24.05.2019

Cmpe362 Homework III Report

Frequency sound synthesis and lowpass filters

Serkan Özel
2015400123

Code Explanations:

1-SonifiedDeepSpace.m

```
hubble = imread('Hubble-Massive-Panorama.png');

amplitudes = zeros(900,1024);  % Frequencies and amplitudes are stored
seperately.
freqs = zeros(900,1024);

for i = 1:1024
   for j = 1:900
       if ([hubble(j,i,1) hubble(j,i,2) hubble(j,i,3)]< [30 30 30]) ~= [1 1
1] % If  Not Black
           amplitudes(j,i) = 11-ceil(j/90); % Give amplitude as stated in
desc
           freqs(j,i) = j; %% Give freq as stated in desc
       else
           amplitudes(i,j) = 0; %% Amplitude is 0 if it is black pixel
           freqs(j,i) = j;
       end
   end
end
data = zeros(1024,1001);

for i = 1:1024
    sum = zeros(1,1001);
    for k=1:900
        sum = sum + amplitudes(k,i)*cos(2*pi*freqs(k,i)*0:0.001:1);
    end
    data(i,:) = sum;
end
clear i;
concatanated = [];
for i = 1:1024
    concatanated = [concatanated data(i,:)];
end
audiowrite('hubble.wav',concatanated/max(abs(concatanated)),1001); % Outputs
a hubble.wav file
sound(concatanated,1001);
```

This code takes a 900x1024 pixel png file called “Hubble-Massive-Panorama.png” and based on an algorithm it synthesizes a wav file called “hubble.wav”. It works as follows:

For each column the program creates a one second audio with by using frequency spectrum. The Inverse Fourier Transform technique is used to transform sound signal from frequency domain to time domain. Then built in MATLAB function audiowrite() is used.

2-AdvancedPeakFreqFilter.m

```
newData1 = importdata("PinkPanther30.wav");


% No filter applied
pks = findpeaks(newData1.data);
no_filter_peaks_size = size(pks);

y1 = lowpass(newData1.data,1000,newData1.fs); %%
lowpass(data,limit_freq_of filter,sampling_freq)
y2 = lowpass(newData1.data,2000,newData1.fs);
y3 = lowpass(newData1.data,3000,newData1.fs);
y4 = lowpass(newData1.data,4000,newData1.fs);

% 1k Filter
pks1 = findpeaks(y1);
one_k_filter_peaks_size = size(pks1);  % size() finds number of peaks
given peak points as matrix

% 2k Filter
pks2 = findpeaks(y2);
two_k_filter_peaks_size = size(pks2);

%3k Filter
pks3 = findpeaks(y3);
three_k_filter_peaks_size = size(pks3);

%4k Filter
pks4 = findpeaks(y4);
four_k_filter_peaks_size = size(pks4);

plot(0:4,[no_filter_peaks_size(1) one_k_filter_peaks_size(1)
two_k_filter_peaks_size(1) three_k_filter_peaks_size(1)
four_k_filter_peaks_size(1)]);
```

This code is for observing the effects of lowpass filter to number of peaks of a signal. The built-in lowpass() function is used to implement this idea.

First, a sound file called “PinkPanther30.wav” is loaded. Number of peaks with no filter is measured using findpeaks() function.

Then, 4 lowpass filter functions are applied to the signal seperately each of them having different cut off frequencies from 1000 to 4000. Then number of peaks are plotted to see the result: