Image Processing with MATLAB

CME 192 LECTURE 7

02/19/2026

Stanford University

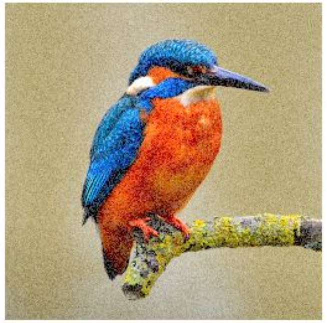
Outline

- Image Processing Toolbox
- Computer Vision Toolbox
- Image Acquisition Toolbox
- Image Processing Demos with Livescripts
- Intro to Signal Processing (to be finished next lecture)

Image Processing

Working with Images

Original image


Noisy image

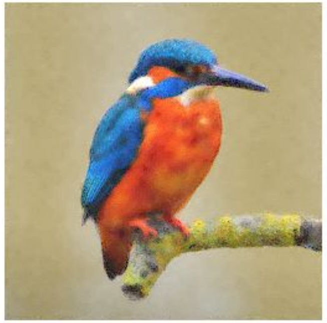

Denoised image

Image Processing Toolbox

Image Processing Toolbox provides tools and algorithms for:

- Image segmentation
- Image enhancement
- Noise reduction
- Geometric transformations
- Image Registration
- 3D image processing
- And more!

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

Basic image operations

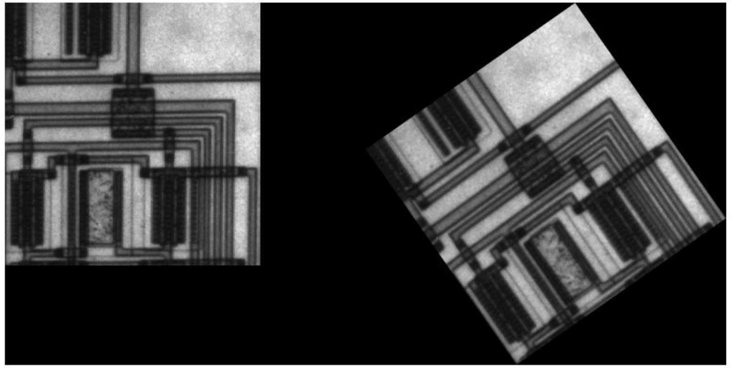
Import, display, and resize images
- Read images with imread; inspect using size, class
- Display using imshow, montage
- Resize with imresize; convert types with im2double / im2uint8

Convert color images to grayscale
- RGB images are M×N×3
- Use rgb2gray (weighted R/G/B)
- Result is single-channel intensity image

Color images (RGB)
- Three channels: Red, Green, Blue
- Access channels: img(:,:,1/2/3)

Grayscale images
- Single intensity value per pixel
- Useful for filtering, histograms, contrast ops

Rotate and compare images
- Rotate with imrotate
- Compare versions using imshowpair or montage



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 image contrast by remapping intensity values; by default it saturates the lowest/highest 1% of intensities to stretch the dynamic range.

- Use when: the image looks “washed out” or occupies a narrow intensity range.

histeq performs global histogram equalization, redistributing pixel intensities so the output histogram is roughly uniform.

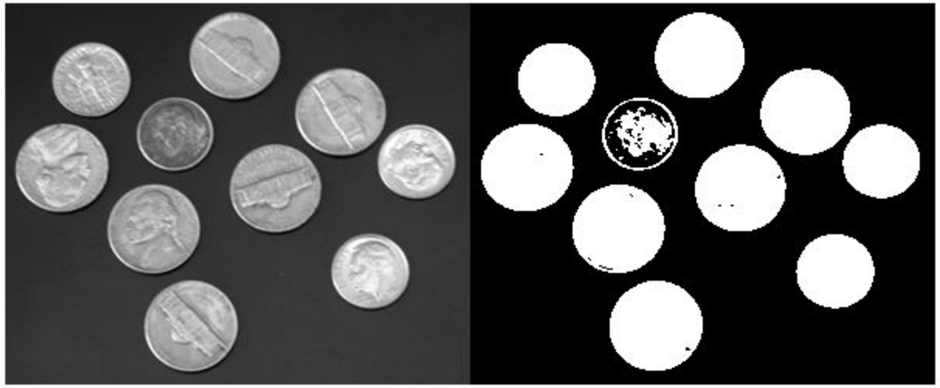
- Use when: you want stronger global contrast in images where the background and foreground are not well separated.

adapthisteq (CLAHE) applies contrast-limited adaptive histogram equalization on small tiles instead of the entire image. It boosts local contrast while preventing noise amplification.

- Use when: the image has uneven lighting, local shadows, or regions of interest that need localized contrast enhancement without blowing up noise.

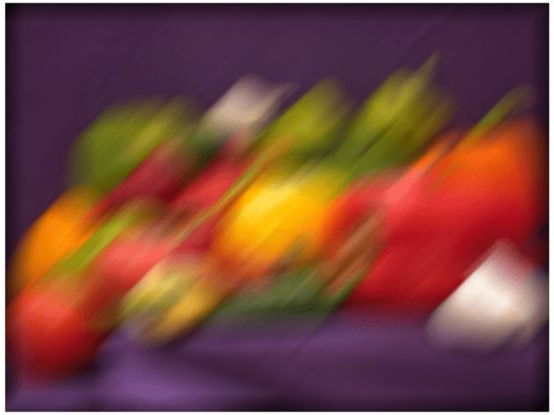
Working with Binary Images

- Convert a grayscale image to a binary (logical) image by thresholding pixel intensities.
- Use **imbinarize(I, METHOD)** for automatic threshold selection.
- Global thresholding: one threshold applied to the entire image; good when lighting is uniform.
- Local (adaptive) thresholding: threshold computed per-region; useful for images with uneven illumination or shadows.
- Binary images are often used for segmentation, object detection, and measuring regions (area, perimeter, bounding boxes) using functions like **bwlabel, regionprops,** and **bwmorph**.



Filtering 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.
- **imfilter(A,H)** filters the multidimensional array A with the multidimensional filter 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.

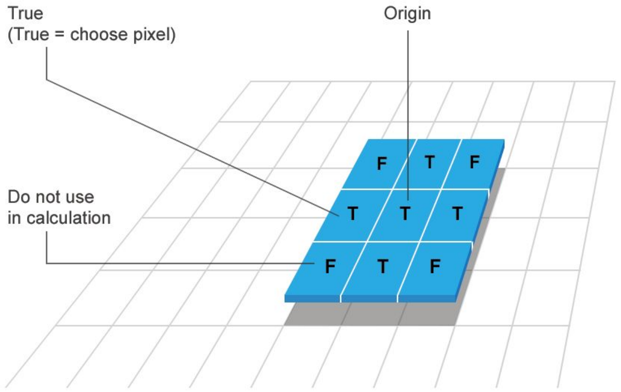
Structuring Elements

Structuring Element

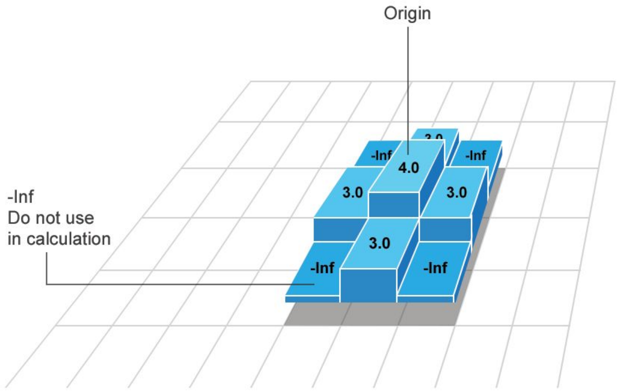
- A matrix that identifies the pixel in the image being processed and defines the neighborhood used in the processing of each pixel.
- The center pixel of the matrix, called the origin, identifies the pixel in the image that is being processed

Two types of structuring elements:

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



FLAT (binary valued)

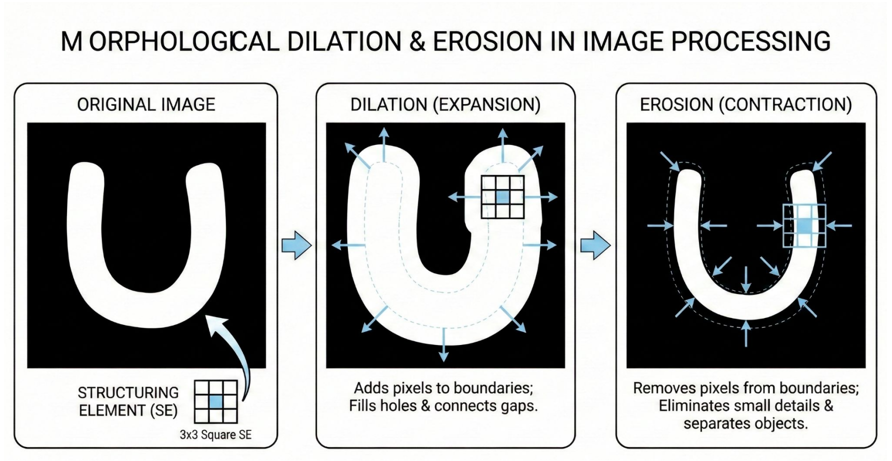


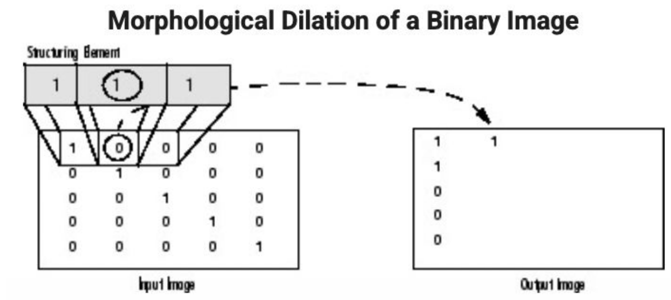
NON-FLAT (real valued)

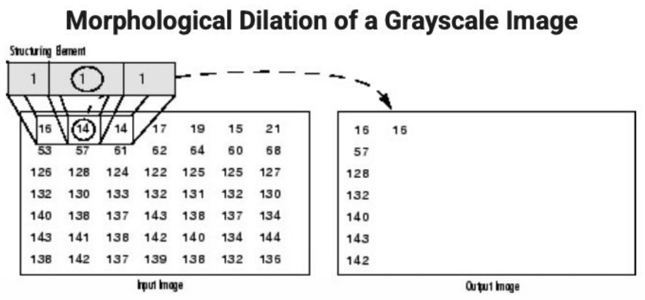
Two Basic Morphological Operations

Dilation expands shapes in a binary image by adding pixels along object boundaries. Objects appear thicker, gaps become smaller, and small holes may close.

Erosion shrinks shapes by removing pixels along their boundaries. Objects become thinner, small protrusions disappear, and gaps grow larger.

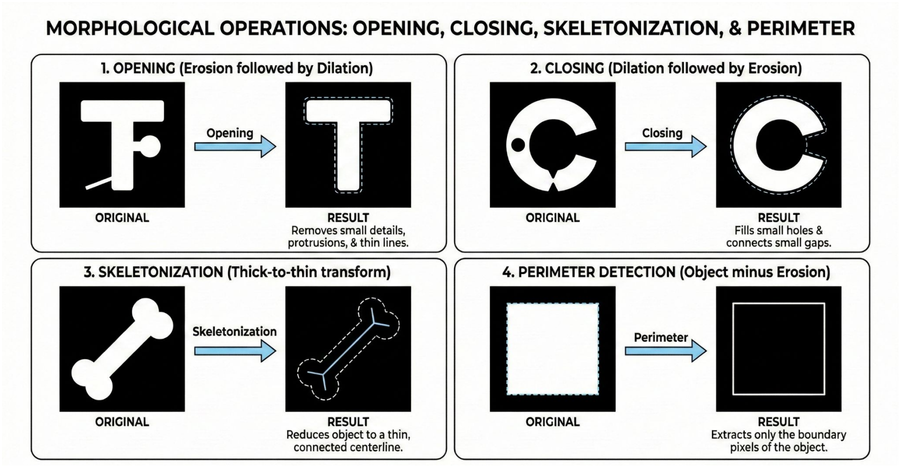

M ORPHOLOGICAL DILATION & EROSION IN IMAGE PROCESSING
ORIGINAL IMAGE
STRUCTURING ELEMENT (SE)
3x3 Square SE
DILATION (EXPANSION)
Adds pixels to boundaries; Fills holes & connects gaps.
EROSION (CONTRACTION)
Removes pixels from boundaries; Eliminates small details & separates objects.





Other Morphological Operations

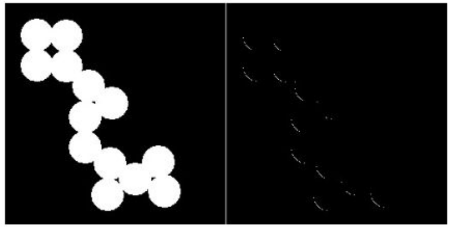
- **Morphological opening** (`imopen`): Erode an image, then dilate the result. Useful for removing small objects or thin artifacts while preserving the shape of larger structures.
- **Morphological closing** (`imclose`): Dilate an image, then erode the result. Useful for filling small holes or gaps while maintaining the overall shape of objects.
- **Skeletonization** (`bwskel`): Reduces objects to their centerlines while preserving their connectivity and overall structure.
- **Perimeter detection** (`bwperim`): Computes the perimeter pixels of objects in a binary image.


MORPHOLOGICAL OPERATIONS: OPENING, CLOSING, SKELETONIZATION, & PERIMETER
1. OPENING (Erosion followed by Dilation)
ORIGINAL
Opening
RESULT
Removes small details, protrusions, & thin lines.
2. CLOSING (Dilation followed by Erosion)
ORIGINAL
Closing
RESULT
Fills small holes & connects small gaps.
3. SKELETONIZATION (Thick-to-thin transform)
ORIGINAL
Skeletonization
RESULT
Reduces object to a thin, connected centerline.
4. PERIMETER DETECTION (Object minus Erosion)
ORIGINAL
Perimeter
RESULT
Extracts only the boundary pixels of the object.

Other Morphological Operations

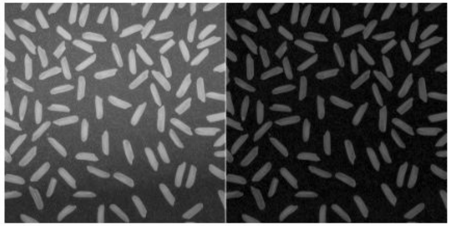
Binary hit–miss transform (`bwhitmiss`)

- Preserves pixels whose neighborhood matches one structuring element and does not match a second, disjoint structuring element.
- Useful for detecting specific patterns or shapes in a binary image.



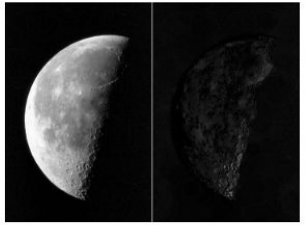
Morphological top-hat transform (`imtophat`):

- Opens an image, then subtracts the opened image from the original.
- Useful for enhancing bright details and correcting uneven illumination in grayscale images.



Morphological bottom-hat transform (`imbothat`):

- Closes an image, then subtracts the original image from the closed image.
- Useful for detecting dark features or troughs in grayscale images.



Background Subtraction

Remove the background of an image.

- Convert to a grayscale image
- Define a flat structuring element
- Perform a 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 increase the saturation of the iamge by multiplying the S channel by a scale factor.
- Convert the processed HSV image back to the RGB color space.

Color Saturation Practical Notes

- RGB mixes color and brightness; HSV separates Hue, Saturation, Value.
- Saturation controls colorfulness at a given brightness (Value).
- Saturation adjustment workflow:
  - Convert RGB to HSV.
  - Modify the S channel (e.g., scale or clamp).
  - Convert HSV back to RGB.

Image Moments

Image moments summarize the intensity distribution and shape of an object.

Raw (spatial) moments

$$m_{pq} = \sum_x \sum_y x^p y^q I(x,y)$$

Centroid (first moments)

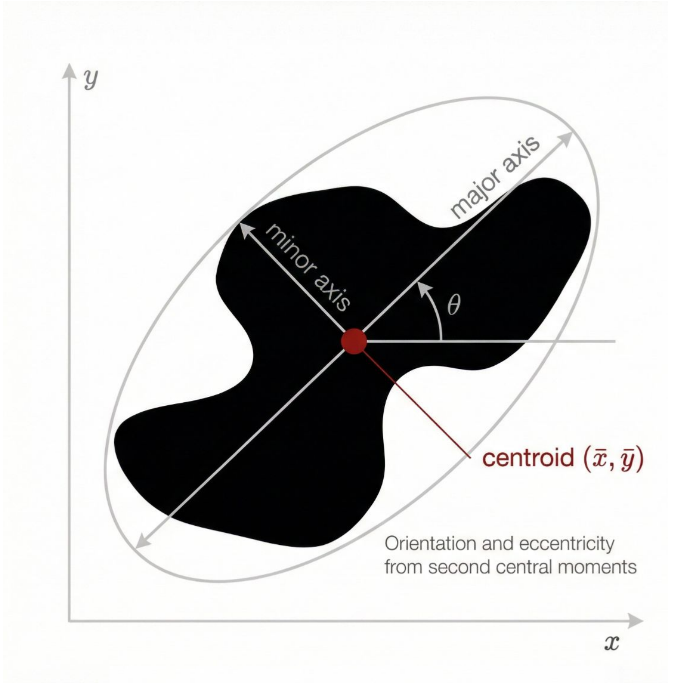
$$\bar{x} = \frac{m_{10}}{m_{00}}, \qquad \bar{y} = \frac{m_{01}}{m_{00}}$$

Central moments

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q I(x,y)$$

- Second-order moments relate to covariance, orientation, and eccentricity.

MATLAB:
`regionprops(BW,'Centroid','Orientation','MajorAxisLength','MinorAxisLength')`



Orientation and eccentricity from second central moments

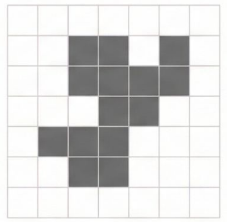
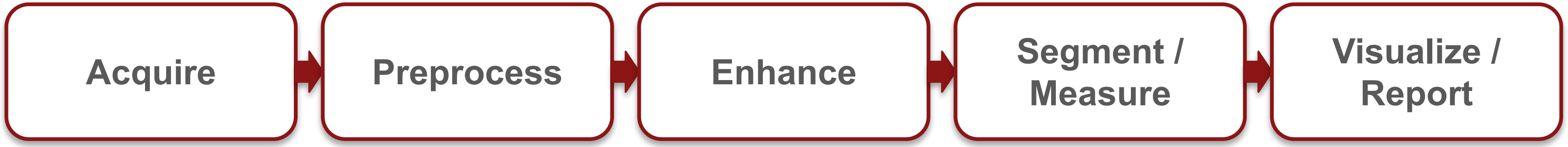

Weighted sums over pixel intensities

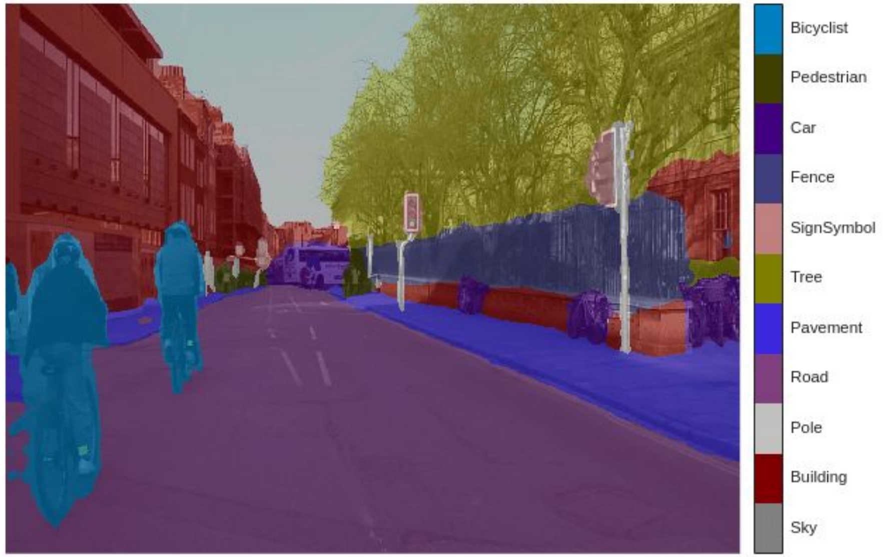
Image Processing Workflow (MATLAB)



- Acquire: read images / video, or capture from hardware (Image Acquisition Toolbox).
- Preprocess: denoise, normalize, convert color space.
- Enhance: contrast adjustment, histogram methods, filtering.
- Segment / Measure: thresholding, morphology, feature extraction.
- Visualize / Report: plots, overlays, measurements, exports.

Computer Vision

Computers “reasoning” about images and video



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
- … and more!

Computer Vision Overview

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
   a. Object detection, object recognition, image deblurring, and scene segmentation, …
   b. Training convoluted neural networks (CNNs)
   c. Transfer learning uses pretrained networks

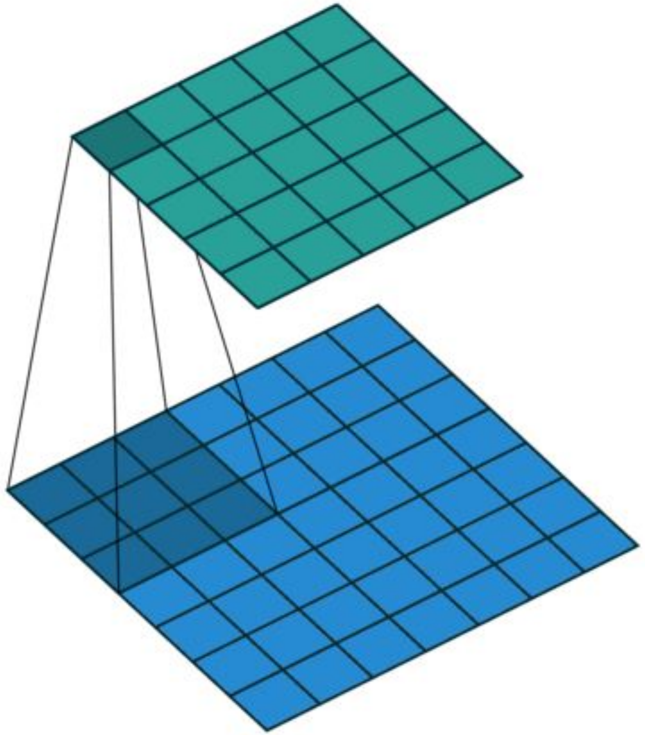
Convolutional Neural Networks (CNNs)

What a CNN convolution does

- Slides a learnable **k x k filter** over the input feature map: **dot product** at each location -> output activation map
- **Local connectivity + weight sharing**: far fewer parameters than fully connected layers
- **Receptive field** grows with depth (stacked convs “see” more context)

MATLAB (Deep Learning Toolbox)

- Use `convolution2dLayer(filterSize,numFilters, ...)` for standard CNN conv
- For custom / low-level ops: `dlconv(X,W,B, ...)` inside `dlnetwork` workflows



Dilated CNNs

What dilated convolution changes

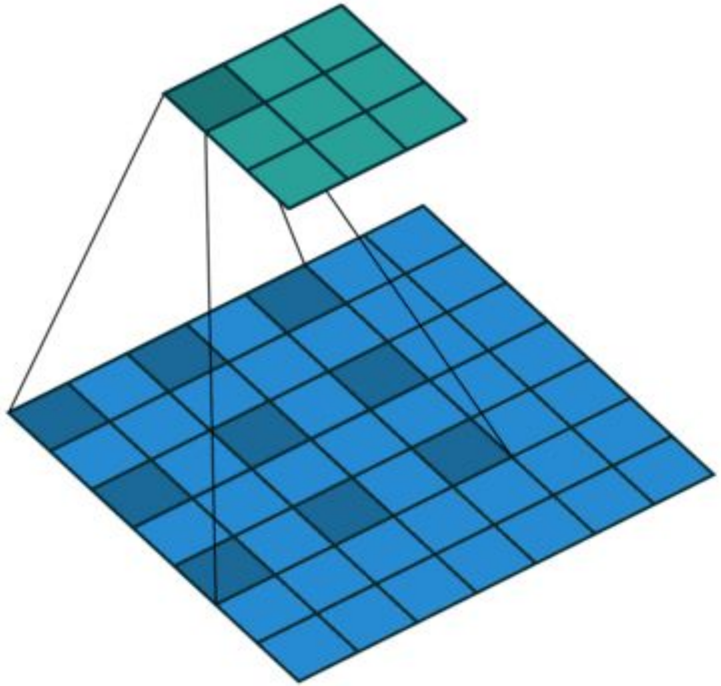
- Inserts “gaps” between kernel elements (**dilation factor d**): larger receptive field **without increasing parameter count**
- Effective kernel size: **k_eff = k + (k−1)(d−1)** (e.g., 3×3 with d=2 acts like 5×5 coverage)

MATLAB

- `convolution2dLayer([3 3], numFilters, "DilationFactor", d, ...)`
- `dlconv(..., "DilationFactor", d, ...)`

When to use dilations

- Segmentation / dense prediction / time-series where you want **more context at the same resolution**
- Trade-off: too many large dilations can create **“gridding” artifacts** -> often mix dilation rates



Padding

Why padding?

- Without padding, output shrinks each layer
- Padding preserves spatial size and keeps edge information

Common types

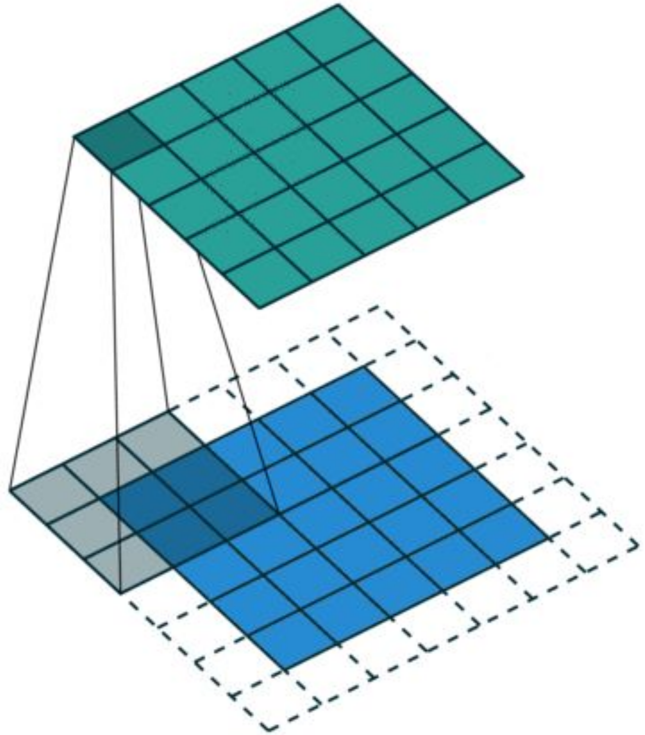
- **Valid:** no padding -> smaller output
- **Same:** output size = input size
- **Explicit:** manually set `[top bottom left right]`

MATLAB

- `convolution2dLayer(...,"Padding","same")`
- `convolution2dLayer(...,"Padding",[t b l r])`
- `padarray` for demos

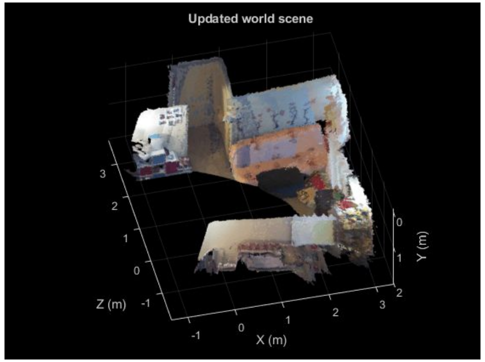
With dilation

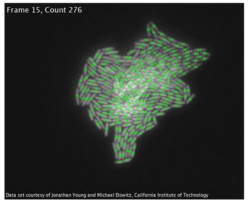
- Larger effective receptive field
- Often need more padding to keep size

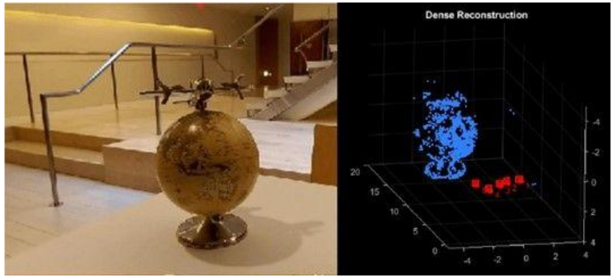


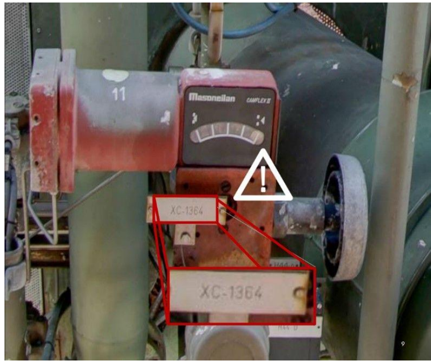
Example use-cases

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




Frame 15, Count 276
Data set courtesy of Jonathan Young and Michael Elowitz, California Institute of Technology





Object Detection

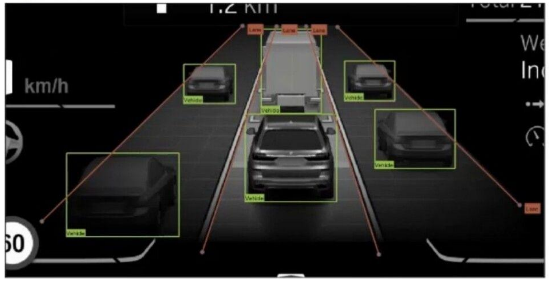
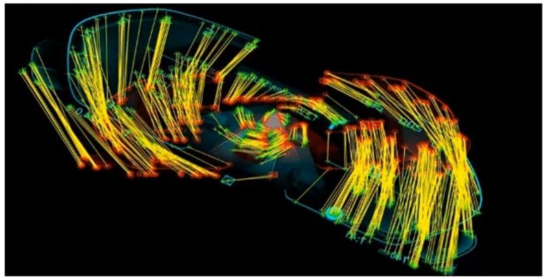
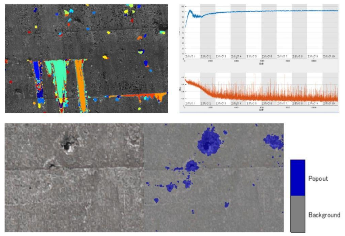


Image Registration





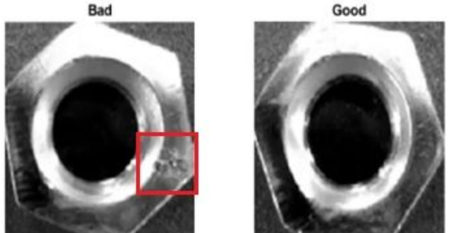


Image Acquisition

Advanced Tools for Images and Signals

Image Acquisition Toolbox

It provides tools and algorithms for:

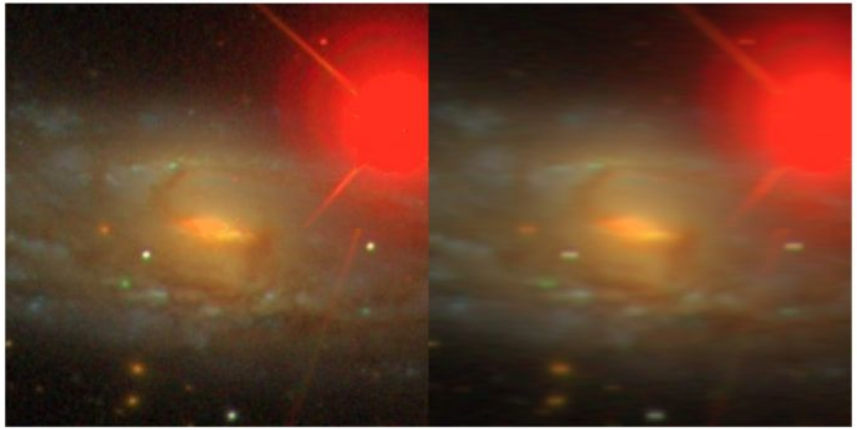
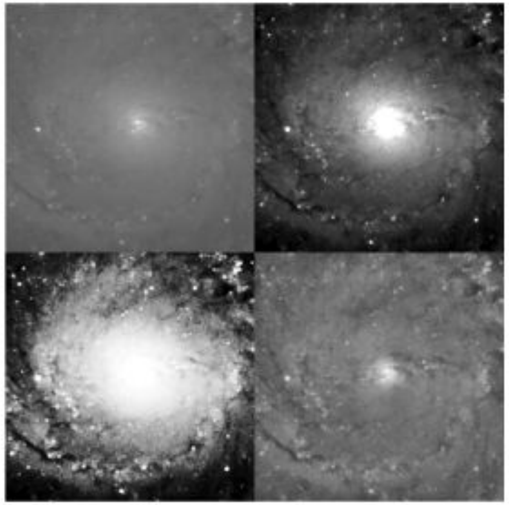
- connecting cameras to MATLAB and Simulink
- processing in-the-loop
- hardware triggering
- background acquisition
- synchronizing acquisition across multiple devices
- ...

It can be used with:

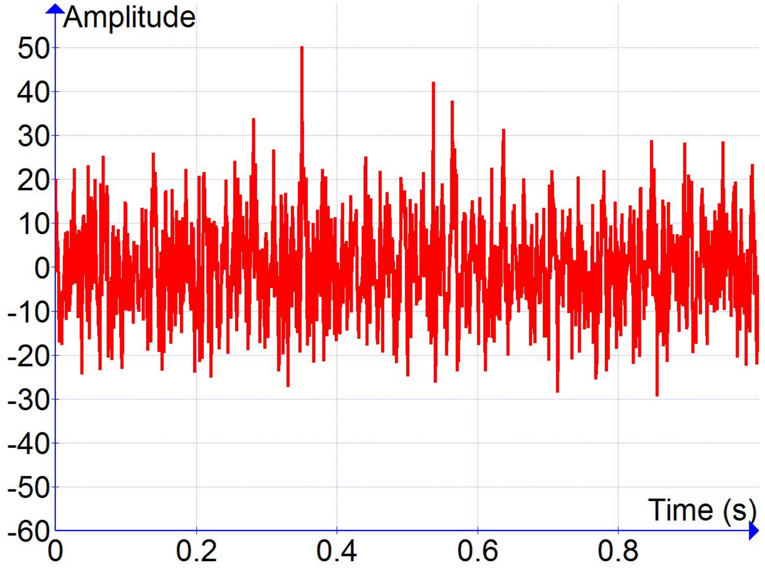
- Built-in camera
  - Image Acquisition Toolbox Support Package for OS Generic Video Interface
  - use adaptor '`macvideo`', '`winvideo`', etc.
- USB Video Class (UVC) compliant webcam
  - MATLAB Support Package for USB Webcams
  - `webcamlist, webcam`
- Other hardwares
  - DCAM, GenTL, Matrox frame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

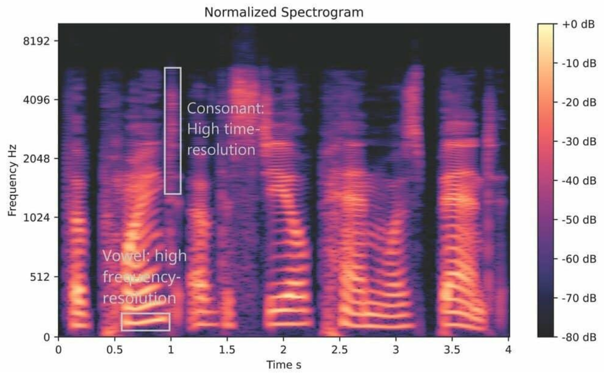
Livescripts on Image Processing

Shoot for the moon:
Astronomy Image Processing
with MATLAB

Signal Processing





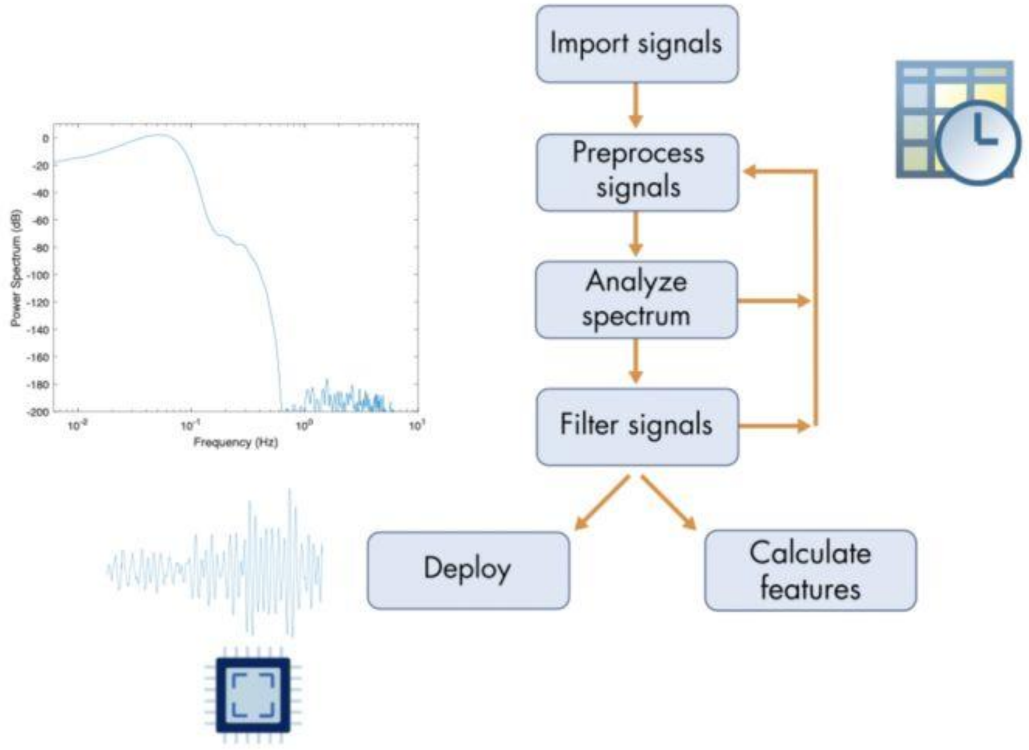
Signal Processing Toolbox

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

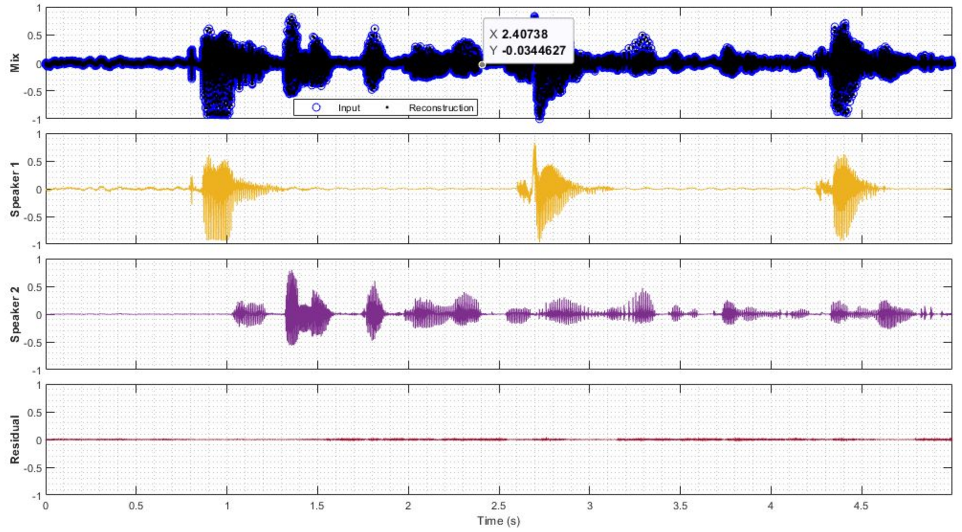
- `s = spectrogram(x)` returns the Short-Time Fourier Transform (STFT) of the input signal x. Each column of s contains an estimate of the short-term, time-localized frequency content of x.
- `spectrogram(___)` with no output arguments plots `ps` in decibels.

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.

Audio Toolbox



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.

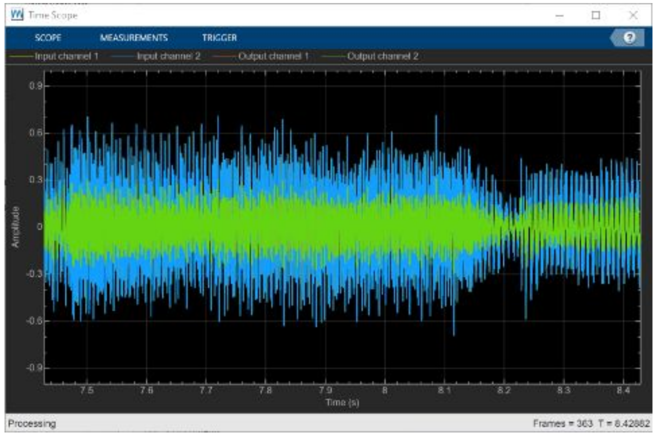
Call `soundsc` with the audio data and sample rate to play the audio to your default speakers.

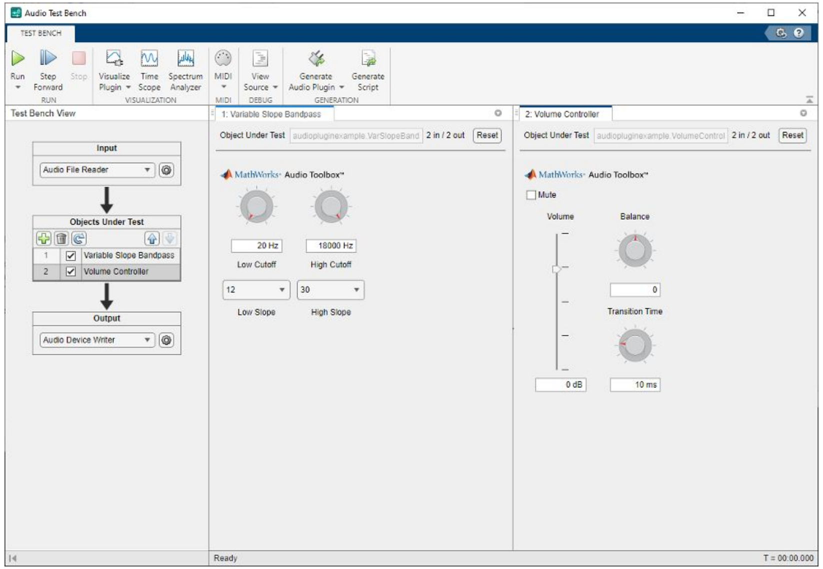
Call `audiowrite` with the file name, the audio data, and the sample rate to write the audio to a file.

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





Test Bench View

Input: Audio File Reader

Objects Under Test:
1 ✓ Variable Slope Bandpass
2 ✓ Volume Controller

Output: Audio Device Writer

1: Variable Slope Bandpass
Object Under Test: audiopluginexample.VarSlopeBand 2 in / 2 out Reset
Low Cutoff: 20 Hz
High Cutoff: 18000 Hz
Low Slope: 12
High Slope: 30

2: Volume Controller
Object Under Test: audiopluginexample.VolumeControl 2 in / 2 out Reset
Mute
Volume: 0 dB
Balance: 0
Transition Time: 10 ms

T = 00:00.000

Analyze the audio signal in the time and frequency domains