Welcome back to **CME 292**

Advanced MATLAB for Scientific Computing

WINTER 2023

Stanford University

Advanced Tools for Images and Signals

CME 292 Lecture 7

1/31/2023

Stanford University

Outline

Image Processing Toolbox

Computer Vision Toolbox

Image Acquisition Toolbox

Signal Processing Toolbox

Audio Toolbox & DSP System Toolbox

Image Processing

Advanced Tools for Images and Signals

Image from MathWorks

Image Processing Toolbox

Image Processing Toolbox provides tools and algorithms for:

- image segmentation
- image enhancement
- noise reduction
- geometric transformations
- image registration
- 3D image processing
- ... ...

Install: Add-Ons → Get Add-Ons → search for the toolbox → install

Basic image operations

- Import, display, and resize images
- Convert color images to grayscale
  - color image: RGB
  - grayscale image: intensity values
- Rotate and compare images

More on Intensity

Intensity Profile

- The set of intensity values taken from regularly spaced points along a line segment or multi-line path in an image.

```
imshow(I,[]);
improfile
```

- The line segment (or segments) can be defined by specifying their coordinates as input arguments or interactively using a mouse.

Intensity Histograms

- It separates pixels into bins based on their intensity values, e.g.
  dark images have many pixels binned in the low end of the histogram.

```
gs = rgb2gray(I);
imhist(gs)
```

- Useful for normalizing the brightness

Techniques for **grayscale image contrast enhancement**

- `imadjust` increases the contrast of the image by mapping the values of the input intensity image to new values such that, by default, 1% of the data is saturated at low and high intensities of the input data.
- `histeq` performs histogram equalization.
  - It enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram.
- `adapthisteq` performs contrast-limited adaptive histogram equalization.
  - Unlike `histeq`, it operates on small data regions (tiles) rather than the entire image. Each tile's contrast is enhanced so that the histogram of each output region approximately matches the specified histogram.
  - The contrast enhancement can be limited in order to avoid amplifying the noise which might be present in the image.

Binary Image

Threshold the intensity values of a grayscale image with automated threshold selection process to binarize an image.

`imbinarize(I, METHOD)`

The method calculates the "best" threshold for the image

- Global
- Local

Filter Noise

Images taken in low light often become noisy due to the increase in camera sensitivity required to capture the image.

To reduce the impact of this noise on the binary image, preprocess the image with some filter.

```
B = imfilter(A,H)
```

filters the multidimensional array A with the multidimensional filter H.

```
H = fspecial(TYPE)
```

creates a two-dimensional filter H of the specified type, e.g., average filter, motion filter.

Morphological Operations

Morphology is a broad set of image processing operations that process images based on shapes.

Morphological operations apply a structuring element to an input image, creating an output image of the same size.

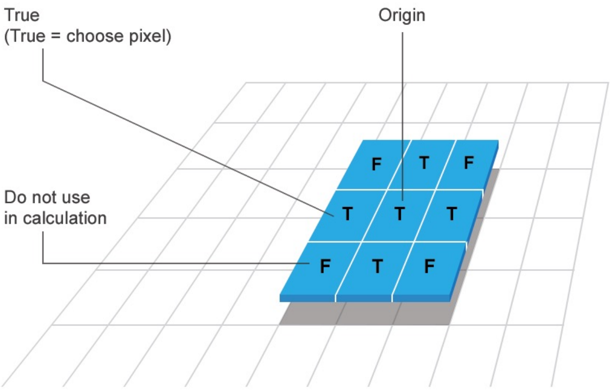
In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors.

Structuring Element

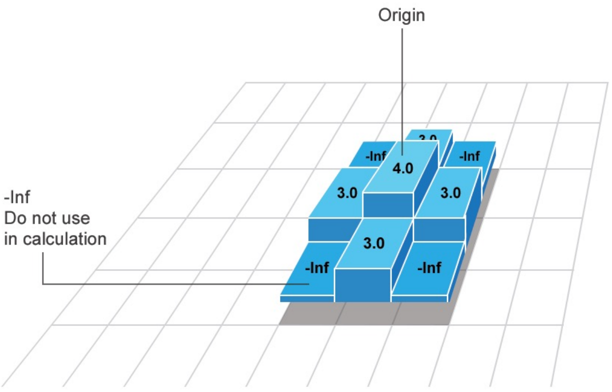
- A matrix that identifies the pixel in the image being processed and defines the neighborhood used in the processing of each pixel.
- The center pixel of the matrix, called the *origin*, identifies the pixel in the image that is being processed

Two types of structuring elements:

- *Flat*: created by `strel`
- *Nonflat*: created by `offsetstrel`



FLAT (binary valued)

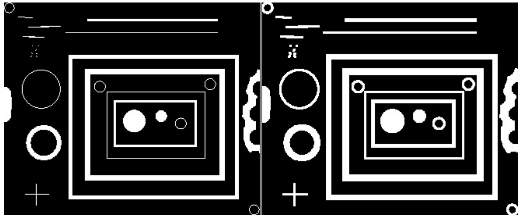


NON-FLAT (real valued)

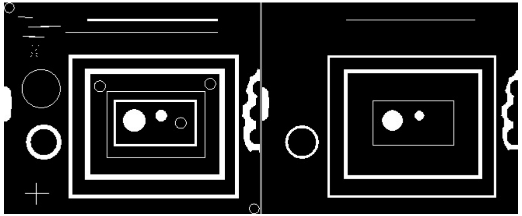
Images from MathWorks

2 basic morphological operations:

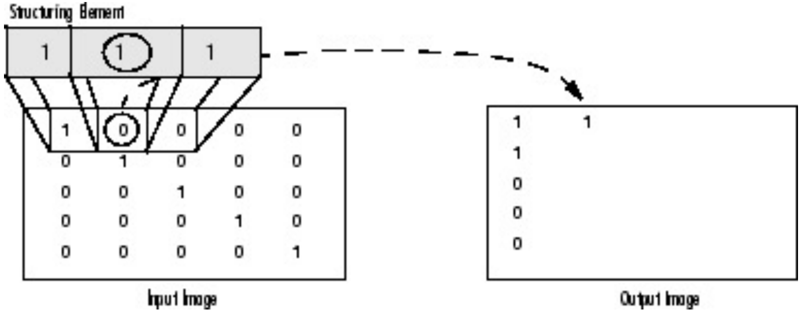
- **Dilation** adds pixels to the boundaries of objects in an image.



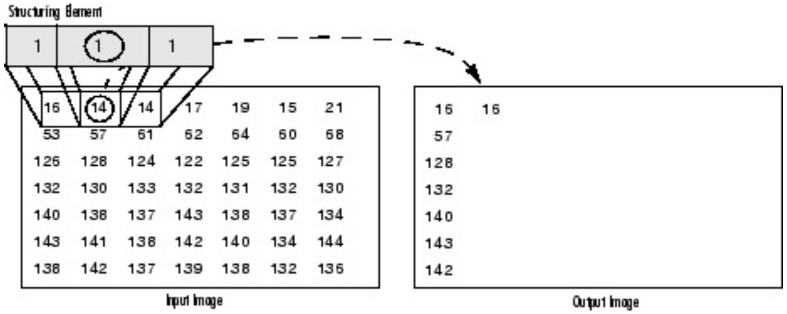
- **Erosion** removes pixels on object boundaries.



Morphological Dilation of a Binary Image



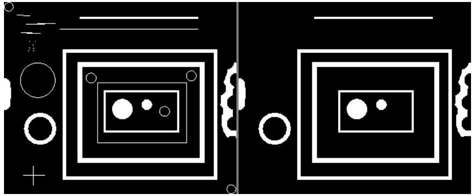
Morphological Dilation of a Grayscale Image



Images from MathWorks

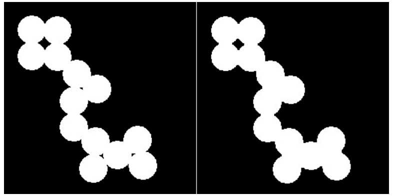
Other operations:

Morphological opening: `imopen`



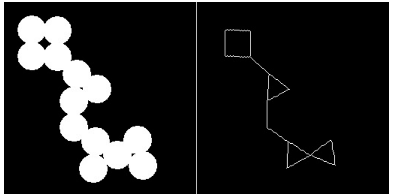
- erode an image and then dilate the eroded image
- useful for removing small objects and thin lines while preserving the shape and size of larger objects

Morphological closing: `imclose`



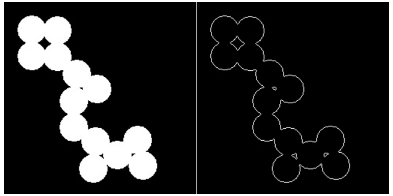
- dilate an image and then erode the dilated image
- useful for filling small holes while preserving the shape and size of large holes and objects

Skeletonization: `bwskel`



- erode all objects to centerlines without changing the essential structure of the objects

Finding perimeter: `bwperim`

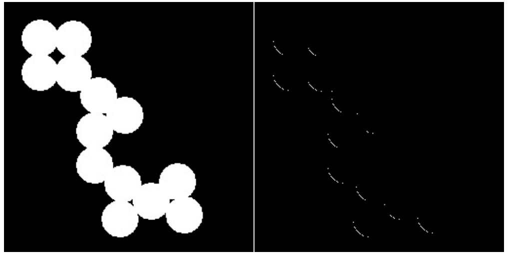


- find the perimeter of objects in a binary image.

Images from MathWorks

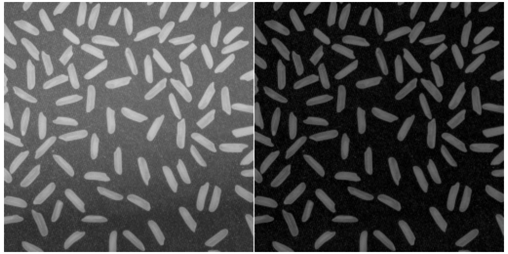
Binary hit-miss transform: `bwhitmiss`

- preserve pixels in a binary image whose neighborhoods match the shape of one structuring element and do not match the shape of a second disjoint one
- useful for detecting patterns in an image



morphological top-hat transform: `imtophat`

- open an image, then subtract the opened image from the original image
- useful for enhancing contrast in a grayscale image with nonuniform illumination



morphological bottom-hat transform: `imbothat`

- close an image, then subtract the original image from the closed image
- useful for finding intensity troughs in a grayscale image

Background Subtraction

Remove the background of an image.

- Convert to a grayscale image
- Define a flat structuring element
- Perform morphological closing on the grayscale or binary image
- Subtract the grayscale from the closed image
- Binarize (and reverse)

Image Saturation

Adjust the saturation of a color image by converting the image to the HSV color space (hue, saturation, value).

- Convert the image to the HSV color space.
- Process the HSV image and increases the saturation of the image by multiplying the S channel by a scale factor.
- Convert the processed HSV image back to the RGB color space.

Computer Vision

Advanced Tools for Images and Signals

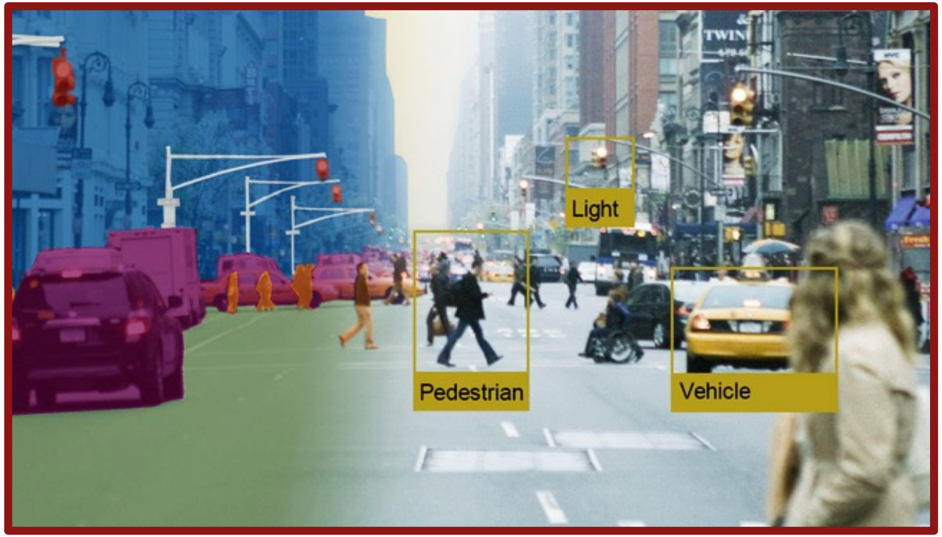


Image from MathWorks

Computer Vision Toolbox

It provides tools and algorithms for:

- object detection and tracking
- feature detection, extraction, and matching
- automating calibration workflows for single, stereo, and fisheye cameras
- visual and point cloud SLAM, stereo vision, structure from motion, and point cloud processing
- ... ...

Computer vision is a set of techniques for extracting information from images, videos, or point clouds.

Computer vision includes image recognition, object detection, activity recognition, 3D pose estimation, video tracking, and motion estimation.

Real-world applications include face recognition for logging into smartphones, pedestrian and vehicle avoidance in self-driving vehicles, and tumor detection in medical MRIs.

How it works?

1. Image processing techniques
- a preprocessing step in the computer vision workflow

2. Point cloud processing
- Point clouds are a set of data points in 3D space that together represent a 3D shape or object.

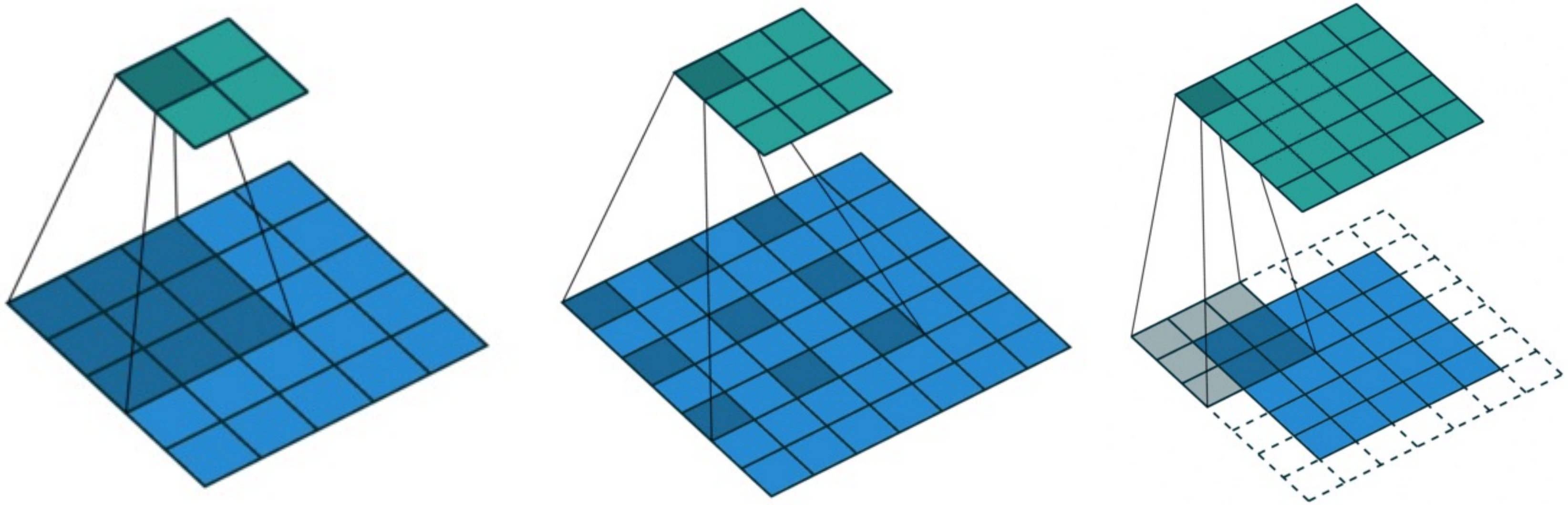
3. 3D Vision Processing
- estimating the 3D structure of a scene using multiple images taken with a calibrated camera

4. Feature-based techniques
- image alignment, video stabilization, object detection, …

5. Deep learning-based techniques
- Object detection, object recognition, image deblurring, and scene segmentation, …
- Training convoluted neural networks (CNNs)
- Transfer learning uses pretrained networks

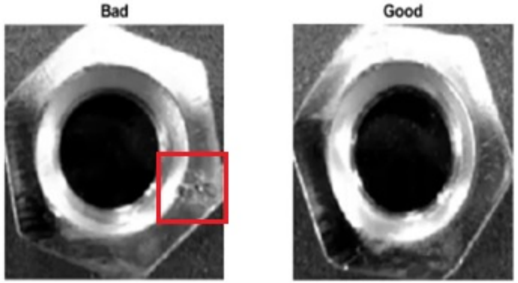
CNN, dilated CNN, and padding

Images from MathWorks

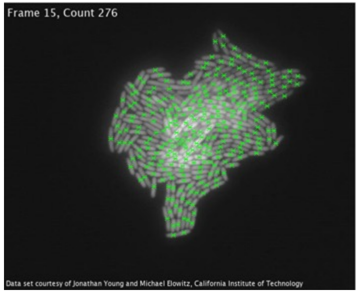


Computer Vision with MATLAB

defect detection
object detection and tracking
autonomous system simulation
localization and mapping
object counting



Images from MathWorks



Stanford University

Demo:
Semantic Segmentation Using Dilated Convolutions

32-by-32 triangle images

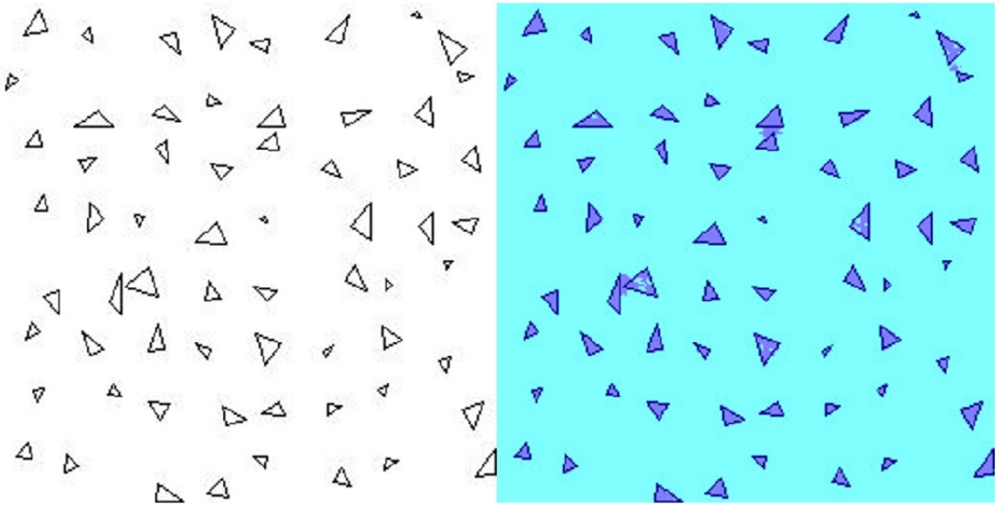


Image from MathWorks

Image Acquisition

Advanced Tools for Images and Signals

Image from MathWorks

Image Acquisition Toolbox

It provides tools and algorithms for:

- connecting cameras to MATLAB and Simulink
- processing in-the-loop
- hardware triggering
- background acquisition
- synchronizing acquisition across multiple devices
- ...

It can be used with:

- Built-in camera
  - Image Acquisition Toolbox Support Package for OS Generic Video Interface
  - use adaptor '`macvideo`', '`winvideo`', etc.
- USB Video Class (UVC) compliant webcam
  - MATLAB Support Package for USB Webcams
  - `webcamlist, webcam`
- Other hardwares
  - DCAM, GenTL, Matrox frame grabbers, Point Grey, GigE vision, Kinect for Windows Sensor, National Instruments frame grabbers, Teledyne DALSA Sapera cameras, Hamamatsu, QImaging, IP Cameras

Signal Processing

Advanced Tools for Images and Signals

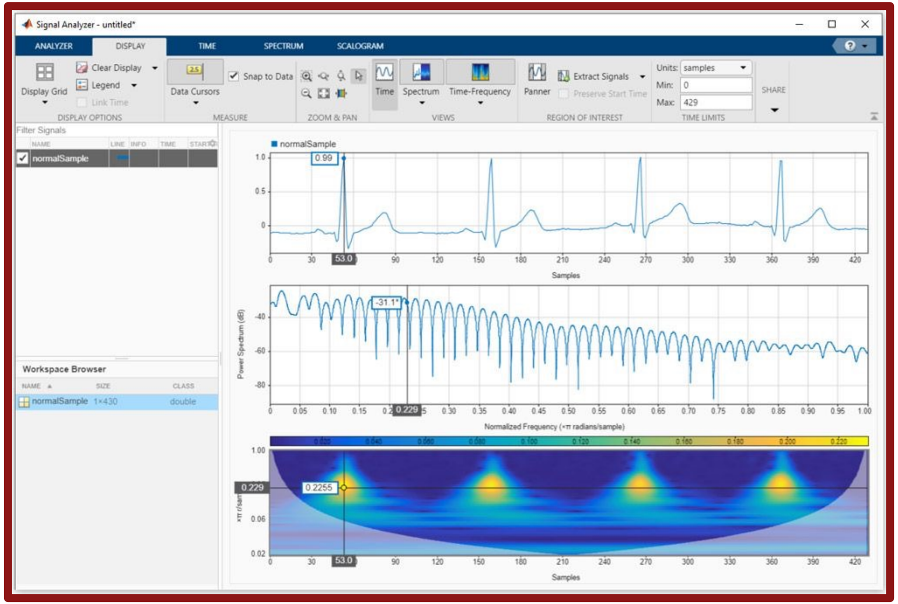


Image from MathWorks

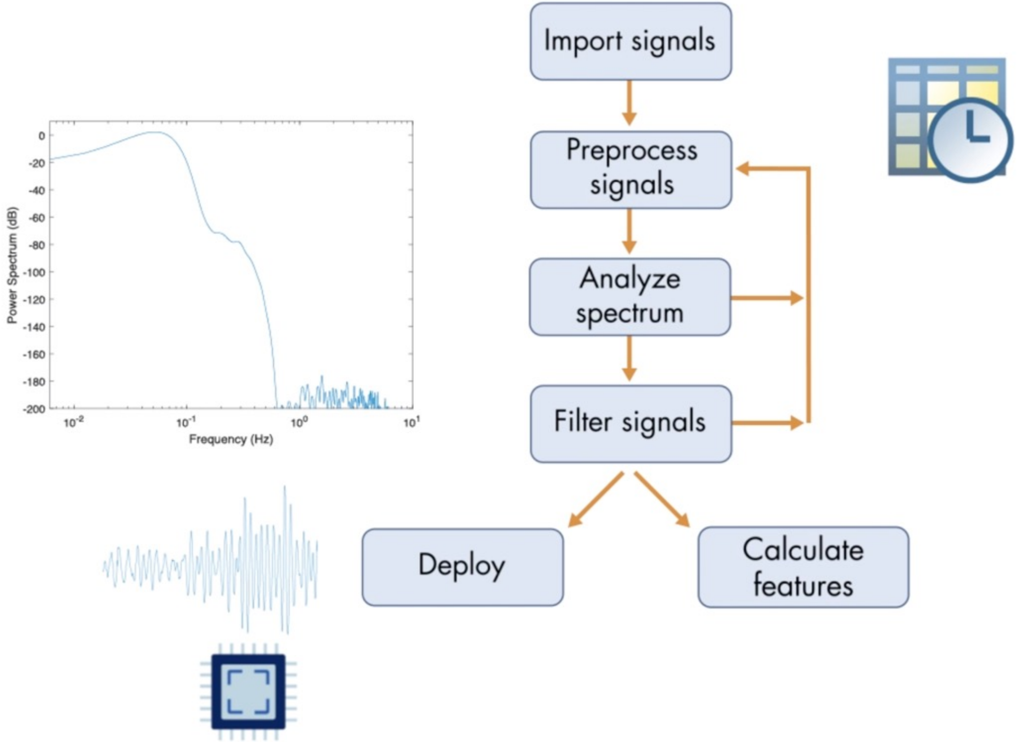
Signal Processing Toolbox

Analyze, preprocess, and extract features from uniformly and nonuniformly sampled signals

It provides tools and algorithms for:

- filter design and analysis
- resampling, smoothing, detrending
- power spectrum estimation
- extracting features like changepoints and envelopes
- finding peaks and signal patterns
- quantifying signal similarities
- performing measurements such as SNR and distortion
- ... ...

Signal Processing Pipeline



Some basic preprocessing steps:

- Resampling
  - `y = resample(x,p,q)`
  - resample the sequence x at p/q times the original sample rate
  - The length of the result y is p/q times the length of x.
- Normalizing
  - `normalize`
- Aligning
  - `synchronize`
  - `finddelay` (estimate the delay between signals)

Spectral Analysis

Any physical signal can be decomposed into a number of discrete frequencies, or a spectrum of frequencies over a continuous range.

Power Spectrum

- For a given signal, the power spectrum gives a plot of the portion of a signal's power (energy per unit time) falling within given frequency bins.
- `P = pspectrum(X)` returns the power spectrum of `x`.
- `pspectrum(...)` with no output arguments plots the spectral estimates.
- A dB scale is usually used to visualize power spectrums by calculating `10*log10(p)`, where *p* is the spectrum.

Spectrogram

- `s = spectrogram(x)` returns the Short-Time Fourier Transform (STFT) of the input signal x. Each column of s contains an estimate of the short-term, time-localized frequency content of x.
- `spectrogram(___)` with no output arguments plots ps in decibels.

Filtering

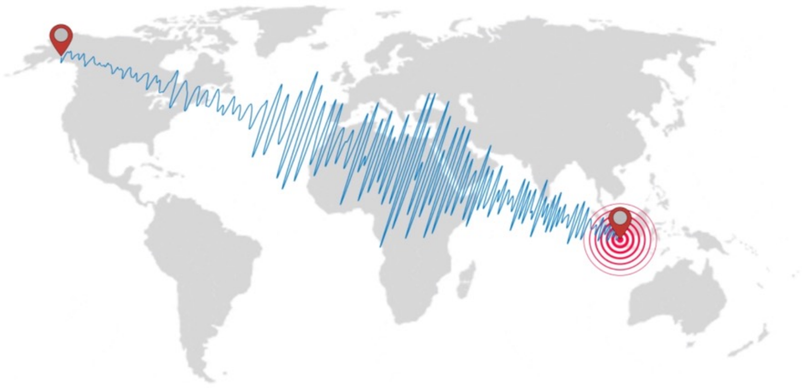
Remove some frequencies or frequency bands from a signal.

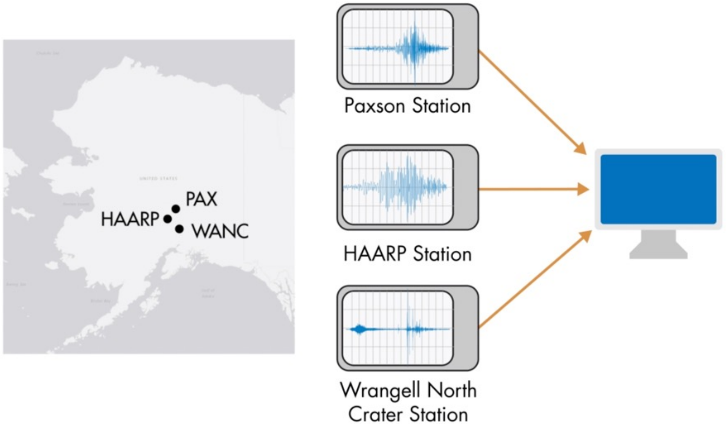
- Lowpass filter
  - Low frequencies are passed, high frequencies are attenuated.
  - `y = lowpass(x,wpass)` filters the input signal x using a lowpass filter with normalized passband frequency wpass in units of π rad/sample.
- Highpass filter
  - High frequencies are passed, low frequencies are attenuated.
  - `y = highpass(x,wpass)` filters the input signal x using a highpass filter with normalized passband frequency wpass in units of π rad/sample.
- Bandpass filter
  - Only frequencies in a frequency band are passed.
  - `y = bandpass(x,wpass)` filters the input signal x using a bandpass filter with a passband frequency range specified by the two-element vector wpass and expressed in normalized units of π rad/sample.

Demo:
Signal processing methods for spectral analysis

Earthquak data (December 26, 2004, Indonesia)

The vibrations travel all the way to Alaska, and are picked up by three seismic stations located in different places.





Audio

Advanced Tools for Images and Signals

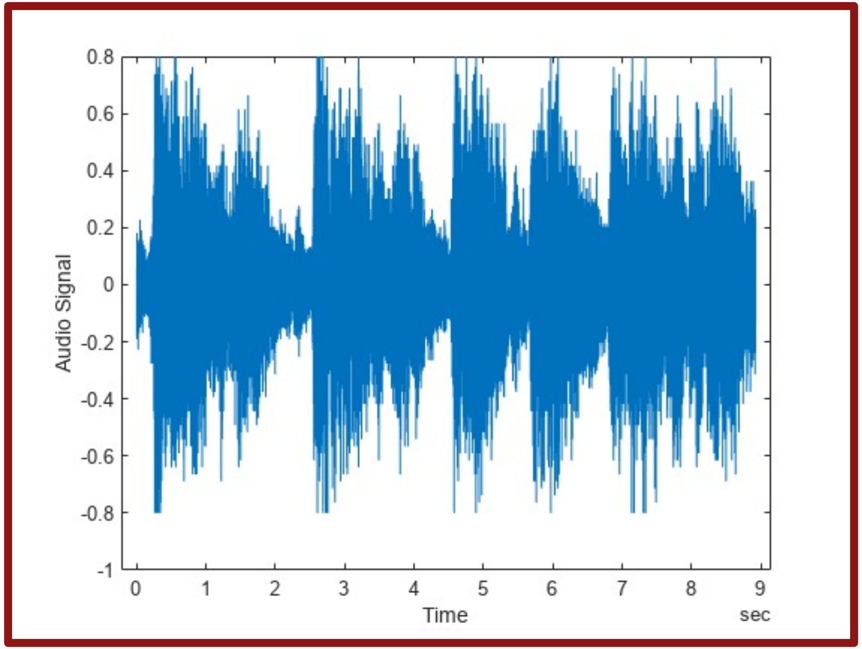


Image from MathWorks

Audio Toolbox

It provides tools and algorithms for:

- processing audio signals such as equalization and time stretching
- estimating acoustic signal metrics such as loudness and sharpness
- extracting audio features such as MFCC and pitch
- advanced machine learning models, including i-vectors
- pretrained deep learning networks, including VGGish and CREPE
- live algorithm testing
- impulse response measurement
- signal labeling
- ... ...

- With Audio Toolbox you can import, label, and augment audio data sets, as well as extract features to train machine learning and deep learning models. The pre-trained models provided can be applied to audio recordings for high-level semantic analysis.

- You can prototype audio processing algorithms in real time or run custom acoustic measurements by streaming low-latency audio to and from sound cards. You can validate your algorithm by turning it into an audio plugin to run in external host applications such as Digital Audio Workstations.

Read and Write Audio Files

Call `audioread` with a file name to read the entire audio file and the sample rate of the audio.

Call `soundsc` with the audio data and sample rate to play the audio to your default speakers.

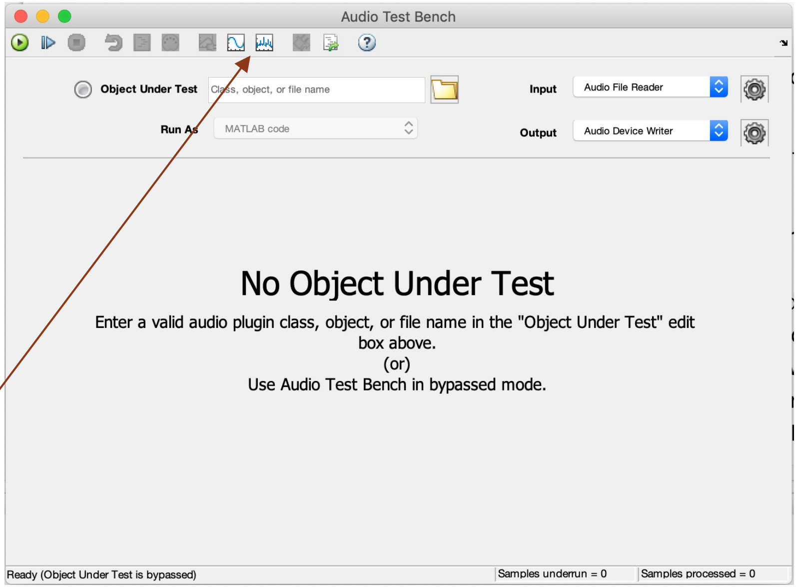
Call `audiowrite` with the file name, the audio data, and the sample rate to write the audio to a file.

Audio Test Bench

The Audio Test Bench app allows graphically setting up the audio input and output, processing audio, and opening common analysis tools like timescope and spectrumAnalyzer.

`audioTestBench`

analyze the audio signal in the time and frequency domains



DSP System

Advanced Tools for Images and Signals

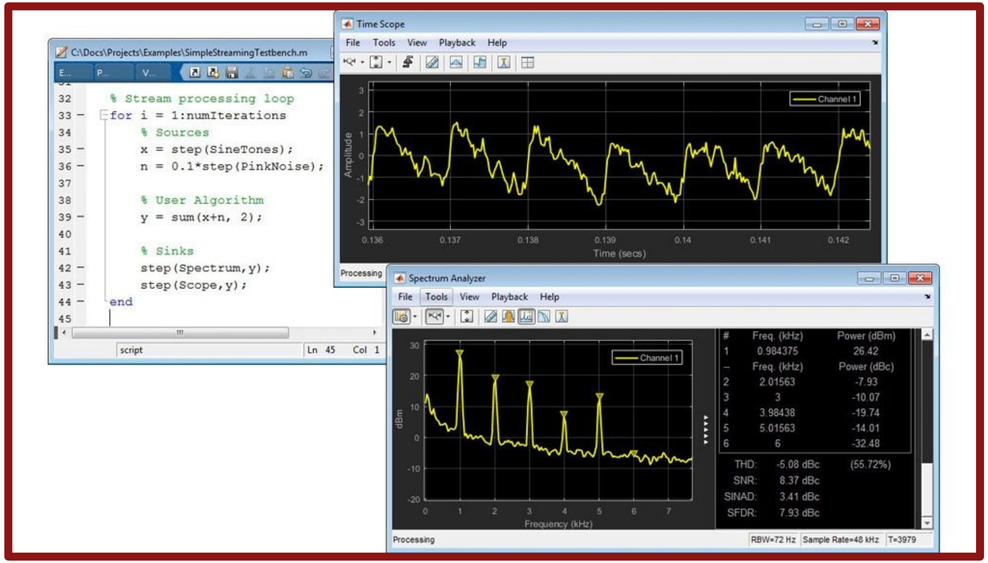


Image from MathWorks

Stanford University

DSP System Toolbox

Design and simulate streaming signal processing systems

It provides tools and algorithms for:

- design and analyze FIR, IIR, multirate, multistage, and adaptive filters
- stream signals from variables, data files, and network devices for system development and verification
- dynamically visualize and measure streaming signals using the Time Scope, Spectrum Analyzer, and Logic Analyzer
- support C/C++ code generation for desktop prototyping and deployment to embedded processors
- support bit-accurate fixed-point modeling and HDL code generation from filters, FFT, IFFT, and other algorithm
- ... ...

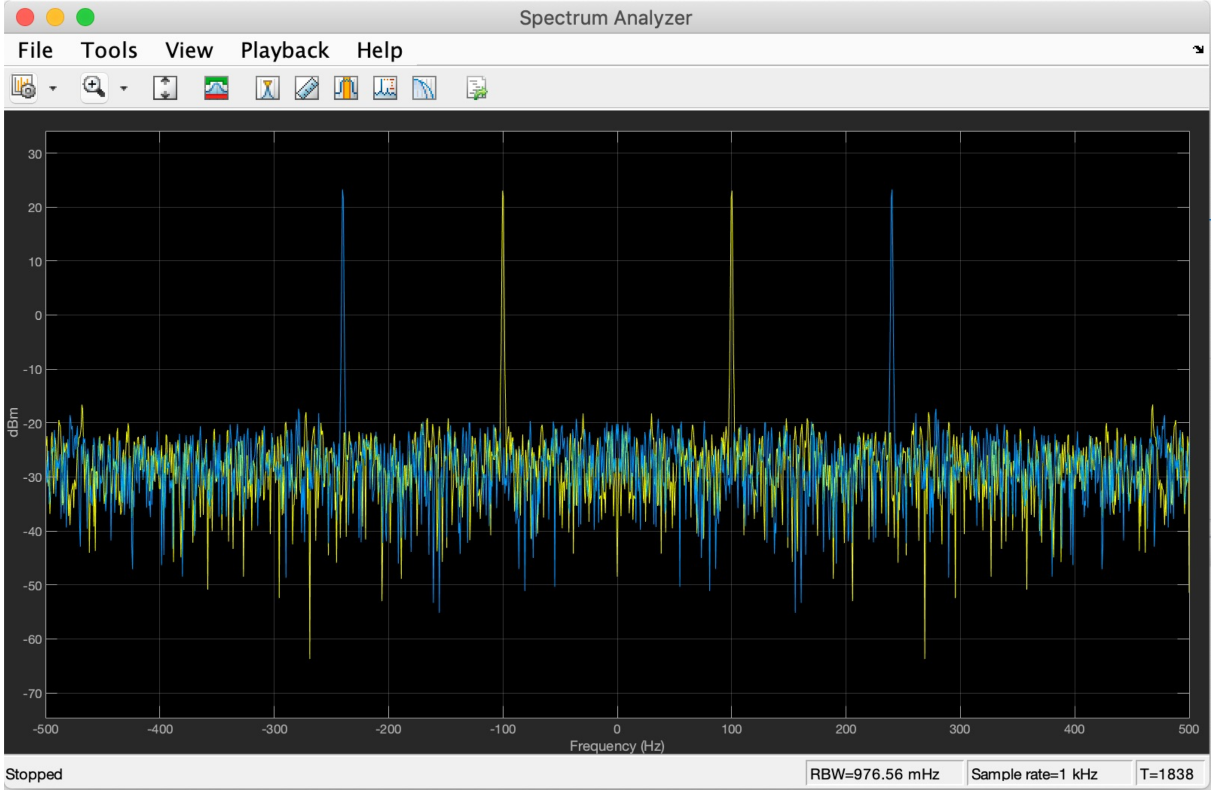
Visualize signals in frequency domain

The **Spectrum Analyzer** object displays the frequency spectrum of time-domain signals.

Frame size is the first dimension of the input vector. The number of input channels must remain constant.

- Create the `dsp.SpectrumAnalyzer` object and call the object with arguments to display the spectra of signals in the Spectrum Analyzer.
- Set the `NumInputPorts` property to display multiple signals.
- `scope(signal1,signal2,...)` displays multiple signals in the spectrum analyzer. The signals must have the same frame length, but can vary in number of channels.
- Release system resources of a System object named `obj` by `release(obj)`.

Spectrum Analyzer

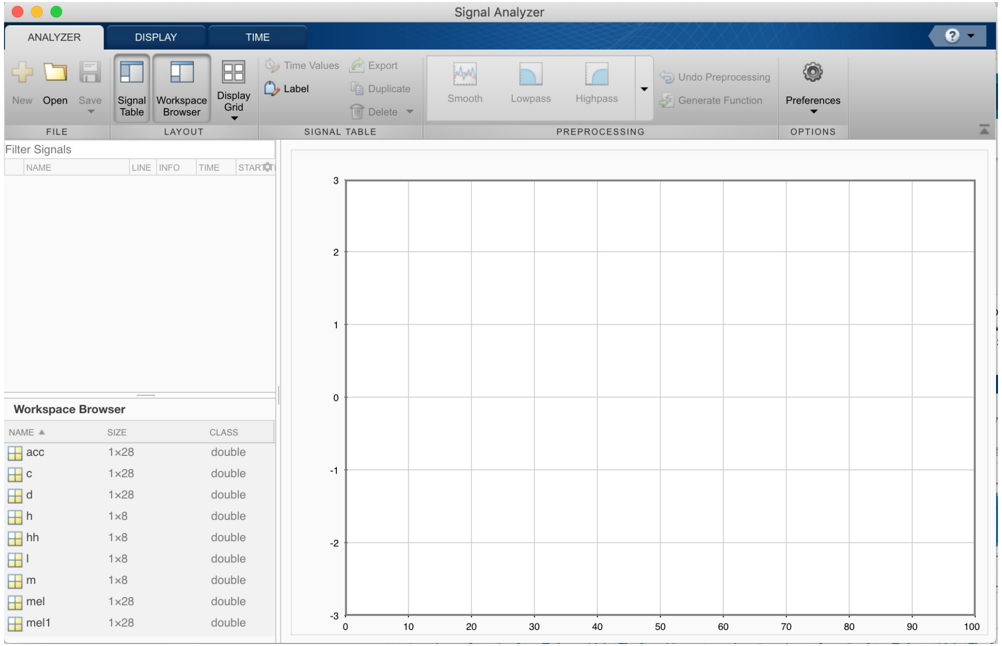


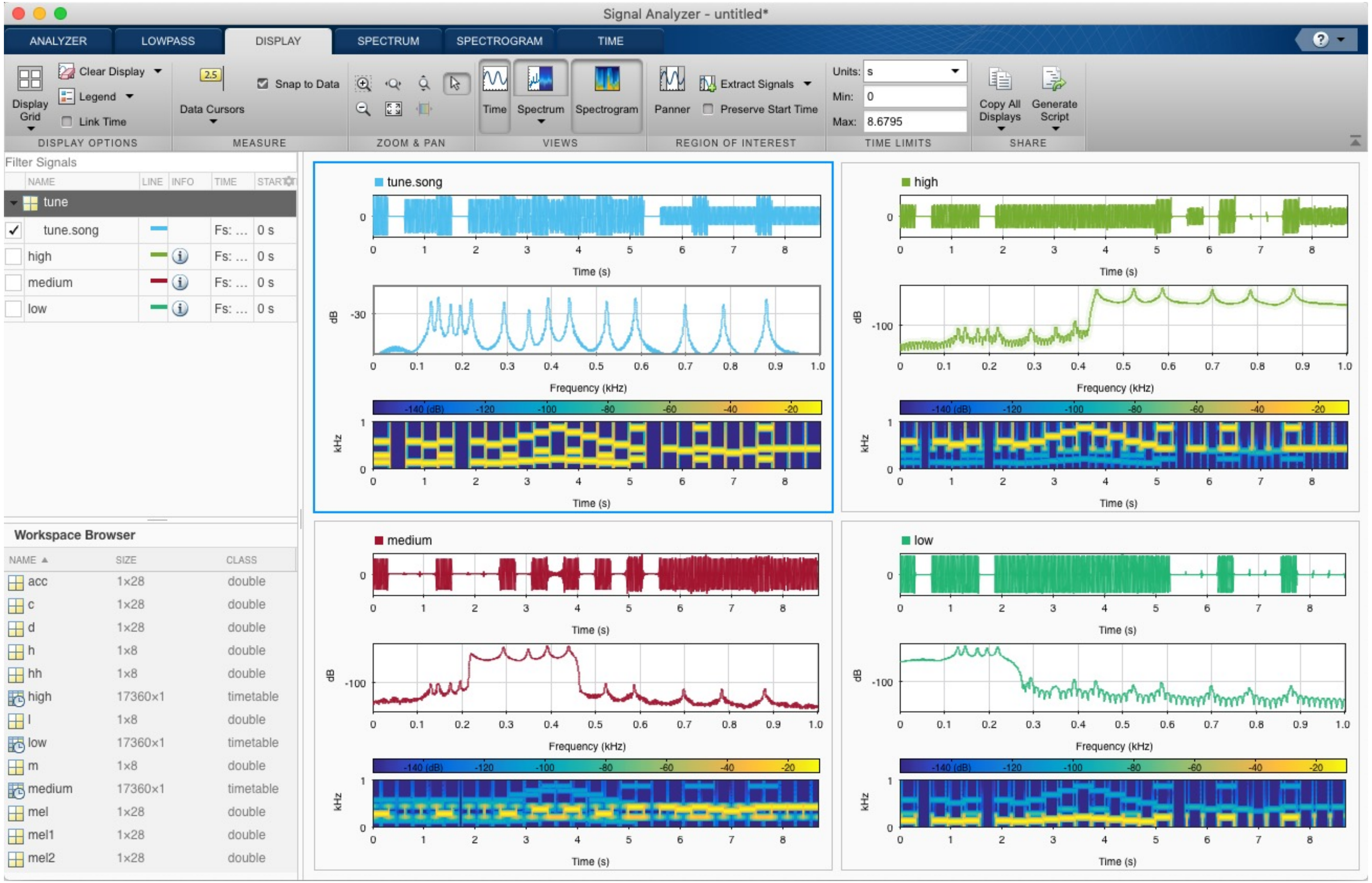
Demo

Demo: extract voices from music signal

Implement a basic digital music synthesizer and use it to play a traditional song in a three-voice arrangement.

Signal Analyzer





Stanford University

Fun with MATLAB

Type `xpsound` in Command Window.

Next Lecture

Project presentations!

Addition Topic