

From Sensor Fusion to Autonomous Systems

Agentic AI for RAG Systems

Pi Wei Chen

Talk Roadmap

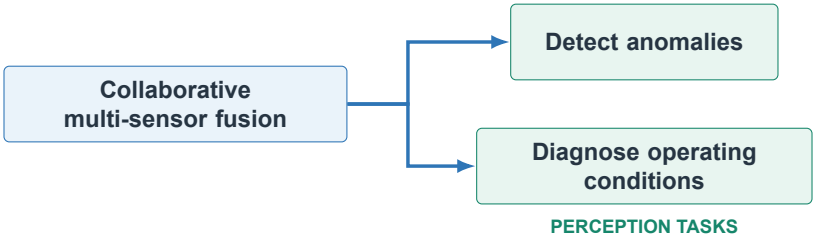
1. **Introduction: What does the DC5 project aim to build?**
2. **From the Project to My Current Research: Why does autonomy lead to Agentic RAG?**
3. **Problem Formulation: What retrieval-policy problem are we solving?**
4. **Solution Intuition: How can experience improve policy without weight updates?**
5. **Prototype Framework: How does the proposed system operate?**

Project objective $\rightarrow$ research transition $\rightarrow$ problem $\rightarrow$ intuition $\rightarrow$ prototype

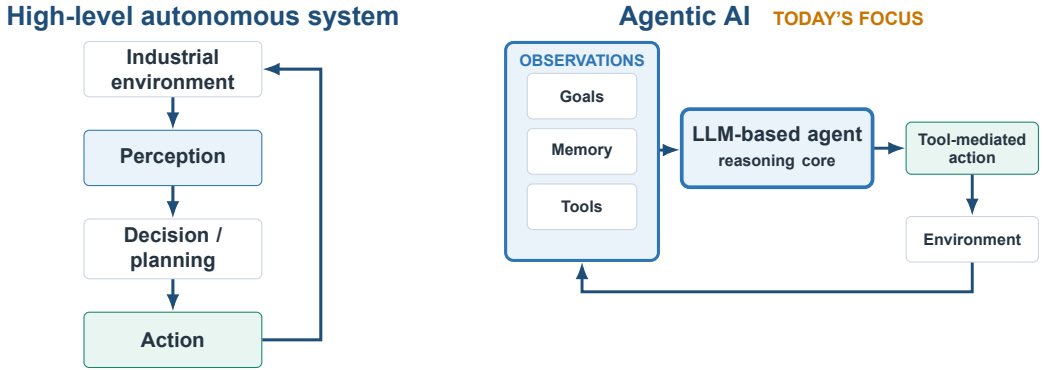
Project objective:

Enhancing Perception Accuracy through Collaborative Sensor Fusion Framework in Autonomous Systems

- **Project objective:** develop an autonomous system enabled by collaborative sensor fusion.
- **Research path 1 — perception:** develop sensor-fusion algorithms for prediction tasks, including anomaly detection and condition-based maintenance.
- **Research path 2 — autonomy:** enable the system to access prediction results and take the corresponding action.



Agentic AI provides the observe–reason–act architecture needed for industrial autonomy



Perception ↔ Observation **Decision** ↔ LLM reasoning **Action** ↔ Tool use

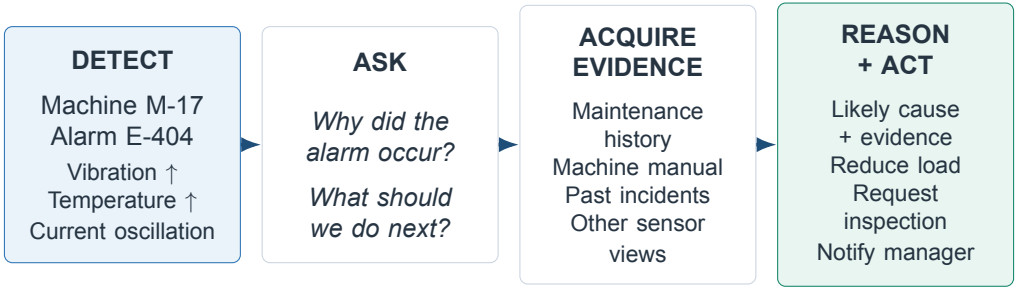
Foundation: fused sensor evidence → autonomous behavior. **Today's focus: Agentic AI.**

Talk Roadmap

1. Introduction: What does the DC5 project aim to build?
2. From the Project to My Current Research: Why does autonomy lead to Agentic RAG?
3. Problem Formulation: What retrieval-policy problem are we solving?
4. Solution Intuition: How can experience improve policy without weight updates?
5. Prototype Framework: How does the proposed system operate?

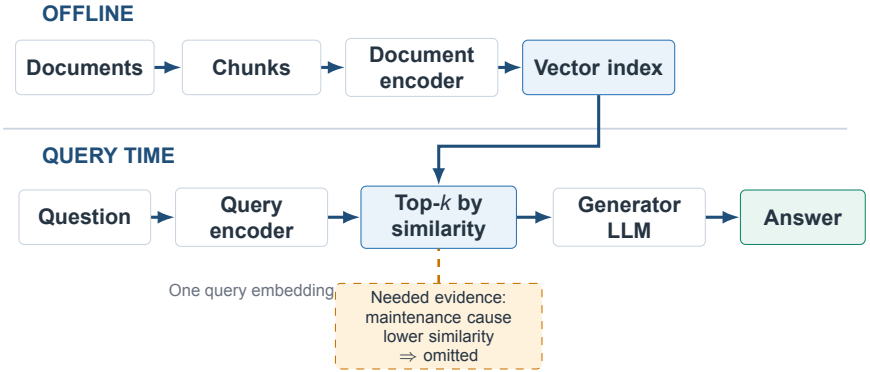
Project objective $\rightarrow$ research transition $\rightarrow$ problem $\rightarrow$ intuition $\rightarrow$ prototype

An autonomous industrial system must turn an anomaly into a grounded and safe response



Detection alone is insufficient: the system must acquire the right evidence before choosing an action.

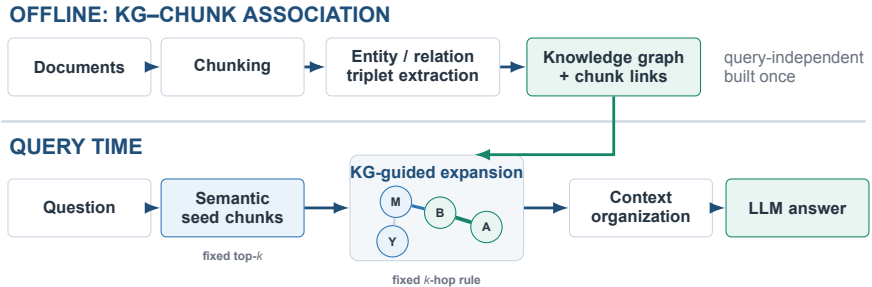
Vanilla RAG grounds the LLM, but retrieves by relevance rather than evidence need



The retriever ranks relevance, but does not reason about what evidence is still missing.

Adapted from Lewis et al. (2020), Fig. 1, arXiv:2005.11401.

KG^{2}RAG connects dispersed evidence, but its traversal policy is predefined

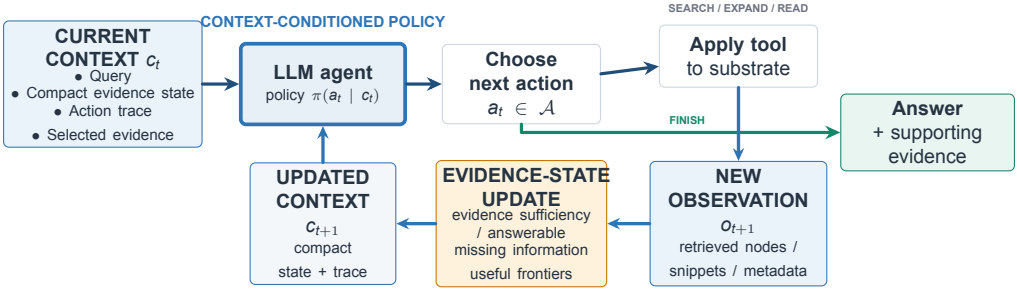


Offline graph structure improves connectivity; query-time traversal is still predefined.

Adapted from Zhu et al. (2025), KG^{2}RAG Fig. 2, arXiv:2502.06864.

Agentic RAG uses retrieval as an active perception process

- Retrieval is the **perception capability** that enables the agent to access external evidence.
- At every step, the policy uses the **current context**—not only the latest observation—to decide what to do next.



Raw observation → evidence-state update → updated context; FINISH → answer with supporting evidence.

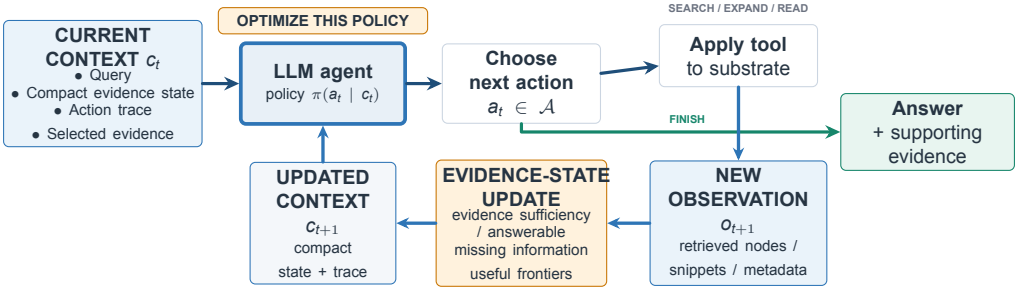
Talk Roadmap

1. Introduction: What does the DC5 project aim to build?
2. From the Project to My Current Research: Why does autonomy lead to Agentic RAG?
3. **Problem Formulation: What retrieval-policy problem are we solving?**
4. Solution Intuition: How can experience improve policy without weight updates?
5. Prototype Framework: How does the proposed system operate?

Project objective $\rightarrow$ research transition $\rightarrow$ problem $\rightarrow$ intuition $\rightarrow$ prototype

We optimize the retrieval policy for accuracy and efficiency

- Optimize the LLM retrieval policy π so it maps each **current context** to the best corresponding action.



$$a_t \sim \pi(a_t \mid c_t), \qquad a_t \in \mathcal{A}$$

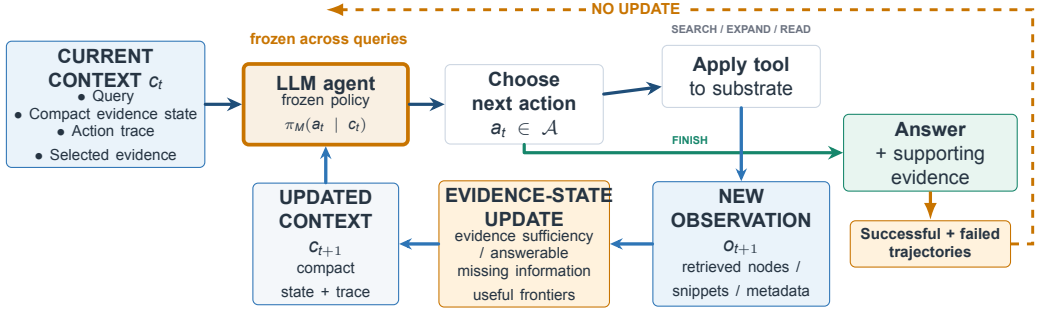
c_t: current context π: LLM retrieval policy

$\mathcal{A}$: action space a_t: selected action

Accuracy ↑ sufficient evidence + correct answer

Efficiency ↑ fewer calls, tokens, and latency

Training-free Agentic RAG cannot optimize its policy from experience

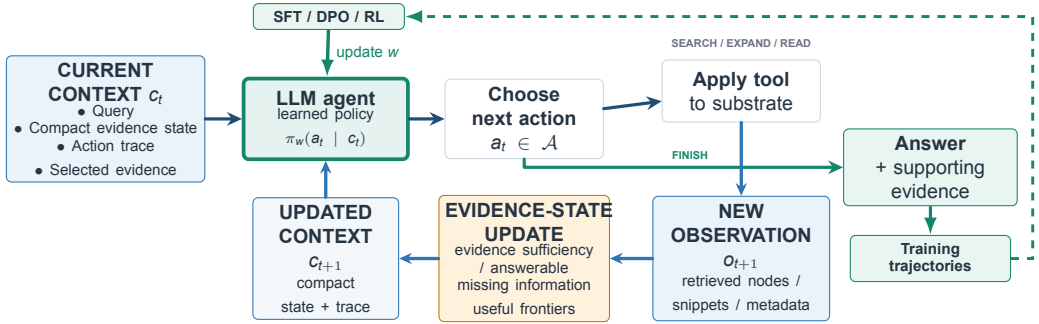


Adapts within a query, but successful and failed trajectories do not improve the policy used on later queries.

The action policy remains bounded by the frozen LLM and handwritten retrieval instructions.

Representative training-free interfaces: A-RAG and ARK.

Weight-tuned methods store the retrieval policy in model weights



High training cost	Closed-model barrier	Retune after model/tool changes

Experience is learned, but stored in model-specific weights.

Representative methods: EVO-RAG (Ji et al., 2025) and MARAG-R1 (Luo et al., 2025).

RESEARCH QUESTION

Can retrieval policy improve from experience without modifying LLM weights?

Talk Roadmap

1. Introduction: What does the DC5 project aim to build?
2. From the Project to My Current Research: Why does autonomy lead to Agentic RAG?
3. Problem Formulation: What retrieval-policy problem are we solving?
4. **Solution Intuition: How can experience improve policy without weight updates?**
5. Prototype Framework: How does the proposed system operate?

Project objective $\rightarrow$ research transition $\rightarrow$ problem $\rightarrow$ intuition $\rightarrow$ prototype

Agent skills encode reusable behavior in free-form representations

- **An agent skill is a reusable decision rule that guides an agent on what to do next.**
- A skill can be expressed as natural-language guidance, a conditional rule, or an executable workflow.

One skill, multiple levels of formality

Agent Skill

Natural language

"If lexical overlap is low, use Dense Search."

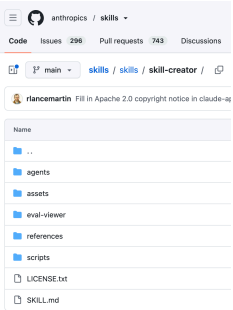
Decision rule

If overlap $< \tau \rightarrow$ Dense Search

Executable workflow

Dense $\rightarrow$ Expand $\rightarrow$ Read $\rightarrow$ Answer

In practice: a skill package

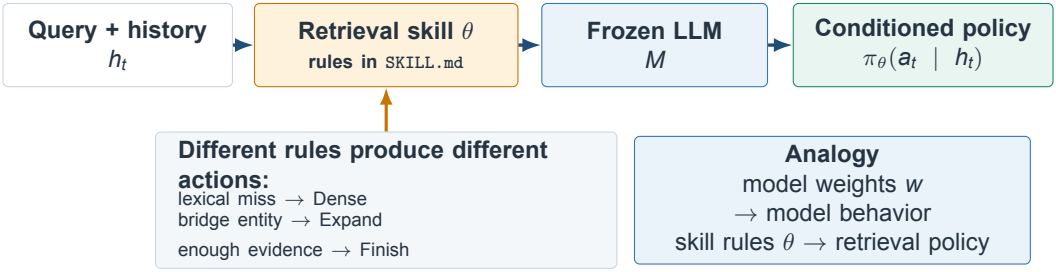


`SKILL.md` + optional agents, assets, references, and scripts

Screenshot: Anthropic skills repository

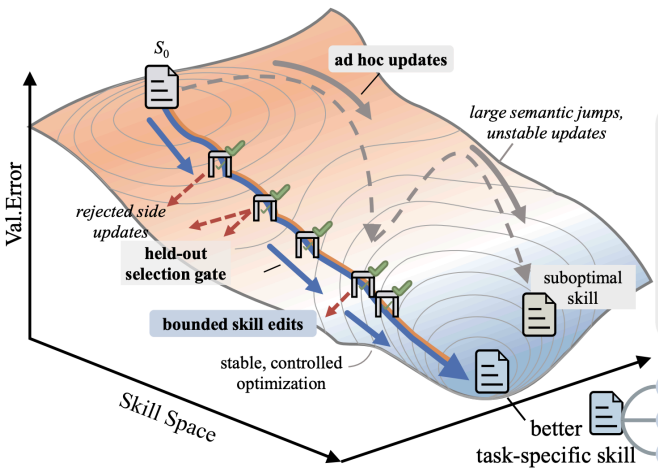
A skill can be treated as an external policy parameter

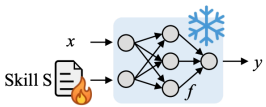
- A skill can be viewed as an external parameter θ that conditions a frozen LLM.
- Changing rules in `SKILL.md` changes the retrieval policy, just as changing model parameters changes a forward pass.



Editing skill θ changes the policy without modifying the underlying LLM weights.

Updating skills like deep-learning parameters





Text-space optimization analogy

parameter	→	skill document
gradient direction	→	trajectory-derived edit direction
learning rate	→	edit budget
validation check	→	held-out selection gate
stable training setting	→	batch / minibatch/ schedule / gate

- better task-specific ability
- cross-model /harness generalization
- low-cost transfer and iteration

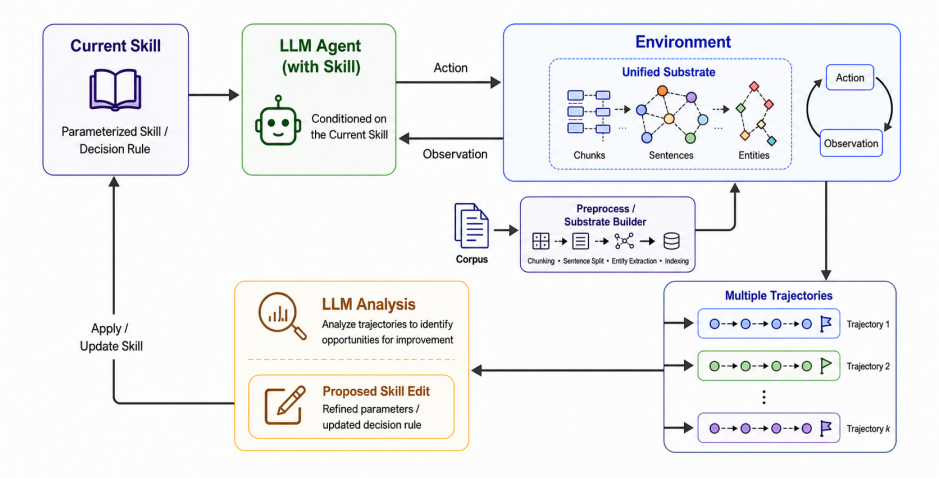
SkillOpt text-space optimization analogy, adapted from Yang et al. (2026), arXiv:2605.23904. Figure supplied by the presenter.

Talk Roadmap

1. Introduction: What does the DC5 project aim to build?
2. From the Project to My Current Research: Why does autonomy lead to Agentic RAG?
3. Problem Formulation: What retrieval-policy problem are we solving?
4. Solution Intuition: How can experience improve policy without weight updates?
5. **Prototype Framework: How does the proposed system operate?**

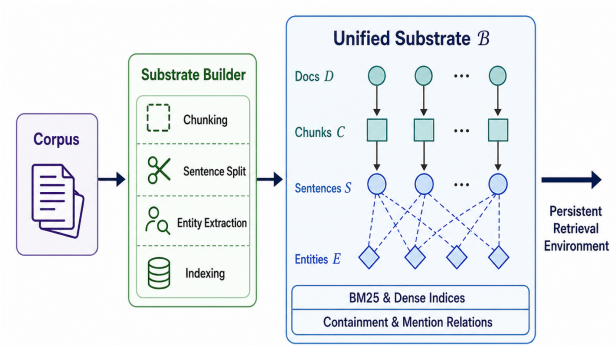
Project objective $\rightarrow$ research transition $\rightarrow$ problem $\rightarrow$ intuition $\rightarrow$ prototype

Proposed method: skill optimization for Agentic Retrieval



The target LLM and retrieval substrate remain fixed; experience updates only the retrieval skill.

Documents become a multi-granular retrieval substrate

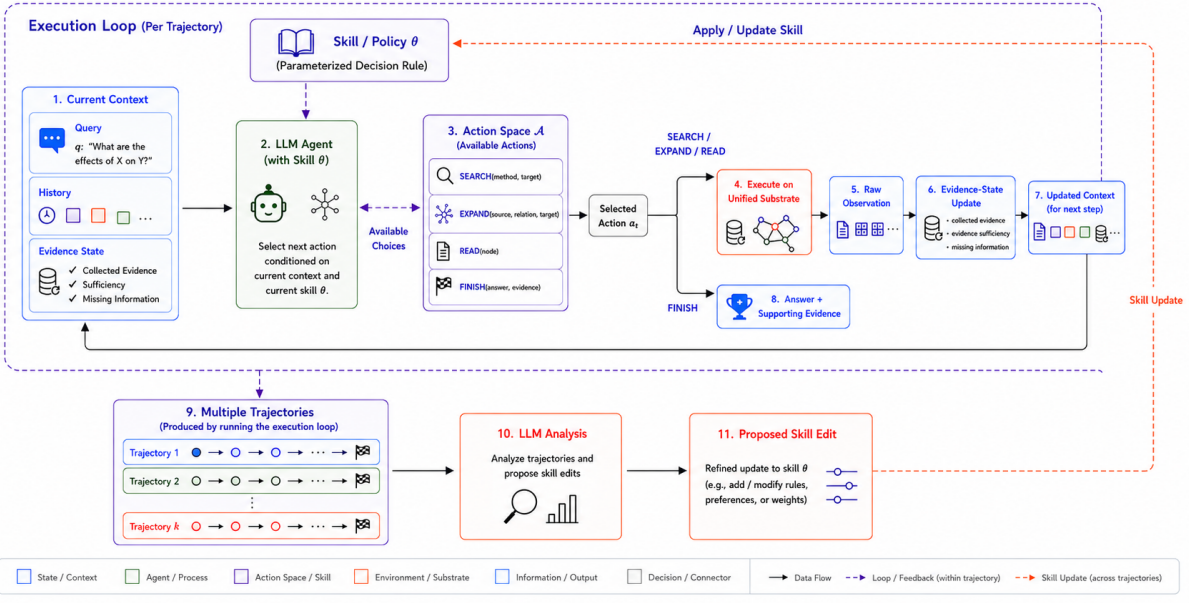


Why a unified substrate?

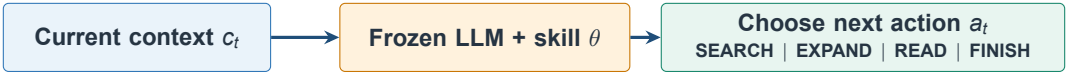
- **Multi-granularity traversal:** documents, chunks, sentences, and entities let the agent move from high-level context to fine-grained evidence.
- **Graph-like multi-hop navigation:** containment and mention relations let the agent hop across chunks, sentences, and entities for multi-hop reasoning.

Graph-like navigation without constructing a full knowledge graph.

Overview of Skill Optimization for Agentic Retrieval



At query time, the skill selects among four substrate-aware actions



Action	Call	Role at query time
SEARCH	`SEARCH(q, method, target, top_k)` method: BM25 | Dense target: Entity | Sentence | Chunk	Locate candidate evidence from a new query or retrieval intent.
EXPAND	`EXPAND(source_ids, relation, target, top_k)` relation: MENTIONS | MENTIONED_IN CONTAINS | CONTAINED_BY	Traverse typed substrate edges to obtain multi-hop evidence.
READ	`READ(node_ids)`	Load the full sentence or chunk after localization.
FINISH	`FINISH(answer, evidence_ids)`	Stop only when evidence is sufficient; return answer and provenance.

SEARCH / EXPAND / READ $\rightarrow$ new observation and evidence-state update; FINISH $\rightarrow$ final answer with supporting evidence.



Silesian University of Technology

Q&A

Can retrieval experience improve policy without modifying LLM weights?

Pi Wei Chen