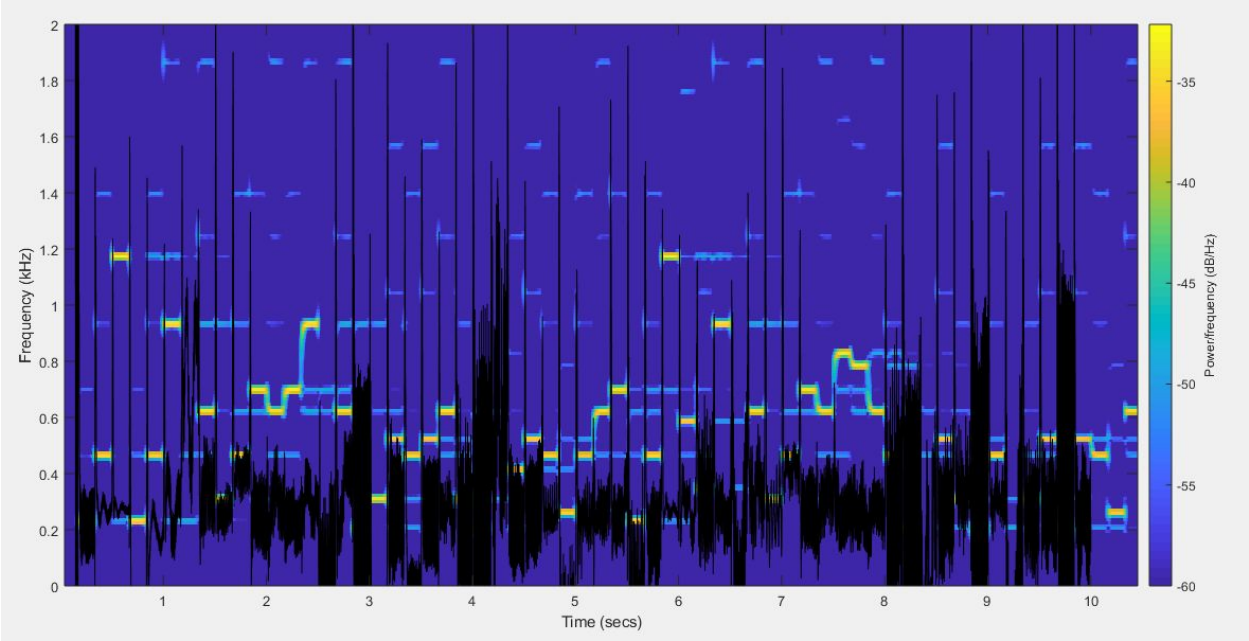
Brandon Reed
A01577421

Computer Assignment 5 Report

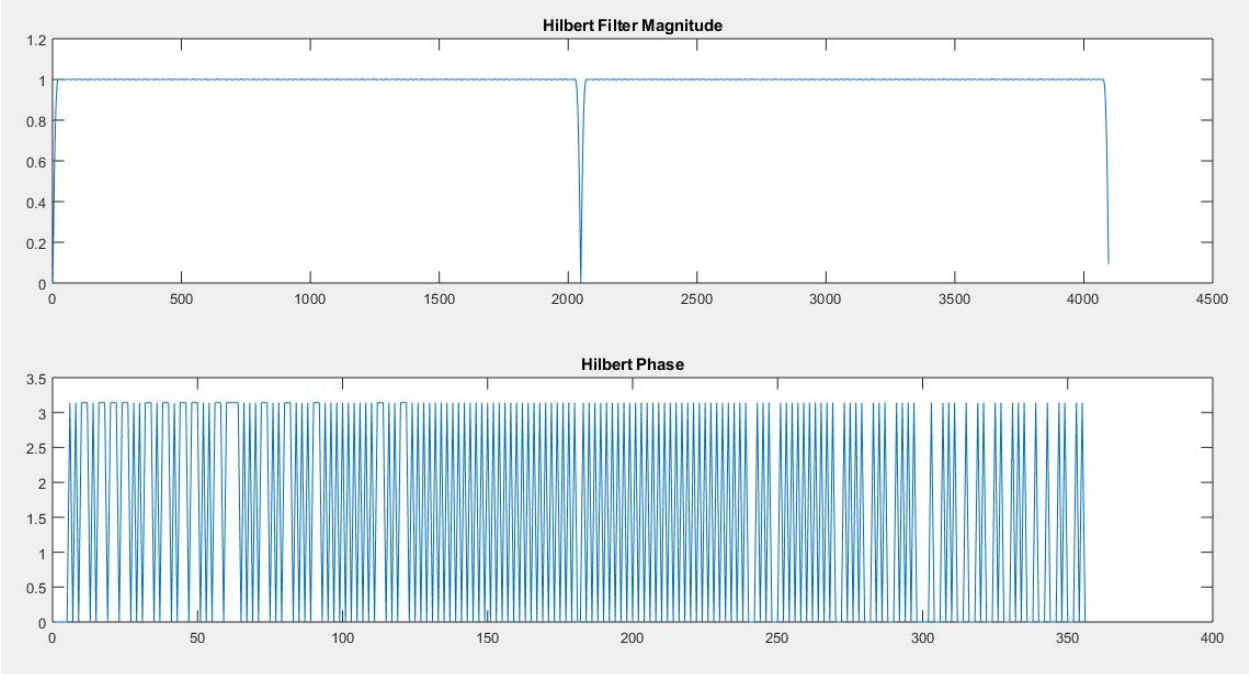
Spectrogram



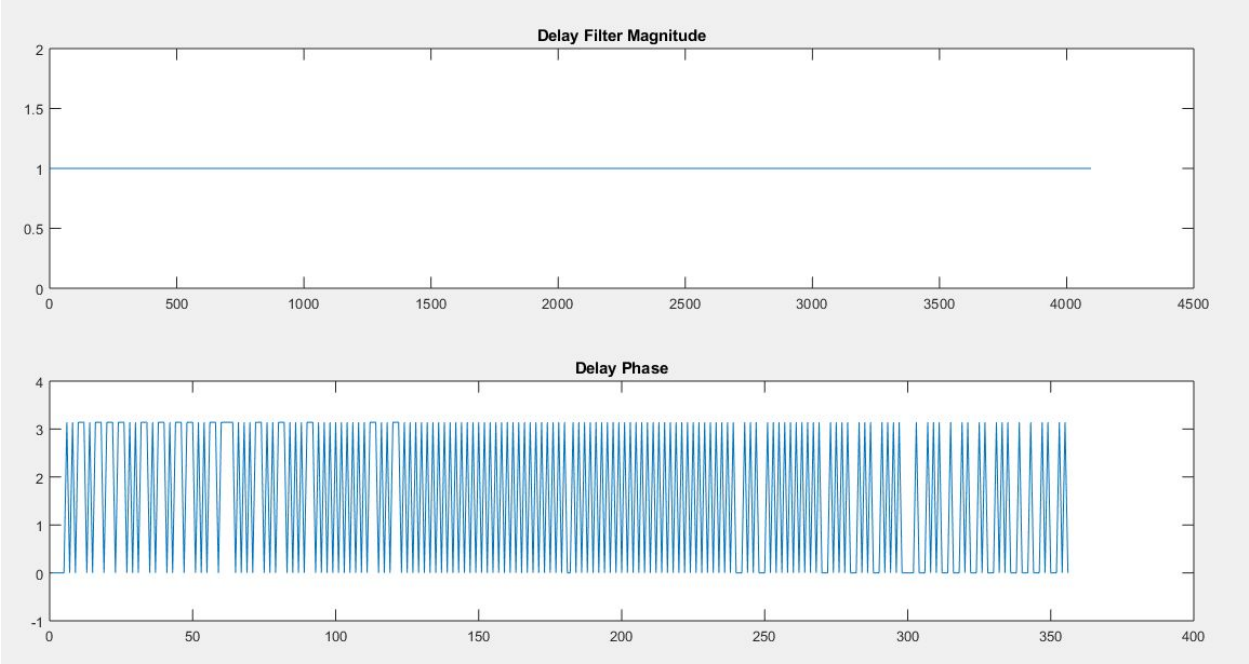
filtered instantaneous frequency in black

The instantaneous frequency is just a shifted version of the spectrogram. It displays the dominant frequencies(y axis) at any given time(x axis)

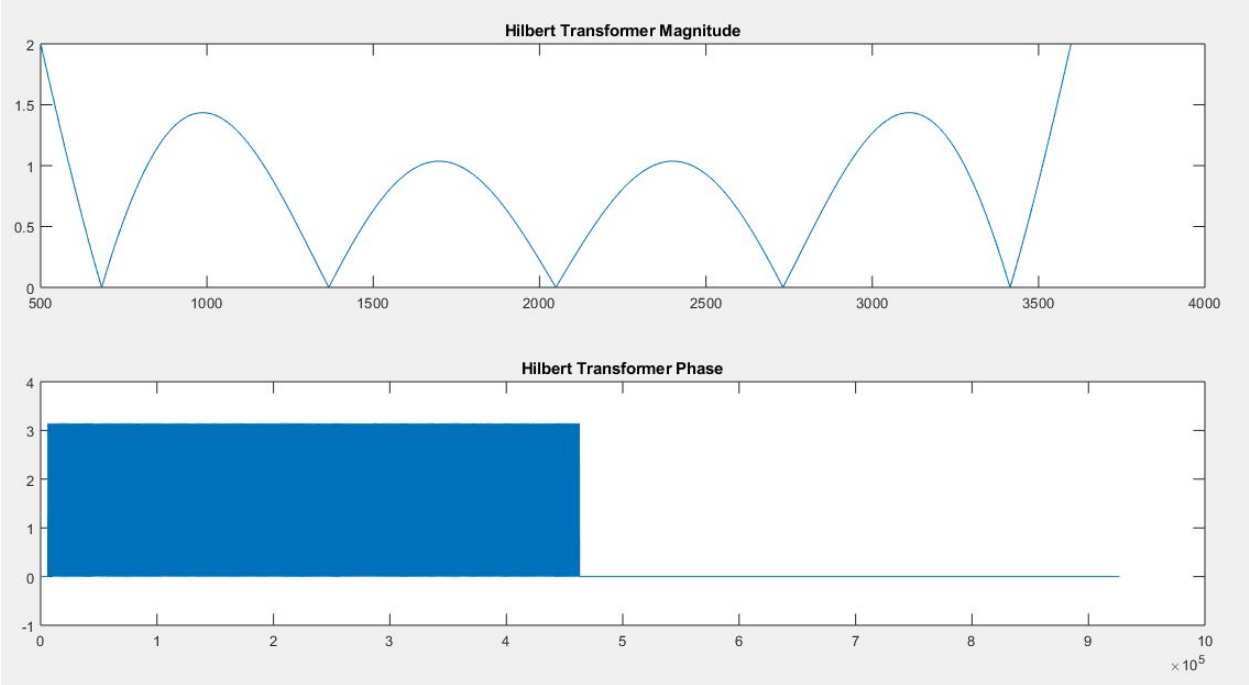
Hilbert Filter



Delay Function

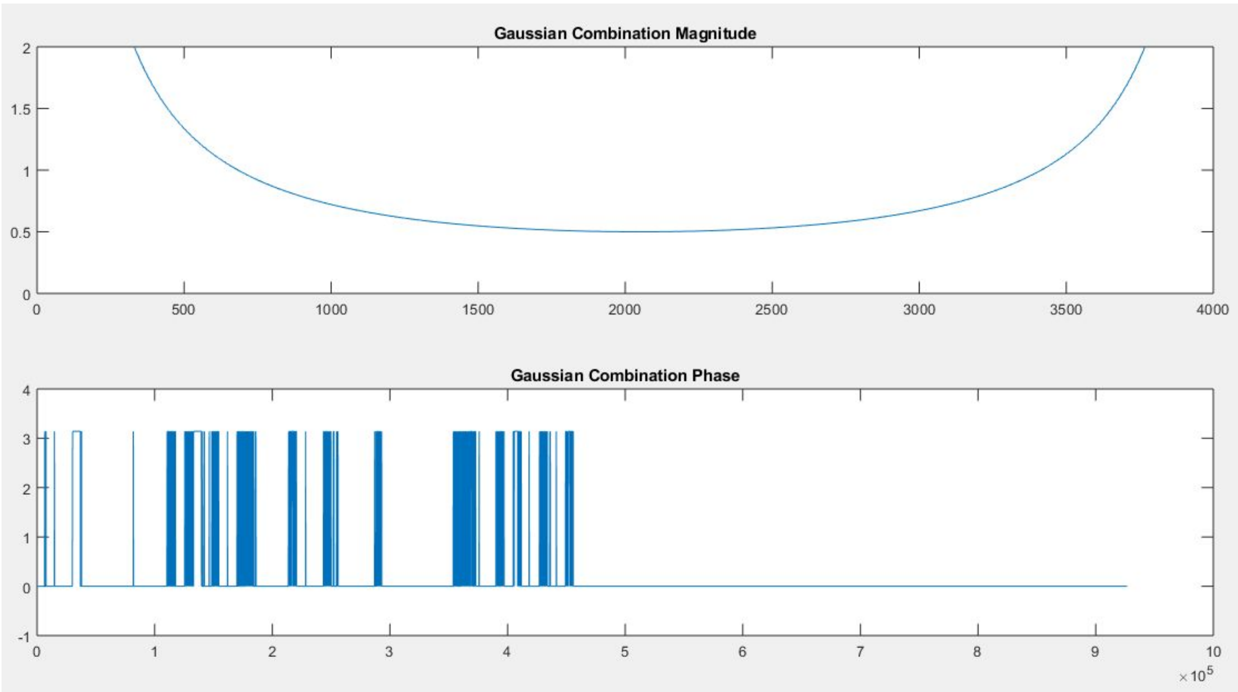


Hilbert Transformer



The Hilbert transformer is the convolution of the delayed firefly signal and a filtered version of the firefly signal. Which means that the the frequency response will be shifted and filtered in the shape of the hilbert filter.

Gaussian Combination



Appendix A
Matlab Code:

```
cd 'C:\Users\Brandon\Downloads\discreteSignals-master\discreteSignals-master\Lab5\CA5\CA5'

instant = "instantaneous.bin"
final = "final_product.bin"
hilbert = "hilbert_filter.bin"
gaussian = "gaussian_2_filter.bin"
delay = "delay.bin"

g = bin2audio("firefly.bin", "firefly.wav");
[fireflydata, fs] = audioread("firefly.wav");
nfft = 2^12;
w = hamming(nfft);
overlap = round(0.9*nfft);

spectrogram(fireflydata,w,overlap,nfft,fs,'yaxis','MinThreshold', -60)
hold on
finalData = fopen(finalfile,'rb');
if(finalData ==-1) fprintf('ERROR : Could not open file'); end
[xFinal, cnt2] = fread(finalData, inf, 'float');
last = 10*44100;
xFinal = xFinal(1:last);
y = [0:length(xFinal)-1]/44100;

plot(y,xFinal*7,'w')
ylim([0 2])
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Graphs of the hilbret filter
hilbertData = fopen(hilbert,'rb');
if(hilbertData ==-1) fprintf('ERROR : Could not open file'); end

[hilData, cnt4] = fread(hilbertData, inf, 'float');
hd = abs(fft(hilData,2^12));
y = [0:cnt4-1];
figure
subplot(2,1,1)
plot(hd)
hold on
title('Hilbert Filter Magnitude')
hold off
subplot(2,1,2)
plot(angle(hilData))
hold on
title('Hilbert Phase')

%Graphs of the delay filter

delayFile = fopen(delay,'rb');
if(delayFile ==-1) fprintf('ERROR : Could not open file'); end

[delayData, cnt4] = fread(delayFile, inf, 'float');
hd = abs(fft(delayData,2^12));
y = [0:cnt4-1];
figure
subplot(2,1,1)
plot(hd)
hold on
title('Delay Filter Magnitude')
ylim([0 2])
hold off
subplot(2,1,2)
plot(angle(hilData))
hold on
title('Delay Phase')
ylim([-1 4])

%graph of hilbert transformer

instantFile = fopen(instant,'rb');
if(delayFile ==-1) fprintf('ERROR : Could not open file'); end

[instantData, cnt4] = fread(instantFile, inf, 'float');
hd = abs(fft(instantData,2^12));
y = [0:cnt4-1];
figure
subplot(2,1,1)
plot(hd)
hold on
title('Hilbert Transformer Magnitude')
ylim([0 2])
hold off
subplot(2,1,2)
plot(angle(instantData))
hold on
title('Hilbert Transformer Phase')
ylim([-1 4])

%graph of result after gaussian transform

finalFile = fopen(final,'rb');
if(delayFile ==-1) fprintf('ERROR : Could not open file'); end

[finalData, cnt4] = fread(finalFile, inf, 'float');
```

```
hd = abs(fft(finalData,2^12));
y = [0:cnt4-1];
figure
subplot(2,1,1)
plot(hd)
hold on
title('Gaussian Combination Magnitude')
ylim([0 2])
hold off
subplot(2,1,2)
plot(angle(finalData))
hold on
title('Gaussian Combination Phase')
ylim([-1 4])
Hold off
```

Appendix B
C Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "dsp_file.h"

void convolution(char *inf1, char *inf2, char *outf);
void extract(char *inf1, char *inf2, char *outf);
void filter();
#define IOBUFFSIZE 15


int main()
{
	printf("now beginning\r\n");
	convolution("firefly.bin", "delay.bin", "delayconv.bin");
	printf("delay convolution complete\r\n");
	convolution("firefly.bin", "hilbert_filter.bin", "hilbertconv.bin");
	printf("hilbert filter convolution complete\r\n");
	extract("hilbertconv.bin", "delayconv.bin", "instantaneous.bin");
	printf("extraction complete\r\n");
	convolution("instantaneous.bin", "gaussian_2_filter.bin", "final_product.bin");
	printf("final gaussian filter convolution complete\r\n");

	printf("finished");
	while (1)
	{
	}

}

void extract(char *inf1, char *inf2, char *outf) {

	FILE *fin1, *fout, *fin2;
	if (NULL == (fin1 = fopen(inf1, "rb")))
	{
		printf("unable to read input file");
		return 0;
	}
	if (NULL == (fout = fopen(outf, "wb")))
	{
		printf("unable to write output file");
		return 0;
	}
	if (NULL == (fin2 = fopen(inf2, "rb")))
	{
```

```
		printf("unable to read impulse function file");
		return 0;
}

printf("input files read successfully\r\n");

int lx, lh, ly, lz;
dsp_file_header hin, hout, himp;
fread(&hin, sizeof(dsp_file_header), 1, fin1);
fread(&himp, sizeof(dsp_file_header), 1, fin2);
lh = himp.dim0;
lx = hin.dim0;
ly = lx + (lh - 1);
lz = lx + 2 * (lh - 1);
memcpy(&hout, &hin, sizeof(dsp_file_header));
hout.dim0 = ly;
fwrite(&hout, sizeof(dsp_file_header), 1, fout);

char len1[15];
char len2[15];

sprintf(len1, "%d", lh);
sprintf(len2, "%d", lx);

printf("dimensions from files read\r\n");
printf("length of file 1: ");
printf(len1);
printf("\r\nlength of file 2: ");
printf(len2);
printf("\r\n");

if (hin.nchan > 1) {
		printf("error signal has more than one chanel");
}

float *insig1 = (float*)calloc(lz, sizeof(float));
float *outSig = (float*)calloc(ly, sizeof(float));
float *insig2 = (float*)calloc(lh, sizeof(float));

//fread(insig1 + lh - 1, sizeof(float), lx, fin);

int i, m;
float pz, fi, pzold, pz0, M_PI;
M_PI = 3.1415926;
fread(insig1, sizeof(float), lh, fin1);
fread(insig2, sizeof(float), lh, fin2);

pzold = atan2(insig1[0], insig2[0]);
pz = atan2(insig1[1], insig2[1]);
fi = -(pz - pzold);
if (fi > M_PI) { fi -= 2.0*M_PI; }
else if (fi < -M_PI) { fi += 2.0*M_PI; }
pzold = pz;
outSig[0] = pzold;

for (i = 1; i < lh; i++) {
		pz = atan2(insig1[i], insig2[i]);
		fi = -(pz - pzold);
		if (fi > M_PI) { fi -= 2.0*M_PI; }
		else if (fi < -M_PI) { fi += 2.0*M_PI; }
		pzold = pz;
		outSig[i] = fi;
}

fwrite(outSig, sizeof(float), ly, fout);
```

```
	fclose(fin1);
	fclose(fin1);
	fclose(fin2);
}


void convolution(char *inf1, char *inf2, char *outf) {

	FILE *fin, *fout, *fimp;
	if (NULL == (fin = fopen(inf1, "rb")))
	{
		printf("unable to read input file");
		return 0;
	}
	if (NULL == (fout = fopen(outf, "wb")))
	{
		printf("unable to write output file");
		return 0;
	}
	if (NULL == (fimp = fopen(inf2, "rb")))
	{
		printf("unable to read impulse function file");
		return 0;
	}
	printf("input file and impulse response read successfully\n");

	int lx, lh, ly, lz;
	dsp_file_header hin, hout, himp;
	fread(&hin, sizeof(dsp_file_header), 1, fin);
	fread(&himp, sizeof(dsp_file_header), 1, fimp);
	lh = himp.dim0;
	lx = hin.dim0;
	ly = lx + (lh - 1);
	lz = lx + 2 * (lh - 1);
	memcpy(&hout, &hin, sizeof(dsp_file_header));
	hout.dim0 = ly;
	fwrite(&hout, sizeof(dsp_file_header), 1, fout);

	if (hin.nchan > 1) {
		printf("error signal has more than one chanel");
	}


	float *inSig = (float*)calloc(lz, sizeof(float));
	float *outSig = (float*)calloc(ly, sizeof(float));
	float *impSig = (float*)calloc(lh, sizeof(float));

	fread(inSig + lh - 1, sizeof(float), lx, fin);

	int i, m;
	float t;
	fread(impSig, sizeof(float), lh, fimp);
	for (i = 0; i < lh / 2; i++) {
		t = impSig[i];
		impSig[i] = impSig[lh - 1 - i];
		impSig[lh - 1 - i] = t;
	}

	for (i = 0; i < ly; i++) {
		t = 0.0;
		for (m = 0; m < lh; m++) {
			t += impSig[m] * inSig[i + m];
		}
		outSig[i] = t;
	}
	printf("convolution complete\n");
```

```
		fwrite(outSig, sizeof(float), ly, fout);

		fclose(fin);
		fclose(fin);
		fclose(fimp);
}


void filter()
{
		FILE *fin, *fout, *fimp;
		if (NULL == (fin = fopen("firefly.bin", "rb")))
		{
			printf("unable to read input file");
			return 0;
		}
		if (NULL == (fout = fopen("filterFireFly.bin", "wb")))
		{
			printf("unable to write output file");
			return 0;
		}
		if (NULL == (fimp = fopen("lpf_260_400_44100_80dB.bin", "rb")))
		{
			printf("unable to read impulse function file");
			return 0;
		}

		printf("input file and impulse response read successfully\n");



		int lx, lh, ly;
		dsp_file_header hin, hout, himp;
		fread(&hin, sizeof(dsp_file_header), 1, fin);
		fread(&himp, sizeof(dsp_file_header), 1, fimp);
		lh = himp.dim0;
		lx = hin.dim0;
		ly = lx;
		memcpy(&hout, &hin, sizeof(dsp_file_header));
		hout.dim0 = ly;
		fwrite(&hout, sizeof(dsp_file_header), 1, fout);

		if (hin.nchan > 1) {
			printf("error signal has more than one chanel");
		}

		float *h = (float*)calloc(lh, sizeof(float));
		float *g = (float*)calloc(lh, sizeof(float));
		float x[IOBUFFSIZE], y[IOBUFFSIZE];

		int n;
		float t;
		fread(h, sizeof(float), lh, fimp);
		for (n = 0; n < lh / 2; n++) {
			t = h[n];
			h[n] = h[lh - 1 - n];
			h[lh - 1 - n] = t;
		}


		int len, i, k = lh - 1;
		len = fread(x, sizeof(float), IOBUFFSIZE, fin);
		while (len > 0) {
			for (i = 0; i < len; i++) {
				g[k] = x[i];
```

```
                                t = 0.0;
                                for (n = 0; n < lh; n++) {
                                        t += h[n] * g[(n + k) % lh];
                                }
                                y[i] = t;
                                k--;
                                k = (k + lh) % lh;
                        }
                        fwrite(y, sizeof(float), len, fout);
                        len = fread(x, sizeof(float), IOBUFFSIZE, fin);
                }

                fclose(fin);
                fclose(fin);
                fclose(fimp);
}
```