Bonsai: An event-based framework for processing and controlling data streams

Gonçalo Lopes[1*], Niccolò Bonacchi[1], João Frazão[1], Joana Neto[1], Bassam Atallah[1], Sofia Soares[1], Luís Moreira[1], Sara Matias[1], Pavel M. Itskov[1], Patrícia Correia[1], Roberto Medina[1], Elena Dreosti[2], Joseph J. Paton[1], Adam R. Kampff[1]

[1] Champalimaud Neuroscience Programme, Champalimaud Centre for the Unknown, Lisbon, Portugal

[2] Department of Cell and Developmental Biology, University College London, Gower Street, London WC1E 6BT, UK

* Correspondence should be addressed to GL:

goncalo.lopes@neuro.fchampalimaud.org

Abstract

The design of modern scientific experiments requires the control and monitoring of many parallel data streams. However, the serial execution of programming instructions in a computer makes it a challenge to develop software that can deal with the asynchronous, parallel nature of scientific data. Here we present Bonsai, a modular, high-performance, open-source visual programming framework for processing data streams. We will describe Bonsai's core principles and architecture while specifically highlighting some of the applications that were developed during one year of use in an active neuroscience research institute.

Introduction

The design and implementation of modern scientific experiments depends crucially on the control and monitoring of many parallel streams of data. Multiple measurement devices ranging from video cameras, microphones and pressure sensors to neural electrodes, must simultaneously send their data in real-time to a recording system. General purpose digital computers have gradually replaced many of the specialized analog and digital technologies

used for this kind of data acquisition and experiment control, largely due to the flexibility of programming and the exponential growth in computing power. However, the serial nature of programming instructions and shared memory makes it a challenge, even for experienced programmers, to develop software that can elegantly deal with the asynchronous, parallel nature of scientific data.

Another challenge arises from the need for software integration. Each hardware vendor provides their own set of drivers and programming interfaces for configuring and acquiring data from their devices. Furthermore, the advent of the open-source software movement has greatly increased the number of freely available technologies for different data processing domains. Nonetheless, integration of all these diverse software components remains a major challenge for researchers.

These difficulties lead to increased development times in setting up an experiment. Moreover, it requires the experimenter to undergo specialized training outside his domain of research. This limits the ability to rapidly prototype and try out new designs and can quickly become a constraining factor in the kinds of questions that are amenable to scientific investigation.

In the following we describe Bonsai, an open-source visual programming framework for processing data streams. The main goal of Bonsai is to simplify the development of software for acquisition and processing of the many heterogeneous asynchronous data sources commonly used in (neuro)scientific research. The framework has already been successfully used in many applications. In this work we emphasize its use in neuroscience for monitoring and controlling behaviour and physiology experiments.

Architecture

Bonsai was developed on top of the Reactive Extensions for the .NET framework (Rx)[1]. Rx represents asynchronous data streams using the notion of an observable sequence. As the name implies, elements in an observable sequence follow one after the other. The name observable simply specifies that the way we access elements in the data stream is by listening to (i.e observing) the data as it arrives, in contrast with the static database model, in which the desired data is enumerated.

In Bonsai, these observable sequences are created and manipulated graphically using a dataflow[2,3] representation (Fig. 1a). Each node in the dataflow represents an observable sequence. Nodes can either be observable sources of data, if they receive no incoming

connections, or combinators, if they represent observable operators that combine one or more inputs. Combinators can be further specialized into transforms and sinks depending on how they manipulate an observable sequence. Transforms change only incoming data elements from a single input sequence. One example would be taking a sequence of numbers and producing another sequence of numbers containing every original number multiplied by two. Sinks, on the other hand, simply introduce processing side-effects without modifying the original sequence at all. One example would be printing each number in the sequence to a text file. The act of printing in itself changes nothing about the sequence, which continues to output every number, but the side-effect will generate some useful action. If a combinator is neither a transform nor a sink, it is simply called combinator. Taking two sequences of numbers and merging them together into a single sequence where elements are combined whenever both sequences produce a new value would be an example of a combinator which does not change the elements themselves but alters the flow of data downstream.

A common requirement when designing and manipulating dataflows is the ability to visualize the state of the data at different stages of processing. We have included a set of visualizers to assist debugging and inspection of data elements, including images and signal waveforms (Fig. 1b). These visualizers are automatically associated with the output data type of each node and can be launched at any time in parallel with the execution of the dataflow. Furthermore, it is often desirable to be able to manipulate processing parameters online for calibration purposes. Each node has a set of properties which parameterizes the operation of that particular source or combinator. This allows, for example, changing the cutoff frequency of a signal processing filter, or setting the name of the output file in the case of data recording sinks.

We have built into Bonsai the ability to group nodes hierarchically. In its simplest form, this feature can be used to encapsulate a set of operations into a single node which can later be reused elsewhere (Fig. 1c). This is similar to defining a function in a programming language and is one of the ways to create new reactive operators in Bonsai. Encapsulated nodes are also used to specify more complicated, yet powerful, operators such as iteration constructs which can be cumbersome to specify in pure dataflow visual languages[3]. Consider the example of specifying a sliding window algorithm (Fig. 1d). When using sliding windows, the original data sequence is sliced into a set of smaller overlapping sub-sequences, the windows. Some operation is then performed over the elements of each window, like computing its total sum. By merging the results of processing each window together, we would get a new sequence of elements. A concrete example would be computing the convolution of a data stream with a

kernel. In Bonsai, Rx provides operators that create these windows from any observable sequence. The specification of the operations to apply on each window is then represented by encapsulating a dataflow inside a SelectMany group. The input to this group will now be each of the sliding windows and the different outputs will be merged together to produce the final result.

Finally, we also included the possibility of externalizing node properties into the dataflow (Fig. 1e). Externalizing a property means pulling out one of the parameters of a specified node into its own node in the dataflow. This allows for two different possibilities. On the one hand, it becomes possible to connect the output of another node to the exposed property. This allows for the dynamic control of node parameters. On the other hand, if externalized properties are given a name and placed inside an encapsulated dataflow, they will show up as properties of the group node itself. This allows for the parameterization of nested dataflows and increases their reuse possibilities.

Bonsai was designed to be a modular framework, which means it is possible to extend its functionality by installing additional packages containing sources and combinators developed for specific purposes. New packages can be written by using C# or any of the .NET programming languages. IronPython scripts can be embedded in the dataflow as transforms and sinks, allowing for rapid integration of custom code. All functionality included in Bonsai was designed using these modular principles, and we hope to encourage other researchers to contribute their own packages and thereby extend the framework to other application domains. At present, the available packages include computer vision and signal processing modules based on the OpenCV[4] library as well as several device interfaces which provide the sources and sinks for Bonsai, including support for Arduino[5] microcontrollers, serial port devices and basic networking using the OSC[6] protocol. Given the specific applications in the domain of neuroscience, we also integrated a number of neuroscience technology packages. The Ephys package, for example, builds on the Open Ephys initiative for the sharing of electrophysiology acquisition hardware[7] by providing support for the Rhythm open-source USB/FPGA interface (Intan Technologies, US). This means the next generation tools for electrophysiology can be used inside Bonsai today and the acquired data integrated with all the other data streams in a common framework.

Results

The validation of Bonsai was performed by using the framework to implement a number of application use cases in the domain of neuroscience (Fig. 2). The breadth of technologies at use in this field demands that modern experiments be able to handle many heterogeneous

sources of data. Experimenters need to routinely record video and sensor data describing the behaviour of an animal simultaneously with electrophysiology, fluorescent reporters of neural activity or other physiological measures. Online manipulation and visualization of data is a fundamental part of the experiment protocol for many of the reported techniques. In the following we highlight some of these practical applications of Bonsai in more detail in order to illustrate both best practices and implementation challenges.

One of the first main use cases driving the development of Bonsai was the automated online tracking of animal behaviour using video. To this end we developed an integration with the open-source computer vision library OpenCV and exposed much of its functionality and data types as Bonsai combinators. Drivers for several cameras and other imaging devices were also integrated as Bonsai sources. One of the most common tracking applications involves chaining together operators for image segmentation and binary region analysis to allow the extraction of the spatial location of an animal over time (figs. 2a and 2b). The same technique can easily carry over to track different kinds of objects such as eyes or experimental manipulanda in human psychophysics experiments (fig. 2c) provided adequate illumination contrast and a choice of an appropriate method for segmentation. These image processing tools can also be used to acquire and process physiological data in neural imaging setups, where it is now possible to record bioluminescent or fluorescent reporters of neural activity during behaviour. For example in fig. 2b, measures of bulk fluorescence were recorded using a CCD sensor and a fiberoptic setup.

Raw video data from modern high-resolution, high-speed cameras can be expensive to store. Online video compression and storage sinks were implemented taking advantage of parallelism to avoid frame loss. Video compression and storage is time-consuming and can compromise data acquisition if reading the next frame has to wait for the previous frame to be fully encoded. One solution is to buffer incoming frames and compress them in parallel with the rest of the processing. By encapsulating this behaviour into a Bonsai sink, it becomes easy to incorporate video recording functionality into any image processing pipeline (figs. 2a-e, 2g-h).

While simple image processing techniques can easily extract continuous two-dimensional measures of animal location over time, it often becomes the case that the experimenter is concerned with tracking the detailed behaviour of specific features in the animal's body, such as the head orientation. Identifying such features and reconstructing their location in 3D space is a challenging computer vision problem. A common solution is to use planar fiducial markers of known geometry[8,9]. The computer vision research community has been releasing some open-

source software solutions to this problem, such as the recent ArUco library[8], which has now been integrated into Bonsai and used successfully for online tracking of 3D head movements (fig. 2d).

One final but important application of video stream processing is in the development of closed-loop interfaces, where the actions of an animal directly modulate the manipulations under the experimenter's control. This kind of experiments requires fast online analysis of behaviour variables of interest which are subsequently coupled to hardware control interfaces. In fig. 2e, a conditioned place preference assay was implemented by analyzing in real-time the position of an animal in a square arena. Whenever the animal found itself inside a specified region of interest, a signal was sent to an Arduino controller which was then used to drive optogenetic stimulation of specific brain circuits.

Another key data type that must be integrated and processed inside Bonsai dataflows is buffered time-series data. This type of data usually arises from audio, electrophysiology or other digital acquisition systems where multiple data samples, from one or more channels, are synchronously acquired, buffered and streamed to the computer. These buffers are often represented as data matrices, where rows are channels and columns represent individual data samples through time, or vice-versa. Support for simple band-pass filters, thresholding and triggering allowed us to build flexible spike detection and waveform extraction systems (fig. 2f). Using Intan's Rhythm API, we integrated into Bonsai support for a variety of next-generation electrophysiology devices using Intan's digital amplifier technology, such as the Open Ephys acquisition system[7] or Intan's evaluation board (RHD2000, Intan Technologies, US). This system was successfully used to acquire and visualize simultaneous recordings from dense silicon probes where spikes from a loose-patch juxtacellular pipette were used as triggers to align and extract waveform data appearing on the multi-channel extracellular probe. Responses from every silicon probe site can then be superimposed on an accurate rendition of the probe geometry in real-time.

The ability to rapidly integrate new modules allowed us to support the development and cross-validation of new tools for behavioural neuroscience. A paradigmatic example was flyPAD, a new method for quantifying feeding behaviour in Drosophila melanogaster by measuring changes in electrode capacitance induced by the proboscis extension of a fly[10]. The integration of flyPAD in Bonsai aimed to allow researchers to quickly get started using this approach to setup new experiments. Furthermore, it also allowed the validation of the tool by enabling

simultaneous acquisition of high-speed video recordings of fly behaviour which were later used for annotation and classification of the sensor feeding traces (fig. 2g).

One of the most interesting use cases we tackled was a variation on the popular two-alternative forced choice (2AFC) decision-making task for rodents (fig. 2h). In this family of tasks, animals are placed in an environment with three “ports”. They are presented with a stimulus in the center port and afterwards report their perception of the stimulus by going either to the left or right choice ports. In the variation we present in this work, the two choice ports were replaced by regions of interest where the activity of the animal is analyzed using computer vision. This example offered unique challenges as it combined sophisticated sequential control of a task environment with continuous data stream processing of video and sensor data.

One of the classical approaches to model these kinds of environments in behavioural neuroscience is to use a finite state machine formalism (fig. 3a). In this framework, nodes represent states, e.g. reward availability or reward delivery, and edges represent transitions between states that are caused by events in the assay, e.g. a nose poke. Often when using state machines for control, each state represents different modes of operation. For example, a state machine for controlling an automatic door may have two different modes of operation, one for opening the door and another one for closing it. Inside each of these discrete modes of operation, different dynamical variables may be updated using different rules. For example, the dynamics of closing the door may be gentler than the dynamics of opening it, to avoid the risk of accidents.

In a dataflow model, nodes consume and produce event sequences. We can model states in a dataflow by representing transitions into the state as the input sequence to a node (i.e. the state becomes active whenever some input arrives), and transitions out from the state as the output sequence (i.e. an output is generated whenever the state terminates). This representation implies two main differences from the traditional model. First, there is no strict decoupling between states and events. Each state is responsible for handling its own transition events, and to decide when it starts and terminates. Second, because transition events move along every edge in the dataflow, we can easily allow for parallel states simply by branching the output. The mutually exclusive conditional transitions usually found in a state machine can be implemented by filtering out unwanted events from specific branches.

Furthermore, it becomes possible to specify dynamic control structures inside each state. By using the SelectMany operator as mentioned in the previous section, we can specify an arbitrary

dataflow to process each state. Specifically, whenever an event arrives into the state a new observable sequence will be created and started, representing the processing and control dynamics that govern that state. This sequence can initiate other event sources like a camera, for example, in order to continuously process data while the state is active. As soon as the state terminates, a signal would be sent as output to, potentially, trigger the activation of other states.

Discussion

After about a year of using Bonsai in an active neuroscience research institute, dozens of different experimental protocols and data analysis pipelines have been successfully implemented using a subset of the provided building blocks. We were surprised by the diversity of applications and by the pace at which new modules and devices were developed and integrated.

The performance achieved by Bonsai dataflow processing was an important consideration throughout. Video processing can be particularly challenging to handle given the bandwidth required to quickly acquire and process large data matrices. In order to correlate continuous measures of behaviour with neural activity, it is useful for those measurements to have both high spatial and high temporal resolution. Using Bonsai we were able to routinely process grayscale images from high sensor resolutions (1280x960) and frame rate (120 Hz) using standard off-the-shelf desktop computers. In fact, many of the reported assays use multiple such video streams with success and actually process the behaviour video online either to control states of the behaviour protocol or to preprocess video data for offline analysis.

One of the areas where we see the application of Bonsai becoming most significant is in the development of dynamic behaviour assays (environments) using reactive control strategies. Brains evolved to generate and control behaviours that can deal with the complexity of the natural world. Unfortunately, when neuroscientists try to investigate these behaviours in the lab, they are often constrained by available technology in their efforts to reproduce equivalent environmental complexity in a controlled manner. As an example, consider a simple foraging scenario in which a land animal must collect, in a timely manner, food items that become available at random intervals in many sites. If the item is not collected in time, it rots or gets eaten by other competitors. In the case of a single foraging site, a finite state machine description intuitively represents the workings of the environment (fig. 3a). Let us now consider a situation where the environment has two of these food sites operating independently. Fig. 3b shows one possible, non-exhaustive model of such an environment. In the classical state

machine formalism the machine can only be in one state at a time, which means that we now need to model this state as the combination of the individual states at each reward location. Furthermore, because transitions between these states are asynchronous and independent, we get edges in between nearly every pair of nodes, as each reward site can change its state at any point in time relative to the other.

In order to make the design space more intuitive we need to introduce the parallel and asynchronous nature of the real-world situation into our modeling formalism. One simple example of this idea for the 1-site foraging task is depicted in fig. 3c. In this case, we have two sources of events from the environment: one timer signaling the availability of reward (A); and a sampling event (S) which happens every time the animal checks the location for food. Both of these events can occur pretty much independently of each other, but when a sampling event coincides with reward availability (C), then reward (R) is delivered. Because this description is intrinsically asynchronous, it makes it extremely easy to scale the task to a larger set of locations: just replicate the dataflow for each of the other locations (fig. 3d).

Another difficulty of the classical state machine formalism is dealing with continuous variables. The natural environment provides constant real-time feedback that can tightly correlate to the actions of an animal. Reproducing such closed-loop interactions and manipulating their dynamics can be a valuable tool for investigating brain function. Such models are virtually impossible to represent in a machine of finite states, precisely due to the potential infinitude of the feedback responses. However, the dataflow formalism of asynchronous event sources can easily accommodate such models. In fact, this is their natural battleground, where nodes represent reactive operators that promptly respond to input values broadcasted by event sources. These models of asynchronous computation appear then well suited to subsume the complexities of both discrete and continuous aspects of the natural environments brains evolved to deal with.

Acknowledgments

We thank Joana Nogueira, George Dimitriadis and all the members of the Intelligent Systems Laboratory for helpful discussions and comments on the manuscript. We also thank all the members of the Champalimaud Neuroscience Programme who used Bonsai to setup their data analysis and acquisition experiments and in so doing provided valuable feedback to improve the framework. This project was supported by the NeuroSeeker Project Grant (FP7-ICT-2011-9:600925) and the Bial Foundation. GL is supported by the PhD Studentship

SFRH/BD/51714/2011 from the Foundation for Science and Technology. The Champalimaud Neuroscience Programme is supported by the Champalimaud Foundation.

References

1. Microsoft Open Technologies. Reactive Extensions for the .NET framework (Rx). (2014). at <https://rx.codeplex.com/>

2. Johnston, W. M., Hanna, J. R. P. & Millar, R. J. Advances in dataflow programming languages. *ACM Comput. Surv.* **36,** 1–34 (2004).

3. Mosconi, M. & Porta, M. Iteration constructs in data-flow visual programming languages. *Comput. Lang.* **26,** 67–104 (2000).

4. Itseez. OpenCV (Open Source Computer Vision). (2014). at <http://opencv.org/>

5. Banzi, M., Cuartielles, D., Igoe, T., Martino, G. & Mellis, D. Arduino. (2014). at <http://www.arduino.cc/>

6. Wright, M., Freed, A. & Momeni, A. Open Sound Control: State of the Art 2003. in *Proc. 2003 Conf. New Interfaces Music. Expr.* 153–159 (2003).

7. Siegle, J. *et al. Open Ephys.* **In preparation,** (2014).

8. Garrido-Jurado, S., Munõz-Salinas, R., Madrid-Cuevas, F. J. & Marín-Jiménez, M. J. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.* (2014).

9. Kato, H. & Billinghurst, M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. in *Augment. Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE ACM Int. Work.* 85–94 (IEEE, 1999).

10. Itskov, P. M. *et al. Automated monitoring and quantitative analysis of feeding behavior in Drosophila*. *Nat. Commun.* **In press,** (2014).

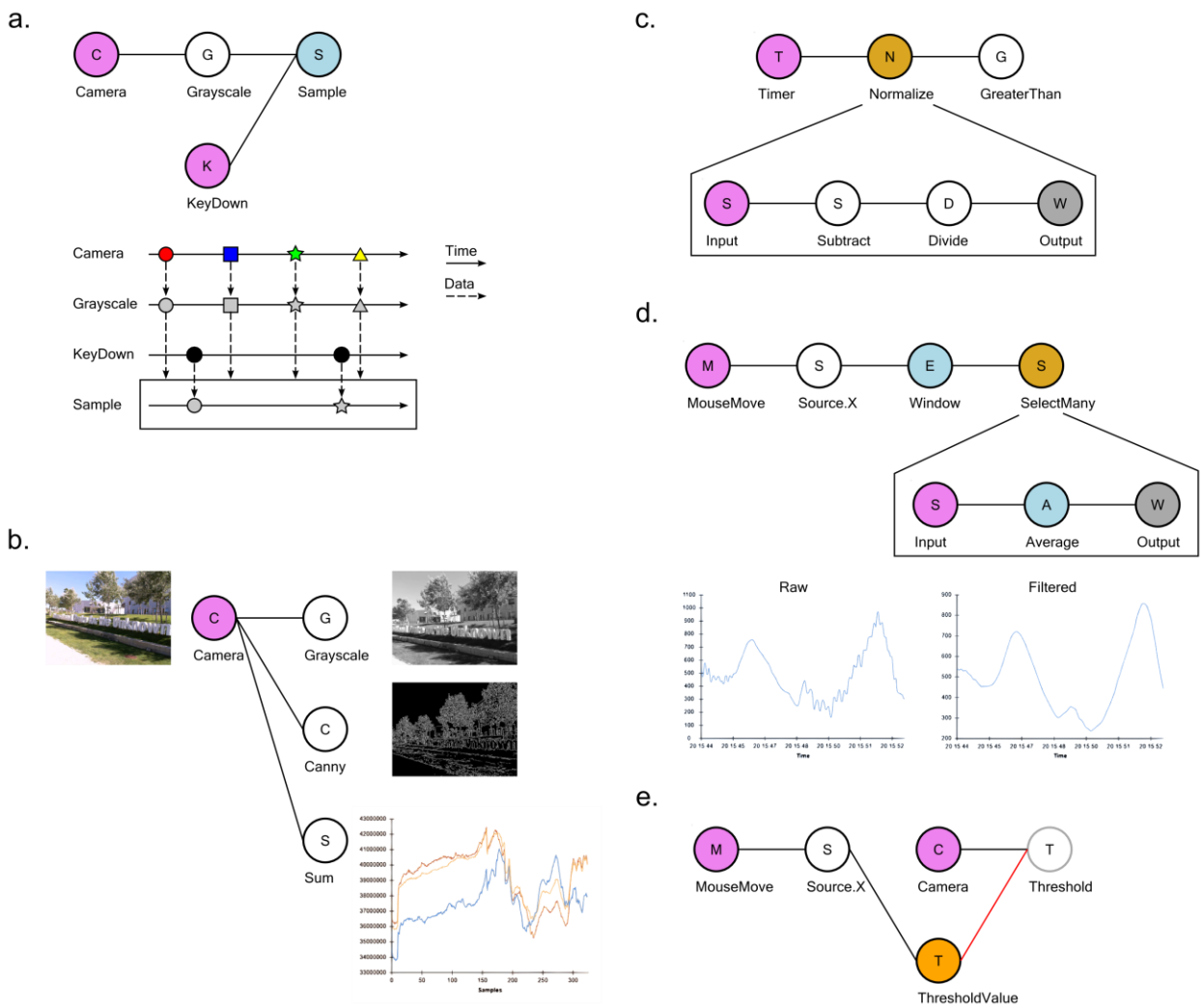
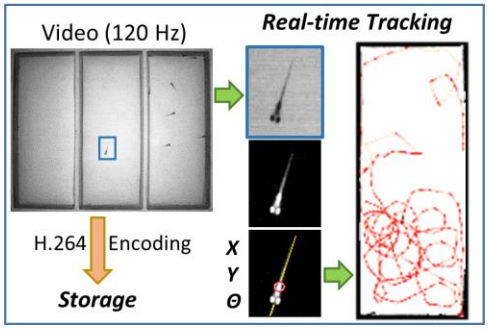


Figure 1: Examples of dataflow processing pipelines using Bonsai

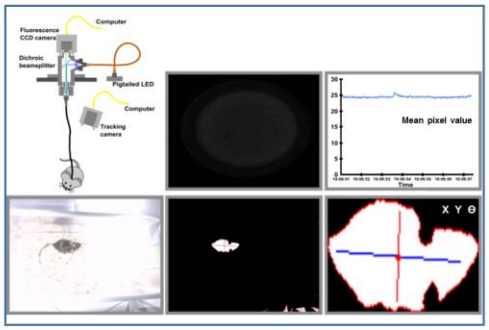
a. Taking grayscale snapshots from a camera whenever a key is pressed. Top: graphical representation of the Bonsai dataflow for camera and keyboard processing. Data sources are colored in violet; transform operators in white; combinators in light blue. Bottom: marble diagram showing an example execution of the dataflow. Colored tokens represent frames arriving from the camera. Black circles represent key press events from the keyboard. **b.** Visualizing image processing filters. The inset next to each node represents the corresponding Bonsai data visualizer. **c.** Grouping a set of complex transformations into a single node. In the nested dataflow, the source represents incoming connections to the group and the sink represents the data output. **d.** Low-pass filtering of mouse movements. The Window combinator is used to build sliding windows of the raw mouse input. The SelectMany combinator is used to compute the average of each window sequence. The results are merged together to produce the

resulting filtered time series. **e.** Dynamic modulation of an image processing threshold using the mouse. The x-coordinate of mouse movements is used to directly set the externalized ThresholdValue property (orange). The updated threshold value will be used to process any new incoming images.

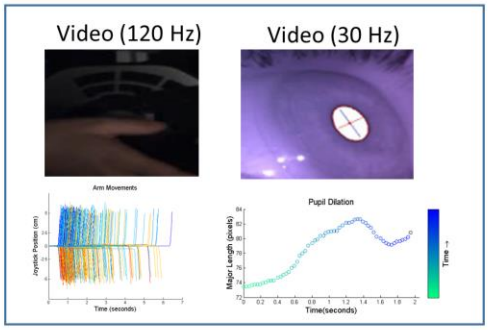
a. High-Speed Tracking of Zebrafish Behaviour



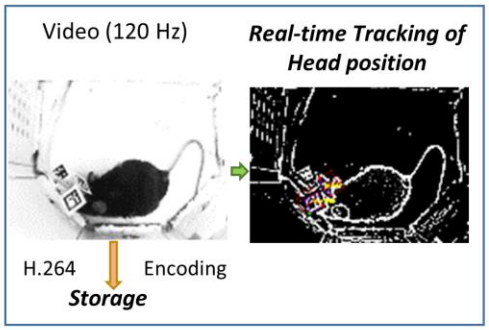
b. Mouse tracking and bulk fluorescence measurement of neuronal calcium activity



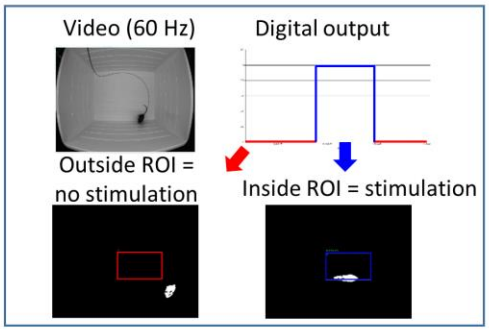
c. Tracking Human Behaviour During a Stochastic Sound Discrimination Task



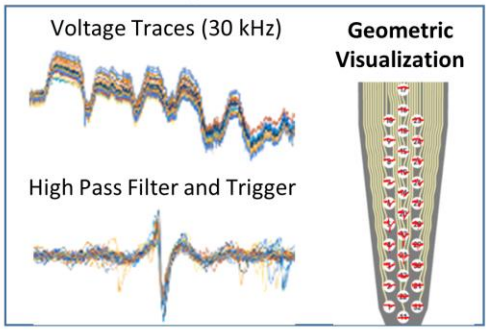
d. 3D Tracking of Rodent Head Position



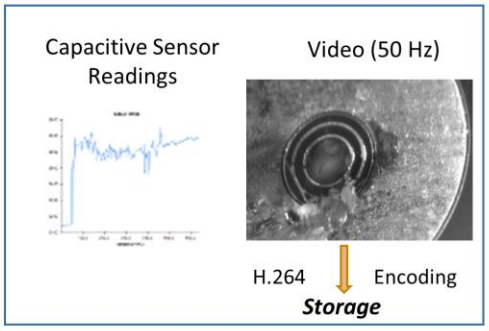
e. Real-time stimulation conditioned to a region in space



f. Acute Recordings from Dense Silicon Probes



g. Recording Drosophila Feeding Behaviour



h. 2AFC Task using Video Triggered Reward

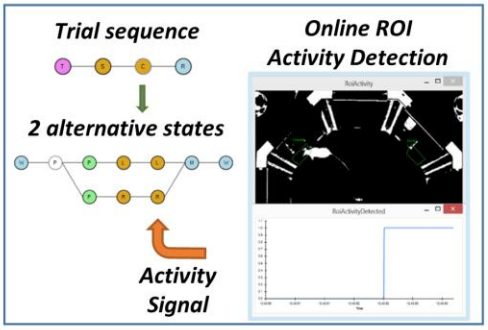


Figure 2: Example use cases of neuroscience experimental setups using Bonsai

a. High-speed tracking of zebrafish behaviour. **b.** Mouse tracking and bulk fluorescence measurement of neuronal calcium activity. **c.** Tracking human behaviour during a stochastic sound discrimination task. **d.** 3D tracking of rodent head position. **e.** Real-time stimulation conditioned to a region in space. **f.** Acute recordings from dense silicon probes. **g.** Recording drosophila feeding behaviour. **h.** 2AFC task using video triggered reward.

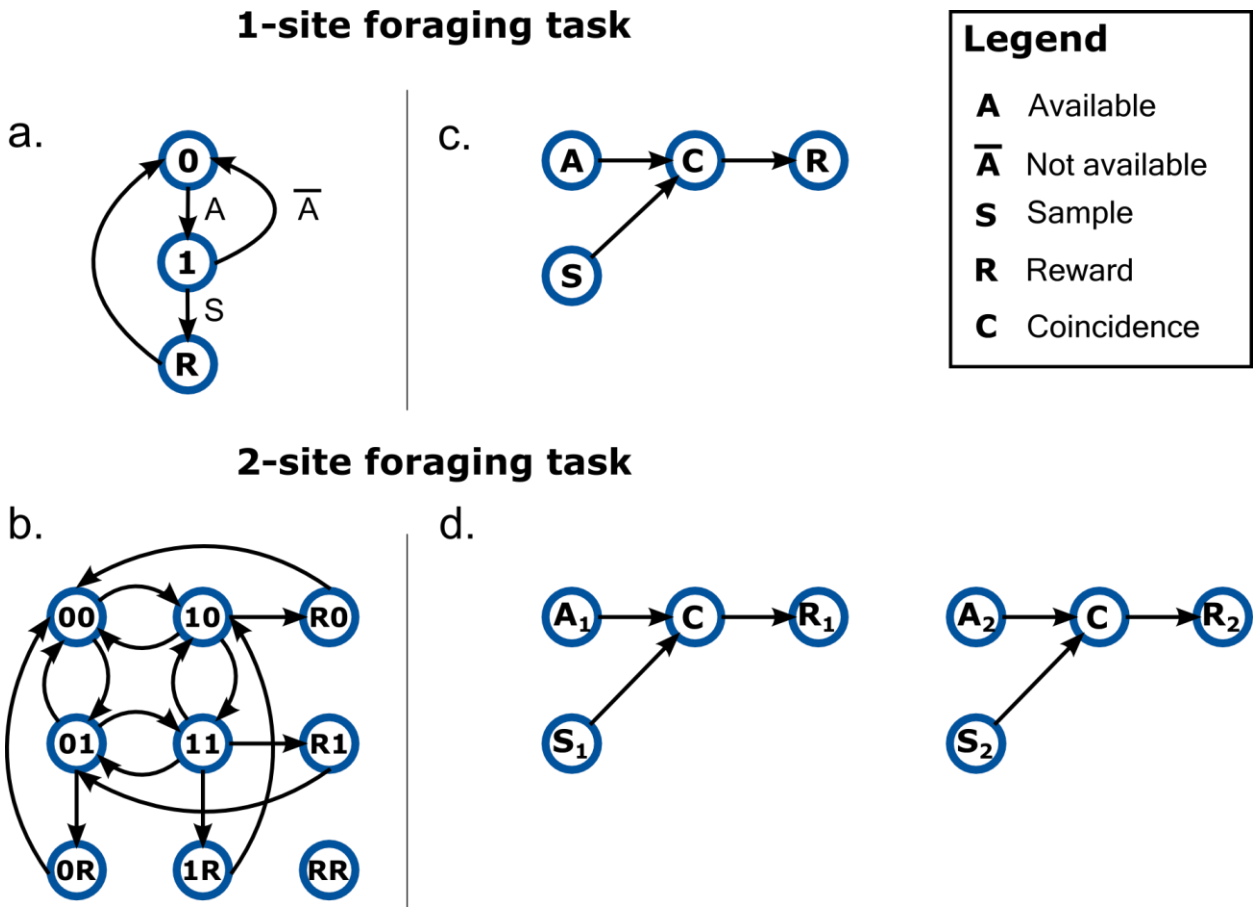


Figure 3: Describing the behaviour of dynamic environments using either state-machines or dataflows

a. A state-machine model of the 1-site foraging task. Zero indicates non-availability of reward at the site. One indicates reward is now available at the site. Labels on edges indicate event transitions. **b.** A non-exhaustive state-machine model for a foraging task with two sites. The active state is now a combination of the state of the two sites (indicated by a two character label) and all possible state combinations are tiled across the model. Event labels are omitted

for clarity. Notation is otherwise kept. **c.** A dataflow model of the 1-site foraging task. Events in the state-machine model are now modelled as data sources. The coincidence detector node (C) propagates a signal only when the sample event closely follows reward availability. **d.** A dataflow model for a foraging task with two sites. The number subscripts denote foraging site index.

Supplementary Materials and Methods

The Bonsai framework can be freely downloaded from https://bitbucket.org/horizongir/bonsai.

All experiments were approved by the Champalimaud Foundation Bioethics Committee and the Portuguese National Authority for Animal Health, Direcção-Geral de Alimentação e Veterinária (DGAV).

Zebrafish

Larval zebrafish (6 dpf) were filmed with a high-speed monochrome video camera (Flea3, Point Grey, CA) under IR illumination. Fish swam freely in a custom-built arena that was laser cut from transparent acrylic that consisted of 3 separate chambers, each 40 x 100 mm. The position and orientation of the zebrafish in the central chamber was continuously tracked in real-time, while the video of the entire arena (1.3 Megapixel) was compressed to a high-quality H.264 encoded file that was used for subsequent offline analysis of the behaviour of a group of zebrafish placed in either of the side chambers.

Fluoro

Freely behaving mice were filmed with a video camera (PlayStation Eye, Sony, JP) under white light illumination in their own homecages. A fiberoptic setup was developed to monitor bulk fluorescence changes in specific neuron populations using genetically encoded calcium indicators. Changes in fluorescence caused by neuronal activity were transmitted by an optical fiber and recorded with a CCD camera (Pike, Allied Vision Technologies, DE). The position and orientation of the mice was continuously tracked in real-time, while the mean pixel value of the area of the camera facing the fiber optic was continuously calculated. Both videos were compressed to high-quality H.264 encoded files to be used in subsequent offline analysis.

Human

Bonsai was used to acquire timestamped images from 3 cameras (eye, person's view, and arm) simultaneously. The videos are synced with presented sound stimulus by using the arm camera to also capture 2 IR LEDs which are turned on at sound on-set and off-set, respectively. The

arm is tracked by using an IR LED mounted on a joystick and processing the video online at 120 Hz to minimize noise from compression. All the videos are compressed to a MP4 encoded file for offline analysis of the eye movements, pupil dilation, and syncing of all events with the sounds. The eye videos are captured at 30 Hz using the IR illuminated pupil headset (https://code.google.com/p/pupil/).

Marker Tracking

Adult mice performing a two alternative forced choice task were filmed with a high-speed monochrome video camera (Flea3, Point Grey, CA). A fiber optic cable was attached to the mouse's head. The 3D position and orientation of the head was tracked in real-time using square fiducial markers from the ArUco tracking library (Munõz-Salinas, 2014). The video (0.24 Megapixel) was simultaneously compressed to a high-quality H.264 encoded file that was used for subsequent offline analysis of the behaviour.

Real-time stimulation

Black mice were recorded with a high speed video camera (Flea3, Point Grey, CA), while exploring an open field arena (50x40 cm, LxW), under white illumination (~250 lux). The x and y position, body centroid and orientation of the animal in the arena was continuously tracked in real-time. Mice were implanted with an optical fiber connected to a laser, in order to receive photostimulation with blue light. A region of interest (ROI, 13x10.5 cm, LxW) was defined and a python script was written to outline the conditioning protocol. A digital output signal was sent to a microcontroller board (Uno, Arduino, IT), each time the body centroid of the animal entered in the ROI, producing photostimulation. All data for the animal tracking and digital output was saved in a .csv file, as well as the video, for subsequent offline analysis of the behaviour.

Silicon Probes

Recordings of spontaneous neural activity in motor cortex were performed in anesthetized rodents by means of silicon probes comprising a dense electrode array (A1x32-Poly3-5mm-25s-177-CM32, Neuronexus, US). An open-source electrophysiology acquisition board (Open Ephys) was used along with a RHD2000 series digital electrophysiology interface chip that filters, amplifies, and digitally multiplexes 32 electrode channels (Intan Technologies, US). Extracellular signals sampled at 30 kHz with 16-bit resolution in a frequency band from 0.1 to 7500 Hz were saved into a raw binary format for subsequent offline analysis. Online analysis of neural spike waveforms across all probe channels was performed by aligning the multi-channel raw data on spike events from a selected channel of interest. A custom Bonsai visualizer was written using OpenGL to display all channel traces superimposed on the geometric arrangement

of probe sites. It was possible to examine the details of extracellular activity in the spatial distribution.

Drosophila Feeding

Individual Drosophila melanogaster flies were allowed to freely feed on the flyPAD (Itskov, 2014) while their feeding behaviour was monitored at 50 Hz with a video camera (Flea3, Point Grey, CA) mounted on a Zeiss Discovery v.12 Stereo Microscope (Carl Zeiss, DE). FlyPAD measures fly`s behaviour on the food source by recording the capacitance at 100 Hz between the electrode on which the fly stands and the food. Videos were compressed to high-quality H.264 encoded files and subsequently manually annotated by a human observer to be used as a benchmark for the development of the automatic algorithms for the extraction of feeding behaviour from the capacitive trace.

2AFC Mice

Adult PWD female mice were tested in behavioural experiments using restricted social interaction with adult C57BL6 and PWK males as reward. The behavioural paradigm consists of a custom built arena made of modular acrylic pieces assembled in an aluminum frame. The contact zone between the female and the male (composed of four holes with r = 0.5 cm) was either available for a fixed period of time, or physically restricted by a vertically sliding door controlled by a servomotor. Subjects initiated the interaction by nose-poking in an infrared beam port that would trigger the opening of the door and subsequent availability of the contact zone. Videos were recorded using high-speed monochrome video cameras (Flea3, Point Grey, CA). Performance, monitoring and control of the behaviour box was done using a Motoruino board (Motoruino, Artica, PT) and custom Bonsai scripts.