GESTALT: Augmenting geospatial search with micro-terrain detail

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

Nicole Schneider[†]
nsch@umd.edu
University of Maryland
USA

Hanan Samet
hjs@cs.umd.edu
University of Maryland
USA

ABSTRACT

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. 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, which 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 *the **G**eospatially **E**nhanced **S**earch with **T**errain **A**ugmented **L**ocation **T**argeting (**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 under uncertainty constraints and contribute a new gold standard Swan Valley Wineries dataset and a proof of concept implementation of the proposed architecture.

ACM Reference Format:
Kent O'Sullivan[†], Nicole Schneider[†], and Hanan Samet. 2023. GESTALT: Augmenting geospatial search with micro-terrain detail . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 15 pages. https://doi.org/XXXXXXX.XXXXXXX

† Equal contribution by the authors.

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 cognitive maps of terrain for navigation and route planning [? ? ? ?] Kent: fill in this ref, we know that people tend to anchor memories of a location around its visible objects and landmarks, likely doing so hierarchically, or separately relating global and local features [?]. For example, a user may remember a location by a series of visual features encountered near it, like a large building, a bus stop, 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.

A more concrete version of the problem concerns geospatial and open-source intelligence analysis trying to identify locations from incomplete information. Traditionally, analysts will get to a *near-enough* start point and then begin the excrutiating manual review of remote sensing imagery, photography and other reports to try match the objects they know about to the location they are searching for in the geospatial configuration they are expecting. Recent related work from *Bellingcat* highlights that identifying locations based on collections of objects associated with them is a technique actively used by investigators. The investigative requirements are evidently great enough to necessitate Bellingcat developing an *OpenStreetMap Search Tool* to seaech for loations using objects[1]. Bellingcat's tool puts a user-friendly interface on the OSM Overpass-Turbo Interface[2] Our testing indicates a rapid degradation in query execution time as scale increases, and is distinct from our work in that it only queries OSM data, only checks for set intersections of objects within a given proximity threshold of each other and does not allow for pictorial querying. *GESTALT* approaches the same problem in a more comprehensive way, aiming to help investigators with their *last-mile* search to identify locations of interest based on the objects that they know (or suspect) are associated with the location.

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.

[1]https://www.bellingcat.com/resources/how-tos/2023/05/08/finding-geolocation-leads-with-bellingcats-openstreetmap-search-tool/
[2]https://overpass-turbo.eu/

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Specifically, *GESTALT* provides the following functionality:

(1) Multiple methods for ingesting location and object tags, including doing so automatically using computer vision based object detection methods on geotagged images
(2) Density based clustering of object tags to fuzzily assign objects to nearby locations, enabling users to ask membership questions, like "Which locations contain a swimming pool, a statue, and a palm tree?"
(3) Mapping objects associated with each location to a matrix (termed a *ConceptMap*) to facilitate spatial querying of object-object cardinal relations like "A bus stop Northwest of a pond" and object-location relations like "A driveway Southeast of a winery".
(4) Performing probabilistic search over locations based on object membership queries and optionally pruning the search results with spatial constraints using the ConceptMaps generated in (3)

We contribute a new gold standard hand-labeled *Swan Valley Wineries* benchmark dataset for the last-mile spatial search problem and provide a proof of concept implementation of the proposed *GESTALT* architecture, reporting Nicole: RESULTS on the wineries dataset, SCALABILITY results on noisy DC data?

The rest of this paper is organized as follows. In section 2, we define the *GESTALT* architecture and discuss how each subsystem contributes to our human-centric approach to automating the last-mile search. Next, in section 3 we describe the process used to generate the gold standard wineries dataset and extract noisy object tags from geotagged images. Then, we describe the object ownership assignment process in section 4, the concept mapping step in section 5, and the search problem in 6. Finally, we summarize related work in section 7 and conclude by identifying future research directions in section 8.

2 ARCHITECTURE

The architecture of *GESTALT* is outlined in Figure 1. The components cover four essential functions: data acquisition, object ownership assignment, concept mapping, and search. We briefly describe each of them below, focusing on their core purposes, and leaving the detailed implementation details and motivation vis-a-vis human-centric search for sections 3 through 6.

2.1 Data Acquisition

The purpose of the Data Acquisition subsystem is to ingest heterogenous sources of objects and locations into *GESTALT*, aiming to maximize the recall of all possible objects. *GESTALT* currently supports ingestion of hand-labelled objects from KML Files, crowd-sourced object labels from OSM and automatically generated object tags we extract from images geolocated within our region of interest by Flickr. All of these data sources are fused into a common format and assigned porbabalistic scores reflecting the likelihood that the object detected is actually present in the real world. The Data Acquisition subsystem ends when all of the data sources have been stored as JSON files, ready for ingestion by the Ownership Assignment subsystem.

2.2 Ownership Assignment

The purpose of the Ownership Assignment subsystem is to identify which *location* each identified *object* belongs to. The system accepts a JSON file of *locations* and a series of JSON files of geolocated *objects* both generated by the Data Acquisition subsystem. We use the DBSCAN Kent: cite dbscan to cluster the objects prune out objects unlikely to be associated with any paticular location by assigning them to a *Null Cluster*. The resulting cluster centroids are instantiated into a KD-Tree with the location centroids and a nearest-neighbour search determines what location is closest to that cluster of objects, and assigns it as the parent *location* for that *object cluster*. The Ownership Assignment subsystem returns a dataframe of objects, their predicted parent locations and their respective coordinates.

2.3 Concept Mapping

The purpose of the Concept Mapping subsystem is to create the data structures that will enable for advanced pruning and geospatial searching of objects and locations, including the pictorial specification of queries, leveraging the human tendency to draw scratch-maps to describe locations and directions. We create three data structures: an inverted index, location-object index and a object-object matrix. The inverted index has the obejcts as keys and locations as values. It supports exact and fuzzy set membership querying to prune the search space for downstream geospatial searching by only returning locations that contain the objects being searched for. If the object doesn't exist at a location, there is no point in searching for its relative location. The Concept Mapping subsystem accepts as input the dataframe of objects, their predicted parent locations and the coordinates of both. Using the input dataframe, the subsystem creates two distinct data structures, intended to support different types of spatial queries. The first location-object concept map treats the coordinates of the *location* as the division point on both the north-south and west-east axes. For each location, the data structure contains four lexicographically ordered lists, one for each quadrant NW, NE, SW and SE. Each list contains the objects belonging to that location which reside in that quadrant, relative to the location centroid. The second object-object concept map makes no assumptions about the position of the objects relative to the location, and rather is a representation of each object, relative to every other object. The object-object concept map is sparse M x M matrix, where M is the number of objects assigned to the location, and an object at position [i,j] is the i^{t}h object from the north and the j^{t}h object from the west. The concept mapping subsystemreturns these three data structures - the object inverted index, the location-object indexes and the object-object matrices.

2.4 Search

The purpose of the search subsystem is to enable the user to identify locations of interest based on the collection and geospatial arrangement of objects known to them. It accepts as input a user query - either through a keyword search or a pictorial query specification and the three data structures created by the concept mapping subsystem. It is capable of exact and fuzzy searching. The search subsystem balances precision in reducing the number of candidate locations with maximizing the recall of possible candidate locations.

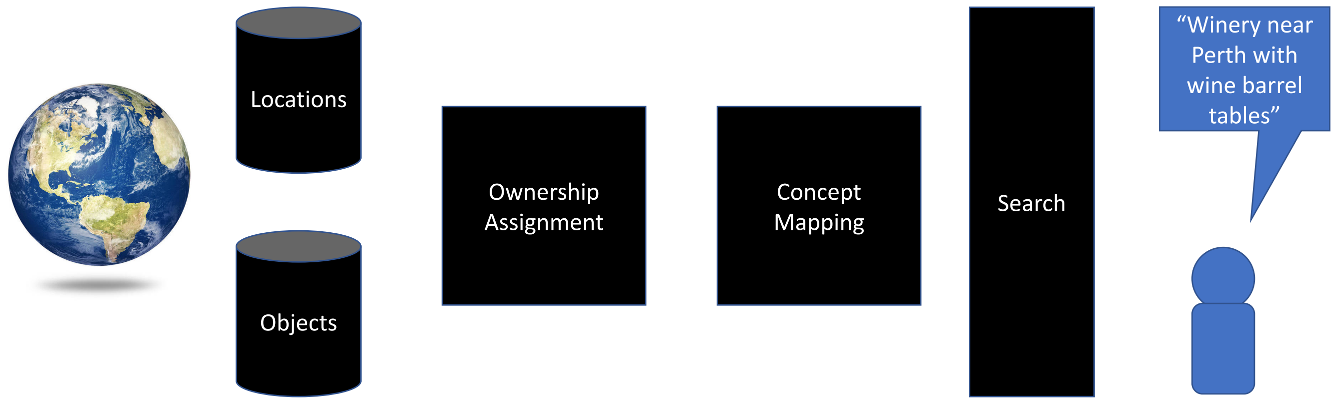


Figure 1: The architecture of *GESTALT* consists of the data collection subsystem, the ownership assignment process, the concept mapping process and the search subsystem.

The recall is prioritized based on the probability that they are the user's intended location.

2.5 Summary of Architecture

The Architecture of *GESTALT* is designed to be lightweight and modular. The core requirement is to automate the last-mile search problem and allow users to search for locations of interest based on partial information. Achieving this outcome necessitates collecting and processing objects and locations autonomously, at scale. Given a large quantity of noisy object tags, *GESTALT* performs fuzzy Object assignment, allowing for objects to be assigned to multiple nearby locations, and thereby improving the recall of the system. The subsequent concept mapping and spatial search processes maintain the system's precision using the relative directional relationships between query objects to extract implicit information about the location the user seeks.

3 DATA ACQUISITION

GESTALT enables last-mile search by encoding visual and geospatial data within a given *region*, using two types of geotags: *object* tags and *location* tags.

Regions represent a limited physical area of interest within which last-mile search is to be performed. Regions can be of arbitrary size, however the performance of the search functionality is related to the amount of objects and locations containted within the region, and so should be adjusted proportional to the density of objects and locations. Regions are defined by bounding-boxes for compatibility with the OSM and Flickr APIs.

Objects represent any physical entity located within the region of interest. For example, a *tree, building, lake, bridge, gate* or *sign* could be an object. *Objects* can also have attributes that provide amplifying information about them, including things like *color, material, size, species etc.*.

Locations represent physical entities that *do* something, giving them a purpose beyond that of objects. Locations can contain meaningful groupings of objects determined by ownership, proximity or utility. Examples of locations include *businesses, attractions, properties etc.*. *Locations* have *objects* associated with them, and *GESTALT* enables users to query for locations given a partial set of knowledge about the objects at those locations.

3.1 Object Tags

To maximize dataset coverage, and demonstrate the flexibility of *GESTALT* we support three methods of ingesting objects:

1. Ingesting KML Files that contain manually annotated objects and their coordinates
2. Querying the Open Street Maps (OSM) API to ingest crowd-sourced object tags and their coordinates
3. Automatically detecting objects in geolocated photos pulled from the Flickr API

Upon ingest, each object is assigned a confidence score, reflecting the certainty that the object tagged at those coordinates actually exists and is of the type annotated. For results reported in this paper, we adopt the rule that hand-labeled objects receive a confidence score of 1.0, OSM objects receive a score of 0.75 and objects labeled by the object detector receive the confidence score reported by the object detection model. Table 2 contains a summary of the objects currently in *GESTALT*.

Source	Ingest Method	# tags	# unique
Swan Valley Wineries	Hand-labeled	146	41
OSM Bounding Box[3]	Crowd-Sourced	2466[4]	38
Flickr Bounding Box	Object detection	1893[5]	55
OSM Query (Winery/Brewery)	Locations	36	36
OSM Bounding Box	Locations	308	308

Table 1: Summary of the Objects and Locations in the Swan Valley Wineries Dataset. Data available Kent: insert data link here

Source	Ingest Method	# tags	# unique
DC	Hand-labeled	0	0
OSM Bounding Box[6]	Crowd-Sourced	60123[7]	175
Flickr Bounding Box	Object detection	31065[8]	80
OSM Bounding Box	Locations	12179	12179

Table 2: Summary of the Objects and Locations in the DC Dataset. Data available Kent: insert data link here

3.2 Location Tags

We support two methods of ingesting locations:

(1) Ingesting KML Files that contain manually annotated locations and their coordinates.
(2) Querying the Open Street Maps (OSM) API to ingest crowd-sourced location tags and their coordinates.

Kent: 0.75 for OSM and yes we sure do, they're the parent container for the objects.

3.3 Ingest Methods

3.3.1 Ground Truth Hand Labeled Tags. Hand labeled objects and locations are those that have been manually annotated by a trustworthy source, and are assumed to be correctly labeled and correctly geotagged. *GESTALT* accepts hand-labeled tags to allow for prior manual annotation work to be folded into the database, and to provide a reliable means to report results on ground truth data, so that we can compare *GESTALT* with future architectures that might attempt to solve the last-mile search problem with a different approach.

For benchmarking purposes we curated the *Swan Valley Wineries* dataset containing 31 ground truth location tags for wineries (and five for breweries) in the Swan Valley Region of Western Australia and 146 ground truth object tags associated with six of those wineries. The wineries dataset tags are stored in Keyhole Markup Language [9] (KML). The tags consist of an object name, its latitude & longitude, and any descriptive markings written as key:value pairs.

The object tagging was conducted manually by a single annotator using *Google Earth Professional* [10] *version 7.3.* The data sources include on-the-ground knowledge, with manual inspection of satellite imagery, street-view imagery, and publicly available photos of the area. The objects tagged are representative, not exhaustive. Attributes of the objects (e.g. color, size, material) are recorded in the comments field as key:value pairs. Each object from the hand-labeled dataset is assigned a confidence score of 1.0, since it was manually identified and tagged.

The object tags are aligned with their corresponding winery location in the dataset. Figure 2 shows the winery locations in the Swan Valley Wineries dataset and Figure 3 shows each of the 6 wineries that was hand labeled with ground-truth object tags, along with those tags and their spatial locations with respect to the winery location.

3.3.2 Open Street Maps Tags. *GESTALT* can also ingest object and location tags by querying the Open Street Maps (OSM) API. OSM is a knowledge collective that contains open-source Geodata [?], which can be easily extended via their online editing interface. The basic unit in the OSM database is an *element*[11]. The key element subtypes are *Nodes* (point data for objects and locations) *ways* (line data for roads, creeks, railways etc) and *relations* (relationships between elements, a suburb is a sub-relation of a state for example). While businesses, attractions and other *locations* are commonly annotated by the OSM community, *objects* are more sparsely tagged, since few people have the patience to manually label apparently inconsequential things like trees, statues, fountains and telephone poles. *GESTALT* ingests *nodes* through the overpass API using the OSMPythonTools package [12]. The API query asks for all nodes within a given bounding box that have at least one tag (e.g "name" or "craft" or "landuse" or "address" could all be tags.) The initial list of nodes is then pruned to remove results likely to not refer to objects (e.g. nodes with street addesses, phone numbers or names are unlikely to be objects and may be better classified as locations) or that lack detail to be resolved to objects. The remaining nodes are transformed into a standard JSON format and stored for use by future subsystems in the *GESTALT* pipeline.

Each object ingested from OSM is assigned a confidence score of 0.75 by default. While any user is able to edit tags in the OSM database without review, the open-source community does have some degree of self-regulation and prior work has examined the application of machine learning for detecting anomalous behavior in OSM edits [?], the OSM data can still be wrong in many cases [?].

Businesses, attractions and other higher-level *locations* are commonly annotated as nodes in OSM by the open-source community or business owners themselves. We retrieve them from the OSM Overpass API using the reciprocal method to the above - still collecting all of the nodes with more than one tag, but this time excluding ones that lack features like names, street addresses, phone numbers etc.

While OSM provides a rich, free and accessible data source to leverage the power of the crowds to generate tags for GESTALT, the completeness of OSM is generally unassured, so scaling *GESTALT* beyond the trivial requires an automated method for object detection and resolution.

3.3.3 Noisy Image-based Tags. The third, and most important method by which *GESTALT* ingests objects, is through automatic object detection. We source relevent images through the Flickr API, returning all images that have been uploaded within a given bounding box since January 1^st 2020. We

[3] BoundingBox:['115.96168231510637', '-31.90009882641578', '116.05029961853784', '-31.77307863942101']
[4] 3056 objects originally returned, 590 dropped for not being objects
[5] from 462 photos
[6] BoundingBox:['-77.120248', '38.791086', '-76.911012', '38.995732']
[7] 113339 objects originally returned, 53216 dropped for not being objects
[8] from 4249 photos. Most objects in a photo was 61, average number of photos was 7.3

[9] https://developers.google.com/kml/documentation/kml_tut
[10] https://www.google.com/earth/about/versions/
[11] https://wiki.openstreetmap.org/wiki/Elements
[12] https://pypi.org/project/OSMPythonTools/

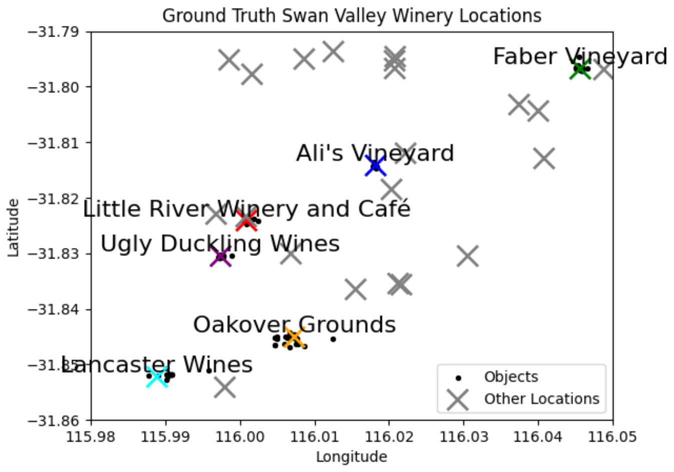


Figure 2: The Swan Valley Wineries dataset has six locations with annotated objects, and another 30 without manually annotated objects, but with confirmed location coordinates.

temporarily store the images locally until this step is complete. We also store relevant geospatial metadata Flickr holds on those images in a JSON format. Given the set of images and their EXIF data the Object Detection module uses YOLO [13] to identify objects in each image. Those objects are then tagged with the geolocation of the image, and stored in an updated version of the JSON created from the initial Flickr API pull. For our experiments on the Swan Valley Region, we pull 462 images since January 1^st 2020 and use pre-trained YOLO v.8 to detect objects from 80 classes (based on the COCO dataset Nicole: footnote). The DC dataset over the same timeframe returns over 12,000 images. We process a subset of 4000 to demonstrate the functionality of GESTALT over larger scales. Each object identified is assigned the classification confidence score returned by the object detection model.

4 OWNERSHIP ASSIGNMENT

We formulate the last-mile search problem in terms of *locations*, which are searchable entities that 'own' or contain any number of *objects*.

Kent: Might it be worth bringing the shawshank quote right up to the front? It's really good - shows locations, hinting things on geospatial relations, and has a fun pop culture hook element too

Recall the famous line from *The Shawshank Redemption*[14] *"There's a big hayfield up near Buxton... It's got a long rock wall with a big oak tree at the north end ... At the base of that wall, you'll find a rock that has no earthly business in a Maine hayfield. Piece of black, volcanic glass."*. Real people, in the real world don't think, or remember by latitude and longitude. They live in the world, and experience it thorugh the things that they can see, hear, touch, smell and taste. We design *GESTALT* to enable users to search for locations using the same intuitive tools and tecniques that one might use to tell a friend how to get to a camping site they like, or, in Andy Dufrense's case, where Red can find buried treasure. We enable users to use their experiences of locations in the real world as the search conditions (i.e. a location that contains <object1, object2, etc.>, or a location that contains <object1 Northeast of object2>). To enable these types of queries, *GESTALT* must associate objects with their parent locations. We call this problem *Object Ownership Assignment*, and define it as follows:

[13]YOLO
[14]https://www.imdb.com/title/tt0111161/

Kent: not sure what you mean by definition formatting? Nicole: definition latex fo

Given a collection of locations and objects within a region, *Ownership Assignment* seeks to correctly assign each object to its 'parent' location. This amounts to a clustering problem where points (objects) are assigned to centroids (locations) that are known apriori. The objects are not uniformly distributed across locations, and some locations may not have tagged objects associated with them.

Further, the human eye does not see the world in regular grid lines such as on a map, and so the ownership assignment process is naturally inexact, and objects are 'shared' between locations where appropriate. For example, a winery may have a shed out back that is visible both from that winery, and from a neighboring one. In this case, the object can be useful when searching for either location. We address this aspect of the problem by modifying our method to allow for fuzzy object-location assignment, effectively increasing the recall of the method.

4.1 Datasets and Metrics

We perform the Ownership Assignment task on two datasets: the hand-labeled Swan Valley Wineries dataset alone and a *Combined* dataset that includes all of the object and location tags from the Swan Valley Wineries dataset in addition to the noisy object tags from the Flickr dataset and the crowd-sourced object tags from the OSM dataset. Kent: Worth mentioning the DC data here Since we have ground truth labels for the winery locations and their objects in the Swan Valley dataset, we report both precision and recall on this dataset. The Combined dataset presents a more realistic and challenging scenario for Ownership Assignment than the Swan Valley Wineries dataset alone, given its larger set of locations and very large set of object tags to assign to those locations. However, since the Combined dataset includes noisy tags for which we have no ground truth labels, we do not report precision on the Combined dataset. Instead, we measure only the recall on the same set of Winery locations and their known objects that were hand-labeled as part of the Swan Valley dataset. The Combined dataset tests how well the recall performance of our Ownership Assignment method holds up to additional noise and more densely packed objects and locations.

The Swan Valley Wineries dataset consists of 31 wineries and five breweries retrieved from OSM and 146 objects hand-labeled across five of those 31 locations with ground-truth labels. Table 3 contains the recall (and precision, where applicable) on those locations and objects.

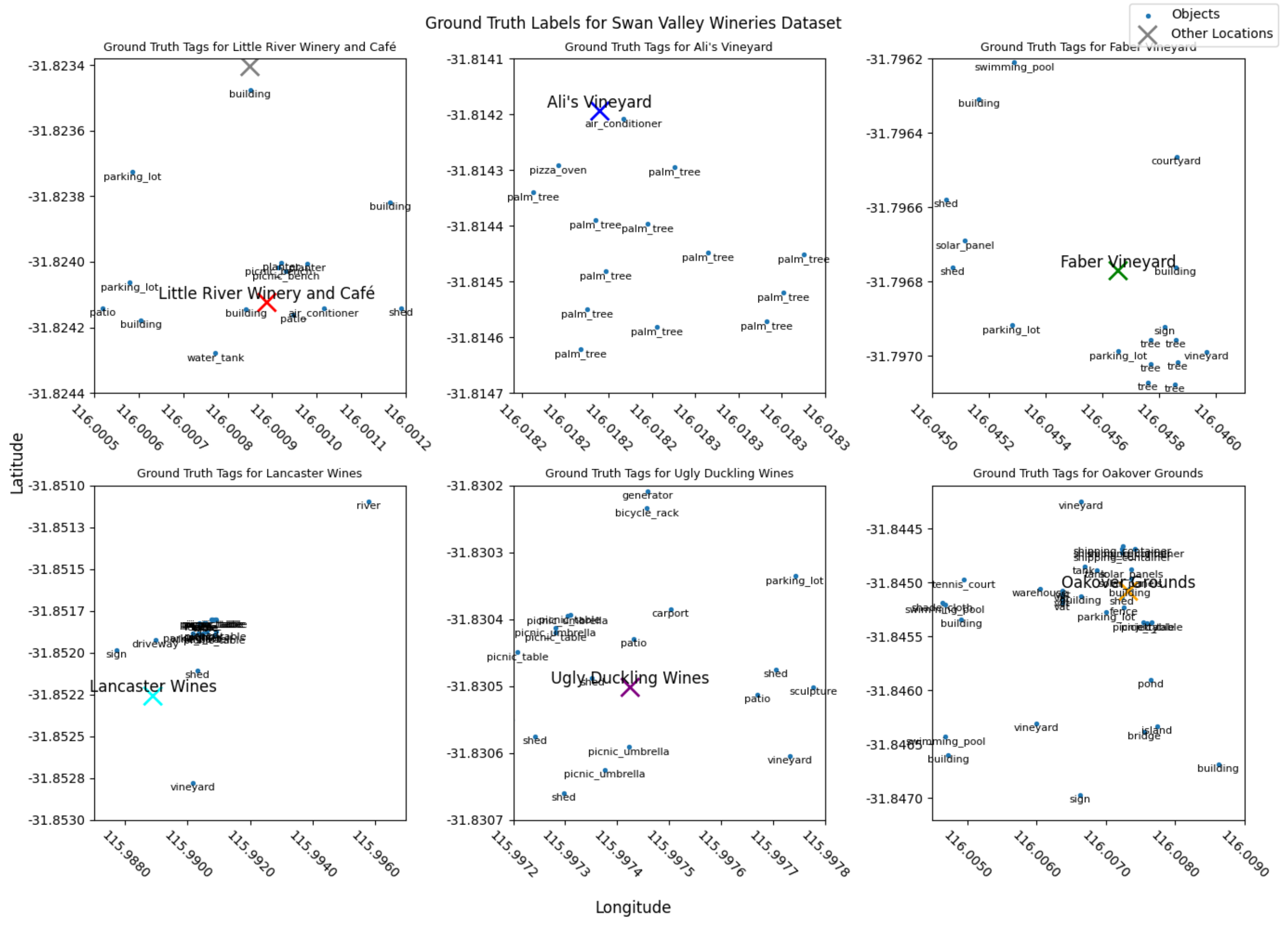


Figure 3: A respesentative view of the six wineries provided in the dataset shows the geospatial heterogenerity of object positions between locations.

4.2 Method

Our method for Object Ownership Assignment is outlined in Algorithm 1 . After clustering the objects, we determine the centroids of the relevant object clusters, calculate the confidence scores in the object-cluster assignments, and then map the cluster centroids to their nearest known location tags. When calculating the confidence scores, we normalize the object-centroid distances (omitting objects in the null cluster, which has no meaningful centroid and would skew the normalization). This confidence score (between 0 and 1) measures how far a given object is from it's cluster's centroid, assigning higher scores to objects near the centroid and lower ones to objects far from the centroid. We take this approach rather than using a static threshold parameter (like within x distance) to account for variety in object density of the region under search. We adopt the convention of assigning null cluster objects a confidence score of 0.5 since these objects are deemed to be noise and are not relevant or useful in finding locations.

To further account for uncertainty in object tags, we add an adjustable parameter c to the method which allows for a varying degree of *fuzzy assignment* of objects to clusters. By adjusting the parameter between 0 and 1, we can allow for objects to be assigned to multiple clusters if they are close to the centroid (within $100 * c\%$ of the largest object-centroid distance in the dataset, the same value used to normalize the confidence scores). We discuss how changing this parameter affects the precision and recall in subsection 4.3.

4.3 Results

Kent: Nicole you'll need to update these numbers after you run them through your analysis again. This section discusses the empirical analysis of the clustering methods tested during the implementation of the Ownership Assignment process.

Nicole: interpret results....recall increases with fuzziness etc.

Kent: Blocked pending analysis results

The benefit of using DBSCAN in the Object Ownership Assignment context is that it acts as a noise reducer, filtering out objects which do not belong to any locations (i.e. a mis-tagged singular object with nothing else around it for miles in any direction). The downside of using DBSCAN in this context is that it determines the centroids of the object clusters based on the object density, and then we must map those to our known location centroids in a post-processing step. Ideally, a better solution would start with the centroids and cluster around them.

Algorithm 1 Object to Location Ownership Assignment Algorithm

```
***Locs*** a list of locations and their tagged coordinates
***Objs*** a list of objects and their coordinates
***Clusters*** a list of object clusters
***C*** an individual cluster in $Clusters$
***C.x*** a cluster centroid
***C.l*** the predicted $location$ for a $cluster$
***O*** an individual object in $Objects$
***O.c*** confidence score that object was correctly assigned
- - - - - - - - - -
procedure ObjectOwnershipAssignment($Locs$,$Objs$)
    $Clusters \leftarrow$ DBSCAN($Objs$)
    for All $C$ in $Clusters$ except NULL cluster do
        $C.x \leftarrow$ Calculate Cluster Centroid
        for $O$ in $C$ do
            $O.c \leftarrow 1 - \frac{1}{dist(O,C.x)}$
        end for
    end for
    for All $O$ in NULL Cluster do
        $O.c = 0.5$
    end for
    for All $C$ in $Clusters$ Except NULL Cluster do
        $C.l \leftarrow$ Closest $L$ in $Locs$ to $C.x$
    end for
end procedure
```

Kent: DVBSCAN doesn't do what I thought it did. I think one for future work is modifying DBSCAN to start with known possible centroids (or cluster seeds) We leave a detailed comparison of Object Ownership Assignment techniques as an interesting avenue of future study for the last-mile search problem.

4.4 Scalability

Kent: Report on Swan Valley: Time to build clusters for fuzze = 0, 0.5 and 1.0

To test the scalability of the Ownership Assignemnt process, we use the *DC Dataset* we create from OSM and Flickr queries, containing 12,179 locations, 91,188 objects and over 200 distinct object classes across a bounding box sufficent to cover the entire district of columbia.

Kent: Report Clustering Time here for fuzzy = 0, 0.5 and 1.0, update these numbers now that I've fixed the OSM sanitization

Nicole: - complexity analysis - present timings/ etc.

5 CONCEPT MAPPING

The core of *GESTALT*'s geospatial search capability resides in what we call the *Concept Mapping* component. Concept mapping is the process of extracting and explicitly encoding the implicit geographic relationships between objects and locations. By encoding these spatial relationships in a manner that can be compared with visual queries issued by the user, we enable a form of spatial last-mile search that (to our knowledge) does not presently exist in any GIS tools. We implement two forms of concept mapping: object-location concept mapping and object-object concept mapping. In both cases, concept mapping involves two phases: the encoding phase (offline) and the search phase (online). We discuss the details of the encoding phase in this section and leave the search details to section 6.

5.1 Object-Location relations

Object-location relations encode whether an object is North, South, East or West of a location. This type of information supports simple queries, such as when a user knows that a location has a lake on its western side. Each individual object-location relationship is defined independently of other objects associated with that location, and multiple constraints can be combined to form a more specific query, such as a lake west of the location and a pond south of the location.

Figure 4 shows how we encode a location (4a) as a geospatial index based on realtive position to the location centroid (4b) , and then how queries are processed (4c).

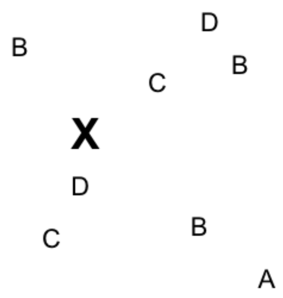
To encode a location, we begin by setting the centroid of the location provided by OSM during Data Acquisition to be the root of a quartering of the search space into NorthWest, NorthEast, SouthWest and SouthEast quadrants. Each object's coordinates are compared to the location centroid and it is assigned to the appropriate quadrant. Where a point lies on the border we adopt the convention that it belongs to the southern and western quadrants. To query the location-object index, a user imagines themselves standing at the centre of the location and enters the query terms into the quadrant they believe it belongs in. That query map is then compared to each candidate location. An object in the query is in the same quadrant as an object in a candidate location, the number of correct terms increments. The query returns the count of correct assignments, which are then used to rank the candidate locations from most to fewest matches.

Dataset	Method	Fuzzy Param	Precision	Recall
Wineries	Exact	$c = 0.0$	1.0	0.9272366522366523
	Fuzzy	$c = 0.1$	1.0	0.9371639544053337
	Fuzzy	$c = 0.2$	1.0	0.9476125333854473
	Fuzzy	$c = 0.3$	1.0	0.9532960538279688
	Fuzzy	$c = 0.4$	1.0	0.9557047250443477
	Fuzzy	$c = 0.5$	1.0	0.958204892379114
	Fuzzy	$c = 0.6$	1.0	0.9585494229562027
	Fuzzy	$c = 0.7$	1.0	0.9594202898550726
	Fuzzy	$c = 0.8$	1.0	0.9599579242636747
	Fuzzy	$c = 0.9$	1.0	0.9541992532697066
	Fuzzy	$c = 1.0$	1.0	0.9547516522596055
Combined	Exact	$c = 0.0$	-	0.9348124098124098
	Fuzzy	$c = 0.1$	-	0.9471508069381088
	Fuzzy	$c = 0.2$	-	0.9382094943240453
	Fuzzy	$c = 0.3$	-	0.9412204625439919
	Fuzzy	$c = 0.4$	-	0.9337391384811933
	Fuzzy	$c = 0.5$	-	0.9306786154945126
	Fuzzy	$c = 0.6$	-	0.9299095431931251
	Fuzzy	$c = 0.7$	-	0.9276190476190475
	Fuzzy	$c = 0.8$	-	0.9255792621327837
	Fuzzy	$c = 0.9$	-	0.9249924084835232
	Fuzzy	$c = 1.0$	-	0.9178969436169039

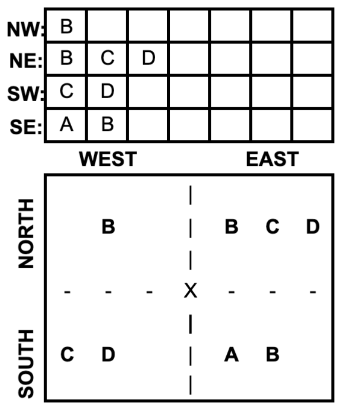
Table 3: Kent: All run with $\epsilon = \frac{0.1}{6371}$ and MinCluster=3 ....Combined means hand-labeled Swan Valley Wineries dataset tags and Flickr noisy object-detected tags which we have no ground truth for. We only report recall for the combined dataset since the hand-labeling was not exhaustive, and some results from the noisy data may be correct but not in the ground truth. C represents the fuzziness parameter. higher c means more fuzzy and greater chance objects are assigned to multiple clusters.

5.2 Object-object relations

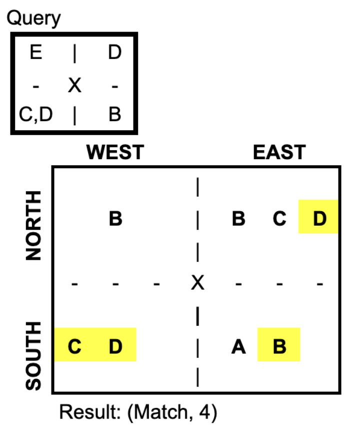
The other type of spatial relationships *GESTALT* supports searching are object-object relations, which in the genral case are more complex than



(a) A candidate location X has named objects A-D with the spatial layout depicted above.



(b) The objects are binned into spatial quadrants based on their relative position to the location centroid, X.



(c) Searching counts the number of query terms that correspond to a location's quadrants and returns the count, allowing multiple candidate solutions to be ranked by closeness of match.

Figure 4: Generate and Query an Object-Location Concept Map.

object-location relations. Object-object relations refer to a configuration of objects that each relate to one another spatially. For example, a tree, a pond, and a sign might form a triangle, with each object having a directional constraint with respect to every other object in the query (i.e. tree northwest of pond, pond southeast of sign, and sign southwest of tree). As the number of objects in the object-object configuration increases, the number of pairwise directional relationships needed to specify the query grows Nicole: exponentially??

Kent: I don't think so. Thinking of fully connected graphs, for V:E, 1:0, 2:1, 3:3, 4:6, 5:10, 6:15 ... that's not exponential growth right? . As such, object-object relations lend themselves particularly well to pictorial query specification Nicole: CITE . This method of specifying the spatial positioning of objects aligns nicely with how humans tend to think about and describe landmarks in the world- by drawing a map.

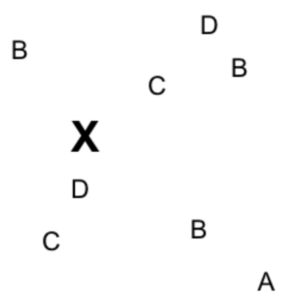
Figure **??** shows how we encode a location (**??**) as a matrix of relative object positions (5b) , and then how queries are processed (5c).

For intuition on this approach, consider the limitations of the location-object approach. It assumes that the user knows where the location centroid is and which way north was at that location. For large locaitons it doesn't account for a situation were a user may information about objects in one qudrant, limiting their ability to make good matches. In these circumstances, we can instead search for the relative position of objects to each other, and ignore their absolute position within the location. To generate the object-object matrix concept map, we sort all objects from north to south in one list, and all objects west to east in another. Their position in the matrix is then determined by their position in the matrix. For example in figure reffig:CM-OO-Setup object **"A"** is in the 8th position from the North, and the 8th Position from the west, so in the matrix it is assigned to index [0,0]. To then query this sparse matrix, approached it as you might approach looking for something on a map - by eliminating confusing or redundant information until we are left with a very simple question to answer. We would probably just zoom in a little closer on our mapping software to see if we can see the arrangement of objects we are looking for - and so our approach mimics the 'zooming-in' approach. Our implemenation aggressively prunes the search space using recursive grid search Kent: Cite . In Figure **??** the query has a "C" object as the most northern object. So, we can prune the entire search space north of the north most C, because no matches in that region can satisfy the query. Next, recurse and we prune on the matrix from the west, knowing that no terms west of the west-most D can possibly satisfy the query. The pruning continues from the north, then the south until there is only a single search term left. If that term exists anywhere in the unpruned area, the query matches. Note that this approach returns any collection of objects that match the pattern specified in the query, regardless of other objects being present or how many times the sub-pattern occurs in the query space.

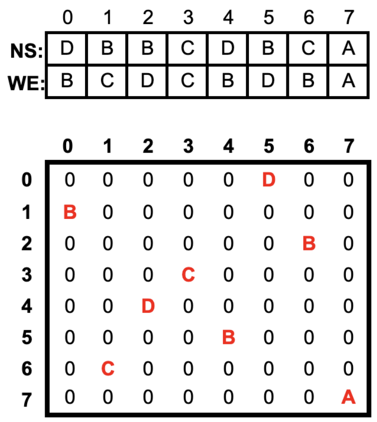
5.3 Encoding the Spatial Relationships of objects and locations

5.3.1 Object-location encoding. Object-Location concept maps are encoded as a per-location index that subdivides the search space into four quadrants - NorthWest, NorthEast, SouthWest and SouthEast, splitting on the centroid of the location. The divison of space aims to emulate how people tend to relate the world to their position in it geospatially,

5.3.2 Object-object encoding. Rather than use a fully-connected graph, we encode both the locations, and later, the pictorial query inputs as matrices with zeros representing unallocated space and the names of objects. The matrices are NxN, where N is the number of objects at a location. Using algorithm 6, we assign each object in a given location to a position (i, j) in the Matrix where i is its order of appearance from north to south and j is the same object's order of appearance from west to east. The result is a matrix in which each row and column has only a single object. Where ties are experienced in the real world (due to a lack of GPS precision, or very close position) they are broken lexicographically in the object-object concept map instantiation.



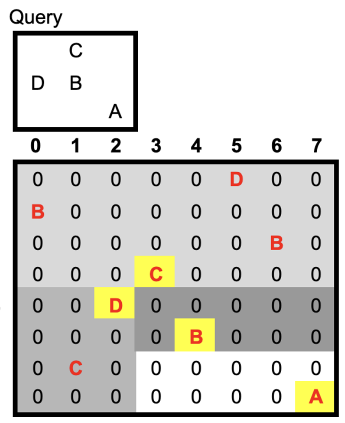
(a) A candidate location X has named objects A-D with the spatial layout depicted above.



	0	1	2	3	4	5	6	7
NS:	D	B	B	C	D	B	C	A
WE:	B	C	D	C	B	D	B	A

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	D	0	0
1	B	0	0	0	0	0	0	0
2	0	0	0	0	0	0	B	0
3	0	0	0	C	0	0	0	0
4	0	0	D	0	0	0	0	0
5	0	0	0	0	B	0	0	0
6	0	C	0	0	0	0	0	0
7	0	0	0	0	0	0	0	A

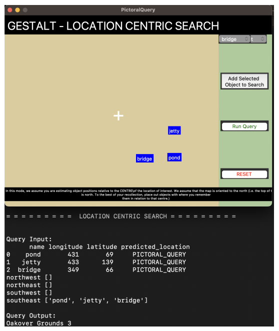
(b) During pre-processing, the objects associated with location X are ordered North to South (NS) and West to East (WE) and mapped into a matrix with corresponding indices.



	0	1	2	3	4	5	6	7
0	0	0	0	0	0	D	0	0
1	B	0	0	0	0	0	0	0
2	0	0	0	0	0	0	B	0
3	0	0	0	C	0	0	0	0
4	0	0	D	0	0	0	0	0
5	0	0	0	0	B	0	0	0
6	0	C	0	0	0	0	0	0
7	0	0	0	0	0	0	0	A

(c) Searching recursively prunes for ANY match. Each recursion is a darker shade, with unpruned area in white. Objects highlighted in yellow are found to match the query configuration; candidate location X is a match for the query.

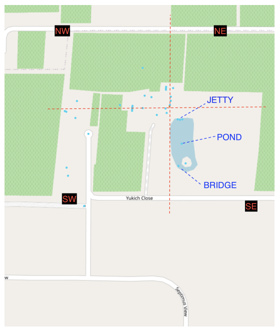
Figure 5: Generate and Query an Object-Object Concept Map.


PictoralQuery
GESTALT - LOCATION CENTRIC SEARCH
bridge
Add Selected Object to Search
Run Query
RESET
jetty
bridge
pond

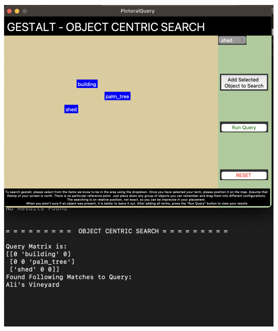
In this mode, we assume you are estimating object positions relative to the CENTRE of the location of interest. We assume that the map is oriented to the north (i.e. the top of t
is north. To the best of your recollection, place out objects with where you remember
them in relation to that centre.)

```
= = = = = = = = =  LOCATION CENTRIC SEARCH = = = = = = = = =


Query Input:
      name longitude latitude predicted_location
0    pond       431       69     PICTORAL_QUERY
1   jetty       433      139     PICTORAL_QUERY
2  bridge       349       66     PICTORAL_QUERY
northwest []
northeast []
southwest []
southeast ['pond', 'jetty', 'bridge']

Query Output:
Oakover Grounds 3
```

(a) The Location Object GUI uses the nominal centre of the location "+" to divide the search space into quadrants.



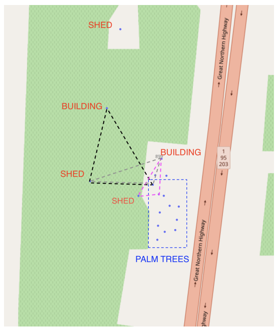
(b) The location of the Oakover coordinates shows that it won't necessarily be the centre of the object cluster that the space is subdivided on.


PictoralQuery
GESTALT - OBJECT CENTRIC SEARCH
shed
Add Selected Object to Search
Run Query
RESET
building
palm_tree
shed

To search gestalt, please select from the items we know to be in the area using the dropdown. Once you have selected your term, please position it on the map. Assume that thetop of your screen is north. There is no particular reference point. Just place down any group of objects you can remember and drag them into different configurations.
The searching is on relative position, not exact, so you can be imprecise in your placement.
When you aren't sure if an object was present, it is better to leave it out. After adding all terms, press the "Run Query" button to view your results

```
No Results Found


= = = = = = = = =  OBJECT CENTRIC SEARCH = = = = = = = = =

Query Matrix is:
[[0 'building' 0]
 [0 0 'palm_tree']
 ['shed' 0 0]]
Found Following Matches to Query:
Ali's Vineyard
```

(c) The Object Object GUI allows users to arrange objects in any manner they see fit.



(d) The triangles show matching candidatesm any of which would cause the query to return true.

Figure 6: Users of the prototype Pictorial Query interface in both modes add objects from the drop-down menu and position them on the screen by dragging the lables. Beneath each window is the encoded query and the result from searching GESTALTs database. The Pictoral Query Interface allows users of GESTALT to mimic how they would explain features of location to another person, in this case by drawing a quick map.

6 SEARCH

The core function of *GESTALT* is to perform last-mile search given partial or uncertain information. The user is assumed to know the general region of interest, which could be an administrative boundary like a city or suburb, or a general geographic area. The user is also expected to know some information about the objects at the location they seek. Under these conditions, the search problem can be framed in several ways, described in the subsequent subsections in increasing order of complexity and utility.

6.1 Exact Membership Search

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.

Nicole: - problem definition - complexity

Algorithm 2 Search

```
QT a list of query terms to search for
II an inverted index with objects as keys and locations as values
- - - - -
procedure SEARCH(QT,II)
    Locs = []
    for Each q in QT do
        Retrieve set of Locations II[q] and add to Locs
    end for
    return intersection of sets of Locations in Locs
end procedure
```

6.2 Ranked Membership Search

Nicole: - problem definition - complexity

Algorithm 3 Rank

```
QT a list of query terms to search for
II an inverted index with objects as keys and locations as values
- - - - -
procedure RANK(S,q,II)                    ▷ S results from q query terms in II
    LR = Empty Ordered Dictionary                          ▷ Location Rank
    for Each L in S do                                          ▷ Location
        p = 1                                  ▷ Probability location correct
        for O in q do
            OP = Prob of Obj@Locin II                  ▷ Object Probability
            P = PxOP
        end for
        LR[L] = P
    end for
    returnLR ordered most to least likely
end procedure
```

6.3 Fuzzy Membership Search

Nicole: - problem definition - complexity

Algorithm 4 Fuzzy Search

```
    QT a list of query terms to search for
2:  II an inverted index with objects as keys and locations as values
    - - - - -
4:  procedure FUZZYSEARCH(QT,II)
        S= II.SEARCH(QT)
6:      if S is Empty then
            q = Pop most discriminative term from QT
8:          S= II.SEARCH(QT)
            if S is not Empty then
10:             Skip to Line 6
            else
12:             S= II.SEARCH(QT.remove(q)
            end if
14:     end if
        if S has more than 1 item then
16:         return II.RANK(S,Q)
        end if
18: end procedure
```

6.4 Spatial Search

The most powerful form of search *GESTALT* enables is spatial search, where users can query for locations based on the spatial configuration of objects in relation to each other or to the location itself. However, as discussed in section 5, this problem quickly becomes intractable as the number of objects increases. To combat this effect and still enable users to specify spatial queries, we employ a successive pruning approach, where the search space is scoped down in stages until the resulting search is of reasonable scale to perform quickly.

6.4.1 Pruning the Search Space. The first round of pruning occurs when the user selects the *region* that they wish to search (a pre-condition to performing last-mile search). The second layer of pruning happens when the exact membership search method is applied to the query objects. The set-membership test prunes out any locations that do not contain the objects of interest, thereby avoiding unnecessary computation associated with determining the spatial configuration of objects that are irrelevant to the query (i.e. that belong to locations that are not candidates). The third layer of pruning occurs during the search of the concept maps. The recursive grid search (Algorithm 6) successively segments the search grid into smaller and smaller sections, eliminating impossible results at each step to reduce the total search space to a tractable size.

6.4.2 Specifying the Query Pictorially. A picture is worth a thousand words, and maps are an ingerently visual medium. *GESTALT* recognises the expressive limitations of words alone for geospatial querying and invites a user to query its database with quick sketch maps. The pictorial query specification abstracts away the need for the user to describe the relations of objects using the confusing language around cardinal directions, and replaces it with the simple visual task of trying to recreate the pattern being held in your mind's eye. We implement a very rudimentary prototype pictorial query interface using Python Tkinter. The interface has two modes. Figure 6a shows the location-object configuration, where the coordinates of the location are placed at the centre of the canvas and all objects positioned around that centrepoint by the user. The second mode shown in Figure 6c is the object-object configuration, where the canvas space allows users to place patterns of objects that they recall in space without concern for their loations respective to the location itself. If that pattern of objects exists in the database, *GESTALT* returns the result. Detail on how the underlying data structures are consturcted are available in the Appendices in Appendix 6.

6.4.3 Searching the Concept Maps. Searching the object-location concept maps requires the user to specify a query, either pictorially or with keywords, for each quadrant of an unknown candidate location, what objects they expect to find there. Using that query input, the searher iterates through each location and inspects its NW/NE/SW/SE subdivisions of space, comparing them to the query. Where the same object occurs in the same quadrant in the query and a candidate location, one point is added to the location's score. When all locations have been checked, the candidate locations (i.e. locations with > 1 match) are returned, ranked from most to fewest matches.

Searching the object-object concept maps requires the user to specify a query pictorially. That query is converted into a concept map matrix, and the search terms within that matrix are ordered into a queue with the most northern point at index 0, the most western at index 1 and continuing until the most southEasterly query term is at the tail of the queue. Given the query list, algorithm 5 details the process of recursively pruning the search space. The broad idea is that while finding all possible instances and configurations of a geospatial pattern rapidly approaches NP-Completeness, we can determine if there is any possible match quite easily by using the query term to prune sections of the concept map that are irrelevant. For example, any objects in the location that are north of the most northerly query term that exists in the object cannot possibly satisfy the query, and so we prune those rows. The same occurs for the western direction, and then recursively north and west until the query is eithers satisfied or rejected. The search occurs for each candidate location and requires less than a single matrix traversal to determine if anywhere in that location the query pattern occurs.

Algorithm 5 Recursive Grid Search

```
M A ConceptMap Matrix with objects or 0s; [0,0] is NW most point
L A NW to SE ordered list of objects to search for
D The direction of Pruning, 'northToSouth' or 'westToEast'
- - - - -
procedure recursiveGridSearch(M,L,D)
    if L has only 1 item then                              ▷ Base Case
        if pop(L) in M then
            return True
        else
            return False
        end if
    end if
    if D is northToSouth then
        NorthernObject = pop(L)
        if NorthernObject not in M then
            return False
        end if
        M = Prune all objects north of NorthernObject
        return recursiveGridSearch(M,L,westToEast)
    end if
    if D is eastToWest then
        WesternObject = pop(L)
        if WesternObject not in M then
            return False
        end if
        M = Prune all objects west of WesternObject
        return recursiveGridSearch(M,L,northToSouth)
    end if
end procedure
```

6.4.4 *Results on Ground Truth Spatial Queries.*

Nicole: write up results

End to end including object-object and object-location concept mapping

Kent: Need to work ouut how to do this. What queries to run etc.

- table listing the 12 location-object queries and expected results

- visually show the object object ground truth queries in a grid of subplots?

6.5 Scalability

Nicole: write this section 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.

Spatial queries are the most computationally intensive aspect of the search. Object-Centric queries require a concept map that describes all n-ary object-object relations, where Location-Centric queries only need describe the set of binary object-location relations. These problems of searching by geographic sets of objects have previously been characterized as graph-matching problems. As a graph-matching problem, the difference between a fully connected graph, and a hub-and-spoke graph is significant, particularly as the number of vertices increases. It quickly becomes intractable. Kent: [CITE HERE] We handle both these types of queries by focusing on pruning the search space down to a reasonable size before checking the spatial relationships between objects.

- for spatial results report number of locations pruned? timings?

The worst case complexity is for a search of N object terms where every object category is mapped to every location L. In that case we have to do N lookups to pull N sets of size L and intersect them. Realistically this would not happen and some object classes would be very discriminative and we could intersect those first to be more efficient.

7 RELATED WORK

signpost here

7.1 OSM Search Tools

On 08 May 2023 the Open Source Intelligence platform *Bellingcat*[15] released *Bellingcat OSM Search*[16] [**?**], a tool also aimed at pruning search space using tagged objects. Using drop-downs and sliding bars, a user can specify a query for OSM tagged objects that appear within an adjustable distance from each other inside a region of interest. Although aiemd at addressing the same *last-mile* search problem as *GESTALT*, there are several? .... key limitations to the Bellingcat OSM Search Tool that *GESTALT* overcomes.

(1) They can do progressive search by executing multiple queries sequentially – ok
(2) They can't do concept mapping type queries – depends if we implement it or not
(3) They return generic points meeting criteria rather than tagged locations – ok
(4) They don't use objects not tagged in OSM, which are global objects people are certain enough about to tag (optimizing for precision and majorly missing recall) – this is key
(5) They don't rank results – tied to the point above, also key – Instead we pull many many tags via object detection on images geo-ed for locations, and associate them with the location(s) near to the geotag, which is noisy but captures much more of the possible things people might remember (optimizing for recall). Note we would not want to add these noisy tags to OSM (which would allow Belingcat to search on them), since they are inherently noisy. We return ranked results to handle the uncertainty w.r.t assigning objects to multiple locations and just having rogue object tags that aren't really there – the best locations matching the query will cover most of the search terms. This allows flexibility beyond what OSM object tags can cover (anything a detection model or human annotator wants to put as the tag).
(6) They return 0 results if a search term fails to hit in the window of interest. – we should mention this – People can search anything they remember, and even if some don't hit (weren't tagged in OSM or by the object detection step) the results won't be 0 like Belingcat does, instead will be based on the other search terms. This is much more natural to how humans do search...
(7) By relying on OSM API queries, their tool can do any region as long as its sufficiently small, at the expense of being slow (API calls are slow). – can we compare the same queries on swan valley in our tool and theirs? – We need to address our scalability issues

7.2 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.

[15]Bellingcat Website
[16]Bellingcat OSM Search Tool

GeoGuessr. A popular online game called *Geoguessr*[17] 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 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.

7.3 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 labeling 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.

[17] Geoguessr Website

7.4 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.

7.5 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.

7.6 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.

7.7 Human Spatial Cognition

"Global objects were already in their original locations and could not be moved. This was because (a) they had not been targets before, (b) it is rather unusual to move them in everyday life, and (c) a recent VR scene building experiment by Draschkow and Võ (2017) shows that they are usually placed earlier in the building process and then serve as anchors to guide the arrangement of smaller objects that we typically interact with, which makes them appropriate location recall cues in our study." [**?**]

robots that do perception need anchoring too [**?**]

This study of cognitive strategies used in navigation and route planning shows that people form spatial representations of a virtual environment in differnt ways, including likely hierarchically, or with separate local and global features [**?**]. The implications are that our approach is consistent with how people remember spatial locations using a hierarchical cognitive map of objects/landmarks and/or using a hierarchical representation of objects/landmarks. Explain how these two possibilities are both served by the GESTALT searching method...

8 CONCLUSION

8.1 Future Work

Throughout this paper, we identify many avenues for future work. Sections **??** and 2 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 **??** highlights the need to develop datasets in dense suburban and urban locations to enable robust testing of the ownership assignment process. Sections 2 and **??** 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 **??** and 7 emphasize 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 **??** 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 7 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 prioritized for a fully functional *GESTALT* prototype before it expands to full scale.

8.2 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.

FUTURE: online clustering: [**?**]

Time decay on confidence score to account for changes in environment Fuzzy string matching Account for camera bearings to adjust coordinates of objects detected in images to be more precise Incorporate user feedback to update tags that might be wrong in the data Using object attributes to allow mroe detailed queries and stronger pruning Grid-ifying the earth and making a gestalt for each grid cell to enable last-mmile search anywhere on earth

Enable searching for object quantities: 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. Issue: multiple sources may tag same object, need to deduplicate them

8.3 Normalizing Search Terms

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 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 [**?**].

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.

Algorithm 8 Generating Location Centric Structure

```
T a Location Table with ID, Name, Lat & Long of its objects
X a (x,y) tuple of the centroid for the location
- - - - -
procedure MakeLocationCentricStructure($T$,$X$)
    $Q${$NW$:[], $NE$:[], $SW$:[], $SE$:[]}              ▷ Dictionary
    for $R$ in $T$ do                      ▷ R is a row describing a single object
        if $R.y < X.y$ then
            if $R.x <= X.x$ then
                Add $R.Name$ to $Q.SW$
            else
                Add $R.Name$ to $Q.SE$
            end if
        else
            if $R.x <= X.x$ then
                Add $R.Name$ to $Q.NW$
            else
                Add $R.Name$ to $Q.NE$
            end if
        end if
    end for
end procedure
```

Algorithm 9 Querying Location Centric Structure

```
Q a Dictionary of lists of objects of form: {NW:[], NE:[], SW:[], SE:[]}
NS a query Direction of "North_of" or "South_of"
EW a query Direction of "West_of" or "East_of"
S the name of an object to searc for
- - - - -
procedure QueryLocationCentricStructure($T$,$X$)
    if $NS = North_of$ and $EW = None$ then
        if $S$ in $Q.NW$ OR $Q.NE$ then
            return$True$
        end if
    end if
    if $NS = North\_of$ and $EW = West\_of$ then
        if $S$ in $Q.NW$ then
            return$True$
        end if
    end if
    if $NS = North\_of$ and $EW = East\_of$ then
        if $S$ in $Q.NE$ then
            return$True$
        end if
    end if
    if $NS = South\_of$ and $EW = None$ then
        if $S$ in $Q.SW$ OR $Q.SE$ then
            return$True$
        end if
    end if
    if $NS = South\_of$ and $EW = West\_of$ then
        if $S$ in $Q.SW$ then
            return$True$
        end if
    end if
    if $NS = South\_of$ and $EW = East\_of$ then
        if $S$ in $Q.SE$ then
            return$True$
        end if
    end if
    if $NS = None$ and $EW = West\_Of$ then
        if $S$ in $Q.NW$ OR $Q.SW$ then
            return$True$
        end if
    end if
    if $NS = None$ and $EW = East\_of$ then
        if $S$ in $Q.NE$ OR $Q.SE$ then
            return$True$
        end if
    end if
```

A APPENDICES

A.1 Concept Map Creation

Algorithm 6 Creating a Concept Map

```
T a Location Table with ID, Name, Lat & Long of its objects
- - - - -
procedure createConceptMap($T$)
    LonList = []                                        ▷ A list
    LatList = []                                        ▷ A list
    $M$ = [[][]]                                  ▷ A matrix of 0s
    Sort rows from North to South
    for $R$ in $T$ do                            ▷ Where $R$ is a Row
        Add R.$ID$ to $LatList$            ▷ $R.ID$ is the obj unique ID
    end for
    Sort rows from West to East
    for $R$ in $T$ do
        Add R.$ID$ to $LonList$
    end for
    for $ID$ in $Lonlist$ do
        get index of $ID$ in $LonList$ as $i$
        get index of $ID$ in $LatList$ as $j$
        $M[i][j] = ID.Name$                  ▷ Name of object with $ID$
    end for
    return $M$                         ▷ Having $[0, 0]$ as NW corner
end procedure
```

A.2 Object-Object Search Term Ordering

Algorithm 7 Ordering the Search Terms

```
M A ConceptMap Matrix with objects or 0s
- - - - -
procedure ORDER SEARCH TERMS($M$)
    if $M$ has single Item then
        Return $M$
    end if
    $Traversed$ = []
    $R$, $C$ = Number of rows and columns
    $Rd$ = ($R$+$C$)-1
    for $r$ in range $Rd$ do
        for $i$ in range $r$+1 do
            $j$ = $r$-$i$
            if $i < R$ and $j < C$ then
                if $M[i][j]$ != 0 then
                    Add $M[i][j]$ to $Traversed$
                end if
            end if
        end for
    end for
    return $Traversed$
end procedure
```

A.3 Location-Centric Strucutre Generation

A.4 Location Centric Querying