



Autonomous GIS: the next-generation AI-powered GIS

Zhenlong Li & Huan Ning

To cite this article: Zhenlong Li & Huan Ning (2023) Autonomous GIS: the next-generation AI-powered GIS, International Journal of Digital Earth, 16:2, 4668-4686, DOI: [10.1080/17538947.2023.2278895](https://doi.org/10.1080/17538947.2023.2278895)

To link to this article: <https://doi.org/10.1080/17538947.2023.2278895>



© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



[View supplementary material](#)



Published online: 12 Nov 2023.



[Submit your article to this journal](#)



Article views: 17241



[View related articles](#)



[View Crossmark data](#)



Citing articles: 47 [View citing articles](#)



Autonomous GIS: the next-generation AI-powered GIS

Zhenlong Li^a and Huan Ning^{a,b}

^aGeoinformation and Big Data Research Laboratory, Department of Geography, University of South Carolina, Columbia, SC, USA; ^bDepartment of Geography, The Pennsylvania State University, University Park, PA, USA

ABSTRACT

Large Language Models (LLMs), such as ChatGPT, demonstrate a strong understanding of human natural language and have been explored and applied in various fields, including reasoning, creative writing, code generation, translation, and information retrieval. By adopting LLM as the reasoning core, we introduce Autonomous GIS as an AI-powered geographic information system (GIS) that leverages the LLM's general abilities in natural language understanding, reasoning, and coding for addressing spatial problems with automatic spatial data collection, analysis, and visualization. We envision that autonomous GIS will need to achieve five autonomous goals: self-generating, self-organizing, self-verifying, self-executing, and self-growing. We developed a prototype system called LLM-Geo using the GPT-4 API, demonstrating what an autonomous GIS looks like and how it delivers expected results without human intervention using three case studies. For all case studies, LLM-Geo returned accurate results, including aggregated numbers, graphs, and maps. Although still in its infancy and lacking several important modules such as logging and code testing, LLM-Geo demonstrates a potential path toward the next-generation AI-powered GIS. We advocate for the GIScience community to devote more efforts to the research and development of autonomous GIS, making spatial analysis easier, faster, and more accessible to a broader audience.

ARTICLE HISTORY

Received 31 May 2023



Accepted 30 October 2023


KEYWORDS

Spatial analysis; autonomous agent; artificial intelligence; large language models; ChatGPT

1. Introduction

The rapid development of Artificial Intelligence (AI) has given rise to autonomous agents, also known as intelligent agents (Brustoloni 1991), which are computer systems capable of performing tasks and making decisions with minimal or no human intervention (Wooldridge and Jennings 1995). Autonomous agents have shown great potential in various domains, such as robotics, healthcare, transportation, and finance (Russell and Norvig 2022). They offer numerous benefits, including increased efficiency, reduced errors, and the ability to handle complex tasks that would be time-consuming or impossible for humans to perform (Stone et al. 2022). Different from automatic systems that can perform expected tasks based on the given commands, inputs, and environmental settings (e.g. automatic doors and vending machines), autonomous agents (e.g. vacuum robots and autonomous vehicles) are typically more complex, more adaptable to environments and can make informed decisions based on the data it collects. The most challenging work for

CONTACT Zhenlong Li  zhenlong@sc.edu  Geoinformation and Big Data Research Laboratory, Department of Geography, University of South Carolina, 29072, Columbia, SC, USA

 Supplemental data for this article can be accessed online at <http://dx.doi.org/10.1080/17538947.2023.2278895>

© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

implementing autonomous agents is to build a decision-making core, which needs to react appropriately according to its perception, while the perception and associated actions may not be pre-programmed. Most existing rule- or algorithm-based decision-making cores can only work in specified or closed environments, and their reasoning ability is limited.

The recent advancement of generative models, especially large language models (LLMs) such as GPT-3 (Brown et al. 2020) and GPT-4 (OpenAI 2023), has accelerated the research and development of autonomous agents. These models have demonstrated a strong understanding of human natural language, enabling them to perform tasks in various fields, including reasoning, creative writing, code generation, translation, and information retrieval. As a result, LLMs have been increasingly used as the decision-making core of digital autonomous agents, advancing the development of AI-powered systems. For example, Vemprala et al. (2023) used ChatGPT for robotics by testing various tasks, such as interaction, basic logical, geometrical, and mathematical reasoning. Researchers also used LLM to command pre-trained foundation models (Liang et al. 2023; Shen et al. 2023) to accomplish human tasks via natural languages, such as image manipulation (Wu et al. 2023) and captioning (Orlo 2023). Meanwhile, digital autonomous agents backed by LLMs have been developed, such as AutoGPT (Richards 2023), BabyAGI (Nakajima 2023), and AgentGPT (Reworkd 2023). As an AI-freelancer platform, NexusGPT (“NexusGPT” 2023) released various digital autonomous agents to serve customers, providing services such as business consulting and software development.

The GIScience community has been incorporating AI into geospatial research and applications in recent years, leading to GeoAI, a subfield focusing on the intersection of AI and GIScience (Gao 2020) ; VoPham et al. 2018; Zhanget al. 2021). Li (2020) summarized the three-pillar view of GeoAI in terms of computing, geospatial big data, and AI, emphasizing the computational foundation of GeoAI. Other scholars explored the potential of deep learning techniques to model spatial phenomena, such as using generative adversarial neural network for spatial interpolation (Zhu et al. 2020), using graph convolutional neural network for spatial regression (Zhu et al. 2022) and flow analytics (Gallo et al. 2021). Meanwhile, GeoAI enriches the data acquisition approaches and inspires new applications, such as real-time flood risk assessment from social media (Alizadeh Kharazi and Behzadan 2021; Ning et al. 2020), metric mapping using street view imagery (Ning et al. 2022a; Ning et al. 2022b; Ning et al. 2022c), and rapid environmental monitoring based on remote sensing imagery (Jiang et al. 2022; Shafique et al. 2022). These advancements demonstrate AI-enhanced spatial applications by developing efficient data extraction algorithms and deep learning spatial models.

While GeoAI has made significant strides in utilizing AI for various geospatial applications, the exploration and adoption of artificial general intelligence (AGI) (e.g. LLM as an early stage of AGI) in spatial analysis and GIS remains in its early stages. One notable by Mai et al. (2023) explores opportunities and challenges of developing an LLM-like foundation model for GeoAI. In a recent envision of GIS from the humanistic aspect, Zhao (2022) classified GIS into four categories: Embodiment GIS, Hermeneutic GIS, Autonomous GIS, and Background GIS by considering their relation (distance) to human, GIS, and place. Zhao regarded autonomous GIS ‘as either an independent agent or a place’, such as drones, robot vacuums, and autonomous vehicles, or models trained to recognize land objects in images (e.g. buildings). Other attempts include using LLM to automate simple data operations (e.g. loading) in QGIS (QChatGPT 2023; Mahmood 2023) or adopting pre-trained models to segment images (Wu 2023).

In this paper, we take a different approach to explore the integration of AI and GIS by narrowing down the connotation of Autonomous GIS as an AI-powered GIS that leverages LLM’s general abilities in natural language understanding, reasoning, and coding for addressing spatial problems with automatic spatial data collection, analysis, and visualization. While autonomous GIS follows a similar concept to autonomous agents, most existing agents discussed above focus on text-based information retrieval, analysis, and summation, such as for business reports and travel planning, which do not necessarily have a unique, correct answer. Their design principles and architectures cannot be well adapted for GIS and spatial analysis, which are data-intensive and typically have only one

correct answer, e.g. the population living within 10 kilometers of a hospital. From the implementation aspect, we suggest that autonomous GIS should be designed as a programming- and data-centric framework that uses automatic coding to address the GIScience questions of deterministic nature.

To demonstrate the feasibility of autonomous GIS, we developed a proof-of-concept prototype called LLM-Geo, which can conduct spatial analysis in an autonomous manner. LLM-Geo receives tasks (spatial problems/questions) from users and generates a solution graph (geoprocessing workflow) by decomposing the task into successive connected data operations as a directed acyclic graph. Each operation is a function to be implemented by the LLM, which is GPT-4 in this study. Next, LLM-Geo generates a combined program by integrating the codes of all operations and executes the combined program to produce the final result of the task. Three case studies are used to test the ability of LLM-Geo. The results indicate that the integration of LLMs into GIS has the potential to revolutionize the field by automating complex spatial analysis tasks and making GIS technology more accessible to individuals without GIS backgrounds. The rapid advancements in AI prompted a moonshot by Janowicz et al. (2020): ‘Can we develop an artificial GIS analyst that passes a domain-specific Turing Test by 2030?’. Our study, prototyping such an artificial GIS analyst, resonates with this ambitious vision. We are optimistic that the realization of this moonshot may come to fruition even before 2030. We believe that autonomous GIS serves as a potential path toward the next generation of AI-powered GIS.

The remainder of this paper is organized as follows: Section 2 elaborates on the concept and design considerations of autonomous GIS as an AI-powered autonomous system. Section 3 introduces the implementation of LLM-Geo as the prototype of autonomous GIS, followed by three case studies in Section 4. Sections 5 and 6 discuss what we have learned from LLM-Geo, its limitations, and potential future research directions. Section 7 concludes the paper.

2. Autonomous GIS as an AI-powered autonomous system

Autonomous systems are designed to make decisions and perform tasks without human intervention. They can adapt to changing conditions, learn from their environments, and make informed decisions based on the data they collect. By incorporating LLMs (or AGI) as the decision-making core for generating strategies and steps to solve spatial problems, autonomous GIS will be capable of searching and retrieving needed spatial data either from extensive existing online geospatial data catalogs or collecting new data from sensors, and then using existing spatial algorithms, models, or tools (or developing new ones) to process gathered data to generate the final results, e.g. maps, charts, or reports. LLM can be considered as the ‘brain’ of autonomous GIS or the ‘head’ if equipped with environmental sensors, while executable programs (e.g. Python) can be considered as its digital ‘hands’. Similar to other autonomous agents (Sifakis 2019), we propose that autonomous GIS requires five critical modules, including decision-making (LLM as the core), data collecting, data operating, operation logging, and history retrieval. These modules enable GIS to achieve five autonomous goals: self-generating, self-organizing, self-verifying, self-executing, and self-growing (Table 1).

Table 1. Autonomous goals and modules of autonomous GIS.

Autonomous Goal	Involving Modules	Functionality
Self-generating	Decision-making, Data collecting	Generate solutions and data operation programs
Self-organizing	Decision-making, Operation logging	Ensure the operations are executed in the correct order, and data is stored in an appropriate manner
Self-verifying	Decision-making, Data operating	Test and verify the generated workflow, code, and programs
Self-executing	Data operating, Operation logging	Execute generated workflows, code, or programs
Self-growing	Operation logging, History retrieval	Reuse the verified operations

Unlike many autonomous agents for non-deterministic tasks with no strict standards to assess answers such as writing a poem, GIScience applications require quantitative computation and analysis of spatial data to provide answers. Deterministic systems such as GIS are those in which the system's future state can be precisely predicted, given sufficient information about its initial conditions and the underlying rules governing its behavior. In a deterministic system, there is a direct relationship between the initial conditions and the outcome, with no room for randomness or uncertainty. In this sense, autonomous GIS needs a strictly controllable and explainable approach for the answer, and such answers should be unique and quantitatively correct based on the given data, e.g., the population living within 10 kilometers of a hospital. Therefore, we suggest that autonomous GIS should be designed as a programming-centric framework to use automatic coding to address the GIScience questions of deterministic nature. Note that programming-centric refers to the internal implementation of autonomous GIS; the users do not necessarily need to interact with the code as a programmer.

3. LLM-Geo: a prototype of autonomous GIS

We implemented two critical modules of autonomous GIS in LLM-Geo: decision-making and data operating, achieving three autonomous goals: self-generating, self-organizing, and self-executing. Additional modules are currently under development. The decision-making module adopts an LLM (GPT-4 in this study) as a core, or a 'brain', to generate step-by-step solution workflow and develop associated codes of each step for addressing various spatial questions. The data operating module is a Python environment to execute the generated code, such as spatial data loading, processing, visualization, and saving.

Figure 1 shows the overall workflow of how LLM-Geo answers questions. The process begins with the user inputting the spatial question along with associated data locations such as online data URLs, REST (REpresentational State Transfer) services, and API (Application Program Interface) documentation. Then, the LLM generates a solution graph similar to a geoprocessing workflow. Based on the solution graph, LLM-Geo sends the requirements of each operation node to LLM, requesting code implementation. LLM-Geo then gathers all operation code implementations and asks LLM to generate an assembly program that connects the operations based on the workflow. Finally, LLM-Geo executes the assembly program to produce the final answer.

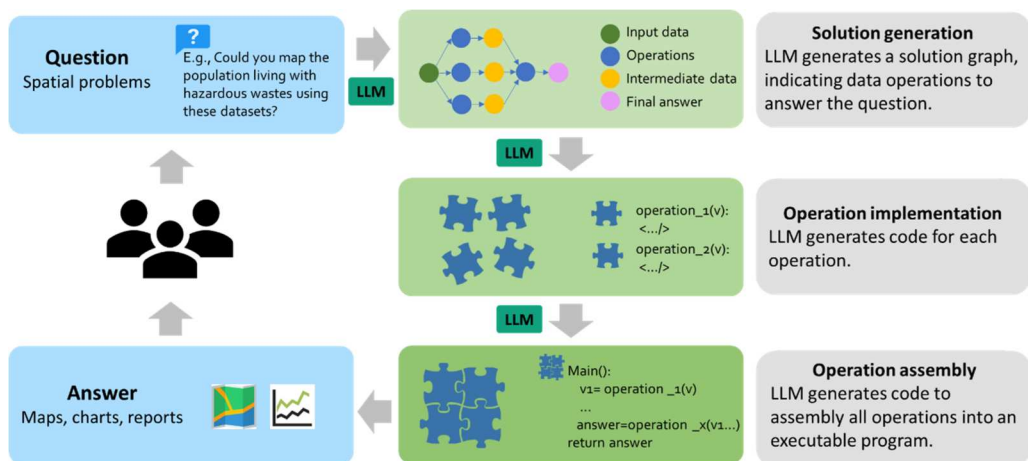


Figure 1. The overall workflow of LLM-Geo.

3.1. Solution generation

A solution in LLM-Geo refers to a geoprocessing workflow consisting of a series of connected operations and data. Performing these operations to process data in succession generates the final result for the question, such as maps, charts, tables, or new spatial datasets. We use a directed graph to represent the workflow and classify all nodes into two categories: data and operation. A *data node* refers to an operation’s input data or output data, consisting of three types: input data node, intermediate data node, and output data node. The input data nodes tell the system where to load input data through local data paths, URLs, or REST APIs for programmatic data access. The output data nodes are the final results for the task or question which can be numeric data, interactive or static maps, tables/charts, new datasets, or other user-requested types. All other data nodes are considered intermediate data nodes. An operation node is a process to manipulate data with input and output. Its inputs can be the input data nodes or intermediate data nodes from the ancestor operation nodes. Its output can be intermediate data nodes for descendent operation nodes or the output nodes (final result). The directed edges indicate the data flow. Disconnected nodes are not allowed in the solution graph, as all data flows will need to be converged at the output nodes to produce the final results. These structural constraints (guidance) of the solution graph are fed to LLM (GPT-4) via API along with the question from the user. Using these guidance, LLM-Geo automatically generates the solution graph by determining the needed data nodes and operation nodes as well as the executing order of the operations based on the reasoning of a specific spatial problem. Since spatial analysis is essentially a geoprocessing workflow consisting of a series of connected spatial data processing tasks, the solution graph ensures that data flows seamlessly through the processing steps, ultimately converging at the final results (Figure 2). Online Appendix 1 shows a sample of the prompt and LLM returned code for the solution graph.

Our experiments indicate that the granularity of an operation node is determined by LLM on-the-fly based on the complexity of the question and the maximum token length supported by the API. Due to the token length limit, LLM may have difficulty decomposing complex tasks into detailed graphs, resulting in a solution graph with lower granularity and fewer nodes. As all current LLMs have token limits (including state-of-the-art models such as GPT-4), this practical constraint necessitates a recursive approach for complex tasks. This approach can further decompose an operation node into a sub-solution graph containing more granular operations until the granularity reaches an appropriate level for accurate code generation and reusability.

3.2. Operation implementation

The generated solution graph serves as a ‘data processing plan’ for a specified spatial question where the input data, output data, and data operations are defined. In the operation implementation stage, each operation will be implemented as an executable code snippet following the operation definition

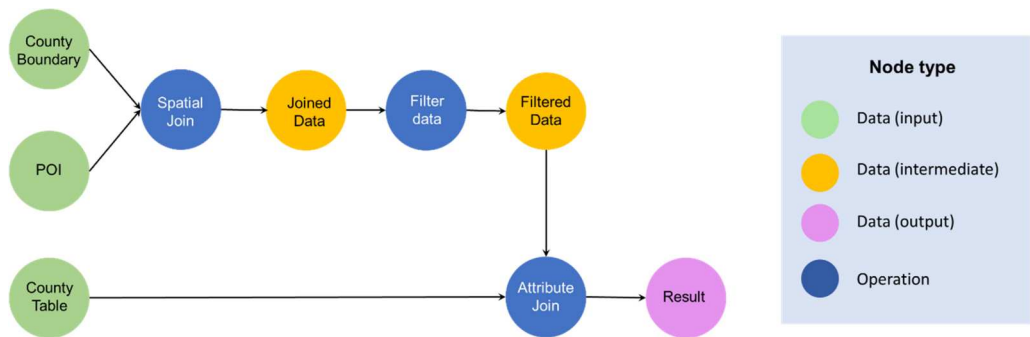


Figure 2. Illustration of a solution graph with different node types.

in the solution graph. We use Python runtime as the data operation module in LLM-Geo, so the operation implementation is to generate Python code by LLM. LLM-Geo uses an algorithm to create interfaces between nodes to ensure the operation nodes and data nodes connect to their adjacent nodes. For example, the current version of LLM-Geo generates a Python function for each operation; the function definition and return data (i.e. names of functions and function input/output variables) are pre-defined in the solution graph. This strategy is used to reduce the uncertainty of code generation.

According to our experiments, GPT-4 requires sufficient information and extra guidance for reliable code generation. In this context, all information related to the solution graph paths preceding the current operation node is needed. For example, an ancestor node might create a new column or file, and LLM must know and use the name of that column and file when generating function code for the current operation node. Additionally, information about descendant nodes is also needed to guide the code generation process. Therefore, LLM-Geo feeds the generated code in ancestor nodes and information of descendant nodes to LLM to request code implementation for each operation node. Besides node information, extra guidance is necessary. For instance, we found that GPT-4 has hazy memory regarding the prerequisites of some spatial data operations, such as the same column data type for table joining and identical map projection for overlay analysis. GPT-4 has learned to use GeoPandas, a popular Python library for spatial data processing; however, it seems unaware that this library does not support on-the-fly reprojection. In such cases, extra guidance is needed for LLM to generate correct code implementation for operation nodes. Table 2 presents some guidance that LLM-Geo used in the operation implementation stage. Online Appendix 2 provides a sample of the prompt and LLM returned code for an operation.

3.3. Operation assembly

After generating the code (functions) for all operation nodes, LLM-Geo collects the codes and submits them to LLM, along with the solution graph and pre-defined guidance, to create a final program for the task. Based on the guidance, LLM uses the solution graph to determine the execution sequence of the operation nodes in the final program. Intermediate variables are generated to store the output data of operation nodes, which are then fed to the subsequent operation nodes. Finally, LLM-Geo executes the assembly program to produce the final result of the spatial question. The entire implementation structure and workflow of LLM-Geo based on GPT-4 API

Table 2. Example guidance for operation code generation.

- 1 DO NOT change the given variable names and paths.
- 2 Put your reply into a Python code block(enclosed by ```python and ```), NO explanation or conversation outside the code block.
- 3 If using GeoPandas to load zipped ESRI files from a URL, load the file directly, DO NOT unzip ESRI files. E.g. gpd.read_file(URL)
- 4 Generate descriptions for input and output arguments.
- 5 You need to receive the data from the functions, DO NOT load data in the function if other functions have loaded the data and returned it in advance.
- 6 Note the module 'pandas' has no attribute 'StringIO'
- 7 Use the latest Python module methods.
- 8 When doing spatial analysis, convert the involved layers into the same map projection.
- 9 When joining tables, convert the involved columns to string type without leading zeros.
- 10 When doing spatial joins, remove the duplicates in the results. Or please think about whether it needs to be removed.

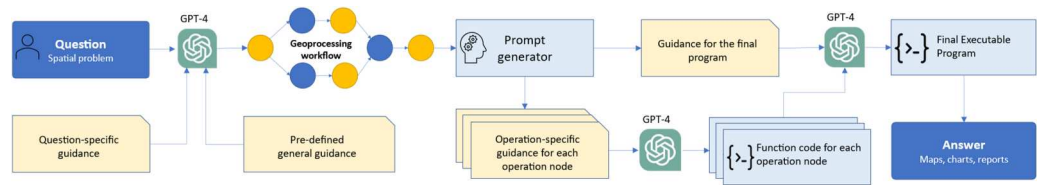


Figure 3. Implementation workflow of LLM-Geo with GPT-4 API.

is shown in Figure 3. Online Appendix 3 includes a sample of the prompt and LLM returned code for an assembly program.

4. Case studies

4.1. Case 1: counting population living near hazardous wastes

This spatial problem is to find out the population living with hazardous wastes and map their distribution. The study area is North Carolina, United States (US). We input the task (question) to LLM-Geo as shown in Box 1.

Box 1. The spatial problem submitted to LLM-Geo for Case 1.

Task:

- 1) Find out the total population that lives within a Census tract that contain hazardous waste facilities. The study area is North Carolina, US.
- 2) Generate a map to show the spatial distribution of population at the tract level and highlight the borders of tracts that have hazardous waste facilities.

Data locations:

1. NC hazardous waste facility ESRI shape file location: https://github.com/gladcolor/LLM-Geo/raw/master/overlay_analysis/Hazardous_Waste_Sites.zip
2. NC tract boundary shapefile location: https://github.com/gladcolor/LLM-Geo/raw/master/overlay_analysis/tract_shp_37.zip. The tract ID column is 'Tract'
3. NC tract population CSV file location: https://github.com/gladcolor/LLM-Geo/raw/master/overlay_analysis/NC_tract_population.csv. The population is stored in 'TotalPopulation' column. The tract ID column is 'GEOID'

This question asks LLM-Geo to find out the total population living with hazardous wastes and generate a map for the population distribution with sufficient details for the analysis, including the data location and used column names. Note that the vector layers (hazardous facilities and tract boundaries) are not in the same map projection, and the tract ID data type is *text* in the boundary layer, but it is *integer* when Pandas reads the population CSV file. GPT-4 has hazy memory to pre-process these inconsistencies, so LLM-Geo needs to remind GPT-4 with extra guidance in the prompt: 'When doing spatial analysis, convert the involved layers into the same map projection. When joining tables, convert the involved columns to string type without leading zeros.'

Figure 4 shows the outputs of LLM-Geo for this case study, including a solution graph detailing the data processing steps, the final assembly Python program, and the final results for the question: the total population (5,688,769) that lives within a tract that contains hazardous waste facilities and a map showing the spatial distribution of population at the Census tract level and highlight the borders of tracts that have hazardous waste facilities. Our manual verification confirmed the accuracy of the number and the map. With the working code generated from LLM-Geo, users can easily adjust and re-run the code to customize the map visualization styles if needed.

4.2. Case 2: human mobility data retrieval and trend visualization

This task investigates the mobility changes during COVID-19 pandemic in France in 2020. First, we asked LLM-Geo to retrieve mobility data from the ODT Explorer using REST API (Li et al. 2021), and then compute and visualize the monthly change rate compared to January 2020. We input the task to LLM-Geo as shown in Box 2. Figure 5 shows the results from LLM-Geo for this case study, including a solution graph detailing the data processing steps, a map matrix showing the spatial distribution of the mobility change rate, a line chart showing the trend of the mobility change rate, and the final assembly program that produced the outputs. All results are correct per our manual verification.

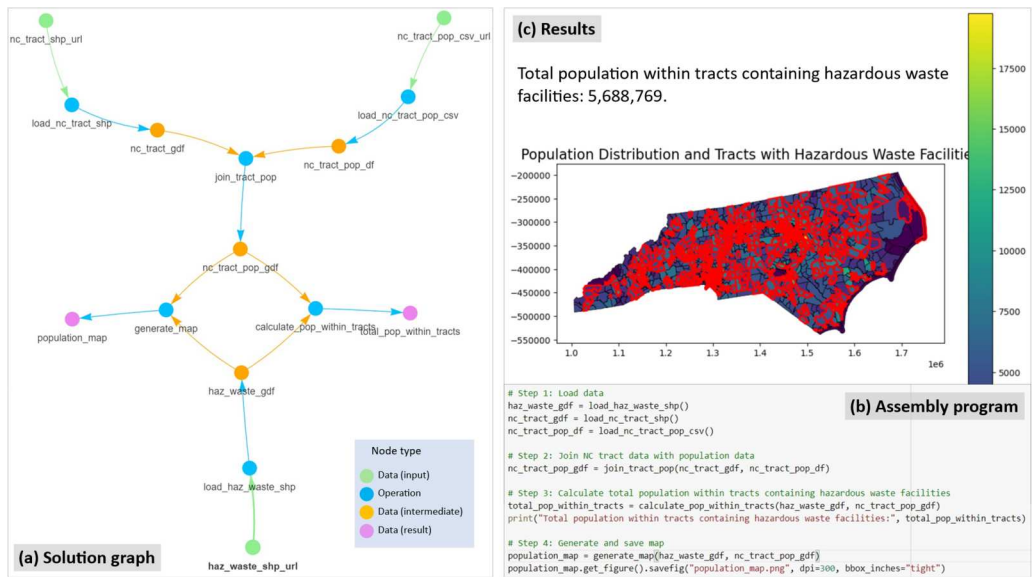


Figure 4. Results automatically generated by LLM-Geo for counting the population living near hazardous wastes. (a) Solution graph, (b) assembly program (Python codes), and (c) returned population count and generated map.

Box 2. The spatial problem submitted to LLM-Geo for Case 2.

Task:

- 1) Show the monthly change rates of population mobility for each administrative regions in a France map. Each month is a sub-map in a map matrix. The base of the change rate is January 2020.
- 2) Draw a line chart to show the monthly change rate trends of all administrative regions. The x-axis is month.

Data locations:

1. ESRI shapefile for France administrative regions: https://github.com/gladcolor/LLM-Geo/raw/master/REST_API/France.zip. The 'GID_1' column is the administrative region code, 'NAME_1' column is the administrative region name.
2. REST API URL with parameters for mobility data access: http://gis.cas.sc.edu/GeoAnalytics/REST?operation=get_daily_movement_for_all_places&source=twitter&scale=world_first_level_admin&begin=01/01/2020&end=12/31/2020. The response is in CSV format. There are three columns in the response: place, date (format:2020-01-07), and intra_movement. 'place' column is the administrative region code, France administrative regions start with 'FRA'.

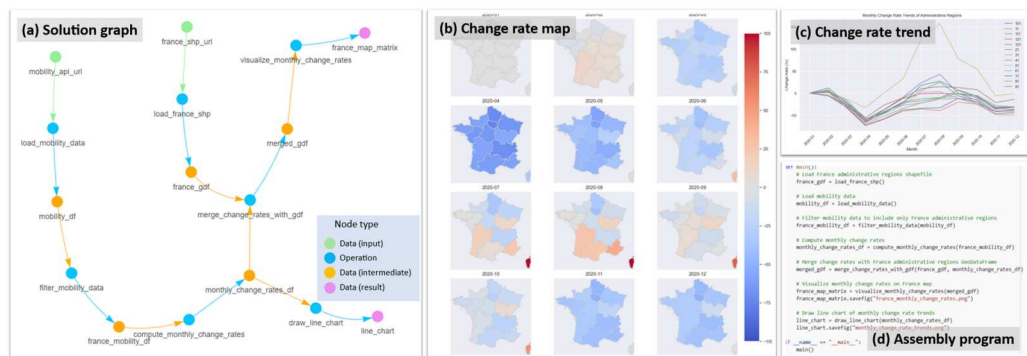


Figure 5. Results automatically generated by LLM-Geo for human mobility data retrieval and trend visualization. (a) Solution graph, (b) map matrix showing the spatial distribution of mobility change rate, (c) line chart showing the trend of the mobility change rate, (d) assembly program.

4.3. Case 3: COVID-19 death rate analysis and visualization at the US county level

The spatial problem for this case is to investigate the spatial distribution of the COVID-19 death rate (ratio of COVID-19 deaths to cases) and the association between the death rate and the proportion of senior residents (age ≥ 65) at the US county level. The death rate is derived from the accumulated COVID-19 data as of December 31, 2020, available from New York Times (2023), based on state and local health agency reports. The population data is extracted from the 2020 ACS five-year estimates (US Census Bureau 2022). The task asks for a map to show the county level death rate distribution and a scatter plot to show the correlation and trend line of the death rate with the senior resident rate. Box 3 shows the spatial problem along with the needed data-set locations submitted to LLM-Geo. The results are presented in Figure 6.

Box 3. The spatial problem submitted to LLM-Geo for Case 3.

Task:

- 1) Draw a map to show the death rate (death/case) of COVID-19 among the contiguous US counties. Use the accumulated COVID-19 data of 2020.12.31 to compute the death rate. Use scheme = 'quantiles' when plotting the map. Set map projection to 'Conus Albers'. Set map size to 15*10 in..
- 2) Draw a scatter plot to show the correlation and trend line of the death rate with the senior resident rate, including the r -square and p -value. Set data point transparency to 50%, regression line as red. Set figure size to 15*10 in..

Data locations:

- 1) COVID-19 data case in 2020 (county-level): <https://github.com/nytimes/covid-19-data/raw/master/us-counties-2020.csv>. This data is for daily accumulated COVID cases and deaths for each county in the US. There are 5 columns: date (format: 2021-02-01), county, state, fips, cases, deaths.
- 2) Contiguous US county boundary (ESRI shapefile): https://github.com/gladcolor/spatial_data/raw/master/contiguous_counties.zip. The county FIPS column is 'GEOID'.
- 3) Census data (ACS2020): https://raw.githubusercontent.com/gladcolor/spatial_data/master/Demography/ACS2020_5year_county.csv. The needed columns are: 'FIPS', 'Total Population', 'Total Population: 65 to 74 Years', 'Total Population: 75 to 84 Years', 'Total Population: 85 Years and Over'.

In this example, we provided more requirements on the visualization style of the map and chart, such as 'Use scheme = 'quantiles' when plotting the map. Set map projection to "Conus Albers". Set map size to 15*10 in.'. Without such guidance, the system often picks different settings for each request of the same problem. The entire generated executable program can be found in Online Appendix 4. By changing the date '2020-12-31' in the code and re-run the program, one can easily explore the COVID-19 death rate distribution and its association with the senior resident rate for other periods. Readers can find more cases in the GitHub code repository.

5. Discussion and lessons learned

5.1. Forms of autonomous GIS

Autonomous GIS aims to mimic human operations in spatial analysis rather than changing its foundation and procedure. In this sense, autonomous GIS can function as a standalone application (local or cloud-based), acting as a GIS analyst who takes questions from users and produces answers. One potential graphic user interface (GUI) for such an application is illustrated in Figure 7 (a), where users input the description of a spatial problem, click the *Submit* button, and receive the results directly within the application. The application should also be able to display the generated solution graph (geoprocessing workflow) and executable codes that produce the final results for monitoring, debugging, and customization purposes. Autonomous GIS's self-grow ability can enable the system grow rapidly by accumulating and reusing the generated code and workflows, especially the cloud version that serves many users.

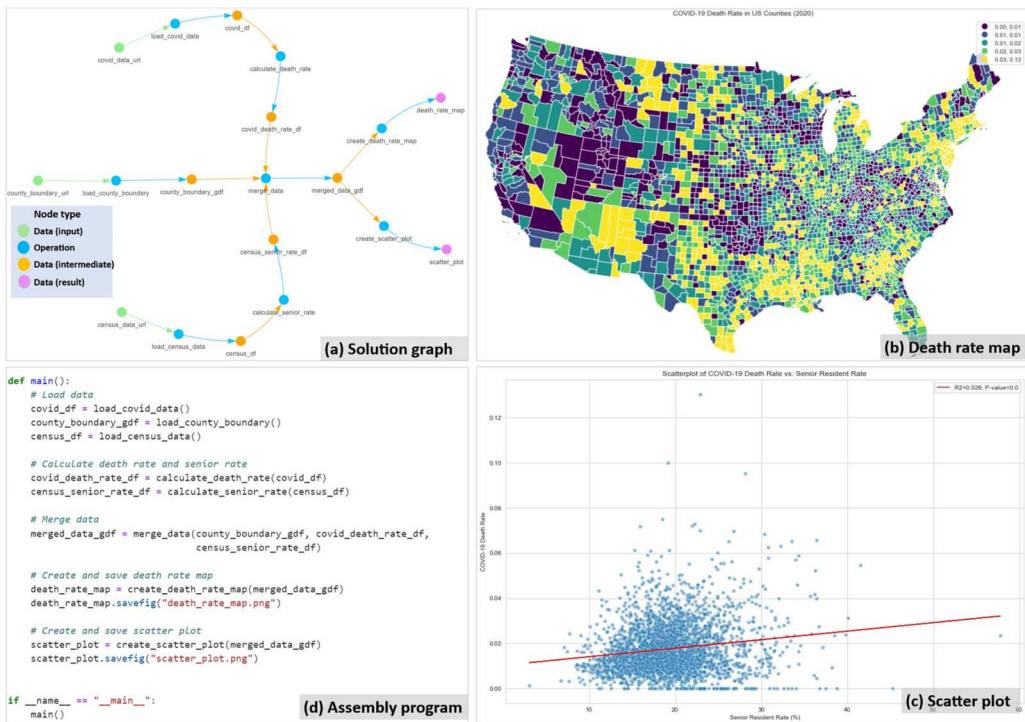


Figure 6. Results automatically generated by LLM-Geo for the US county level COVID-19 death rate analysis and visualization. (a) Solution graph, (b) county level death rate map of the contiguous US, (c) scatter plot showing the association between COVID-19 death rate and the senior resident rate at the county level, (d) assembly program.

In addition to the standalone form, autonomous GIS can serve as a co-pilot for traditional GIS software, using natural language to communicate with users and automate spatial data processing and analysis tasks. This setup is similar to Microsoft's Co-pilot for its Office family, which automates office tasks such as report writing and slide creation (Microsoft 2023). For example, an autonomous GIS panel alongside the map view can be integrated into ArcGIS and QGIS, displaying the chatbox, generated solution graph, and codes as illustrated in Figure 7 (b). The result will be shown in the built-in map view. Users can modify the workflow by editing the solution graph or operation parameters. If an operation is implemented by the generated code, users can locate and edit the code

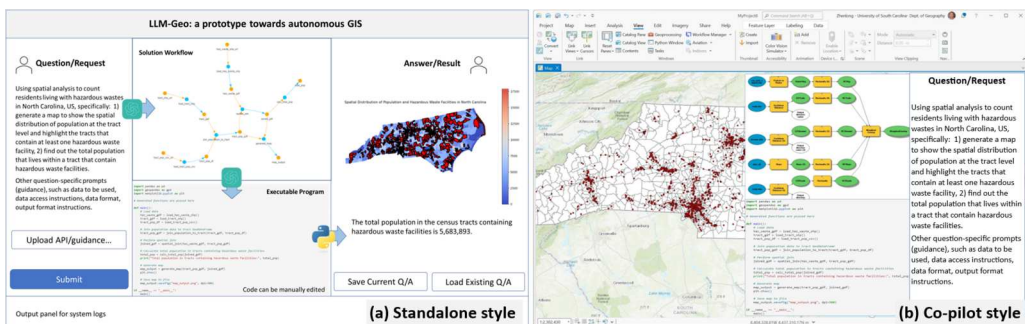


Figure 7. Illustration of the user interface for (a) standalone autonomous GIS, (b) co-pilot style autonomous GIS integrated in existing GIS software (e.g. ArcGIS Pro or QGIS).

Note: the question, workflow, codes and results in this figure are for demonstration purposes only.

by clicking the solution graph nodes. Overall, the solution graph is similar to the Module Builder of ArcGIS or Model Designer in QGIS. Implementing autonomous GIS based on existing GIS platforms may be the most practical and efficient approach at present, as mature GIS platforms (e.g. ArcGIS) already have a rich number of operations (e.g. the tools in the ArcGIS toolbox) along with well-established documentation that LLM can quickly learn and use to generate the solution graph.

More broadly, autonomous GIS can be used as a ‘plugin’ integrated with applications requiring geospatial data manipulation, such as a public health surveillance dashboard for COVID-19 or other diseases. Such an autonomous GIS-enabled dashboard will have the ability to interact with users using natural language to generate customized maps, charts, reports, new datasets based on the users’ specific needs and preferences. This level of interaction and customization allows users to have a more personalized and efficient experience when working with geospatial data in various applications.

It is worth noting that although autonomous GIS is designed and implemented as ‘programming-centric’, users do not necessarily be programmers when interacting with autonomous GIS. From the user perspective, autonomous GIS is user friendly since the input can be natural language description for the analysis task. No matter the standalone or co-pilot forms, we think there should be a visual interface to show and edit the solution graph and a code editor to revise code directly. If the autonomous GIS fails to generate correct results, users can edit the solution graph in the visual interface or modify the code. ArcGIS Model Builder and QGIS Model Designer are good examples of visualizing and editing the generated solution graph by an autonomous GIS. The visualization brings easy understanding and manipulation for GIS beginners, while code representation may be more efficient and flexible for GIS experts. One of the ultimate goals of autonomous GIS is ‘programming-free’ even for GIS advanced users.

5.2. Difference between autonomous GIS and AI assistants

There are various AI assistants backed by LLMs available in the market. For example, Cursor (Cursor 2023) and GitHub Co-pilot X (GitHub 2023) are similar products that utilize LLMs to generate code and comments to enhance programmers’ productivity. Microsoft’s Co-pilot for its Office family is able to automate office tasks such as report writing and slide creation. What distinguishes autonomous GIS from these AI assistants lies in the inputs, final products, and the deterministic and data-centric workflow generating the results. The AI assistants often receive text and generate text-based content for users, such as paragraphs or code. In contrast, the input for GIS analysis is spatial data, which is further processed to derive quantitative results (e.g. maps, charts) to support decision-making.

Furthermore, GIS users typically create geoprocessing workflows (Scheider et al. 2021) involving spatial data and GIS tools based on their analysis objectives, experience, and knowledge. These workflows can take the form of scripts, graphically connected data and operations using a model builder, or a sequence of manual steps in GIS software. These workflows are deterministic, data-centric, and can be replicated by others. The design of the autonomous GIS prototype (LLM-Geo) follows a similar problem-solving approach by first generating a data-centric geoprocessing workflow, implementing the workflow with programming, and executing the workflow to produce the final results in an automated manner. It is important to note that LLM does not directly manipulate spatial data; instead, it generates and commands operations (programs) to handle the data.

5.3. Divide-and-conquer

One might reasonably question why we do not opt for a more direct approach — asking LLMs like GPT-4 for solutions without generating a solution graph and operations. Indeed, our tests have shown that with extra guidance embedded in the prompt, GPT-4 is capable of generating correct

code for relatively simple spatial analysis tasks. However, this direct method may limit the LLM's capacity to tackle more complex tasks such as identifying the most promising fishing site in a specific sea area or assessing the accessibility of Autism intervention services at a national level using smartphone mobility data APIs and web crawling. These tasks are akin to writing a long novel, and it is unlikely that LLMs would generate comprehensive, one-step solutions that involve extensive programs. Such tasks are challenging not only for humans but also for models (Maeda 2023).

Human problem-solving often employs a divide-and-conquer strategy when facing complex tasks. By adopting this approach in the design of LLM-Geo, it is intended to address spatial analysis tasks by breaking complex problems into smaller, more manageable sub-problems that LLMs can handle (i.e. operations). It then addresses these sub-problems (i.e. developing a function for each sub-problem) and ultimately combines these functions (i.e. assembling a program) to yield the final results. This divide-and-conquer methodology has been employed in other digital autonomous agents, such as AutoGPT (Richards 2023) and AgentGPT (Reworkd 2023). We plan to further explore the feasibility of this approach with tasks that are more complex than those demonstrated thus far. Additionally, the divide-and-conquer strategy aids in the production of verified operation nodes (e.g. code snippets). These validated sub-solutions (operations) become valuable assets for autonomous GIS as they can be reused for future tasks, bolstering the autonomous goal of self-growing.

5.4. The need for sufficient information in GIS and LLM

As previously discussed, GIS is deterministic in nature that allows for precise predictions of their future states given enough knowledge about their starting conditions and the rules that govern their operations. In these deterministic systems, outcomes are intrinsically tied to their initial conditions, leaving no room for randomness or uncertainty. Sufficient information is vital in deterministic systems, as it facilitates accurate prediction and comprehension of the system's behavior. For GIS, metadata or details like layer projections, column names, and data types, are critical for subsequent spatial programming. Much like a human analyst, autonomous GIS needs to be aware of these constraints and information to successfully accomplish the given task. For example, one is unable to perform an attribute join operation without prior knowledge of the column names or conduct an overlay analysis without information on layer projections. Therefore, we believe that the provision of sufficient information is necessary for autonomous GIS's accurate reasoning, task planning, and action execution. Possible methods to supply such information include data sampling, web searches, self-reflection, and manual provision.

5.5. Recall the hazy memory of LLM

LLMs can correct previous error output by incorporating additional information. They understand the steps required to complete a task, but often overlook practical constraints in spatial analysis, such as the necessity of matching map projection for overlay analysis and precise data type matching for the common joining column of two attribute tables. This sort of 'hazy' memory in LLM is likely resulted from limited GIS training materials. To enhance LLMs' recall of these hazy memories, and thus improving their capacity for successful reasoning and coding, it is crucial to embed necessary reminders within the task description. Despite the potential variation in this hazy spatial analysis memory across different LLMs and training datasets, it is feasible to formulate a relatively universal reminder list to aid autonomous GIS in producing accurate results. This list is similar to a checklist for GIS users. For instance, during spatial analysis tasks, GIS users must ensure that the common joining columns have the same data type (string or int) and leading zero digits. Experienced GIS users typically verify these prerequisites in advance, while novices may overlook these steps until they encounter errors or

consult a checklist. The specificity of this guidance, or checklist, will vary among users (e.g. shorter for experienced GIS users) and will also differ for autonomous GIS depending on the sophistication of the employed LLMs.

5.6. Explainability and trustworthiness concerns

Explainability (Creswell and Shanahan 2022; S. Bills et al. 2023) and trustworthiness (Bhandari and Brennan 2023; Huang et al. 2023) are two pivotal concerns in LLMs. Deliveries of autonomous agencies such as LLM-Geo need to consider to what degree the decisions from LLMs can be explained and trusted. However, it is still challenging to understand what is happening inside LLMs when they generate results. In addition, results verification is not straightforward when conducting reasoning tasks. Research on the explainability of LLMs is still under development. Steven Bills et al. (2023) investigated the use of GPT-4 to explain ‘what patterns in the text cause a neuron to activate.’ The low fitness between the actual and simulated activation levels indicates a long way to go to LLM explanation.

Creswell and Shanahan (2022) adopted multiple reasoning step to improve LLM explainability. Two fine-tuned LLMs collaborated for reasoning: the first one chooses elements or fact from given context to avoid hallucination, and the second one executes reasoning according to the selected context. Such a trace shows improvement and transparency on reasoning. Similarly, LLM-Geo employs a multi-step strategy tailored for spatial analysis tasks. Central to the approach is the generation of a solution graph. While users do not know the internal workings leading to the creation of such a solution graph, the graph itself offers a clear, human-readable data processing workflow which can be manually verified. The transparency of the solution graph and operation code is a supplement to the blackbox nature of LLM, providing the control of reasoning back to the users.

Trustworthiness refers to the extent users can rely on a LLM to produce accurate, unbiased, and safe outputs (Huang et al. 2023; Kang, Gao, and Roth 2022). In the context of autonomous GIS, akin to humans, LLM makes mistakes. For example, the mis-joining in a spatial join operation caused by different data types of columns can lead to unexpected results. We recommend the GIScience community to develop a benchmark to assess the trustworthiness of autonomous GIS and establish inspection rules to review the generated solution graph and operations.

6. Limitations and future work

While LLM-Geo validates the concept of autonomous GIS, it is still in its infancy with a number of limitations, such as the inability to collect data or to process images. The advent of reasoning capabilities in LLMs brings the GIScience community to the dawn of a new era, yet there is still a vast landscape of opportunities to explore and challenges to address. We have identified several potential avenues for further research and development to tackle the existing limitations of LLM-Geo and lay the groundwork for a fully operational autonomous GIS in the future.

6.1. The adaptivity of LLM-Geo needs to be improved

We developed a number of case studies to test LLM-Geo and reported three typical cases in Section 4 with a success rate of about 80% based on GPT-4 with self-code-review and self-debugging modules without human intervention. Our observations indicate that the generated solution graphs were accurate. The majority of failures could be traced back to faulty code during the operation implementation stage; even with the divide-and-conquer approach, a single error statement can crash the entire program. Although GPT-4 usually utilizes the wide-used GeoPandas library (Jordahl et al. 2020) for spatial analysis, it often struggles with generating correct code in a single attempt. Furthermore, the rate of errors increases when it uses other libraries such as Plotly

(2023). Currently, LLM-Geo lacks a mechanism to test and verify the generated code. These modules are needed to achieve the autonomous goal of *self-verifying*.

Another observation is that LLM-Geo tends to develop complex functions from scratch based on GeoPandas or other open-source spatial libraries for a spatial operation (e.g. nearest distance calculation) that is already available in the traditional and well-developed GIS packages or toolboxes (e.g. QGIS and ArcGIS). Such generated complex functions are error-prone, even with appropriate guidance. Compared to the current code generation approach, we believe that LLM-Geo could be more adaptive and robust by utilizing the well-established and tested GIS toolboxes coupled with the code debugging and verification module. Another limitation is that LLM-Geo is currently incapable of knowing the data, such as map projections, data types, and attributes. A sophisticated mechanism needs to be designed to manage these interactions between LLM and code and data in an automated manner.

In addition, arithmetic operations are sometimes needed in spatial analysis. However, LLMs like GPT-4 display limitations in arithmetic reasoning as noted by Qian et al. (2022). Efforts to address these shortcomings are underway, as documented by Lu et al. (2023). Various strategies have been proposed, including employing specific training datasets (Lewkowycz et al. 2022; Yuan et al. 2023), implementing causal analysis (Stolfo et al. 2023), developing a ‘chain of thought’ approach (Wei et al. 2023), and integrating plug-ins (e.g. Wolfram). Notably, Imani, Du, and Shrivastava (2023) introduced a multi-verification technique called ‘MathPrompter.’ This technique uses various algebraic expressions or Python functions to tackle a mathematical problem in different approaches and then cross-check the results to increase the likelihood of correctness. Such techniques can be adopted in future autonomous GIS development to enhance its arithmetic reasoning ability.

6.2. A memory system is needed

Every autonomous system needs a memory component to store the contextual and long-term information, data, and results for future retrieval or reuse, a feature that LLM-Geo currently lacks. While LLM-Geo does use a simple method to store contextual memory, including prompts and LLM responses, this is insufficient for an autonomous system. Autonomous GIS’s memory can be data sources, pre-defined analysis guidance, and verified codes generated in previous tasks. These memory elements are expected to expand over time to achieve the self-growing autonomous goal. Most LLM-based autonomous agents (e.g. AutoGPT) use vector databases such as Pinecone (Pinecone 2023) to store and retrieve conversations. Autonomous GIS faces a more complicated requirement than text storage and retrieval as it needs to store the solution graph (geoprocessing workflow), which would become hierarchical to recursively solve complex tasks. Moreover, the verified solutions and operation codes should be part of the long-term information to be memorized. Thus, a memory system along with a logging module is needed to achieve the autonomous goal of *self-growing*.

6.3. Categorized guidance maintained by GIS community

Despite being aware of the hazy memory exhibited by GPT-4, we managed to supplement it with extra guidance, steering it toward generating accurate operation nodes for data manipulation. Like the documentation of open-sourced packages like GDAL/OGR, GIS guidance for LLMs contains the experience of GIS analysts and serves as a handbook for LLMs, which can play a critical role in automating spatial analysis. The current guidance is being updated throughout the development of LLM-Geo. However, such guidance summarized from a few case studies is far from covering the most common spatial analysis scenarios. For example, the case studies did not include raster data analyses, which are typical in GIS applications. We invite the GIS community to contribute GIS guidance for LLMs to enhance their spatial programming (code generation) capabilities. Given the potential length of such guidance documents, they should be categorized. We could, for

example, have separate guidance documents for the analysis of vector data, raster data, network data, and geo-visualization. The memory system of autonomous GIS should be designed to retrieve necessary guidance only for specific tasks rather than read them all. Researchers can also investigate how to summarize spatial analysis guidance based on the autonomous GIS's trials and errors. This includes exploring ways to guide LLM to generate and execute various analyzing tasks to produce guidance using reinforcement learning.

6.4. Trial-and-error, a more robust problem-solving approach

The current implementation of LLM-Geo assumes that once all necessary information is fed into LLM, the correct result will be returned. This expectation is somewhat idealistic as spatial analysis is often complex in terms of data sources, algorithms, and hypotheses. In practice, analysts are more likely to adopt a trial-and-error problem solving strategy, exploring various possible paths and pruning unsuccessful branches in the solution graph. This strategy mirrors the human problem solving approach, as it is common for even experienced programmers to encounter errors during code development. Yao et al. (2023)'s experiments about 'tree of thoughts' demonstrated substantial improvement in the reasoning of LLMs on the game of 24, creative writing, and mini crosswords. Their 'tree of thoughts' is similar to a trial-and-error strategy: evaluate a possible branch, and then determine whether to cut it off. However, the simplicity of the tasks used in their experiments limits the generalizability of their findings in more complex domains such as spatial analysis.

6.5. Online geospatial data discover and filtering

Our case studies provided data for LLM-Geo, such as data file paths, URLs, or data access APIs, as well as data descriptions. However, an autonomous GIS should be capable of collecting required data independently to finish the task. Most LLM-based autonomous agents are equipped with search engines (using APIs) to search and retrieve data from the internet. Fortunately, numerous geospatial datasets, standard geospatial web services (e.g. OGC WMS, WFS, WCS, WPS), and REST APIs for geospatial data access have been established in recent decades. For example, US population data can be extracted from the US Census Bureau (2023) via API, as well as from OpenStreetMap (OpenStreetMap 2023). Large online geospatial data catalogs have also been created by various national or international organizations and government agencies, such as the Google Earth Engine Data Catalog, NOAA and NASA Data Catalogs, EarthCube, and the European Environment Agency (EEA) geospatial data catalog. The challenge lies in finding and selecting suitable and high-quality data layers in terms of spatiotemporal resolution, spatiotemporal coverage, and accuracy. Autonomous GIS developers need to establish practical strategies for LLMs to discover, filter, and utilize the most relevant and accurate geospatial datasets (including those that require an approved account, password, or token to access) for a given spatial analysis task.

6.6. Answer 'why' questions

LLM-Geo, powered by GPT-4, illustrates the ability of knowing 'how' to perform spatial analysis and automate a multitude of routine geospatial tasks. The next challenge lies in the ability of autonomous GIS to address 'why' questions, which often requires a more profound understanding and investigation of geospatial knowledge. Questions such as 'According to the smartphone mobility data, why did my customer numbers drop by 20% this month?' or 'Why did these migratory birds alter their path in the past decade?' extend beyond basic spatial analysis and delve into the domain of hypothesis generation, data selection, and experimental design. To answer these questions, autonomous GIS needs to have the ability to design research by formulating informed hypotheses based on the posed question, available data, and context.

6.7. Build a Large Spatial Model (LSM)

LLMs, trained on extensive text corpora, have developed language skills, knowledge, and reasoning abilities, but their spatial awareness remains limited due to the scarcity of spatial samples within these corpora. Many resources in GIScience, such as abundant historical remote sensing images, global vector data, detailed records of infrastructure and properties, and vast amounts of other geospatial big data sources, have yet to be fully incorporated into the training of large models. Consider the potential of a Large Spatial Model (LSM), trained on all available spatial data (e.g. all raster data pixels), mirroring the way LLMs are trained on extensive text corpora. Such a model could potentially possess a detailed understanding of the Earth's surface, accurately describe any location, and comprehend the dynamics of ecosystems and geospheres. This would not only enrich the spatial awareness of future artificial general intelligence, but could also transform the field of GIScience, empowering autonomous GIS to answer 'why' questions. We advocate for further research and efforts towards the training of Large Spatial Models that can more accurately represent the Earth's surface and human society.

7. Conclusion

In this study, we presented autonomous GIS as the next-generation of AI-powered geographic information systems. The goal of autonomous GIS is to accept tasks via natural language and solve spatial problems with minimal to no human intervention, aimed at making GIS and spatial analysis more accessible and user-friendly. By employing LLM as the core reasoning mechanism, we suggest that autonomous GIS should achieve five autonomous objectives: self-generating, self-organizing, self-verifying, self-executing, and self-growing. To illustrate this concept, we developed a prototype of autonomous GIS, LLM-Geo, using the GPT-4 API within a Python environment, demonstrating what an autonomous GIS could look like and how it can deliver results autonomously. In the three case studies, LLM-Geo successfully produced the expected results, such as aggregated numbers, charts, and maps, significantly reducing manual operation time. Although still in its infancy, LLM-Geo demonstrates the concept and feasibility of autonomous GIS. Echoing the vision of Zhu et al. (2021) that the 'Next generation of GIS must be easy', we believe autonomous GIS as the next-generation AI-powered GIS holds great promise towards achieving this goal. We encourage the GIScience community to dedicate more efforts towards the research and development of autonomous GIS, making spatial analysis easier, faster, and more accessible to a broader audience.

Acknowledgements

We thank Professor Gregory J. Carbone for helping revise an earlier draft of this manuscript.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Author contributions

The authors declare equal contribution to this paper.

Data and codes availability statement

The source code of LLM-Geo and case study data are provided at <https://github.com/gladcolor/LLM-Geo>.

References

- Alizadeh Kharazi, Bahareh, and Amir H. Behzadan. 2021. "Flood Depth Mapping in Street Photos with Image Processing and Deep Neural Networks." *Computers, Environment and Urban Systems* 88: 101628. <https://doi.org/10.1016/j.compenvurbsys.2021.101628>.
- Bhandari, Prabin, and Hannah Marie Brennan. 2023. "Trustworthiness of Children Stories Generated by Large Language Models." <http://arxiv.org/abs/2308.00073> (August 13, 2023).
- Bills, Steven, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. "Language Models Can Explain Neurons in Language Models." May 9, 2023. <https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html>.
- Brown, Tom B., et al. 2020. "Language Models Are Few-Shot Learners." <http://arxiv.org/abs/2005.14165> (May 4, 2023).
- Brustoloni, Jose C.. 1991. Autonomous agents: Characterization and requirements. School of Computer Science, Carnegie Mellon University.
- Creswell, Antonia, and Murray Shanahan. 2022. "Faithful Reasoning Using Large Language Models." *arXiv preprint arXiv:2208.14271*.
- Cursor. 2023. "Cursor | Build Fast." <https://www.cursor.so/> (May 5, 2023).
- Gallo, Massimo, Alessandro Finamore, Gwendal Simon, and Dario Rossi. 2021. "FENXI: Deep-Learning Traffic Analytics at the Edge." In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, 202–13.
- Gao, Song. 2020. "A Review of Recent Researches and Reflections on Geospatial Artificial Intelligence." *Geomatics and Information Science of Wuhan University* 45 (12): 1865–1874. <https://doi.org/10.13203/j.whugis20200597>.
- GitHub. 2023. "Introducing GitHub Copilot X." *GitHub*. <https://github.com/features/preview/copilot-x> (May 5, 2023).
- Huang, Yue, Qihui Zhang, Philip S. Y, and Lichao Sun. 2023. "TrustGPT: A Benchmark for Trustworthy and Responsible Large Language Models." <http://arxiv.org/abs/2306.11507> (August 13, 2023).
- Imani, Shima, Liang Du, and Harsh Shrivastava. 2023. "MathPrompter: Mathematical Reasoning Using Large Language Models." <http://arxiv.org/abs/2303.05398> (August 6, 2023).
- Janowicz, Krzysztof, et al. 2020. "GeoAI: Spatially Explicit Artificial Intelligence Techniques for Geographic Knowledge Discovery and Beyond." *International Journal of Geographical Information Science* 34 (4): 625–636. <https://doi.org/10.1080/13658816.2019.1684500>.
- Jiang, Huiwei, et al. 2022. "A Survey on Deep Learning-Based Change Detection from High-Resolution Remote Sensing Images." *Remote Sensing* 14 (7): 1552. <https://doi.org/10.3390/rs14071552>.
- Jordahl, Kelsey, et al. 2020. "Geopandas/Geopandas: V0.8.1." *Zenodo*. <https://ui.adsabs.harvard.edu/abs/2020zndo...3946761J> (May 19, 2023).
- Kang, Yuhao, Song Gao, and Robert Roth. 2022. "A Review and Synthesis of Recent GeoAI Research for Cartography: Methods, Applications, and Ethics." In *Proceedings of AutoCarto*, 2–4.
- Lewkowycz, Aitor, et al. 2022. "Solving Quantitative Reasoning Problems with Language Models." <http://arxiv.org/abs/2206.14858> (August 7, 2023).
- Li, Wenwen. 2020. "GeoAI: Where Machine Learning and Big Data Converge in GIScience." *Journal of Spatial Information Science* 20: 71–77.
- Li, Zhenlong, et al. 2021. "ODT FLOW: Extracting, Analyzing, and Sharing Multi-Source Multi-Scale Human Mobility." *PLoS One* 16 (8): e0255259.
- Liang, Yaobo, et al. 2023. "TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs." <http://arxiv.org/abs/2303.16434> (April 25, 2023).
- Lu, Pan, et al. 2023. "A Survey of Deep Learning for Mathematical Reasoning." <http://arxiv.org/abs/2212.10535> (August 7, 2023).
- Maeda. 2023. "Schillace Laws of Semantic AI." <https://learn.microsoft.com/en-us/semantic-kernel/howto/schillacelaws> (May 6, 2023).
- Mahmood, Mohammed. 2023. "QGPT Agent Documentation." Python. <https://github.com/momaabna/QGPTAgent>.
- Mai, Gengchen, et al. 2023. "On the Opportunities and Challenges of Foundation Models for Geospatial Artificial Intelligence." <http://arxiv.org/abs/2304.06798> (April 20, 2023).
- Microsoft. 2023. "Introducing Microsoft 365 Copilot – Your Copilot for Work." *The Official Microsoft Blog*. <https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/> (May 5, 2023).
- Nakajima, Yohei. 2023. "BabyAGI." <https://github.com/yoheinakajima/babyagi> (May 4, 2023).
- New York Times. 2023. "Coronavirus (Covid-19) Data in the United States (Archived)." <https://github.com/nytimes/covid-19-data> (May 18, 2023).
- "NexusGPT." 2023. <https://nexus.snkipic.io> (May 2, 2023).
- Ning, Huan, Zhenlong Li, Michael E. Hodgson, and Cuizhen (Susan) Wang. 2020. "Prototyping a Social Media Flooding Photo Screening System Based on Deep Learning." *ISPRS International Journal of Geo-Information* 9 (2): 104. <https://doi.org/10.3390/ijgi9020104>.

- Ning, Huan, Zhenlong Li, Cuizhen Wang, et al. 2022a. "Converting Street View Images to Land Cover Maps for Metric Mapping: A Case Study on Sidewalk Network Extraction for the Wheelchair Users." *Computers, Environment and Urban Systems* 95: 101808. <https://doi.org/10.1016/j.compenvurbsys.2022.101808>.
- Ning, Huan, Zhenlong Li, Xinyue Ye, et al. 2022b. "Exploring the Vertical Dimension of Street View Image Based on Deep Learning: A Case Study on Lowest Floor Elevation Estimation." *International Journal of Geographical Information Science* 36 (7): 1317–1342. <https://doi.org/10.1080/13658816.2021.1981334>.
- Ning, Huan, Xinyue Ye, et al. 2022c. "Sidewalk Extraction Using Aerial and Street View Images." *Environment and Planning B: Urban Analytics and City Science* 49 (1): 7–22. <https://doi.org/10.1177/2399808321995817>.
- OpenAI. 2023. "GPT-4 Technical Report." <http://arxiv.org/abs/2303.08774> (May 4, 2023).
- OpenStreetMap. 2023. "Databases and Data Access APIs - OpenStreetMap Wiki." https://wiki.openstreetmap.org/wiki/Databases_and_data_access_APIs (May 5, 2023).
- Orlo. 2023. "AI Caption Generator Powered by ChatGPT." Orlo. <https://orlo.tech/features/generate/> (May 4, 2023).
- Pinecone. 2023. "Vector Database for Vector Search." Pinecone. <https://www.pinecone.io/> (May 19, 2023).
- Plotly. 2023. "Plotly: Low-Code Data App Development." <https://plotly.com/> (May 24, 2023).
- QChatGPT. 2023. Python. KIOS Research and Innovation Center of Excellence, University of Cyprus. <https://github.com/KIOS-Research/QChatGPT>
- Qian, Jing, et al. 2022. "Limitations of Language Models in Arithmetic and Symbolic Induction." <http://arxiv.org/abs/2208.05051> (August 7, 2023).
- Reworkd. 2023. "AgentGPT: Autonomous AI in Your Browser." <https://agentgpt.reworkd.ai/> (May 4, 2023).
- Richards, Toran Bruce. 2023. "Auto-GPT: An Autonomous GPT-4 Experiment." <https://github.com/Torantulino/Auto-GPT> (April 13, 2023).
- Russell, S. J., and Peter Norvig. 2022. *Artificial Intelligence: A Modern Approach*, 4th, Global Ed..
- Scheider, Simon, Enkhbold Nyamsuren, Han Kruiger, and Haiqi Xu. 2021. "Geo-Analytical Question-Answering with GIS." *International Journal of Digital Earth* 14 (1): 1–14. <https://doi.org/10.1080/17538947.2020.1738568>.
- Shafique, Ayesha, et al. 2022. "Deep Learning-Based Change Detection in Remote Sensing Images: A Review." *Remote Sensing* 14 (4): 871. <https://doi.org/10.3390/rs14040871>.
- Shen, Yongliang, et al. 2023. "HuggingGPT: Solving AI Tasks with ChatGPT and Its Friends in HuggingFace." <http://arxiv.org/abs/2303.17580> (April 2, 2023).
- Sifakis, Joseph. 2019. "Autonomous Systems – An Architectural Characterization." In *Models, Languages, and Tools for Concurrent and Distributed Programming: Essays Dedicated to Rocco De Nicola on the Occasion of His 65th Birthday, Lecture Notes in Computer Science*, edited by Michele Boreale, Flavio Corradini, Michele Loret, and Rosario Pugliese, 388–410. Cham: Springer International Publishing. (May 3, 2023). https://doi.org/10.1007/978-3-030-21485-2_21
- Stolfo, Alessandro, et al. 2023. "A Causal Framework to Quantify the Robustness of Mathematical Reasoning with Language Models." <http://arxiv.org/abs/2210.12023> (August 7, 2023).
- Stone, Peter, et al. 2022. "Artificial Intelligence and Life in 2030: The One Hundred Year Study on Artificial Intelligence." <http://arxiv.org/abs/2211.06318> (May 4, 2023).
- US Census Bureau. 2022. "American Community Survey 2016-2020 5-Year Data Release." *Census.gov*. <https://www.census.gov/newsroom/press-kits/2021/acs-5-year.html> (May 25, 2023).
- US Census Bureau. 2023. "Available APIs." *Census.gov*. <https://www.census.gov/data/developers/data-sets.html> (May 5, 2023).
- Vemprala, Sai, Rogerio Bonatti, Arthur Buckner, and Ashish Kapoor. 2023. *ChatGPT for Robotics: Design Principles and Model Abilities*.
- VoPham, Trang, Jaime E. Hart, Francine Laden, and Yao-Yi Chiang. 2018. "Emerging Trends in Geospatial Artificial Intelligence (GeoAI): Potential Applications for Environmental Epidemiology." *Environmental Health* 17 (1): 1–6. <https://doi.org/10.1186/s12940-017-0345-y>.
- Wei, Jason, et al. 2023. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." <http://arxiv.org/abs/2201.11903> (August 7, 2023).
- Wooldridge, Michael, and Nicholas R. Jennings. 1995. "Intelligent Agents: Theory and Practice." *The Knowledge Engineering Review* 10 (2): 115–152. <https://doi.org/10.1017/S0269888900008122>.
- Wu, Qiuhseng. 2023. "Segment-Geospatial." <https://github.com/opengeos/segment-geospatial> (May 19, 2023).
- Wu, Chenfei, et al. 2023. "Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models." *arXiv.org*. <https://arxiv.org/abs/2303.04671v1> (April 8, 2023).
- Yao, Shunyu, et al. 2023. "Tree of Thoughts: Deliberate Problem Solving with Large Language Models." <http://arxiv.org/abs/2305.10601> (May 24, 2023).
- Yuan, Zheng, et al. 2023. "How Well Do Large Language Models Perform in Arithmetic Tasks?" <http://arxiv.org/abs/2304.02015> (August 7, 2023).
- Zhang, Yongsheng, et al. 2021. "Progress and Challenges of Geospatial Artificial Intelligence." *Acta Geodaetica et Cartographica Sinica* 50 (9): 1137.
- Zhao, Bo. 2022. "Humanistic GIS: Toward a Research Agenda." *Annals of the American Association of Geographers* 112 (6): 1576–1592. <https://doi.org/10.1080/24694452.2021.2004875>.

- Zhu, Di, et al. 2020. "Spatial Interpolation Using Conditional Generative Adversarial Neural Networks." *International Journal of Geographical Information Science* 34 (4): 735–758. <https://doi.org/10.1080/13658816.2019.1599122>.
- Zhu, Di, Yu Liu, Xin Yao, and Manfred M. Fischer. 2022. "Spatial Regression Graph Convolutional Neural Networks: A Deep Learning Paradigm for Spatial Multivariate Distributions." *GeoInformatica* 26 (4): 645–676. <https://doi.org/10.1007/s10707-021-00454-x>.
- Zhu, A-Xing, Fang-He Zhao, Peng Liang, and Cheng-Zhi Qin. 2021. "Next Generation of GIS: Must Be Easy." *Annals of GIS* 27 (1): 71–86. <https://doi.org/10.1080/19475683.2020.1766563>.