BrainOS: Software for Everyday Robots

BrainOS is a software system from Brain Corporation addressing the technical\(^1\) and business\(^2\) challenges surrounding everyday robots. “Everyday robots” is a catchall phrase for systems that interact with us in everyday life with a degree of intelligence that relieves us of the need to attend to them frequently: it suggests a vast but still poorly defined set of applications and markets, each with its challenges and special requirements, that we might today variously call consumer robots, service robots, internet of things, security systems, drones, personal automation,\(^3\) etc. In spite of this diversity, all of these applications have the common requirement that they need a central software framework on which to develop and deliver solutions that (1) supports critical technologies like computer vision, machine learning, and motor control, (2) interfaces to common peripherals like cameras, IMUs, GPS, etc., and (3) runs efficiently on distributed, heterogeneous computing hardware. BrainOS is that central software framework. As described in more detail below, BrainOS consists of two components: a message passing framework (BrainOS Core) and collections of algorithms for perception, navigation, manipulation, control and learning (BrainOS Library).

BrainOS targets the commercialization of everyday robots. It employs both open-source and closed-source, proprietary components with the goal of making systems that employ the best available technology, are easily developed, are actively maintained, and are financially rewarding to those who use BrainOS in their products and to those who contribute components to it. BrainOS is lightweight, efficient, and intended for production use.

BrainOS is available in two versions

- **BrainOS** addresses the interests of hobbyists, makers, educators, students and early stage commercial developers. It offers easy development of new projects using algorithms included within the BrainOS libraries and / or algorithms developed during the project. It runs on popular low-cost computing hardware like the Raspberry Pi 2 and the Snapdragon 410c. It interfaces easily with Arduino controllers and components. BrainOS is available for free for evaluation and non-commercial development by download from our website. When your project is ready for commercialization, it is easy to obtain a BrainOS distribution license from Brain Corporation and/or upgrade to BrainOS Pro for more advanced development.

- **BrainOS Pro** addresses the interests of later stage commercial developers with challenging products and applications. In addition to the features of BrainOS, BrainOS Pro includes addition library elements from Brain Corporation, third party libraries, additional development tools and direct support from Brain Corporation. An alpha version of BrainOS Pro is available on Brain Corporation’s bStem single board computer and hardware development kit and other platforms by request. Please contact Brain Corporation for more information on BrainOS Pro and working with Brain Corporation to develop your product.

\(^1\) see our white paper “Everyday Robots: Technical Challenges”  
\(^2\) see our white paper “Everyday Robots: Business Challenges”  
\(^3\) see our white paper “Personal Automation: Living Inside the Robot”
Applications with BrainOS

BrainOS Applications can be understood in terms of a software/hardware “stack” as illustrated in the adjacent diagram. At the very top of the BrainOS Application stack, the User Interface layer allows the end-user to operate the robot or IoT device. Depending on the application, the User Interface could be a smartphone, a speech recognition system, or buttons and controls on the device. The next level of the application stack is BrainOS, which contains both BrainOS Core and BrainOS Library. **BrainOS Core** is a message-passing framework that is optimized for integrating machine learning algorithms with perception and control. It serves as a high-level application layer between application/robot-specific user interfaces and underlying algorithms and application libraries. **BrainOS Library** consists of algorithms that can be assembled to create applications. These perform most of the computation, carrying out tasks like face recognition, object categorization, visual tracking, machine learning, and the controlling of robot motors, switches, IoT devices, etc. These algorithms are implemented through function calls to the Application Libraries. Application Libraries provide basic numerical, graphical, and computational routines which can be used in many algorithms and applications. Some of the Application Libraries are based on hardware-specific offerings by chipset companies, like FastCV from Qualcomm, Inc. The lower levels in the stack pertain to the operating system of the base compute hardware. BrainOS base compute hardware typically runs the Linux Operating System, which excels at soft and hard real-time execution with low overhead. The Linux kernel manages memory allocation, multithreaded processes, and access to compute hardware resources. The final level of the stack includes the compute hardware and the robotic or IoT hardware.

How BrainOS Applications Work

BrainOS applications consist of software nodes interconnected through the BrainOS Core message passing framework. Nodes can be loosely organized into categories depending on their capabilities and how they are used. General categories of nodes include: Feature Extractors, Predictors, Controllers, Combiners (signal mixers), Web Servers, Media Streaming Servers, among others. These nodes are instantiated, configured, and linked together via BrainOS Core. The BrainOS architecture allows complex control software to be broken down into simpler, reusable modules and efficiently mapped to hardware.
One way to use BrainOS is to arrange nodes in a way that supports supervised learning. This can be useful, for example, if the application benefits from learning or parametrization of a robot or IoT device at the point of use by the purchaser of the device. Supervised learning might also be useful if the application developer desires to add functionality into the device that might be easier or less costly to teach rather than program.

**BrainOS Application for Supervised Learning in a Robot.** Illustration of a running BrainOS Application in the context of a robotic application where the robot is being trained to fetch a ball.

The general form of a supervised learning configuration is shown in the above figure. Here, a *Feature Extractor* node is configured to take high dimensional input from sensors (like cameras) and reduce the dimensionality by finding meaningful features for the application. For example, the Feature Extractor might scan for visually salient objects that stand out from the background in a scene. A *Predictor* node tries to discover relationships between the output of the Feature Extractor and the desired behavior of the robot or application, which is usually an even lower dimensional signal (e.g., turning on an IoT device, or moving a motor). For example, a Predictor might learn whether the desired behavior depends on the number of salient objects, on their location, or on their pattern of movement. Finally, a *Combiner* node determines how control from a human user is combined with the output of the Predictor and fed back into the system for learning. Whether a particular behavior can be learned -- as well as how quickly and precisely -- will depend on the particular combination of Feature Extractor, Predictor, and Combiner nodes used in BrainOS Core.
Example Use Case of Supervised Learning in IoT. The “Home Animator” Demo Kit is an example of BrainOS Application showcasing supervised learning in the Internet-of-Things (IoT) space. The selected Feature Extractor is sensitive to visual change, and the system is configured to control Belkin WeMo switches that have lights plugged in to them. The result is a BrainOS Application that allows gestures or other visual changes to trigger the connected lights to turn on and off.

Example Use Cases involving Perception, Machine Learning and Action
In many applications, learning is important because the perceptual (e.g. visual) aspects of the environment and the specific resulting behaviors desired by the user cannot be fully specified in the factory. BrainOS addresses this challenge by enabling the system to learn and, therefore, adapt to the specifics of the end-users application and environment. Here are a few examples.

- A BrainOS enabled robotic arm is trained to keep tools sorted on a workbench in a garage or workshop.
- A BrainOS enabled home automation system is trained to turn on a reading light when a person sitting in a particular chair or lying in a bed opens a book. The system turns off the light when the book is closed (see figure) or when the person falls asleep.
- A BrainOS enabled security and monitoring system is trained to generate one kind of alert when people enter a store, but a different kind of alert when people leave the store.

Cloud / Distributed Applications Welcome
Nodes in a BrainOS application do not need to be hosted on the same compute hardware to work together. BrainOS Core allows for interacting nodes to be seamlessly distributed with ease across multiple computational devices. This is important for use cases that call for inexpensive deployed hardware with limited computational capacity, but also need computationally intensive perception (e.g., deep networks) or a complex control algorithm.
BrainOS Technical Details

Integrating BrainOS into your Application

Most BrainOS application developers will write code at two levels: the User Interface layer and the Robot/Robot Driver layer (application developer focus is symbolized by the “hammer” icons in the figure to the right.) At the User Interface layer, application developers write custom interfaces that connect their end-user to their robot, IoT or similar application. The user interface allows the end-user (1) to control and/or train the application, (2) to send commands which switch the state of the application, e.g., between learning and autonomous operation, and (3) to see information about the internal state of the application. The User Interface communicates with BrainOS using industry standard protocols like REST calls, UDP, and/or Websockets.

The User API supports RESTful HTTP calls for creating, loading, saving, and configuring behaviors. The User API also supports teaching supervised learning algorithms via HTTP and UDP. For applications that cannot support local computation, BrainOS Core supports remote offloading of computing nodes (to the cloud, or to a nearby hub device).

Application developers wishing to use algorithms that are not in the BrainOS Library have a few options: they can hire Brain Corporation’s Technical Services Division to create the algorithms, find algorithms from the open source community, or implement their own algorithms. (This latter optional role is represented in the figure by the faded “hammer” icon in the figure.) Algorithm developers can keep their own algorithms as proprietary code, or choose to make the algorithms available in a variety of ways -- including marketing them through Brain Corporation. In such cases, Brain Corporation can provide services to validate the algorithms and test their compatibility with other algorithms in the library.
At the Robot layer, application developers also implement (or more often customize) the necessary drivers to connect BrainOS to their specific robot or application. Communication between BrainOS and the Robot can use any protocol that the application developer desires, including TCP/IP, UDP, or even infrared communication if that is supported by their hardware platform.

Finally, for any application the developer must configure BrainOS with an appropriate selection of computationally focused nodes such as Feature Extractors, Predictors, and Combiners. Brain Corporation’s Technical Services Division is available to support application developers with such decisions and to provide custom BrainOS configurations to meet their unique requirements.

**BrainOS Core**

Features of the BrainOS Core include the following

- **Development.** BrainOS Core makes it easy for large teams to collaborate, for third parties to integrate their own software, to divide the function of a complicated system into understandable parts, and to distribute processing across computing hardware.

- **Simplicity.** BrainOS Core emphasizes simplicity, modularity, asynchrony, language independence, debuggability, and cloud support. BrainOS Core’s focus on synchronous message passing helps ensure modularity, efficiency, and maximal reusability of components.

- **Programming Model.** The BrainOS Core computational model is that of dataflow through a directed graph of nodes. As such, BrainOS Core can support a wide array of programming models and learning regimes, depending on how nodes are structured and connected. For example, BrainOS Core supports Supervised Learning and Reinforcement Learning algorithms. Despite differences in the graph and data content, from the "OS perspective", distinct architectures are treated in the same way.

- **Configuration.** Developing applications with BrainOS Core is rapid because once implemented, nodes can be easily instantiated and interconnected by simply editing a JSON configuration file describing the graph.

- **Distributed.** BrainOS Core allows for multiple nodes to share a single runtime process. The result is that co-located nodes can share address space and take advantage of zero-copy messaging. The advantage is that message passing becomes as fast as making function calls, resulting in very high performance if nodes are operating within the same instance of BrainOS Core.

- **Dynamic.** BrainOS Core supports the creation and deletion of nodes from the graph at run-time.

- **Multilingual.** Write your nodes in C++, Python, or Objective-C (contact us for information on support for other languages).

- **Virtual.** Simulate your robots or devices in a virtual world (coming soon).
BrainOS Library
The following nodes will be available in the initial, free version of BrainOS. Expect that this library will be constantly updated and expanded.

Hardware Interface Nodes
- **Universal Camera Input.** Camera node that works with a variety of camera boards, either via OpenCV/USB or via MIPI interface. At startup this node scans for all available cameras connected to the board. (Priority is given to USB cameras because the user could simply unplug it if they don’t want to use that camera.)
- **Rover Robot.** Controls 4-wheeled mobile robots, w/optional gripper, via Arduino.
- **WeMo Switches.** Controls WeMo switches on the local network via TCP/IP or hotspot mode.

Virtual Robot Nodes *(experiment with/learn about BrainOS without robot hardware)*
- **Virtual Light Bulbs.** Home animator demo on the computer screen.
- **Virtual Turtle Robot.** A virtual 2D robot on the computer screen, like ROS’ “turtle”.

Feature Extractor Nodes
- **Frame Difference.** Returns the difference between the current video frame and a frame one second ago. Useful for applications depending on visual change. On its own, is sensitive to color and lighting. Open Source.
- **Color Histogram Tracker.** Returns a heat map, bounding box, or center coordinates of colors falling into a range. Open Source.
- **Two-Color Tracker.** Returns location and orientation of an object based on pairs of color stickers places on the object. Open Source.
- **Saliency Map Algorithms.** Detects salient objects on the floor or against backgrounds. Used in Mantis. Closed Source.
- **Background Averaging Difference.** Returns a person’s silhouette against a background, insensitive to color and lighting. Can be combined with One-Sec Frame Diff to yield sensitivity to change and directional motion (like “gestures”). Closed Source.
- **Saliency Tracker.** Can be primed on an object with a combination of colors; relatively robust tracking of that combination of colors across lighting conditions. Closed Source.

Machine Learning Nodes
- **Discrete KNN Teachable Predictor Module** for learning in Home Animator. Closed Source.
- **Multilayer Perceptron**
- **Deep Convolutional Network**
User Interface Nodes

- HTML5 Webserver for Home Animator.
- iOS Joystick App for Mantis robot control (e.g., telepresence, training Navigation Predictor).

Working with Brain Corporation

Brain Corporation provides technical services to complement and expand the offerings within BrainOS and BrainOS Pro. If your company wants to develop complex cyber-physical applications, our experts can examine the details of your use cases and provide recommendations on available or emerging technologies. We can then create a proposal to select appropriate algorithms, develop prototypes, and assist in integrating advanced technologies like vision and machine learning into your product. Depending on the complexity of our customers’ needs, we have accommodated short-term, medium-term and long term contracts spanning weeks, months or even multiple years in application domains like robotics, home automation, IoT, security and vision systems. Please contact us for more information or to initiate a discussion about how we might assist you.