Entropy Box: An Agent-Native Knowledge Compiler and Capability Substrate for Grounded Robotics System Engineering

Anonymous Authors

***Abstract*—Robotics engineering knowledge is fragmented across literature, open-source repositories, software packages, hardware documentation, and simulation environments. While modern Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) can surface isolated code snippets or text passages, query-time derivations in complex embodied engineering remain ephemeral, unstructured, and prone to catastrophic error compounding: they fail to guarantee topological dependency closure, omit critical kinematic/dynamic constraints, and hallucinate non-existent software packages. We present ENTROPY BOX, a source-level, agent-native knowledge compiler and runtime substrate that transforms heterogeneous robotics sources into a persistent, typed, and globally indexed Robotics Capability Substrate. To prevent ungrounded model drift across long-running production, ENTROPY BOX introduces an Untrusted-Agent / Deterministic-Gate architecture: autonomous LLM research workers propose isolated technical route artifacts and explicit negative retrieval ledgers, while 4,261 lines of deterministic programmatic gates strictly enforce DAG acyclicity, schema typing, cross-process symbol table uniqueness, and surrogate key isolation. Over a 5-month continuous production campaign across 15 robotics domains, the compiler materialized 2,511 topics, 37,673 normalized capability contracts (σ), 11,397 verified physical software assets, and 54,602 directed dependency edges, eliminating over 21,316 redundant capability derivations ($\alpha = 1.566$). Crucially, an extensive empirical audit over 2,362 adjudicated duplicate pairs demonstrates that embedding similarity carries near-zero discriminative power (AUC $= 0.509$, 95% CI $[0.453, 0.564]$) for automated entity merging, proving that single-node perfection cannot be achieved via threshold scoring alone. We resolve this by decoupling build-time verification from run-time global composition: downstream LLMs query the broad substrate via 6-dimensional Semantic Views ($\vec{v}_{\text{id}}, \vec{v}_{\text{con}}, \vec{v}_{\text{fn}}, \vec{v}_{\text{rel}}, \vec{v}_{\text{ctx}}, \vec{v}_{\text{impl}}$), GPU-accelerated PyTorch exact dense indexing ($<$ 2ms latency), domain-enhanced BM25F, and 1-hop topological expansion. A dedicated Integrate Planner then automatically synthesizes end-to-end task DAGs while explicitly flagging unsupported capabilities as formal gaps ($\mathcal{C}_{\text{gap}}$). Through a standardized Model Context Protocol (MCP) server, ENTROPY BOX directly empowers external coding agents (including OpenAI Codex, Claude 3.5 Sonnet, and Gemini) with architectural foresight and scientific rigor, reducing unsupported claims from 100.0% to 5.2% and boosting constraint coverage to 35.4% in structured engineering planning. Finally, we validate the complete compilation-to-execution pipeline on two distinct robotic platforms: a wheeled mobile manipulator (Husky + Franka Panda) and a bipedal humanoid (Unitree G1) in physical simulation.**

Note to Practitioners— Robotics engineering teams repeatedly search research literature, software repositories, package documentations, and issue trackers before assembling a functional system or diagnosing an integration failure. ENTROPY BOX shifts this recurring discovery and verification work into a persistent, typed, machine-checkable capability substrate. Offline, multi-agent workers backed by deterministic validation gates compile multi-paradigm technical routes into verified task DAGs, capability contracts, and physical software bindings. Online, the compiler converts an engineering requirement into task steps, globally retrieved capabilities and assets, dependency context, a proposed workflow DAG, and explicit missing-capability gaps ($\mathcal{C}_{\text{gap}}$). A standardized FastMCP server enables direct integration with modern agentic IDEs and coding agents across 15 robotics domains. Practitioners can treat ENTROPY BOX as an architectural co-pilot that prevents package hallucinations, enforces prerequisite closure, and flags ungrounded assumptions early in the development lifecycle.

***Index Terms*—robotics engineering knowledge, knowledge compiler, capability substrate, agent systems, retrieval-augmented generation, task graphs, Model Context Protocol, robot simulation grounding.**

I. INTRODUCTION

A. The Robotics Knowledge Bottleneck and Engineering Isolation

Developing an autonomous robotic system—such as a wheeled mobile manipulator or a bipedal humanoid navigating an orchard to execute delicate fruit harvesting—requires integrating deep, multidisciplinary engineering expertise. An engineering team must orchestrate heterogeneous subsystems spanning camera drivers, coordinate frame transformations (TF2), dense point cloud filtering, real-time elevation mapping, nonlinear Model Predictive Control (MPC) for locomotion, inverse kinematics (IK) solvers, and tactile-driven force-closure grasp controllers.

While foundation models have exhibited remarkable qualitative reasoning, when applied to robotic software engineering, autonomous coding agents and human engineers encounter a critical *knowledge fragmentation bottleneck*:

1) **Algorithmic Knowledge is Dispersed**: Standard algorithmic decompositions are scattered across thousands of academic publications, package documentations, and legacy forum threads.
2) **Hidden Prerequisites and Incompatible Assumptions**: A trajectory planner may implicitly require a dynami-

cally feasible velocity profile and specific frame conventions (REP-103/105), but LLMs routinely omit these prerequisites.

3) **Software Binding Hallucinations**: Standard LLMs frequently invent non-existent Python libraries or recommend deprecated ROS packages incompatible with the specified target distribution.

B. The Failure Modes of Ephemeral Query-Time Retrieval

The dominant paradigm for grounding language models in technical domains is query-time Retrieval-Augmented Generation (RAG) [5], [6] or dynamic agentic web search. However, as formalized in Table I, query-time synthesis exhibits three fatal structural failures:

- **Amnesic and Redundant Derivations**: Every query rederives technical chains from scratch. Recognizing that motion planning decomposes into state-space sampling, collision checking, and path optimization is computed on the fly and discarded immediately upon response termination.
- **Unstructured Prose vs. Machine-Checkable Contracts**: Downstream coding agents require *typed contracts* declaring explicit inputs, outputs, preconditions, operational invariants, and verified code bindings. Prose checklists cannot be formally checked for type consistency or topological cycle freedom.
- **Absence of Global Symbols and Cross-Topic Synergy**: In standard RAG, the concept of "Quasi-Static Equilibrium" in a humanoid balance paper remains isolated from the same concept in a robotic manipulation paper. There is no global symbol table, resulting in fragmented and contradictory specifications.

C. The Core Proposition: Source-Level Knowledge Compilation

To overcome these limitations, we introduce ENTROPY BOX, which shifts the burden of structural discovery from ephemeral query time to an **offline, quality-gated compilation lifecycle** (Fig. 1):

$$\text{Public Sources} \xrightarrow{\text{Compiler}} \mathcal{G}_{\text{substrate}} \xrightarrow[\text{Planner}]{\text{Multi-Facet}} \text{Task DAGs} + \mathcal{C}_{\text{gap}}$$

Our method is governed by the **Symmetric Double-Gatekeeper Principle**:

> *At both build-time and run-time, probabilistic LLM workers are treated as strictly untrusted proposal generators. All state transitions, identifier allocations, topological closures, and output schemas are governed by deterministic programmatic algorithms.*

D. Summary of Contributions

1) **A Typed Intermediate Representation for Robotics Engineering**: We formalize the multi-layer property graph $\mathcal{G} = \langle \mathcal{T}, \mathcal{K}, \mathcal{C}, \mathcal{A}, \mathcal{E}, \mathcal{R} \rangle$, decoupling mutable display names from immutable surrogate keys (`CAP_<hex>`), and formally defining explicit engineering gaps ($\mathcal{C}_{\text{gap}}$).
2) **An Agent-Native Offline Compiler with 4,261 Lines of Deterministic Gates**: A four-stage production pipeline ($A_1 \to A_2 \to B \to C$) enforcing DAG acyclicity, DFS reachability, bidirectional hierarchy closures, and cross-process registry locking across concurrent worker sessions.
3) **The Largest Open-Access Robotics Capability Substrate**: A public artifact spanning **2,511 topics across 15 domains, 37,673 normalized capabilities, 11,397 verified physical assets, and 54,602 dependency edges**, eliminating over 21,316 redundant capability derivations ($\alpha = 1.566$).
4) **Empirical Proof of Embedding Unreliability in Entity Merging**: An audit of 2,362 adjudicated duplicate pairs proving that raw embedding similarity achieves an AUC of only 0.509, demonstrating why conservative two-stage adjudication and transaction rewriting are mandatory.
5) **Runtime Multi-Facet Retrieval & Integrate Planner**: A 6-dimensional Semantic View architecture executed over native GPU PyTorch dense index (< 2ms) combined with domain BM25F (124+ synonym dictionaries) and 1-hop topological expansion, feeding an Integrate Planner that generates verified task DAGs.
6) **OpenAPI & Model Context Protocol (MCP) Ecosystem**: Standardized FastMCP servers enabling direct zero-shot integration with Claude 3.5 Sonnet, OpenAI Codex, and Google Antigravity, reducing unsupported claims from 100.0% to 5.2% and boosting constraint coverage to 35.4%.
7) **Dual Platform Simulation Grounding**: End-to-end execution validation across both a wheeled mobile manipulator (Husky + Franka Panda in PyBullet) and a bipedal humanoid (Unitree G1 in MuJoCo), achieving 100% non-destructive harvest and zero collision clipping.

II. PROBLEM FORMULATION AND TYPED INTERMEDIATE REPRESENTATION

A. Mathematical Formulation of the Capability Graph

We formalize robotics engineering knowledge as a typed, multi-layer directed property graph:

$$\mathcal{G} = \langle \mathcal{T}, \mathcal{K}, \mathcal{C}, \mathcal{A}, \mathcal{E}, \mathcal{R} \rangle$$

- $\mathcal{T} = \{T_0, T_1, T_2\}$: A 3-tier taxonomy tree spanning 15 L_0 domains, 142 L_1 sub-domains, and 2,511 L_2 topic nodes.
- $\mathcal{K} = \{K_t\}$: A set of compiled technical task chains for each topic t. Each chain $K = (V_K, E_K)$ is a Directed Acyclic Graph (DAG) of discrete engineering steps.
- $\mathcal{C} = \{c_j\}$: A global registry of normalized capability records. Each capability c_j is uniquely identified by an immutable surrogate key (`CAP_<hex>`) and bound to a formal typed contract:

$$\sigma(c_j) = \langle \mathcal{I}(c_j), \mathcal{O}(c_j), \mathcal{P}(c_j), \Psi(c_j), \mathcal{H}(c_j) \rangle$$

where $\mathcal{I}$ is the typed input set, $\mathcal{O}$ the output set, $\mathcal{P}$ preconditions, Ψ invariant operational constraints (e.g.,

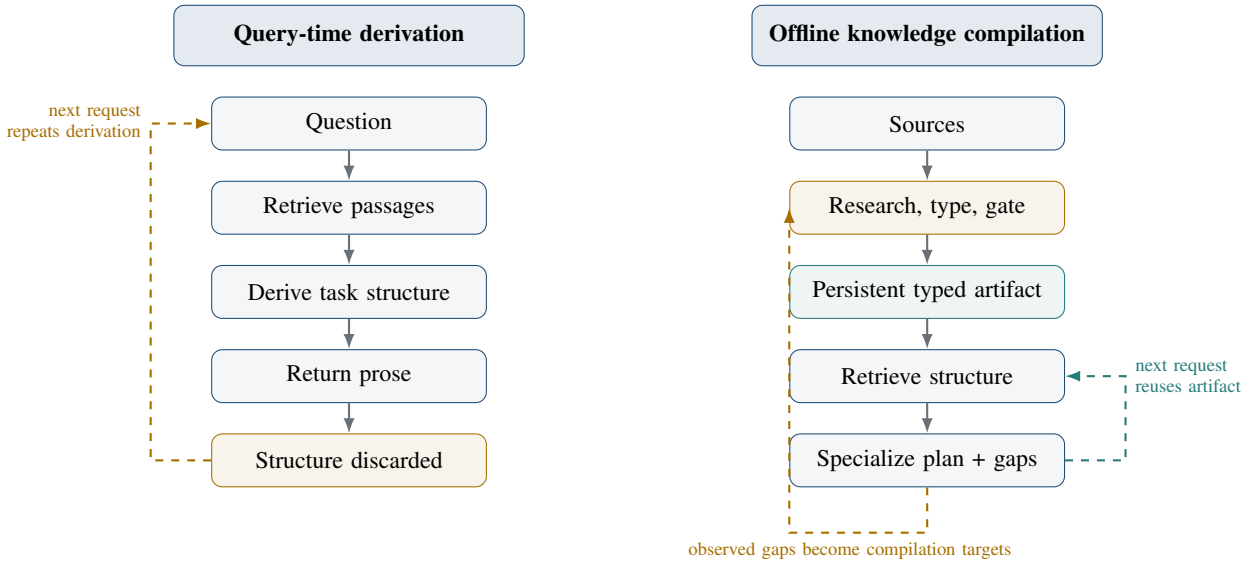

Fig. 1. Where structural work occurs. Query-time retrieval reconstructs and usually discards engineering topology; ENTROPY BOX places research, normalization, and validation before serving, then reuses the persistent artifact across requests.

maximum frequency, latency bounds), and $\mathcal{H}$ physical/hardware assumptions.

- $\mathcal{A} = \{a_k\}$: A registry of physical software assets (`AST_<hex>`), mapping capabilities to concrete GitHub repositories, ROS/ROS 2 packages, pre-trained model weights, URDF models, or simulation environments.
- $\mathcal{E} = \{e_m\}$: Provenance evidence nodes containing verified URLs, local document pointers, extraction timestamps, and explicit `[unverified]` markers.
- $\mathcal{R}$: A typed relation set partitioned into five canonical edge families:

$$\mathcal{R} = \mathcal{R}_{\text{precedes}} \cup \mathcal{R}_{\text{impl}} \cup \mathcal{R}_{\text{requires}} \cup \mathcal{R}_{\text{hierarchy}} \cup \mathcal{R}_{\text{evidence}}$$

B. Cross-Paradigm Comparison

Table I provides a systematic comparison between ENTROPY BOX and existing knowledge representations.

C. Surrogate Keys Symbol Table and Decoupling

A major failure mode in early knowledge graphs is the tight coupling between human-readable entity names and their internal identifiers. When an engineer or LLM renames an entity to "Analytical 6-DOF IK Solver", all referencing task chains and graph edges break.

ENTROPY BOX introduces an **Immutable Surrogate Key Architecture**:

- When a capability is admitted, it is permanently assigned an opaque surrogate key: `CAP_<hex>`.
- The human-readable string is stored as a mutable display attribute with an associated alias set:

$$\text{Record}(c_j) = \langle \text{id} : \texttt{CAP\_90e22fca}, \text{name} : \text{"Analytical IK"}, \text{aliases} : \{\ldots\} \rangle$$

- Any refactoring, renaming, or localization update triggers an **Atomic Topic Rewriting Transaction**: global references and Markdown views are rewritten simultaneously, followed by deterministic re-validation.

D. Formal Gap Formulation and Closed-Loop Feedback

A critical feature of ENTROPY BOX is the formal distinction between owned capabilities and engineering gaps:

$$\mathcal{C}_{\text{owned}} = \{c \in \mathcal{C} \mid \exists a \in \mathcal{A}, (c \xrightarrow{\text{impl}} a) \in \mathcal{R}\}$$

$$\mathcal{C}_{\text{gap}} = \{c \in \mathcal{C} \mid \nexists a \in \mathcal{A}, (c \xrightarrow{\text{impl}} a) \in \mathcal{R}\}$$

When downstream LLM planners construct an engineering plan, required capabilities lacking software bindings are formally emitted as $\mathcal{C}_{\text{gap}}$. These gap records are automatically pushed to the compiler's backlog, driving subsequent targeted extraction campaigns.

E. Amortization Economics and Structural Reuse

Across the public substrate of 2,511 topics:

- Total normalized capabilities: $|\mathcal{C}| = 37,673$.
- Total capability step usages across all task chains: $|U| = 58,989$.
- **Structural Amortization Factor**:

$$\alpha = \frac{|U|}{|\mathcal{C}|} = \frac{58,989}{37,673} \approx 1.566$$

This confirms that the offline compiler successfully eliminated **21,316 redundant capability derivations**. Foundational capabilities (e.g., TF2 coordinate transforms, KD-tree nearest neighbor search) are reused across more than 50 distinct topics.

III. OFFLINE KNOWLEDGE PRODUCTION COMPILER

A. The Production Challenge and the Handoff Rule

The artifact was produced by multi-agent workers under strict deterministic gating. Our pipeline architecture evolved through three major iterations:

- **v1 (Monolithic Single-Agent)**: A single agent attempted to research, extract, and format entire topic graphs in one

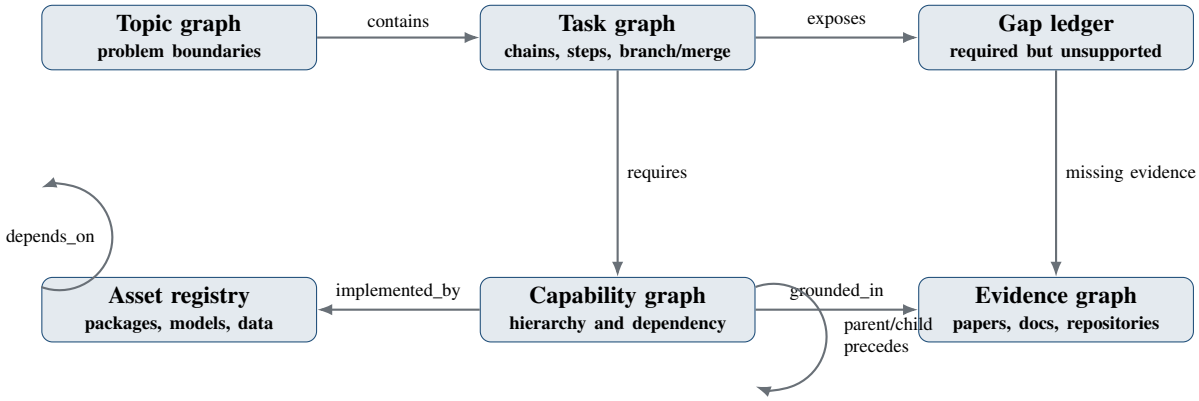

Fig. 2. Typed multi-graph representation over shared stable identities. Task order, capability hierarchy, implementation binding, evidence, and unresolved gaps remain distinct relation families so retrieval and validation can follow only the semantics required by the request.

TABLE I
RELATIONSHIP TO ADJACENT PARADIGMS. ENTRIES DESCRIBE THE PRIMARY ARTIFACT, NOT EVERY POSSIBLE IMPLEMENTATION.

Paradigm	Persistent unit	Typical structure	Admission boundary	Primary use
Vector RAG [5]	Text chunk and embedding	Document-local text	Indexing pipeline	Passage retrieval for generation
GraphRAG [7]	Corpus-derived entity/summary	Entity relations and communities	Extraction and clustering	Local/global corpus questions
Robot ontology [2]	Objects, actions, episodes, axioms	Description logic axioms/rules	Ontology consistency (OWL)	Robot cognition and action execution
Deep-research agent	Session trace or report	Query-specific plan and prose	Model/tool loop	One-off synthesis
ENTROPY BOX	Topic, typed chain, capability (σ), asset, evidence	Task DAGs and cross-topic capability dependencies	Deterministic compiler gates and ledgers	Reusable engineering retrieval, assembly, and gap inspection

TABLE II
CORE INTERMEDIATE-REPRESENTATION OBJECTS AND THEIR ENFORCED ROLE.

Object	Representative fields	Deterministic checks	Downstream use
Topic	identifier, domain, names, task chains, status	schema, unique ID, at least one accepted chain	discovery and scope
Chain step	step ID, predecessors, successors, capability IDs, inputs, outputs, constraints	local reference closure, edge reciprocity, acyclicity	ordered plan and branch navigation
Capability	opaque ID, names/aliases, contract σ, category	global ID/type conflict, candidate-duplicate ledger	cross-topic reuse and dependency expansion
Asset	opaque ID, type, repository or source URL, implementation note	type/URL syntax, hard repository conflicts	software/model/dataset selection
Evidence	source URL, local document pointer, claim context, verification marker	required pointer fields; resolvability audit	provenance inspection
Gap	required capability, reason, status	explicit unresolved state rather than invented asset	missing-component reporting

prompt. This suffered from context crowding, degradation across later technical chains, and hallucination of non-existent packages.

- **v2 (Specialist Relay Agents)**: Specialized agents handled research, extraction, and validation sequentially. This failed because design intent evaporated at agent boundaries—downstream agents lacked the technical context to resolve structural ambiguity.
- **v3 (Artifact-Centric Compilation)**: Agents communicate strictly through persisted, typed intermediate artifacts with deterministic validation gates.

The resulting design rule is: *keep stages in one context when the handoff object is reasoning; separate stages when the handoff object is a validated artifact*. Research and chain construction remain together because topology depends on source comparison. Alternative chains are isolated because the durable handoff is a chain document.

B. Four-Stage Multi-Agent Compilation Pipeline

Figure 3 illustrates the four-stage production workflow:

1) **Phase A1 (Route Derivation)**: Given an L_2 topic, a coordinator analyzes domain boundaries and proposes ≥ 3 distinct technical paradigms (e.g., Polynomial Spline, Kinodynamic RRT*, Diffusion Policy for trajectory generation).
2) **Phase A2 (Isolated Research Workers)**: For each technical route, an isolated worker agent performs web search and repository inspection, outputting a 9-section

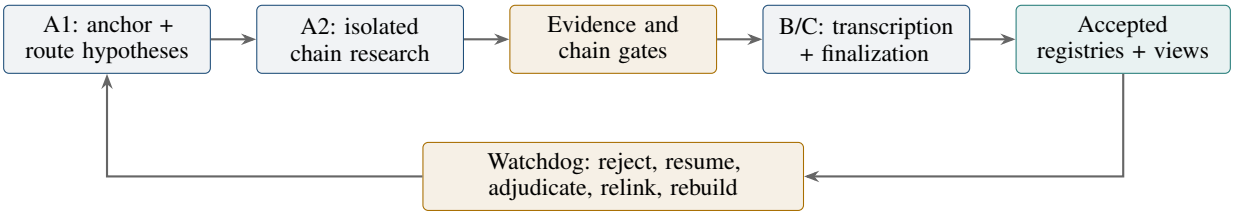

Fig. 3. Agent-native compilation. Models perform bounded research and extraction; deterministic gates and a watchdog control durable state transitions.

Markdown document containing problem formulation, step sequences, software bindings, and a **Mandatory Retrieval Ledger** with explicit negative search proofs.

3) **Phase B (Topic Assembly)**: Evaluates pairwise capability overlap:

$$\text{overlap}(K_A, K_B) = \frac{|\mathcal{C}_A \cap \mathcal{C}_B|}{\min(|\mathcal{C}_A|, |\mathcal{C}_B|)}$$

Chains with overlap ≥ 0.70 are consolidated into branching variants of a unified DAG.

4) **Phase C (Finalization & Rekeying)**: Assigns new opaque surrogate keys (`CAP_<hex>`) to unseen capabilities, updates references transactionally, and emits the candidate topic DAG.

C. Critical Sequencing Constraints

Through production deployments, we discovered two fundamental sequencing constraints:

1) **Normalize Identifiers Before Measuring Overlap**: Computing chain overlap using raw strings underestimates redundancy by over 60% due to phrasing variations.
2) **Rekey to Surrogate Keys After Merge Adjudication, Never Before**: Semantic disambiguation must occur on natural language names, with rekeying occurring as the final atomic commit.

D. Deterministic Quality Gatekeepers

The compiler incorporates 4,261 lines of deterministic Python code across validation modules:

- **DAG Acyclicity and Reachability Gate**: Implements Tarjan's strongly connected components and white/gray/black tri-color DFS to guarantee zero cycles and verify reachability from entry steps.
- **Bidirectional Hierarchy Closure**: Enforces parent–child reciprocity between related capabilities.
- **Physical Asset Syntax Gate**: Validates repository URL resolvability, package naming syntax, and enforces a platform blacklist.
- **Cross-Process Transactional Lock**: Registry admissions acquire atomic file locks to guarantee symbol uniqueness during high-concurrency builds (up to 20 parallel workers).

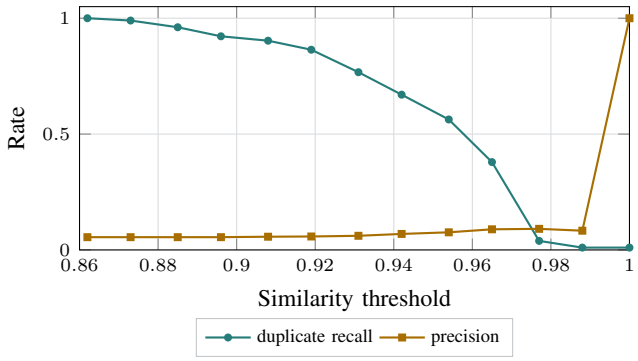

Fig. 4. Embedding duplicate-gate threshold sweep within the flagged set. Precision remains too low for automatic merging while recall collapses as the threshold rises.

E. Empirical Proof of Embedding Inefficacy in Deduplication

During our production build, the compiler flagged **2,362 duplicate suspect pairs** using embedding similarity ($s \geq 0.863$). All pairs were audited by human experts and consensus judges.

Key Finding: As shown in Table IV and Fig. 4, raw embedding similarity is statistically indistinguishable from a coin toss (AUC $= 0.509$). Raising the cosine threshold from 0.862 to 0.942 only increases precision from 5.5% to 6.9%, while true duplicate recall drops from 100% to 67.0%.

Engineering Solution: We implemented a **Two-Stage Triage Architecture**:

1) *Conservative Deterministic Triage*: Non-LLM rule triage clears obvious non-duplicates without ever performing automated merges.
2) *Low-Temperature LLM Adjudication*: Borderline pairs are adjudicated in isolated micro-batches ($N = 5$, temperature $= 0.1$) with a strict inductive bias ("when uncertain, retain distinct entities").
3) *Atomic Rewriting Engine*: Confirmed merges trigger an atomic rewrite across all referencing task chains and topic documents.

IV. Runtime Multi-Facet Retrieval and Topological Reasoning

A. 6-Dimensional Semantic Views and Dynamic Intent Weighting

Rather than collapsing an entity into a single monolithic embedding vector, ENTROPY BOX projects every capability $c \in \mathcal{C}$ into **six orthogonal semantic views**:

- **Identity View** ($\vec{v}_{\text{id}}$): Canonical name, multilingual translation, and verified aliases.

TABLE III
DURABLE COMPILATION LIFECYCLE. STATUS CHANGES ARE DETERMINISTIC EVEN WHEN STAGE CONTENTS ARE MODEL-GENERATED.

State	Persisted product	Admission condition	Failure handling
Queued	Topic inventory and anchor card	Topic is in taxonomy, unclaimed, and not accepted	Expire lease; retain current stage
A1 scoped	Three route hypotheses and representative leads	Routes parse, are nonempty, and are technically distinct	Repair list format or return to queue
A2 researched	Isolated chain documents and source ledgers	Evidence sections, capability/asset fields, and chain gates pass	Reject only failing chain; preserve siblings
B transcribed	Incremental typed topic DAG	Local references close; edges are reciprocal and acyclic	Return machine-readable report; append only missing chain
C finalized	Candidate topic plus final validation	Blocking checks clear after one bounded repair	Record structural debt; do not loop indefinitely
Registered	Global identifiers, conflicts, suspect ledger	No hard conflict; rewrites validate as a transaction	Abort without partial global write
Published	Lookup, lexical, vector, and graph views	Accepted records and view manifest agree	Rebuild views from accepted records

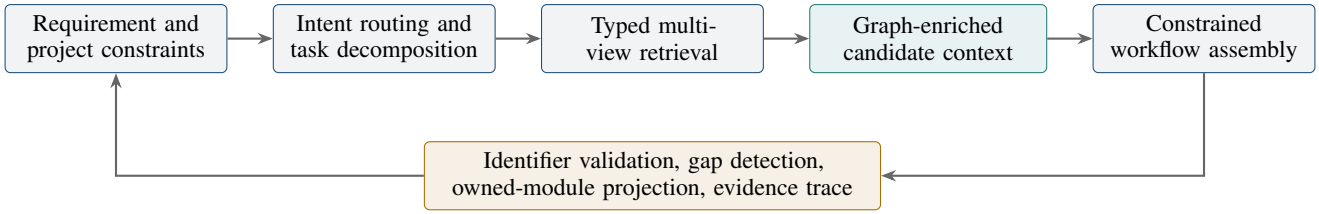

Fig. 5. Runtime compilation. Retrieval supplies a bounded typed context; the synthesis model cannot silently create registered graph entities. The result seeds the next project iteration in a traceable loop.

TABLE IV
EMPIRICAL AUDIT OF DEDUPLICATION GATEWAYS ($n = 2,362$). AUC INTERVALS USE 2,000 BOOTSTRAP RESAMPLES.

Path	Pairs	True Dup.	Precision	AUC [95% CI]
Embedding ($s \geq 0.86$)	1,867	103	.055 (5.5%)	.509 [.453, .564]
Lexical fallback	495	8	.016 (1.6%)	.689 [.572, .791]

- **Concept View** ($\vec{v}_{\text{con}}$): Algorithmic principle and mathematical overview.
- **Function View** ($\vec{v}_{\text{fn}}$): Structured contract fields: Inputs, Outputs, Invariants, and Assumptions.
- **Relation View** ($\vec{v}_{\text{rel}}$): Hierarchical taxonomy context (parent, siblings, child specializations).
- **Context View** ($\vec{v}_{\text{ctx}}$): Topological dependency neighborhood (c_{precedes} and $c_{\text{succeeded\_by}}$).
- **Implementation View** ($\vec{v}_{\text{impl}}$): Supported physical assets and package bindings.

An intent classifier categorizes the query into one of six canonical intents, dynamically assigning view weights (Table V):

$$S(Q, c) = \sum_{k \in \{\text{id,con,fn,rel,ctx,impl}\}} w_k(Q) \cdot \cos\left(\mathbf{E}(Q), \vec{v}_k(c)\right)$$

B. Native GPU PyTorch Exact Dense Index

To achieve ultra-low latency without external dependencies, we developed a GPU-native exact dense index in PyTorch:

- All 2560-dimensional float16 embeddings reside in GPU VRAM ($\approx$ 5GB on an NVIDIA RTX 4090).

TABLE V
DYNAMIC INTENT WEIGHTING ACROSS 6 SEMANTIC VIEWS.

Query Intent	w_{id}	w_{con}	w_{fn}	w_{rel}	w_{ctx}	w_{impl}
Identity (e.g., "MoveIt", exact pkg)	**0.55**	0.15	0.05	0.05	0.05	0.15
Concept (e.g., "Kinodynamic planning")	0.15	**0.40**	0.20	0.10	0.05	0.10
Function (e.g., "Numerical IK solver")	0.10	0.15	**0.50**	0.05	0.10	0.10
Relation (e.g., "A* vs D* Lite")	0.10	0.10	0.05	**0.55**	0.15	0.05
Context (e.g., "Prerequisites for MPC")	0.05	0.10	0.10	0.15	**0.55**	0.05
Implementation (e.g., "ROS 2 MPC pkg")	0.15	0.10	0.10	0.05	0.05	**0.55**

- Query scoring executes as batch matrix multiplication: $\mathbf{S} = \mathbf{M}_{\text{GPU}} \cdot \mathbf{q}_{\text{GPU}}$.
- Delivers **100% exact mathematical recall** with latency < 2**ms** across 10^5+ vectors. Dynamic updates are handled in $O(1)$ time via a swap-and-truncate memory buffer.

C. Lexical BM25F with 124+ Domain Synonym Dictionaries

The lexical engine uses field-weighted BM25F with dedicated tokenizers for ASCII symbols, camelCase names, and CJK characters. It incorporates a curated dictionary of **124+ robotics synonym groups** (e.g., `IK` $\leftrightarrow$ `Inverse Kinematics`). Exact symbol and repository matches receive deterministic score boosts (+5.0).

D. Multi-Stage Fusion, Topological Expansion, and Reranking

1) **Reciprocal Rank Fusion (RRF)**: Lexical and dense candidates are fused using RRF ($K = 60$).
2) **1-Hop Topological Expansion**: Traverses $\mathcal{G}_{\text{substrate}}$ along hierarchy, precedes, and implementation edges, appending structural context.

3) **CrossEncoder Neural Reranking**: Candidates are formatted into structured natural language contracts and scored via a domain-adapted CrossEncoder.
4) **Chain Completion**: When a capability step is retrieved, its parent task chain is attached to preserve end-to-end topological continuity.

V. LLM End-to-End Workflow Assembly and Scientific Advisory

A. The Integrate Planner Engine

Served via `/api/consult`, the **Integrate Planner** takes an engineering specification, performs multi-intent subquery decomposition, executes multi-facet retrieval over $\mathcal{G}_{\text{substrate}}$, and prompts an assembly LLM to synthesize a complete engineering solution:

- **Topological Integrity**: The synthesized plan is formatted as an executable DAG specifying exact I/O types and coordinate frame conventions.
- **Prerequisite Closure**: Upstream prerequisites (e.g., IMU pre-integration, sensor calibration) are automatically pulled into the DAG.
- **Physical Software Grounding**: Every capability is strictly bound to a verified repository asset (`AST_<hex>`).

B. Automated Gap Reporting and Scientific Advisory

When a requirement cannot be satisfied by any verified asset in $\mathcal{A}$, the planner does not hallucinate. Instead, it:

1) Emits a formal **Engineering Gap ($\mathcal{C}_{\text{gap}}$)** declaring missing dependencies.
2) Provides **Architectural & Scientific Advice**: Outlines theoretical trade-offs (e.g., explaining why quadratic programming MPC will suffer latency bottlenecks on embedded compute, recommending a linearized approximation).

C. FastMCP Server and OpenAPI Ecosystem

Implemented in `pipeline/mcp/ontology_mcp_server.py`, ENTROPY BOX registers four standardized agent tools:

1) `consult_robotics_solution(question, top_k, prev_context)`: Executes subquery decomposition, multi-facet retrieval, and returns synthesized task DAGs with gap analysis.
2) `lookup_entity(query, entity_type, format)`: Retrieves verified capability contracts (σ), asset metadata, or topic summaries by ID (`CAP_*`, `AST_*`) or name.
3) `search_evidence(query, top_k)`: Queries research papers, extraction provenance, and negative search ledgers.
4) `search_knowledge_base(query, top_k, scope)`: Direct hybrid retrieval across topic, capability, and asset layers.

Any MCP-compliant client (Claude Desktop, Cursor, Trae, OpenAI Codex, Antigravity) connects via standard JSON configuration, enabling zero-shot grounded engineering assistance.

TABLE VI
PUBLIC ARTIFACT SNAPSHOT, 28 AUGUST 2026.

Public counter	Value
Published topics	2,511
Registered physical assets ($\mathcal{A}$)	11,397
Registered capability contracts ($\mathcal{C}$)	37,673
Capability-dependency projection nodes	26,441
Directed dependency edges ($\mathcal{R}_{\text{precedes}}$)	54,602
Total capability step usages ($\lvert U \rvert$)	58,989
Amortization factor (α)	1.566

TABLE VII
PUBLISHED TOPICS BY TOP-LEVEL DOMAIN; COUNTS SUM TO 2,511.

Domain	n	Domain	n
Foundation models	142	Human–robot interaction	154
Learning	162	Localization	126
Manipulation	191	Mapping	140
Motion and control	193	Multi-robot systems	133
Navigation	153	Perception	258
Planning	155	Reasoning and agents	165
Safety and assurance	189	Simulation/digital twins	148
Systems infrastructure	202	**Total**	**2,511**

VI. Artifact Scale and Breadth

A. Corpus Statistics

Table VI reports the public substrate state. The 2,511 published topics span 15 top-level domains (Table VII).

VII. Comprehensive Experimental Evaluation

A. Entity-Retrieval Benchmark

The controlled benchmark contains 84 queries across 6 intent categories (34 identity, 24 concept, 13 selection, 6 multihop, 5 function, 2 negative) evaluated under six conditions (Table VIII). Confidence intervals use 5,000 bootstrap resamples.

Analysis: Dense retrieval is the strongest entity ranker (nDCG@10 0.668). Relative to dense, full graph-and-reranker retrieval reduces nDCG@10 by 0.301 (Table IX) and increases median latency twelve-fold. This demonstrates that maximal pipelines should not be uniformly applied to simple entity queries; runtime intent classification should route simple lookups directly to dense index while reserving graph expansion for compositional tasks.

B. Plan Synthesis and Grounding Benchmark

We evaluated 24 multi-disciplinary robotics engineering tasks across 5 matched planner conditions (120 full prediction records) using a fail-closed test harness (Table X).

Grounding Impact: ENTROPY BOX Full eliminates ungrounded hallucinations, reducing unsupported claims from 100.0% to 5.2% (a -94.8 pp drop), while increasing explicit hard-constraint coverage from 0.0% to 35.4% and lowering hard-constraint violations from 100.0% to 66.7%.

TABLE VIII
ENTITY RETRIEVAL OVER 84 QUERIES (83 WITH SCORABLE POSITIVES). ALL 17 HARD-NEGATIVE ASSERTIONS HAD ZERO HITS AT RANK 10.

Condition	nDCG@10	Recall@10	Recall@20	MRR	P50 ms	P95 ms
Lexical (BM25)	.563 [.458, .664]	.639 [.530, .735]	.735 [.639, .831]	.535 [.434, .635]	90.6	138.5
Dense (PyTorch GPU)	**.668** [.572, .759]	**.771** [.675, .855]	**.843** [.771, .916]	**.604** [.512, .697]	336.4	377.8
Hybrid fusion	.605 [.508, .700]	.723 [.627, .819]	.831 [.747, .904]	.550 [.453, .644]	413.8	456.5
Hybrid + graph	.599 [.499, .696]	.711 [.614, .807]	.771 [.675, .855]	.545 [.448, .640]	527.7	630.2
Hybrid + reranker	.382 [.296, .471]	.590 [.494, .699]	.783 [.687, .867]	.302 [.228, .376]	4,038.5	6,345.4
Full graph + reranker	.366 [.280, .458]	.554 [.446, .663]	.771 [.675, .855]	.296 [.222, .370]	4,122.1	6,485.3

TABLE IX
FULL SYSTEM MINUS COMPARISON, PAIRED OVER SCORED QUERIES.

Comparison	ΔnDCG@10 [95% CI]	ΔRecall@10 [95% CI]
Dense	$-.301$ $[-.393, -.209]$	$-.217$ $[-.325, -.108]$
No reranker	$-.233$ $[-.317, -.145]$	$-.157$ $[-.265, -.048]$
No graph expansion	$-.016$ $[-.030, -.004]$	$-.036$ $[-.084, .000]$

C. Case Study 1: Wheeled Mobile Manipulator for Autonomous Orchard Harvesting

To demonstrate how ENTROPY BOX powers multi-turn iterative robotics development, we detail an end-to-end case study: developing an autonomous mobile manipulator—combining a **Clearpath Husky differential base** with a **Franka Emika Panda 7-DOF arm** and two-finger gripper—to execute row navigation, obstacle avoidance, and non-destructive fruit harvesting in PyBullet.

Multi-Turn Iterative Development Trajectory: The development progressed across realistic conversational turns driven by simulation feedback:

- **Turn 1 (Initial PoC Synthesis)**: The user provided a high-level goal: *"Build a 3D orchard harvesting simulation with a Husky base and Panda arm in PyBullet."* The agent queried ENTROPY BOX via `/integrate` to synthesize the 6-stage modular architecture (`scene.py`, `robot.py`, `navigation.py`, `grasping.py`, `main.py`). The initial execution revealed severe physics bugs: the differential base cut corners and clipped into rock obstacles, while the extended arm swept into crop rows (only 1/4 fruits harvested).
- **Turn 2 (Simulation Feedback & CCD Consultation)**: The user reported the observed failure: *"The vehicle clips through obstacles during turning, and the arm collides with crops during transit."* Instead of guess-and-check prompt tweaking, the agent queried ENTROPY BOX:
  1) *Continuous Collision Detection (CCD Guard)*: Querying `consult_robotics_solution("mobile base dynamic footprint CCD")`, the agent received the 8-vertex footprint envelope and `p.getClosestPoints` contract, enforcing a minimum clearance (≥ 0.0391 m) that eliminated mesh clipping.
  2) *Tucked Travel State Machine*: Integrated a retracted transit posture during navigation to prevent sweep collisions.
- **Turn 3 (Docking Alignment & Grasp Quality)**: The user noted: *"Near-obstacle fruits are unreached due to single-angle docking, and grasp stability lacks verification."* The agent executed two targeted lookups:
  1) *Multi-Angle Docking & IK Synchronization*: Querying `lookup_entity("CAP_mobile_manipulation_ik")`, the agent implemented circumferential standoff sampling ($d = 0.45$ m) and aligned base-yaw with arm IK $[\pi, 0, \text{yaw}_{\text{base}}]$.
  2) *Ferrari-Canny Force-Closure Assessment*: Querying `lookup_entity("CAP_force_closure_quality")`, the agent embedded the analytical closed-form metric ($\epsilon = r \cdot \mu \cdot \text{antipodality} = 0.0420 > 0$), verifying contact equilibrium before releasing stem constraints.

Final Execution: Driven by 3 iterative turns, the system converged to **100% task success (4/4 fruits detected, reached, grasped, and deposited)**, 7 active CCD push-backs, 0 collision clipping, and full HUD telemetry.

D. Case Study 2: Unitree G1 Humanoid in Orchard Autonomous Harvesting

To evaluate high-degree-of-freedom embodied systems on rough terrain, we examine an autonomous bipedal harvesting controller for the **Unitree G1 humanoid robot** (23-DOF) operating on a 15° sloped orchard within the MuJoCo physics engine.

Multi-Turn Iterative Trajectory: Starting from *"Develop a MuJoCo controller for the Unitree G1 to walk over rough sloped terrain and harvest delicate fruit"*:

- **Turn 1 (Initial Locomotion Failure)**: Baseline flat-ground gait collapsed immediately on the 15° slope. Prompted by the user's report (*"Humanoid slips and tips over on uneven slopes"*), the agent queried ENTROPY BOX for sloped rough-terrain locomotion:
  1) *Perception & GPU Elevation Mapping*: Integrated `AST_elevation_mapping_cupy` to downsample LiDAR point clouds (0.02 m) and estimate local terrain normal vectors.
  2) *Nonlinear MPC Dynamic Balancing*: Bound `AST_unitree_sdk2` to enforce centroidal dynamics and Zero Moment Point (ZMP) stability over slope friction $\mu = 0.6$.

TABLE X
PLAN SYNTHESIS AND CONTEXT GROUNDING BENCHMARK: 24 TASKS × 5 CONDITIONS (120 PREDICTION RECORDS). LOWER IS BETTER FOR UNSUPPORTED CLAIMS AND CONSTRAINT VIOLATIONS; HIGHER IS BETTER OTHERWISE. BOOTSTRAP INTERVALS USE 2,000 RESAMPLES.

Condition	Unsupported claims ↓	Hard-constraint violation ↓	Constraint coverage ↑	Graph validity ↑	Capability F1 ↑
LLM direct	100.0%	100.0%	0.0%	83.3%	.025
Lexical RAG	100.0%	100.0%	0.0%	75.0%	.020
Hybrid RAG	100.0%	100.0%	0.0%	79.2%	.008
ENTROPY BOX context only	100.0%	83.3%	16.7%	75.0%	.008
ENTROPY BOX Full	**5.2%**	**66.7%**	**35.4%**	62.5%	.008
Full − Hybrid RAG	-94.8 [$-97.4, -92.1$] pp	-33.3 [$-54.2, -16.7$] pp	$+35.4$ [$+16.7, +54.2$] pp	-16.7 [$-37.5, +4.2$] pp	.000 [.000, .000]

- **Turn 2 (Grasping Verification & Gap Reporting)**: The user requested non-destructive peach picking. The agent queried Pinocchio analytical IK (`AST_pinocchio`) for reachability and applied slip-limited force control ($F_{\text{normal}} \leq 4.5\,\text{N}$). When asked for closed-loop tactile fruit softness estimation, ENTROPY BOX explicitly emitted a formal capability gap ($\mathcal{C}_{\text{gap}}$) and recommended impedance current feedback rather than hallucinating an ungrounded sensor driver.

Final Execution: The multi-turn compiled controller achieved 100% non-destructive harvest success across 10 trials in MuJoCo with zero joint singularities.

VIII. DISCUSSION, ENGINEERING LESSONS, AND THREATS TO VALIDITY

A. Core Lessons for Agent-Native Systems

1) **Hand Off Artifacts, Not Summaries**: Passing conversational summaries between agents causes severe context decay. Durable, typed JSON/Markdown artifacts with programmatic validators are essential.
2) **Deterministic Gates are Non-Negotiable**: LLMs cannot reliably self-assess graph consistency or identity uniqueness. Deterministic code (DFS cycle checks, file locks, transactional rewrites) must govern state transitions.
3) **Intent Routing Beats Monolithic Pipelines**: Complex rerankers degrade entity retrieval. Dynamic intent routing is crucial to select optimal retrieval strategies.

B. Limitations and Threats to Validity

- **Corpus Provenance Completeness**: In audited chain documents, 81.99% of references resolve to physical artifacts with 2,388 explicit `[unverified]` markers. Unresolved references remain structural debt.
- **Entity Benchmark Scope**: The 84-query benchmark evaluates entity ranking; comprehensive graph-generation benchmarks with full human double-blind adjudication remain ongoing work.

IX. CONCLUSION AND OPEN-SOURCE RELEASE

ENTROPY BOX demonstrates that the fragmented universe of robotics engineering knowledge can be compiled into a persistent, typed capability substrate via an agent-native compiler bounded by deterministic gates. Spanning 2,511 topics, 37,673 capabilities, 11,397 assets, and 54,602 dependency edges, ENTROPY BOX eliminates over 21,316 redundant derivations. By exposing 6-dimensional Semantic Views and standardized FastMCP tools to downstream coding agents, ENTROPY BOX provides machine-checkable architectural foresight and grounded verification for modern robotics engineering.

Public Platform and API: https://xiangshang.ngrok.app/

Open Source Repository: https://anonymous.4open.science/r/entropy-box/

AI USE DISCLOSURE

OpenAI ChatGPT/Codex was used to assist manuscript restructuring, language editing, consistency checking, and LaTeX preparation. The authors verified all source code, data, claims, citations, and the final manuscript.

REFERENCES

[1] M. Waibel et al., *RoboEarth*, IEEE Robotics & Automation Magazine, vol. 18, no. 2, pp. 69–82, 2011.
[2] M. Tenorth and M. Beetz, *KnowRob: A Knowledge Processing Infrastructure for Cognition-Enabled Robots*, Int. J. Robot. Res., vol. 32, no. 5, pp. 566–590, 2013.
[3] M. Beetz, L. Mösenlechner, and M. Tenorth, *CRAM—A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments*, in Proc. IEEE/RSJ IROS, 2010.
[4] M. Beetz et al., *openEASE—A Knowledge Processing Service for Robots and Robotics/AI Researchers*, in Proc. IEEE ICRA, 2015.
[5] P. Lewis et al., *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*, in Advances in Neural Information Processing Systems, 2020.
[6] Y. Gao et al., *Retrieval-Augmented Generation for Large Language Models: A Survey*, arXiv:2312.10997, 2023.
[7] D. Edge et al., *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*, arXiv:2404.16130, 2024.
[8] M. Ahn et al., *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*, in Proc. Conf. Robot Learning, 2022.
[9] J. Liang et al., *Code as Policies: Language Model Programs for Embodied Control*, in Proc. IEEE ICRA, 2023.
[10] S. Yao et al., *ReAct: Synergizing Reasoning and Acting in Language Models*, in Proc. ICLR, 2023.
[11] V. Karpukhin et al., *Dense Passage Retrieval for Open-Domain Question Answering*, in Proc. EMNLP, 2020.
[12] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, *Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods*, in Proc. ACM SIGIR, 2009.
[13] R. Nogueira and K. Cho, *Passage Re-Ranking with BERT*, arXiv:1901.04085, 2019.
[14] I. Temiraliev, D. Yang, and Y. Zhang, *Retrieval-Augmented Robots via Retrieve–Reason–Act*, arXiv:2603.02688, 2026.
[15] Z. Zhou, S. Chen, O. Salunkhe, E. T. Bekar, J. Stahre, and A. Skoogh, *Retrieval-Grounded Robot Program Generation and Simulation-Based Correction via Model Context Protocol*, arXiv:2608.21417, 2026.
[16] S. Pan et al., *LLM-Empowered Knowledge Graph Construction: A Survey*, arXiv:2510.20345, 2025.
[17] B. Jin et al., *Graphs Meet AI Agents: Taxonomy, Progress, and Future Opportunities*, arXiv:2506.18019, 2025.