Time-Frequency Analysis with Gábor Transform

Casper Hsiao

2020-02-06

Abstract: The objective of this project is to implement time-frequency analysis on audio signals. In part one, a portion of Handel's Messiah was analyzed with Gábor transform. Different sampling rate were utilized to study the resolution of the result. Furthermore, different Gábor windows with different widths were implemented to analyze the signal. The results showed that the wider the Gábor window the less information resolution extracted from the time domain. In part two, two audio signal files were analyzed with Gábor transform to reproduce the music score of the audio. The music scores were reproduced successfully with a balanced time-frequency resolution.

I. Introduction and Overview

Audio signals are important in our lives as we use voice to communicate with each other. Every person's voice has their own unique characteristic signatures. Since audio signals are non-stationary signal, where the center frequency varies with time, time-frequency analysis is implemented to provide both time and frequency information of the signals. However, there is trade-off between time and frequency resolution of the Gábor transform.

II. Theoretical Background

II.A. Gábor Transform

The Gábor transform, also known as the short-time Fourier transform, was proposed by Gábor Dénes. The Gábor transform is developed due to the limitation of Fourier transform, the information loss in time domain. The key concept to Gábor transform is the windowed Fourier transform, where the signal is fragmented into short time windows. As showed in Eq. 1, Gábor modified the Fourier transform kernel so that it can localize both time and frequency.

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t) \tag{1}$$

The new term $g(\tau - t)$ was introduced to localize both time and frequency, which acts as a time filter for localizing the signal over a short window of time as shown in Eq. 3. The integration over τ allows the transformation to slide the short-time window over the entire signal. As a result, certain frequency information is captured at certain time instance, which produce a localization for both time and frequency. However, there exists a trade-off between the resolution of time and frequency. The resolution of the time is determined by the width of the short-time window, the narrower the window, the higher the resolution in time domain and the lower the resolution in frequency domain. There are some assumptions made on g which are g is real and symmetric with $||g(t)|| = 1$, $||g(\tau - t)|| = 1$, and $||\cdot||$ denotes the L_2 norm.

$$G[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-i\omega\tau}d\tau = (f, \bar{g}_{t,\omega}) \tag{2}$$

The discrete Gábor transform is computed through discretizing the frequency and time domain, which creates a lattice of time an frequency. This simplifies the Gábor transform as shown in Eq. 3. The m and n are the lattice integer of the frequency and time domain.

$$\tilde{f}(m,n) = \int_{-\infty}^{\infty} f(t)e^{i2\pi m\omega_0 t}g(t - nt_0)dt = (f, g_{m,n}) \qquad where \qquad g(t - nt_0) = e^{-a(t-nt_0)^2} \tag{3}$$

If $0 < t_0, \omega_0 < 1$, the signal is oversampled, which yields excellent localization of the signal. if $t_0, \omega_0 > 1$, the signal is under-sampled, which is unable to reproduce the signal information. The overlap between the Gábor time window ensures good resolution in time and frequency.

II.B. Wavelet Theory

The Gábor transform involves the Gábor window to filter out signals outside of the short-time window. The Gábor window consists of the translation, τ, and scaling, a, to capture the desired signal within the short-time window. The trade-off between resolution in time and frequency is determined by the scaling of the Gábor window. The fundamental principle of the wavelet theory is to vary the scaling of the Gábor window to obtain both excellent time and frequency resolution.

The Haar wavelet is a piecewise constant function as shown in Eq. 4. The sqaure wavelet is a piecewise cosntant function as shown in Eq. 5. Furthermore, the convolution of the Haar and sqaure wavelet is the triangular wavelet. The Mexican hat wavelet is shown in Eq. 6.

$$\psi(t) = \begin{cases} 1 & 0 \le t < 1/2 \\ -1 & 1/2 \le t < 1 \\ 0 & otherwise \end{cases} \tag{4}$$

$$\psi(t) = \begin{cases} 0 & t < 0 \\ 1 & 0 \le t \le 1 \\ 0 & t \ge 1 \end{cases} \tag{5}$$

$$\psi(t) = (1 - t^2)e^{-t^2/2} \tag{6}$$

III. Algorithm Implementation and Development

III.A. Part One

The portion of Handel's Messiah was loaded into the work space. The time domain and the audio signal was extracted from the loaded data, furthermore, the frequency domain was created from the time domain. The Gábor transform is discretized in the time domain with a sampling rate of $t_0 = 0.3719$. Using for loop, the Gábor transform is computed by multiplying the Gábor window, $g(t) = e^{-a(t-\tau)^2}$, with the audio signal at different translations, τ, and FFT each windowed signal. Furthermore, 3 different scaling, a, of the Gábor window were tested by implementing another for loop outside the Gábor transform. A lower sampling rate was also used to demonstrate the effect of under-sampling. Other types of wavelets, such as Haar, square, triangle and Mexican hat, were utilized as Gábor window for the transformation.

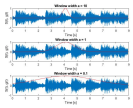
III.B. Part Two

Two audio signals of "Marry had a little lamb" were loaded into the work space. The first audio signal was the piano version, and the second audio signal was the recorder version. The time domains and the audio signals were extracted from both signals, and the frequency domains were created. Using the same algorithm as part one, the Gábor transform of the two audio signals were generated. Furthermore, the center frequencies are determined by identifying the frequency with the maximum amplitude at each time discrete. In addition, a Gaussian filter was added around the center frequency of the signal at each time discrete. As a result, this creates a clear music score without overtones.

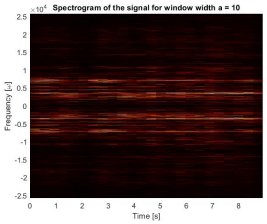
IV. Computational Results

IV.A. Part One

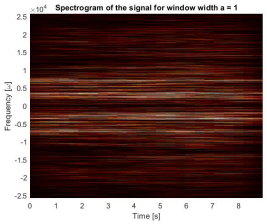
Figure 1a showed the three different Gábor window scaling, $a = 0.1, 1, 10$, utilized for the Gábor transform of the signal. Figure 1b, 1c and 1d showed the spectrogram of the Gábor transform of the signal with three different window width shown in Figure 1a. It is apparent that the window width trades off time and frequency resolution at the sacrifice of each other. Figure 1d has the broadest Gábor window width, therefore, the frequency resolution is excellent and fails to localize time information. Furthermore, Figure 1b has the narrowest Gábor window width, therefore, the localization in the time domain is excellent, but the high frequency information is suppressed.



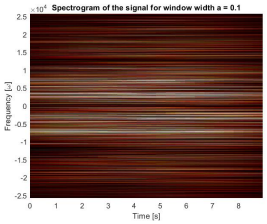
(a) Gábor window width $a = 10, 1.0, 0.1$



(b) The Gábor transform with $a = 10$.



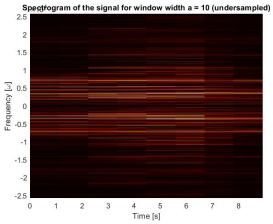
(c) The Gábor transform with $a = 1.0$.



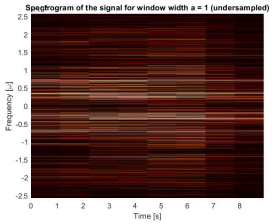
(d) The Gábor transform with $a = 0.1$.

Fig. 1: The spectrogram of the Gábor transform of the signal with three different window widths.

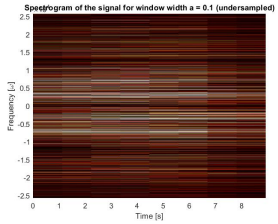
Figure 2 showed spectrogram of the under-sampled Gábor transform of the signal with the three window width as shown in Figure 1a. It is apparent that the under-sampled Gábor transform has a lower resolution at all three different window width as compared to the oversampled Gábor transforms shown in Figure 1.



(a) The under-sampled Gábor transform with $a = 10$.



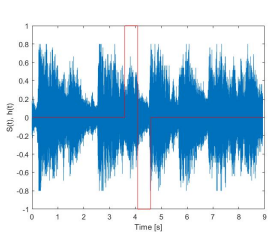
(b) The under-sampled Gábor transform with $a = 1.0$.



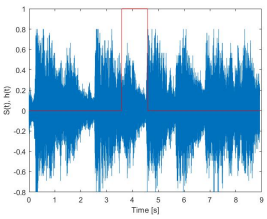
(c) The under-sampled Gábor transform with $a = 0.1$.

Fig. 2: The spectrogram of the under-sampled Gábor transform of the signal with three different window widths.

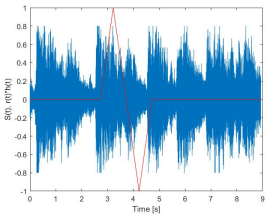
Figure 3 showed the four other different wavelets utilized for the Gábor transform of the signal. The spectrogram of the Gábor transform of the signal with these four wavelets are shown in Figure 3. Figure 3g and 3h has the lowest resolution in the time domain because the triangle and Mexican hat wavelet has the broadest Gábor window as compared to the other two wavelets as shown in Figure 3.



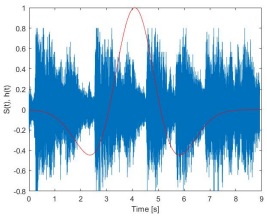
(a) The Haar wavelet.



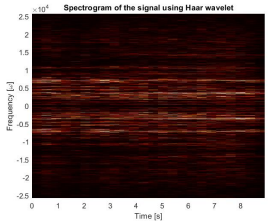
(b) The square wavelet.



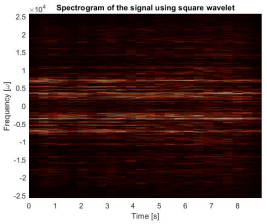
(c) The triangle wavelet.



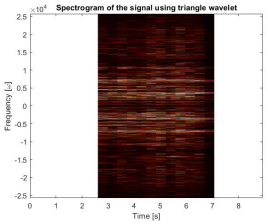
(d) The Mexican hat wavelet.



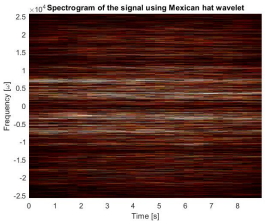
(e) The Gábor transform with Haar wavelet.



(f) The Gábor transform with square wavelet.



(g) The Gábor transform with triangle wavelet.

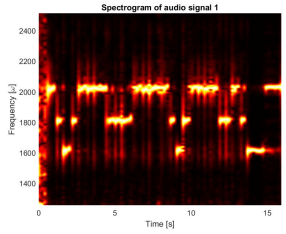


(h) The Gábor transform with Mexican hat wavelet.

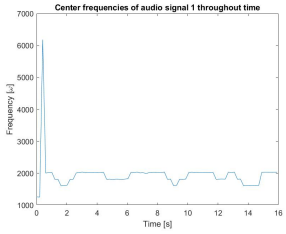
Fig. 3: The four different wavelet utilized as the Gábor window for the Gábor transform of the signal.

IV.B. Part Two

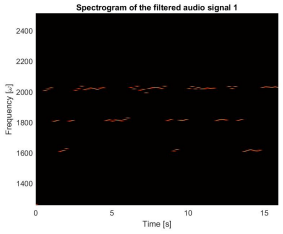
Figure 4 showed the Gábor transform analysis of the piano signal. The center frequencies identified over the time domain are 325Hz, 290Hz and 260Hz, which corresponds to the notes "E", "D" and "C".



(a) The Gábor transform.



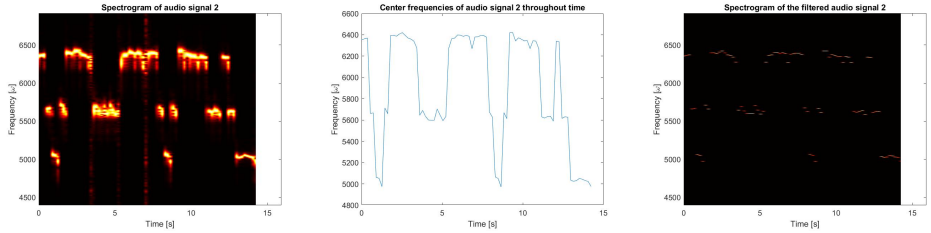
(b) The center frequencies.



(c) The filtered Gábor transform.

Fig. 4: The Gábor transform of the piano signal.

Figure 5 showed the Gábor transform analysis of the piano signal. The center frequencies identified over the time domain are 1050Hz, 930Hz and 830Hz, which corresponds to the notes “C”, “A-Sharp” and “G-Sharp”.



(a) The Gábor transform. **(b)** The center frequencies. **(c)** The filtered Gábor transform.

Fig. 5: The Gábor transform of the recorder signal.

It is apparent that the notes played by the recorder has a higher frequency as compared to the notes played by the piano. Furthermore, there are differences in their overtones which is related to the timbre of the instrument being played, therefore, produces distinctive sounds.

V. Summary and Conclusions

The Gábor transform is an powerful tool for analyzing non-stationary signals. The most valuable feature of Gábor transform is that it is able to extract both time and frequency information. However, the resolution in time is traded off by the resolution in frequency. This demonstrates the Heisenberg relationship. The relationship between time and frequency resolution is discovered through varying the scaling and translation of the Gábor window in the project. Furthermore, analysis of music piece and overtones were discovered through Gábor analysis and filtering.

Appendix A

$linspace(x1, x2, n)$: Creates a vector of n points evenly spaced between $x1$ and $x2$.
$fftshift(x)$: Rearranges x by shifting the zero-frequency component to the center.
$meshgrid(x, y, z)$: Creates a three dimensional grid with the coordinates of x, y and z.
$reshape(x, n, n, n)$: Rearranges x into a n-by-n-by-n matrix.
$cell(n, n)$: Creates an empty n-by-n cell array.
$zeros(n, n)$: Creates an n-by-n matrix with entries of zero.
$fftn(x)$: Computes the multidimensional fast Fourier transform of x.
$max(x)$: Finds the maximum value in x and the index of the value.
$abs(x)$: Computes the complex magnitude of each element in x.
$ind2sub(sz, I)$: Computes the subscript value that corresponds to the linear indices I for a matrix of size sz.
$isosurface(X, Y, Z, V, ISOV)$: Computes the isosurface plot with the three dimensional grid X, Y and Z from the volume data V at the isosurface value specified $ISOV$.
$ifftshift(x)$: Rearranges x by the inverse of $fftshift$.
$ifftn(x)$: Computes the inverse multidimensional fast Fourier transform of x.
$plot3(x, y, z)$: Computes plot of x, y and z coordinate in a three dimensional space.

Appendix B

```
clc; clear all; close all;

load ('handel');

%% Plot the original signal
S = y';
t = (1:length(S))/Fs;

% figure(1)
% plot(t, S);
% xlabel('Time [sec]');
% ylabel('Amplitude');
% title('Signal of Interest (Original)');

% p8 = audioplayer(v, Fs);
% playblocking(p8);

%% Setup the time and frequeuncy domain
n = length(t) - 1;
t = t(1 : n);
S = S(1 : n);
L = t(n);
k = 2*pi/L*[0:(n/2 - 1), (-n/2):-1];
ks = fftshift(k);

%% FFT the signal
St = fft(S);

%% Plot the signal in frequency and time domain
figure(2)
plot(t, S);
xlabel('Time [s]');
ylabel('S(t)');
title('Signal of Interest (Time Domain)');

figure(3)
plot(ks, fftshift(abs(St)));
xlabel('Frequency [\omega]');
ylabel('FFT(S(t))');
title('Signal of Interest (Frequency Domain)');

%% Compute spectrograms using Gaussian window with
different widths
a = [10, 1, 0.1];
```

```
tslide = linspace(0, t(n), 25); % t0 = 0.3719
Sgt_spec = cell(1, length(a));


for i = 1 : length(a)
    for j = 1 : length(tslide)
        g = exp(-a(i)*(t - tslide(j)).^2);
        Sg = g.*S;
        Sgt = fft(Sg);
        Sgt_spec{i} = [Sgt_spec{i}; fftshift(abs(Sgt))];
        figure(4)
        if j == 12
            subplot(3, 1, i);
            plot(t, S);
            hold on;
            plot(t, g, 'r');
            xlabel('Time [s]');
            ylabel('S(t), g(t)');
            title(['Window width a = ' num2str(a(i))]);
        end
    end
    figure(i + 4)
    pcolor(tslide, ks, Sgt_spec{i}.');
    xlabel('Time [s]');
    ylabel('Frequency [\omega]');
    title(['Spectrogram of the signal for window width a =
' num2str(a(i))]);
    shading interp;
    colormap(hot);
end



%% Compute spectrograms using Gaussian window with
undersampling size
tslideUS = linspace(0, t(n), 5);    % t0 = 2.2312
SgUSt_spec = cell(1, length(a));

for i = 1 : length(a)
    for j = 1 : length(tslideUS)
        gUS = exp(-a(i)*(t - tslideUS(j)).^2);
        SgUS = gUS.*S;
        SgUSt = fft(SgUS);
```

```
        SgUSt_spec{i} = [SgUSt_spec{i};
fftshift(abs(SgUSt))];
    end
    figure(i + 7)
    pcolor(tslideUS, ks, SgUSt_spec{i}.');
    xlabel('Time [s]');
    ylabel('Frequency [\omega]');
    title(['Spectrogram of the signal for window width a =
' num2str(a(i)) ' (undersampled)']);
    shading interp;
    colormap(hot);
end

%% Compute spectrogram using square, Haar & ramp wavelet
Sst_spec = [];
Sht_spec = [];
Srt_spec = [];

for j = 1 : length(tslide)
    s = zeros(1, length(t));
    h = zeros(1, length(t));

    for i = 1 : length(t)
        tau = tslide(j);
        low = tau - a(2)/2;
        high = tau + a(2)/2;
        if t(i) < low || t(i) >= high
            s(i) = 0;
        else
            s(i) = 1;
        end

        if t(i) >= low && t(i) < tau
            h(i) = 1;
        elseif t(i) >= tau && t(i) < high
            h(i) = -1;
        else
            h(i) = 0;
        end
    end

    r = conv(s, h, 'same');
    r = r/max(r);
```

```
    if j == 12
        figure(11)
        plot(t, S);
        hold on;
        plot(t, s, 'r');
        xlabel('Time [s]');
        ylabel('S(t), h(t)');

        figure(12)
        plot(t, S);
        hold on;
        plot(t, h, 'r');
        xlabel('Time [s]');
        ylabel('S(t), h(t)');

        figure(13)
        plot(t, S);
        hold on;
        plot(t, r, 'r');
        xlabel('Time [s]');
        ylabel('S(t), r(t)*h(t)');

    end

    Ss = s.*S;
    Sst = fft(Ss);
    Sst_spec = [Sst_spec; fftshift(abs(Sst))];

    Sh = h.*S;
    Sht = fft(Sh);
    Sht_spec = [Sht_spec; fftshift(abs(Sht))];

    Sr = r.*S;
    Srt = fft(Sr);
    Srt_spec = [Srt_spec; fftshift(abs(Srt))];
end

figure(14)
pcolor(tslide, ks, Sst_spec.');
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of the signal using square wavelet');
shading interp;
colormap(hot);
```

```
figure(15)
pcolor(tslide, ks, Sht_spec.');
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of the signal using Haar wavelet');
shading interp;
colormap(hot);

figure(16)
pcolor(tslide, ks, Srt_spec.');
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of the signal using triangle wavelet');
shading interp;
colormap(hot);

%% Compute spectrogram using Mexican hat wavelet
Smt_spec = [];

for j = 1 : length(tslide)
    tau = tslide(j);
    m = (1 - (t - tau).^2).*exp(-(t - tau).^2./2);

    figure(17)
    if j == 12
        plot(t, S);
        hold on;
        plot(t, m, 'r');
        xlabel('Time [s]');
        ylabel('S(t), h(t)');
    end

    Sm = m.*S;
    Smt = fft(Sm);
    Smt_spec = [Smt_spec; fftshift(abs(Smt))];
end

figure(18)
pcolor(tslide, ks, Smt_spec.');
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of the signal using Mexican hat
wavelet');
```

```
shading interp;
colormap(hot);
```

```
clc; clear all; close all;

%% Plot the original signal
[y1, Fs1] = audioread('music1.wav');
S1 = y1';
t1 = (1:length(S1))/Fs1;
% % figure(1)
% % plot(t1, S1);
% % xlabel('Time [s]');
% % ylabel('Amplitude');
% % title('Mary had a little lamb (piano)');
% p81 = audioplayer(S1, Fs1);
% playblocking(p81);

[y2, Fs2] = audioread('music2.wav');
S2 = y2';
t2 = (1:length(S2))/Fs2;
% % figure(2)
% % plot(t2, S2);
% % xlabel('Time [sec]');
% % ylabel('Amplitude');
% % title('Mary had a little lamb (recorder)');
% p82 = audioplayer(S2, Fs2);
% playblocking(p82);


%% Setup the time and frequency domain
n1 = length(t1);
L1 = t1(n1);
k1 = (2*pi/L1)*[0:n1/2-1 -n1/2:-1];
ks1 = fftshift(k1);

n2 = length(t2);
L2 = t2(n2);
k2 = (2*pi/L2)*[0:n2/2-1 -n2/2:-1];
ks2 = fftshift(k2);

%% FFT the signals
S1t = fft(S1);

S2t = fft(S2);

%% AUDIO 1 Compute spectrograms using Gaussian window
tslide1 = linspace(0, t1(n1), 80);
```

```
S1gt_spec = zeros(80, n1);
a1 = 80;
k01 = zeros(1, 80);
tau = 0.5;
S1gtf_spec = zeros(80, n1);
for i = 1 : length(tslide1)
    g1 = exp(-a1*(t1 - tslide1(i)).^2);
    S1g = g1.*S1;
    S1gt = fft(S1g);
    [M, I] = max(fftshift(abs(S1gt)));
    k01(i) = abs(ks1(I));
    gt1 = exp(-tau*(ifftshift(ks1) - k01(i)).^2);
    S1gtf = gt1.*S1gt;
    S1gtf_spec(i, :) =
fftshift(abs(S1gtf))/max(abs(S1gtf));
    S1gt_spec(i, :) = fftshift(abs(S1gt))/max(abs(S1gt));
    if i == 40
        figure(3)
        plot(t1, S1);
        hold on;
        plot(t1, g1);
        xlabel('Time [s]');
        ylabel('S1(t), g1(t)');
        title('Audio signal 1 and Gaussian window');
        saveas(gcf, 'P2_1.jpg');
    end
end
figure(4)
plot(tslide1, k01);
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Center frequencies of audio signal 1 throughout
time');
saveas(gcf, 'P2_1_CF.jpg');

figure(5)
pcolor(tslide1, ks1, S1gt_spec.');
shading interp;
colormap(hot);
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of audio signal 1');
axis([0 L1 200*2*pi 400*2*pi]);
saveas(gcf, 'P2_1_Spec.jpg');
```

```
figure(6)
pcolor(tslide1, ks1, S1gtf_spec.');
shading interp;
colormap(hot);
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of the filtered audio signal 1');
axis([0 L1 200*2*pi 400*2*pi]);
saveas(gcf, 'P2_1_Spec_Filter.jpg');

%% AUDIO 2 Compute spectrograms using Gaussian window
tslide2 = linspace(0, t2(n2), 80);
S2gt_spec = [];
a2 = 70;
k02 = zeros(1, 80);
tau = 0.2;
S2gtf_spec = zeros(80, n2);
for i = 1 : length(tslide2)
    g2 = exp(-a2*(t2 - tslide2(i)).^2);
    S2g = g2.*S2;
    S2gt = fft(S2g);
    [M, I] = max(fftshift(abs(S2gt)));
    k02(i) = abs(ks2(I));
    gt2 = exp(-tau*(ifftshift(ks2) - k02(i)).^2);
    S2gtf = gt2.*S2gt;
    S2gtf_spec(i, :) =
fftshift(abs(S2gtf))/max(abs(S2gtf));
    S2gt_spec(i, :) = fftshift(abs(S2gt))/max(abs(S2gt));
    if i == 40
        figure(3)
        plot(t2, S2);
        hold on;
        plot(t2, g2);
        xlabel('Time [s]');
        ylabel('S1(t), g1(t)');
        title('Audio signal 2 and Gaussian window');
        saveas(gcf, 'P2_2.jpg');
    end
end
figure(4)
plot(tslide2, k02);
xlabel('Time [s]');
ylabel('Frequency [\omega]');
```

```
title('Center frequencies of audio signal 2 throughout
time');
saveas(gcf, 'P2_2_CF.jpg');

figure(5)
pcolor(tslide2, ks2, S2gt_spec.');
shading interp;
colormap(hot);
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of audio signal 2');
axis([0 L1 700*2*pi 1100*2*pi]);
saveas(gcf, 'P2_2_Spec.jpg');

figure(6)
pcolor(tslide2, ks2, S2gtf_spec.');
shading interp;
colormap(hot);
xlabel('Time [s]');
ylabel('Frequency [\omega]');
title('Spectrogram of the filtered audio signal 2');
axis([0 L1 700*2*pi 1100*2*pi]);
saveas(gcf, 'P2_2_Spec_Filter.jpg');
```