Augmenting geospatial search with micro-terrain detail

Kent O'Sullivan
osullik@umd.edu
University of Maryland
USA

ABSTRACT

Humans spend a lot of time searching for things. With the advent of tools like google maps and open street maps, people can search through geospatial data at a whim. These tools focus on providing exact matches to queries or a list of candidate locations based on the user's query. Frequently, searchers only have access to partial information. Whether it has been a long time since visiting a location, they have a vague recommendation from a friend or are an investigator trying to identify a location to solve a crime- a common problem is how to find a location of interest based on partial information. This project designs *the* ***G****eospatially* ***E****nhanced* ***S****earch with* ***T****errain* ***A****ugmented* ***L****ocation* ***T****argeting (****GESTALT****)*, and implements a proof-of-concept of the proposed architecture. Based on a new best-case dataset developed for this project, *The Swan Valley Wineries dataset*, demonstrates the functionality and utility of *GESTALT* while identifying substantial opportunities for future work.

ACM Reference Format:
Kent O'Sullivan. 2023. Augmenting geospatial search with micro-terrain detail . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Geographic information systems (GIS) provide users with a means to efficiently search over spatial data given certain key pieces of information, like the coordinates or exact name of a location of interest. However, current GIS capabilities do not enable users to easily search for locations about which they have imperfect or incomplete information. From psychology and neuroscience research about how humans develop mental models of terrain [**? ? ?**], we know that people tend to anchor memories of a location around its visible objects and landmarks, using those to find that location again in the future. For example, a user may remember a location by a bus stop, a uniquely designed building, and a brightly colored sign, but fail to recall its exact address or physical coordinates. In these cases, GIS tools may help with narrowing down to the general region of interest, but a manual *last-mile* search must then be performed by the user to find the exact location of interest within that region. This last-mile search typically involves the visual inspection of remote sensing imagery data or street-view images to identify distinct landmarks or terrain features that match the partial information known about the location. This step of the search process is a bottleneck, as it encumbers the user with the burden of sifting through many possible candidate locations until the correct one is visually identified.

Taking inspiration from the way humans recall and search for information [**?**], we present*GESTALT*, an end-to-end pipeline for extracting geospatial data, transforming it into coherent object-location relations, storing those relations, and searching over them. We address the geospatial object ownership assignment inference task a modified clustering assignment algorithm and contribute a new gold standard *Swan Valley Wineries* dataset and a proof of concept implementation of the proposed architecture.

The rest of this paper is organized as follows. In section 2 we describe the process used to generate the gold standard wineries dataset. Next, in section 3, we define the *GESTALT* architecture and discuss how each subsystem contributes to our human-centric approach to automating the last-mile search. Then, we present implementation details in section 4 and a detailed comparison of object ownership assignment methods in section 5. Finally, we summarize related work in section 6 and conclude by identifying future research directions in section 7.

This paper will explore the sourcing and creation of datasets for the *GESTALT* project in section 2. It will then explore the architecture and define the goals for each subsystem in section 3 before exploring the implementation in section 4. A detailed comparison of Object ownership assignment methods is in section 5 before exploring related work in section 6. The paper concludes by identifying future work in section 7.

2 DATASETS

There are two critical datasets required for the GESTALT project. A list of all *objects* with their coordinates and a list of all *locations* with their coordinates. Here, an *object* is any physical entity. For example, a *tree, building, lake, bridge, gate* or *sign* could be objects. *Objects* can also have attributes that provide amplifying information about the object, including things like *colour, material, size, species etc.*. Locations are physical entities that *do* something. They are the meaningful grouping of objects determined by ownership, proximity or utility. They have some purpose other than being an object. They could be a *business, attraction, property etc.*.

2.1 Data Sources

Before delving into what data is needed, quickly reviewing the available data sources will highlight opportunities.

2.1.1 Google Maps. Google Maps[1] is a suite of applications maintained by Google for users to *"explore and navigate* [their] *world"*.

[1]Google Maps

While it is open for community contributions of locations, it is a commercial platform that charges for API access[2]. In addition to coordinating information, Google Maps contains Street View imagery, user reviews, and user-submitted pictures.

2.1.2 Open Street Maps (OSM). The open-source alternative to Google Maps, *Open Street Maps (OSM)* is a *"knowledge collective that creates Open Geodata as its main objective"*[?]. The data already contained within OSM is open-source, and the only requirement for adding data is registering with a user account. Their online editing interface makes adding data easy. The primary concern with OSM is the accuracy and validity of crowd-sourced data [?]. However, a combination of human review and an application of machine learning for detecting anomalous behaviour in OSM edits [?] goes a long way toward addressing these concerns.

2.2 Objects

For *GESTALT* to scale, it needs access to the coordinates of all *objects* within an area of interest. Google Maps does not record *objects* unless they are monuments or landmarks of significance. OSM records objects' coordinates as a mixture of point coordinates and bounding polygons. OSM has a defined and curated ontology that defines the labelling scheme, maximizing interoperability. The OSM objects are crowd-sourced and of varying granularity and completeness. Incompleteness is part of the initial scope of OSM, with the founder noting that it's typically only what people want to add that gets added [?]. In general, the completeness of OSM is unassured, so scaling *GESTALT* beyond the trivial requires an automated method for object detection and resolution. Because of the lack of completeness and to enable evaluation with ground-truth labels, the Author used Google Earth to annotate objects present at six wineries within the Swan Valley Region of Western Australia. The wineries are separated in space and a semi-rural environment. The tags consist of an object name, its latitude & longitude, and any descriptive markings written as key:value pairs. The objects are stored by their ground truth parent location in a KML file. Additional dataset creation for the small New South Wales town of Buladelah aims to fill a 'suburban' setting and of the Darby St Restaurant Strip in Newcastle, New South Wales, for an 'urban' environment.

2.3 Locations

For *GESTALT* to scale, it needs access to the coordinates of all *locations* within an area of interest. Google Maps maintains *locations* as coordinate points with associated metadata. The locations are generally current and complete. Google Maps does not support bounding polygons at the location level; it appears to extend to as granular as ZIP Codes or suburb boundaries and no further. OSM Supports locations, but is less complete than Google Maps (at least for Australian Wine Regions.) In general, for *GESTALT* to function optimally, the input locations should be the union of Google Maps and OSM. However, given the limitations of Google Maps API usage, a dataset was manually curated in OSM using publicly available information and the Author's world knowledge. Creating the Swan Valley Winery dataset for this project has the benefit of yielding 31 additional nodes and associated metadata for the OSM project.

[2]Pricing for Google Maps API

2.4 Automation

Automating the labelling process is essential to scaling *GESTALT* beyond a trivial size. Options for automation are explored in detail in sections 3 and 6, but essentially rely on combining remote sensing imagery, ground-based imagery and image metadata to generate mappings of objects to coordinates and parent locations. The ability to autonomously determine object locations will set the conditions for the remaining elements of GESTALT to scale.

3 ARCHITECTURE

The architecture of *GESTALT* is in Figure **??**. It decomposes into four essential functions: data acquisition, ownership assignment, concept mapping and search.

3.1 Data Acquisition

The data acquisition component of *GESTALT* begins with a visual encoding of the world. The visual encoding is primarily remote sensing imagery providing a top-down view of the earth's surface, but it also includes street-view imagery and other photographs. The system collects two types of information from this imagery, *locations* and *objects*. These two information types are discussed in section 2. Briefly recapping, objects are any physical thing in the world, and locations are the specific uses of a place that usually contains collections of objects.

The vision for a mature data collection system enables the autonomous collection of location and object data. The collection system will source location data from open-access systems like OSM and relies on crowd-sourced information. Businesses, attractions and other higher-level 'locations' are likely to be annotated by the open-source community or business owners themselves. Objects, the core of *GESTALT*, are much less likely to be annotated. Few people have the patience to manually tag the geolocation of apparently inconsequential things like trees, statues, fountains and telegraph poles. An automated solution aims to leverage publicly available remote sensing imagery data (Bing Maps Satellite data, for example) and public streetview and photo contributions to automatically identify objects, geo-locate them and add those tags to a database.

The design for this subsystem breaks maps into small geographically-bounded chunks (approximately the size of a 'location'). It will use remote-sensing imagery to create a grid of objects / not-object. It will retrieve ground-level imagery within and adjacent to that box. The first challenge is classifying an image as 'indoors' (where no objects will be visible from RSI and the closest building will 'own' it) or 'outdoors', where objects can map to the RSI. Numerous approaches exist to the indoor/outdoor scene classification [?]. Each object in an outdoor photo's distance from the camera geolocation will be estimated using a myriad of depth estimation techniques [? ?]. Where multiple images cover the same area from different perspectives, the composite of these images will be used to estimate the positions of objects, as has been shown in prior work like IM2GPS from Carnegie Mellon University [?] and numerous other efforts over internet-available images [?]. As discussed in the

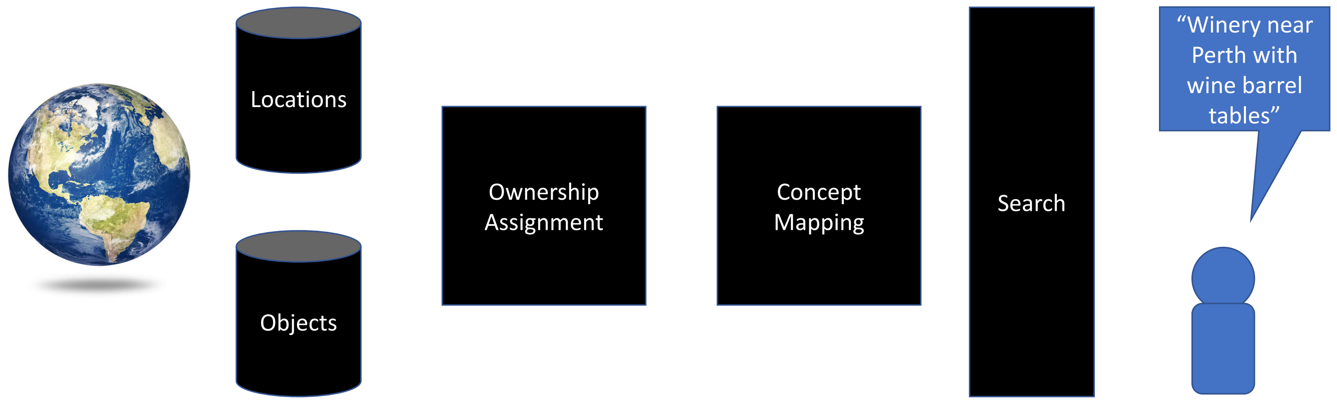


Figure 1: The architecture of *GESTALT* consists of the data collection subsystem, the ownership assignment process, the concept mapping process and the search subsystem.

following sub-section, some errors are permissible here and being 'close enough' is good enough as a starting point for the following systems.

3.2 Ownership Assignment

Ownership assignment is the unsupervised process through which objects are associated with locations. Objects need to be associated with locations in *GESTALT* because for the *concept mapping* process and *search* subsystem to work, they need to know which objects belong to each location. For example, assuming two adjacent wineries, a fountain between them would be west of one but east of the other. The mapping will be incorrect unless it is clear which winery it belongs to. Similarly, for search, the underlying idea for *GESTALT* is that people will remember particular objects at locations and use them as clues to find them again. Without an accurate object-to-location assignment, the search functionality will not work. The ownership assignment process needs to be unsupervised to enable scaling. Aspirationally, *GESTALT* will index the world's objects to allow searchers to find any location with *GESTALT*. Processing a world's worth of data necessitates an unsupervised approach.

The Ownership Assignment process accepts two inputs, a collection of locations with their coordinates and a collection of objects with their coordinates. The process works to assign each object to its parent location. The process ends when objects are mapped to their parent locations. Of note, because the human eye can see over property boundaries and other invisible lines on maps, we can accept a small margin of error where objects from neighbouring locations might be mislabelled. For example, perhaps there is a large red shed at the back of a location's property that is not visible to the main part of the parent location but is clearly visible to the neighbouring location. There will also exist some objects which plausibly could be seen and remembered by patrons of several locations, for example, a lake or a giant statue. This multiple-ownership situation is one of the driving requirements for implementing concept mapping to extract additional discriminatory information between locations based on the geospatial layout of objects.

3.3 Concept Mapping

Concept mapping is the process of determining the geospatial relations between objects. Much information is implicit in the relative positioning of objects within a location. For *GESTALT*, there are three types of relations. The first is *Static Cardinal Relations* which encodes whether an object is North, South, East or West of a location. Static Cardinal Relations support simple queries where the user knows that a location has a lake on its western side. The second is *Dynamic Cardinal Relations* which determines whether an object is North, South, East or West of another object within a location. These queries support cases where a searcher might remember standing at a lake northeast of a location and that there was a swingset to the immediate west of them but still in the northeast of the location overall. The third are *Positional Relations* which are applied to the two other types to enable reasoning about objects that are left, right, up, down, beside, behind etc., other objects. Positional relation use will be extensive because few humans think in cardinal directions, and most spatial reasoning is conducted from the person's perspective. Positional Relations enable users to query for locations where there is a letterbox on the left of the driveway as you enter the driveway while the house is in front of you.

Concept mapping needs to be unsupervised and support aggregation. Tracking every object's relative location to every other object quickly becomes intractable, so mechanisms to aggregate depending on the level of granularity need to be applied. Accordingly, the underlying data structure must support aggregation and relative position querying.

3.4 Search

The core function of *GESTALT* is searching. The search function assumes that the searcher only has partial information about a location. There are two elements of partial information. First is a general idea of the region in which the location occurs. Here region means the area surrounding a location. For example, in searching for a winery, it is assumed that the searcher knows that they are in the swan valley region of Western Australia. A region could be an administrative boundary like a city, suburb, or general geographic area. Either way, we assume that the searcher can prune their search space to the commencement of the *last-mile* search before using *GESTALT*. The second assumes that the searcher knows a subset of the objects associated with a location. They may or may not know any of the attributes of those objects (for example, material, colour etc).

The search problem can then be framed in several ways. A *set membership problem* is the most straightforward and most efficient. Given a set of locations, each of which has a set of objects it 'owns' and a set of objects in the search term, which locations have complete coverage of the search set. Bloom Filters are the obvious choice of data structure to support this search method. A limitation of using bloom filters is that if the user has little information to discriminate locations, almost any location will satisfy a query. For example, searching "tree" would return every winery in the Swan Valley region. A second limitation is the lack of support for aggregation. Searching for 'tree' might yield nothing, but searching for '30 trees' would considerably prune the result.

The second approach to search incorporates the concept of mapping and becomes a spatial search with a specific method depending on the objects' underlying data structure. The general case is as follows: Given a set of locations with geospatial mappings of their child objects and a subset of those geospatial mappings of child objects which location does that subset match? If using a graph structure, each location's objects become a graph where the objects are nodes and the geospatial relations are edges encoding the spatial relations (e.g. west of, north of). It is a subgraph matching problem, which is, of course, NP-Complete. Alternately, representing each location's objects as a KD-Tree rooted on the centroid of the object cluster would allow for dynamic searching. For example, assuming an initial split on the longitude of objects, we could immediately tell that all objects in the left subtree would be west of that root. For either of these geospatial approaches, a translation layer from the positional relational to cardinal relational will need to occur.

Regardless of the formulation of the search problem, there is a clear requirement for semantic search across objects. For simple spelling variations (e.g. 'colour' in the King's Australian English versus 'color' in American English), a string distance metric like *Levenshtein* distance would suffice. But a richer semantic search is required for more pronounced linguistic variations like 'water fountain' versus 'drinking fountain' versus 'bubbler'. The first option to reduce the likelihood of inconsistently named objects is to enforce compliance with the Open Street Maps ontology, which is an extensive definition of locations, objects and their descriptions. While adherence to the ontology enforces internal consistency, it does not overcome the issue of a user searching with unknown terms. A straightforward option could be to use the vector embedding of a word as a starting point and use the k-nearest words in vector space as alternate search terms. It is unlikely that this will significantly impact the false positive rate once an appropriate similarity threshold is set but may increase the system's overall recall. A more complicated approach could leverage an external semantic data source like DBPedia or WikiData, or even WordNet to search for semantically similar terms to substitute in the search.

The search problem must balance precision and recall while not being computationally intractable. An effective search process will use bloom filters to prune the search space for the more complicated geospatial search. Semantic enrichment should be applied independently at each stage of the search in an attempt to improve the recall of *GESTALT*.

3.5 Sumamry of Architecture

The Architecture of *GESTALT* is designed to be lightweight and modular. The core requirement is to improve the ability to find locations of interest based on partial information. The search subsystem needs to balance precision in reducing the number of candidate locations with maximising the recall of possible candidate locations. The recall is prioritised. The search space should be pruned with set membership checks based on the intuition that there is no point in running an expensive geospatial query over a location that doesn't contain the objects in question. Bloom filter checks are cheap; the human eye doesn't see invisible lines on a map. Accordingly, the ownership assignment subsystem can be inexact, and objects should be 'shared' between locations where appropriate. That location sharing in membership assignments improves the recall of the system. As a corollary, the subsequent spatial search process maintains the system's precision using the concept mappings of objects to extract implicit information about the location. Underlying the search problem requires collecting and processing objects and locations autonomously, at scale. Collecting and processing objects and locations is the first stage explored in section 4, implementation.

4 IMPLEMENTATION

The implementation section of this report discusses the engineering work conducted so far to implement the *GESTALT* architecture and reviews high-level results. Specific experimental results are in section 5.

4.1 Data Collection

Due to the complexity of the task and significant concurrent work in the area, the development of automated object detection and geolocation was scoped out. Instead, the author developed a small dataset manually, as discussed in section **??**. The data extraction, cleaning and loading are implemented in Python in two parts, the *KML parser* for object extraction and the *Open Street Maps* query interface for location retrieval. The KML parser leverages the *fastKML*[3] and ingests a KML file divided by region (where each region is a bounding box covering an arbitrary number of locations). Within each region (for this test dataset), each location is separated, with its objects stored as its children. Attributes of the objects (e.g. colour, size, material) are recorded in the comments field as key:value pairs. The KML Parser extracts the objects into dictionaries organised by location before exporting the files as JSON for future analysis.

The OSM query interface leverages the *OSMPythonTools*[4]. It passes a bounding box to the OSM Overpass-Turbo API[5] and requests the relevant location nodes in the area. The results are arranged into a dictionary and exported as JSON for further analysis.

Implemented so far is the translation from inputs of objects and locations, in different formats, to a standard JSON format. Not yet implemented is the automatic detection and geolocation of objects.

4.2 Ownership Assignment

Ownership assignment is implemented in Python in two ways. The first is the trivial implementation, where the location returned from the OSM query is a bounding polygon. In this case, if the point lies within the minimal enclosing rectangle of the polygon, it is added as a 'member' of that location. The second, more common (and more challenging approach) formulates an unsupervised learning problem using clustering libraries from *scikit-learn*[6]. Given a collection of objects and a collection of locations within a bounding box region, clustering assigns each object to its 'parent' location. Under the assumption that the number of locations equals the number of clusters, K-Means clustering proved to be the most effective approach. After clustering the objects, we determine the centroid of the object cluster. Given a KD tree constructed from location point coordinates, a nearest neighbour search on the KD tree with a query parameter of the object centroid yields the nearest location and is assigned ownership of that cluster A detailed performance comparison is in section 5.

Overall, initial proof-of-concept clustering uses K-Means and DBSCAN clustering. An optimal solution to the ownership assignment problem is an exciting and unusual clustering problem. Assuming that the collection of locations is complete and that the point coordinate of the location is central to the collection of objects that belong to that location, the clustering problem is the assignment of an arbitrary number of objects to any of a set of possible centroids. Not every centroid will have objects, and objects are not uniformly distributed. Initial investigation into the DVBSCAN algorithm Ram et al. proposed in 2010 presents a promising direction to resolve this problem.[**?**]

Overall, the Ownership Assignment process needs to produce a data structure that will permit set membership checking for search and set the conditions for the concept mapping process. It is the most complete component of *GESTALT*. The choice of clustering algorithm needs refinement, and location-based bloom filters need to be implemented to support efficient search, but it is otherwise functional.

4.3 Concept Mapping

Concept mapping has been partially implemented in Python leveraging the *Scipy* library[7]. Two different approaches have been trialled. The first is simple dynamic arithmetic on the coordinates stored in a Pandas data frame. If one set of coordinates is above, below, left or right of another, it is north, south, east or west, respectively. While these calculations are in constant time for straightforward comparisons of known objects (e.g. "is the pond west of the bridge"), the time complexity rapidly increases as soon as aggregations are employed. Queries of "Give me everything west of the duck pond" would execute in $O(N)$ time as each element has to be examined. Worst-case queries would run in $O(N \text{ sup } 2)$ time, where every object is checked for its position relative to every other object.

The second (better) approach (only partially implemented) instantiates the objects within a location into a KD-Tree. Assuming that the object centroid is the root, we can quickly complete queries like "Give me everything west of the duck pond" by leveraging the structure of the subtrees to return the requested set. Similarly, getting the relative positions of two objects searches for a common ancestor. It uses the path between the children and their ancestor node to infer their spatial relation to each other.

A third approach, designed to leverage the *Neo4J Python Library*[8] to connect to a *Neo4J Graph Database*[9] but not implemented frames concept mapping as a graph traversal problem. In this formulation, each object is a node on a graph. Weighted, labelled edges exist between each node within a given proximity threshold to the node. The edge labels describe the neighbouring node's cardinal direction and the distance weights. After constructing the object graph, queries for 'give me everything west of the duck pond' would freely explore nodes connected by west, north and south edges. It can only traverse along an east edge so long as the total cost of travelling east would be less than the cumulative value of the 'west' travel up to this point.

Overall, concept mapping aims to enable geographic search over objects by explicitly representing their geospatial relationships to each other. The author implemented a very basic approach using coordinate arithmetic was quickly determined to be infeasible for the extensive data sets that *GESTALT* anticipates processing. KD-Trees for the objects in each location have been implemented, as have the KD-Trees for the locations themselves. This conceptual KD-Tree of KD-Trees approach performs a natural aggregation function which, provided that regions are created consistently, will allow for relative spatial queries at different levels of granularity.

[3]Fast KML PyPI Repo
[4]OSMPythonTools PyPI Repo
[5]Overpass-Turbo API
[6]Scikit-Learn PyPI Repo

[7]SciPy PyPI Repo
[8]Neo4J PyPI Repo
[9]Neo4J Website

Empirical evaluation of the performance of the arithmetic, KD-Tree and Graph-based approaches is yet to be completed.

4.4 Search

The search function has been implemented using the Python *Pandas*[10]Pandas PyPI Repo library. This approach assumes a single data frame of objects and their determined locations because of the number of possible attributes an object can have and the relatively few that they possess, this is a sparse data structure. The sparseness does indicate the discriminatory power of remembering attributes. For example, a 'door' is not informative, but a 'blue door' on your favourite seaside restaurant is more likely to prune the search space. Because *GESTALT* is designed only for the last-mile search and assumes a small starting region, it may remain feasible to use a simple data structure like a Pandas data frame containing all the objects for all the locations for the query region. More work with the aggregation functions is required to determine if it can support all the necessary aggregation queries comparing object collections.

Semantic search has not been implemented. However, the Levenshtein string distance metric (with $threshold = 0.8$) checks for small spelling discrepancies in input words. The priority weights towards retrieving all possible objects, so we accept the increased risk of mistakenly including an object to move the recall closer to 100%. The next component to be implemented is a nearest-neighbour retrieval mechanism using word embeddings. Prior work indicates that developing databases of embeddings is trivial[**?**], but using existing datasets tools like word2vec, GloVe and fasttext can generate embeddings over large, publicly available corpora that can be recreated.

As discussed in the subsection on Ownership Assignment implementation, bloom filters are a much more efficient operator for set membership testing. The KD-Tree is more suited for geospatial queries, so the Pandas Dataframe currently supports the gaps between the two in supporting aggregation queries. More work is required to integrate these data structures into a coherent search pipeline that maximises recall while actively pruning the search space at every step so that the searcher can find their locations of interest. Natural language querying is an active area of research yet to present a solution capable of effectively translating natural language queries and their SQL solutions. Given the relatively constrained domain of this problem set, it is a good candidate for implementation as a low priority for improvement.

4.5 Summary of Implemenation

The implementation of *GESTALT* is incomplete, but initial work demonstrates its feasibility. The priority for future work is to implement concept mapping fully, add bloom filters to locations, and implement semantic similarity searching. Improving the query interface is another 'easy win'. The work that will unlock the 'real-world' potential of *GESTALT* is the automation of object geolocation. Automating data collection is also the most challenging part of the problem and will require substantial work.

[10]https://pypi.org/project/pandas/

5 RESULTS

The results section discusses the empirical analysis of the clustering methods tested during the implementation of the Ownership Assignment process. The experiment design is simple and designed to provide an upper bound for performance on an optimal dataset with no noise (the Swan Valley wineries dataset.) The Swan Valley Wineries dataset consists of 31 wineries retrieved from OSM and $\tilde{1}50$ objects hand-labelled across 6 of those 31 locations with ground-truth location labels. The objective of the experiment was to see which clustering method most accurately predicts the location that the objects 'belong' to. Here accuracy is measured in two dimensions. First: the predicted location / true location label match. Second, the creation of the correct number of clusters (6). Testing of K-Means and DBSCAN occurs under optimal, realistic and worst-case parameter conditions. For both DBScan and K-Means, the location is inferred to be the location coordinate closest to the cluster centroid.

5.1 K-Means

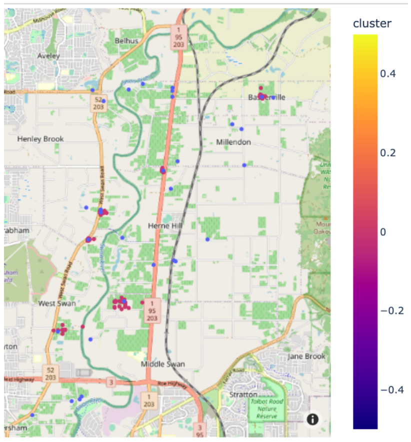
K-Means clustering accepts the input of a collection of object coordinates and a parameter K of the number of clusters to create. Optimal conditions assume the number of clusters is known ($K = 6$). Realistic conditions assume the number of clusters equals the number of locations ($K = 31$), and worst-case conditions assume only one cluster ($K = 1$).

Unsurprisingly, the optimal situation performed best with all objects assigned to their correct clusters and all clusters assigned to their correct label. One reported misclassification was determined to be a labelling error in the training data. Surprisingly, under 'realistic' conditions, though the number of clusters is far more than there should be, it correctly assigns the ownership of almost all objects with only five incorrectly labelled. Under the worst-case conditions, it only creates a single cluster and assigns all objects to the (same) incorrect location.

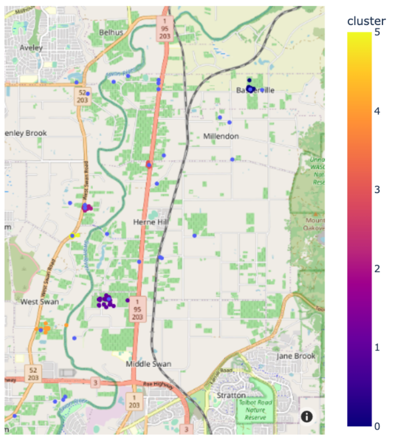
5.2 DBSCAN

The DBSCAN algorithm was introduced in 1996 by Ester et al. [**?**]. It is a clustering algorithm that accepts the input of a collection of object coordinates and two parameters ϵ, the distance permitted between coordinates before they transition to a different cluster and N the number of coordinates required to be within ϵ of each other to form a cluster. A 2017 paper by Schubert et al. [**?**] provides useful guidance on turning these parameters, and their experiments reveal that the value of ϵ is much more sensitive than the value of N. Based on their guidance, optimal conditions would see $\epsilon \leftarrow (2xdim) - 1 = 3metres$. Their guidance further notes that domain knowledge should be used where appropriate, so here we set ($\epsilon = 10metres$). Realistic sets ($\epsilon = 100metres$) and worst case sets ($\epsilon = 1000metres$). $N = 3$ in all tests.

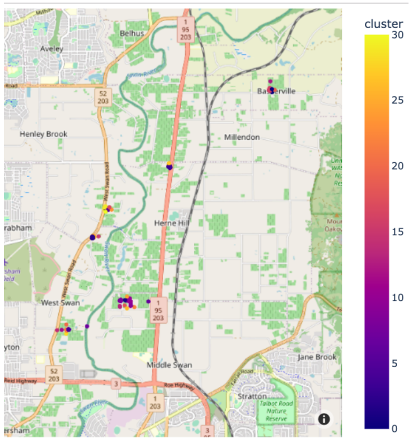
Discarding objects is problematic as it degrades the recall of objects that are distant from other objects. The experiments indicate that when ϵ values are small, more of the data is regarded as 'noise' and disregarded by the clustering algorithm. With $\epsilon = 10m$, it forms 6 clusters, splitting *Oakover Grounds* in half and discarding 12 objects of the 150 as noise. Conversely, as ϵ increases, clusters



(a) In the worst case of $K = 1$, all objects are incorrectly assigned to the central location of the region.

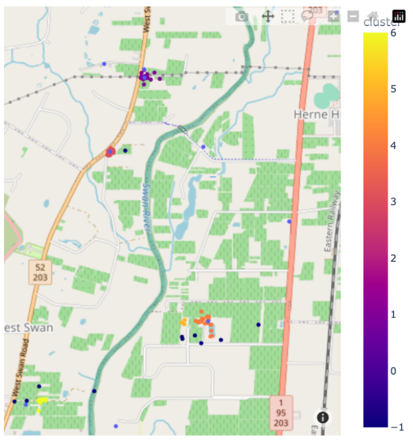


(b) With optimal conditions $K = 6$, the clustering performs ideally, while locations are sparse across the region.

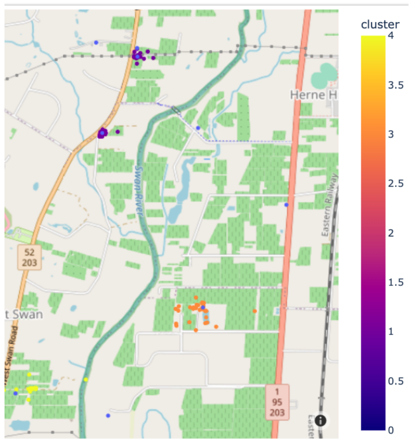


(c) In realistic settings where $K = NumberofLocations$, it clusters incorrectly but assigns the correct owners, which is unlikely to hold when location density increases

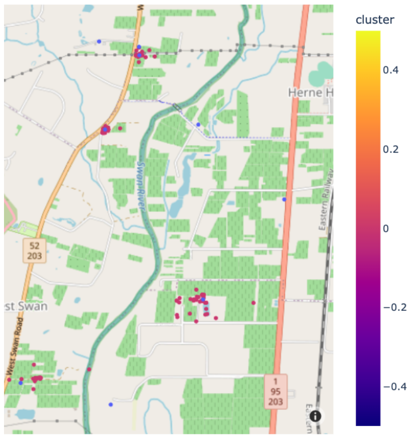
Figure 2: K-Means Performance. In regions with sparse locations K-Means performs well even if too many clusters exist; this is unlikely to be true in dense regions.



(a) In the optimal case of $\epsilon = 10$ 12 objects are incorrectly pruned as 'noise' (cluster = -1).



(b) With realistic conditions $\epsilon = 100$ the *ugly duckling wines* and *Little River Winery* merge because the locations are too close.



(c) In worst-case settings where $\epsilon = 1000$ (too large), it groups all objects into a single cluster and incorrectly assigns to the central location of the region

Figure 3: DBSCAN Performance. DBSCAN performs suboptimally when locations are close together and when ϵ is too small or too large

rapidly begin to merge. Despite being distinct locations, the location merging problem is evident in the Winery Dataset when *Ugly Duckling Wines* and *Little River Winery and Cafe* combine. The worst case, where ϵ is set too large, merges all objects into a single, average location. The worst-case performance of DBSCAN matches the worst-case performance of K-Means with all objects belonging to the *Siteall a* winery.

5.3 Error Analysis

Examining the errors reveals the following insights about each clustering technique.

When K-Means clusters incorrectly, labels are still correct. When K exceeds the number of clusters, it fragments the actual clusters. However, in ideal conditions like the Wineries Dataset,

Algorithm	Variant	Num Clusters	Accuracy
K-Means	$K = 1$	1	0/146
	$K = 6$	6	146/146
	$K = 31$	31	141/146
DBSCAN	$\epsilon = 10m, N = 3$	7 (+ 12 'noise')	134/146
	$\epsilon = 100m, N = 3$	7 (+ 0 'noise')	126/146
	$\epsilon = 1000m, N = 3$	1 (+ 0 'noise')	0/146

Table 1: K-Means with perfect information performs best. DBSCAN handles dense inter-location clusters and variance in intra-object cluster density poorly

where locations are separated, the centroids of these cluster fragments are still closest to the correct location. As a consequence, they are correctly labelled despite being incorrectly clustered. We expect the accuracy will drop when locations are more densely packed. However, when we aim to process all objects and locations in a region concurrently, the number of clusters will likely approach the number of locations, and the issue will be less pronounced.

DBSCAN excludes outlying examples. To prevent all locations in a region from being merged, a small ϵ is better. However, a small epsilon increases the number of points determined to be 'noise' and hence are not added to any cluster or provided with a label. The failure of DBSCAN to achieve 100% recall of objects is problematic, as *GESTALT* needs the maximal number of objects to be associated with candidate locations. Making ϵ larger, however, results in locations merging. As discussed in section 4, the DVBSCAN algorithm is a promising approach that will improve the ability to use a larger ϵ to improve recall without merging adjacent clusters.

5.4 Summary of Experimental Results

. The experiments reveal that small changes in the parameters of both K-Means and DBSCAN dramatically impact the output. K-Means performed the best under optimal conditions and better than DBSCAN under 'realistic' conditions. However, in datasets where locations have a higher density in the region, the benefit is expected to level, and so experimentation with an algorithm that can accept as a parameter the list of locations to use as clustering centroids seeks to overcome this issue in dense localities. DVBSCAN is the most promising avenue for implementing the improved ownership assignment.

6 RELATED WORK

On 08 May 2023 the Open Source Intelligence platform *Bellingcat*[11] released their own version of *GESTALT*, *Bellingcat OSM Search*[12] [?]. The blog post accompanying the release highlights the importance of pruning possible search space using objects known to be present. In their use-case, they start with a photo and attempt to geolocate it by performing the *Last-Mile* search based on the presence of OSM Map Features. Their system implements a more user-friendly interface over the OSM Overpass-Turbo API, allowing searchers to use dropdown lists and sliders rather than generating complicated queries. Their tool demonstrates the importance and utility of the problem that *GESTALT* seeks to solve. It also shows how much data is already contained on OSM that can be leveraged by GESTALT. The obvious limitation is in performance. As it directly queries the OSM API, the query executes very slowly. For example, running the query to return all wineries in the swan valley region took 21 seconds to complete. In *GESTALT*, it takes less than 1 second. Their system does not address data collection, ownership assignment or concept mapping. It also doesn't allow for spatial queries expected by GESTALT searchers. The Bellingcat tool demonstrates the utility of a system like *GESTALT* and highlights how careful consideration of data structure design is essential for developing a performant system.

[11]Bellingcat Website
[12]Bellingcat OSM Search Tool

6.1 Psychology of Geospatial Reasoning

Several ideas from criminology, psychology and neuroscience drive the underlying notion of *GESTALT* that people remember objects and anchor on those 'things' that they see while experiencing a location to find it again.

GeoGuessr. A popular online game called *Geoguessr*[13] demonstrates that for many people, figuring out where they are in the world can be a source of much joy. A 2023 journal article analyzing the strategies employed by a top player identifies several successful strategies that are used by a leading player [?]. The Geoguessr problem is a distinct and, in many ways, reciprocal problem to the *last-mile* search problem *GESTALT* seeks to solve. While the last mile search assumes a general region is known, the geoguessr frequently has no idea where in the world they are and needs to use clues from the interface (powered by google street view) to determine the country, state, county etc., that they are in. While not directly relevant, Geoguessr demonstrates the importance of searching for landmarks, and often even benign objects like bus stops, in locating where in the world an image is.

Winthropping. The Winthrop Method is a geospatial search method developed by Captain Winthrop of the Royal Engineers for use in Northern Ireland in the 1970s[?] to detect clandestine weapons caches and concealed improvised explosive devices. The underlying logic is that to find something, a human has to have some method of navigating to it and that the objects in our environment help to form mental models of the terrain, we can use to navigate by. A popular example is the closing scene of the film *The Shawshank Redemption* where the protagonist Andy Dufrense provides instructions to his fellow prisoner Ellis Redding on how to find a gift left for him *"It's got a long rock wall with a big oak tree at the north end... find that spot. At the base of that wall, you'll find a ...piece of black, volcanic glass. There's some thing buried under it I want you to have."* The two reasons that Winthropping works are *Affordance* and *Satisficing*. Affordance refers to the interaction of an agent with its environment and, in simple terms, means that particular objects will have a more significant impact on us and remain in our memory than others. When combined with the idea of satisficing, in which an agent makes a satisfactory or sufficient but possibly sub-optimal decision, it illustrates the value of object-based search. We cannot remember every detail of a location, so our brains will record only a few key objects or experiences for

[13]Geoguessr Website

us to leverage. A 2013 Neuroscience study shows that when we revisit those locations, the objects we remember serve as keys to other memories of that location[?], *GESTALT* aims to exploit these geospatial aspects of memory for helping searchers to find the locations they are looking for.

6.2 Remote Sensing Imagery

Relevant to the data collection subsystem if *GESTALT*, 2018 efforts to improve the state of the art in object detection from remote sensing imagery focused on developing datasets for training and evaluating models. The xView project supports the detection of 60 classes of objects[?] using horizontal bounding boxes. The DOTA project supports a much more modest 20 classes[?]. Both focus towards shipping and industrial applications and so will not generalize well. The 2021 update to the DOTA project highlighted that remote sensing object detection continues to suffer from the arbitrary rotation of objects and the vast disparities in the clustering of objects[?]. DOTA version 2 tries to address these issues by employing orientation bounding boxes but is still very constrained in the classes of objects it supports. A separate effort by Li et al. in 2020 can detect objects less constrained to heavy industry but only accounts for 20 or so objects[?]. Overall current work on remote-sensing object detection indicates that it is still an emerging field incapable of supporting the labelling of micro-terrain features required for the proposed system. An area growing parallel with remote sensing object detection is image captioning and visual question answering. In addition to implementing previously discussed object detection techniques, they employ alternate data sources to augment their ability to provide answers to natural language questions about remote sensing imagery. In 2017 Shi and Zou demonstrated that it is possible to automate caption generation for remote sensing imagery. However, their experimentation showed it to be ineffective at tasks like counting objects[?], an essential requirement of micro-terrain analysis.

6.3 Visual Question Answering

The paper that initiated the domain of Remote Sensing Visual Question Answering (RSVQA) was published in 2020 by Lobry et al. and showed an excellent ability to answer direct questions about a given remote sensing image, including area estimates, object counts and determining the relative locations of objects[?]. These capabilities are all helpful in the concept mapping component of *GESTALT*. Importantly they also incorporated geospatial information from OpenStreetMap into their system. A fundamental limitation they identified is that the lack of information in OSM about specific micro-terrain features and the inability of object detection models to provide it presents a significant gap holding back the advancement of the RSVQA field. My work may contribute towards closing that gap. Later work by Zheng et al. and Yuan et al. highlight that RSVQA is very much in its infancy, and they focus on improving the underlying models used in RSVQA systems[? ?]. In addition to the gap identified by Lobry et al., one limitation is that the RSVQA approach focuses on answering questions about the things the user is already looking at. In my partial information use case, the user doesn't know exactly where to look, so using the RSVQA approach would render no improvement to performance over the visual inspection itself. The real challenge is identifying where to look, not what they are already looking at.

6.4 Geospatial Question Answering

Towards identifying where the user could be looking, the field of geospatial question answering, related to geospatial information retrieval, offers promising directions. In 2018 Punjani et al. sought to determine whether geospatial information could be incorporated into a question-answering system[?]. They used the established Frankenstein variant of the Qanary approach to developing question-answering pipelines to develop GeoQA. GeoQA can answer questions in several useful geospatial categories, including point, range, and property-based queries. They built their system on linked data collected from OpenStreetMaps and WikiData. More recently, the efforts to develop WorldKG, a geospatial knowledge graph of the world, extend the linked-data approach and allow users to generate SPARQL queries to answer complex geospatial questions across the fused knowledge of OpenStreetMaps, DBPedia and WikiData[?]. These linked data approaches to geospatial question answering are valuable advancements but do not include enough micro-terrain detail to satisfy the requirement of my project to allow a user to find a location based on partial information about the micro-terrain of the location.

6.5 Pictoral Querying

Prior work in pictoral querying shows the utility of identifying locations from maps based on knowledge of where a subset of locations on a given map are [?]. Their approach focuses on matching locations from a user-defined pictorial input to an underlying database of maps. In the 27 years since Soffer and Samet first specified the Pictoral Query Language, considerable advances in digital storage and access to geospatial data have addressed some of the initial scale limitations that the conversion of maps to digital pictorial representations presents. For example, the role of a system like MAGELLAN [?] in *GESTALT* is replaced by sourcing the locations from OSM, and the requirement for a separate system to efficiently index and query map tiles like MARCO [?] is now handled by OSM in its entirety. Additional work in pictorial querying notes the problem of searching when there are multiple instances of the objects. The problem is NP-Complete when formulated as a subgraph matching problem[?]. *GESTALT* addresses the complexity issues by limiting the size of the initial search space to the *last-mile* search over a region and by pruning results at each step, only checking locations that pass the set-membership test of the bloom filter, for example. The query interface is the most appealing part of the work in Pictoral Querying. Being able to specify queries pictorially, as a user's internal concept map, to *GESTALT*, is likely to improve user experience and leverage the benefits of human geospatial memory and will be considered for future work.

7 CONCLUSION

7.1 Future Work

Throughout this paper, we identify many avenues for future work. Sections 2 and 3 explain the requirement for large-scale pictorial to geospatial scene mapping to enable the large-scale identification of objects to fuel Gestalt's search. Section 2 highlights the need to develop datasets in dense suburban and urban locations to enable robust testing of the ownership assignment process. Sections 3 and 4 identify the requirement to trial the DVBSCAN algorithm to improve the ownership assignment process and the need to test the concept mapping proposed using KD-Trees robustly. Sections 4 and 6 emphasise the need for a user query interface. Work on pictoral querying offers an exciting direction that enables abstracted user querying and leverages the cognitive advantages of geospatially constructing their query. The search component of *GESTALT* needs to be tested at scale. While section 4 notes that the assumption of only a *last-mile* search allows us to assume small datasets, the performance of the belling tool discussed in section 6 shows how quickly performance degrades if not managed well. Finally, and most importantly, though the psychology literature indicates that the *GESTALT* approach should be helpful to a searcher, there is no work evaluating this theory and measuring the extent to which it is functional. A user study should be prioritised for a fully functional *GESTALT* prototype before it expands to full scale.

7.2 Conclusion

The *GESTALT* project aims to reduce the time a searcher spends on the *last-mile* of searching for a location. It assumes that the *last-mile* is in a constrained geographical region and allows users to search for objects they are likely to remember from candidate locations. *GESTALT* is designed to collect geospatial information about locations and objects within geographic regions from open-source geospatial and pictorial data. It infers the associations between objects visible to searchers in the real world and the location that they belong to and stores them using bloom filters and KD-Trees for efficient representation and search. *GESTALT* implements concept mapping to allow a user to query the implicit geospatial relations between objects in candidate locations, leveraging the inherent ordering of the multidimensional KD-Tree data structure for efficient search. *GESTALT* demonstrates at a trim level, on the Swan Valley wineries dataset, that the approach is feasible and identifies future work in scaling it.