TWO YEARS OF THE THOUSAND BRAINS PROJECT: WHAT WE ACCOMPLISHED AND WHERE WE ARE TODAY

Viviane Clay
vclay@thousandbrains.org

Scott Knudstrup
sknudstrup@thousandbrains.org

Niels Leadholm
nleadholm@thousandbrains.org

Hojae Lee
hlee@thousandbrains.org

Ramy Mounir
rmounir@thousandbrains.org

Benjamin Rothman
brothman@thousandbrains.org

Tristan Slominski
tslominski@thousandbrains.org

Thousand Brains Project
Santa Rosa, CA, USA

September 22, 2026

1 The Thousand Brains Project

The Thousand Brains Project (TBP) has been around for about two years now, and a lot has happened since we started. The following report will recap all the exciting progress we have made since then and provide a high-level overview of our direction.

The project was first announced at Stanford's HAI5 event on June 5th, 2024, where our founder, Jeff Hawkins, closed his talk by predicting that "sensorimotor learning, not today's AI, will be the technology at the heart of [the transformation of the 21st century]". The Thousand Brains Project would be dedicated to building this novel sensorimotor AI as an open-source platform.

With funding from the Gates Foundation, among others, we set out to hire a small team of researchers and engineers, as well as a community manager to support our goals of opening up the existing research efforts at Numenta. On October 23rd, 2024, the TBP was incorporated as a non-profit company, and most of the team was hired and busy working on our mission to:

1. Create core sensorimotor algorithms guided by neuroscience.
2. Create an open-source platform for building sensorimotor products.
3. Increase awareness of sensorimotor technologies for AI and robotics.

Over the following month, we open-sourced our code base, `tbp.monty`, under the permissive MIT license, put existing patents under a non-assert pledge, and started to release past and present meeting recordings on our YouTube channel.

Before we dive into the different aspects of the project (research, platform, open-source community), we want to provide a brief overview of how they relate to one another and how they tie into our overall workflow for building a sensorimotor AI platform. Figure 1 provides an overview of the different steps involved in going from neuroscience theory to building something that can be used in real-world applications.

We use the Thousand Brains Theory of Intelligence [Hawkins et al., 2019, Hawkins, 2021, Hawkins et al., 2025] as the foundation for the principles implemented in our platform (referred to as Monty throughout this document). Theoretical ideas are first implemented and tested as research prototypes. Issues discovered during prototyping can feed back to adjust the theory, and when encountering difficult open problems, we often consult the neuroscience literature on how the brain might be solving them. So far, constraints of the brain have been a useful guideline for us and have helped us find elegant solutions to difficult problems.

Once a research prototype has proven to deliver the expected benefits to the overall system, the developing researcher coordinates with the engineering team to integrate it into Monty as a new feature. We refer to this process as an *Implementation Project*. This ensures that new features are integrated into our platform while preserving platform stability and quality.

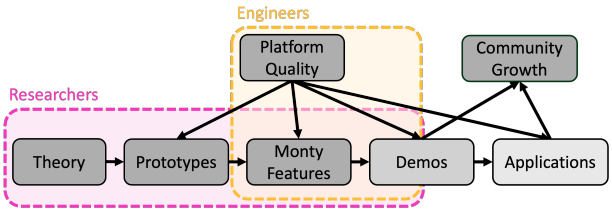

Figure 1: From theory to applications. Our workflow and each team's focus, where arrows indicate one element contributing/leading-to to another.

In addition to integrating new features into Monty, our engineering team works to improve the overall platform's quality and usability. This means making it easier to use, understand, and contribute to the `tbp.monty` code base. Platform improvements can accelerate prototyping, new feature integration, and the ease of use and compatibility for various demos and applications.

Ultimately, making Monty more capable aims at making it more useful in real-world applications. Although building specific applications is not our team's focus, we occasionally build demos to better understand challenges Monty will face outside the simulated experimental world. However, for the most part, we focus on the core, general-purpose algorithm and platform development.

As we want this technology to be accessible to anyone and to increase awareness of sensorimotor AI based on the principles of the neocortex, we also engage with external people and communicate our work openly. We are working to build an active open-source community around the project, with outside contributors helping us accelerate our progress and providing valuable input. As Monty becomes more capable, we also hope to see more people test Monty in real applications and want to make this as easy as possible.

2 Developing the `tbp.monty` Platform

2.1 Overview

Part of the vision of the Thousand Brains Project is to develop a platform for building AI and robotics applications using the same principles as the human brain. To achieve this vision, we pursue platform work while conducting active research. Both the research and the platform are essential to the project's vision. We need to succeed at both.

Therein lies an inherent challenge to achieving our vision. From the perspective of supply-and-demand competition in the marketplace, research and platform building differ

in their characteristics. Some of what makes for good research directly conflicts with what makes for a good platform, and vice versa.

For example, because of the nature of research, we expect experiments not to work, at least initially. Our tolerance for failure is high. For a platform to be useful, it cannot fail at the same rate as research. Initially, users may be willing to tolerate some failure due to our growing pains; ultimately, the goal is for failure to become surprising and not generally tolerated.

Similarly, in research, we want to minimize the cost of change. Ideas can change rapidly. We want to rapidly try new experiments, new theories, and new implementations. We want minimal friction between an idea and experimental data. For a platform, there is only so much change that users can absorb. Even if we rapidly iterate on platform internals, any externally facing interface benefits from stability.

All platform development has been a balancing act between these kinds of differing research and platform constraints. For more, see RFC 14 Conducting Research While Building a Stable Platform.

2.2 From Research to Platform Code

The codebase was open-sourced under the MIT license on November 13th, 2024. Since then, the team shipped 54 public releases, reaching v0.47.0 in August 2026, accelerating from a release every few weeks to a weekly cadence. The team shipped 918 changes, including 239 code refactors and 197 explicitly breaking changes. This highlights the ongoing effort to convert a research codebase into a stable, documented platform that external researchers and product builders can depend on. Instead of rewriting Monty as a separate "platform" codebase, the team is evolving the existing code into the platform. Rapid research prototyping happens on forks, and stable capabilities are merged back in through dedicated implementation projects.

2.3 Establishing an Open-Source Platform

As part of the open-sourcing process, the team established an open-source governance framework. The Executive Director selects the Lead Maintainers who oversee Maintainer decisions. Maintainers can come from the internal team or from the community. For more details, see the governance documentation. The team established an RFC process to provide a consistent and controlled path for substantial changes and further simplified it in RFC 5 Easier RFCs. RFC 2 PR and Issue Review established how the Platform accepts contributions. To minimize friction, the team created an optimized process for accepting Contributor License Agreements.

RFC 7 Monty Versioning established a formal versioning policy that defines a contract between Maintainers and the community about how Platform changes are communicated. To further improve communication, the team adopted RFC 10 Conventional Commits.

The standards for code contribution were clarified in RFC 11 Code Organization Guide, RFC 16 Project Typing Guidelines, RFC 18 Code Guidance for Researchers and Community, and RFC 19 Repository Support Levels.

In addition to open-sourcing the codebase, the team open-sourced the Platform planning. Platform plans and progress are transparently tracked in the Current Reality Tree and Future Reality Tree artifacts. These summarize the desired effects contributors are working toward and the obstacles being removed to reach them.

So far, 16 outside contributors have authored 121 merged pull requests. Some of them took on core, breaking-change refactors under maintainer review. Substantial highlights:

- AgentRev - 16 pull requests. The most architecturally significant outside work: a sustained series of refactors reshaping the simulator/environment abstractions into clean protocols (deprecated `ActionSpace`, added `Observations/SensorID/Modality` newtypes, `ProprioceptiveState`, split `EmbodiedEnvironment` into protocols, moved RNG into `RuntimeContext`). This directly enabled Monty's Runtime/Experiment decoupling work. Also, AgentRev authored the Getting Started on Windows Subsystem for Linux tutorial.
- TimothyAlexisVass - 36 pull requests. Codebase-wide spelling fixes, ruff lint compliance, and a major test-suite reorganization, amongst others.
- carver - 30 pull requests. Systematic `os.path` to `pathlib` migration, some security hardening, and bug fixes in regression tests.
- mf-maymay - 12 pull requests. Typing modernization: PEP 604 syntax, TypedDict for `SensorObservations`, np.quaternion typing, mypy error reduction.
- annark - 4 pull requests. Environment-interface refactors and renames.

2.4 Motor System, Policies, and Sensor Module Goals

The motor system and its uses were refactored to improve data flow and ease of understanding (#234, #263, #845, #864). The platform gained the ability to switch motor policies at runtime via a policy-selector mechanism, and policies were split into composable, reusable, finer-grained components (#889, #917, #898, Video summary: Updates to Monty's Motor System). Sensor module goals were introduced into the Monty framework so that low-level, model-free sensory information could have a more direct impact on the actions generated by the motor system (RFC 12, #511, #505).

2.5 Simulation, Configuration, and Packaging Modernization

Migration from the established Habitat simulator to MuJoCo is ongoing (#361, #951, #1012, #876). Integrating with the MuJoCo simulator enables the platform to leverage a standard, widely available simulator with a large user base, making onboarding easier. The essential components, such as embodied agents and sensors, have already been migrated and verified as working. What remains is completing the ongoing migration of the test suite and the final benchmark validation.

The team replaced bespoke Python configuration code with an industry-standard declarative configuration using Hydra (#540, #857). This makes experiments easier to define, compose, and share. Additionally, the configuration was restructured to be discoverable and installable by external users who use Monty as a dependency. Ongoing work is progressing toward improved compositionality enabled by dependency injection (#975, #976).

All of this is part of an overall modernization effort migrating away from Habitat, Python 3.8, and conda package management to MuJoCo, modern Python, and standard uv and pip packaging.

2.6 A Clean Runtime and Experiment Split

To better define the boundary between production runtime and experimental needs, the platform was split into runtime and experiment components. Each major component within Monty has been split into explicit Runtime and Experiment aspects. First, this required decoupling of previously entangled concerns, followed by explicit protocol specification and separation at a module level. The end goal is for operators to be able to run the platform without ever knowing or having to account for the fact that researchers have an experimental framework to run experiments on the platform. Achieving this separation in software enables the project to finally serve the distinct and conflicting needs of ongoing operations and experimental work.

3 Improving Monty's Capabilities

3.1 Background: The Compositional World

When the Thousand Brains Project formed, Monty already manifested many of the principles of the Thousand Brains Theory [Hawkins et al., 2019, Hawkins, 2021]. The emergent capabilities were formally evaluated in our paper "Thousand-Brains Systems: Sensorimotor Intelligence for Rapid, Robust Learning and Inference" [Leadholm et al., 2026] - see also Section 6 for additional discussions on evaluating Monty's capabilities. However, the primary focus of our research team is to push the frontier of Monty's capabilities.

Over the past two years, this effort has focused on Monty's ability to model compositional objects. In the original version of Monty, there was an inherent assumption that only one object existed in the world at any point in time. Naturally, this is an impoverished approximation of the complexity of the real world, which is inherently compositional. For example, a living room typically contains multiple objects, within which nested compositionality exists. The living room can be considered a parent object of a child object, such as a table. This table is a parent to other objects, such as wooden legs. Resting on the table might be a mug, which has a handle and a body. On the mug's side might be a printed logo, where the logo is composed of shapes or letters. Being able to efficiently learn and infer the existence of such compositional structure is paramount to an intelligent system.

To learn a compositional world, a first key step was to provide Monty with a hierarchical architecture (Figure 2). Learning modules (LMs) are the core units of Monty, analogous to cortical columns in the brain. They learn structured representations of objects, and so a compositional model of the world requires a hierarchy of LMs, where higher-level learning modules (HL-LMs) model parent objects, and lower-level learning modules (LL-LMs) model child objects. In and of itself, however, this architectural change is insufficient to support compositional modeling. In the following sections, we describe the key features we have worked on to unlock this capability.

3.2 Burst Sampling: Hypotheses to Accommodate an Ever-Changing World

As our eyes move around a room, or our hand travels across a table, the compositional nature of the world means that the objects responsible for the incoming sensory data rapidly change. In order for Monty to accommodate this reality, we needed to significantly change how hypotheses are initialized and maintained.

One of the strengths of Monty is that it uses sensory information to initialize its hypotheses about what it is perceiving. For example, when sensing the side of a mug, the local surface being sensed provides some information about how an object like a mug (or an alternative hypothesis like a soup can) would be oriented in space. The initial hypotheses are tested over the course of sensorimotor inference. However, the early version of Monty was only able to carry out this hypothesis initialization process at the very start of a discrete, experimenter-controlled "episode". Before showing another object, Monty's internal representations needed to be reset, enabling another hypothesis initialization phase in the subsequent episode. The solution to this limitation was to introduce what we term *burst sampling*.

Burst sampling is a method to dynamically grow and shrink the hypothesis space. In short, if sensory information is not being adequately predicted by the internal representations of an LM, then the LM will "burst", initializing a set of new hypotheses based on the current sensory information. As

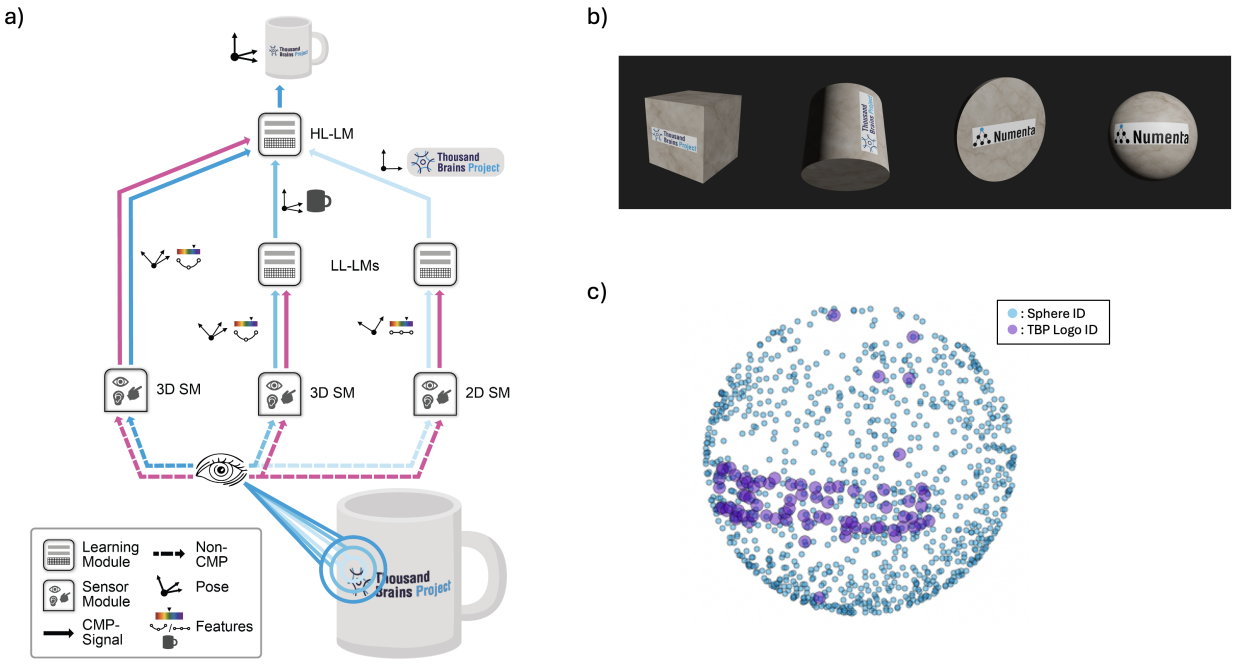

Figure 2: **Compositional Connectivity, Objects and Models.** The world is inherently compositional, with many objects sharing hierarchical relations. a) Monty now implements a hierarchical (or more specifically heterarchical - see Hawkins et al. [2025]) architecture, where learning modules at different levels of the system can independently represent objects. Based on the sensory data they receive, some learning modules have a preference for representing 2D or 3D objects, yet the internals of these learning modules are identical. A higher-level learning module (HL-LM) receives input from lower-level learning modules (LL-LMs), as well as direct sensor module (SM) input. Signals within Monty conform to the Cortical Messaging Protocol (CMP), and consist of both location/movement data (pink arrows) and sensory features (blue arrows). Different sensor modules can have receptive fields of different sizes (indicated by the co-located circles over the mug). See Leadholm et al. [2026] for more details. b) Examples of compositional objects that can exist in the world and that consist of both 2D (logos) and 3D (cube, cylinder,...) objects. c) An example of a learned compositional model in Monty. Blue points correspond to where a lower-level learning module has recognized a 3D object (the sphere), while purple points correspond to where a different lower-level learning module has recognized a 2D object (the Thousand Brains Project logo). These child objects are stored as inputs in the model of a higher-level learning module; the resulting compositional model is visualized here. In the following figures, we demonstrate the key elements that enable Monty to learn such compositional models.

a result, Monty can naturally accommodate moving onto new objects in the world. We quantified this capability with our unsupervised inference benchmarks (link), and it represents a crucial mechanism for supporting compositional objects (Figure 3).

Together with a mechanism to eliminate hypotheses that are no longer reflective of incoming sensory information, burst sampling enables Monty to rapidly adapt its hypothesis space while maintaining computational efficiency. While the core aim of this feature was to support the compositional nature of the world, bursting also enables Monty to better handle noise. In particular, a noisy sensory observation can result in a poorly initialized hypothesis space. By carrying out additional instances of hypothesis initialization, burst sampling enabled us to improve Monty's performance across our core benchmarks of single-object recognition, all while maintaining computational efficiency (#783). For more details, you can read our documentation on burst sampling at this link.

3.3 Model-Free Exploration Policies: Saccades Driven by Inhibition of Return and Saliency

Monty currently implements two sensorimotor embodiments, which influence how it is able to move within, and sense, the world. The first of these is the "surface agent", which moves like a finger over the surface of objects, only sensing things that are nearby. The other is called the "distant agent", and it moves like an eye, performing saccades that bring different parts of the world (which may be far away) into focus.

One of the key limitations of the original distant agent was that saccades were limited to small, random reorientations of the camera; these effectively resulted in performing a random walk over the visible surface of an object. In order to efficiently handle the compositional nature of the world, we rapidly move our eyes to relevant objects and explore each region until something is recognized. For example, if looking around a living room, our attention might be rapidly drawn to a table, or a mug on the table, or a logo on the side of the mug. A series of small, random eye movements would only reach all of these locations after an excessively long time.

We therefore carried out a line of work implementing efficient saccade policies in Monty. Most sensor modules (SMs) in Monty receive a narrow sensor patch (see the square patches in Figure 3), and pass processed feature and location information to a learning module. To address our new need, we implemented a specialized sensor module that receives input from a large visual field, processes saliency, and can propose regions to attend to; this roughly corresponds to the role of the subcortical superior colliculus [Basso and May, 2017].

More specifically, SMs were equipped with the ability to propose candidate goals for the motor system to enact, just as LMs could already produce goals for hypothesis testing. These SM-generated goals define a location that should be sensed, and a Motor System policy was implemented such that the distant agent would then focus its local sensors on the target location.

To propose goals, we implemented two strategies. The first was a simple inhibition of return policy - if a region of sensory space has been visualized before, this reduces the chance of returning there. The second was a mechanism to use heuristics around color, depth, and edges to propose salient regions to attend to. This latter work represented a model-free means of proposing candidate areas for exploration, and was based on VOCUS2 [Frintrop et al., 2015].

With these strategies, the distant agent is able to rapidly move to regions of the world where relevant objects may exist, as well as systematically explore the available visual space (Figure 4a). This is an important, albeit insufficient, change to enable efficient learning and inferring of compositional objects.

An additional requirement is that, once on a potential object, we stay there until it is recognized; this is the subject of Section 3.5. In the interim, we observed an improvement across our distant-agent benchmarks with the more efficient inhibition-of-return policy (Figure 4b) (#1003). Similarly, the saliency-based saccade policy demonstrated an important step towards sparser models that still support robust inference (#1041).

3.4 Distortions: 2D Sensor Modules for Learning Invariant 2D Models

Not all objects in the world are well represented as structures that are rigid in 3D space. The lettering of a word can distort as it follows a curving surface on which it is printed, yet it is still legible. Similarly, a T-shirt can be crumpled, or a wire tied in a knot, without affecting the fundamental relations between the parts of the object (the sleeve of the T-shirt attaches to the shoulder, which continues until it reaches the neck, etc).

One way of describing this is that these fundamental relations are preserved in a lower-dimensional structure - 2D in the case of the logo or T-shirt, 1D in the case of the wire. To enable compositional modeling of the diverse objects present in the world, we need to ensure that Monty can learn and recognize objects whose true structure may be lower-dimensional.

A classic object we have discussed over the years is a mug with a printed logo on its side. While seemingly a simple object, it captures the nuance that different objects in the world can have underlying 2D or 3D structures, and that these can interact compositionally. As such, we began this work focused on the case of 2D objects. To realize the desired capability, we implemented a specialized SM that is able to extract features and movements corresponding to 2D space.

In Monty, morphological features - those that define a local pose - are key to the representations it builds. In 3D space,

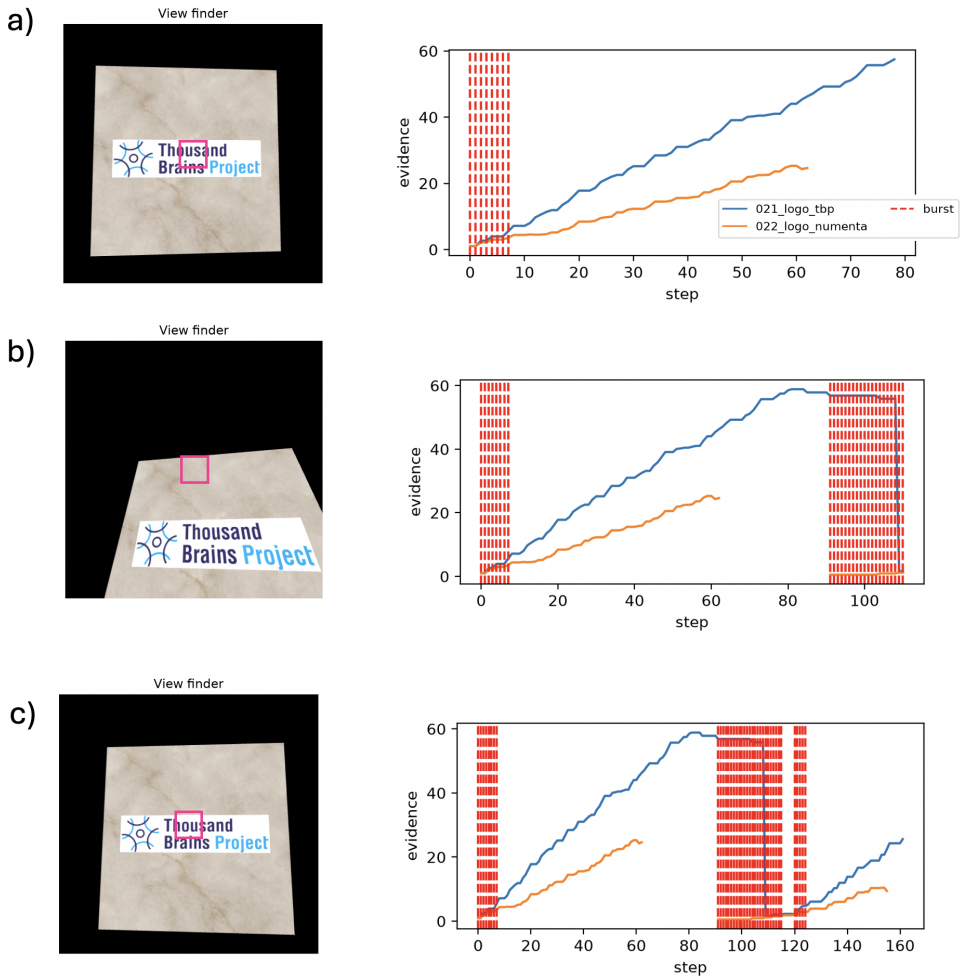

Figure 3: **Burst Sampling Supports Adaptation.** An intelligent system needs to constantly update its hypotheses about objects that are present in the outside world. Shown are a series of steps from an inference episode with Monty. On the left is a view-finder image providing context for where Monty is currently sensing, which is specified by the narrow sensory patch (pink square). On the right are Monty's top hypotheses for objects as a function of episode steps. "Evidence" reflects the confidence that has built up about the existence of a particular object. Dashed red lines show burst sampling. a) Monty has just spent several steps exploring the sticker, and has therefore become confident that it is seeing the Thousand Brains Project logo. The earlier discontinuation of the orange line for the Numenta logo corresponds to when all hypotheses for that object were eliminated. b) When Monty moves off the sticker, its existing hypothesis is no longer relevant - this hypothesis therefore loses evidence rapidly, and Monty begins bursting to identify new, relevant hypotheses. Note that, in this case, Monty does not have a model for the marble texture, or 3D objects like the cube, and so it continues to burst. c) As Monty moves back onto the logo, it bursts again. Some of these newly sampled hypotheses are able to explain the sensory data coming in, and their evidence accumulates.

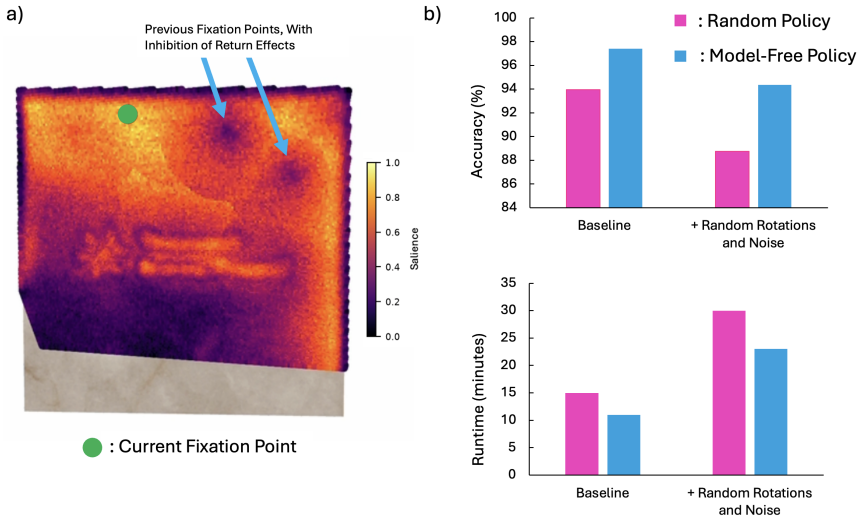

Figure 4: **Saliency and Inhibition of Return for Efficient Exploration.** Monty can leverage a model-free policy to rapidly explore the world. This combines a bias towards looking at salient regions of an object (as determined by an adapted version of VOCUS2 [Frintrop et al., 2015]), and a bias against revisiting previously observed locations (inhibition of return). a) Monty is looking up at the cube with the TBP logo; the receptive field of the saliency SM corresponds to the extent of the dark region superimposed on the 3D model. Bright regions in the saliency map correspond to areas that are deemed worth visiting with sensors that pass features to LMs. The green dot represents the current fixation point. The dark, ring-like regions correspond to locations that Monty has recently attended to, and which therefore have their saliency reduced. b) Prior to the introduction of this strategy, Monty would randomly perform small saccades across the surface of the object ("Random Policy"). Introducing inhibition-of-return significantly improves Monty's ability to accurately and rapidly recognize objects ("Model-Free Policy"), including in the presence of random object rotations and sensory noise.

these correspond to the surface normal and principal curvature vectors sensed in a local surface patch. In 2D space, these correspond to oriented edges. The first responsibility of the new `TwoDSensorModule` was therefore to extract the dominant edge and its orientation within a sensory patch.

The second key requirement of the `TwoDSensorModule` was for it to estimate the 2D movement that takes place across a surface, given that sensed locations are received as displacements in 3D space. We implemented a series of transformations leveraging the sensed curvature of a surface in order to accurately estimate such 2D displacements.

By passing 2D morphological features and 2D displacements to a standard learning module, the latter is able to learn 2D models of objects such as logos (Figure 5a). As hoped, we observed that these representations generalized at inference time to instances of logos on different types of surfaces, despite the distortions they induce (Figure 5b). While their internal algorithms remained unchanged, LMs that were provided with 2D data, and therefore learned 2D models, demonstrated superior performance on objects with underlying 2D structure (Figure 5c). Further details on these results are available in our expanded documentation on the 2D SM.

3.5 Model-Free Attention: Focusing on Child Objects

As noted in Section 3.3, a policy that efficiently explores the world is useful, but in and of itself insufficient to deal with compositional objects. In particular, a sensorimotor system needs to attend to a particular object and recognize it before it moves on. Without such a mechanism, hypotheses will constantly shift (see burst sampling in Section 3.2), and there will be no opportunity for a higher-level LM to receive reliable information about child objects in the world during learning or inference.

A key element to address this was to provide Monty with a model-free mechanism to identify a region associated with a (potential) object, and to stay in this area until an object is recognized. The mechanism should be model-free, as until Monty has reasonable hypotheses about what it is seeing, model-based methods will not be useful. We therefore implemented a mechanism in a specialized sensor module that segments a candidate region to attend to. While this candidate region changes from one fixation to the next, a central attention region merges them together over time (Figure 6).

The attention region constrains where Monty moves its sensors next; rather than moving to any region in the image, Monty only targets locations within the actively attended area. This had the desired effect of constraining Monty to stay on an object for several inference steps - sufficient for LMs to develop good hypotheses about the objects they are perceiving. To complement this, Monty's LMs occasionally propose moving to a new location on the objects being sensed, ensuring Monty does not remain in the same area indefinitely.

3.6 Bringing it Together: Modeling Compositional Objects

The ultimate aim of the above efforts was to enable Monty to learn and infer complex objects with compositional representations. By combining the various elements we developed, we observed that Monty is indeed able to learn compositional models of objects such as a sphere with a logo printed on its side (Figure 2c; see Figure 7a for additional examples).

During inference, these compositional models enable Monty to recognize compositional objects much more quickly (Figure 7b). Without compositionality, a learning system is forced to use "monolithic" representations - a single model attempts to learn all of the sensory features associated with a particular arrangement of objects in the world (such as the TBP logo printed on the mug). The monolithic model is unable to learn that the TBP logo and mug correspond to different objects that repeat in the world with different, potentially limitless configurations. Compositional models enable a different approach. First, LL-LMs in Monty can specialize in representing particular child objects, where different LL-LMs can learn objects with different assumptions about their dimensionality. In this case, one LM senses features and movement in 3D space, learning representations of the 3D objects such as the cylinder and mug. Another LL-LM senses features and movement in 2D space, and learns representations of the 2D objects, such as the different logos. Due to the hierarchical architecture of the system, the HL-LM receives inputs from both of the LL-LMs, building compositional models that combine this information. During inference, each LM can recognize the objects it has learned, including the HL-LM using the incoming information from the LL-LMs to rapidly converge on a compositional object.

The elements outlined earlier all play a key role here. Burst sampling is essential for an LL-LM to change its hypotheses as it moves through the world; this is necessary to ensure that the representations it passes to the HL-LM are meaningful during both learning and inference. The exploratory saccade policies enable us to efficiently visit the set of objects that may be encountered, while this is constrained by the model-free attention that focuses Monty on an individual object at any given point in time. Finally, the 2D SM ensures that one of the LL-LMs can learn models of 2D objects, with two key consequences. Firstly, this enables more accurate recognition of the particular 2D object; without 2D edges as features, the logos are very difficult to distinguish. Secondly, by estimating movement in 2D surface space, the same logo representation can be reused across different instances (such as when it appears on the surface of a sphere, on the surface of a cube, or simply on its own), without needing to relearn it.

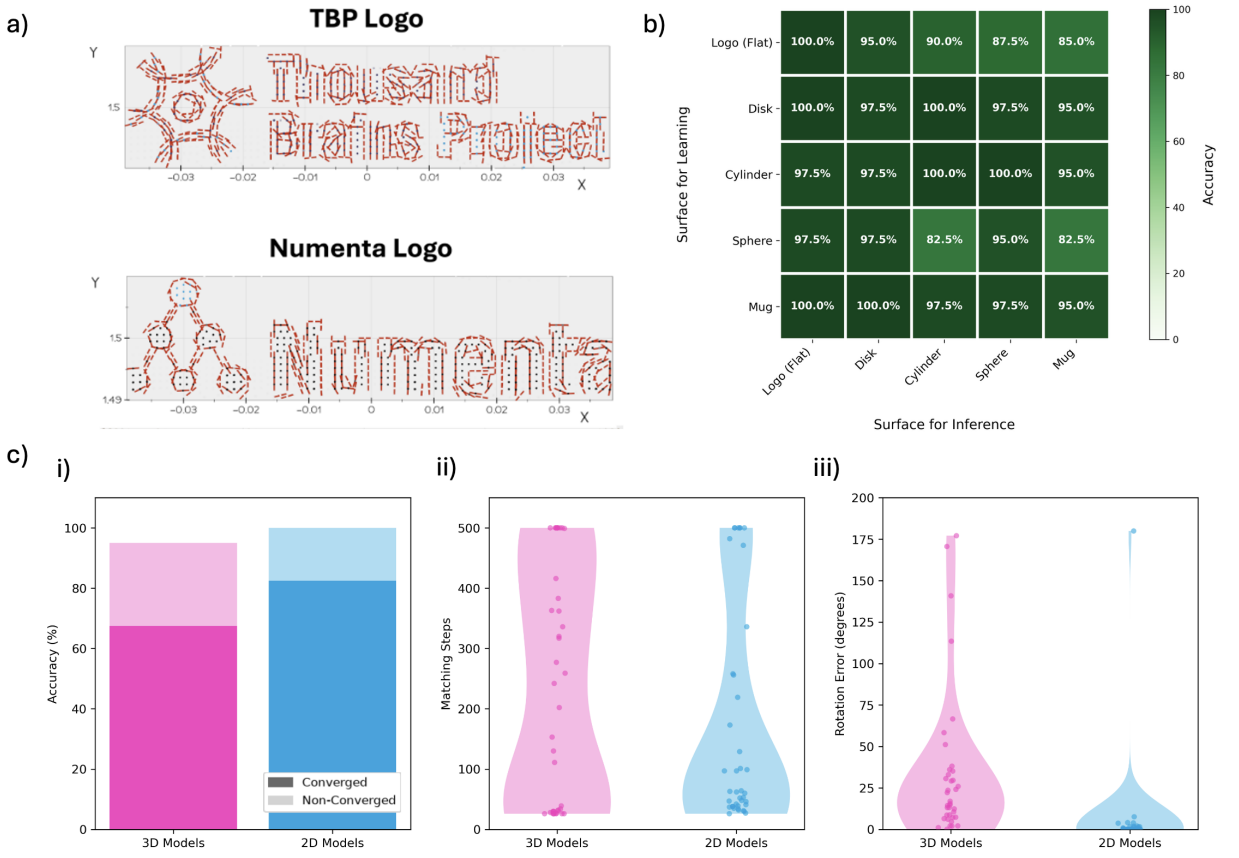

Figure 5: **Modeling 2D Objects.** By simply changing the kind of sensory features and movement data a learning module receives, the same internal processing results in the learning of 2D rather than 3D models. a) These models are defined by the orientation of prominent edges, rather than the orientation of surface normals and principal curvature that define Monty's 3D models. b) As well as extracting the prominent edge of a sensory patch, the 2D sensor module estimates the 2D movement across the surface of an object, based on locally sensed geometry. As a result, the learned LM models are invariant to distortions caused by the curvature of 3D objects that the 2D objects (here logos) interact with. This manifests as a consistent ability of the LM to recognize the logos, despite learning and inference taking place on surfaces with different geometry (chance performance 50%). c) i and ii) An LM that learns 2D models of 2D objects infers them on a curved surface more quickly and more accurately than an LM that represents them with a 3D model; this includes achieving a higher proportion of instances where Monty converges to a high-confidence, correct classification ("Converged"). iii) The use of 2D features also supports more precise pose estimation in comparison to an LM that uses 3D models to represent 2D objects.

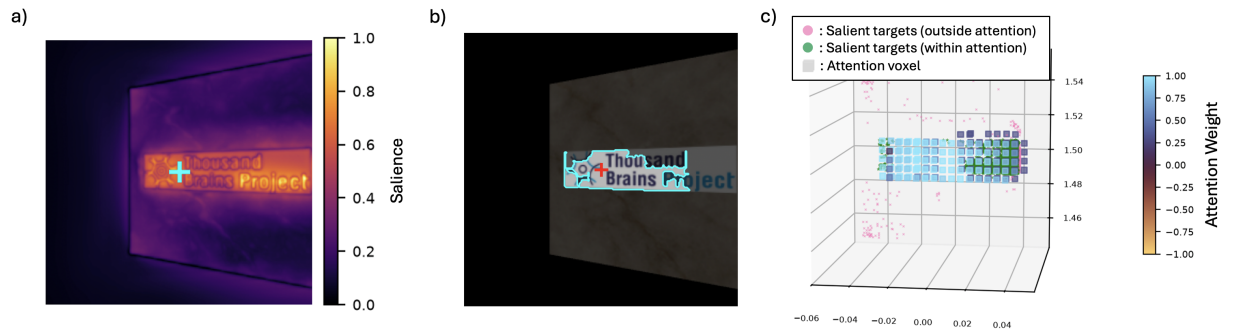

Figure 6: **Model-Free Attention to Remain on an Object.** Similar to the model-free saliency that informs Monty how to rapidly explore the visual world, we implemented a model-free system to coarsely segment and thereby encourage Monty to attend to potential objects in the world. a) At any given moment in time, an SM determines what parts of the world are salient, and thereby would be worth moving sensors to. b) Parallel to this, a specialized SM extracts a segmentation mask, corresponding to a candidate object around the point of fixation. c) These segmentation masks are accumulated over time in a body-centric *attention region*, which is represented by weighted voxels. After just a few fixations, this attention region often corresponds to an object of interest (such as the logo-sticker shown here). This attention region then determines what target goals can be acted upon by Monty, such as saccading to the "P" in "Project" (allowed, as within the attention region) and not saccading to the top-left corner of the cube (filtered out by attention).

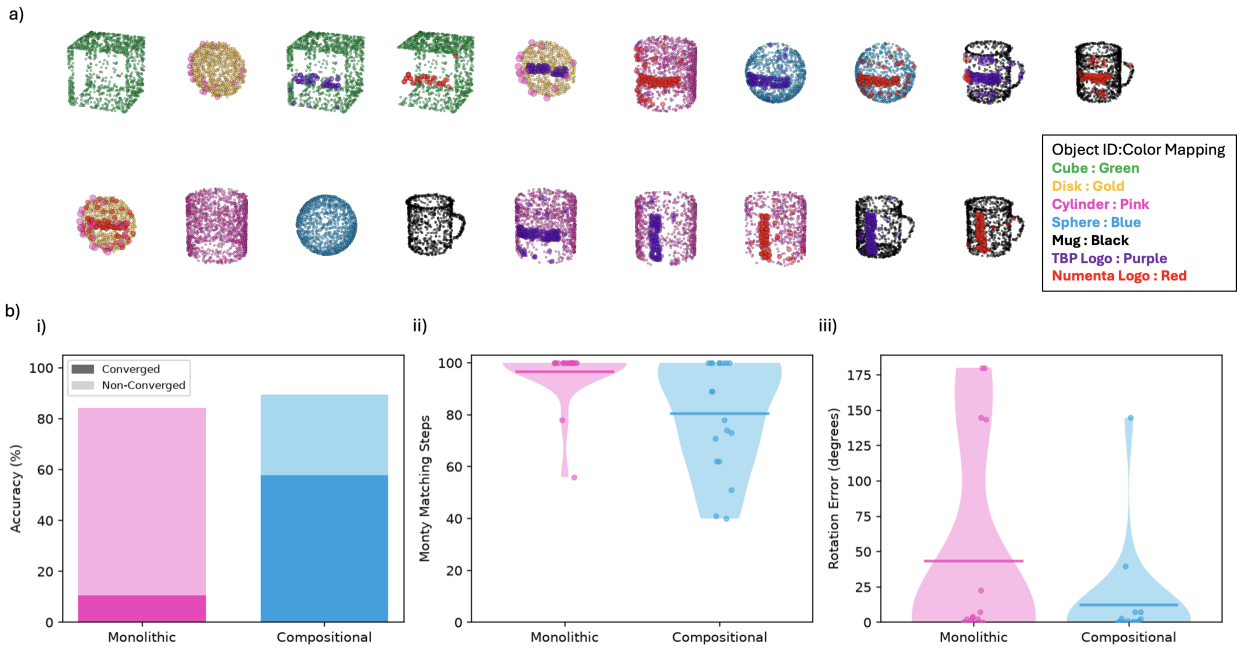

Figure 7: **Compositional Models in Monty.** By combining the various elements developed over the last two years, Monty is able to learn and infer compositional objects. a) Examples of compositional models learned by Monty. The colors correspond to the object ID passed by a lower-level learning module, and stored in a reference frame within the higher-level learning module. In this dataset, the logos can be oriented both vertically and horizontally on parent objects. Best viewed in color. b) In a "monolithic" version of Monty, a single learning module is responsible for learning all elements of an object, including the sensory features that result from something like a logo being printed on its side. In a compositional version of Monty, LL-LMs are able to recognize distinct child objects, such as the logo, and then pass this information up the hierarchy. i + ii) As such, compositional models provide faster inference that is more likely to converge to a confident classification ("Converged"). iii) The 2D learning module is also able to better infer the orientation of logos (see Figure 5c); this likely contributes to the compositional Monty achieving lower pose error when inferring compositional objects, supporting the benefit of distributed representations.

Future work will explore the additional benefits of compositional models for learning efficiency and predictive representations. These include supporting sparse representations, and top-down projections from an HL-LM causing an LL-LM to rapidly instantiate a representation where it is expected.

4 Developing the Theory Further

A large part of developing the theory involves us meeting for long brainstorming sessions to discuss problems, ideas, and constraints. Often, the hardest challenge is precisely defining the problem we are trying to solve. This also involves some of the constraints we want to impose on the possible solution space, given the realities of what the brain can do (e.g., locally available information, the amount of data available for learning, the speed of inference and adaptation, etc.). Once all this is clear, a solution can present itself in surprising "aha" moments (there are rarely multiple solutions that fit all the constraints), and we play through various scenarios to see whether it actually solves all the problems we expect it to, and to refine it further.

Over the past two years, we have spent countless hours in such brainstorming sessions, including five focus weeks dedicated solely to brainstorming together on a whiteboard, as well as weekly research meetings that can last up to four hours. If you are interested in what such meetings look like, you can find recordings of many of them here.

As ideas matured, we wrote them up more formally and concisely. One major update we shared is the paper *"The Thousand Brains Theory 2.0: An Extension for the Long-Range Connections of the Neocortical Heterarchy"* [Hawkins et al., 2025]; this work has been accepted for publication in Neural Computation, although it has yet to appear in print there. In this work, we describe several key new ideas that we have developed since the first publications on the Thousand Brains Theory [Hawkins and Ahmad, 2016, Hawkins et al., 2017, 2019, Hawkins, 2021]. In particular, we describe the hypothesized role of the thalamus in transforming between egocentric and object-centric reference frames. We also describe the role of hierarchical connections for modeling compositional objects.

In addition, we made significant progress in developing a theory of how the brain could model object behaviors. This refers to modeling how objects change over time. We published a write-up of the key ideas and a plan for testing them in Monty in our documentation.

Beyond this, we made theoretical progress on various other open questions and topics, such as attention, shared learning and inference between columns, fast learning and its relationship to the hippocampus, causal interactions and goals, and object deformations. For further resources on these topics, see section 7.1.

Overall, we got a much better grasp on the remaining problem space. Previously, there were large uncovered areas in the theory, but now, we feel like we have laid out and discussed all the major remaining problems. Although we have not yet settled on solutions for all of them or tested all the ones we came up with, it seems we finally have a good grasp of what remains to be done. This gives us hope for the coming years and makes us more certain that we are on the right track to building a fundamentally new kind of sensorimotor AI.

5 Community and Communications

After open-sourcing the project in November 2024, we published a range of supporting resources and set up communication channels to regularly share our work.

To provide people with an overview of our project and approach, we published a whitepaper in December 2024. In it, we describe the project's goals, the Thousand Brains Principles, and the current algorithm implemented in Monty.

We also wrote and published extensive documentation at docs.thousandbrains.org. This documentation has been continually maintained and expanded over the past two years. We set up an edit workflow that allows team members and external contributors to edit our documentation via pull requests to `tbp.monty`. The documentation covers various topics, including the project's vision, our approach, the latest theory updates, the algorithm we implement, the different components of Monty, getting started instructions, follow-along tutorials, benchmark experiments and results, application criteria, demos, ways to contribute to the project, how to customize Monty, style guides, project governance, our project's roadmap, and future work items. Since its original release, we have made over 200 pull requests to update the documentation and ensure it remains a useful resource for anyone coming to the project. We also integrated an interactive table of our future work items, allowing people to easily find a part of the project they can contribute to, based on their skills, time commitment, and interests.

We started our two main continuous communication channels: YouTube (@thousandbrainsproject) and our Discourse forum (forum.thousandbrains.org). We publish recordings of all of our research meetings, as well as other supplemental material, on our YouTube channel. This way, people can follow along with our progress, and our work is transparent. By now, we have published 185 long-form videos (mostly meeting recordings), 98 shorts, and 1 live stream. On Discourse, we provide a platform for interested people to ask questions about the project, share their ideas and work, and collaborate with us. As of today, people have created 543 topics on our forum, with 2,384 replies overall. Our entire team is active on the forum and enjoys interacting with the community, with $\sim$1,100 replies and posts from our team members.

To further promote the project, we have given various invited talks and interviews about the project and our approach. Additionally, we organized events such as the

2024 launch symposium, a robot hackathon to demonstrate what Monty can do today, and our Meet Monty 2026 talk series.

We have fostered various external collaborations. For instance, we worked with students on their final thesis projects, with researchers in industry on Monty-related projects, and with engineers who contributed to our platform roadmap (see 2.3). Wherever possible, we conduct these collaborations in the open, either through conversations on our Discourse forum or through recordings of working group meetings and final presentations shared on YouTube. Additionally, we met with application groups, particularly in the medical ultrasound space, to scope out opportunities where Monty could make a real-world impact in the near term.

We would like to thank the community, contributors, and people we have collaborated with over the past two years. Our team thoroughly enjoyed their interactions with you, valued the conversations and ideas, and appreciated your contributions. For an overview of the wide variety of contributions, see our monthly community shout-outs, where we highlight some of them each month.

Lastly, we developed various tools around Monty to make it easier to explain, understand, and use. For instance, we developed and published:

- `tbp.plot` for interactive, 3D visualizations of what is going on inside "Monty's brain".
- `tbp.teleop` to interactively control Monty's actions during an experiment and watch it learn and infer live.
- `lazyconfigs` to quickly create new Monty experiment configs using a TUI.
- `tbp.floppy` to count the number of FLOPs Monty performs during learning and inference and measure its computational efficiency.

In addition to making Monty more accessible, these tools are also used by our own research team to accelerate prototyping and debugging.

6 Monty's Proving Grounds: Evaluations in Simulation and the Real World

Over the past two years, we have evaluated Monty's performance under a variety of conditions. Some of these were performed internally, by our own team, while others related to work outside of the TBP. Some evaluations took place in simulated environments with digital objects, while others were based on sensorimotor data captured in the real world. The results from these experiments point to a new type of intelligence that, while still in its early stages of development, demonstrates compelling capabilities across various metrics.

6.1 Demonstrating Monty's Capabilities: A Key Publication

In our paper, "Thousand-Brains Systems: Sensorimotor Intelligence for Rapid, Robust Learning and Inference", we rigorously evaluated Monty's performance as it existed at the outset of the Thousand Brains Project. These evaluations focused on Monty's ability to rapidly learn and infer digital copies of common household objects in a simulated environment.

We observed that the structured representations Monty learns through sensorimotor exploration naturally yield a diverse set of impressive capabilities. These include:

- Robustness to noise and unfamiliar rotations of objects: 88.1% accuracy on the YCB dataset with these perturbations combined, despite them never being seen during learning; chance classification accuracy is 1.3%.
- The use of object shape more than local textures when recognizing objects, as observed in humans: this manifested as an accuracy of 73.1% on YCB, despite all color or texture information being corrupted.
- The ability to identify the elusive notion of symmetry without this being directly supervised.
- Rapid inference through intelligent action policies and a consensus-forming algorithm called voting.
- Rapid learning: approximately 50% classification accuracy after learning on only a single view of each YCB object.
- Extreme computational efficiency during learning, requiring approximately 9 orders of magnitude less compute than a comparable deep learning network; furthermore, Monty outperformed this deep neural network on our accuracy metrics (object ID and rotation error).
- Ability to do continual learning, a setting where deep learning displays catastrophic forgetting.

Any one of the above represents a major challenge for most deep learning systems, and so their collective demonstration gives us confidence that the TBT principles are a promising roadmap for developing sensorimotor AI.

6.2 Other Evaluations in Simulation

Following the above publication, our own internal evaluations focused on Monty's ability to model compositional objects (see Section 3), including Monty's ability to perform inference when it is not informed that the objects in the environment may change (what we term unsupervised inference). However, researchers outside the TBP have performed additional evaluations of Monty's performance in simulated environments.

To examine Monty's inference in settings other than 3D objects, Korea Electronics Technology Institute scientist Kyou-Jung Son explored the ability of Monty to learn and recognize objects in the MNIST and Omniglot datasets. This exploration was performed before Monty's 2D sensor module was implemented, limiting the nature of features that Monty could extract from images. Furthermore, compositional models, as well as elements of unsupervised learning, will be essential to performing well on these datasets. Nevertheless, these explorations serve as useful baselines for Monty going forward.

Recent master's graduate Xavier Servot wrote his thesis on the computational scaling of Thousand Brains systems in different types of hardware. Using simulated Thousand Brains systems, he was able to demonstrate how Processor-in-Memory architectures may be particularly well-suited for the highly parallel computations that the former manifest.

6.3 Monty in the Real World

Monty's very first demonstration with real-world data used the depth camera of an iPad. It was dubbed "Monty Meets World", and took place in March 2023. In the last two years, we and others have further explored a variety of real-world settings where Monty's performance can be evaluated. Much of this work took place during a one-week hackathon in May 2025. There, the team worked on three independent projects.

The first of these projects used a Raspberry Pi together with LEGO sensors and servos to set up a robotic platform. This platform was able to rotate an object placed on it, while a separate sensor would move up and down, sensing the object at specific locations.

The second project used a quadcopter drone. While extracting meaningful depth values with a standard RGB camera proved more challenging than anticipated, this represented an additional proof of concept of a robotic agent manifesting Monty's sensorimotor algorithms.

The third project combined an ultrasound probe with a tracking system (originally designed for virtual reality) to explore Monty's ability to learn and recognize objects perceived through ultrasound. Objects were placed in a water bath, and a human operator moved the probe over them while the tracking device recorded the probe's precise movements.

Outside of the TBP, recent undergraduate student Zachary Danzig completed his thesis on the ability of Monty to perform perception via a robotic arm equipped with a depth camera. This work also demonstrated a re-implementation of the Monty Meets World pipeline on an alternative hardware setup.

While performance in the above settings does not yet match humans, these early explorations have provided us with invaluable experience in the challenges and opportunities presented by real-world applications of Monty.

6.3.1 Extended Work to Apply Monty to Ultrasound

We have placed a particular emphasis on investigating Monty's applicability to medical ultrasound. Since the POC was developed during the hackathon, the ultrasound work has been expanded to further demonstrate Monty's capabilities in this domain. Medical ultrasound is well-suited for Monty's capabilities: the ability to learn structured, interpretable models from small amounts of data, and the importance of sensorimotor interactions for both learning and inference. The extensions to our initial prototype consisted of further improvements to the data collection pipeline, scanning a full dataset of objects for offline processing, and the open-sourcing of this dataset, the benchmark experiments, and the associated code-base. We extensively documented and shared this novel real-world task as an open challenge that the community could contribute to. Additionally, we met with several ultrasound manufacturers and companies that apply AI to medical ultrasound to get a realistic view of the challenges they encounter in real-world settings.

7 What's Next

7.1 Research

We envision Monty as a general-purpose AI system with all the capabilities of the neocortex, though several remain absent to date. While a detailed breakdown of all the desired capabilities can look overwhelming, we roughly categorize them into three major milestones:

1. *Compositional Models (unsupervised and scaled up):* While we outlined above that Monty now has the ability to model compositional objects, this ability is not fully mature yet. In particular, Monty still requires substantial external supervision during learning, and we have not tested this ability at scale. A next step will be to improve Monty's ability to learn unsupervised, including forgetting and memory consolidation, and test learning with more learning modules, including the effects of attention and voting.
2. *Object Behaviors:* Our world is dynamic, and things are constantly changing. Objects move predictably and can be interacted with. Monty's current models are static and have no temporal dimension. Over the past 1.5 years, we've been working on a theory on how the brain could be modeling object behaviors and how a similar mechanism could be implemented in Monty. We put together a writeup of our planned approach and aim to test this as the next major research effort on our roadmap.
3. *Causal Interaction with the World:* At present, all of Monty's policies are aimed at learning and inference, meaning they move the sensors for better information acquisition, but they don't aim

to change the state of the world. To achieve goals by interacting with the world, Monty will need to learn about causality (how actions and behaviors can change the state of its learned models) and use this to plan and execute goals. It will need to break down goals into subgoals and eventually send them to the motor system. We have explored ideas around many of these topics, but the exact approach is still under development.

Each of these major milestones will unlock a wide range of new applications in which Monty can be useful. They will also touch on many other topics not explicitly listed above, such as attention, categories and generalization, voting, object deformations, and learning more or less temporary models.

Although we have ideas around many of these topics, the exact scope and solutions will only be discovered as we start implementing and testing them. This will also involve setting up novel test beds and evaluation measures.

7.2 Platform

We share details about our platform roadmap in the form of Current and Future Reality Trees, which are updated on a regular basis. If someone is interested in helping us push the platform forward, this is a great place to look to see where one could contribute.

Our team is currently working on three major items on this roadmap:

1. *Supporting a second simulator (MuJoCo) and retiring the Habitat simulator.* This will allow us to update the Python version Monty runs on, expand operating system support (e.g., Windows native), and move from conda to pip install. All of these make Monty easier for the community to learn, use, and adopt.
2. *Separating Monty from Experiment and Environment.* Eventually, we want Monty to only contain a light-weight runtime wrapper. Any experimental framework support will reside outside of Monty. Also, most environments (or the real world) are external to Monty, so any interaction needs to be mediated either by the runtime or by the experimental framework. Clearly separating these domains will make it easier to use Monty across a variety of real-world applications, rather than defaulting to an experimental research context.
3. *Emitting telemetry out-of-band from control flow.* Decoupling telemetry gathering from control flow will create a telemetry stream API boundary that should allow an ecosystem of telemetry consumers to flourish. This will make it easier for users to collect any necessary information for an application or experiment and visualize it using third-party or custom software, all without altering or rewriting Monty internals.

These efforts will continue throughout the next year alongside support for implementation projects to integrate new Monty features.

7.3 Community and Applications

Ultimately, we want Monty to be a generally useful AI system that can be easily adopted and make a positive impact on various sensorimotor applications. Adding new features to Monty and making Monty easier to use are aimed at enabling this.

To accelerate feedback on how well Monty is doing in applications, where features are missing, and the primary developer pain points, we take various steps. One of the biggest aspects of this is making our project open-source, having our team work in the open, creating resources to learn about our approach, and actively engaging with the community. We plan to continue all of these efforts. We also hope to encourage people to build early demos through new initiatives, such as a solutions incubator, and other collaborations. Additionally, we occasionally build demos ourselves and produce documentation around how people can build similar demos to spark ideas in others and show what is possible today.

7.4 Conclusion

The Thousand Brains Project was born of a core hypothesis: that replicating the computational principles of the neocortex will lead to a fundamentally different kind of AI. This form of AI would learn by moving, sensing, and interacting with the world, just like humans do. It is our belief that the benefits of this new approach to AI should be broadly shared through open source research and development.

Two years in, we have made exciting progress turning that hypothesis into a working open source platform called Monty. There is still much to discover and build, and we look forward to continuing this work in the open, alongside researchers, developers, and others who want to help shape the future of AI.

8 Funding

This work was supported by the Gates Foundation (Grant Number: INV-075337). The conclusions and opinions expressed in this work are those of the author(s) alone and shall not be attributed to the Foundation.

This work was also financially supported by the Ministry of Trade, Industry and Energy (MOTIE) and Korea Institute for Advancement of Technology (KIAT) through the "International Cooperative R&D program" (Task No. P0028486).

This work was also supported by the Thousand Brains Project, a non-profit research organization.

This work was also supported by Numenta, a privately held company. Its funding sources are independent investors and venture capitalists.

References

Jeff Hawkins, Marcus Lewis, Mirko Klukas, Scott Purdy, and Subutai Ahmad. A framework for intelligence and cortical function based on grid cells in the neocortex. *Frontiers in Neural Circuits*, 2019. ISSN 16625110. doi:10.3389/fncir.2018.00121.

J. Hawkins. *A Thousand Brains: A New Theory of Intelligence*. Basic Books, 2021. ISBN 9781541675810. URL `https://books.google.de/books?id=FQ-pzQEACAAJ`.

Jeff Hawkins, Niels Leadholm, and Viviane Clay. The thousand brains theory 2.0: An extension for the long-range connections of the neocortical heterarchy. *arXiv preprint arXiv:2507.05888*, 2025.

Niels Leadholm, Viviane Clay, Scott Knudstrup, Hojae Lee, and Jeff Hawkins. Thousand-brains systems: Sensorimotor intelligence for rapid, robust learning and inference. *Neural Computation*, 38(6):845–896, 2026.

Michele A Basso and Paul J May. Circuits for action and cognition: a view from the superior colliculus. *Annual review of vision science*, 3:197–226, 2017.

Simone Frintrop, Thomas Werner, and German M Garcia. Traditional saliency reloaded: A good old model in new shape. In *2015 ieee conference on computer vision and pattern recognition (cvpr)*, pages 82–90. IEEE, 2015.

Jeff Hawkins and Subutai Ahmad. Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. *Frontiers in Neural Circuits*, 10, 2016. ISSN 16625110. doi:10.3389/fncir.2016.00023.

Jeff Hawkins, Subutai Ahmad, and Yuwei Cui. A Theory of How Columns in the Neocortex Enable Learning the Structure of the World. *Frontiers in Neural Circuits*, 11(October):1–18, 2017. ISSN 1662-5110. doi:10.3389/fncir.2017.00081. URL `http://journal.frontiersin.org/article/10.3389/fncir.2017.00081/full`.