Creating a Live Motion Visualizer using STL Files and FFT

Table of Contents

Connecting to a Smartphone 1
Initializing a Live Combined Acceleration Plot 1
Initializing a Live Fast Fourier Transform Plot 2
Instantiating Live 3d Model Visuallizer 3
Create Live Updating Plot Data 5

Using the MATLAB mobile app and connecting it to this program, the accelerometer and orientation data from a smartphone is used to create a polychromatic visualization experience. There are two different ways the size and number of objects can be displayed in this visualizer. In this version of the code, there are three 3d objects being displayed at once, each one's size dependent on the acceleration on a different axis. However, there remains a combined acceleration amplitude (that removes acceleration due to gravity) that is commented out but can be applied to a single 3d object or all three if the user does not desire to have three independently changing objects. Other aspects of this visuallizer include rotating the 3d objects with the orientation of the phone around a central point outside the 3d object for an orbiting effect. Finally, there are 3 lights shining on the metallic textured objects whose colors depend on the frequency of phone movement.

Connecting to a Smartphone

The code below allows data to be streamed from a smartphone to MATLAB using the MATLAB mobile app. Once the app is downloaded the sensors for acceleration and orientation data must be enabled, and stream the data to MATLAB.

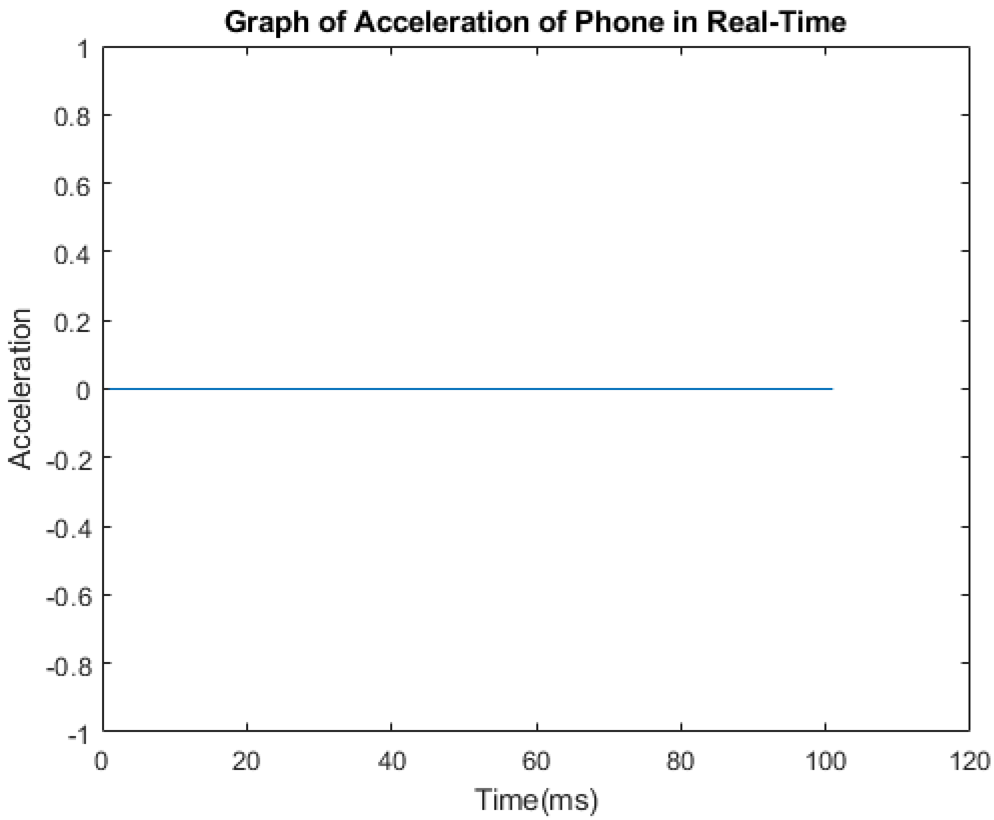
```
clear all
clear m
m = mobiledev;
m.AccelerationSensorEnabled = 1;
m.Logging = 1;
```

Initializing a Live Combined Acceleration Plot

Here the acceleration plot is initialized before it is live updated and displayed. The acceleration plot does not display the three individual acceleration by axis but the overall magnitude of the three accelerations summed together and removing gravity. The code for this live updating is further down.

```
data_accel = zeros(101,1); %zero points that will be replaced later
data_fft = zeros(101,1); %zero points that will be replaced later
% Initialize accerlation Plot
figure(1)
p = plot(data_accel); % define length of data_accel to be seen at a
 time
update_limit= 750; % update_limit in ms
g = 9.80865; % gravity constant
```

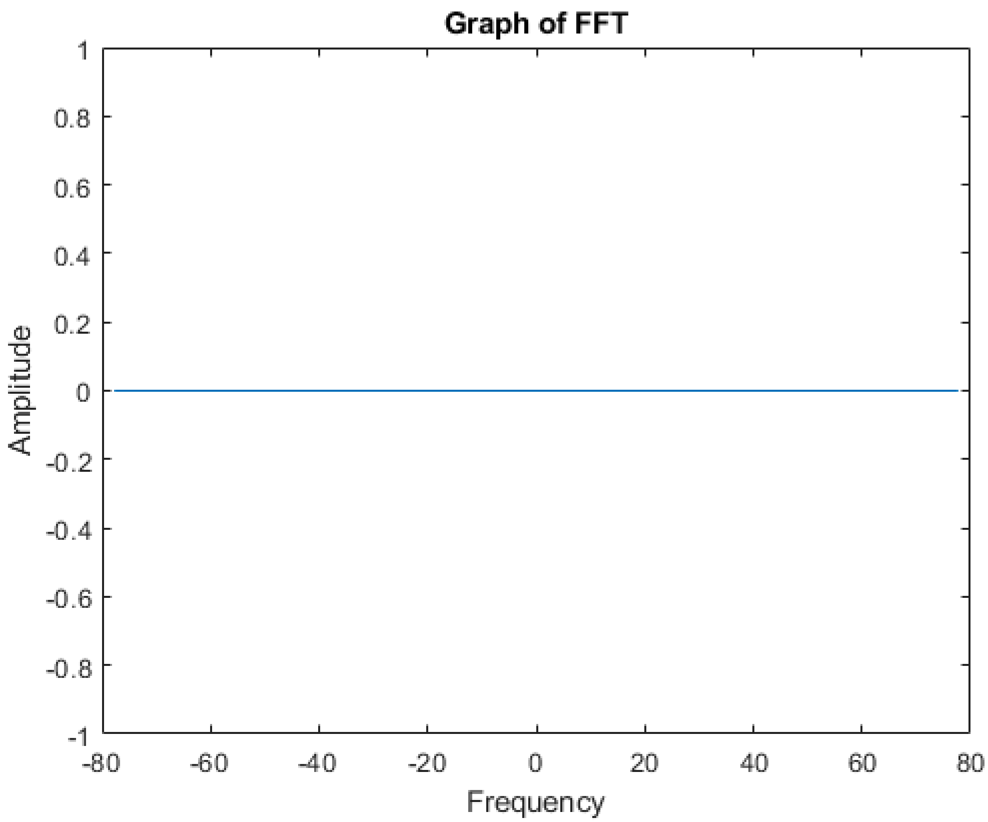
```
title("Graph of Acceleration of Phone in Real-Time")
xlabel("Time(ms)")
ylabel("Acceleration")
```



Initializing a Live Fast Fourier Transform Plot

Similar to above, this graph does not yet have the data but is setting up the plot to be filled later with the rest of the updates.

```
Fs = 25;
N = length(data_accel);
figure(2)
frequencies_shifted = linspace(-pi, pi-2/N*pi, N) + pi/N*mod(N,2);
frequency_fs = frequencies_shifted*Fs;
data_fft = (abs(fftshift(fft(abs(data_accel)))));
q = plot(frequency_fs,data_fft);
title("Graph of FFT")
xlabel("Frequency")
ylabel("Amplitude")
```



Instantiating Live 3d Model Visuallizer

Here three 3d objects can be made from an stl file, currently in the shape of a Neato vacuum cleaner. Simpler files will run significantly faster.

```
figure(3)
animal = stlread("neat0 v1.stl"); %load stl file
%extract points
 points1 = animal.Points ;
 points2 = animal.Points;
 points3 = animal.Points;
 %move object so that it starts at origin
 points1(:,1) = points1(:,1) - 22; %center object
 points1(:,2) = points1(:,2) - 22;
 points1(:,3) = points1(:,3) -14;
 points2(:,1) = points1(:,1) - 70;
 points2(:,2) = points1(:,2) - 70;
 points2(:,3) = points1(:,3) - 70;
 points3(:,1) = points1(:,1) + 48;
 points3(:,2) = points1(:,2) + 48;
 points3(:,3) = points1(:,3) + 56;
 points1 = [points1(:,1) points1(:,2) points1(:,3)];
 points2 = [points2(:,1) points2(:,2) points2(:,3)];
 points3 = [points3(:,1) points3(:,2) points3(:,3)];
cl = animal.ConnectivityList;
material metal %change 3d model specular value to higher
```

```
hold on
q_again1 = plot3(-1500*ones(101,1),frequency_fs+750,data_fft); %adding
 fft
q_again2 = plot3(frequency_fs+750, -1500*ones(101,1),
 data_fft); %adding fft
r = trimesh(cl,
 points1(:,1),points1(:,2),points1(:,3), 'LineStyle', 'none');
r2 = trimesh(cl,
 points2(:,1),points2(:,2),points2(:,3), 'LineStyle', 'none');
r3 = trimesh(cl,
 points3(:,1),points3(:,2),points3(:,3), 'LineStyle','none');
hold off


set(gca,'color', 'black'); % make background black
%define axes limits and hide
 axis([-1500 1500 -1500 1500 -1500 1500]);
 set(gca,'XTick',[], 'YTick', [],'ZTick', [])
 %instantiate lighting
 lighting gouraud
 a1 = light('Position',[1000 1000 -1000],'Style','local');
 a1.Color = [1 0 0];
 a2 = light('Position',[-1500 1000 1000],'Style','local');
 a2.Color = [1 0 1];
 a3 = light('Position',[-1000 -1000 1000],'Style','local');
 a3.Color = [0 0 1];

pause(1)
%time loop
tic
hold on
```

Create Live Updating Plot Data

```
while (toc < 10000)%run for 10000 secs
    %N = 101
    N = length(data_accel);

    %get new acceleration and orientation values
    [a,~] = accellog(m);
    [t,~] = orientlog(m);

   %if data_accel elapses updated time, update the plot from the right
    data_accel(1:length(a)) = sqrt(a(:,1).^2+a(:,2).^2+(a(:,3)).^2 -
g.^2);
    % data_accel is the combined magnitude of accelerations mentioned
    % earlier, however it is not used in the current version of the
 code.
    % Data may vary by phone since different models may have different
    % axes.
    data_accel1(1:length(a)) = a(:,1); %x axis
    data_accel2(1:length(a)) = a(:,2); %y axis
    data_accel3(1:length(a)) = a(:,3); %z axis
    data_theta1(1:length(t)) = t(:,1); %Azimuth
    data_theta2(1:length(t)) = t(:,2); %Pitch
    data_theta3(1:length(t)) = t(:,3); %Roll
```

```
    %Creating updated values for the FFT graph
    val = 100;
    if length(a) > val
        current_accel = data_accel(length(a)-val:length(a));
        data_fft = (abs(fftshift(fft(abs(current_accel)))));
    else
        data_fft = (abs(fftshift(fft(abs(data_accel)))));
    end

    %Decrease the amplitude of points between -5 and 5.
     freqs_to_filter = abs(frequency_fs) < 5;
    if abs(length(data_fft)) > freqs_to_filter
        data_fft(freqs_to_filter) = data_fft(freqs_to_filter)/50;
    end
    N = val + 1;

    %Update frequencies
    frequencies_shifted = linspace(-pi, pi-2/N*pi, N) + pi/N*mod(N,2);
    frequency_fs = frequencies_shifted*Fs;

    %Return max frequency and amplitude
    index_f = find(data_fft == max(data_fft), 1, 'first');
    defined_freq = abs(frequency_fs(index_f));
    defined_amp = max(data_fft);

    %Update accel graph
    p.YData = abs(data_accel);

    %Update FFT graph
    q.XData = frequency_fs;
    q.YData = data_fft;

    %Update FFT graph on 3d plot
    q_again1.YData = 15*frequency_fs;
    q_again1.ZData = 13*data_fft - 1500;
    q_again2.XData = 15*frequency_fs;
    q_again2.ZData = 13*data_fft -1500;
    %drawnow

    amp_constant = defined_amp/40;
    %drawnow

    %scaled_points is used for a single object being scaled by the
overall
    %graph amplitude.
    %scaled_points = amp_constant*points;

    %Scaling the three objects by acceleration along their respective
axes.
    scaled_points1 = (data_accel1(end))*points1;
    scaled_points2 = (data_accel2(end))*points2;
    scaled_points3 = (data_accel3(end))*points3;

    %Rotating the three objects
```

```
    xrot_points1 = rotx(-data_theta1(end))*scaled_points1';
    yrot_points1 = roty(-data_theta2(end))*xrot_points1;
    zrot_points1 = rotz(-data_theta3(end))*yrot_points1;
    new_points1 = zrot_points1';

    xrot_points2 = rotx(-data_theta1(end))*scaled_points2';
    yrot_points2 = roty(-data_theta2(end))*xrot_points2;
    zrot_points2 = rotz(-data_theta3(end))*yrot_points2;
    new_points2 = zrot_points2';

    xrot_points3 = rotx(-data_theta1(end))*scaled_points3';
    yrot_points3 = roty(-data_theta2(end))*xrot_points3;
    zrot_points3 = rotz(-data_theta3(end))*yrot_points3;
    new_points3 = zrot_points3';

    %Changing the color of the lights depending on the frequency of
    %movement. Colors are currently soft pastels.
    freq_constant = defined_freq/30;
    if freq_constant > 1
        freq_constant = 1;
    end

    col_constant = freq_constant;
    shift_spot = 170/255;
    shift_spot2 = 210/255;

    if col_constant > 1
        col_constant = shift_spot;
    end

    % Colors of lights
    if (freq_constant > shift_spot) && (freq_constant < shift_spot2)
        a1.Color = [abs(real(col_constant)) (abs(1 -
real(col_constant))) shift_spot];
        a2.Color = [abs(1 - real(col_constant)) shift_spot
abs(real(col_constant))];
        a3.Color = [shift_spot abs(real(col_constant)) abs(1 -
real(col_constant))];
    elseif freq_constant < shift_spot
        a1.Color = [shift_spot abs(1 - real(col_constant))
abs(real(col_constant))];
        a2.Color = [abs(real(col_constant)) abs(1 -
real(col_constant)) shift_spot];
        a3.Color = [abs(1 - real(col_constant)) shift_spot
abs(real(col_constant))];
    else
         a1.Color = [abs(1 - real(col_constant)) shift_spot
abs(real(col_constant))];
         a2.Color = [shift_spot abs(real(col_constant)) abs(1 -
real(col_constant))];
         a3.Color = [abs(real(1 - col_constant))
abs(real(col_constant)) shift_spot];
    end
```

```
    %set color of 3d plot
    set(gca,'color', 'black');

    %Update visualizer graph
    r.Faces = cl;
    r.Vertices = new_points1;
    r.FaceColor = [.71 .43 .47]; %rose gold
    r2.Faces = cl;
    r2.Vertices = new_points2;
    r2.FaceColor = [.93 .91 .67]; %pale gold
    r3.Faces = cl;
    r3.Vertices = new_points3;
    r3.FaceColor = [.70 .80 .88]; %steel blue
    drawnow
end
hold off
```

```
Invalid or deleted object.

Error in crabdance (line 167)
    q_again1.YData = 15*frequency_fs;
```

Published with MATLAB® R2020b