CMPE 362 - HW 2 Report

Muhammet Ali Topcu - 2020400147

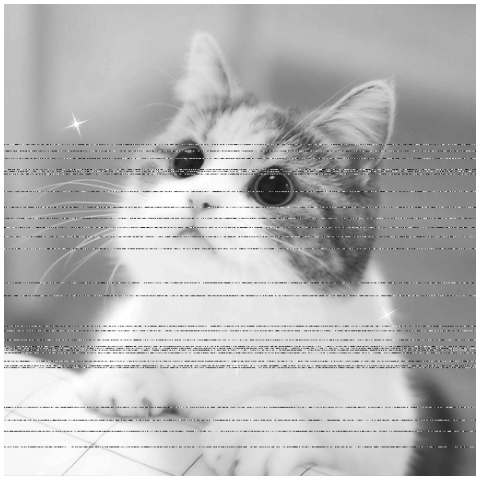
Project Description

Imagine you are transmitting an image over a channel but during this transmission, an error has occurred and image data is corrupted. What might be a clever way to hide this problem and show the image more or less similar to the original data? In this homework, you will implement one solution to this task and simulate this process.

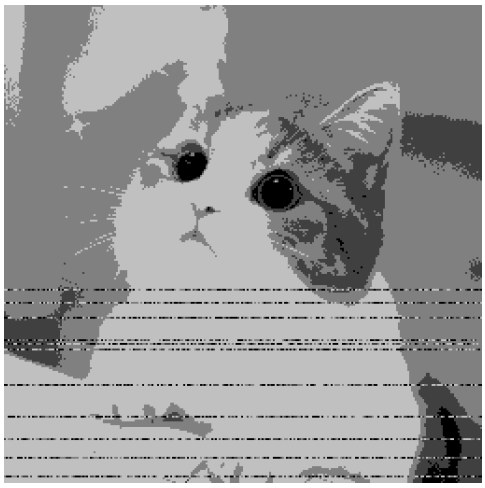
Core algorithm is as follows:
1. Create 4 copies of the original image (downsampled half the size) (Downsampling algorithm: select only the odd numbered rows and columns, check PS3 materials) . Concatenate these 4 images into a single one as in the figure.
2. Get the most significant n bits of the downsampled image and hide this data in least significant n bits of the original image.
3. Transmit this new image over a channel (we will just assume this has happened as we are only imagining a situation)
4. An error has occurred. Let's, for example, say 30 rows of the original image is corrupted with random numbers. (You simulate this manually)
5. Select a downsampled copy in an uncorrupted quadrant and extract the hidden data from least significant n bits. Since they are in least significant n positions, don't forget to shift these bits back to their original positions. Now, upsample back to original size using copying each pixel 4 times. (this method of upsampling is in PS3 materials.).
6. Replace corrupted image with the image you got from previous step.

Implementation:

Firstly, program reads the given png image and converts it to gray scale image. Then, one copy of the image is downsampled to store in the original image. After downsampling, 4 small copy of the image is stored within the original image and sent to the filter. In the filter, the transmitted image will be corrupted by row randomly. An example corrupted image is (a):



(a)



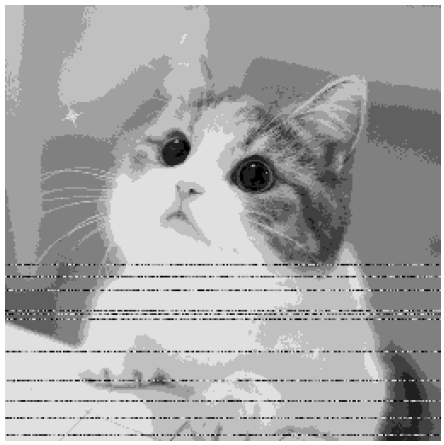
(b)

As it could be seen, the upper half of the image is less corrupted. So, I picked the image, which
will be recovered, from the upper part. (b) is an example recovered image for n = 2.

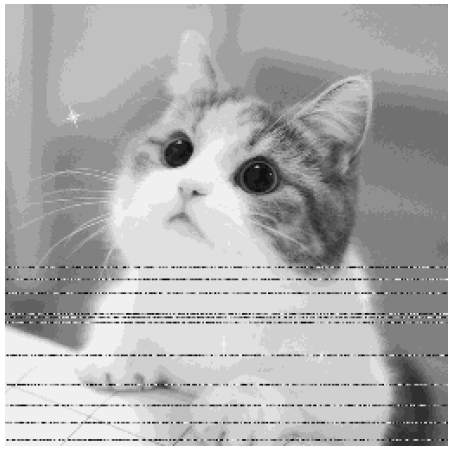
Results:

CAT:

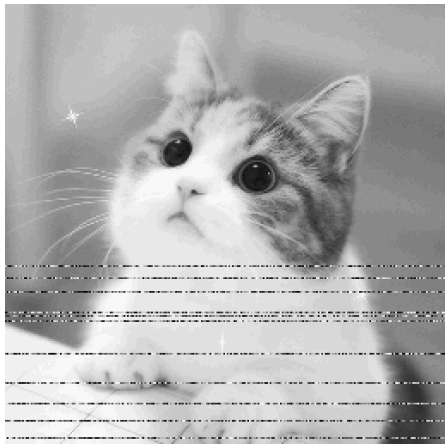
n = 2



n = 3



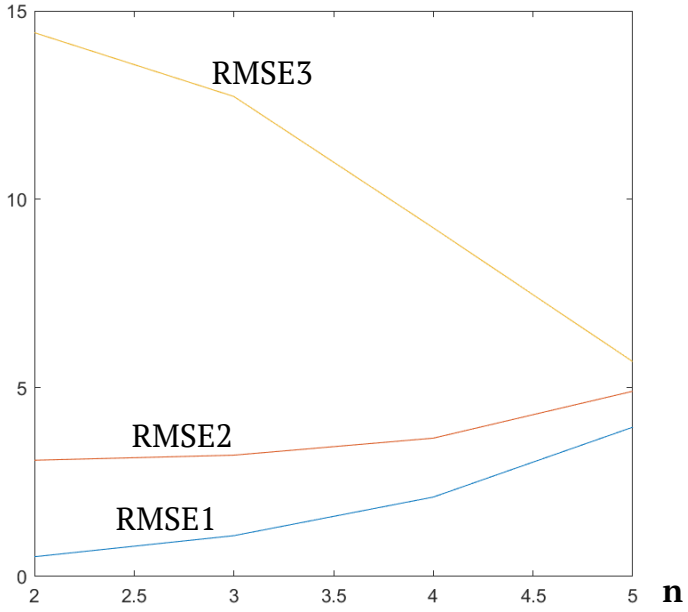
n = 4



n = 5

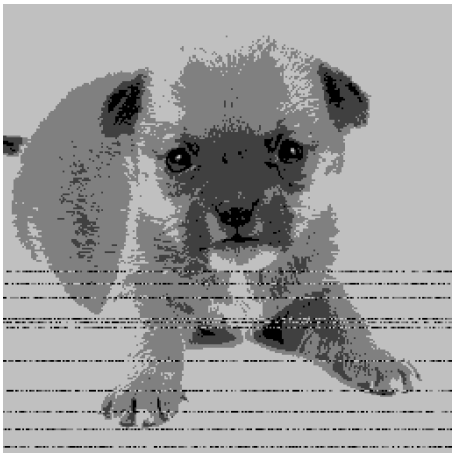
Plot & Table:

RMSE



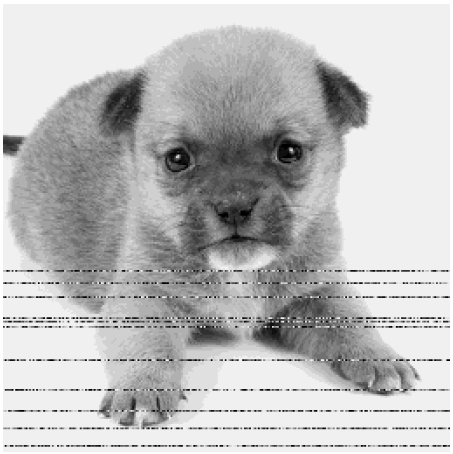
	RMSE1	RMSE2	RMSE3
n = 2	0.5257	3.0841	14.4222
n = 3	1.0838	3.2183	12.7301
n = 4	2.1075	3.6694	9.2474
n = 5	3.9606	4.9138	5.6988

DOG:



n = 2

n = 3

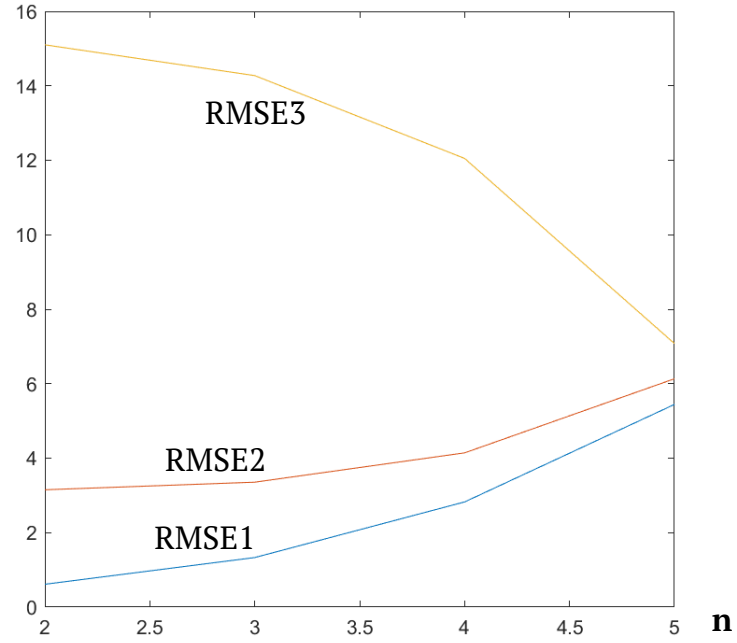


n = 4

n = 5

Plot & Table:

RMSE



	RMSE1	RMSE2	RMSE3
n = 2	0.6139	3.1512	15.1006
n = 3	1.3312	3.3552	14.2742
n = 4	2.8247	4.1424	12.0502
n = 5	5.4431	6.1328	7.0782

OTTER:

n = 2

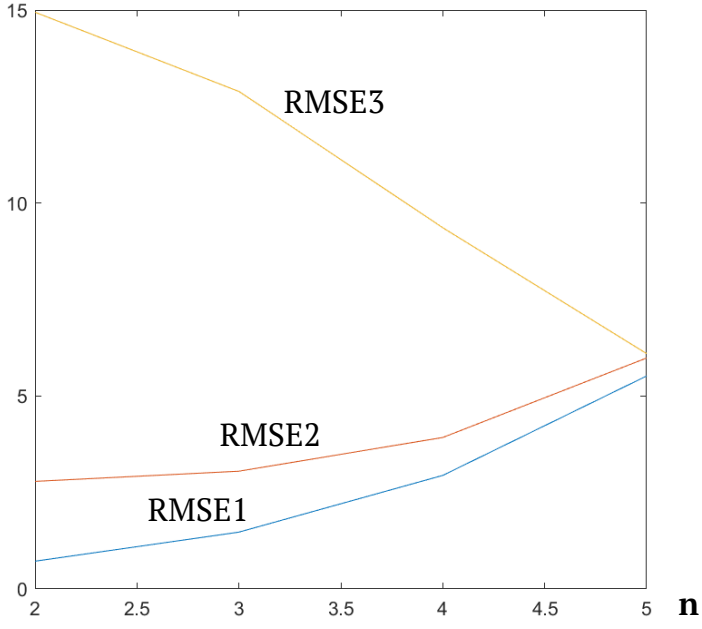
n = 3

n = 4

n = 5

Plot & Table:

RMSE



	RMSE1	RMSE2	RMSE3
n = 2	0.7195	2.7877	14.9443
n = 3	1.4727	3.0513	12.8917
n = 4	2.9430	3.9252	9.3601
n = 5	5.5190	5.9847	6.0983

Commentary:

It is obvious in the plots above that as n increases RMSE1 and RMSE2 values increases, RMSE3 values decreases.

So, as we increase bits stored in least significant bits of the original image, RMSE values between original grayscale image and transmitted image (RMSE1) and between original grayscale image and corrupted transmitted (RMSE2) increases since we use more bits to store the downsampled images.

On the other hand, as we increase bits stored in least significant bits of the original image, RMSE values between original grayscale image and recovered image (RMSE3) decreases since we could recover the image from the least significant bits and as n increases, we store more information about the original image.