'aerial4.tif'. The xray is a much smoother image. Also, for the xray image, as you put more bits into the JPEG compression, you rapidly hit a point of diminishing return. For the aerial image, you get diminishing returns as well, although it doesn't happen at such a low bit rate.



3. JPEG: Successive Approximation Progressive Mode

a) When we divide by 16 and round off to the nearest integer, we get the following block:

2	1	-1	0	0	0	0	0
2	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0

The (runlength,value) pairs for the AC coefficients are:

(0,1), (0,2), (1,1), (0,-1), (3,1)

and there is also an EOB that will need to be encoded.

When the decoder receives these pairs, it can decode the block shown above, and then multiply by 16 to obtain:

32	16	-16	0	0	0	0	0
32	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

b) There are many possible correct answers here. Since the values will tend to be smaller (as a result of dividing by 16) we could eliminate some of the AC categories altogether. Or, we could modify the probabilities of the low-valued categories, since these will be more probable and should now have shorter codewords. Or we could do both: eliminate categories that can't occur, and also modify probabilities of the remaining categories. The EOB symbol can be made more probable (shorter codeword) since it will occur earlier in the stream.

c) It is possible that sending an additional refinement bit for a coefficient with non-zero history could degrade the representation of that coefficient at the decoder. For example, suppose the coefficient has value 17, and it gets represented as 16. In the next scan, we send a 1 to indicate that it is in the upper half of the uncertainty interval, and it will now get represented as 20. The error has gone up. If we were to continue to more scans, the error will go down again— eventually the step index will be exactly reconstructed (which does not mean the coefficient is lossless, there is still the basic level of quantization error).

4. **Lempel-Ziv coding:**

The decoder decodes as follows:

Read in from Encoder	Decode it as	Add string to dictionary	As Entry
6	t	Can't add yet	
3	h	th	7
4	i	hi	8
5	s	is	9
2	(space)	s(space)	10
3	h	(space)h	11
1	a	ha	12
6	t	at	13
2	(space)	t(space)	14
9	is	(space)i	15
11	(space)h	is(space)	16
16	is(space)	(space)hi	17
12	ha	is(space)h	18
14	t(space)	hat	19
4	i	t(space)i	20
20	t(space)i	it	21
10	s(space)	t(space)is	22
8	hi	s(space)h	23
23	s(space)h	his	24
13	at	s(space)ha	25

The first column shows the sequence of dictionary indexes coming from the encoder. The second column shows what string that dictionary index gets decoded into (table look up). The third column shows what string the decoder will add to its dictionary as a result of that decoding. The string being added is always the concatenation of the string that was decoded on the previous line, plus the first

item in the string currently decoded. The last column show the entry number being assigned to the current string added to the dictionary.

The final decoded string is:
this hat is his hat it is his hat

5. Scalar and Vector Quantization

- a) A globally optimal 2-bit Lloyd-Max scalar quantizer will produce the same rate (2 bits) but lower (or equal) distortion.
- b) A Huffman coded uniform 2-bit scalar quantizer will produce the same distortion D_1 , but a lower (or equal) rate. More specifically, the rate will be between 1 and 2 bits per symbol.
- c) A globally optimal 1-bit Lloyd-Max scalar quantizer will produce a lower rate (1 bit instead of 2). The distortion might be higher or lower than D_1 , depending on the distribution.
- d) Since a 4-bit vector quantizer is used on a pair, the bit rate is still 2 bits per component, the same rate as R_1 . The distortion might be higher or lower than D_1 , depending on the distribution *and* on the random initialization.
- e) As in part (d), the rate will be 2. If we use D_a to denote the distortion achieved in part (a), the distortion achieved here will be less than or equal to D_a , since the VQ starts off with a distortion of D_a per component, and the Generalized Lloyd Algorithm (GLA) can only make things better (or stay the same). So, from part (a) we had that $D_a \leq D_1$, for this part we can also say that the distortion $D_e \leq D_1$. In fact, by initializing the VQ with the (outer) product code from (a), we are initializing at a minimum point, so the GLA will produce no change. It is possible that this is not the global minimum for the vector quantizer. You don't have to include this in your answer. It is sufficient to say that the distortion here is less than or equal to D_1 .

6. Order of Operations:

(a) Clearly, interpolating before JPEG will mean we have to encode 4 times as many pixels. Doing median filtering before JPEG will mean that we have fewer high-magnitude high frequency coefficients to represent. So the system with the fewest bits will be C, where the helpful operation (median filtering) comes before JPEG, and the unhelpful operation (interpolating) comes after. For the same reason, system D will be the worst. B and F are exactly equal, since JPEG comes first. They are second. It remains to compare E and A. We expect that the median filtering will do a better job of removing the spikes if the median filtering is done first. This is because if the interpolation is done first, the noise spikes will spread out, and, unless the median filter also grows larger, it will be less able to remove them. So our final order is:

$$C < B = F < E < A < D$$

(b) As mentioned above, median filtering will do a better job of removing the noise if it is done before upsampling.

Next, consider upsampling and JPEG. The compression step, by and large, introduces two kinds of artifacts: blocking artifacts (noticeable boundaries between the 8x8 blocks), and loss of high frequency information (because of the quantization). If we upsample first, the JPEG encoder will receive a

smoother, more correlated image to compress. There will be less loss in the quantization, and less of a blocking problem (since 8x8 blocks will now be more similar to each other). If we compress first, there will be more loss of high frequencies, which will not get restored by the subsequent upsampling. On the other hand, the interpolation might smooth out the blocking artifacts a bit. On balance, I would say it is probably better to upsample first, but if you give any sensible argument that would be acceptable.

Lastly we consider median filtering and JPEG. Median filtering coming first will do a better job of cleanly removing the spikes.

These statements can be summarized as:

MF before $\uparrow 2$, $\uparrow 2$ before JPEG, MF before JPEG.

From this we can conclude that E would have the best quality, and B would be worst. Further ranking is difficult, because one has to estimate how strong the different effects are. You would get full credit if you got to this point, with no further ranking.

I suspect that median filtering before upsampling is the most significant effect, and I would give an overall ranking as follows:

$$B < D < A < F < C < E$$

but if you wind up with a different ranking on the intermediate systems, you can still get full credit on this part.

7. JPEG:

Considering just the natural images, b should compress to a smaller size than a. Consider that if you keep dividing the pixel values by 2, the dynamic range shrinks and the DCT coefficients will likewise keep getting smaller, eventually all coefficients would fall below the default quantization thresholds.

Considering just the artificial images, d will compress the most, since there is no variation at all.

Most people thought that c and f would be the same, that vertical and horizontal strips would be equivalent. A couple of people thought that they would be different, since the quantization tables are not quite symmetrical. Actually, the quantization tables in this particular situation do not matter, because both the vertical and horizontal strips of width 8 ensure that every block has 63 zero-valued AC coefficients. Only the DC coefficient will be non-zero, in either case. The asymmetry of the quantization tables pertains only to the AC coefficients. However, since the difference coding (DPCM) of the DC coefficients proceeds in raster scan order, the horizontal strips will compress better than the vertical strips, simply because the DC coefficient will be nicely predictable along the horizontal strips. In fact, image c might compress pretty much the same as d (the constant-valued image).

Clearly image e will not be as compressible as c,d,f.

Lastly, one needs to compare the artificial images with the natural images. Clearly c,d,f have less going on than a,b, and will be more compressible. But it is a little hard to say how an artificial image like e would compare to a and b. (If the random unit-width strips use the full dynamic range, likely they would be less compressible than airplane.) Full credit was given either way.