Digital Signal Processing - Assignment 4a

Joshua Perrett (jdp63)

January 2023

Introduction

For the mini-project, I opted for the first option: "HDMI video cable eavesdropping". The steps provided are fairly clear, and so I will describe my results more or less in the order that corresponds to the instructions given.

Manual Adjustment of Parameters

Resampling

To display an image, the data needed resampling to some frequency f_r that was a multiple of f_p (to have an integer number of samples per line). I chose $fr = 2 * f_p$, under the assumption that the highest frequencies in the data that were worth preserving were f_p (and using the Nyquist criterion). **TODO** try again with $f_r = f_p$, and make the effort to bandpass filter it

I initially tried performing perfect sinc interpolation, which worked fine for a test input, but took far too long on the IQ data (even for just a couple of frames).

```
function change_fs(old_len, old_ys, old_fs :: Int, new_fs :: Int)
    new_len = floor(Int, old_len * new_fs/old_fs);
    new_xs = collect(0:new_len-1) / new_fs;

    new_ys = zeros(new_len)
    for i=0:old_len-1
      new_ys += old_ys[i+1] * sinc.(old_fs * (new_xs .- (i / old_fs)));
    end

    return new_len, new_xs, new_ys
end;
```

After looking for more computationally-feasible approaches to resampling, I found a resource that implied that changing "the sampling rate by a rational factor" required interpolating and then decimating [1]. If the rational factor has a very large numerator, this process can require a great deal of memory. To mitigate this, the process can be divided into multiple stages, by splitting the factor into a series of smaller factors with smaller numerators and denominators [1]. This seemed surprisingly complicated for such a straightforward operation, but it appears (from reading the source code) that the Julia DSP library uses this method exactly (**TODO** cite source). **TODO** details - `resample` function I think? By default uses a Kaiser filter

TODO if you get around to it, show and compare different ways of resampling (linear, sinc, kaiser)

Parameter Values

Adjusting by eye, I found that f_p was +0.1% over the expected 25.175 MHz. By vertically joining two separate frames that were several frames apart, I updated my estimate to +0.0997%.

Adjusting by **TODO** discuss alignment (achieved by discarding samples). probably don't need to discuss too much here

TODO show code, show figure

Result

Automation Adjustment of Parameters

Mention autocorrelation, how that has worked for you, why it worked. Talk about how you found the range of values for f_s/f_h and f_s/f_v. Show graphs.

TODO Autocorrelation of complex values?

Mention that the estimates for f_s/f_h and f_s/f_v are to the nearest integer, which limits the precision of your estimates for f_h and f_v. The f_v estimate is better than the f_h estimate (explain, show images). A rough explanation for why is that f_h is much larger than f_v, so f_s/f_h is much smaller, and this means that you lose more precision by rounding to the nearest integer.

Show various graphs

Talk about crosscorrelation between frames, and note how autocorrelation is really just crosscorrelation of a vector with itself. Talk about how you had to use the previous estimate (by performing autocorrelation) to narrow down the range of where the last frame might be (because that of course also depends on f_p, which is precisely what we are trying to find out, and simply setting bounds as we did for the autocorrelation would result in far too large a range, i.e. too much computation).

Show various graphs

TODO

Show averaged picture.

The RMS for the averaged image without phase unrotation was 3.86×10^{-6}, with phase unrotation it was 3.78×10^{-5}.

Conclusion

References

[1] Resampling, Mar 2017.