AUTOMATING EARTH OBSERVATION ANALYTICS PIPELINES WITH AGENT RAVEN

Gereon Dusella[*], *Haralampos Gavriilidis*[*], *Binger Chen*[*],
Begüm Demir[*], *Volker Markl*[*,‡], *Eleni Tzirita Zacharatou*[§]

[*]BIFOLD & Technische Universität Berlin, [‡]DFKI, [§]HPI & Universität Potsdam

ABSTRACT

Efficient integration of vector databases, such as those containing administrative boundaries and land parcels, with remote sensing images is essential for various Earth Observation (EO) applications. Zonal statistics (ZS) offer a powerful tool for this purpose, but their computation remains challenging due to fragmented system interfaces, diverse preprocessing needs, and inconsistent performance across systems. Current methods optimize execution within single systems but lack support for dynamic, cross-system workflows. To address this, we present Agent Raven, the first AI-driven multi-agent system designed to autonomously manage the full lifecycle of ZS computation and deployment. Building on the Raven framework, Agent Raven dynamically selects execution backends, optimizes query pipelines, and adaptively manages workflows based on previous experiments. Our work represents a step forward in intelligent orchestration across heterogeneous systems in EO data analytics.

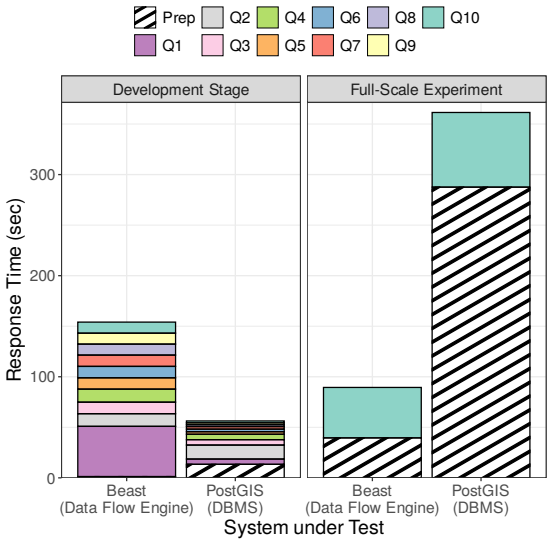

Fig. 1: Performance in different development phases

1. INTRODUCTION

The availability of remote sensing imagery has significantly increased [1, 2, 3] due to advancements in satellite technology. Programs like Copernicus [13] provide vast amounts of freely available raster data, while the volume of vector datasets (e.g., OpenStreetMap, governmental geospatial data) is also expanding. To effectively utilize these data for Earth Observation applications (e.g., climate monitoring, wildfire prediction, urban planning) [18, 17, 14, 16], efficient processing techniques are essential. A key step in these applications is the computation of Zonal Statistics (ZS), where pixel-based raster data are aggregated within defined vector-based geometries, such as city boundaries or farmland parcels. For example, to identify deforested areas, one can apply ZS on satellite images and polygons that define forest boundaries [15].

Computing ZS requires combining raster (gridded cells) and vector (geometric features) data. Geospatial systems such as PostGIS[1] and Beast [5] handle these data types, but their APIs and performance vary widely. This variability forces data scientists to navigate multiple systems, adding complexity and inefficiency. The architecture of each system also impacts its suitability for different stages of a data science project. For example, PostGIS is well-suited for development involving multiple queries on smaller datasets, while Beast is better for large-scale experiments with single-shot queries. Figure 1 illustrates this distinction.

While optimizing individual queries is well-studied in data systems engineering, optimizing the iterative process of refining an analytics pipeline is significantly more challenging. This challenge is even greater for ZS queries, as not all geospatial systems support raster-vector joins equally, forcing users to adjust queries for each system's unique API. In addition, pre-processing steps, such as rasterization, vectorization, format conversion, and coordinate reference system (CRS) alignment, are often necessary, depending on the data model of each system. As queries evolve, the optimal combination of parameters can change significantly. These challenges make it difficult for users to efficiently manage evolving ZS workflows across heterogeneous geospatial systems.

Given these challenges, emerging AI agent technologies offer promising solutions for automating complex geospatial workflows. AI agents are bridging the gap between computer scientists and other users, enabling almost anyone to accom-

[1] https://postgis.net/

plish tasks that once required years of expertise. They are now used across various fields for tasks like visual reasoning [9], code generation [11], scientific experimentation [8], and model interpretation [10]. In geospatial applications, there has been a shift from manually created processing scripts to intelligent agents that autonomously manage remote sensing data, select tools, and refine outputs for tasks like land cover mapping, change detection, or geospatial question answering [7, 12, 6]. These advancements lay the groundwork for dynamic, multi-system geospatial workflows.

We propose Agent Raven, our vision for an AI-powered assistant that supports data scientists at all stages of ZS experimentation - from initial development to continuous deployment. Agent Raven interfaces with the Raven core component [4], our framework for executing ZS queries seamlessly across multiple geospatial systems. By selecting the optimal backend and applying query optimizations based on a database of past experiments, Agent Raven learns and enhances performance over time. To the best of our knowledge, Agent Raven is the first system to offer deep, end-to-end integration of geospatial data science tasks, aiming to significantly reduce development time and operational complexity.

Our contributions are twofold. First, in Section 2, we describe how Raven integrates heterogeneous geospatial systems, providing uniform access and enabling seamless switching between them. This integration simplifies interoperability and lessens the workload for data scientists. Second, in Section 3, we propose Agent Raven, our vision for an AI-driven assistant that supports data scientists across the full lifecycle of ZS experimentation. By leveraging past experiment data, Agent Raven accelerates the transition from early-stage development to robust production pipelines.

2. PLAIN RAVEN FRAMEWORK

Today's data scientists face multiple challenges when implementing zonal statistics, due to the varying interfaces and configuration parameters exposed by existing geospatial systems, the varying pre-processing steps that these systems require, and their divergent runtime performance capabilities. In response to these challenges, Raven aims to: 1) offer an easy-to-use zonal statistics interface; and 2) highlight performance differences in geospatial systems. To achieve this, Raven exposes a declarative zonal statistics interface based on a DSL that we developed. Using this DSL, Raven can transparently optimize and execute a given zonal statistics task on multiple geospatial systems. As a result, Raven provides system independence, thereby helping users avoid vendor lock-ins. Furthermore, by automating execution and providing detailed performance results, Raven simplifies selecting the most efficient system for a given workload. In the following, we give a brief overview of Raven's components.

2.1. Architecture Overview

Figure 2 presents Raven's architecture. Raven accepts a ZS task expressed in its DSL (the query) and relies on its `Pipeline Planner` for optimization. Combined with a `Capabilities` file specifying any system limitations, the planner identifies any necessary pre-processing steps, such as format or CRS conversions, and builds a Pipeline representation that it passes to the `Execution Interface`. This system-developer-provided interface includes a `IR` (Internal Representation) `Converter` and a `GSS` (Geospatial System) `Connector`. The IR Converter translates Raven's abstract syntax tree (AST) into system-specific code using parameterized templates, and the GSS-Connector enables execution on the underlying systems and result retrieval. Additionally, Raven stores execution metrics, e.g., runtime and resource consumption for each step, in its experiment database, which is accessible to other systems. The current systems supported by Raven are PostGIS, Beast, Apache Sedona[2], HeavyDB[3], and RasDaMan[4].

2.2. Zonal Statistics Parameters

To simplify Zonal Statistics (ZS) queries across different geospatial systems, Raven provides a domain-specific language (DSL, Listing **1**) that abstracts system-specific syntax and allows users to define and tune ZS queries in a structured way. We have identified four key operator classes that a tunable ZS query consists of: *Dataset operators* (L. 2–4) specify the raster and vector datasets used for analysis. *Aggregation operators* (L. 6–7) define how pixel values within vector-defined zones are processed, including grouping, filtering, and computing summary statistics. *System operators* (L. 9) determine which geospatial system executes the query. *Execution Parameter operators* (L. 11–12) allow fine-tuning of execution, such as raster tile size adjustment, vector simplification, and CRS alignment.

2.3. Zonal Statistics Pipelines and Optimizations

The AST generated by Raven's `Pipeline Planner` (cf. Figure 2) encapsulates the end-to-end processing of a ZS task. This includes pre-processing operations, such as changing format to support loading into the given system, aligning CRSs, filtering the datasets, as well as the join and aggregation operations. Raven then allows a data scientist or AI agent to configure each of these parameters individually, enabling fine-grained control over the pipeline execution.

We can categorize these parameters into three groups. First, they can reduce the number of processed pixels and vector features as early as possible. Second, they can tune the

[2] https://sedona.apache.org/
[3] https://heavy.ai
[4] https://rasdaman.org/

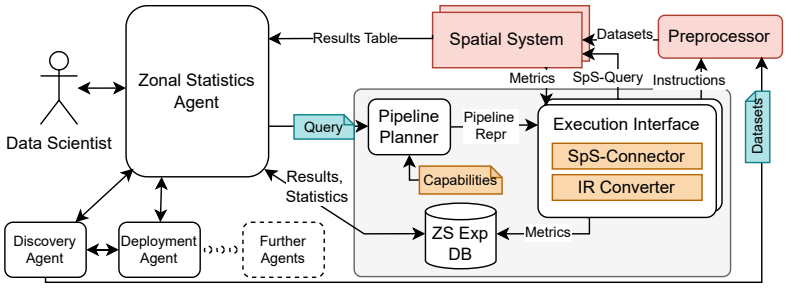

Fig. 2: Agent Raven Architecture

```
# Datasets definition
zs_result = ZSGen(
    raster="/data/vegetation_idx",
    vector="/data/plots") \
# Aggregation operations
  .group("oid") \
  .summarize({"avg": ZSGen.AVG}) \
# Systems
  .system([ZSSystem.Beast]) \
# Parameter settings (optional)
  .raster_tile_size("auto") \
  .vector_simplify([0, 200])
```

Listing 1: ZS Query in Raven's DSL

partitioning of raster and vector data to improve query execution. Third, they can minimize computational overhead by avoiding expensive operations when possible. These tuning techniques leverage the available ZS parameter operators and incorporate methods from existing research. Together, they can speed up ZS queries significantly [4].

2.4. Benchmarking Mode

The performance of ZS tasks in different geospatial systems can vary significantly depending on the data and workload. To uncover this, Raven features a dedicated benchmarking mode. This mode allows users to execute multiple pipelines and produce detailed performance plots, e.g., breakdown performance of different pipeline stages, facilitating easy comparison of different systems and parameter combinations. As a result, users can gain insights into potential bottlenecks and enhance system performance by fine-tuning available parameters. Overall, Raven's integrated benchmarking component provides valuable tools for optimizing zonal statistics tasks across diverse geospatial systems.

3. AGENT RAVEN: AUTONOMOUS ZONAL STATISTICS BEYOND RAVEN

Calculating the results of a ZS task is only one part of broader multi-stage geospatial data science pipelines. To cover other parts of the pipelines, we propose Agent Raven, a multi-agent extension of the plain Raven framework. Agent Raven allows users to describe analytical goals in natural language, while internal agents automatically discover datasets, construct pipelines, invoke tools like plain Raven, and manage execution from development to deployment. It can also retrieve external resources, such as data catalogs or tool manuals, to assist its reasoning during task planning and execution.

3.1. Background on AI Agents for Geospatial Workflows

An AI agent is a system that perceives its environment, reasons about goals, and acts autonomously to fulfill user requests. These agents typically leverage large language models and tools to interpret user intent, plan tasks, access external systems, and manage workflows adaptively. While traditional single-agent systems struggle with scalability, specialization, and responsiveness when workflows become complex, multi-agent systems organize multiple specialized agents under an orchestrator that coordinates their collaboration [6]. Each agent focuses on a smaller set of capabilities, such as dataset discovery, ZS, or deployment, while the orchestrator handles planning, task assignment, and execution monitoring. Multi-agent frameworks can more easily scale across domains, integrate heterogeneous tools, recover from errors, and provide faster intermediate feedback.

Agent Raven uses a multi-agent framework consisting of three agents: the `Discovery Agent`, which identifies and retrieves relevant input datasets; the `Zonal Statistics Agent`, which constructs ZS pipelines and selects appropriate geospatial systems by invoking the core Raven system as a tool; and the `Deployment Agent`, which manages downstream tasks such as continuous monitoring and scheduled deployment. A centralized memory allows the system to improve decision-making across tasks. This shared memory will replace the database used in the plain version of Raven.

3.2. Workflow Example of Agent Raven

We envision Agent Raven as a deeply integrated multi-agent system, where specialized AI agents collaborate to automate the end-to-end ZS workflow. Instead of requiring users to manually script queries, Agent Raven allows users to simply express their goals in natural language. The agents then automatically handle dataset retrieval, pipeline construction, execution, and continuous pipeline deployment.

Consider a data scientist interested in monitoring a specific geospatial area over an extended period, such as tracking the percentage of trees in a given region to observe deforestation. They would interact with Agent Raven by specifying the task and suggesting relevant criteria for suitable datasets. In response, the orchestrator in Agent Raven schedules dataset retrieval to its Dataset Discovery Agent, which searches for and returns a selection of candidate datasets, complete with metadata. To minimize perceived latency and improve user experience, Agent Raven proactively initiates multiple parallel actions. While the Dataset Discovery Agent retrieves

datasets, the ZS Agent begins preparing preliminary pipeline templates based on the user's task description. When candidate datasets are identified, the ZS Agent automatically generates the pipeline representation and invokes the underlying plain Raven engine as a tool to execute the ZS operations. This process includes selecting an efficient geospatial system capable of handling the candidate datasets. To provide early feedback and save resources, Agent Raven initially executes the pipeline on a small geospatial subset, quickly producing preliminary results. In cases where datasets are particularly large, Agent Raven may suggest applying approximate query processing to reduce the dataset size and speed up the query, while trading accuracy. Additionally, Agent Raven leverages its shared memory, which records all past task steps, parameters, and outcomes, to predict optimal configurations based on prior experience, further reducing the need for user intervention. If a dataset appears highly promising, Agent Raven can even pre-run partial queries while awaiting final user confirmation, further improving the perceived latency.

Once the user is satisfied with the preliminary results, Agent Raven will switch over to full-scale experiment mode. It again analyzes all parameters, considering available resources and time, selecting the best system and execution strategy. The final results will be passed to the continuous Deployment Agent, which manages ongoing deployments and regularly updates Agent Raven on its operations. This allows Agent Raven to adapt any parameters if necessary.

4. OUTLOOK

This paper presents Agent Raven, an AI-powered multi-agent system that automates ZS-based EO analytics pipelines across diverse geospatial systems. By adaptively selecting execution systems, optimizing queries, and managing workflows based on historical performance, Agent Raven enhances both the efficiency and accessibility of EO applications.

Looking ahead, we plan to extend Agent Raven with real-time data streaming support and integrate additional geospatial backends. Moreover, we plan to incorporate fault-tolerant execution strategies within the multi-agent framework to ensure robust execution in dynamic environments.

REFERENCES

[1] European Space Agency. Copernicus data space ecosystem, 2024. URL https://dataspace.copernicus.eu/.

[2] Ahmet Kerem Aksoy, Pavel Dushev, Eleni Tzirita Zacharatou, Holmer Hemsen, Marcela Charfuelan, Jorge-Arnulfo Quiané-Ruiz, Begüm Demir, and Volker Markl. Satellite image search in AgoraEO. *PVLDB*, 15(12):3646–3649, 2022.

[3] Arne de Wall, Björn Deiseroth, Eleni Tzirita Zacharatou, Jorge-Arnulfo Quiané-Ruiz, Begüm Demir, and Volker Markl. Agora-EO: A Unified Ecosystem for Earth Observation – A Vision for Boosting EO Data Literacy –. In *Proc. Big Data from Space (BiDS)*, 2021.

[4] Gereon Dusella, Haralampos Gavriilidis, Laert Nuhu, Volker Markl, and Eleni Tzirita Zacharatou. Multi-Backend Zonal Statistics Execution with Raven. In *SIGMOD/PODS*, pages 532–535. ACM, 2024. doi: 10.1145/3626246.3654730.

[5] Ahmed Eldawy et al. Beast: Scalable Exploratory Analytics on Spatio-temporal Data. In *CIKM*, pages 3796–3807. ACM, 2021.

[6] Chaehong Lee et al. Multi-agent geospatial copilots for remote sensing workflows. *ArXiv*, abs/2501.16254, 2025.

[7] Chenyang Liu et al. Change-agent: Towards interactive comprehensive remote sensing change interpretation and analysis. *CoRR*, abs/2403.19646, 2024.

[8] Chris Lu et al. The AI scientist: Towards fully automated open-ended scientific discovery. *CoRR*, abs/2408.06292, 2024.

[9] Dídac Surís et al. Vipergpt: Visual inference via python execution for reasoning. In *ICCV*, pages 11854–11864. IEEE, 2023.

[10] Tamar Rott Shaham et al. A multimodal automated interpretability agent. In *ICML*. OpenReview.net, 2024.

[11] Tanmay Gupta et al. Codenav: Beyond tool-use to using real-world codebases with LLM agents. *CoRR*, abs/2406.12276, 2024.

[12] Wenjia Xu et al. Rs-agent: Automating remote sensing tasks through intelligent agents. *ArXiv*, abs/2406.07089, 2024.

[13] European Commission. Copernicus programme. https://www.copernicus.eu/en, 2025.

[14] Stefanie Holzwarth and et al. Earth Observation Based Monitoring of Forests in Germany: A Review. *Remote Sensing*, 12 (21):3570, January 2020. doi: 10.3390/rs12213570.

[15] Parag Kadam, Nicholas Magnan, and Puneet Dwivedi. A spatial dependence approach to assessing the impacts of Sustainable Forestry Initiative's Fiber Sourcing certification on forestry Best Management Practices in Georgia, United States. *Forest Policy and Economics*, 157:103071, 2023. doi: 10.1016/j.forpol.2023.103071.

[16] Paul J. Pinter, Jr., Jerry L. Hatfield, James S. Schepers, Edward M. Barnes, M. Susan Moran, Craig S.T. Daughtry, and Dan R. Upchurch. Remote Sensing for Crop Management. *Photogrammetric Engineering & Remote Sensing*, 69(6):647–664, 2003. doi: 10.14358/PERS.69.6.647.

[17] Jerry C. Ritchie, Paul V. Zimba, and James H. Everitt. Remote Sensing Techniques to Assess Water Quality. *Photogrammetric Engineering & Remote Sensing*, 69(6):695–704, June 2003. doi: 10.14358/PERS.69.6.695.

[18] Kali E Sawaya, Leif G Olmanson, Nathan J Heinert, Patrick L Brezonik, and Marvin E Bauer. Extending satellite remote sensing to local scales: Land and water resource monitoring using high-resolution imagery. *Remote Sensing of Environment*, 88(1):144–156, 2003. doi: 10.1016/j.rse.2003.04.006.