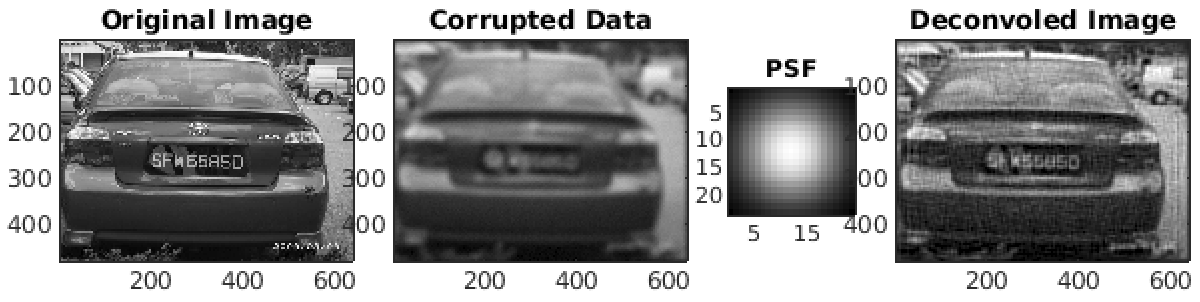
Justin Andre E. Po Oscar Vian L. Valles

CMSC 178 - A

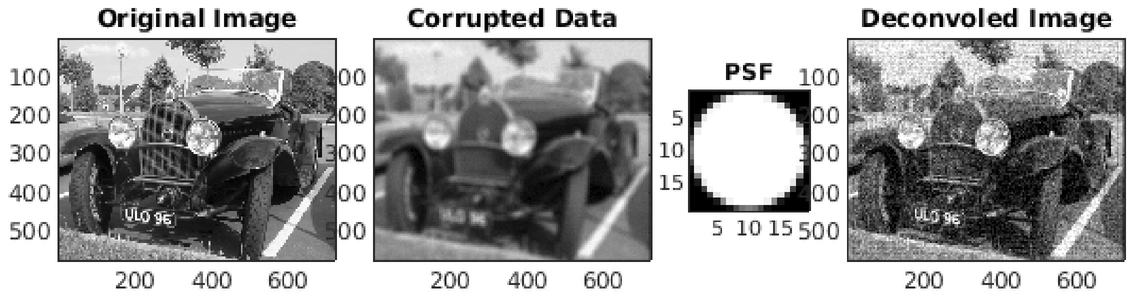
Assignment 2

2A - De-convolution using a Wiener Filter

Results (deconv_test.m)



example.png



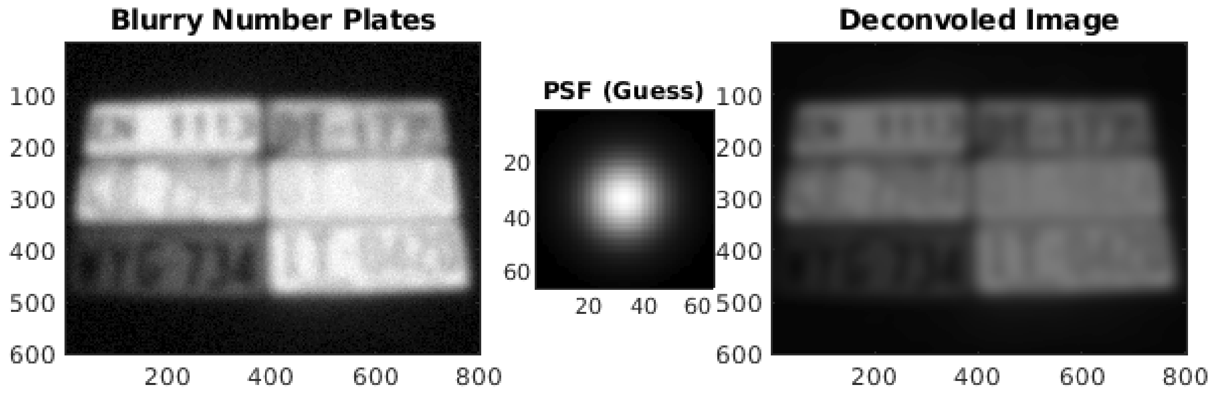
example2.png

Comments

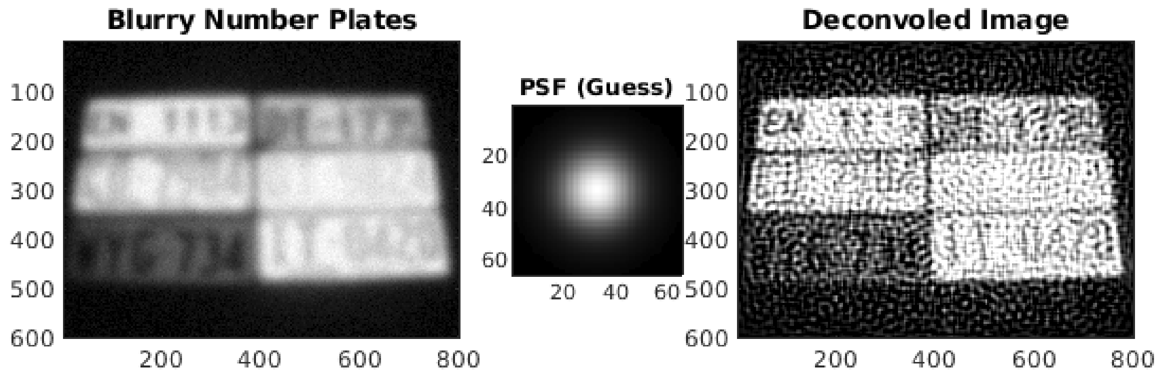
The `deconv_test` with our modified `wiener_deblur` function was able to successfully deconvolve the corrupted sample images. It took a total of 0.107s to run the function, with the call to the `edgetaper` function taking the most time (0.072s).

Through the use of the inverse filter, both `example.png` and `example2.png` were deconvolved to a point where the license plates are somewhat legible. Although they aren't nearly as clear as their original images, we were still able to recover significant amounts of data.

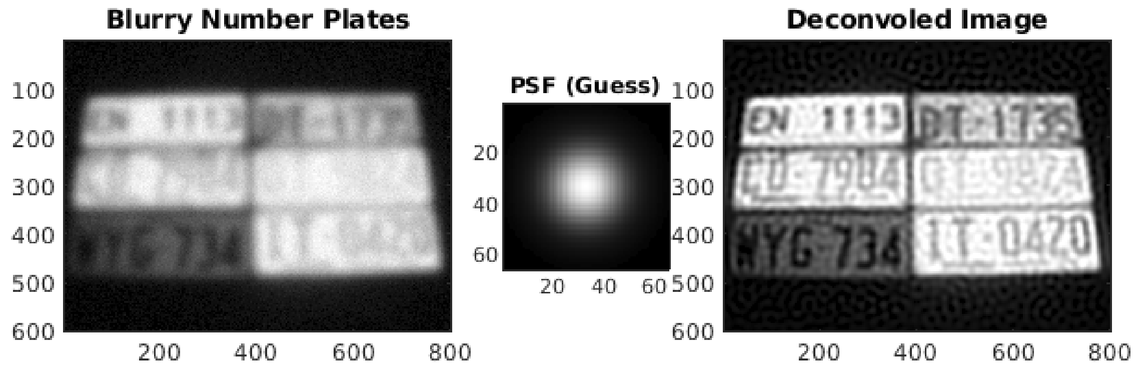
Results (deconv_test2.m)



Output of deconv_test2.m with k = 1



Output of deconv_test2.m with k = 0.00001



Output of deconv_test2.m with k = 0.0004

Comments

The `deconv_test2` function with our modified `wiener_deblur` function was able to successfully deconvolve the image with blurry number plates. It took a total of 0.077s to run the function, with the `edgetaper` function call still taking the most time (0.053s).

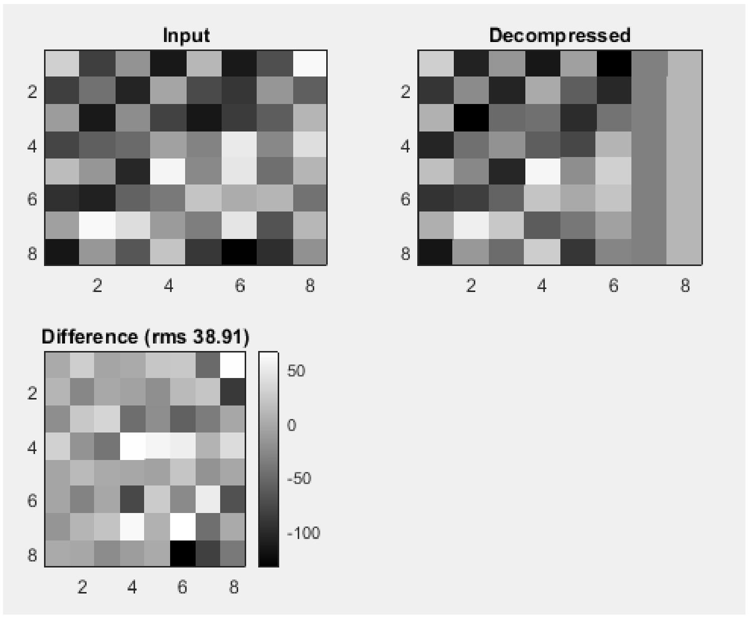
By using the maximum k value (k = 1) for `deconv_test2.m`, we noticed that it would result in a less noisy but very dark image. On the other hand, using the minimum k value (k = 0.00001) would result in a sharper image, but at the cost of more noise being introduced. The usage of both k values resulted in images whose plate numbers were nowhere near legible.

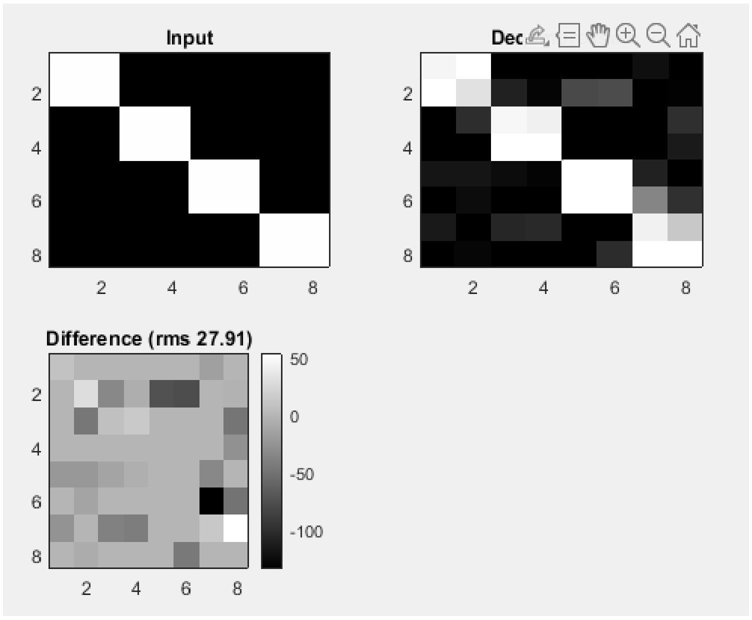
Through trial and error, we found that the optimal k value for this function is 0.0004. It manages to produce a sharp image with legible plate numbers, while containing very little noise.

2B - Motion Image Compression

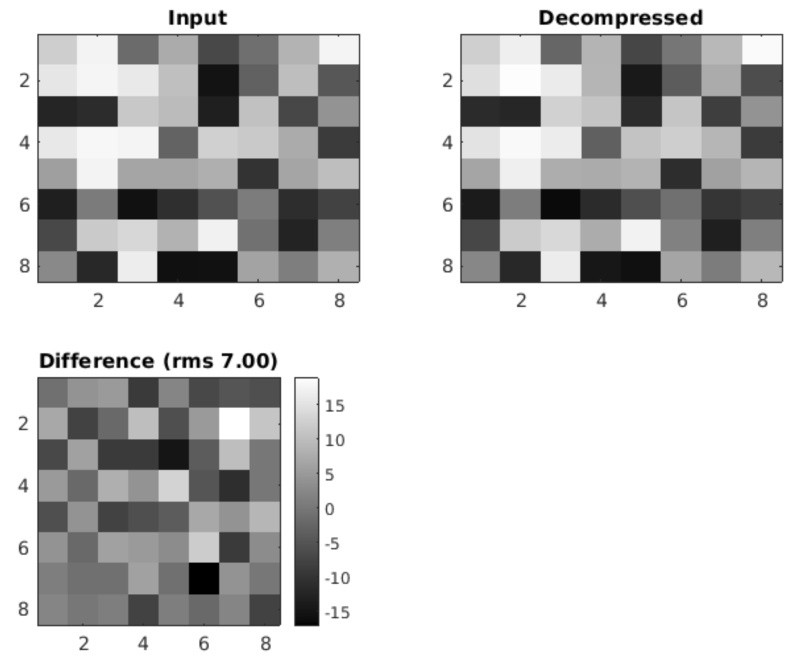
JPEG 8x8

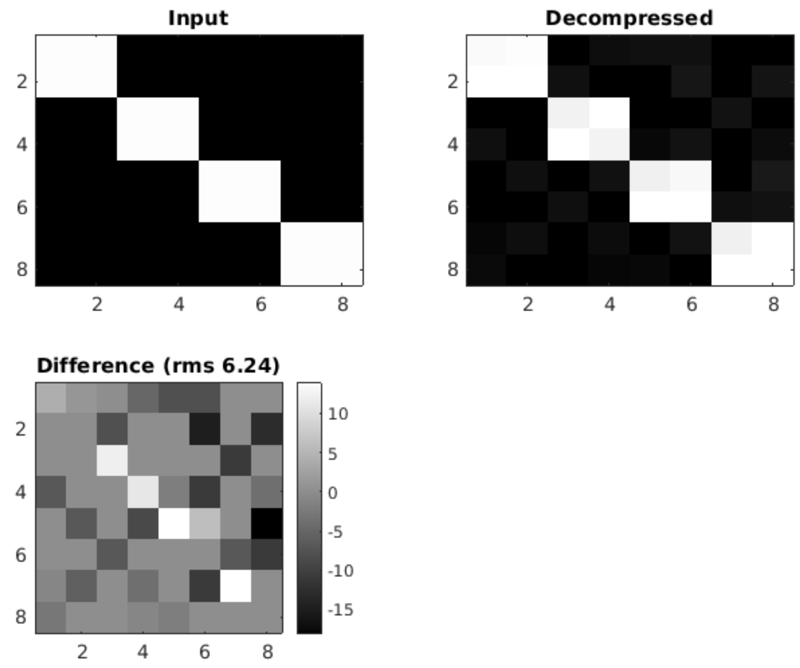
Results (Q=20)



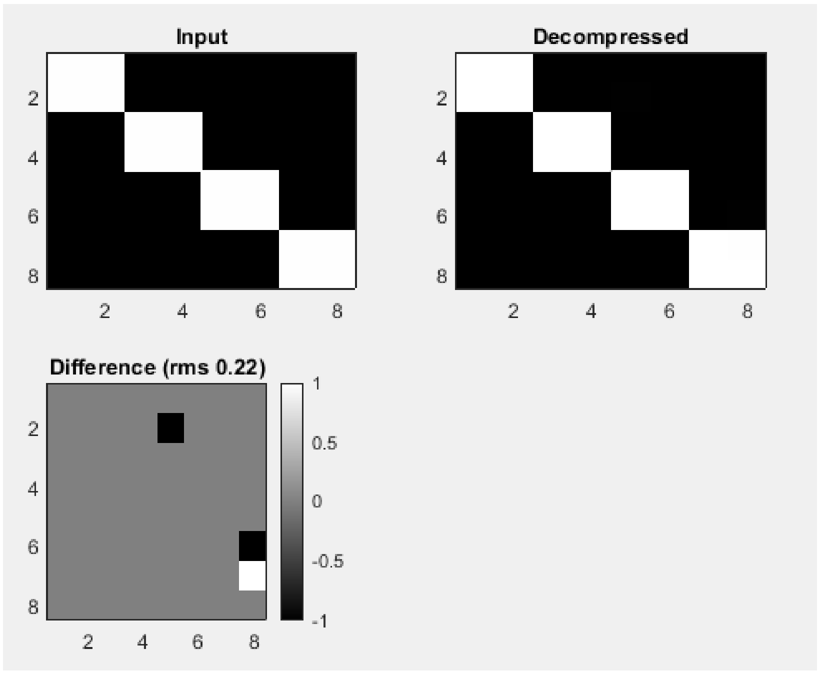


Results (Q=80)

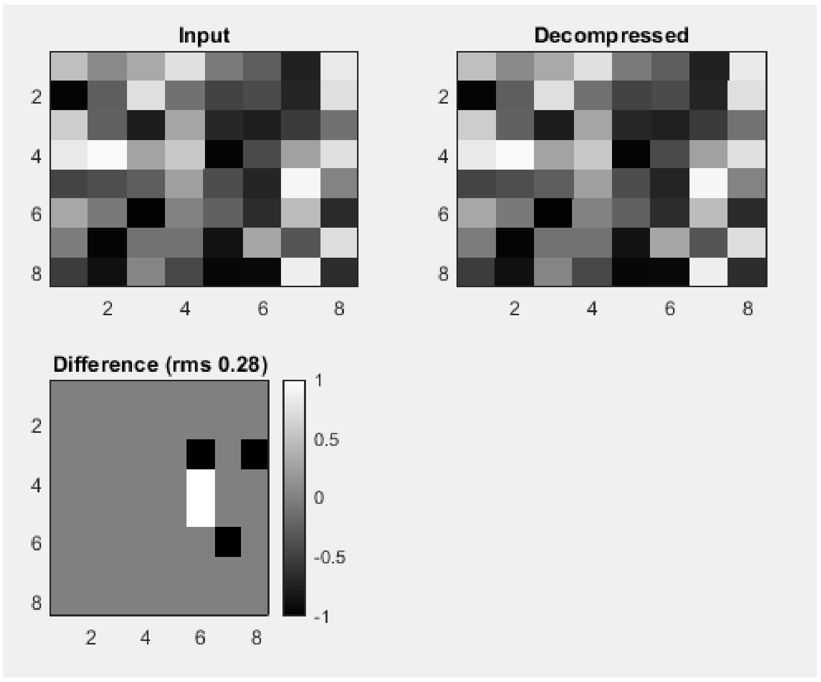

Input
Decompressed
Difference (rms 7.00)
2
4
6
8
15
10
5
0
-5
-10
-15


Input
Decompressed
Difference (rms 6.24)
2
4
6
8
10
5
0
-5
-10
-15

Results (Q=99)



Input
Decompressed
Difference (rms 0.22)
2
4
6
8
1
0.5
0
-0.5
-1



Input
Decompressed
Difference (rms 0.28)
2
4
6
8
1
0.5
0
-0.5
-1

Comments

The `djpeg_8x8` function successfully computed the quantised coefficients for a given 8x8 jpeg region. `djpeg_8x8()`, when run using `jpege_8x8_test()`, took 0.009s. The line that took the most time was the function call to `idct()`, which inverses the discrete cosine transform. This function call creates the tile based on the z coefficients.

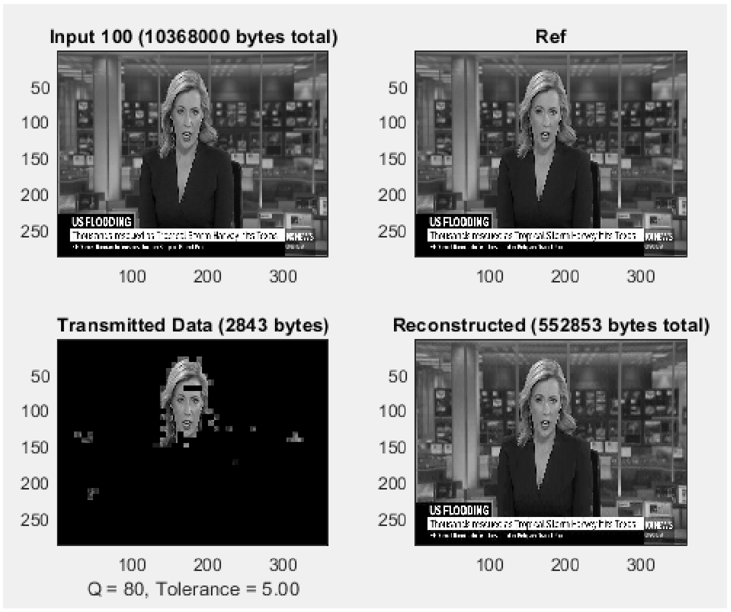
With the quality set to 20, the decompressed images have a lot of visible artifacts, some pixels are completely blended with their neighbors - producing an image that is very different from the input image. The difference between the grey levels of the test patches have a maximum of 50 and a minimum of -100 for both images.

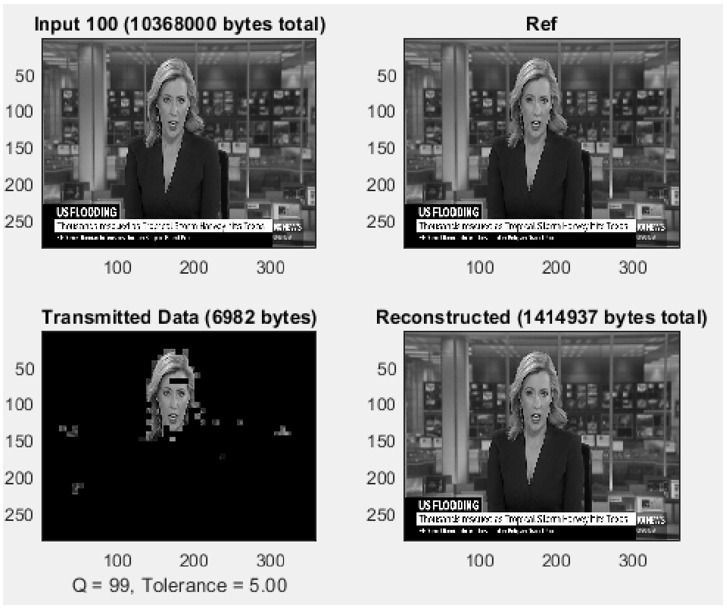
When the quality is set to 80, the decompressed images have some visible artifacts but the individual differences between the tiles are not that much, with the maximum difference being around 10 and 15 grey levels.

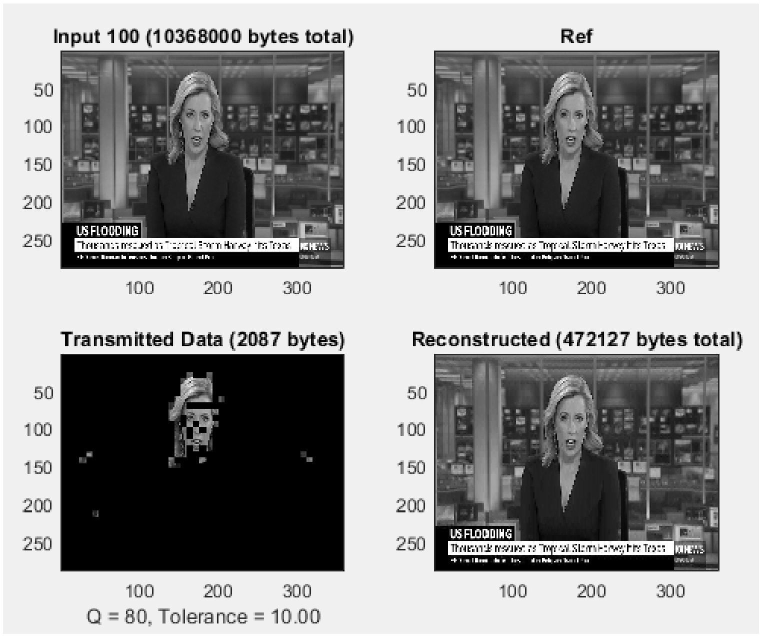
Setting the quality to 99, the decompressed image is almost exactly the same as the input image. The maximum difference of the test patches between the two images is around 1 grey level.

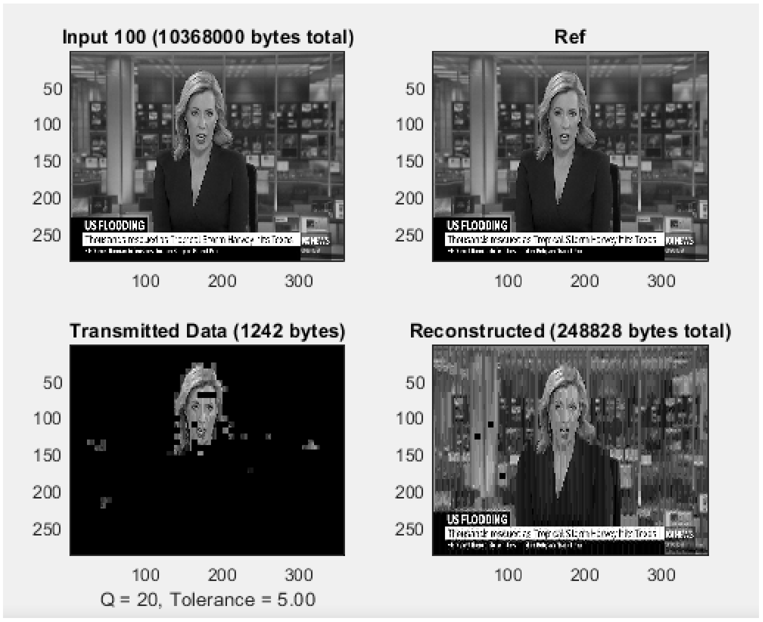
MPEG Test

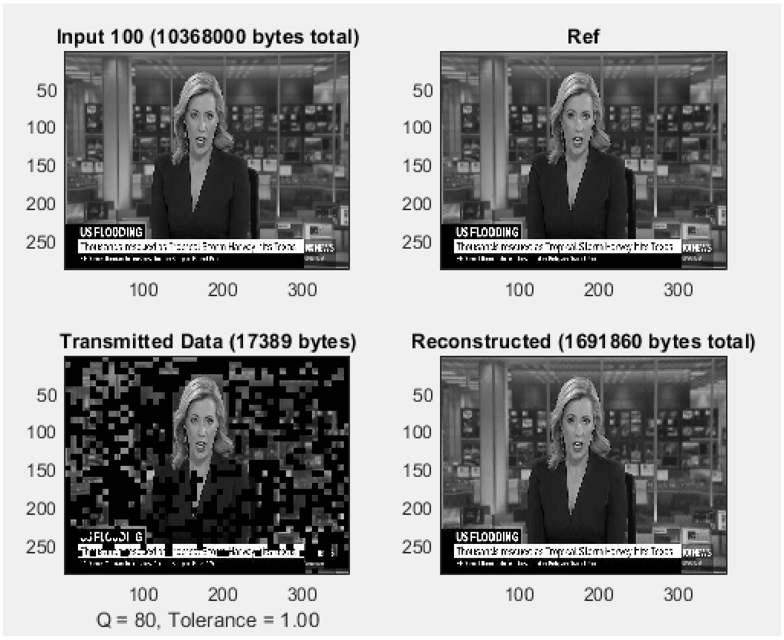
Results


Input 100 (10368000 bytes total)
Ref
Transmitted Data (2843 bytes)
Reconstructed (552853 bytes total)
Q = 80, Tolerance = 5.00


Input 100 (10368000 bytes total)
Ref
Transmitted Data (6982 bytes)
Reconstructed (1414937 bytes total)
Q = 99, Tolerance = 5.00


Input 100 (10368000 bytes total)
Ref
Transmitted Data (2087 bytes)
Reconstructed (472127 bytes total)
Q = 80, Tolerance = 10.00
50
100
150
200
250
100
200
300
US FLOODING


Input 100 (10368000 bytes total)
Ref
Transmitted Data (1242 bytes)
Reconstructed (248828 bytes total)
Q = 20, Tolerance = 5.00
50
100
150
200
250
100
200
300
US FLOODING


Input 100 (10368000 bytes total)
Ref
Transmitted Data (17389 bytes)
Reconstructed (1691860 bytes total)
Q = 80, Tolerance = 1.00
50
100
150
200
250
100
200
300
US FLOODING

Comments

With Q = 80, and tolerance = 5.0, hereby named Test 2B.1, the total time to run `mpeg_test()` is 148.199s. The majority of the time is taken by `Huff06()` which is called by both `simple_mpeg()` and `simple_dmpeg()`. `simple_mpeg()` took 49.218s, with the majority of the time taken by `Huff06()`. `simple_dmpeg()` took 86.640s, with 91.0% of the total time, 78.833s taken by `Huff06()`. `djpeg_8x8()` took 8.8% of the time, 7.665s.

After 100 frames, the total amount of bytes transferred was 552,853 bytes. This saves a total of 9,815,147 bytes. This method removes 94.67% of the image data, greatly reducing the bandwidth needed to transfer the 100 frame video.

Setting the Q to 99, with the tolerance still at 5.0, the total time to run `mpeg_test()` balloons to 336.237s. Once again, the same distribution of time was present, with Huff06 taking the majority of the slice. The total amount of bytes transferred was 1,414,937, a 2.5 times increase from 2B.1. The total percentage of bytes saved reduced to 86.35%.

Setting the Q to 20, with the tolerance still at 5.0. The total time to run `mpeg_test()` reduced to 82.624s. The number of bytes transferred also decreased to 248,828. However, the visual quality of the video greatly suffered.

Setting the Q to 80, with the tolerance to 10, The total time spent decreased from 148.199s to 132.929s. The total number of bytes transferred also decreased to 472,127, an 85% decrease from Test 2B.1. There was no noticeable decrease in video quality.

Setting the Q to 80, with the tolerance set to 1, the total time spent increased from 148.199s to 424.307s. The total number of bytes increased to 1,691,860 bytes.

From these tests, a conclusion can be drawn that tolerance is inversely proportional to the total time spent and it is also inversely proportional to the total number of bytes sent. On the other hand, quality is proportional to both the time spent and total number of bytes sent.

2C - Written Questions

1. In terms of the simple linear transformations ‘translation’, ‘isometric’, ‘linear conformal’, ‘affine’ and ‘projective’. Ignoring lens distortion, which transformations best describe the relationships between images in the following situations (you must explain your reasoning for each your answers.):

A. A camera mounted on a telescope looking up at a celestial body (eg. Jupiter). The telescope can tilt up and down and left and right (assume the scene itself is static).

   - The transformation that would best describe this relationship would be similarity. Tilting the telescope in any of the four directions would capture rotation and translation but wouldn’t preserve the euclidean distance.

B. A series of photographs of a document sitting on a desk taken from different positions throughout the room (just consider the transformation of the document itself, not the rest of the room).

   - The transformation that would best describe this relationship would be projective, since photographs of a document sitting on a desk taken from different positions would be considered a 3D projection of a 2D planar surface.

2. Linear transformations are transformations of the form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Assuming the following matrix T represents a 'linear conformal' transformation what are the underlying translation, rotation and scale values of this transformation?

$$T = \begin{bmatrix} 1.7321 & 1 & 10 \\ -1 & 1.7321 & -20 \\ 0 & 0 & 1 \end{bmatrix}$$

You must clearly state your reasoning for each of the four estimates.

$$x\ Translation = 10$$
$$y\ Translation = -20$$
$$Rotation = 30^\circ$$
$$Scale = 2$$

Proof:

Definition of Linear Conformal Transformation:

$$T = \begin{bmatrix} sR_2 & t \\ 0 & 1 \end{bmatrix}$$

Definition of Rotation Matrix

$$R_2 = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

Therefore:

$$T = \begin{bmatrix} s(cos\theta) & s(sin\theta) & t_x \\ -s(sin\theta) & s(cos\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Given:

$$T = \begin{bmatrix} 1.7321 & 1 & 10 \\ -1 & 1.7321 & -20 \\ 0 & 0 & 1 \end{bmatrix}$$

Hence:

$$s(cos\theta) = 1.7321$$
$$s(sin\theta) = 1$$
$$t_x = 10$$
$$t_y = -20$$

Solving for theta:

$$s = \frac{1.7321}{cos\theta}$$
$$s = \frac{1}{sin\theta}$$
$$\frac{1.7321}{cos\theta} = \frac{1}{sin\theta}$$
$$\frac{sin\theta}{cos\theta} = \frac{1}{1.7321}$$
$$tan\theta = \frac{1}{1.7321}$$
$$\theta = arctan(\frac{1}{1.7321})$$
$$\theta = 30^\circ$$

Solving for scale:

$$s = \frac{1}{sin(30^\circ)}$$

$$s = 2$$

3. Bilinear interpolation uses a 2x2 patch of image data to estimate pixel values after a transformation has been applied. If the 2x2 patch of an image for coordinates (20,10) to (21,11) is given by the grey level values:

128	64
64	32

What are the nearest and bilinear estimates for pixel (20.6, 10.2) ?

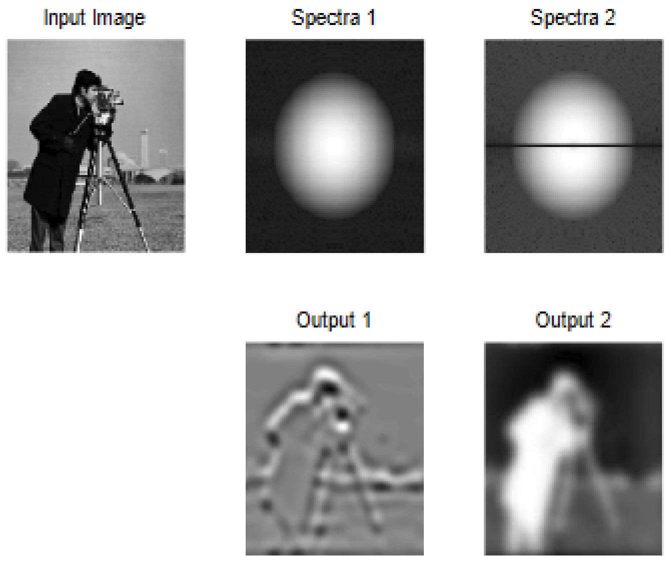
Nearest Neighbour:

$$N(20.6, 10.2) = I([20.6], [10.2])$$
$$N(20.6, 10.2) = I(21, 10)$$
$$N(20.6, 10.2) = 64$$

Bilinear:

$$x = \lfloor 20.6 \rfloor = 20$$
$$y = \lfloor 10.2 \rfloor = 10$$
$$a = 20.6 - 20 = 0.6$$
$$b = 10.2 - 10 = 0.2$$
$$\begin{gathered} I(20.6, 10.2) = (1-a)(1-b)I(x,y) + \\ (1-a)(b)I(x, y+1) + \\ (a)(1-b)I(x+1, y) + \\ (a)(b)(x+1, y+1) \end{gathered}$$
$$\begin{gathered} I(20.6, 10.2) = (1-0.6)(1-0.2)(128) + \\ (1-0.6)(0.2)(64) + \\ 0.6(1-0.2)(64) + \\ (0.6)(0.2)(32) \end{gathered}$$
$$I(20.6, 10.2) = 80.64$$

4. An image has been processed using two different filters. The spectra of each filter is shown along with the two outputs. Based on the filter spectra carefully explain which filter produced which output:



Which spectra gave rise to which output?

- Spectra 1 gave rise to Output 2, while Spectra 2 gave rise to Output 1. We can conclude that Spectra 1 gave rise to Output 2 because of the abundance of black and white in the spectra. This in turn resulted in the cameraman being highlighted in white with a mostly black background and a small amount of gray. On the other hand, we can conclude that Spectra 2 gave rise to Output 1 because of the mostly gray background visible in the image, with the black line in the middle indicating the presence of blacks in the edges of the objects in the image.