Mini Project #1: Audio↔Image Signal Synthesis using a Time-Frequency Representation

Mr. Dan Jacobellis and Prof. Brian Evans

Assigned on Sunday, September 14, 2025
Due on Friday, September 19, 2025, by 11:59 pm via Gradescope submission

Late submission is subject to a penalty of two points per minute late.

Reading: McClellan, Schafer and Yoder, *Signal Processing First*, 2003, Ch. 3. Errata.
Companion Web site with demos and other supplemental information: http://dspfirst.gatech.edu/
Web site contains solutions to selected homework problems from *DSP First*.

E-mail Mr. Dan Jacobellis (TA) at danjacobellis@utexas.edu. Please consider posting questions on Ed Discussion, which can be answered by anyone in the class. You can post anonymously.

Lecture and office hours for Mr. Jacobellis and Prof. Evans follow. Prof. Evans also holds office hours in person in EER 6.882 and online on Zoom.

Time Slot	*Monday*	*Tuesday*	*Wednesday*	*Thursday*	*Friday*
11:00 am		**Evans (EER 1.516)**		**Evans (EER 1.516)**	
11:30 am		**Evans (EER 1.516)**		**Evans (EER 1.516)**	
12:00 pm		**Evans (EER 1.516)**		**Evans (EER 1.516)**	
12:30 pm		**Jacobellis (EER 1.810)**			
1:00 pm		**Jacobellis (EER 1.810)**			
1:30 pm		**Jacobellis (EER 1.810)**			
2:00 pm	**Evans (EER 6.882)**	**Jacobellis (EER 1.810)**			
2:30 pm	**Evans (EER 6.882)**				
3:00 pm	**Evans (EER 6.882)**			**Jacobellis (Zoom)**	
3:30 pm			**Evans (EER 6.882)**	**Jacobellis (Zoom)**	
4:00 pm			**Evans (EER 6.882)**	**Jacobellis (Zoom)**	
4:30 pm			**Evans (EER 6.882)**	**Jacobellis (Zoom)**	
5:00 pm				**Jacobellis (Zoom)**	**Jacobellis (EER 1.810)**
5:30 pm					**Jacobellis (EER 1.810)**
6:00 pm					**Jacobellis (EER 1.810)**

1.0 Introduction

When students described what they had hoped to get out of our class, several mentioned

- Audio signal processing
- Machine learning
- Image processing

This project combines elements of the above.

For an introduction to image processing, please watch this sequence of videos by Prof. Shree Naya at Columbia University on YouTube:

- "Overview". 3:40.
- "Pixel Processing". 2:46.

Please work in a group of three and create one report together. Each person would submit the same report on Gradescope. Be sure the report represents the independent work of the authors. See Appendix B.

In this project, you will design and implement a system, $\mathcal{T}$, and its inverse, $\mathcal{T}^{-1}$, that converts a discrete-time audio signal $A[n]$ to a discrete space image signal $I[m,k]$ and vice versa using a time-frequency transformation.

$$\underbrace{I[m,k]}_{\substack{\text{visual}\\ \text{representation}}} = \mathcal{T}\,\underbrace{\{A[n]\}}_{\substack{\text{audio}\\ \text{signal}}}$$

$\mathcal{T}^{-1}$ is the approximate inverse. [1]

$$\mathcal{T}^{-1}\{I[m,k]\} = \mathcal{T}^{-1}\{\mathcal{T}\{A[n]\}\} \approx A[n]$$

$\mathcal{T}$ consists of three components:

$$\mathcal{T}\{A[n]\} = \underbrace{\mathcal{C}}_{\substack{\text{color}\\ \text{mapping}}} \circ \underbrace{F}_{\text{companding}} \circ \underbrace{\mathbf{STFT}\{A[n]\}}_{\substack{\text{time-frequency}\\ \text{transformation}}}$$

1. The discrete short-time Fourier transform (STFT), which is a variant of the Fourier series

$$S[m,k] = \mathbf{STFT}\{x[n]\}[m,k] = \sum_{n=0}^{N-1} x[n]w[n-m]e^{-j2\pi\frac{k}{N}n}$$

   Unlike the standard Fourier series, the STFT includes a window function $w[n-m]$ that isolates the part of the signal centered around m. For this project, we will use a rectangular window with $N = 448$ samples.

$$w[n] = \text{rect}_{N=448}[n] = \begin{cases} 1 & 0 \le n < 448 \\ 0 & \text{otherwise} \end{cases}$$

2. A companding operation F that reduces the STFT range so that it can be displayed as an image:

$$F(x) = \text{sign}(x)\left((|x| + \epsilon)^p - \epsilon^p\right)$$

[1] Finite precision and dynamic range of the image signal will result in some information loss, so the recovered audio signal will not be exactly the same as the original signal.

3. A color mapping $\mathcal{C}$ that allows a complex-valued number to be represented by a color.

$$(\text{red intensity}, \text{green intensity}, \text{blue intensity}) = \mathcal{C}(\text{Re}\{x\} + j\ \text{Im}\{x\})$$

2.0 Loading an audio signal

1. Using MATLAB, load an audio signal whose length is at least 24 seconds. [2]
2. Resample the audio signal to 16,000 Hz. [3]
3. If the audio file is stereo (two channel) convert it to mono (single channel) by averaging the two channels together.
4. Truncate the signal to exactly 448,000 samples (28 seconds).

2.1 Computing the STFT and inverse STFT

1. Divide the audio signal into 1000 windows of 448 samples each. [4]
2. Use MATLAB's `fft` function to compute the frequency spectrum of each window. The result of this step is a complex-valued matrix $S[m,k]$. [5]
3. Verify that you can recover the audio signal from $S[m,k]$.
   a. Perform the Fourier synthesis of each column using MATLAB's `ifft` function.
   b. Reshape the matrix back into a vector, then play the audio signal using `soundsc`.
4. Convert the complex-valued $S[m,k]$ into two real matrices using the following two methods. [6]
   a. Magnitude $S_{\text{mag}} = |S[m,k]|$, and phase $S_{\text{phase}} = \angle S[m,k]$.
   b. Real $S_{\text{real}} = \text{Re}\{S[m,k]\}$ and imaginary $S_{\text{imag}} = \text{Im}\{S[m,k]\}$.

2.3 Companding

1. Plot the histograms of S_{mag}, S_{phase}, S_{real}, and S_{imag}. [7]
   a. Describe the shape of the distribution.
   b. What is the range of each?
2. Apply the companding function F with $\epsilon = 0.1$ and $p = 0.4$.[8]

[2] The `audioread` function can be used to load an audio signal.

[3] The `resample` function from the MATLAB Signal Processing Toolbox can be used to change the sampling rate. The `interp1` function can also be used to resample an audio signal without the toolbox.

[4] Use the `reshape` function to arrange $A[n]$ into a 448×1000 matrix (1000 windows of 448 samples each).

[5] The `fft` function performs a discrete-time version of Fourier series analysis—see Appendix A. It returns complex values for both positive and negative frequencies. When the `fft` function is applied to a matrix, it will perform the analysis independently to each column. `fft` returns the positive frequencies first, followed by the negative frequencies. The `fftshift` function sorts the frequencies in ascending order.

[6] In MATLAB, the functions to compute the magnitude, phase, real, and imaginary parts are `abs`, `angle`, `real`, and `imag`, respectively

[7] Use the histogram function, which can be directly applied to a vector or matrix.

[8] MATLAB code for the companding function F appears below and its inverse appears in the footnote at the bottom of the next page:

```
function y = F(x)
    eps=0.1; power=0.4;
    y = sign(x) .* ((abs(x) + eps).^power - eps.^power);
end
```

   a. Plot the histogram of $F(S_{\text{mag}})$. What is its range?
   b. Plot the histogram of $F(S_{\text{Real}})$. What is its range?

2.4 Mapping complex values to a color image

1. Use the `imagesc` function to visualize the components of S using an automatic color mapping. [9]
   a. Display $F(S_{\text{mag}})$ using imagesc. Describe any connections you see between this visualization and the sound of the original audio signal. [10]
   b. Display S_{phase}, $F(S_{\text{Real}})$, and $F(S_{\text{Imag}})$ using `imagesc`.
2. Design a mapping $\mathcal{C}$ that allows the complex matrix S to be represented by a color image. [11]

$$(\mathrm{R}, \mathrm{G}, \mathrm{B}) = (\text{red intensity}, \text{green intensity}, \text{blue intensity}) = \mathcal{C}(\mathrm{Re}\{S\} + j\ \mathrm{Im}\{S\})$$

   Consider using either:

   a. A mapping from $F(S_{\text{mag}})$ and S_{phase} to (R, G, B)

```
function RGB = MagPhase2RGB(S)
  FSmag = F(abs(S))
  Sphase = angle(S)
  R = % TODO: map FSmag, Sphase to a red intensity between 0 and 1
  G = % TODO: map FSmag, Sphase to a green intensity between 0 and 1
  B = % TODO: map FSmag, Sphase to a blue intensity between 0 and 1
  RGB = uint8(255*cat(3, R, G, B)); % represent as an RGB image
end
```

   b. A mapping from $F(S_{\text{real}})$ and $F(S_{\text{Imag}})$ to (R, G, B)

```
function RGB = RealImag2RGB(S)
  FSreal = F(real(S))
  FSimag= F(imag(S))
  R = % TODO: map FSreal, FSimag to a red intensity between 0 and 1
  G = % TODO: map FSreal, FSimag to a green intensity between 0 and 1
  B = % TODO: map FSreal, FSimag to a blue intensity between 0 and 1
  RGB = uint8(255*cat(3, R, G, B)); % represent as an RGB image
end
```

3. Save the image to a file using the `imwrite` function.

3.0 Recovering audio from the time-frequency image.

1. Load the image file that you created in step 2 using `imread`.
2. Apply $\mathcal{C}^{-1}$, the inverse of your color mapping, to recover S from the image.

```
function x = Finv(y)
    eps=0.1; power=0.4;
    x = sign(y) .* ((abs(y) + eps.^power).^(1./power) - eps);
end
```

[9] After calling `imagesc`, the `colorbar` function can be used to add a legend.

[10] A visual representation of the magnitude of the STFT is also called the magnitude spectrogram.

[11] When displaying an digital image in MATLAB the (R,G,B) triplet is expected to fall in the range [0,1].

a. If you used $F(S_{\mathrm{mag}})$ and S_{phase}:

```
function S = RGB2Complex(RGB)
  R = double(RGB(:,:,1));
  G = double(RGB(:,:,2));
  B = double(RGB(:,:,3));
  FSmag = % TODO: map (R,G,B) to FSmag
  Sphase = % TODO: map (R,G,B) to Sphase
  S = Finv(FSmag) .* exp(j*Sphase)
end
```

b. If you used $F(S_{\mathrm{real}})$ and $F(S_{\mathrm{Imag}})$

```
function S = RGB2Complex(RGB)
  R = double(RGB(:,:,1));
  G = double(RGB(:,:,2));
  B = double(RGB(:,:,3));
  FSreal = % TODO: map (R,G,B) back to FSreal
  FSimag = % TODO: map (R,G,B) back to FSimag
  S = Finv(FSreal) + j* Finv(FSimag)
end
```

3. Recover the audio signal and play it back using the same procedure as step 2.1.3.
4. Compute the peak signal to noise ratio (PSNR) of the reconstruction. [12]
5. Describe any differences you hear between the original audio signal and the recovered audio signal.

[12] For a pair of signals in the range [-1,1] with length L, the PSNR is:

$$\begin{aligned}\mathrm{PSNR}(x[n],y[n]) &= 20\log_{10}\left(\frac{(I_{\max}-I_{\min})^2}{\mathrm{MSE}(x[n],y[n])}\right)\\ &= 20\log_{10}2 - 10\log_{10}\big(\mathrm{MSE}(x[n],y[n])\big)\\ &= -10\log_{10}\big(\mathrm{MSE}(x[n],y[n])\big) + 6.02\ \mathrm{dB}\end{aligned}$$

where

$$\mathrm{MSE}(x[n],y[n]) = \frac{1}{L}\sum_{0}^{L}(x[n]-y[n])^2$$

In MATLAB:

```
PSNR = -10*log10(mean(abs(x - y).^2)) + 6.02
```

Appendix A: **Connections Between Continuous-Time and Discrete-Time Fourier Series**

Continuous-Time Fourier Series. A continuous-time periodic signal $x(t)$ with period T_0 sec is composed of a constant term plus frequency components at integer multiples (harmonic) of a fundamental frequency f_0 where $f_0 = 1 \;/\; T_0$. Fourier series analysis computes the constant term a_0 plus the magnitude and phase of each frequency term a_k where k is any integer:

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{j2\pi k f_0 t} \quad \text{where} \quad a_0 = \frac{1}{T_0}\int_0^{T_0} x(t)\, dt \quad \text{and} \quad a_k = \frac{1}{T_0}\int_0^{T_0} x(t) e^{-j2\pi k f_0 t} dt$$

An *infinite* number of coefficients could be needed to exactly represent $x(t)$.

Discrete-Time Fourier Series. A discrete-time periodic signal $x[n]$ with period N samples is composed of a constant term plus frequency components at integer multiples (harmonic) of a fundamental frequency of $2\pi/N$ rad/sample which is equivalent to $f_s \,/\, N$ in Hz. Fourier series analysis computes the constant term $X[0]$ plus the magnitude and phase of each frequency term $X[k]$ for $k = 1, \ldots, N\text{-}1$.

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, e^{j\left(k\frac{2\pi}{N}\right)n} \quad \text{where} \quad X[0] = \sum_{n=0}^{N-1} x[n] \quad \text{and} \quad X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\left(k\frac{2\pi}{N}\right)n}$$

A *finite* number of Fourier series coefficients would always be needed to exactly represent $x[n]$.

In the Fourier series coefficients $X[k]$, index k corresponds to continuous-time frequency $f_k = k\frac{f_s}{N}$ in Hz for $k = 0, 1, \ldots, \frac{N}{2} - 1$ and $f_k = f_s\left(\frac{k}{N} - 1\right)$ for $k = \frac{N}{2}, \frac{N}{2} + 1, \ldots, N - 1$, assuming N is even.

Normalization. The scaling of the Fourier series coefficients is different in continuous-time vs. discrete-time. When using the discrete-time Fourier series to compute continuous-time Fourier series coefficients, we would have to divide the discrete-time Fourier series coefficient by N due the $(1/N)$ term in the equation for $x[n]$.

Fast Fourier Transform (FFT) is a fast algorithm to compute the N discrete-time Fourier series coefficients $X[k]$ from the N samples of a discrete-time signal $x[n]$. As with the continuous-time Fourier series, the discrete-time Fourier series assumes that the N samples of $x[n]$ represents the fundamental period of an infinitely long signal in the time domain. Unlike the continuous-time Fourier series, the discrete-time Fourier series always has a finite number of terms, N. One of the reasons for this is due to the Sampling Theorem, which says that the sampling rate $f_s > 2\, f_{\max}$ where $f_{\max}$ is the highest frequency of interest and hence $f_{\max} < ½\, f_s$. Sampling only captures continuous-time frequencies $(-½\, f_s, ½\, f_s)$ whereas continuous-time signals have frequencies. You can think of the discrete-time Fourier series as computing the Fourier series coefficients for frequency components from $-½\, f_s$ to $-½\, f_s$, which is a finite number because $f_s > 0$.

The Fast Fourier Transform (FFT) is requires $2M \log_2 M$ real-valued multiplications and additions and $4M$ words of memory instead of the $4\, M^2$ and $M^2 + 4M$, respectively, for the direct matrix-vector implementation of the discrete-time Fourier series. The direct matrix-vector implementation to compute $X[k]$ would create a complex-valued N x N matrix of the term $\exp(-j\;(2\pi/N)\;kn)$ for $k = 0, 1, \ldots, N\text{-}1$ in one dimension and $n = 0, 1, \ldots, N\text{-}1$ in the other, form a column vector of $x[0], x[1], \ldots, x[N\text{-}1]$, and multiply the matrix and vector to find the column vector of $X[0], X[1], \ldots, X[N\text{-}1]$.

Appendix B: Homework and Mini-Project Guidelines

Here are some things you should follow for all assignments.

Amount of work to show:

1. An explanation should be given for every single answer. Answers written without explanation will lose two-thirds of the points allotted for that part.
2. Only "standard" formulas (like Euler's formula, trigonometric formulas, etc.) can be used without a reference. If you're using something non-standard, then please put a reference to the formula number in the book, or whatever source you got it from. Just using the final result of a similar problem done in the class, and omitting the intermediate steps, is not okay. You have to show your work.
3. There shouldn't be big jumps in logic from one step to the next.
4. For everything, expect to show at least one intermediate step between the first line and the answer. Even if it seems unnecessary to you, please err on the side of caution. Things that seem obvious to you when you're writing the solution are not quite so obvious for someone reading it.
5. If you're in any doubt about how much work to show, please ask the instructor or the teaching assistant.

MATLAB source code guidelines:

1. Put a comment before the solution of each part, telling the question number of the solution.
2. If you're using complicated logic, leave a comment telling what that block of code is supposed to do.
3. Use variable names that related to their meaning/use.
4. Avoid using two different variables for the same thing.
5. Try to avoid using "magic numbers" in the code. If you're using a number, write a comment telling me how you derived it.
6. Make sure that your code will compile & run in a clean workspace; i.e., one without any variables present. Use a clear all; at least once before submitting it.
7. No marks will be deducted based on the efficiency of the code unless the problem asks you to write efficient code.

Technical points:

1. Merge all the files together into one PDF file.
2. Please adjust the contrast, exposure etc., to get a good scan quality so that the TA can easily read what you write. Take extra care to get a good scan for parts written in pencil.
3. For the MATLAB code you write for an assignment, please copy the code into Word or include a screenshot showing the code. Do not submit handwritten code.

Other things:

1. All plots must have axis labels, with units.
2. Final answers must be boxed, or underlined or otherwise differentiated from the rest of the solution.

3. All final answers must have units, if they exist.
4. Read the questions carefully.
5. Try to answer all parts of a question together. If the solution to some parts of a question is written elsewhere, then leave a note telling the reader where to find it.

Organization of a mini-project report:

Please write a self-contained narrative report. The audience is someone who has taken the equivalent of this class. The report should provide references to the textbook and other sources as needed. Please refer to the hints above, which apply to homework assignments and mini-project reports, as well as the following additional guidelines for the mini-project.

Here are example mini-project #1 reports written by the instructors:

- "FM Synthesis for Musical Instruments" (2018)
- "Sinusoidal Speech Synthesis" (2021)
- "Music Synthesis" (2023)

Please see the homework hints page for specific guidelines for this project.