INSTITUTE FOR PHYSICAL AI @ BMI
The Charlot Lab

Technical Report TR-2026-12
Survey / Review · Preprint v1
7 July 2026

GRAPH OF THE WORLD

Graph of the World: A Hybrid Property-Graph and Vector Substrate for Embodied Memory

One engine that traverses relationships and searches by similarity, on the device that has to act.

David Jean Charlot, PhD

Dean of Physical AI · The Charlot Lab, Institute for Physical AI @ BMI

Correspondence: contact@physicalai-bmi.org · physicalai-bmi.org
Interactive companion: physicalai-bmi.org/research/charlot-lab#topic-graph

Abstract. Perception fills a world model and physics makes it actionable, but the model has to live somewhere. This report argues that it lives as a graph. Every entity an embodied system knows — a part, a material, a surface, a constraint, another agent, a sensor reading — is a node, and every relationship between them is an edge. Because the physical world is continuous rather than discrete, each node also carries a vector embedding, so a system can both traverse relationships and search by similarity in one place. The report reviews the two established lines of work this position rests on: the property-graph and knowledge-graph model on the discrete side, and vector embeddings with approximate nearest-neighbor search on the continuous side. It then describes `hyperdb`, the Charlot Lab's research effort to hold both in a single embeddable engine so that graph traversal and vector search run against the same store, on the device. Section 2 states the review method. Sections 3 through 5 develop the substrate. Section 6 describes what it holds for an embodied system and its relation to the Lab's perception and CAD work. Section 7 states the research position. The report reports no new experimental measurements.

1. Introduction

An embodied system carries a model of its world. Perception writes into that model — this surface is here, that object is graspable, this agent is moving toward me — and a physics or dynamics model reads from it to decide what to do next. The two are usually studied separately, and the interface between them is treated as an implementation detail. It is not. Both perception and reasoning presuppose that there is a thing they read from and write to, a

store of what the system currently believes about its surroundings, and the shape of that store constrains what the system can do. This report is about that store.

The claim is narrow and specific: the natural shape of an embodied world model is a graph. The objects of experience are discrete and related. A gripper is *made of* a material; a material *has* a friction coefficient; a surface *constrains* a motion; one agent *is observed by* a sensor; a reading *supports* a belief. These are nodes and edges, and the whole is a labeled, attributed graph. That much is not new; knowledge graphs and, in robotics, scene graphs already encode structured relations. What this report adds is the insistence that a physical world model cannot be only discrete. The world is continuous. Two surfaces are similar but not identical; a novel object resembles a known one without matching any category exactly; a sensor reading is a point in a high-dimensional space, not a symbol. So each node must also carry a vector, an embedding into a continuous space, and the store must support both kinds of question at once: *what is connected to this*, answered by traversal, and *what is like this*, answered by similarity search.

Holding both in one place, on the device that has to act rather than in a remote service, is the subject of the Charlot Lab's `hyperdb` effort. This report reviews the prior art the position rests on, describes the substrate, and states the research position plainly. It reports no benchmarks; `hyperdb` is early-stage research infrastructure, and where a claim is a design intention rather than a measured result the text says so.

2. Scope and method

This is a review and a research position, not an experimental report. It draws on the primary literature of two established fields — graph data models and knowledge graphs on one side, vector embeddings and approximate nearest-neighbor search on the other — and on the robotics work that has begun to join them in the form of spatial scene graphs. For each claim it cites a primary source, and each cited reference was checked to resolve to its publication. The report states no original measurements and reports no benchmark numbers for `hyperdb` or any other system. Where the text describes `hyperdb`, it describes an active, changing research project whose interfaces are not stable and whose performance is not characterized here; those descriptions are design intent, and are marked as such. The principal limitation of the report is that the hybrid store it advocates is a position about how embodied memory should be built, supported by the maturity of its two halves, rather than a demonstrated end-to-end result.

3. Entities as nodes, relationships as edges

The discrete half of the substrate is the property graph. In this model the data is a set of nodes and a set of directed, typed edges, and both nodes and edges carry properties as key–value attributes[1]. A node has a label — `Part`, `Material`, `Agent` — and attributes such as a mass or a friction coefficient; an edge has a type — `MADE_OF`, `CONTACTS`, `OBSERVED_BY` — and may itself carry attributes such as a contact force or a timestamp. This is a more direct encoding of related entities than the relational model, because the relationships are first-class

objects rather than values to be joined, and traversing from an entity to its neighbors is a local operation rather than a query over a whole table. The property-graph model and its query language Cypher were consolidated in industrial graph databases and later formalized[2,1]; a query in this style matches a pattern of nodes and edges and returns the bindings, so a question such as *which surfaces does the object in my gripper contact, and what are they made of* is expressed as a small graph pattern rather than a chain of joins.

When the graph is used not merely as storage but as a representation of what a system knows about the world, it is a knowledge graph. The recent survey by Hogan and colleagues defines a knowledge graph as a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between them, and reviews the techniques — schemas, identity, context, and both deductive and inductive methods for completing and refining the graph[3]. For an embodied system the relevant entities are physical rather than encyclopedic, but the structure is the same: a graph that grows as the system observes more, that can be queried for what it contains, and that can be extended by inference. The property graph gives the container; the knowledge-graph literature gives the discipline for using it as a model of the world.

4. A continuous world needs embeddings

A purely symbolic graph is brittle in exactly the way the physical world is not. Physical experience does not come pre-sorted into categories. A surface the system has never seen is not a member of any node it already holds, yet it is like surfaces it has held; a sensor reading is a vector in a high-dimensional space, not a token. The standard response, across natural-language processing and perception alike, is the embedding: a map from an item to a point in a continuous vector space arranged so that geometric proximity encodes similarity of meaning or appearance[4]. Words, images, surfaces, and sensor frames can all be embedded, and once they are, similarity becomes distance.

The usual measure is the cosine of the angle between two vectors, which compares direction independently of magnitude:

$$\operatorname{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \, \|\mathbf{v}\|} = \frac{\sum_{i=1}^{d} u_i v_i}{\sqrt{\sum_i u_i^2}\,\sqrt{\sum_i v_i^2}}.$$

Attaching such a vector to every node turns the graph into a continuous object as well as a discrete one. The node for a known part still has its label and its edges, but it also has a coordinate, and the coordinate lets the system ask questions the labels cannot answer: what have I seen that is most like this new thing, even though it matches no category exactly. This is the same operation that underlies retrieval-augmented generation, in which a query is embedded and used to retrieve the most similar stored items before further reasoning[5]. For an embodied system the stored items are not documents but its own accumulated experience of the physical world, and the retrieval is over the graph it has built.

5. One engine: traverse and similarity-search together, on the device

Similarity search over embeddings is only useful if it is fast, and exact nearest-neighbor search in high dimensions is not. The practical answer is approximate nearest-neighbor search, which trades a small, controllable loss of recall for a large gain in speed. The dominant method is the Hierarchical Navigable Small World graph of Malkov and Yashunin[6]. HNSW builds a layered proximity graph over the vectors: the upper layers are sparse and span long distances, the lower layers are dense and local, and a search descends from a coarse entry point through progressively finer layers, greedily following edges toward the query. The expected query cost scales logarithmically in the number of stored vectors,

$$T_{\text{search}} = O(\log N),$$

which is what makes similarity search over millions of items tractable, and it is the same construction — a navigable small-world graph — that GPU-scale similarity libraries build on for billion-scale corpora[7]. The important structural observation is that this index *is itself a graph*. The vector store and the property graph are not two different kinds of thing that happen to be co-located; they are both graphs, one whose edges encode semantic relations and one whose edges encode geometric proximity.

That observation motivates `hyperdb`, the Charlot Lab's research effort to hold both behind one query pipeline. `hyperdb` is an embeddable engine — a library, with no external server or service to run — that stores a property graph of typed nodes and edges and an HNSW index over node embeddings in the same store, and exposes a query pipeline that can interleave the two: begin from a similarity search, then traverse the relations of the results; or begin from a graph pattern, then rank its matches by similarity to a probe vector. Because it is embeddable and dependency-light, it can run on the device that is doing the perceiving and acting, rather than requiring a round trip to a remote database. These are the design goals of the project as it stands; the engine is under active development, its interfaces will change, and this report characterizes neither its throughput nor its recall. The point here is architectural, not empirical: an embodied system should not have to choose, at query time, between asking what is connected to a thing and asking what is like it, and it should not have to leave the device to ask either.

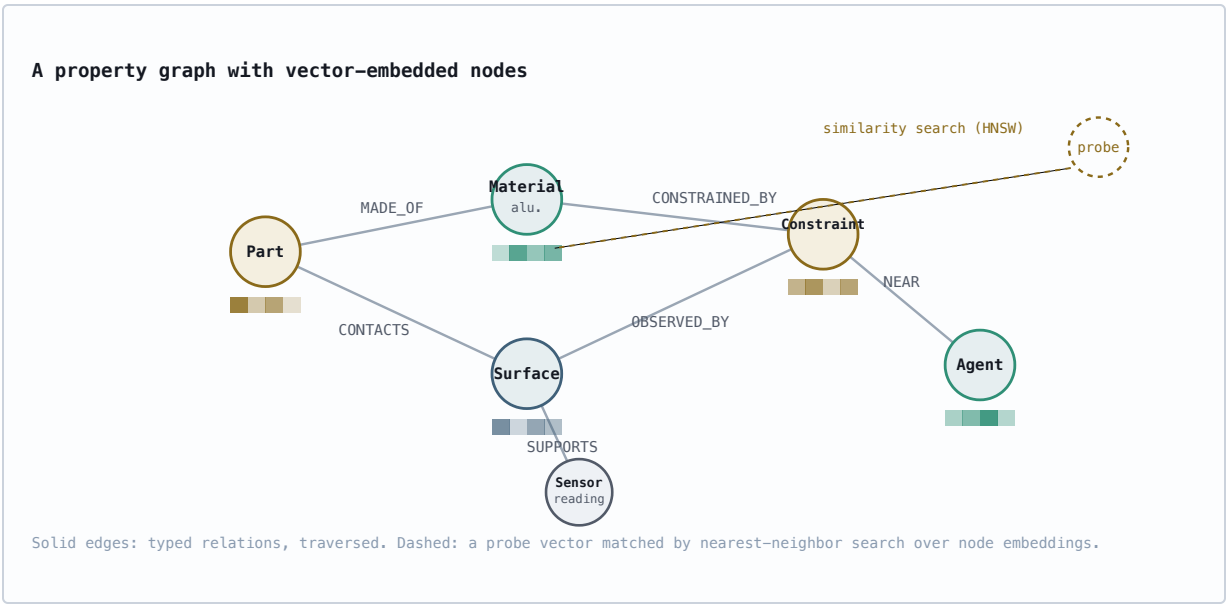

Figure 1. Entities are nodes and relationships are typed edges; every node also carries a vector embedding (the small swatches). A query can follow the solid relational edges by traversal and, in the same engine, find the node most similar to a probe vector by approximate nearest-neighbor search over the embeddings. An interactive version is at physicalai-bmi.org/research/charlot-lab.

6. What it holds for an embodied system

Concretely, the graph holds the furniture of embodied experience. A **part** is a node with a geometry and a mass. A **material** is a node with density, friction, and stiffness, joined to the parts made of it. A **surface** is a node a motion can contact, with a normal and a roughness. A **constraint** is a node — a joint limit, a no-collision region, a force budget — that a plan must respect, joined to the entities it constrains. Another **agent** is a node with a pose and a predicted trajectory. And a **sensor reading** is a node too: a timestamped observation, carrying its raw embedding, joined by an `OBSERVED_BY` edge to the entity it is evidence for and by a `SUPPORTS` edge to the belief it updates. The world model is the accumulation of these, and it is queried both ways at once — traverse from the gripper to the surfaces its held part contacts and the constraints those contacts impose, then rank a novel surface against the ones already known by embedding similarity.

This is the same structure that spatial-perception research in robotics has converged on independently. The 3D scene graph of Armeni and colleagues organizes a building into a layered graph of cameras, objects, rooms, and buildings with the relations among them[8], and the Hydra system of Hughes, Chang, and Carlone builds such a 3D scene graph in real time from sensor data and optimizes it online[9]. These systems show that a graph is a workable runtime representation of a perceived physical space, not only an offline knowledge store. Graph of the World takes the same commitment and adds the continuous half explicitly: every node in the spatial graph also carries an embedding, so recognition and retrieval are operations on the same store as traversal.

Within the Charlot Lab the substrate has two direct consumers. The Lab's perception work, OmniSense, produces the sensor-reading nodes and the entity nodes they support, writing the

discrete and continuous halves of each observation into the graph as it perceives. The Lab's CAD engine, CadFuture, consumes the same graph as its knowledge layer: a design is a graph of parts, materials, constraints, and their relations, and the vector half lets it retrieve prior geometry by similarity rather than by exact name. `hyperdb` is the store both use. The claim of this report is not that these integrations are complete — they are research in progress — but that the substrate they share is the right shape, because perception and design are both, at bottom, the construction and query of an attributed graph of the physical world with embeddings on its nodes.

Table 1. The two halves of the substrate and the questions each answers.

HALF	STRUCTURE	QUESTION IT ANSWERS	PRIMARY METHOD
Discrete	Property / knowledge graph: typed nodes, typed edges, attributes	What is connected to this? What is it made of, what constrains it, what observed it?	Graph traversal, pattern query[2,3]
Continuous	Vector embedding on each node, indexed for search	What have I seen that is most like this, even with no exact match?	Approximate nearest-neighbor (HNSW)[6]
Both, one engine	`hyperdb`: graph store and ANN index behind one query pipeline, embeddable, on-device	Traverse from the results of a similarity search, or rank the matches of a pattern by similarity	Hybrid query pipeline (this work)

7. Discussion and position

The position of this report is that the store is not neutral. It is common to treat the world model as a passive container and to put the interesting work in perception and planning, but the container's shape decides which questions are cheap. A relational table makes relationships expensive to follow; a pure vector index makes structure invisible; a pure symbolic graph makes similarity impossible. A physical world model needs both structure and similarity to be first-class, because embodied experience is both discrete and continuous, and the two must be queryable together and on the device, because an agent that has to leave its body to consult its memory is not operating in real time.

Two qualifications keep the claim honest. First, both halves are mature in isolation. The property-graph and knowledge-graph model is decades old and industrially deployed[1,3], and HNSW-based similarity search is a settled, widely used method[6,7]. The contribution claimed here is not either half but their union in one embeddable engine, and that union is a research effort, not a finished result. Second, the hybrid store is a substrate, not a solution. It makes the right questions cheap; it does not answer them. What perception writes, how beliefs are updated, how the graph is kept consistent as the world changes, and how stale nodes are forgotten are all open problems the substrate enables work on rather than settles.

8. Conclusion

Perception fills a world model and physics makes it actionable, but the model has to live somewhere, and this report has argued that it lives as a graph: entities are nodes, relationships are edges, and because the world is continuous each node also carries a vector so the system can traverse relationships and search by similarity in one place. The discrete half rests on the property-graph and knowledge-graph model; the continuous half rests on vector embeddings and approximate nearest-neighbor search; the contribution of the Charlot Lab's `hyperdb` effort is to hold both behind one query pipeline, embeddable and on the device that acts. That effort is early, and its performance is not characterized here. The durable point is architectural. An embodied system's memory should be a graph of the world with vectors on its nodes, and it should be able to ask what is connected and what is similar without choosing between them and without leaving the body.

References

1. R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, D. Vrgoč. Foundations of Modern Query Languages for Graph Databases. *ACM Computing Surveys* 50(5), 2017. doi:10.1145/3104031.
2. N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor. Cypher: An Evolving Query Language for Property Graphs. *Proc. 2018 Int. Conf. on Management of Data (SIGMOD)*, 2018. doi:10.1145/3183713.3190657.
3. A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann. Knowledge Graphs. *ACM Computing Surveys* 54(4), 2021. doi:10.1145/3447772.
4. T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781, 2013.
5. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, D. Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401, 2020 (NeurIPS 2020).
6. Yu. A. Malkov, D. A. Yashunin. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Analysis and Machine Intelligence* 42(4), 2020. doi:10.1109/TPAMI.2018.2889473; arXiv:1603.09320.
7. J. Johnson, M. Douze, H. Jégou. Billion-scale Similarity Search with GPUs. arXiv:1702.08734, 2017 (*IEEE Trans. Big Data*, 2021).
8. I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, S. Savarese. 3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera. *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019; arXiv:1910.02527.
9. N. Hughes, Y. Chang, L. Carlone. Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization. *Robotics: Science and Systems (RSS)*, 2022; arXiv:2201.13360.

AI-use disclosure. Preparation of this report used a large language model (Claude, Anthropic) for drafting and editing text, organizing the reviewed literature, and preparing the figures and the interactive companion. Cited references were checked to resolve to their sources. The author reviewed the content and is solely responsible for it. Consistent with ICMJE, COPE, and IEEE guidance, the model is a tool and is not credited as an author.

Institute for Physical AI @ BMI · The Charlot Lab
503 McKeever Rd, Arcola, TX 77583, USA

Technical Report TR-2026-12 · Preprint v1

physicalai-bmi.org · contact@physicalai-bmi.org

Released for open scholarly use. No proprietary or novel experimental results are reported.