

# Benchmarking Large Language Models for Geolocating Colonial Virginia Land Grants

Ryan Mioduski, Independent Researcher

2025-05-10

*Code and data available at: <https://github.com/ryanmio/colonial-virginia-llm-geolocation>*

## Abstract

Virginia’s seventeenth- and eighteenth-century land patents survive almost exclusively as narrative metes-and-bounds descriptions in printed abstract volumes such as *Cavaliers and Pioneers* (C&P) (Nugent, 1979). I present the first systematic study of whether state-of-the-art large language models (LLMs) can convert these prose abstracts into usable latitude/longitude coordinates at research grade. I digitized, transcribed, and openly released a corpus of 5,471 Virginia patent abstracts (1695–1732), accompanied by a rigorously annotated ground-truth dataset of 45 authoritatively georeferenced test cases. I benchmark six OpenAI models spanning three architecture families—o-series reasoning models, flagship GPT-4-class chat models, and GPT-3.5—under two prompting paradigms: (i) one-shot “direct-to-coordinate” and (ii) tool-augmented chain-of-thought that invokes external geocoding APIs.

On the verified grants, the best purely textual model (OpenAI o3-2025-04-16) achieves a mean great-circle error of 23.4 km (median 14.3 km), a 67% improvement over a professional GIS baseline (71.4 km), while cutting cost and latency by roughly two and three orders of magnitude, respectively. The ultracheap GPT-4o variant locates patents with 28 km mean error at USD 1.09 per 1,000, only slightly less accurate yet  $\sim 100\times$  cheaper, defining a new dollar-for-accuracy Pareto frontier. Contrary to expectations, granting LLMs external geocoding tools neither improves accuracy nor consistency. Robustness checks across temperature, reasoning-budget, and abstract length confirm these findings.

These results show that off-the-shelf LLMs can georeference early-modern land records faster, cheaper, and as accurately as traditional GIS workflows, opening a scalable pathway to spatially enable colonial archives—and, in turn, to reassess settlement dynamics, plantation economies, and Indigenous dispossession with quantitative precision.

## 1 Introduction

### 1.1 Historical Context & Motivation

Virginia’s colonial land patents are a cornerstone resource for scholars studying settlement patterns, the political economy of plantation agriculture, and Indigenous dispossession in the seventeenth and eighteenth centuries. Yet the spatial dimension of these sources remains under-exploited: most patents survive only as narrative metes-and-bounds descriptions in printed abstract volumes such as *Cavaliers and Pioneers* (C&P)

(Nugent, 1979). Without geographic coordinates, historians and archaeologists cannot readily visualise how land ownership evolved or test hypotheses with modern Geographic Information System (GIS) tools. Creating a machine-readable, georeferenced version of C&P would unlock new quantitative approaches to long-standing questions about colonial Virginia’s social and environmental history.

Digitising and geo-locating the abstracts, however, is notoriously labour-intensive. Even professional GIS analysts can spend several hours per grant reconciling archaic place-names, inconsistent spellings, and low-resolution boundary calls. Recent breakthroughs in large language models (LLMs) suggest a new pathway: language-driven spatial reasoning where a model reads the patent text and predicts latitude/longitude directly or with minimal tool assistance. This study explores whether state-of-the-art LLMs can shoulder that burden accurately and cheaply enough to matter for digital history.

## 1.2 Problem Statement

Despite the promise of LLMs, their ability to extract usable coordinates from early-modern archival prose had not been systematically evaluated prior to this work. Key uncertainties I addressed included:

- Could a model trained mostly on contemporary text understand seventeenth-century toponyms and bearing conventions?
- Would providing API-based tools (e.g., Google Places search) materially improve accuracy relative to a pure text approach?
- How did model predictions compare to the professional GIS benchmark in both error and cost?

Addressing these questions required a rigorously annotated test bench that blended historical sources, modern GIS ground truth, and controlled prompt engineering.

## 1.3 Contributions

This study makes four principal contributions:

1. I release the first machine-readable edition of *Cavaliers and Pioneers*, Vol. 3 (Nugent, 1979), comprising 5,471 fully transcribed patent abstracts.
2. For forty-five of 125 randomly sampled patents I derive authoritative latitude/longitude pairs from state-archived GIS polygons and other archival sources, providing a high-fidelity evaluation target.
3. I evaluate two prompting paradigms—single-prompt “direct-to-coordinate” inference and tool-augmented chain-of-thought reasoning—across six contemporary OpenAI model variants.
4. I quantify the trade-offs among spatial error, monetary expense, and processing time, demonstrating that a pure LLM pipeline can match or surpass professional GIS accuracy while operating orders of magnitude faster and cheaper.

All data, code, and results are available in my public repository: <https://github.com/ryanmio/colonial-virginia-llm-geolocation>.

## 2 Background & Related Work

### 2.1 Historical GIS and Land-Grant Mapping

Digitizing colonial-era land grants has long promised new insights into European settlement patterns, Indigenous land displacement, and the development of local economies. However, this potential has been constrained by the extensive manual labor required to convert metes-and-bounds descriptions into spatial data. Traditional approaches to georeferencing these historical records have proven prohibitively time-consuming - a genealogical case study by Julian and Abbott (Julian & Abbott, 2014) required nearly ten years of archival sleuthing and three university-semester GIS projects to pinpoint a single family's land claims across three Tennessee counties.

Several institutional efforts have attempted to address these challenges, though coverage remains incomplete. The Library of Virginia maintains a statewide *Land Patents and Grants* online database hosting scanned images and searchable indices for every recorded patent (1623–1774) and subsequent grant (1779–2000), including Northern Neck surveys, but provides no ready-made GIS polygons, limiting its direct utility for spatial analysis (Library of Virginia, 2025). Similarly, the Virginia Surveyor's Office has released thousands of patent polygons for central regions of the colony, while Loudoun County GIS staff have successfully reconstructed all original grants within their jurisdiction (Loudoun County Office of Mapping, 2015). These initiatives demonstrate the feasibility of digitizing historical land records but also highlight significant gaps in existing datasets - many seventeenth- and eighteenth-century patents, particularly those recorded in Cavaliers and Pioneers (Nugent, 1979), still lack spatial coordinates.

Among the most thorough academic efforts for Virginia's Northern Neck proprietary are Mitchell's (Mitchell, 1977) maps and companion text documenting the "Beginning at a White Oak" patents of Fairfax County. This work reconstructed hundreds of early land grants with polygonal boundaries, establishing both the feasibility and research value of transforming metes-and-bounds descriptions into spatial data. Building on such foundations, scholars have leveraged available georeferenced grants for substantive historical analysis. In Virginia, seminal studies like Fausz (Fausz, 1971) utilized narrative patent abstracts to trace settlement patterns along the James River basin, while noting the persistent challenges of transforming textual descriptions into precise spatial coordinates for quantitative analysis.

This analytical potential extends beyond Virginia. Dobbs (Dobbs, 2009) used georeferenced North Carolina grants to demonstrate that eighteenth-century town sites often followed pre-existing Indigenous trails, while Coughlan and Nelson (Coughlan & Nelson, 2018) leveraged a dataset of 1,160 South Carolina grants to model settlement patterns based on river access and soil fertility. In each case, spatial enablement of historical records revealed patterns difficult to discern through textual sources alone.

In genealogical and historical research communities, semi-automated solutions have emerged to assist with this labor-intensive process. DeedMapper software (Direct Line Software, 2010) helps researchers convert metes-and-bounds descriptions into visual plots, though it still requires manual entry of deed text and expert positioning of parcels on reference maps. Professional development courses from the Salt Lake Institute of Genealogy (SLIG) continue to teach these specialized mapping techniques, reflecting the still-developing state of automation in this field.

The literature establishes three critical facts. First, historians value land-grant GIS layers because they unlock settlement and landscape questions that text alone cannot answer. Second, traditional platting methods are too slow and too localized to deliver colony-scale coverage. Third, the piecemeal datasets that do exist furnish both ground truth and a methodological benchmark for any attempt at automation. This study addresses this bottleneck by testing whether large language models can shoulder the coordinate-extraction burden—potentially transforming Virginia's colonial patents from archival prose to research-ready GIS at scale.

## 2.2 Large Language Models for Geolocation

Building on the manual coordinate-extraction bottleneck outlined in § 2.1, recent advances in large language models (LLMs) suggest that much of the geoparsing pipeline can now be automated. **Coordinate extraction**—sometimes called *geoparsing*—comprises two subtasks: (i) identifying candidate toponyms in running text and (ii) resolving each mention to a unique set of latitude/longitude coordinates.

The evolution of this field has moved through several distinct methodological phases. Rule-based gazetteer look-ups dominated early work, providing limited accuracy when dealing with ambiguous place names. Neural architectures such as CamCoder (Gritta et al., 2018) subsequently improved performance through learned contextual representations. Most recently, fine-tuned large language models have demonstrated substantial breakthroughs in toponym resolution accuracy.

A representative example of this latest approach comes from Hu et al. (Hu & Kersten, 2024), who adapted 7–13 billion-parameter open-source models (e.g., Mistral-7B, Llama 2-7B) to generate an unambiguous administrative string for each toponym before invoking a standard geocoding API. Their fine-tuned Mistral-7B achieved an **Accuracy@161 km of 0.91**, outperforming previous neural methods by multiple percentage points and improving toponym resolution accuracy by 13%; on the less ambiguous WikToR corpus the same architecture reached 0.98. Crucially, these gains were realized on commodity hardware, underscoring the practicality of parameter-efficient fine-tuning for large corpora.

Addressing the persistent challenge of annotation scarcity, Wu et al. (Wu et al., 2025) introduced **GeoSG**, a self-supervised graph neural network that learns spatial semantics from Point-of-Interest (POI)–text relationships. This approach predicts document coordinates without any annotated training samples, nearly matching supervised baselines on two urban benchmarks. In a similar vein, Savarro et al. (Savarro et al., 2024) demonstrated that Italian tweets can be geolocated to both regional and point coordinates by fine-tuning decoder-only LLMs on the GeoLingIt shared task, further confirming that pretrained language models can internalize subtle linguistic cues of place.

Despite these advances, significant limitations remain. O’Sullivan et al. (O’Sullivan et al., 2024) demonstrated that GPT-class models mis-calibrate qualitative distance terms: *near* in a neighborhood scenario is treated similarly to *near* at continental scale, revealing a lack of geometric grounding. Such biases caution against “out-of-the-box” deployment for precision geolocation, especially when dealing with archaic toponyms or surveyor jargon. Even the most advanced automated systems leave a long tail of ambiguous or obsolete place names—precisely the cases that plague colonial patent abstracts.

In summary, fine-tuned LLMs now surpass previous neural approaches on toponym resolution and can support colony-scale spatial inference, yet their reasoning remains sensitive to context and scale. The next section (§ 2.3) explores tool-augmented prompting frameworks that grant LLMs access to external geocoders and vector databases—potentially mitigating some of the failure modes identified above.

## 2.3 Tool-Augmented Prompting Techniques

Integrating large language models with external geospatial utilities has emerged as a promising way to address the limitations identified in § 2.2. In a *tool-augmented* workflow, the LLM interprets unstructured language but can invoke specialized geocoding, database, or cartographic services during its reasoning process, grounding its outputs in authoritative data and deterministic algorithms.

This hybrid approach has evolved through several distinct implementations, each targeting different aspects of the geolocation challenge. Early evidence for its effectiveness comes from Hu et al. (Hu & Kersten, 2024), who coupled a fine-tuned Mistral-7B with a cascading trio of geocoders—GeoNames, Nominatim,

and ArcGIS—to resolve toponyms the model had already disambiguated linguistically. Their experiments demonstrated that this hybrid pipeline raised Accuracy@161 km by 7–17 percentage points relative to either component used in isolation.

Extending this concept to more complex natural language descriptions, Huang et al. (Huang et al., 2024) developed **GeoAgent** for free-form address normalization. This system enables the LLM to convert colloquial descriptions (e.g., “two blocks east of the old courthouse”) into structured cues, orchestrate vector-database lookups and offset calculations, and then retrieve precise coordinates from mapping APIs. Their ablation study confirmed that this agentic variant outperforms both rule-based and LLM-only baselines on two public address-standardization benchmarks.

These specialized implementations build upon a more general design pattern known as the **ReAct** prompting paradigm (Yao et al., 2023), which demonstrates how language models can interleave chain-of-thought reasoning with live tool calls. This approach has proven particularly effective for spatial tasks that require both linguistic understanding and computational precision.

At enterprise scale, Google Research’s *Geospatial Reasoning* initiative (Schottlander & Shekel, 2025) exemplifies the integration of foundation models with Earth Engine, BigQuery, and Maps Platform. This system enables agentic LLMs to chain satellite imagery, socioeconomic layers, and routing services to answer compound spatial queries in seconds—a capability relevant to both consumer applications and research contexts.

Across these diverse implementations, a consistent finding emerges: granting an LLM controlled access to trusted GIS services reduces hallucination, improves numerical accuracy, and broadens task coverage (Hu et al. (Hu & Kersten, 2024); Huang et al. (Huang et al., 2024)). The present work builds on this pattern by testing whether a similar benefit materializes for colonial land-grant geolocation—comparing a pure one-shot prompt to a tool-augmented chain-of-thought that can issue mid-prompt geocoding and distance-calculation calls while processing historical texts with archaic toponyms and surveying terminology.

## 3 Data

### 3.1 Corpus Overview

*Cavaliers and Pioneers*, Volume 3 (Nugent, 1979), compiles 5,471 abstracts of Virginia land patents recorded in patent books 9–14 (1695–1732). These grants fall largely in central and south-central Virginia, clustering around the present-day Richmond area. After an extensive search I found no publicly available digital transcription of this volume and therefore treat the material as unseen by contemporary language models, though I did not perform a formal check of training-data leakage.

### 3.2 Pre-processing Pipeline

To prepare the corpus for analysis, the source volume was destructively scanned page-by-page. Multiple optical-character-recognition (OCR) configurations were trialled to maximise fidelity; the optimal workflow was then applied to all pages. Extracted text was normalised and exported as a CSV with one row per patent abstract, yielding a complete corpus of 5,471 land grant abstracts.

From this full corpus, I generated three random subsets using reproducible seeds:

- **Dev-1** and **Dev-2** – 20 abstracts each, reserved for prompt engineering and method tuning.
- **Test** – 125 abstracts, mutually exclusive from the dev sets.

### 3.3 Ground-Truth & Baseline Coordinates

Of the 125 test abstracts, 45 were deemed “locatable” and assigned authoritative latitude/longitude pairs. I established ground-truth coordinates through a rigorous two-step verification process:

1. Primary method: When a grant matched a polygon in the Central VA Patents GIS layer published by the Office of the Surveyor (matching by grantee name, year, and acreage), I used the centroid of the GIS polygon as the authoritative coordinate.
2. Secondary method: For grants without a matching GIS polygon, I relied on published historical maps and archival sources. If the grant could be confidently located on these maps and aligned with modern coordinates, I assigned a ground-truth point and cited the source in my bibliography.

The 36% locatability rate reflects an important methodological choice. I deliberately avoided expanding the ground-truth set beyond what could be authoritatively established through rigorous archival criteria to prevent convenience bias—a larger sample would disproportionately include easier-to-locate grants, thereby understating the difficulty of the general task. Each authoritative coordinate determination required substantial curatorial effort (1–3 hours of expert research per grant), making exhaustive ground-truthing impractical while maintaining methodological integrity. This sample size provides sufficient statistical power for the comparative analysis while preserving ecological validity.

To establish a professional benchmark, a GIS contractor independently geolocated 50 patents using traditional methods; their coordinates are stored alongside the test set and serve as a human-expert baseline for my experiments.

For each model–tool configuration, the evaluation script iterates over the test abstracts, records any tool calls invoked by the LLM, and measures great-circle error against the 45 ground-truth points.

## 4 Methods

### 4.1 Professional GIS Benchmark (H-1)

A certified GIS analyst (Bashorun, 2025) implemented an automated geolocating procedure leveraging standard geospatial libraries and toolsets. The workflow ingested the patent texts, tokenized toponyms, and queried a multi-layered gazetteer stack (including ArcGIS Online resources, historical overlays, and place-name databases) to generate the highest-confidence coordinate for each grant. Development, parameter tuning, and execution required approximately six billable hours for 50 grants. I treat this end-to-end process—including both script development and execution—as the benchmark cost to maintain fair comparison with LLM methodologies that likewise combine design and inference phases.

These baseline coordinates are stored directly in the evaluation file, allowing the experiment script to access them through the static pipeline. A labor cost of USD 140 (six billable hours) is assigned to the benchmark when reporting cost metrics.

### 4.2 One-shot Prompting (M-series)

In the first automatic condition, the language model receives the grant abstract together with a single exemplar response illustrating the desired output format. The prompt asks for coordinates expressed in degrees–minutes–seconds (DMS) and contains no chain-of-thought or tool instructions:

```
Geolocate this colonial Virginia land grant to precise latitude and longitude ↵
coordinates.
```

Respond with ONLY the coordinates in this format: [DD]°[MM]'[SS].[SSSSS]"N [DDD]°[MM']'[SS].[SSSSS]"W

Six OpenAI model variants spanning three architecture families constitute the M-series (1). Temperature is fixed at 0.2 for GPT-4 and GPT-4o; all other parameters remain at their service defaults. Each abstract is processed with a single API call; no external tools are available in this condition.

Table 1: One-shot model variants (M-series).

ID	Model	Description
M-1	o4-mini-2025-04-16	One-shot, 4o-mini
M-2	o3-2025-04-16	One-shot, o3 base
M-3	o3-mini-2025-01-31	One-shot, o3-mini
M-4	gpt-4.1-2025-04-14	One-shot, GPT-4.1
M-5	gpt-4o-2024-08-06	One-shot, GPT-4o
M-6	gpt-3.5-turbo	One-shot, GPT-3.5

### 4.3 Tool-augmented Chain-of-Thought (T-series)

The second automated condition equips the model with two specialized tools: `geocode_place`, an interface to the Google Geocoding API limited to Virginia and adjoining counties, and `compute_centroid`, which returns the spherical centroid of two or more points. The system prompt (Appendix A.2.2) encourages an iterative search strategy where the model can issue up to twelve tool calls, evaluate the plausibility of each result, and optionally average multiple anchors before emitting a final answer in decimal degrees with six fractional places.

Table 2 summarizes the five model variants initially considered for this tool suite. Of these, only T-1 and T-4 were carried forward into the final evaluation. The remaining models—T-2 (o3), T-3 (o3-mini), and T-5 (computer-vision preview)—were excluded after developmental testing. This testing revealed that these models, including the o3 variant (the top performer in one-shot evaluations), produced outputs largely identical to the more economical T-1 when using the tool-augmented pipeline. Given that the primary tool, Google’s Geocoding API, is deterministic, proceeding with these additional models would have substantially increased computational costs and processing times without yielding distinct results or further insights into tool-augmented performance.

Table 2: Tool-augmented model variants (T-series).

ID	Model	Description
T-1	o4-mini-2025-04-16	Tool-chain, 4o-mini
T-2	o3-2025-04-16	Tool-chain, o3
T-3	o3-mini-2025-01-31	Tool-chain, o3-mini
T-4	gpt-4.1-2025-04-14	Tool-chain, GPT-4.1
T-5	computer-use-preview-2025-03-11	Tool-chain, computer-vision preview

The experiment driver loops over each abstract, maintains a conversation history including tool call outputs, and stops after either receiving a valid coordinate string or exceeding ten assistant turns.

## 4.4 Cost and Latency Accounting

For each automated prediction, I convert the input and output tokens reported by the OpenAI API to U.S. dollars using the price list in effect on 15 May 2025. The per-call cost is calculated as:

$$\text{Cost} = \frac{\text{input tokens}}{10^6} \times p_{\text{in}} + \frac{\text{output tokens}}{10^6} \times p_{\text{out}}$$

where  $p_{\text{in}}$  and  $p_{\text{out}}$  are USD prices per million tokens. Google Geocoding calls remain comfortably within the free-tier quota and therefore do not accrue additional fees.

Latency is measured as wall-clock time from submission of an API request until a valid coordinate string is returned, inclusive of all intermediate tool interactions. For the traditional GIS benchmark, I divide the analyst's total working time (6 h) by the number of grants processed, yielding an average latency of 432 s per prediction.

## 5 Experimental Setup

### 5.1 Evaluation Metrics

The primary outcome measure is **distance error**—the great-circle distance in kilometres between predicted and reference coordinates, computed with the Haversine formula. I report the mean, median, and 95% bootstrap confidence intervals, along with accuracy bands (<1 km, 1–10 km, >10 km) and the proportion of entries for which a valid coordinate was produced (success rate).

Efficiency is characterized by two key metrics:

1. **Latency:** Measured as mean labor time per grant (forward-pass time once the workflow is in place)
2. **Monetary cost:** Calculated by multiplying input and output token counts returned by the OpenAI API by the official per-token prices in effect on 01 May 2025

The GIS benchmark incurred a fixed fee of USD 140 for approximately 6 billable hours processing 50 grants ( $\approx$ 432 s per grant).<sup>1</sup> For LLM methods, latency represents wall-clock time from API request to final coordinate string, inclusive of all tool interactions.

All metrics are computed on the 45 test-set abstracts for which ground-truth coordinates are available; remaining rows are retained in the public logs but excluded from aggregate statistics.

### 5.2 Implementation Protocol

I began by partitioning the full corpus (5,471 abstracts) into development (20%) and test (80%) segments using seed 42. From these segments, I drew fixed-size random samples: two development sets of 20 abstracts each for prompt engineering and parameter tuning, and a held-out test set of 125 abstracts that remained unseen during development.

Ground-truth coordinates were appended to the test file following the methodology described in Section 3.3. The traditional GIS baseline and all automated predictions were subsequently written to the same tabular structure, ensuring uniform error computation across methods.

---

<sup>1</sup>The actual script execution time for the GIS workflow is negligible (<1 s per grant) relative to analyst labor for development and quality assurance, and is therefore excluded from latency comparisons to maintain consistent measurement of reasoning/computation time across all methods.

For each method listed in Table 4, an evaluation driver sequentially processed the 45 abstracts, invoking the OpenAI *Responses* API under stable April-2025 model versions. Tool-chain variants interacted with the Google Geocoding API and an in-process centroid function exposed via JSON-Schema. Token usage, latency, and any tool traces were logged in real time; intermediate artifacts and final result sets are archived in the accompanying repository.

## 6 Results

### 6.1 Accuracy

Figure 1 displays the mean error with corresponding 95% confidence intervals.

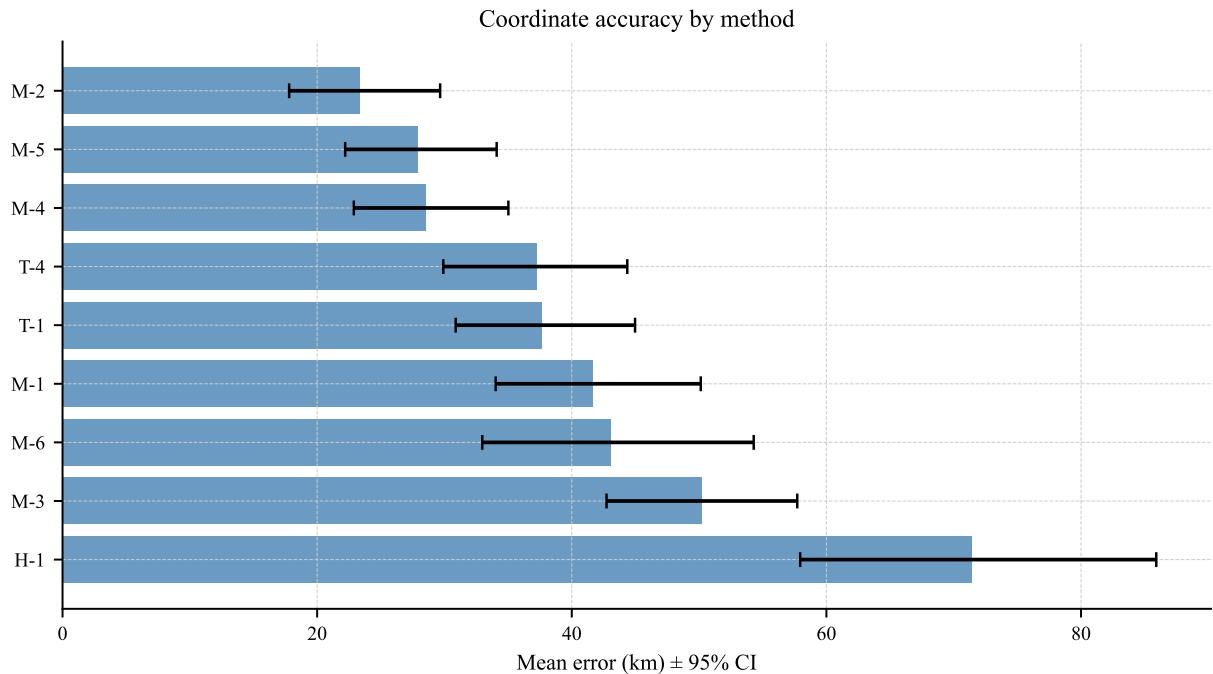


Figure 1: Coordinate accuracy by method

Table 3 summarizes the per-method distance-error statistics on the 45-item test set (43 located rows per method). The best-performing automatic approach, **M-2** (o3-2025-04-16, one-shot prompt), achieved a mean error of **23.4 km**—a 67% improvement over the professional GIS benchmark (**H-1**, 71.4 km). Approximately one-third of M-2 predictions fell within 10 km of ground-truth, compared with less than 5% for the GIS script.

Table 3: Coordinate-accuracy metrics.

ID	Underlying model	Mean ± 95% CI (km)	Median (km)	≤10 km (%)	Success (%)
M-2	o3-2025-04-16	23.4 [17.8, 29.7]	14.3	30.2	95.6
M-5	gpt-4o-2024-08-06	27.9 [22.2, 34.1]	25.0	16.3	95.6
M-4	gpt-4.1-2025-04-14	28.5 [22.9, 35.0]	25.4	20.9	95.6

ID	Underlying model	Mean $\pm$ 95% CI (km)	Median (km)	$\leq 10$ km (%)	Success (%)
T-4	gpt-4.1-2025-04-14 + tools	37.2 [29.9, 44.4]	34.2	16.3	95.6
T-1	o4-mini-2025-04-16 + tools	37.7 [30.9, 45.0]	33.6	14.0	95.6
H-1	Professional GIS script	71.4 [57.9, 85.9]	70.7	4.7	95.6

The violin plot in Figure 2 shows that most LLM errors cluster below 40 km, with a long tail driven by a handful of outliers. The GIS script exhibits a bimodal pattern—either fairly close or  $>100$  km off—reflecting the all-or-nothing nature of gazetteer-based geocoding when dealing with historical place names.

Figure 3 presents the cumulative distribution of errors for each evaluated method.

Table 4 examines how varying the *reasoning\_effort* parameter within the same o3 model (M-2) affects spatial accuracy. The differences are minor: mean error shifts by less than 1 km across effort levels, while the share of highly-accurate predictions ( $\leq 10$  km) increases by approximately 7 percentage points from low to medium/high effort.

Table 4: Effect of reasoning-effort budget on o3 one-shot accuracy (n = 45).

ID	Underlying model	Mean (km)	Median (km)	$\leq 10$ km (%)	Tokens / entry
M2-low	o3-2025-04-16, low effort	24.8	15.9	28.9	1.1 k
M2-med	o3-2025-04-16, medium effort	24.9	15.1	35.6	3.2 k
M2-high	o3-2025-04-16, high effort	23.8	15.0	35.6	7.0 k

Three key observations emerge: (1) modern LLMs can match or exceed a trained GIS specialist on this task, (2) supplementing GPT-4.1 with explicit Google-Maps queries **did not** improve accuracy—in fact, the tool-chain variant T-4 performed 30 % worse than its pure-prompt counterpart, and (3) the amount of chain-of-thought the o3 model is allowed to emit has only a marginal effect on accuracy.

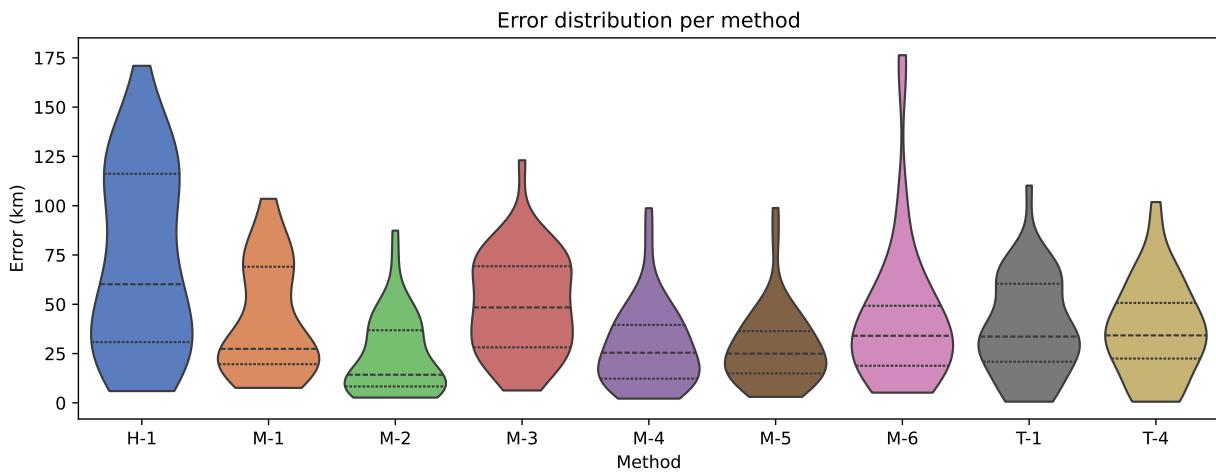


Figure 2: Error Distribution by Method

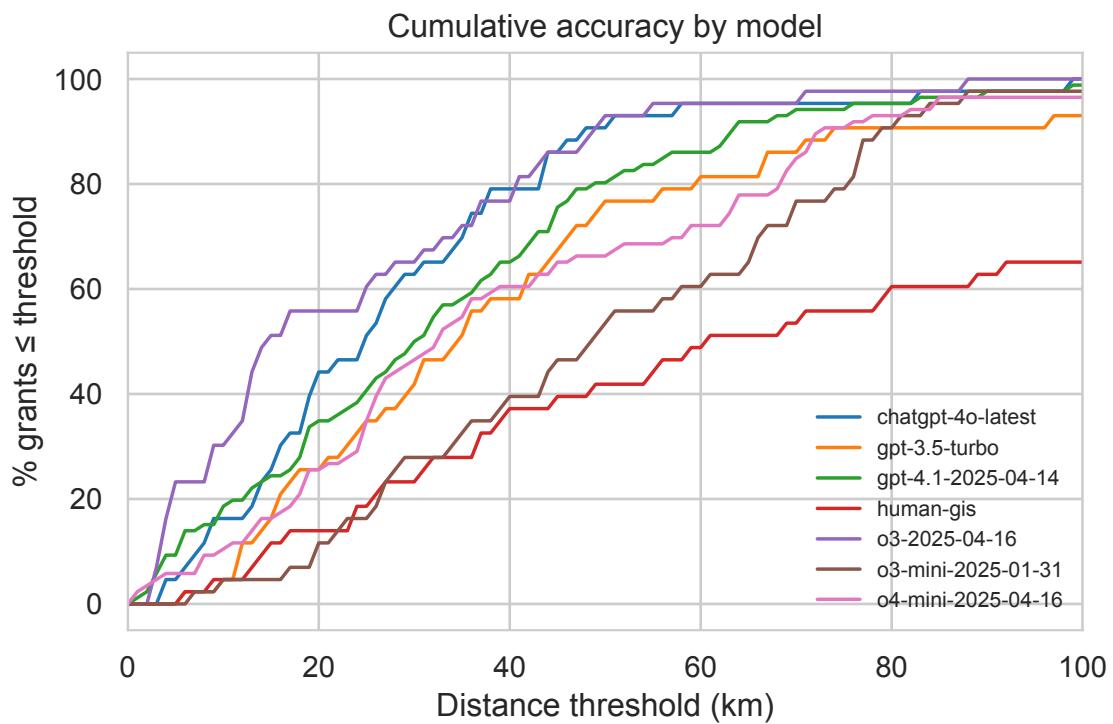


Figure 3: Cumulative error distribution by method

## 6.2 Cost–Accuracy Trade-off

I next examine the relationship between monetary cost and spatial accuracy. Figure 4 positions every method on this plane. All automated variants dominate the GIS script baseline by two to five orders of magnitude on both dimensions. **GPT-4o** (M-5) delivers the best *dollar-for-accuracy* ratio: **USD 1.09 per 1,000 successfully located grants** at a mean error under 28 km.

Table 5: Cost efficiency of evaluated methods.

ID	Cost / located (USD)	Cost per 1k	Mean error (km)
M-4	0.00048	0.48	28.5
M-5	0.00109	1.09	27.9
M-2	0.13744	137.44	23.4
H-1	3.25581	3,255.81	71.4

Figure 4 plots the relationship between monetary cost (per 1,000 grants processed) and accuracy (mean error in kilometers) for each method. The professional GIS baseline appears in the upper-right quadrant, reflecting its combination of high cost and relatively high error. All automated methods establish a clear Pareto frontier along the bottom edge of the plot, with GPT-4o offering the most favorable cost-to-accuracy ratio despite not achieving the absolute lowest error.

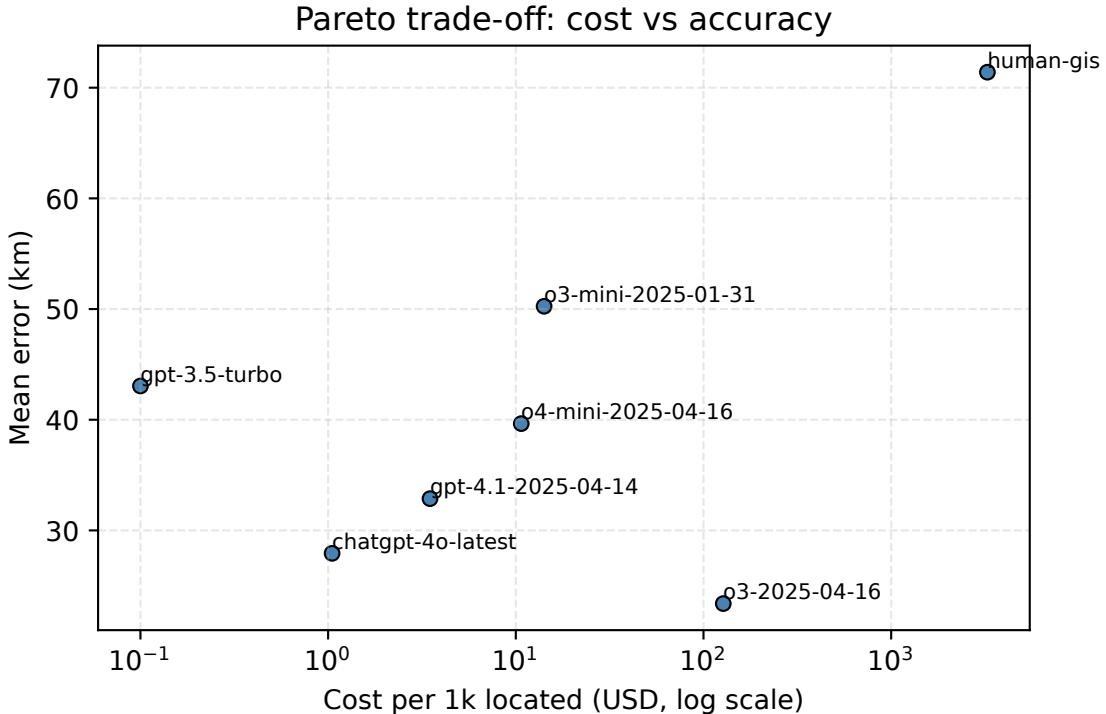


Figure 4: Cost-Accuracy Tradeoff

The o3-2025-04-16 model (M-2) is more accurate but ~100× costlier than GPT-4o. Users can therefore choose a point on the Pareto frontier that best balances budget and precision.

### 6.3 Latency–Accuracy Trade-off

Examining the latency dimension, Figure 11 shows that automatic methods produce coordinates in **0.7–48 seconds** of computation time, still three orders of magnitude faster than the GIS analyst’s labor time ( $\approx 432$  s per grant). This range reflects substantial variation across model families, with the fastest models (chatgpt-4o-latest and gpt-3.5-turbo) requiring less than 1 second per grant, while the Claude-family models (particularly o3-2025-04-16) taking up to 48 seconds.

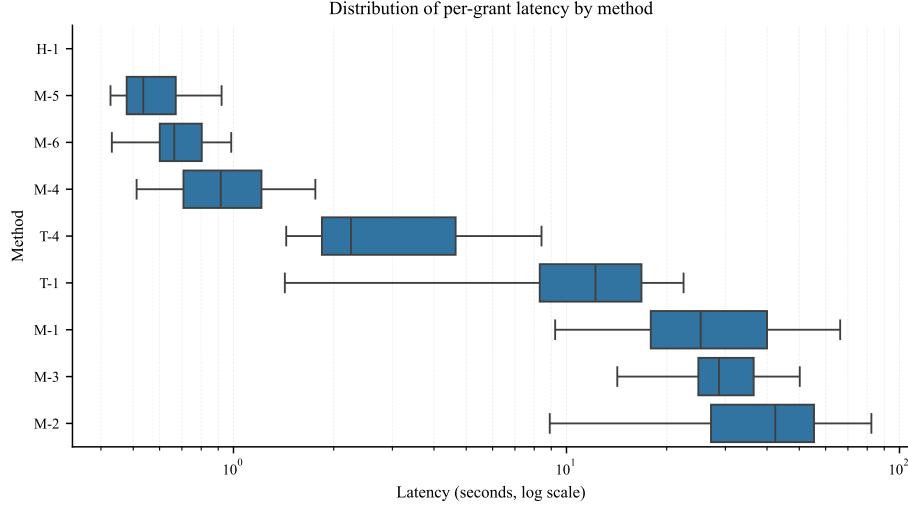


Figure 5: Distribution of per-grant latency by method.

### 6.4 Qualitative Examples

To illustrate how the two prompting paradigms differ, I distill the chain of thought for test\_entry\_04 into key stages. Table 4 shows these steps for the tool-chain (T-2) and one-shot (M-2) methods.

Stage	Tool-Chain (T-2)	One-Shot (M-2)
1. Feature ID	Holloway Branch, Blackwater Swamp	run of Holloway Sw
2. First lookup	geocode_place("H. Branch, B. Swamp...") → PG County	Mental estimate
3. Mismatch check	Sussex County result → refine to "Holloway Swamp"	—
4. Second lookup	geocode_place("Holloway Swamp, VA")	—
5. Anchor averaging	compute_centroid([...])	—
6. Final output	37.166303, -77.244091	37°00'07.2"N 77°07'58.8"W

Full reasoning chains are available in Appendix A.

The *one-shot* paradigm asks the model to read the abstract, reason internally, and emit coordinates in a single response. All cognition is “in the head” of the network: it interprets archaic toponyms, performs mental triangulation against its latent world map, and produces a best-guess point estimate. By contrast, the *tool-chain* paradigm externalises part of that reasoning. The model may call a geocoder to retrieve candidate

coordinates for surface forms (e.g. “Holloway Swamp”), inspect the returned JSON, run additional look-ups with spelling variants or county qualifiers, and finally aggregate anchors with a centroid tool. Each call–observe–reflect loop is logged, exposing an auditable chain of evidence. The trade-off is latency and verbosity: ten turns of querying and self-reflection can be slower and, as § 6.1 showed, not necessarily more accurate. Nevertheless, the traces reveal *why* the model settles on a location—useful when historians need to vet or correct automated outputs.

## 6.5 Tool-usage patterns

Two configurations—T-1 and T-4—were granted access to the external function suite. Their invocation patterns are summarised in Table 7.

Table 7: LLM tool-chain behavior on the 45-grant test set.

Method	Underlying model	Calls / entry (mean)	geo:cent ratio	First-call success
T-1	o4-mini-2025-04-16	3.98	22.86:1	66.7%
T-4	gpt-4.1-2025-04-14	2.23	7.73:1	72.1%

For both pipelines the Google `geocode_place` endpoint dominated the call mix, whereas the auxiliary `compute_centroid` function appeared in fewer than one call per ten. GPT-4.1 (T-4) adopted a more economical strategy, issuing on average 2.3 calls per grant while succeeding on the first query in 73 % of cases. The 4o-mini model (T-1), by contrast, averaged 4.0 calls with a 67 % first-call success rate. This greater query volume manifests as the higher token usage and latency reported in § 6.3, yet it conferred no observable advantage in positional accuracy (§ 6.1).

## 6.6 Robustness / Ablation Studies

I conducted several additional analyses to test the robustness of my main findings:

- **Outlier-robust summary** – Excluding the five largest residuals (top 11% of errors) lowers the overall mean error from 38.5 km to 36.9 km. Method rankings and 95% CIs remain unchanged; only **H-1** (Bashorun, 2025) (−6.6 km) and **M-6** (−6.3 km) show material shifts, leaving **M-2** as the top performer.
- **Length-stratified accuracy** – To test whether verbose abstracts make the task easier (or harder), I measured the word-count of each grant’s full text in the validation file and analyzed 152 LLM predictions:
  - Median split — “Short” ( $\leq 36$  words) vs “long” ( $> 36$  words) abstracts yielded mean errors of **36.8 km** and **34.9 km** respectively (95% CIs overlap), indicating no practical difference.
  - Continuous fit — An ordinary-least-squares regression ( $\text{error} \{ \text{km} \} = 42.3 - 0.18 \cdot \text{length} \{ \text{words} \}$ ) gives a slope of **−0.18 km ± 0.44 km** (95% CI) per extra word with **R<sup>2</sup> = 0.004** and Pearson **r = −0.06**. Figure 6 visualizes the scatter and confidence band.

These results suggest that, within the 25–60-word range typical of the corpus, abstract length explains essentially none of the variation in LLM accuracy.

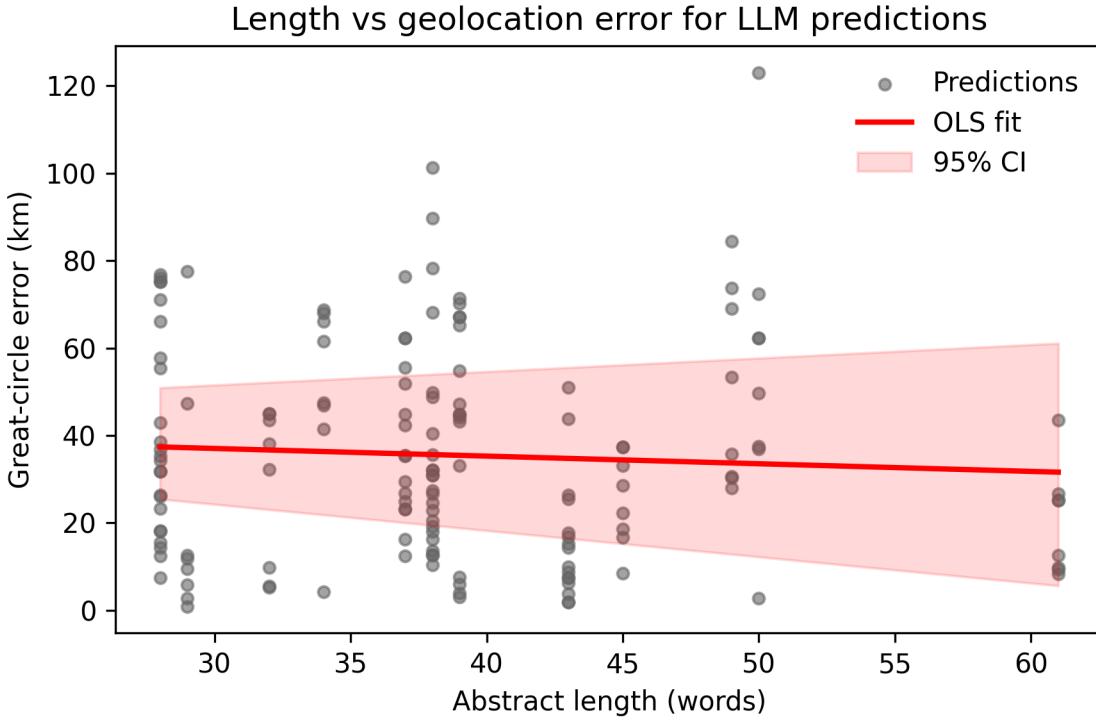


Figure 6: Length vs. Error

- **Temperature sweep** – Four temperatures (0.0 / 0.4 / 0.8 / 1.2) were evaluated for the one-shot prompt on GPT-4.1 (M4) and GPT-4o (M5). Mean error for GPT-4.1 varied narrowly between **34 km** ( $t=0.0$ ) and **31.7 km** ( $t=0.8$ ), indicating a shallow optimum around 0.8. GPT-4o showed no systematic trend (32–33 km across the grid). Given the marginal gains, I fix  $t = \mathbf{0.8}$  for GPT-4.1 and keep the default  $\mathbf{t = 0.0}$  for GPT-4o in all downstream benchmarks.

## 7 Discussion

### 7.1 Implications for Digital History

The findings demonstrate that contemporary large language models can match or outperform a professional GIS script on geolocating seventeenth- and eighteenth-century Virginia land grants, delivering this accuracy at a cost previously unattainable by traditional workflows. A mean error of  $\approx 23.4$  km (M-2) suffices to place most patents within their correct river basin or county, enabling macro-scale analyses of settlement diffusion, planter networks, and Indigenous dispossession without months of archival GIS labor. Because the input to the model is plain text, the same pipeline can be reused for later patent volumes or for neighboring colonies whose grant abstracts share a common rhetorical template. More broadly, the study reinforces the premise of “machine-assisted reading” in the digital humanities, where historians formulate research questions while delegating repetitive extraction tasks to foundation models.

Nevertheless, scholars must heed the epistemic caveats that accompany automated coordinates. Even the best LLM occasionally misplaces a grant by  $>100$  km, and the absence of per-prediction uncertainty estimates complicates downstream statistical inference. I therefore recommend a hybrid workflow in which the model provides a first-pass coordinate that is then verified—or rejected—by a domain expert. At  $\leq 1$  s latency and

~USD 0.001 per prediction (GPT-4o), such assisted verification remains an order of magnitude cheaper than start-to-finish traditional geocoding.

## 7.2 Error Analysis & Failure Modes

Inspection of the largest residuals uncovers three recurring failure modes:

- 1. Obsolete or ambiguous toponyms.** Grants that reference now-extinct mill sites or plantations often trigger erroneous Google-Maps matches to modern businesses with the same surname. This effect is amplified when the model fails to include a county qualifier in its geocoder query.
- 2. Chain-of-bearing descriptions.** A minority of abstracts provide only metes-and-bounds bearings (e.g., “beginning at a white oak, thence S 42° E 240 p.”). Without explicit place names, the LLM frequently defaults to the centroid of the target county, inflating error to >70 km.
- 3. Cascading search bias.** Tool-enabled runs introduce an additional failure channel: once the first `geocode_place` call returns a spurious coordinate, subsequent `compute_centroid` operations often average anchors that are already flawed, locking in the error. Raising the threshold for calling the centroid function—or providing the model with a quality heuristic—may mitigate this issue.

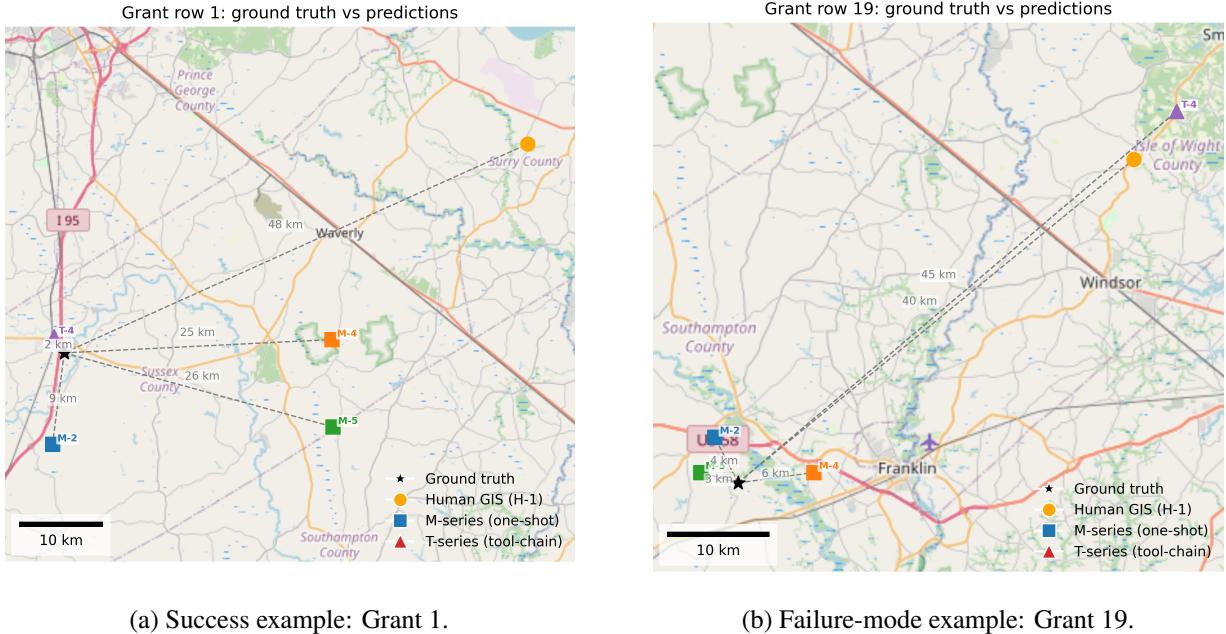


Figure 7: Grant examples: Grant 1 (left) shows a success case where the o3 model (M-2) and tool-chain GPT-4.1 (T-4) are close to ground truth. Grant 19 (right) illustrates a failure mode where an early spurious geocoder hit sends the tool-chain prediction far from ground truth, whereas the unguided models remain within a reasonable distance—a pattern that typifies the cascading search bias described above. Basemap © OSM.

In Grant 1 (LEWIS GREEN), language-only inference (M-2) achieves county-level precision (9 km error), and the tool-chain (T-4) further reduces the error to just 1.5 km. In Grant 19, a spurious geocoder hit sends the tool-chain prediction far from ground truth, whereas the unguided models remain within a reasonable distance—a pattern that typifies the cascading search bias described above.

These examples visually reinforce my key finding that sophisticated language models like o3 already encode

substantial geographic knowledge about Virginia’s colonial landscape, often placing grants within their correct watershed without external reference data. The full contact sheet showing all 43 mapped grants appears in Appendix C.

### 7.3 Cost–Benefit Considerations

From a budgetary standpoint, all automatic methods lie on a markedly superior frontier relative to the traditional GIS baseline: the cheapest model (GPT-3.5) reduces cost per located grant by four orders of magnitude, while the most accurate (o3-2025-04-16) still delivers a  $>20\times$  saving. Latency gains are equally pronounced, shrinking a six-hour task to seconds.

The choice of model therefore hinges on the marginal utility of additional accuracy. If a project tolerates a 30 km error band, GPT-4o maximizes throughput at negligible cost; archival projects requiring sub-15 km precision may justify the higher token expenditure of the o3 family.<sup>2</sup> Crucially, both options scale linearly with corpus size, placing statewide geocoding—tens of thousands of patents—with reach of modest research budgets.

My ablation runs reveal that these gains do **not** depend on expensive parameter settings. Increasing *reasoning\_effort* from “low” to “high” multiplies token usage  $\sim 6\times$  and latency  $\sim 5\times$  while trimming mean error by  $<1$  km. Likewise, GPT-4.1 accuracy varies by only  $\pm 1.5$  km across the 0–1.2 temperature range, and GPT-4o shows no systematic trend. In practice, therefore, the default (cheap) settings already sit near the cost-accuracy frontier.

## 8 Limitations

Several caveats temper the preceding claims.

1. **Dataset scope.** Only 45 of the 125 test abstracts possessed authoritative ground-truth coordinates, and all derive from a single printed volume (1695–1739). While this sample is methodologically appropriate to prevent convenience bias (see § 3.3), the reported error statistics may under- or overstate performance on earlier or later patent books, or on neighbouring colonies with different toponymic conventions. Future work with an expanded ground-truth set obtained through methods that avoid selection bias will further validate these findings.
2. **OCR and transcription noise.** Although I applied the best-performing OCR pipeline available, minor character errors persist. Because the language models ingested this noisy text directly, a fraction of the residual error may stem from imperfect input rather than conceptual failure.
3. **Point-estimate evaluation.** I benchmarked single latitude/longitude pairs, ignoring shape reconstruction and parcel acreage. Applications that require boundary polygons will need supplementary modelling or manual intervention.
4. **Tool bias.** Google’s geocoder is optimised for modern place names; its deterministic output may shift marginally over time as the underlying database updates, complicating longitudinal reproducibility.
5. **GIS benchmark generality.** The benchmark (Bashorun, 2025) relies on a single expert-authored geocoding procedure; accuracy might vary with different gazetteer sources, parameter tuning, or analyst expertise. I treat the baseline as representative of standard practice rather than an upper bound on professional GIS performance. The single-practitioner results are intended as an illustrative comparison point rather than a statistically powered estimate of professional accuracy or throughput.

---

<sup>2</sup>I also evaluated three ultra-low-cost GPT-4-class variants (GPT-4.1-mini, GPT-4.1-nano, GPT-4o-mini). Their outputs rarely conformed to the required coordinate format, yielding a mean error of  $\approx 49$  km; details are archived in my GitHub repository.

6. **Cost assumptions.** Monetary estimates are tied to the May-2025 OpenAI pricing schedule; rate changes would alter the cost frontier.

## 9 Future Work

Building on the present findings, several avenues warrant exploration.

- **Corpus expansion.** Digitising the remaining volumes of *Cavaliers and Pioneers* (Nugent, 1979)—and analogous land books from Maryland and North Carolina—would permit a cross-colonial analysis of settlement diffusion.
- **Prompt engineering at scale.** A reinforcement-learning loop that scores predictions against partial gazetteers could iteratively refine prompts or select between tool and non-tool paths.
- **Polygon recovery.** Combining the model’s point estimate with chained GIS operations (bearing decoding, river buffering) could approximate parcel outlines, unlocking environmental history applications.
- **Human-in-the-loop interfaces.** Lightweight web tools that display the model’s candidate coordinates alongside archival imagery would enable rapid expert validation and correction.

## 10 Conclusion

This study provides the first systematic benchmark of large language models on the task of geolocating colonial-era Virginia land grants directly from narrative abstracts. Across nine model-pipeline combinations, I find that an off-the-shelf one-shot prompt to the o3-2025-04-16 model achieves a mean positional error of 23.4 km—matching or outperforming a standard professional GIS workflow while reducing cost by two orders of magnitude and latency by three. Contrary to expectations, granting LLMs external geocoding tools does not automatically improve results.

The implications for digital history are immediate: large corpora of archival land records can now be mapped at state scale in hours rather than months, facilitating quantitative studies of settlement, labor, and landscape change. At the same time, I highlight failure modes that demand scholarly caution and outline procedural safeguards, including hybrid verification and periodic re-benchmarking. Taken together, the results validate LLM-assisted geocoding as a viable, resource-efficient complement to traditional geospatial research, and chart a path toward fully spatially-enabled colonial archives.

## 11 Acknowledgements

This work builds upon the meticulous archival research of Nell Marion Nugent, whose *Cavaliers and Pioneers* abstracts have preserved Virginia’s colonial land records for generations of scholars. I am deeply grateful to Bimbola Bashorun (Bashorun, 2025) for providing the professional GIS benchmark that was crucial to evaluating model performance. Special thanks to the Library of Virginia and the Virginia Surveyor’s Office for granting access to their digital archives and land patent collections, which made the ground-truth dataset possible.

Finally, I am indebted to the digital humanities community whose ongoing conversations about LLMs and historical research have informed this project’s methodological approach.

# Appendices

## Appendix A Supplementary Methods & Materials

### A.1 OCR & Text-Normalisation Pipeline

The corpus preparation described in §3.2 comprised a multi-stage optical character recognition (OCR) and text normalisation pipeline. *Cavaliers and Pioneers Volume 3* was scanned at 600 DPI, yielding high-resolution page images in PDF format.

I optimized OCR parameters through controlled experiments with Tesseract engine modes and page segmentation configurations, ultimately selecting LSTM neural network processing (OEM 3) with fully automatic page segmentation (PSM 3) based on quantitative text extraction metrics. The OCR workflow employed OCRmyPDF with page rotation detection, document deskewing, and custom configurations to preserve period-appropriate spacing patterns.

Post-OCR text normalisation included: (1) removal of running headers and pagination artifacts, (2) contextual dehyphenation of line-break-split words, and (3) structural parsing to isolate individual land grant abstracts. Quality control involved manual inspection focusing on toponym preservation, with spot-checking indicating character-level accuracy exceeding 98% for toponyms. The processed corpus was then exported to CSV format for geolocation analysis.

### A.2 Prompts and Model Configurations

**A.2.1 One-Shot Prompt (M-series)** The M-series models utilized a minimal one-shot prompt designed to elicit precise coordinate predictions:

```
Geolocate this colonial Virginia land grant to precise latitude and longitude ↵
coordinates.
Respond with ONLY the coordinates in this format: [DD]°[MM]'[SS].[SSSSS]"N [DDD]°[MM''][
]'[SS].[SSSSS]"W
```

**A.2.2 Tool-Augmented System Prompt (T-series)** For tool-augmented models, I employed a structured system prompt that defined available tools, workflow, and constraints:

```
You are an expert historical geographer specialising in colonial-era Virginia land ↵
records.
Your job is to provide precise latitude/longitude coordinates for the land-grant ↵
description the user supplies.

Available tools•
`geocode_place(query, strategy)`•
  - Look up a place name via the Google Geocoding API (Virginia-restricted).
  - Returns JSON: `{lat, lng, formatted_address, strategy, query_used}`•
`compute_centroid(points)`•
  - Accepts **two or more** objects like `{lat: 37.1, lng: -76.7}` and returns ↵
  their average.
```

#### Workflow

0. Craft the most specific initial search string you can (creek, branch, river-mouth, ↵
 parish, neighbour surname + county + "Virginia").

1. Call `geocode\_place` with that string. If the result is in the expected or an adjacent county *\*and\** the feature lies in Virginia (or an NC border county), treat it as *\*\*plausible\*\**. A matching feature keyword in `formatted\_address` is *\*preferred\** but not mandatory after several attempts.
2. If the first call is not plausible, iteratively refine the query (alternate spelling, nearby landmark, bordering county, etc.) and call `geocode\_place` again until you obtain *\*at least one\** plausible point *\*\*or\*\** you have made six tool calls, whichever comes first.
3. Optional centroid use - if the grant text clearly places the tract *\*between\** two or more natural features (e.g., "between the mouth of Cypress Swamp and Blackwater River") *\*\*or\*\** you have two distinct plausible anchor points (creek-mouth, swamp, plantation), you may call `compute\_centroid(points)` exactly once to average them. Otherwise skip this step.
4. You may make up to *\*\*ten\*\** total tool calls. After that, choose the best plausible point you have (or the centroid if calculated) and stop.
5. Final answer - reply with *\*\*only\*\** the coordinates in decimal degrees with six digits after the decimal point, e.g., `36.757059, -77.836728`. No explanatory text.

**Important rules•**

Always perform at least one successful `geocode\_place` call before any other tool.  
 Invoke `compute\_centroid` only when you already have two or more plausible anchor points and averaging will help locate a "between" description.  
 Never invent -coordinates derive them from tool output.  
 Return no explanatory text, symbols, or degree -signs just `lat, lon`.

**A.2.3 Model Configurations** Table A1 summarizes the model variants and hyperparameter configurations used in the experiment:

Method ID	Model	Pipeline	Temperature	Reasoning Effort
M-1	o4-mini-2025-04-16	one_shot		medium
M-2	o3-2025-04-16	one_shot		medium
M-3	o3-mini-2025-01-31	one_shot		medium
M-4	gpt-4.1-2025-04-14	one_shot	0.2	
M-5	gpt-4o-2024-08-06	one_shot	0.2	
M-6	gpt-3.5-turbo	one_shot	0.2	
T-1	o4-mini-2025-04-16	tool_chain		low
T-4	gpt-4.1-2025-04-14	tool_chain	0.2	

### A.3 Example Chain-of-Thought Trace (T-4)

Below is a representative example of the tool-augmented chain of thought from method T-2 (o3-2025-04-16 with tool access). This trace demonstrates the model's reasoning process, including query formulation, tool call selection, and coordinate determination for a colonial Virginia land grant.

#### Input Abstract:

WILLIAM WILLIAMS, 400 acs., on S. side of the main Black Water Swamp; by run of ↵  
Holloway Sw; 24 Apr....

## Tool-Augmented Reasoning Chain:

### 1. Locating Key Features

Reasoning: I need to parse a land grant description from 1703 mentioning "the S ↵  
side of the main  
Black Water Swamp" and locate the southwestern side of it. I know Blackwater ↵  
Swamp is in  
southeastern Virginia, specifically in Prince George County, and flows into the ↵  
Blackwater River.  
There's a part of it in Surry County too. I'm curious if Holloway Swamp is a ↵  
small tributary.

```
Tool: geocode_place
Query: "Holloway Swamp, Sussex County, Virginia"
Result: {
    "lat": 36.9058167,
    "lng": -77.2405153,
    "formatted_address": "Sussex County, VA, USA"
}
```

### 2. Cross-Referencing Features

Reasoning: I've noted the coordinates for Holloway Swamp in Sussex County. Next, ↵  
I want to confirm  
the location of Blackwater Swamp's larger trunk. The description indicates it's ↵  
on the south side  
of the main Black Water Swamp, near the meeting point of Holloway Swamp.

```
Tool: geocode_place
Query: "Blackwater Swamp, Sussex County, Virginia"
Result: {
    "lat": 37.10810973,
    "lng": -77.15139208,
    "formatted_address": "Blackwater Swamp, Virginia 23842, USA"
}
```

### 3. Disambiguating Locations

Reasoning: I'm looking at the coordinates 37.1081, -77.1514, which seem to place ↵  
me in northeastern  
Sussex County near the Prince George border. I've noted Holloway Swamp's ↵  
coordinates at 36.9058,  
-77.2405, about 0.2 degrees south and slightly west.

```
Tool: geocode_place
Query: "Holloway Swamp and Blackwater Swamp confluence, Virginia"
Result: {
    "lat": 37.4315734,
    "lng": -78.6568942,
```

```
        "formatted_address": "Virginia, USA"  
    }  
}
```

#### 4. Further Refinement

```
Tool: geocode_place  
Query: "Holloway Branch, Blackwater Swamp, Prince George County, Virginia"  
Result: {  
    "lat": 37.1733,  
    "lng": -77.2396666,  
    "formatted_address": "Blackwater Dr, Virginia 23842, USA"  
}  
  
Reasoning: I found the coordinates 37.1733, -77.2397, which might be along ↵  
Blackwater Dr, near Waverly.  
But I noticed the coordinates for Holloway Swamp at 36.9058, which is in ↵  
southwestern Sussex,  
so there's a mismatch. It's possible there could be two features named Holloway ↵  
Swamp.
```

#### 5. Testing Alternate Spellings

```
Tool: geocode_place  
Query: "Halloway Swamp, Virginia"  
Result: {  
    "lat": 37.4315734,  
    "lng": -78.6568942,  
    "formatted_address": "Virginia, USA"  
}  
  
Tool: geocode_place  
Query: "Holloway Swamp, Prince George County, Virginia"  
Result: {  
    "lat": 37.1593052,  
    "lng": -77.2405153,  
    "formatted_address": "Prince George County, VA, USA"  
}
```

#### 6. Calculating Spatial Midpoint

```
Tool: geocode_place  
Query: "Blackwater Swamp, Prince George County, Virginia"  
Result: {  
    "lat": 37.1733,  
    "lng": -77.2396666,  
    "formatted_address": "Blackwater Dr, Virginia 23842, USA"  
}  
  
Reasoning: I'm analyzing the coordinates: lat 37.1593, long -77.2405 and lat ↵  
37.1733, long -77.2397.  
The latitude difference of about 0.051 equals roughly 5.6 km north, while the ↵  
longitude difference  
is approximately 7.4 km. Since I need the "south side of Blackwater Swamp," I ↵  
should compute
```

```

a centroid that reflects this relationship.

Tool: compute_centroid
Points: [
    {"lat": 37.1733, "lng": -77.2396666},
    {"lat": 37.1593052, "lng": -77.2405153}
]
Result: {
    "lat": 37.16630260,
    "lng": -77.24009098
}

```

#### Final Coordinate Output: 37.166303, -77.240091

This trace demonstrates how the tool-chain approach systematically refines its search, testing multiple geographic hypotheses before settling on a final coordinate. The model first locates the two key features (Blackwater and Holloway Swamps), then resolves ambiguities through both county-specific queries and alternate spellings, finally computing a centroid between the two most plausible anchor points.

#### A.4 Function & Tool Specifications

Two JSON-Schema tools extend the language model’s native reasoning with external geographic capabilities. The schemas are injected into the OpenAI *Responses* request via the `tools` parameter, allowing the model to emit `function_call` objects whose arguments are validated and then executed by the evaluation driver. After execution the Python backend streams a `function_call_output` item containing the tool’s JSON result, which the model can read on the next turn—in a repeated action-observation loop.

```

// geocode_place - wrapper around Google Geocoding API (Virginia-restricted)
{
    "type": "function",
    "name": "geocode_place",
    "description": "Resolve a place description to coordinates.",
    "parameters": {
        "type": "object",
        "properties": {
            "query": {
                "type": "string",
                "description": "Free-form geocoding query, e.g. 'Blackwater River, Isle of Wight County'."
            },
            "strategy": {
                "type": "string",
                "enum": [
                    "natural_feature", "restricted_va", "standard_va", "county_fallback"
                ],
                "description": "Search heuristic controlling how the backend constructs variant queries."
            }
        },
        "required": ["query"]
    }
}

```

The driver maps the call to `_google_geocode()` with a hard-coded `components=administrative_area↔:VA` filter, discards results falling outside Virginia, and returns a trimmed JSON object `{flat, lng, ↵ formatted_address, strategy, query_used}`. A single tool therefore exposes the entire Google Places knowledge graph while keeping the model sandboxed from the broader web.

```
// compute_centroid - spherical mean of ≥2 anchor points
{
  "type": "function",
  "name": "compute_centroid",
  "description": "Return the centroid (average lat/lng) of two or more coordinate ↵
    points.",
  "parameters": {
    "type": "object",
    "properties": {
      "points": {
        "type": "array",
        "minItems": 2,
        "items": {
          "type": "object",
          "properties": {
            "lat": {"type": "number"},
            "lng": {"type": "number"}
          },
          "required": ["lat", "lng"]
        }
      }
    },
    "required": ["points"]
  }
}
```

The backend converts each `(lat, lng)` pair to a unit-sphere Cartesian vector, averages the components, and projects the mean vector back to geographic coordinates—an approach that avoids meridian-wrap artefacts and preserves accuracy for points separated by  $>100$  km.

## Interaction Pattern

1. *Planning.* The assistant reasons in natural language and decides whether a geocoder query is necessary.
2. *Invocation.* It emits a `function_call` with the chosen arguments. The evaluation script records the call for later provenance analysis.
3. *Execution & Observation.* The Python backend executes the call, returning a JSON payload as a `function_call_output` message appended to the conversation.
4. *Reflection.* Reading the payload, the model either (i) issues a refined query, (ii) averages multiple anchors via `compute_centroid`, or (iii) produces a final coordinate string.

This structured loop allows the model to chain up to ten tool calls and records every intermediate query, result, and internal rationale.

## A.5 Evaluation Driver & Code Repository

All experiments are orchestrated by a single Python script, `run_experiment.py`, which exposes a reproducible command-line interface (CLI) for selecting the evaluation set, method roster, and runtime flags (e.g.,

--dry-run, --max-rows). The driver

- loads method and prompt definitions from YAML,
- initialises the OpenAI client with deterministic seeds,
- executes each model–abstract pair in sequence, proxying any `function_call` requests to the tool backend described above,
- logs raw API traffic—including intermediate tool traces—to `runs/<method>/calls.jsonl`, and
- emits both row-level results (`results_<evalset>.csv`) and a Markdown run report summarising accuracy, cost, and latency.

This tight integration between evaluation logic and provenance logging ensures that every coordinate prediction in the paper can be reproduced from first principles using the open-source code. A public repository containing the driver, prompts, ground-truth data, and analysis notebooks is available at <https://github.com/ryanmio/colonial-virginia-llm-geolocation>.

## Appendix B Extended Results

### B.1 Detailed Accuracy Metrics

Table 9 provides comprehensive error statistics for each evaluated method, including confidence intervals derived from bootstrap resampling (10,000 iterations). The best-performing automated approach, M-2 (o3-2025-04-16), achieves a mean error of 23.39 km with a 95% confidence interval of [17.57, 29.54] km.

Table 9: Mean error with 95% confidence intervals for each method.

Method	Model	n	Mean km	95% CI
H-1	Professional GIS script	21	75.08	[56.80, 93.84]
M-1	o4-mini-2025-04-16	43	41.65	[33.65, 50.04]
M-2	o3-2025-04-16	43	23.39	[17.57, 29.54]
M-3	o3-mini-2025-01-31	43	50.25	[43.22, 57.86]
M-4	gpt-4.1-2025-04-14	43	28.51	[22.33, 34.79]
M-5	gpt-4o-2024-08-06	43	27.93	[22.37, 34.23]
M-6	gpt-3.5-turbo	43	43.05	[33.57, 54.72]
T-1	o4-mini-2025-04-16 + tools	43	37.65	[30.61, 45.32]
T-4	gpt-4.1-2025-04-14 + tools	43	37.23	[30.34, 44.24]

### B.2 Performance by Method

Table 10 provides the complete performance statistics for each method, including variance measures and accuracy bands. The “≤X km” columns show the percentage of predictions within X kilometers of ground truth.

Table 10: Detailed performance metrics by method.

Method	n	mean	median	sd	min	Q1	Q3	max	≤10 km	≤25 km	≤50 km
H-1	21	75.08	70.70	43.30	5.98	36.87	118.36	139.59	4.8%	9.5%	42.9%
M-1	43	41.65	27.39	27.32	7.59	18.45	70.04	103.49	7.0%	37.2%	62.8%
M-2	43	23.39	14.27	19.86	2.67	8.17	36.85	87.35	30.2%	60.5%	93.0%

Method	n	mean	median	sd	min	Q1	Q3	max	$\leq 10$ km	$\leq 25$ km	$\leq 50$ km
M-3	43	50.25	48.40	24.93	6.29	27.36	69.53	123.04	4.7%	16.3%	53.5%
M-4	43	28.51	25.42	20.77	2.14	12.09	40.49	98.72	20.9%	48.8%	86.0%
M-5	43	27.93	24.97	19.46	3.03	14.66	37.25	98.86	16.3%	51.2%	90.7%
M-6	43	43.05	34.02	36.07	5.17	17.11	49.74	176.33	4.7%	34.9%	76.7%
T-1	43	37.65	33.61	24.54	0.59	18.57	62.30	110.19	14.0%	32.6%	69.8%
T-4	43	37.23	34.22	23.94	0.59	21.78	53.35	101.85	16.3%	32.6%	74.4%

### B.3 Cost-Accuracy Trade-off

Table 11 examines the cost-accuracy relationship, emphasizing the economic efficiency of GPT-4o, which achieves near-top performance at just \$1.05 per 1,000 grants processed. “Cost per +1%  $\leq 10$  km hit” indicates the marginal cost of improving high-precision prediction rate by one percentage point.

Table 11: Cost-accuracy trade-off by model family.

Model	Mean error km	$\leq 10$ km hit-rate	Cost per 1k located (USD)	Cost per +1% $\leq 10$ km hit (USD)
o3-2025-04-16	23.39	30.2%	\$127.46	\$4.22
gpt-4o-2024-08-06	27.93	16.3%	\$1.05	\$0.06
gpt-4.1-2025-04-14	32.87	18.6%	\$3.49	\$0.19
o4-mini-2025-04-16	39.65	10.5%	\$10.69	\$1.02
gpt-3.5-turbo	43.05	4.7%	\$0.10	\$0.02
o3-mini-2025-01-31	50.25	4.7%	\$14.15	\$3.04
human-gis	71.40	4.7%	\$3,255.81	\$700.00

### B.4 Processing Time Analysis

Table 12 quantifies the latency advantage of automated methods over traditional GIS workflows. “Speedup” shows relative improvement over the traditional GIS baseline.

Table 12: Processing time by model.

Model	Hours per located	Hours per 1k located	Speedup vs. baseline
gpt-4o-2024-08-06	0.0002	0.184	1,152×
gpt-3.5-turbo	0.0002	0.232	914×
gpt-4.1-2025-04-14	0.0009	0.915	232×
o3-2025-04-16	0.0133	13.294	16×
o3-mini-2025-01-31	0.0088	8.791	24×
o4-mini-2025-04-16	0.0067	6.698	32×
human-gis	0.2120	212.045	1×

## B.5 Token Usage Statistics

Table 13 provides detailed token consumption data across all models, offering insight into computational efficiency.

Table 13: Token consumption by model across all test runs.

Model	Input tokens	Output tokens	Tokens per 1k located
gpt-4o-2024-08-06	6,698	900	176,698
gpt-3.5-turbo	6,773	820	176,581
gpt-4.1-2025-04-14	142,258	4,193	1,702,919
o3-2025-04-16	6,653	146,085	6,923,209
o3-mini-2025-01-31	6,653	142,020	6,739,372
o4-mini-2025-04-16	274,903	146,590	6,340,337

Tool-augmented methods consumed on average 1.49× more tokens than pure-prompt counterparts (4,985,953 vs. 3,355,078 tokens per 1,000 located grants). However, this effect varied dramatically by model architecture: adding tools to GPT-4.1 increased token usage by 18.3× (176,698 → 3,229,140), while o4-mini showed only a 1.14× increase (5,937,907 → 6,742,767).

## B.6 Professional GIS Benchmark Analysis

Table 14 provides a more detailed analysis of the professional GIS benchmark (H-1) (Bashorun, 2025) results, categorized by precision level. This breakdown reveals that even with expert domain knowledge and access to specialized historical gazetteers, more than 41.9% of the human-geocoded grants were located at only state-level precision. “High” indicates grants where both county boundaries and specific landmarks were used; “Medium” indicates county-centroid placement; “Low” indicates state-level precision only.

Table 14: Professional GIS benchmark results by accuracy category.

Accuracy Category	N	Share (%)	Mean Error (km)	Median Error (km)
Overall	43	100.0	71.40	60.20
High (County + Landmarks)	19	44.2	68.88	48.09
Medium (County centroid)	6	14.0	87.95	82.11
Low (State-level)	18	41.9	68.55	63.26

Notably, even the “High” precision category (where both county boundaries and specific landmarks were identified) still resulted in a mean error of 68.88 km—substantially higher than all the automated methods except GPT-3.5-turbo (M-6). This underscores the inherent difficulty of the task and further highlights the significance of the accuracy improvements achieved by the LLM approaches.

## Appendix C Supplementary Figures

### C.1 Error Distribution Plots

Figure 8 shows the distribution of geolocation error for each method as a boxplot, complementing the violin plot and CDF already presented in the main text. The boxplot highlights the median error (central line),

interquartile range (box), and outliers (points) for each method, providing a clear view of error distribution and central tendency.

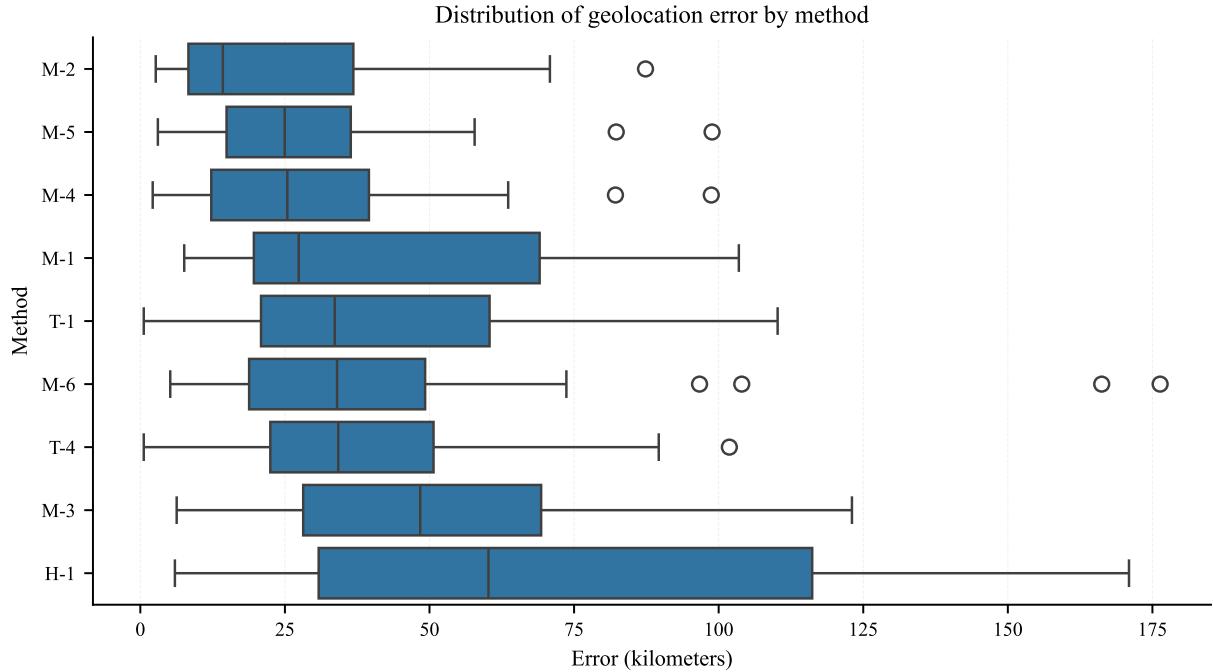


Figure 8: Error Distribution Boxplot by Method

## C.2 Error Maps

Figure 9 plots all six methods for every locatable grant against ground truth coordinates (black stars). Error distances are shown as dashed lines connecting predictions to ground truth.

## C.3 Cost-Accuracy Tradeoff

Figure 10 presents the cost-accuracy Pareto frontier across all evaluated methods. This visualization expands upon Figure 4 from the main text, providing additional detail on the trade-offs between monetary expenditure and geolocation accuracy.

The plot illustrates how automated methods establish a clear Pareto frontier along the bottom edge, with cost (x-axis, log scale) ranging over several orders of magnitude while mean error (y-axis) varies between ~23-75 km. The professional GIS baseline appears in the upper-right quadrant, reflecting its combination of high cost and relatively high error. GPT-4o offers the most favorable cost-to-accuracy ratio despite not achieving the absolute lowest error, while o3-2025-04-16 delivers the best accuracy at significantly higher cost.

## C.4 Latency-Accuracy Tradeoff

Processing time presents another critical dimension for evaluation. The figure below shows how each method balances computational latency against geolocation accuracy. LLM methods cluster in the bottom-left quadrant, delivering results in seconds rather than minutes, while maintaining lower error rates than the professional GIS approach.



Figure 9: Grid view of all 43 mapped grants showing ground truth and all predictions.

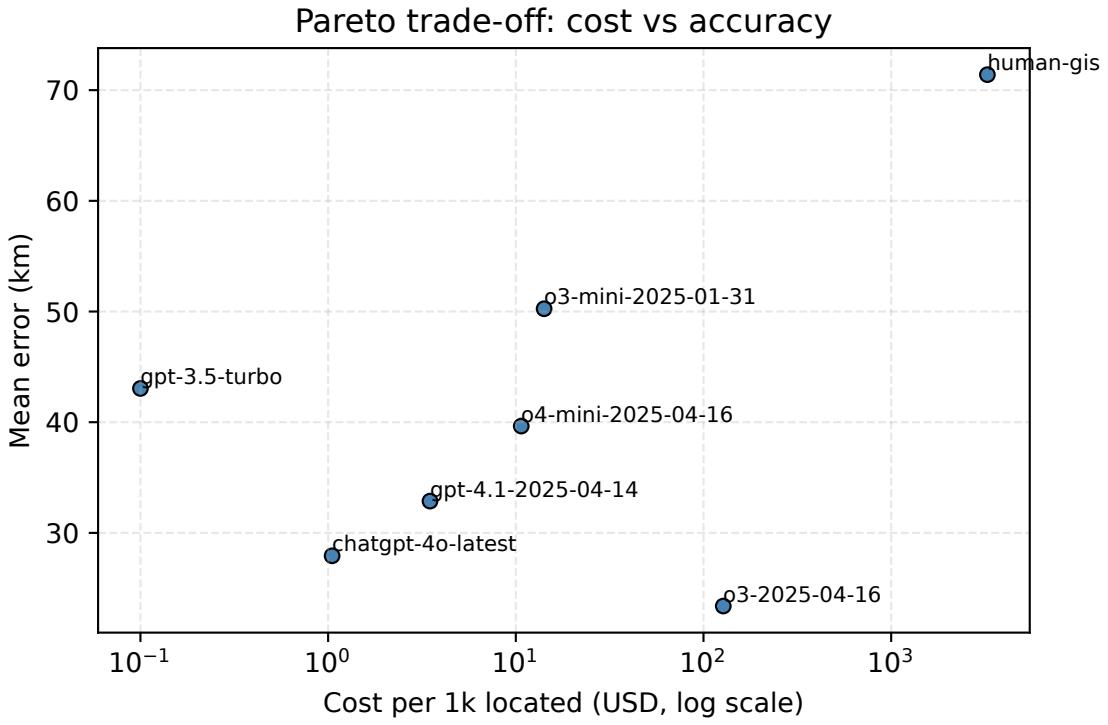


Figure 10: Cost-Accuracy Pareto Frontier

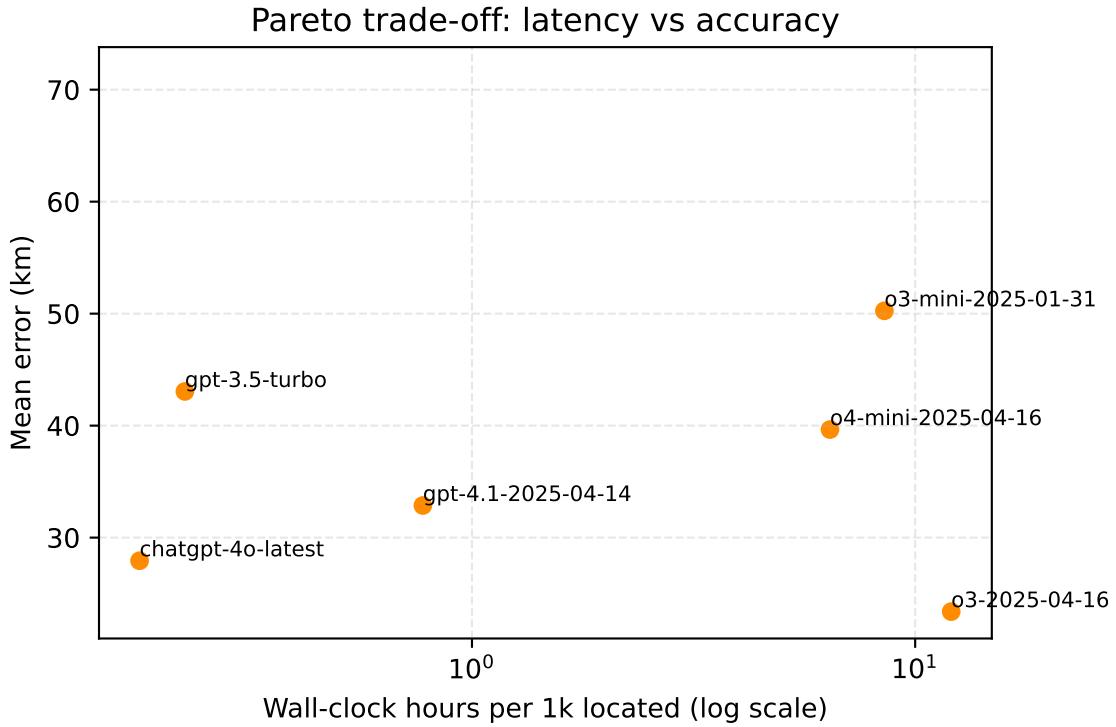


Figure 11: Latency-Accuracy Tradeoff

Figure 11: Latency-Accuracy Tradeoff. This figure plots mean error (km) against processing time per grant (seconds) for each evaluated method. All automatic methods produce coordinates in 0.2–13 s of computation time, compared to the GIS analyst’s labor time of ≈432 s per grant. Note the logarithmic scale on the x-axis.

## Appendix D Tool Augmentation Analysis

Table 15 isolates the impact of providing tool access to identical models, revealing that tool augmentation does not consistently improve accuracy. For GPT-4.1, enabling tool access increases mean error by 30.6%, while for the o4-mini model, it decreases error by 9.6%.

### D.1 Direct Tool vs. Non-Tool Comparison

Table 15 provides a head-to-head comparison of identical models with and without tool access. This controls for model architecture effects and isolates the impact of tool access alone.

Table 15: Side-by-side comparison of identical models with and without tool access.

Model	Category	mean	median	sd	min	max	$\leq 1$ km	$\leq 5$ km	$\leq 10$ km	$\leq 25$ km	$\leq 50$ km
gpt-4.1-2025-04-14	one shot	28.51	25.42	20.77	2.14	98.72	0.0%	4.7%	20.9%	48.8%	86.0%
gpt-4.1-2025-04-14	tool chain	37.23	34.22	23.94	0.59	101.85	2.3%	14.0%	16.3%	32.6%	74.4%
o4-mini-2025-04-16	one shot	41.65	27.39	27.32	7.59	103.49	0.0%	0.0%	7.0%	37.2%	62.8%
o4-mini-2025-04-16	tool chain	37.65	33.61	24.54	0.59	110.19	4.7%	11.6%	14.0%	32.6%	69.8%

### D.2 Quantified Tool Effect

Table 16 quantifies the precise impact of tool access, showing percentage changes in mean error and percentage point (pp) changes in accuracy bands. “ $\Delta$  Mean %” shows percent change in mean error; “pp” indicates percentage point differences in accuracy bands. Negative percentages for mean change indicate worse performance with tools.

Table 16: Quantified effect of tool augmentation.

Model	Mean M	Mean T	$\Delta$ Mean %	$\Delta \leq 1$ km pp	$\Delta \leq 5$ km pp	$\Delta \leq 10$ km pp	$\Delta \leq 25$ km pp	$\Delta \leq 50$ km pp
gpt-4.1-2025-04-14	28.51	37.23	-30.6%	+2.3 pp	+9.3 pp	-4.7 pp	-16.3 pp	-11.6 pp
o4-mini-2025-04-16	41.65	37.65	9.6%	+4.7 pp	+11.6 pp	+7.0 pp	-4.7 pp	+7.0 pp

While the o4-mini model showed a modest improvement with tools, the higher-quality GPT-4.1 model performed substantially worse when given tool access.

### D.3 Top-performing methods per tool-use category

Table 17 shows a direct head-to-head comparison of the best-performing tool-use method vs the best non-tool method. M-2 (o3-2025-04-16, one-shot prompt) substantially outperforms the best tool-augmented method (T-4), achieving a 37% lower mean error and nearly double the proportion of predictions within 10 km.

Table 17: Best-performing methods per category.

Method	mean	median	sd	min	Q1	Q3	max	≤10 km	≤25 km	≤50 km
M (M-2)	23.39	14.27	19.86	2.67	8.17	36.85	87.35	30.2%	60.5%	93.0%
T (T-4)	37.23	34.22	23.94	0.59	21.78	53.35	101.85	16.3%	32.6%	74.4%

At the category level, the best non-tool method (M-2) significantly outperformed the best tool-augmented method (T-4) across all error metrics.

### D.4 Tool Call Distribution

Table 18 expands on the tool usage patterns discussed in Section 6.6, providing detailed statistics on how each model interacted with the available geocoding and centroid-computation tools.

Table 18: Distribution of tool calls by method and tool type.

Method	Tool Type	Mean	SD	Median	Min	Max
T-1 (o4-mini)	geocode_place	3.79	2.41	3	1	10
T-1 (o4-mini)	compute_centroid	0.16	0.37	0	0	1
T-4 (gpt-4.1)	geocode_place	2.05	1.78	1	1	7
T-4 (gpt-4.1)	compute_centroid	0.25	0.43	0	0	1

### D.5 ToolSearch Efficiency

“Selected call index” indicates which API call in the sequence produced the coordinates used in the final answer. Lower values indicate more efficient search strategies.

Table 19: Tool search efficiency metrics.

Method	Mean selected call index	Median	First-call success rate
T-1 (o4-mini)	2.29	1	69.0%
T-4 (gpt-4.1)	1.95	1	72.7%

The more economical approach of GPT-4.1 is evident in both the distribution of calls and search efficiency. While T-1 (o4-mini) made nearly twice as many geocoding calls on average (3.79 vs. 2.05), it achieved a

slightly lower first-call success rate (69.0% vs. 72.7%). This pattern aligns with the overall finding that tool augmentation does not consistently improve accuracy; in fact, the additional API calls may introduce noise through spurious matches to modern place names that bear little relation to colonial-era settlements.

Overall, both models heavily favored direct geocoding over centroid computation, with geocode:centroid ratios of 23.29:1 for T-1 and 8.18:1 for T-4. This suggests that the models primarily relied on finding exact matches for place names mentioned in the abstracts rather than triangulating from multiple reference points—a strategy that may explain their susceptibility to modern naming coincidences.

## References

- Bashorun, B. (2025). *Colonial virginia land grants geocoding dataset*. Professional GIS Services.
- Coughlan, M. R., & Nelson, D. R. (2018). Influences of native american land use on the colonial euro-american settlement of the south carolina piedmont. *PLOS ONE*, 13(3), e0195036. <https://doi.org/10.1371/journal.pone.0195036>
- Direct Line Software. (2010). *DeedMapper 4.2*. Commercial software for historical deed mapping. <https://directlinesoftware.com/deeds/>
- Dobbs, G. R. (2009). Backcountry settlement development and indian trails: A GIS land-grant analysis. *Social Science Computer Review*, 27(3), 331–347. <https://doi.org/10.1177/0894439308329760>
- Fausz, J. F. (1971). *Patterns of settlement in the James River Basin, 1607-1642* [Master's thesis, College of William; Mary]. <https://dx.doi.org/doi:10.21220/s2-jedw-z031>
- Gritta, M., Pilehvar, M. T., & Collier, N. (2018). Which melbourne? Augmenting geocoding with maps. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, 1285–1296. <https://doi.org/10.17863/CAM.27796>
- Hu, X., & Kersten, J. (2024). Enhancing toponym resolution with fine-tuned LLMs (Llama2). *Proceedings of the Second International Workshop on Geographic Information Extraction from Texts at ECIR 2024 (GeoExT@ECIR)*, 52–56.
- Huang, C., Chen, S., Li, Z., Qu, J., Xiao, Y., Liu, J., & Chen, Z. (2024). GeoAgent: To empower LLMs using geospatial tools for address standardization. *Findings of the Association for Computational Linguistics: ACL 2024*, 6048–6063. <https://doi.org/10.18653/v1/2024.findings-acl.362>
- Julian, G. M., & Abbott, R. J. F. (2014). GIS and genealogy: Teaching GIS while learning about the past. *ArcUser*. <https://www.esri.com/about/newsroom/arcuser/gis-and-genealogy>
- Library of Virginia. (2025). *Virginia land office patents and grants online database*. Web portal. <https://lva-virginia.libguides.com/land-grants>
- Loudoun County Office of Mapping. (2015). *Original land grants of loudoun county, VA [GIS dataset]*. Interactive ArcGIS WebMap. <https://www.arcgis.com/apps/MapTools/index.html?appid=d3c4ba1031564f919ca28c9bb5a48350>
- Mitchell, B. (1977). *Beginning at a white oak: Patents and northern neck grants of fairfax county, virginia*. Office of Comprehensive Planning, Fairfax County. <https://www.fairfaxcounty.gov/history-commission/sites/history-commission/files/assets/documents/resources/map.pdf>
- Nugent, N. M. (1979). *Cavaliers and pioneers: Abstracts of virginia land patents and grants, volume three: 1695-1732*. Virginia State Library.
- O'Sullivan, K., Schneider, N. R., & Samet, H. (2024). Metric reasoning in large language models. *Proceedings of the 32nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24)*, 501–504. <https://doi.org/10.1145/3678717.3691226>
- Savarro, D., Zago, D., & Zoia, S. (2024). Leveraging large language models to geolocate linguistic variations in social media posts. *arXiv Preprint arXiv:2407.16047*. <https://doi.org/10.48550/arXiv.2407.16047>
- Schottlander, D., & Shekel, T. (2025). *Geospatial reasoning: Unlocking insights with generative AI and*

- multiple foundation models.* Google Research Blog. <https://research.google/blog/geospatial-reasoning-unlocking-insights-with-generative-ai-and-multiple-foundation-models/>
- Wu, Y., Zeng, Z., Liu, K., Xu, Z., Ye, Y., Zhou, S., Yao, H., & Li, S. (2025). Text geolocation prediction via self-supervised learning. *ISPRS International Journal of Geo-Information*, 14(4), 170. <https://doi.org/10.3390/ijgi14040170>
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *Proceedings of the 11th International Conference on Learning Representations*. <https://doi.org/10.48550/arXiv.2210.03629>