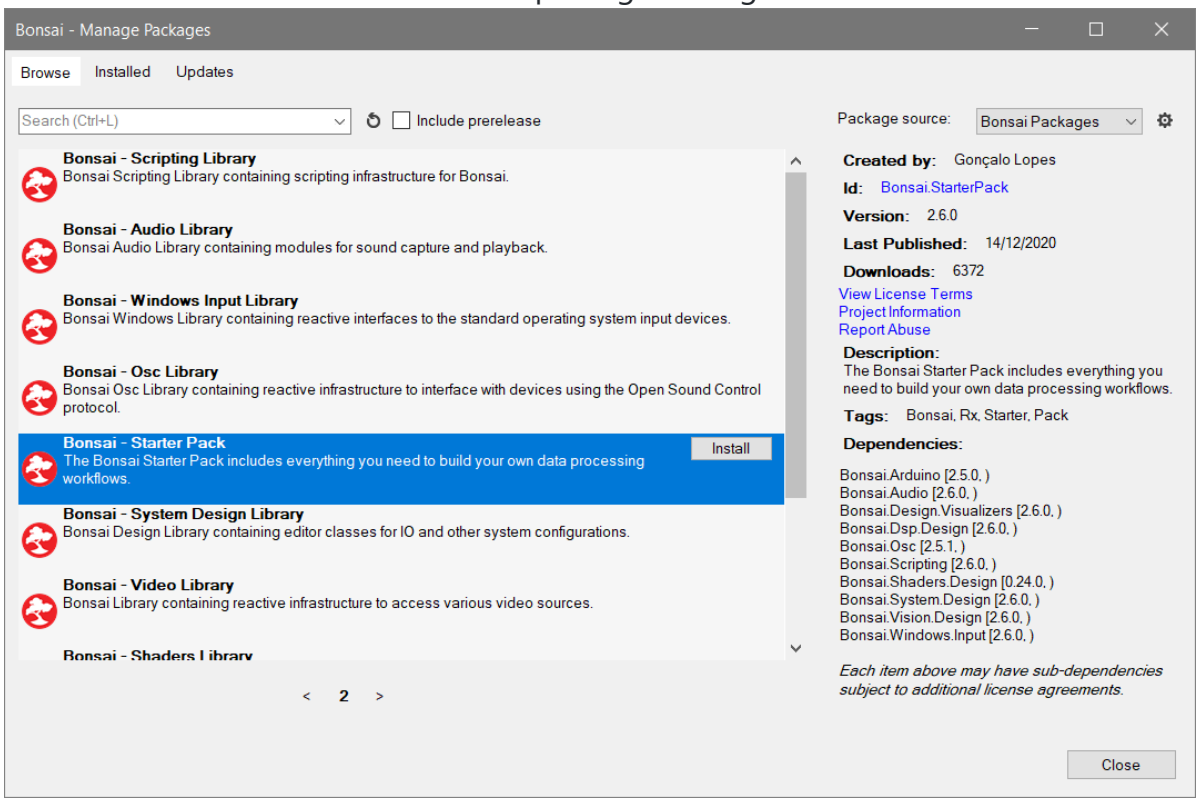
# Acquisition and Tracking

## Getting Started

1. Download Bonsai from http://bonsai-rx.org.
2. Install **Bonsai - Starter Pack** from the package manager.




3. Click on the `Updates` tab at the top of the screen and install any available upgrades.
4. Read http://bonsai-rx.org/docs/editor for an introduction to the user interface.

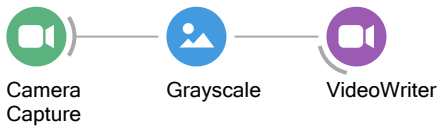
## Video Acquisition

Bonsai can be used to acquire and record data from many different devices. The exercises below will make you comfortable with the most common Bonsai data types. The first data type we will discuss is an image, which is represented as a 2D matrix of pixels. Each pixel represents either a brightness value in a grayscale image, or a BGR colour value in a colour image.

### Exercise 1: Saving a video


- Insert a `CameraCapture` source.
- Insert a `VideoWriter` sink.
- Configure the `FileName` property of the `VideoWriter` operator with a file name ending in `.avi`.
- Run the workflow and check that it generates a valid video file.

### Exercise 2: Saving a grayscale video




- Insert a `Grayscale` transform between `CameraCapture` and `VideoWriter`.
- Run the workflow. The output should now be a grayscale movie.
- Modify the workflow so that it records **simultaneously** a colour and a grayscale movie.

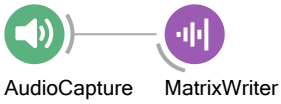
## Audio Acquisition

Audio data is captured at much higher temporal sampling frequencies than video. However, the data is typically buffered into chunks of multiple samples before being sent to the computer. Also, multi-channel data can be acquired simultaneously in the case of a stereo microphone, or high-density ephys probes. Such multi-sample, multi-channel data is typically represented as a 2D matrix, where rows represent channels, and columns represent time.

### Exercise 3: Saving a WAV file

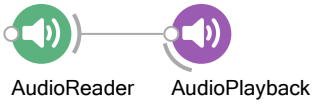

- Insert an `AudioCapture` source.
- Insert an `AudioWriter` sink.
- Configure the `FileName` property of the `AudioWriter` operator with a file name ending in `.wav`.
- Make sure that the `SamplingFrequency` property of the `AudioWriter` matches the frequency of audio capture.
- Run the workflow for some seconds. Playback the file in Windows Media Player to check that it is a valid audio file.

### Exercise 4 (Optional): Saving raw binary waveform data

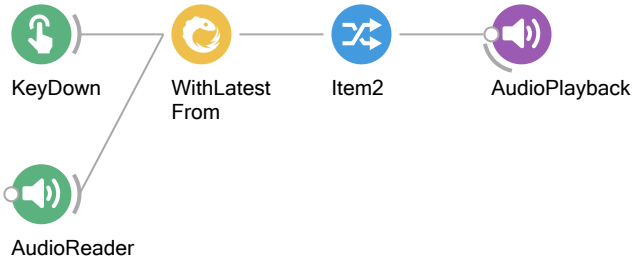



- Replace the `AudioWriter` operator with a `MatrixWriter` sink.
- Configure the `Path` property of the `MatrixWriter` operator with a file name ending in `.bin`.
- Run the workflow for some seconds.
- Open the resulting binary file in MATLAB/Python/R and make a time series plot of the raw waveform samples.
    - **MATLAB:** Use the `fread` function to read the binary file. The source data must be set to `int16`.
    - **Python:** Use the `fromfile` in the `numpy` package to read the binary file. The `dtype` option must be set to `np.int16`.

### Exercise 5: Trigger an auditory stimulus




- Insert an `AudioReader` source.
- Configure the `FileName` property to point to the audio file you recorded in *Exercise 3*.
- Insert an `AudioPlayback` sink.
- Run the workflow and check that the sound is played correctly.




- Insert a `KeyDown` source.
- Set the `BufferLength` property of the `AudioReader` to zero, so that all audio data is read into a single buffer.
- Combine the key press with the audio data using the `WithLatestFrom` combinator.
- Right-click the `WithLatestFrom` operator. Select the `Tuple` > `Item2` member from the context menu.
- Move the `AudioPlayback` sink so that it follows the selected `Item2` member.
- Run the workflow and press a key. What happens if you press the key several times?

## Arduino Acquisition

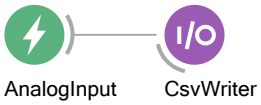
In order to communicate and interact with an Arduino using Bonsai, you must program the microcontroller to send and receive binary data from the computer via the USB cable.

Fortunately, the Arduino environment already comes with a standard implementation of an efficient binary protocol called **Firmata** which can be used for serial communication with external applications such as Bonsai.

## Configure Arduino for real-time communication

- Open the Arduino IDE.
- Upload `StandardFirmata` to your Arduino. The code can be found in `File` > `Examples` > `Firmata`.

## Exercise 6: Saving analog data




- Insert an `AnalogInput` source.
- Configure the `PortName` property to point to the correct serial port where the Arduino is connected.
- Run the workflow and visualize the output of the analog source. What do you see?
- **Optional:** Connect a sensor to the analog input pin, e.g. a potentiometer or a button.
- Insert a `CsvWriter` sink. This operator records input data into a text file.
- Configure the `FileName` property of the `CsvWriter` operator with a file name ending in `.csv`.
- Run the workflow, record some interesting signal, and then open the result text data file.

## Exercise 7: Control an LED


- Insert a `Boolean` source.
- Insert a `DigitalOutput` sink.
- Set the `Pin` property of the `DigitalOutput` operator to 13.
- Configure the `PortName` property.
- Run the workflow and change the `Value` property of the `Boolean` operator.
- **Optional:** Use your mouse to control the LED! Replace the `Boolean` operator by a `MouseMove` source (hint: use `GreaterThan`, `LessThan`, or equivalent operators to connect one of the mouse axis to `DigitalOutput`).

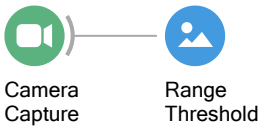
### Exercise 8: Control a servo motor


- Insert a `Timer` source. Set its `Period` property to 500 ms.
- Insert a `Take` operator. Set its `Count` property to 10.
- Insert a `Rescale` operator. Set its `Max` property to 10, and its `RangeMax` property to 180.
- Insert a `Repeat` operator.
- Insert a `ServoOutput` sink.
- Set the `Pin` property of the `ServoOutput` operator to 9.
- Configure the `PortName` property.
- Connect a servo motor to the Arduino pin 9 and run the workflow. Can you explain the behaviour of the servo?
- **Optional:** Make the servo sweep back and forth.

## Video Tracking

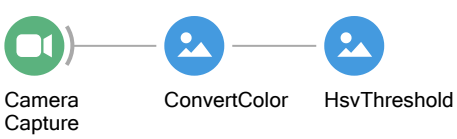
Bonsai allows processing captured raw video data to extract real-time measures of behaviour or other derived quantities. The exercises below will introduce you to some of its online video processing capabilities.

### Exercise 9: Segmentation of a coloured object




- Insert a `CameraCapture` source.
- Insert a `RangeThreshold` transform.
- Open the visualizer for the `RangeThreshold` operator.
- Configure the `Lower` and `Upper` properties of the `RangeThreshold` to isolate your coloured object (hint: click the small arrow to the left of each property to expand their individual values).

This method segments coloured objects by setting boundaries directly on the BGR colour space. This colour space is considered a poor choice for colour segmentation. Can you see why?

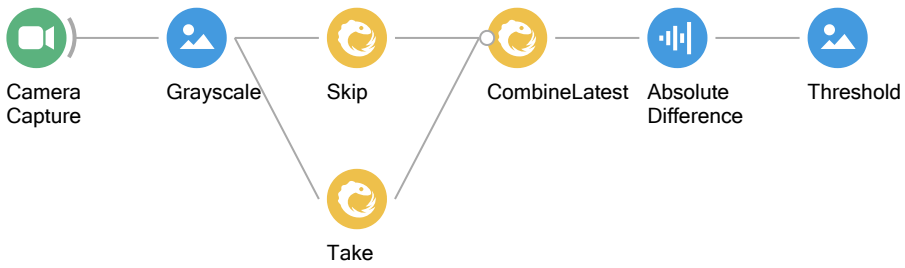


- Replace the `RangeThreshold` operator by a `ConvertColor` transform. This node converts the image from the BGR colour space to the Hue-Saturation-Value (HSV) colour space.
- Insert an `HsvThreshold` transform.
- Configure the `Lower` and `Upper` properties of the `HsvThreshold` to isolate the object.
- Test the resulting tracking under different illumination conditions.

## Exercise 10: Real-time position tracking


- Starting with the workflow from the previous exercise, insert a `FindContours` transform. This operator traces the contours of all the objects in a black-and-white image. An *object* is defined as a region of connected white pixels.
- Insert a `BinaryRegionAnalysis` transform. This node calculates the area, center of mass, and orientation for all the detected contours.
- Insert a `LargestBinaryRegion` transform to extract the largest detected object in the image.
- Select the `ConnectedComponent` > `Centroid` field of the largest binary region using the context menu.
- Record the position of the centroid using a `CsvWriter` sink.
- **Optional:** Open the CSV file in Excel/Python/MATLAB/R and plot the trajectory of the object.

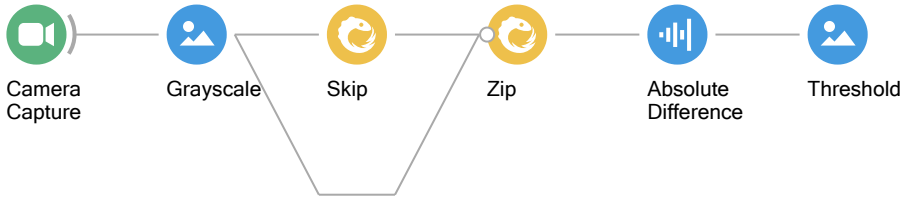
## Exercise 11: Background subtraction and motion segmentation




- Create a grayscale video workflow similar to *Exercise 2*.
- Insert a `Skip` operator. Set its `Count` property to 1.
- In a new branch, insert a `Take` operator. Set its `Count` property to 1.
- Combine the images from both branches using the `CombineLatest` combinator.
- Insert the `AbsoluteDifference` transform after `CombineLatest`.

- Insert a `Threshold` transform. Visualize the node output and adjust the `ThresholdValue` property.

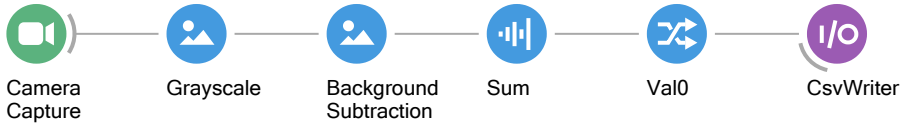
*Describe in your own words what the above workflow is doing.*




- Replace the `CombineLatest` operator with the `Zip` combinator.
- Delete the `Take` operator.

*Describe in your own words what the above modified workflow is doing.*

## Exercise 12: Measuring motion




- Create a grayscale video stream similar to *Exercise 2*.
- Insert a `BackgroundSubtraction` transform. Set its `AdaptationRate` property to 1.
- Insert a `Sum` operator. This operator will sum the values of all the pixels in the image.
- Run the workflow, point the camera at a moving object and visualize the output of the `Sum` operator. Compare small movements to big movements. What happens to the signal when the object holds perfect still?
- Right-click the `Sum` operator. Select the `Scalar` > `Val0` member from the context menu.

**Note:** The `Sum` operator sums the pixel values across all image colour channels. However, in the case of grayscale binary images, there is only one active channel and its sum is stored in the `Val0` field.

- Record the motion of an object using a `CsvWriter` sink.

---