Uniting the World by Dividing it: Federated Maps to Enable Spatial Applications

Sagar Bharadwaj
Carnegie Mellon University

Anthony Rowe
Carnegie Mellon University

Srinivasan Seshan
Carnegie Mellon University

ABSTRACT

The emergence of the Spatial Web – the Web where content is tied to real-world locations has the potential to improve and enable many applications such as augmented reality, navigation, robotics, and more. The Spatial Web is missing a key ingredient that is impeding its growth – a spatial naming system to resolve real-world locations to *names*. Today's spatial naming systems are digital maps such as Google and Apple maps. These maps and the location-based services provided on top of these maps are primarily controlled by a few large corporations and mostly cover outdoor public spaces. Emerging classes of applications, such as persistent world-scale augmented reality, require detailed maps of both outdoor and indoor spaces. Existing centralized mapping infrastructures are proving insufficient for such applications because of the scale of cartography efforts required and the privacy of indoor map data.

In this paper, we present a case for a federated spatial naming system, or in other words, a federated mapping infrastructure. This enables disparate parties to manage and serve their own maps of physical regions and unlocks scalability of map management, isolation and privacy of maps. Map-related services such as address-to-location mapping, location-based search, and routing needs re-architecting to work on federated maps. We discuss some essential services and practicalities of enabling these services.

ACM Reference Format:
Sagar Bharadwaj, Anthony Rowe, and Srinivasan Seshan. 2025. Uniting the World by Dividing it: Federated Maps to Enable Spatial Applications. In *Workshop in Hot Topics in Operating Systems (HOTOS 25), May 14–16, 2025, Banff, AB, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3713082.3730371

1 INTRODUCTION

Many have predicted the future of the Web to be the integration of Web-like content with the real-world. This overlay of virtual content on top of the physical world, which we refer to as the Spatial Web, holds promise for dramatically changing many applications from navigation, to engineering, education, and more. While the vision is promising, the reality is that Spatial Web applications are difficult to create.

Looking at existing systems, we can see that a key enabler for such systems has often been the underlying infrastructure that makes it easy to discover and reference content. For the Web, the Domain Name System (DNS) provided a simple mechanism to convert human-readable names (domain names and URLs) to server IP addresses that can provide relevant content. The Spatial Web requires a similar infrastructure to relate human-readable names (e.g., The White House) with real-world locations and the content associated with that location (e.g., White House web page). In other words, we need a spatial naming system. Since this naming system translates names to physical locations, we use the term *map* or *mapping service* to refer to this naming system.

The design of a naming system significantly influences and constrains the behavior of any distributed system built on top of it. For example, a naming system's mechanisms to add new entities to a distributed system can create bottlenecks to maintaining and scaling a system. The Web and the Internet at large were able to rapidly scale and incorporate a large number of hosts in their early days primarily due to the federated and pseudo-decentralized nature of the DNS. The federated design of the DNS allowed organizations to independently manage and control their level of participation on the Internet. The link between spatial naming and application constraints is no different and we can see this relationship in current deployed systems.

Today's spatial naming systems are digital maps like Google and Apple maps. These digital maps are supported by centralized infrastructures and maintained by large corporations. Only the information that is gathered and exposed by organizations maintaining these centralized maps is available to applications, thereby severely limiting their functionality. In this paper, we make the case that to enable the rapid growth of the Spatial Web, we need an underlying spatial naming system or a map that is federated and can be independently controlled and maintained by disparate organizations.

Extending spatial applications indoors is a use case that especially highlights the importance of a federated mapping infrastructure. Indoor maps contain sensitive information that needs to be owned and controlled by the owner of the

HOTOS'25, May 14–16, 2025, Banff, AB, Canada

physical space. Many organizations, such as stores, would benefit from providing accurate map data for applications such as product search [3, 22], but would not be willing to publicly host detailed maps. Furthermore, the storage and cartography effort required for indoor spaces far outweighs that of outdoor maps. Some estimate that there could be more than 100 billion square feet (and growing) of indoor space in the world [6] and surveying this space will likely be impractical for any single centralized organization.

In this paper, we introduce *OpenFLAME*[1], an architecture for a federated mapping infrastructure. *OpenFLAME* is organized into 'map servers' – independent services deployed by potentially disparate parties that provide map data and location-based services confined to a physical region. *OpenFLAME* provides the means to discover and tie these services together thereby providing a unified spatial naming system that can support spatial applications. A federated mapping infrastructure presents several challenges.

- The maps are heterogeneous and can vary from one another in multiple ways. They can have different fidelities (eg. 2D and 3D maps), laid out in different coordinate systems (eg. indoor maps may not have exact latitude and longitude coordinates as its difficult to align them [10, 25]), and have different labels for common overlapping areas.
- Providing location-based services such as address-to-location lookup, routing, and search on top of federated maps requires a re-design of their architecture.
- We need a system that discovers map providers in a region. The discovery system should account for fuzziness of map boundaries and multiple ownership of physical regions.

In § 2, we give an example of a typical Spatial Web application that would be made possible by a federated mapping infrastructure. § 3 describes the abstraction of map and map servers. § 4 briefly describes the abstraction of location-based services typically used by spatial applications that would act as specifications for services to be enabled by *OpenFLAME*. We also describe how the existing centralized model provides these services. § 5 discusses some practical considerations for realizing the federated mapping infrastructure in practice.

2 EXAMPLE APPLICATION

To better understand the needs of future Spatial Web applications, we start by describing in detail what we consider a typical such application – grocery store navigation.

Let us consider a scenario where a user wishes to search for a product of interest, e.g., a particular flavor of seaweed, near their location. The application then provides the user with pedestrian navigation guidance to the exact shelf in a grocery store nearby that stocks the seaweed.

First, consider how an existing navigation application would support this task. The application would have to rely on a centralized database of destination locations and navigable paths exposed by a map provider such as Google. However, these databases are typically limited to street addresses and public landmarks. The seaweed in the store or the store aisles, for example, would not be a part of the map database unless the store has requested Google to maintain a database of the store inventory indexed by shelf locations (which would involve complicated and expensive integration between Google and the store's systems). Second, the navigation application would rely on a combination of technologies to determine the location of a user with respect to the map data, including GPS, image data from Google Street View, and WiFi/cellular signal strength. The availability of these technologies is limited to outdoor locations for GPS and to StreetView covered regions for image localization (typically public roads). Therefore, the navigation application would not work well within the store even if the Google Maps database included the user's product of interest.

Ideally, we would like the application to provide precise visual guidance along all steps of the path. Existing applications fail to meet this objective in multiple ways – failing to provide precise guidance when localization is inaccurate and, in this case, failing to provide complete guidance as the requested seaweed is not part of Google Map's database.

We envision *OpenFLAME* would enable such spatial applications and more. In this case, the grocery store would maintain its own map independent of Google Maps. The grocery store's map would have a precise map of the store, along with its current inventory laid out against the local map. When the navigation application searches for the nearest seaweed, *OpenFLAME* would discover the grocery store's map nearby and request from it the necessary shelf location information. Once the map and the location of interest within the map have been identified, the application can now estimate the navigation route from the user's current location on the street to the grocery store shelf. The route would be a combination of routes calculated by both Google Maps and the grocery store's map. Google Map route would lead the user to the storefront, while the grocery store's map would lead the user further to the shelf with the desired product. When outdoors, the application could rely on GPS or Street-view imagery to localize the user, but once indoors, the application would switch to the localization service provided by the grocery store's map to localize the user precisely within the store.

[1]OpenFLAME stands for Open Federated Localization and Mapping Engine

2.1 Challenges

The application described above exposes some challenges associated with realizing a federated mapping infrastructure.

Location-based services: Notice that multiple location-based services were used by the application. *Location-based search* to search for a product. *Routing* to calculate the path on which to navigate the user. Finally, *localization* to estimate where the user is with respect to the map while navigating. In a centralized infrastructure, the client application simply requests these services from a single entity that has access to all of the map data. In *OpenFLAME*, the client device first has to discover relevant map servers and request the required services from these map servers, stitching the results if required. § 4 defines the main classes of location-based services. § 5.2 shows how location-based services can be provided on top of a federated mapping infrastructure.

Heterogeneity of maps: Maps can be heterogeneous in multiple ways. Google Maps and the grocery store map in the above example can differ with respect to: *Fidelity* – the grocery store map can be more detailed with precise 3D information of shelves and what is stocked in them. Google Maps would be sparser in comparison. *Coordinate frame of reference* – the grocery store map might not be properly oriented in the geographic coordinate system of latitudes and longitudes as well as Google Maps. Aligning an indoor map accurately with the geographic coordinate system is a notoriously difficult problem [10, 25] and needs expensive survey equipment [32, 33]. The data within the grocery store map is only precisely aligned against its own separate coordinate system. *Common area labels* – even if the maps have an overlapping region (e.g., some portion of the storefront), they might be labeled differently making alignment harder. Heterogeneity makes it challenging for map providers to jointly provide unified location-based services to applications.

Overlapping maps: In the above example, Google Maps also has sparse map data coverage of the region containing the grocery store. Therefore, multiple maps can overlap with each other. Note that in traditional naming systems such as DNS, there is no ambiguity concerning the ownership of a namespace. For example, the domain `www.google.com` is owned by Google and no other organization. However, the possibility of overlapping maps makes building a federated spatial naming system challenging.

3 MAP AND MAP SERVERS

A map is a representation encoding relationships and attributes of spatial entities in a geographic region. While traditionally, a map refers to the visual representation of geographic features, in our context it is the data that underlies such visual representations. We adopt the widely used OpenStreetMap's data model for a map [27]. A map has three major elements – nodes, ways and relations. A *node* represents a point on the map, defined by its coordinates within the map. A *way* is an ordered list of nodes that defines a polyline and used to represent navigable paths, borders, rivers, etc. A *relation* is used to represent a collection of related map elements – nodes, ways or other relations. Each map element can have any metadata associated with it.

A map server is a system that stores the map of a region and provides services such as search and routing on the map. The usefulness of a map server is determined by the services it implements. It can also impose fine-grained security and privacy policies on users and applications (§ 5.3).

A map in *OpenFLAME* is conceptually equivalent to a *zone* in a traditional naming system like the DNS [23]. A *DNS zone* is a portion of the DNS namespace that is independently managed by an organization. Similarly, a map is a portion of the spatial namespace that is independently managed by an organization. A map server is akin to a *name server* in DNS parlance. However, there are two key differences between the concept of a DNS zone and a map as a zone. First, unlike DNS zones, the 'boundary' of a map is fuzzy. Whenever a DNS zone, such as `google.com` is assigned to an organization, it is clear that even slight variations such as `googli.com` is outside the purview of the organization. However, in the spatial world, the polygonal boundary that might define the confines of a map is never exact. For example, the boundary of the grocery store map in the example in § 2 might spill over to other stores nearby. This is inevitable especially for indoor maps, where finding out the exact geographic coordinates of walls and boundaries involves expensive surveying. Second, multiple maps may cover the same physical region. For example, *OpenFLAME* should include both Google and Apple maps in its infrastructure. In DNS, ownership over a namespace is given exclusively to one zone. However, in the spatial naming system, multiple zones may cover the same physical region. In § 5.1, we briefly discuss how these challenges can be tackled in practice.

4 LOCATION-BASED SERVICES

A spatial naming system is only useful to spatial applications if we can provide location-based services on top of it. In this section, we will discuss existing location-based services that spatial applications use, which will act as specifications for the services that our naming system should enable. In § 5.2, we discuss how these services can be implemented in practice on a federated map.

Map providers offer a host of location-based services on their platform [14, 20, 24, 26]. However, all of these services can be derived from a smaller set of base services. Some base services are different kinds of queries on the map data: simple address-to-location lookups, location-based search,

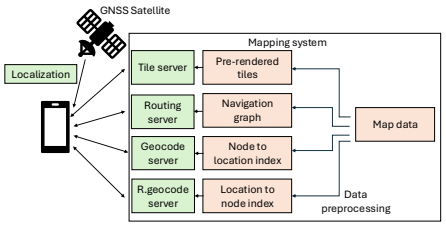

Figure 1: Centralized Architecture

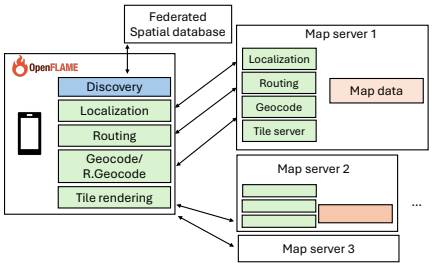

Figure 2: *OpenFLAME* Architecture

and routing. Other base services are application enablers such as localization and map visualization. We describe these base services and how other services are derived from them.

Forward and reverse geocode: The process of converting a text-based address to a location on the map [13] is forward geocode. This underlies the sub-service of placing markers on the map given an address and is also a preliminary step in routing, as applications usually request a path between two addresses rather than two map nodes. The service that coverts a geographic location to a map node is called reverse geocode. It is the underlying service that supports click interactions on the map and snapping raw GPS coordinates to roads on the map while navigating [19, 21].

Location-based search: Searching for map nodes using their metadata or features as keywords in or around a region is called location-based search. This service serves requests of the form "restaurants around me", "parking spot near the theater", etc. Map providers index map node features and metadata against their location to provide this service.

Routing: Routing is the service that provides a path from one map node to another. The path usually optimizes a metric such as distance, travel time, number of turns, toll price etc. Today's map providers typically run versions of shortest-path graph algorithms against their centralized map data to estimate optimized paths [4].

Localization: The service that informs a device of its location and orientation with respect to a map is called localization. Today, since most map data aligns with a global geographic system, location-based applications typically rely on large-scale positioning systems like GPS, WiFi access points, cell towers, and Bluetooth [2, 7, 16].

Tile rendering: Tile rendering powers interactive maps by delivering map tiles—2D images or 3D meshes—based on the user's latitude, longitude, and zoom level. As users drag or zoom, the tile server dynamically loads the appropriate tiles to update the view.

4.1 Centralized map model

Figure 1 shows how today's centralized map infrastructures provide location-based services. The map data of the world is preprocessed into different forms required for each location-based service. For example, to provide the routing service, map data might be converted to a graph and then preprocessed using the contraction hierarchies algorithm which makes routing queries faster to compute [11]. The tile rendering service might pre-render tiles corresponding to latitudes, longitudes and zoom levels even before they are requested by any client. Geocode, reverse geocode, and location-based search would involve indexing map nodes and their metadata against geographic coordinates.

The API calls to each service would then use the pre-processed data to serve requests. For example, the tile service API would fetch appropriate tiles from the pre-rendered set and serve them to the client.

5 PRACTICAL CONSIDERATIONS

Figure 2 shows the *OpenFLAME* architecture. *OpenFLAME* client relies on a discovery mechanism to identify all the map providers in a region. § 5.1 discusses how the challenges presented in § 3 can be tackled in practice while implementing a map server discovery system. § 5.2 discusses the split of responsibilities between the client and map servers for providing services described in § 4. Federation enables a finer grained security and privacy model discussed in § 5.3.

5.1 Map server discovery

The foundation that underlies all location-based services is the discovery layer. The discovery system would essentially maintain the data mapping locations to map servers in that location. The discovery query would involve the coarse location of the device obtained from ubiquitous sources like the GPS. The discovery system would then respond to the query with a list of map providers for the region.

Several existing systems such as spatial databases [17, 18, 29], Geographic Information Systems (GIS) [5, 9], and the Intentional Naming System (INS) [1] can be used for spatial discovery. However, deploying the infrastructure required for these systems from the ground up and maintaining them would be a major impediment to the adoption of the Spatial Web. Let us consider the following observations about the

discovery system that allows us to use an existing widely deployed infrastructure—the DNS. 1. The fuzziness of map boundaries discussed in § 3 does not require a database that maintains precise polygonal boundaries. 2. Discovery queries are exclusively read queries and do not require a full-fledged database with transaction processing. 3. The address of the map servers are not expected to change frequently so the system would benefit from a ubiquitous caching mechanism.

To repurpose the already federated DNS to work as the spatial database, we can leverage spatial indexing systems (e.g., S2 [15], H3 [31]) to convert locations to hierarchical domain names. A polygonal region, or a zone, can be approximated by a collection of domain names. Coarse location in the form of latitude and longitude can also be converted to a domain name. Therefore, the discovery query would be a simple domain look-up on the DNS. Leveraging DNS for our purpose gives us access to its ubiquitous caching mechanisms, large-scale deployments, and infrastructure. Previously, the Spatial Name System [12] has also considered using the DNS to assign and resolve hierarchical location-based names. However, their use of civic addresses as domain names while maintaining geodetic location as part of record data results in inefficient location-based discovery process.

5.2 Location-based services

In this section, we discuss split of responsibilities between *OpenFLAME* client and map servers in providing location-based services from § 4 on top of a federated map.

Geocode: Given a text string of a hierarchical address, the client first uses the geocode service of a large world-map provider (e.g., OpenStreetMap [28]) to get the coarse location of a part of the address. The client then discovers finer map servers in the coarse location which search in their own maps for the exact address.

Reverse geocode and location-based search: Searching for map nodes around a location would begin by the client discovering map servers around a given location. The client would then ask each map server to search for the relevant items within their maps and return relevant results, if any. The client would then rank results from multiple map servers and present them to the application.

Routing: The client first obtains the location of the source and destination addresses using the Geocode service described above. Then it discovers all the map servers that lie along the way from the source to the destination. Each map server would calculate the route that is relevant for the region that they cover. The client would collect paths from all relevant map servers, and stitch them together such that the final path optimizes a metric of interest.

Localization: The process of localizing a device starts with the *OpenFLAME* client discovering map servers in the location. The client might discover multiple overlapping servers or even unrelated maps because of the coarseness of the discovery process. Once the map servers are discovered, the client sends them 'location cues' collected by the device sensors – images, beacon signals, fiduciary tag scans, etc. The location cue sent to the map server depends on the localization technology advertised by the server. The map servers accept location cues, localize the device within their map, and return the results to the client. The client then selects the best one by comparing these results with its own IMU (Inertial Measurement Unit) sensors or local SLAM (Simultaneous Localization and Mapping) algorithm [30]. The most plausible result is returned to the application.

Tile rendering: Each map server would expose a visual representation of its map data as 2D images, 3D meshes or other forms. The client would download these representations from multiple discovered map servers and stitch them together before showing them to the user. For example, stitching together map data in different coordinates and projection systems can be done using manual correspondences between maps (e.g., MapCruncher [8]).

5.3 Security and privacy model

Unlike in the case of a centralized map, map providers in *OpenFLAME* can control access to their data and services in fine-grained ways as they can implement separate authentication processes for each of the services and map data. *User-level control* – A map server covering a university, for example, may only serve users who can authenticate with the university's email address. This ensures users who are not from the university cannot get fine-grained map data. *Service-level control* – A map server, for example, may provide its tile service to a large set of users so they can view the map. However, it may choose to provide localization service only to a small set of users who are supposed to have physical access to the place. *Application-level control* – A university might provide localization service only if it comes from the campus navigation application and trust that the application has implemented its own way of authenticating users.

6 CONCLUSION

In this paper we present the need for a federated spatial naming system, or a map, to enable emerging Spatial Web applications. We then discuss how location-based services that are primarily utilized by today's spatial applications can be provided on top of a federated map. We hope that this paper encourages the systems community to think about the underlying infrastructure required to enable the next generation of the Web.

REFERENCES

[1] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. 1999. The design and implementation of an intentional naming system. In *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles (SOSP '99)*. Association for Computing Machinery, New York, NY, USA, 186–201. https://doi.org/10.1145/319151.319164

[2] Apple. 2024. Core Location Apple. https://developer.apple.com/documentation/corelocation/. (2024). Accessed: 2024-09-01.

[3] AukiLabs. 2024. Auki labs retail solutions. https://www.aukilabs.com/solutions/industries/retail. (2024). Accessed: 2024-09-17.

[4] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. 2016. *Route Planning in Transportation Networks*. Springer International Publishing, Cham, 19–80. https://doi.org/10.1007/978-3-319-49487-6_2

[5] CARTO. 2025. CARTO. https://carto.com/. (2025). Online. Accessed: April 2025.

[6] University of Michigan Center for Sustainable Systems. 2024. Commercial Buildings Factsheet. https://css.umich.edu/publications/factsheets/built-environment/commercial-buildings-factsheet. (2024). Accessed: 2024-09-01.

[7] MDN Web Docs. 2024. Geolocation MDN. https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API. (2024). Accessed: 2024-09-01.

[8] Jeremy Elson, Jon Howell, and John R Douceur. 2007. MapCruncher: integrating the world's geographic information. *ACM SIGOPS Operating Systems Review* 41, 2 (2007), 50–59.

[9] Esri. 2025. ArcGIS. https://www.arcgis.com/index.html. (2025). Online. Accessed: April 2025.

[10] Ryan Felts, Marc Leh, Tracy McElvaney, and Dereck Orr. 2015. *Location-Based Services R&D Roadmap*. US Department of Commerce, National Institute of Standards and Technology, US.

[11] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. 2012. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transportation Science* 46, 3 (Aug. 2012), 388–404. https://doi.org/10.1287/trsc.1110.0401

[12] Ryan Gibb, Anil Madhavapeddy, and Jon Crowcroft. 2023. Where on Earth is the Spatial Name System?. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks (HotNets '23)*. Association for Computing Machinery, New York, NY, USA, 79–86. https://doi.org/10.1145/3626111.3628210

[13] Daniel W. Goldberg, John P. Wilson, and Craig A. Knoblock. 2007. From Text to Geographic Coordinates: The Current State of Geocoding. *Urisa Journal* 19 (2007), 33. https://api.semanticscholar.org/CorpusID:5517082

[14] Google. 2024. Google Maps Platform. https://developers.google.com/maps. (2024). Accessed: 2025-01-14.

[15] Google. 2024. S2 library. http://s2geometry.io/. (2024). Accessed: 2024-09-01.

[16] Google. 2025. Fused Location Provider Android. https://developer.android.com/develop/sensors-and-location/location/retrieve-current. (2025). Accessed: 2024-09-01.

[17] Google. 2025. GeoFire for Javascript. https://github.com/firebase/geofire-js. (2025). Online. Accessed: April 2025.

[18] MongoDB Inc. 2025. MongoDB. https://www.mongodb.com/. (2025). Online. Accessed: April 2025.

[19] Mapbox. 2024. Map Mathcing API. https://docs.mapbox.com/api/navigation/map-matching/. (2024). Accessed: 2025-01-14.

[20] Mapbox. 2024. Mapbox Documentation. https://docs.mapbox.com/. (2024). Accessed: 2025-01-14.

[21] Google maps platform. 2024. Roads API. https://developers.google.com/maps/documentation/roads. (2024). Accessed: 2025-01-14.

[22] MapsPeople. 2024. MapsIndoors for Malls. https://www.mapspeople.com/industries/malls. (2024). Accessed: 2024-09-17.

[23] Paul V Mockapetris. 1987. RFC1034: DOMAIN names – Implementation and Specification. (1987).

[24] Niantic. 2024. Niantic Lightship – Locate. https://www.nianticspatial.com/locate. (2024). Accessed: 2025-01-14.

[25] NIST. 2024. Indoor Localization at NIST. https://www.nist.gov/ctl/pscr/indoor-localization-nist. (2024). Accessed: 2024-09-17.

[26] OpenStreetMap. 2024. List of OSM-based services. https://wiki.openstreetmap.org/wiki/List_of_OSM-based_services. (2024). Accessed: 2025-01-14.

[27] OpenStreetMap. 2024. OpenStreetMap Elements. https://wiki.openstreetmap.org/wiki/Elements. (2024). Accessed: 2025-01-14.

[28] OpenStreetMap. 2025. OpenStreetMap. https://www.openstreetmap.org. (2025). Online. Accessed: April 2025.

[29] PostGIS and OSGeo. 2025. PostGIS. https://postgis.net/. (2025). Online. Accessed: April 2025.

[30] Sebastian Thrun, Woflram Burgard, and Dieter Fox. 2006. *Probabilistic Robotics*. MIT Press, Cambridge, Masachussets, London, England, Chapter 10, 309–334.

[31] Uber. 2024. H3. https://h3geo.org/. (2024). Accessed: 2024-09-01.

[32] Wikipedia. 2024. RTK GNSS. https://en.wikipedia.org/wiki/Real-time_kinematic_positioning. (2024). Accessed: 2024-09-01.

[33] Wikipedia. 2024. Total Station. https://en.wikipedia.org/wiki/Total_station. (2024). Accessed: 2024-09-01.