A MULTI-AGENT SYSTEM TO ORCHESTRATE INTERACTIONS WITH DIGITAL TWINS OF EARTH

M. Tsokanaridou[1], J. Hackstein[2], G. Hoxha[2], S.-A. Kefalidis[1], K. Plas[1], B. Demir[2], M. Koubarakis[1], M. Corsi[3], C. Leoni[3], G. Pasquali[3], C. Pratola[3], S. Tilia[3] and N. Longépé[4]

[1]Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece
[2]BIFOLD and Technische Universität Berlin, Germany
[3]e-GEOS S.p.A., Italy
[4]Φ-lab, ESA ESRIN, Frascati, Italy

ABSTRACT

We present a new-generation, AI-agent-powered digital assistant featuring four specialized engines for satellite imagery: search by image, search by caption, visual question answering, and knowledge graph question answering. At the core of the system is a Task Interpreter, designed as a multi-agent system, which coordinates these engines to address complex user requests for Earth observation data. The Task Interpreter comprises four agents: an Engine Routing Agent that selects the appropriate engine or rejects unmanageable requests; a Conversational Agent that handles general or out-of-scope queries; an Argument Extraction Agent that identifies image type parameters for retrieval tasks; and a Tool Feasibility Agent that assesses the applicability of tools for domain-specific queries. This multi-agent system enables seamless interaction with Digital Twins of Earth, with an emphasis on modularity and extensibility to adapt to the rapid evolution of remote sensing technologies.

***Index Terms*—** Multi-agent systems, digital assistant, digital twins, search by image, search by caption, visual question answering, knowledge graph question answering

1. INTRODUCTION

In Artificial Intelligence (AI), an *agent* is an autonomous entity capable of perceiving its environment, making decisions, and acting upon it to achieve specific goals. Multi-agent systems (MAS) is a subarea of AI studying societies of agents in cooperative or competitive settings and has a long tradition of outstanding research results. With the recent revolution of large language models (LLMs) and foundation models (FMs), the area of MAS is receiving again a lot of attention with the proposal of LLM-powered agent frameworks such as AutoGen [14], LangChain and CrewAI.

As part of these recent developments, we have seen the proposal of agent and multi-agent system architectures powered by LLMs in the Remote Sensing (RS) area [4, 9, 10, 11, 15]. *Remote Sensing ChatGPT* [4] introduces a system where ChatGPT interprets user requests and sequentially invokes specialized RS models for tasks such as object detection and land use classification. *RescueADI* [11] focuses on disaster interpretation, employing a LLM-driven agent to dynamically plan and execute multiple specialized tasks like damage assessment and rescue pathfinding. *RS-Agent* [15] extends this paradigm by integrating high-performance tools and a retrieval-augmented knowledge base to support professional geospatial analysis. *GlobeFlowGPT* [9] applies a multimodal LLM orchestrator to facilitate complex geospatial workflows, including flood forecasting and vegetation monitoring, with containerized tool integration. Similarly, *GeoLLM-Squad* [10] adopts a MAS, using an orchestrator to coordinate specialized agents for a broad range of remote sensing tasks, such as urban monitoring, climate analysis, forestry protection, and agricultural studies. Like our approach, it emphasizes modularity, extensibility, and the separation of orchestration from task-solving components.

Parallel to these developments, the emergence of *Digital Twins of Earth* (DTEs) —high-fidelity, dynamic digital representations of the Earth's systems—has created new demands for intelligent, continuous interaction with massive Earth observation (EO) datasets. DTEs require the ability to access, interpret, and integrate diverse data streams in a flexible, scalable, and context-aware manner. MAS are particularly well suited to meet these needs, enabling specialized tools to work together dynamically to support the complex data requirements of DTEs.

However, despite recent advances, there is currently *no* EO data provider that offers a digital assistant capable of guiding users in finding the EO data they seek. This is a critical functionality gap, especially as the volume of EO data made available through initiatives like *Copernicus* and *Landsat* continue to expand. Without intelligent assistance, this wealth of data remains difficult to access for both expert and non-expert users, such as journalists searching for timely EO

This work was supported by ESA project DA4DTE.

imagery of environmental disasters or policymakers monitoring climate events.

To address this challenge, we introduce the *Digital Assistant for Digital Twins of Earth (DA4DTE)*, an AI-powered multi-agent digital assistant designed to facilitate seamless interaction with EO datasets. In DA4DTE, a *Task Interpreter* operates as a multi-agent system comprising specialized agents that collaboratively interpret user requests and orchestrate the activation of appropriate search engines or tools. We distinguish between the specialised *engines* serving EO tasks, the multi-agent *Task Interpreter* with its agents—autonomous functional components responsible for specific subtasks—and the *assistant*, the overall user-facing system deployed to fulfill complex information retrieval workflows.

2. MULTI-AGENT SYSTEM FOR ORCHESTRATION

DA4DTE enables a user to pose multi-modal requests, that — in addition to text— can include RS images, either uploaded or selected on the User Interface map. The assistant's toolset allows for a variety of requests including geospatial or visual queries, requests for images by describing their visual context or metadata, image search requests, and queries for explanation on image similarity results. Between the user and the DA4DTE engines lies the *Task Interpreter*: a MAS responsible for engine orchestration and the mediation between the user and individual engines. The architecture is illustrated in Figure 1, which highlights the collaborative roles of each agent module and their interactions with the user interface and underlying engine components.

To ensure future extensibility, we categorize orchestration responsibilities into two types: **core** and **assistant** tasks. *Core tasks* are permanent and fundamental to any version of the assistant, regardless of the tools or data sources integrated. In contrast, *assistant tasks* are tailored to the current implementation state and may evolve as functionalities and resources expand. Each task is assigned to a dedicated agent, forming a MAS, implemented using the AutoGen [14] framework and currently comprising the following four agents.

The first agent is the **Engine Routing Agent** (Core). This agent is a zero-shot prompted LLM that selects the most appropriate engine to activate based on the user request. It also has the capability to reject requests that fall outside the scope of all available engines.

The second agent is the **Conversational Agent** (Core). This is a fallback conversational agent designed to handle general, ambiguous, or out-of-domain queries. Although it is a capable LLM, it is specifically prompted not to respond to irrelevant requests, ensuring that the assistant remains task-focused.

The third agent is the **Argument Extraction Agent** (Assistant). This is an agent dedicated to extracting key parameters required by specific tools. In the current implementation, it identifies the requested image type (e.g., Sentinel-1 or Sentinel-2) when the *Search-by-Image* engine is activated.

Finally, the fourth agent is the **Tool Feasibility Agent** (Assistant). This is a utility agent responsible for validating whether a requested operation is feasible under current system capabilities. For example, the *Search-by-Text* engine presently supports only vessel-related queries. If a user request falls outside this domain, the agent triggers a relevant explanatory message to the user.

3. ENGINES AND THEIR FUNCTIONALITIES

DA4DTE integrates four specialized engines, tailored to specific Question Answering (QA) or retrieval tasks.

The first engine is the **Knowledge Graph QA Engine** TerraQ [8]. TerraQ[1] is a QA system that is designed to process natural language requests that include spatiotemporal or metadata related criteria and satisfy the request by retrieving data from a Knowledge Graph (KG). User requests can include references to image metadata (e.g., snow percentage in an image), geoentities (e.g., the country France), administrative divisions (e.g., municipalities, regions), as well as spatiotemporal constraints.

For example, users can make requests like "Give me a hundred images of rivers near ports in France, with less than 20% snow coverage and more than 10% cloud coverage, taken in 2021". The engine then takes this request as input, translates it into a semantically equivalent SPARQL query. To do so, it employs a pipeline of components for natural language understanding and KG grounding. First, relevant entities and classes are extracted from the KG [13]. Then, relations between the retrieved entities and classes are identified, including spatial and temporal relations. At this stage, the core of the query is complete, and the expected return values are identified by a finetuned Llama 2 model. The query generator then produces the complete, executable SPARQL query. This query is subsequently enhanced by a finetuned on SPARQL Mistral-7b-v2 model, and rewritten by GoST[2] to optimize execution efficiency. GoST rewrites queries to replace GeoSPARQL functions with equivalent materialized topological predicates. In the end, the query is executed over a GraphDB endpoint, and the QA process is complete.

The second egine is the **Search-by-Image Engine**. This engine takes a query image and computes the similarity function between the query image and all archive images to find the most similar images to the query in a scalable way. This is achieved based on two main steps: i) the image description step, which characterizes the spatial and spectral information content of RS images; and ii) the image retrieval step, which evaluates the similarity among the considered hash codes and then retrieves images similar to a query image in the order of similarity. Our Search-by-Image Engine is defined based

[1] https://terraq.di.uoa.gr/

[2] https://github.com/AI-team-UoA/GoST

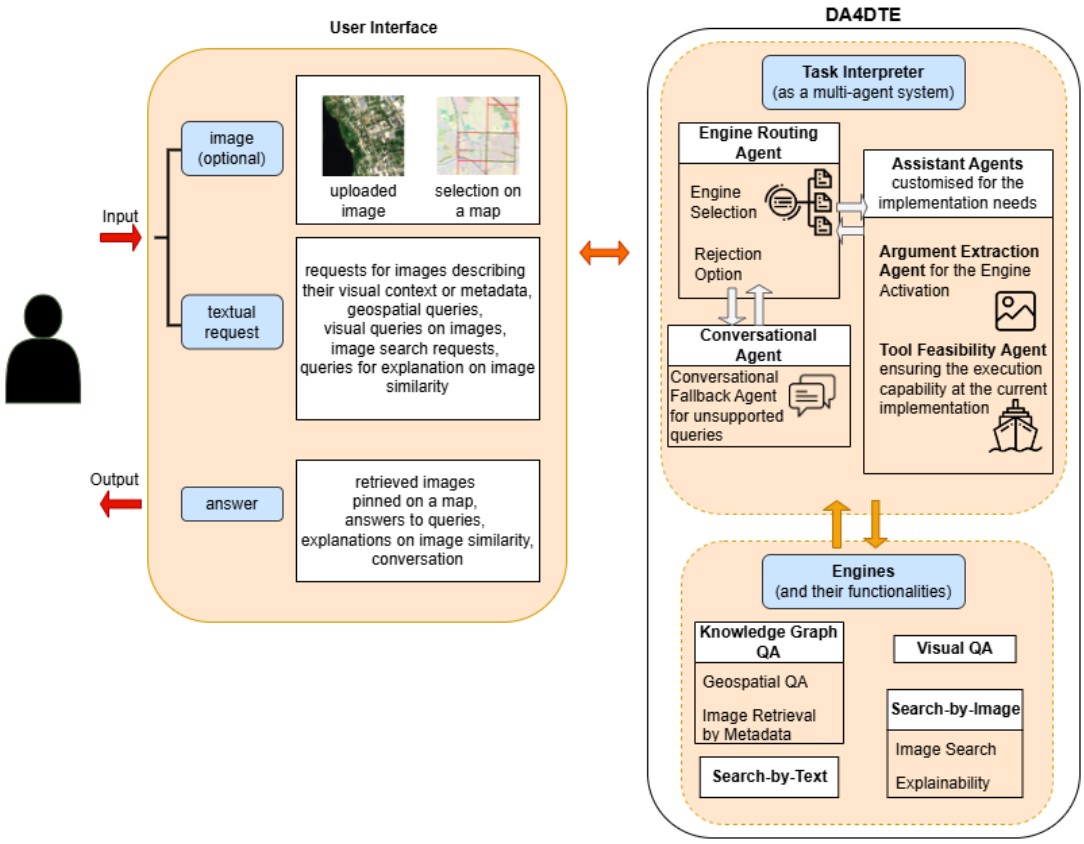

Fig. 1. High-level architecture of the digital assistant (DA4DTE), showing the user interface, multi-agent Task Interpreter, and the specialized engines (figure inspired by Figure 1 of [15]).

on two self-supervised methods: 1) deep unsupervised cross-modal contrastive hashing (DUCH) [12]; and 2) cross-modal masked autoencoder (CM-MAE) [6]. For both methods, the image description step is composed of two modules: 1) a feature extraction module, which learns deep feature representations of RS images by exploiting visual transformers (ViT); and 2) a deep hashing module, which learns to map image representations into hash codes. The first module of the DUCH method is based on contrastive self-supervised image representation learning, while that of the CM-MAE method is based on unsupervised masked image modelling. The second module of each method employs a hashing subnetwork with binarization loss functions. Our engine has both the single-modal (also known as uni-modal) and cross-modal content-based image retrieval capability due to the consideration of the modality-specific encoders.

A key feature of the search-by-image engine is the integration of the *Explainability tools* to understand and explain the decision of the engine in retrieving a particular image given a query image. To this end, we incorporate two explainability tools: Layer-wise Relevance Propagation (LRP) [1] and BiLRP [3]. The LRP highlights areas in the input image supporting a specific class decision by generating heatmaps. Since CM-MAE is self-supervised and lacks class predictions, we train an auxiliary classification head to estimate class probabilities for each image pair. These predictions enable the generation and interpolation of class-specific LRP heatmaps, which emphasize semantically similar regions across image pairs. BiLRP, while more computationally intensive, identifies in the image pairs shared regions without needing a classification head.

The third engine is the **Search-by-Text Engine**. This engine takes a text sentence as a query and efficiently retrieves the most similar images to the query text, achieving scalable cross-modal text-image retrieval. The Search-by-Text Engine is developed by adapting the above-mentioned self-supervised DUCH [12] to be operational on text based queries. To this end, the feature extraction module is adapted to extract feature representations of image-text pairs by exploiting bidirectional transformers (e.g., BERT [2]) as text-specific encoders together with ResNet-152 [7] as image-specific encoders. The second module of each method is adapted to learn cross-modal binary hash codes for image and text modalities by simultaneously preserving semantic discrimination and modality-invariance in an end-to-end manner.

To evaluate DUCH, we constructed a vessel captioning

dataset, consisting of vessel text-image pairs generated via a template-based image captioning approach. This approach consists of creating predefined sentence templates with empty slots. The slots are then filled using semantic cues from vessel bounding boxes (e.g., count, size) and contextual data from OpenStreetMap, particularly coastline proximity (i.e., vessel locations relative to harbors or coastlines). Vessel sizes, derived from bounding box dimensions, were categorized into five classes (very small to very big) and mapped to two vessel types: boats (very small to medium) and ships (big and very big), reflecting typical usage and navigational context.

Finally, the fourth engine is the **Visual QA Engine**. This engine enables users to ask questions about the content of RS images in a free-form manner, extracting valuable information. It employs the LiT-4-RSVQA [5] model, which has been trained and evaluated on RSVQAxBEN[3]. The LiT-4-RSVQA architecture focuses on achieving state-of-the-art performance, while also providing rapid response times. To do so, it employs the following modules: i) a lightweight text encoder module; ii) a lightweight image encoder module; iii) a fusion module; and iv) a classification module. A RS image I and a question Q about this image are considered as input. The encoder modules produce vector representations which are subsequently passed to the fusion module. The feature fusion module consists of two linear projections and a modality combination. The projections map the two modalities with dimensions dt and dv into a common dimension df, where dt and dv denote the dimensions of the flattened output of the text and image encoder modules, respectively. The value of dv differs depending on the used lightweight transformer. The projected features are then elementwise multiplied. The classification module is defined as an MLP projection head.

4. DA4DTE IN ACTION

We now consider a use case scenario for the digital assistant. The assistant welcomes the user and asks them to pose a request. The user asks for a Sentinel-1 image from France during 2020, with snow coverage of more than 50%. Then, the Engine Routing Agent of the Task Interpreter decides that this is a request that should be fulfilled by the Knowledge Graph QA Engine which returns the appropriate image. The interaction goes on with the user asking for a similar Sentinel-2 image and then the Search-by-Image Engine is selected by the Engine Routing Agent. The term "Sentinel-2" is extracted by the Argument Extraction Agent as the modality argument, so the engine is activated and returns the appropriate image. Having selected that Sentinel-2 image, the user asks whether it presents a rural area and the answer by the Visual QA Engine is presented. Finally, the user closes the interaction with the assistant and the Engine Routing Agent of the Task Interpreter calls the Conversational Agent to answer appropriately.

[3] https://zenodo.org/records/5084904

5. FUTURE WORK

We plan to explore several research directions to further improve the capabilities of the system. First of all, we aim to implement an alternative Engine Routing Agent using the Function Calling paradigm in LLMs, to improve control over engine invocation compared to the current zero-shot prompting setup. We also plan to extend the assistant's capabilities to multi-step requests where multiple engines can be activated in a sequence. As the complexity of the system increases, we intend to integrate a Manager Agent to oversee and coordinate the behavior of all other agents within the Task Interpreter.

REFERENCES

[1] S. Bach et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7), 2015.

[2] J. Devlin et al. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[3] O. Eberle et al. Building and interpreting deep similarity models. *PAMI*, 44(3), 2020.

[4] H. Guo et al. Remote Sensing ChatGPT: Solving remote sensing tasks with ChatGPT and visual models. In *IGARSS*, 2024.

[5] L. Hackel et al. LiT-4-RSVQA: Lightweight transformer-based visual question answering in remote sensing. In *IGARSS*, 2023.

[6] J. Hackstein et al. Exploring masked autoencoders for sensor-agnostic image retrieval in remote sensing. *TGRS*, 63, 2025.

[7] K. He et al. Deep residual learning for image recognition. In *CVPR*, 2016.

[8] S. Kefalidis et al. TerraQ: Spatiotemporal question-answering on satellite image archives. In *IGARSS*, 2025.

[9] D. Kononykhin et al. From data to decisions: Streamlining geospatial operations with multimodal GlobeFlowGPT. In *ACM SIGSPATIAL*, 2024.

[10] C. Lee et al. Multi-agent geospatial copilots for remote sensing workflows. https://arxiv.org/abs/2501.16254.

[11] Z. Liu et al. RescueADI: Adaptive disaster interpretation in remote sensing images with autonomous agents. *TGRS*, 2025.

[12] G. Mikriukov et al. Unsupervised contrastive hashing for cross-modal retrieval in remote sensing. In *ICASSP*, 2022.

[13] F. Piccinno and P. Ferragina. From TagME to WAT: a new entity annotator. In *ACM ERD*, 2014.

[14] Q. Wu et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework. https://arxiv.org/abs/2308.08155.

[15] W. Xu et al. RS-Agent: Automating remote sensing tasks through intelligent agents. https://arxiv.org/abs/2406.07089.