Bit error rate = 1 – (total number of bits received correctly) /   
(total number of transmitted bits)

Data-rate = Total number of bits received correctly per second

**Implementation details**:

We use a grid of size 1. I’m testing this on an Nokia 7.1, which is an Android phone and its camera can’t achieve 60 fps, so I’m doing these experiments with 30 fps, which is kind of sad (feels Android man). This means that the video has to be 30 fps in order to actually capture anything. The camera quality is also sort of bad, so the results aren’t very great.

Anyways, each test goes through one round of “preamble” detection where it detects the starting screen, which is initially blank.

After that, we go through the detection using BFSK and at the end, it returns to the preamble phase and the detection terminates. We subtract the average intensity detected from every sample and perform FFT from that. Another approach that probably would’ve worked better is if I took the (minimum + maximum) / 2 and subtracted by that.

The string I am detecting is: ‘plzwork’, which has a binary representation of: 01110000011011000111101001110111011011110111001001101011, which has 56 bits and was translated in seconds of video written using OpenCV VideoWriter.

I would have used a longer string, but it’s running at 30 fps, so that would’ve taken more time. I may run with a longer string if I have time, but this is what I have.

Also note that we calculate intensities as the sum of the color channels for each pixel in our image, as described in the paper.

**Results**

**Note**: Incorrect portions will be highlighted.

**Correct string**: 01110000011011000111101001110111011011110111001001101011

**Case 1**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Distance | Bits received | string | Data-rate  (correct bits/s) | Bit Error Rates |
| 20cm | 000111011101101101111110011101110110111101110010011010110 | Û~work |  |  |
| 40cm | 000101000110110001111010111111110110111111111111010001110 | lzÿoÿG |  |  |
| 80cm | 11110000011011000111101111110111011011110111001101111011 | ðl{÷os{ |  |  |
| 120cm | 11110000010101111111101010110111010100001110001101001000 | ��¿Õº |  |  |

**Case 2**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Distance | Bits received | string | Data-rate  (correct bits/s) | Bit Error Rates |
| 20cm | 111100000110110001111010011101110110111101110010011010110 | ðlzwork |  |  |
| 40cm | 011100000110110001111010011101110111111101110010011010110 | plzwrk |  |  |
| 80cm | 011111111110110001111010011101110110111101110010011010110 | ìzwork |  |  |
| 120cm | 00000001111011000111101001110111011011110111001001101011 | ìzwork |  |  |

**Explanation**: Note that higher data rates correspond with lower bit error rates. We observe that for , we exhibit generally lower bit error rates and higher data rates than . This makes sense, as the differences in perceived intensity is higher for larger , since the different intensities exist on a larger range and can be better recognized/distinguished. We also see performance degrade as the distances increase, which makes sense because there was a lower portion of the image that was actually the screen vs. everything else, so differences between intensities would be relatively smaller.