GESTALT: Augmenting geospatial search with micro-terrain detail

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

Nicole R. 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 images 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 contribute a new gold standard Swan Valley Wineries dataset and a proof of concept architecture that handles querying for spatial configurations of objects with respect to each other, handling uncertainty in the information known about a location, and accounting for the fuzzy boundaries between locations.

ACM Reference Format:
Kent O'Sullivan[†], Nicole R. 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, 13 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 [6, 8, 17], 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 [17]. 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 narrow down the general region of interest. However, the user must then perform a manual *last-mile* search to find the exact location of interest within that region. Last-mile search typically involves visually inspecting images to identify distinct landmarks or terrain features that match the partial information about the location. The last-mile component 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.

Last-mile search is a common problem in the open-source intelligence community, where geospatial analysts must try to identify a specific location within a general area using incomplete or uncertain information about the location. The process boils down to a data fusion and search task, where information from reports is manually compared to the visual features of remote sensing images and street-view photographs until the expected configuration of objects is found, indicating the location is a match. Recent related work from *Bellingcat* [1] highlights the need for search techniques that can accommodate queries for collections of objects in close geographic proximity to each other. However, automating the last-mile search process requires addressing several additional criteria, including accounting for the spatial configuration of objects concerning each other, handling uncertainty in the information known about a location by relaxing the search constraints until a partial match is found, and accounting for the fuzzy boundaries between locations, whereby an object may be visible from several nearby locations and can be associated with all of them to maximize the recall of the search result.

Taking inspiration from the way humans recall and search for information [5, 11, 17], we present *GESTALT*, an end-to-end pipeline for extracting geospatial data, transforming it into coherent object and location relations, storing those relations, and searching over them. Specifically, *GESTALT* provides the following functionality:

(1) Multiple methods for ingesting location and object tags, including automatically using object detection methods on geotagged images.

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

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

(2) Density-based clustering of object tags to fuzzily assign objects to (possibly multiple) 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 pictorially querying spatial relations like "A bus stop Northwest of a telephone pole"
(4) Performing probabilistic search over locations based on object membership queries and returning partial matches when the search constraints are too narrow, and ranked results when they are too broad.

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 precision and recall on the ground truth data and performance metrics on a more extensive set of data to demonstrate the scalability of the search procedures.

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* (Figure 1) covers four essential functions needed to perform last-mile search: data acquisition, object ownership assignment, concept mapping, and search. We describe and motivate each of them below, focusing on their core purposes and leaving the implementation details for sections 3 through 6.

Data Acquisition. The Data Acquisition subsystem of *GESTALT* leverages a variety of data sources to ingest or tag objects and locations, including hand-labeled data in KML files, crowd-sourced object and location labels from OSM, and automatically generated object tags it extracts from Flickr images geolocated within the region of interest. Data Acquisition focuses on fusing data in various formats from various sources into a standard format so the data can be transformed and searched.

Ownership Assignment. The Ownership Assignment subsystem of *GESTALT* maps the *objects* to the *locations* they most likely belong to using a fuzzy clustering that allows objects to be assigned to multiple locations. We use DBSCAN [1] to cluster the objects on the first pass and prune out noisy objects that are unlikely to be associated with any particular location. The resulting cluster centroids are then mapped to locations based on proximity using a nearest-neighbor search on a KD-Tree, and objects are added to nearby clusters using an adjustable 'fuzziness' parameter. *GESTALT*'s assignment protocol accounts for uncertainty in tag coordinates of locations and objects and allows for the possibility that objects can be visible from multiple locations in the real world.

Concept Mapping. The Concept Mapping subsystem of *GESTALT* creates the data structures (inverted index, location-object concept map, object-object concept map) that enable successive pruning for spatial searching of objects and locations. The inverted index maps each object class to the set of locations that contain objects of that type, enabling membership queries, which prune the search space for downstream spatial searching. The location-object concept map treats the coordinates of the *location* as the division point on the north-south and west-east axes and maps the objects for each location to the corresponding quadrant they lie in with respect to the location. The object-object concept map (for each location) encodes the implicit directional relationships between the objects at that location using sparse matrices that can be recursively searched to eliminate candidate locations that do not match the spatial query specification.

Search. The Search subsystem of *GESTALT* enables the user to identify locations of interest based on partial knowledge about the collection and geospatial arrangement of objects at that location. The search process is designed for the way humans naturally conceptualize and describe locations and directions- by drawing a sketch map. Queries can be issued using keywords and cardinal relations (like Northwest, Southeast, etc.) or a pictorial specification, where objects can be added and visually arranged in a configuration that is matched against the concept maps. The Search subsystem supports fuzzy searching to return inexact matches when query terms are too narrowly defined and ranking to reflect the uncertainty in any given object's identification, tagging, and assignment to a location.

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 a last-mile search will be performed. Regions can be arbitrary and represent an administrative boundary like a city, suburb, or general geographic area. 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.

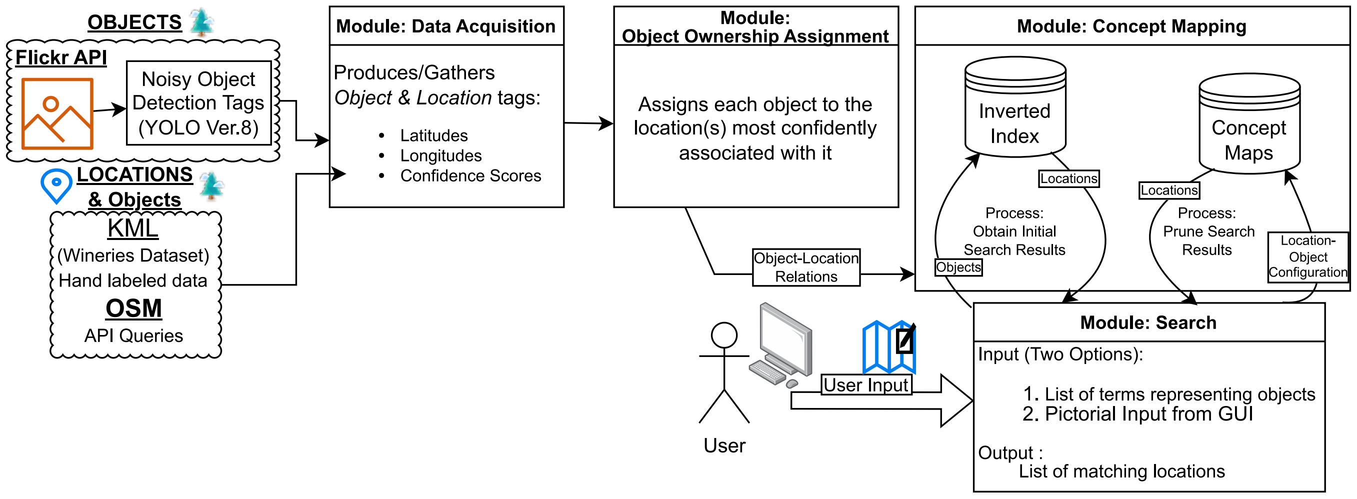

OBJECTS
Flickr API
Noisy Object Detection Tags (YOLO Ver.8)
LOCATIONS & Objects
KML
(Wineries Dataset)
Hand labeled data
OSM
API Queries
Module: Data Acquisition
Produces/Gathers Object & Location tags:
Latitudes
Longitudes
Confidence Scores
Module: Object Ownership Assignment
Assigns each object to the location(s) most confidently associated with it
Object-Location Relations
Module: Concept Mapping
Inverted Index
Concept Maps
Locations
Objects
Process: Obtain Initial Search Results
Process: Prune Search Results
Location-Object Configuration
Module: Search
Input (Two Options):
1. List of terms representing objects
2. Pictorial Input from GUI
Output :
List of matching locations
User Input
User

Figure 1: *GESTALT* consists of a data collection module, an ownership assignment module, a concept mapping module, and a search module.

(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 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 1 contains a summary of the objects currently in *GESTALT*.

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.

Similar to the approach with object tags, we assign a confidence to each location tag ingested based on the source providing it.

3.3 Ingest Methods

3.3.1 Ground Truth Hand Labeled Tags. Hand-labeled objects and locations have been manually annotated by a trustworthy source and are assumed to be correctly labeled and geotagged. *GESTALT* accepts hand-labeled tags to allow for prior manual annotation work to be folded into the pipeline 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.

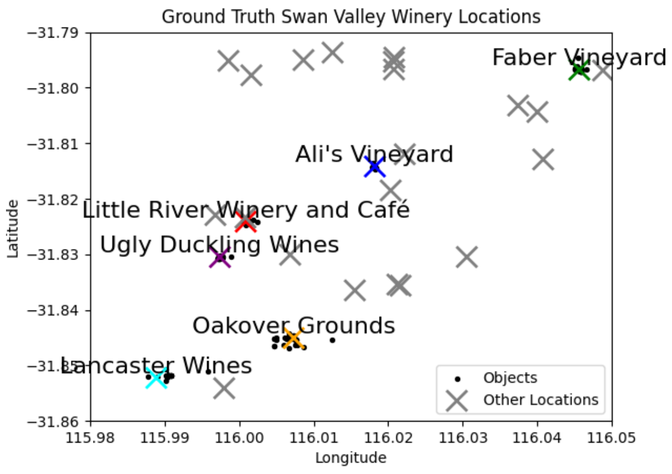


Figure 2: The Swan Valley Wineries dataset has six locations with annotated objects and another 30 without manually annotated objects but with confirmed location coordinates.

The wineries dataset tags are stored in Keyhole Markup Language [2] (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* [3] *version 7.3.* The data sources include on-the-ground knowledge, manual inspection of satellite imagery, street-view imagery, and publicly available area photos. 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

[2] https://developers.google.com/kml/documentation/kml_tut
[3] https://www.google.com/earth/about/versions/

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 six 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 [3] for crowd-sourced Geodata, including *Nodes* (point data for objects and locations), *ways* (line data for roads, creeks, railways, etc.), and *relations* (for relationships between elements, like suburb-state). While businesses, attractions, and other *locations* are commonly annotated by the OSM community, *objects* are more sparsely tagged as they are typically less interesting subject matter. To ingest objects, *GESTALT* pulls all nodes within a given bounding box that have at least one tag (like "name," "craft," etc.) and prunes them to remove results not likely referring to objects or that lack detail to be resolved. To ingest locations, *GESTALT* uses a similar process to objects, but this time excluding results that lack features like names, street addresses, and phone numbers. We assign each location or object tag ingested from OSM a confidence score of 0.75 because while any user can edit tags in the OSM database without review, the open-source community does have some degree of self-regulation [16], and prior work has examined the application of machine learning for detecting anomalous behavior in OSM edits [10].

3.3.3 Noisy Image-based Tags. The third and most important method by which *GESTALT* ingests objects is through automatic object detection. We query the Flickr API for images uploaded within a given region (bounding box) since January 1^{st} 2020. Given those images and their EXIF metadata, the Object Detection module uses pre-trained YOLO v.8 [4] to identify objects in each image from 80 classes (based on the COCO dataset [5]). Those objects become tags in *GESTALT*, with their coordinates determined by the geolocation of the image and their confidence score determined by YOLO's confidence at the detection step. For our experiments we enable safe-search and limit to public photos. For the Swan Valley Region we retrieve 462 images, and for the Washington DC Region, we pull 4,000 images.

Dataset	Type	Source	Labels	# Tags
Swan Valley Wineries[6]	Objects	KML file	Hand Labeled	146
	Objects	OSM B-Box	Crowd-Sourced	2466
	Objects	Flickr B-Box	Object detection	1893
	Locations	KML file	Hand Labeled	31
	Locations	OSM B-Box	Crowd-Sourced	308
Washington D.C.[7]	Objects	OSM B-Box	Crowd-Sourced	60123
	Objects	Flickr B-Box	Object detection	31065
	Locations	OSM B-Box	Crowd-Sourced	12179

Table 1: Summary of location and object datasets in *GESTALT*.

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*.

To support user queries, *GESTALT* must associate objects with their parent locations. We call this problem *Object Ownership Assignment* and define it as follows. Given a collection of locations and objects within a region, *Ownership Assignment* seeks to assign each object to its 'parent' location correctly. Ownership Assignment 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 associated tagged objects.

Further, the human eye does not see the world in regular grid lines like on a map. Hence, 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 a neighboring one. In this case, the object can be helpful 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 evaluate 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. 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 more extensive set of locations and a 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 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 2 contains the recall (and precision, where applicable) on those locations and objects.

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

[4]YOLO

[5]https://cocodataset.org/

[6]BoundingBox:['115.96168231510637', '-31.90009882641578', '116.05029961853784', '-31.77307863942101']

[7]BoundingBox:['-77.120248', '38.791086', '-76.911012', '38.995732']

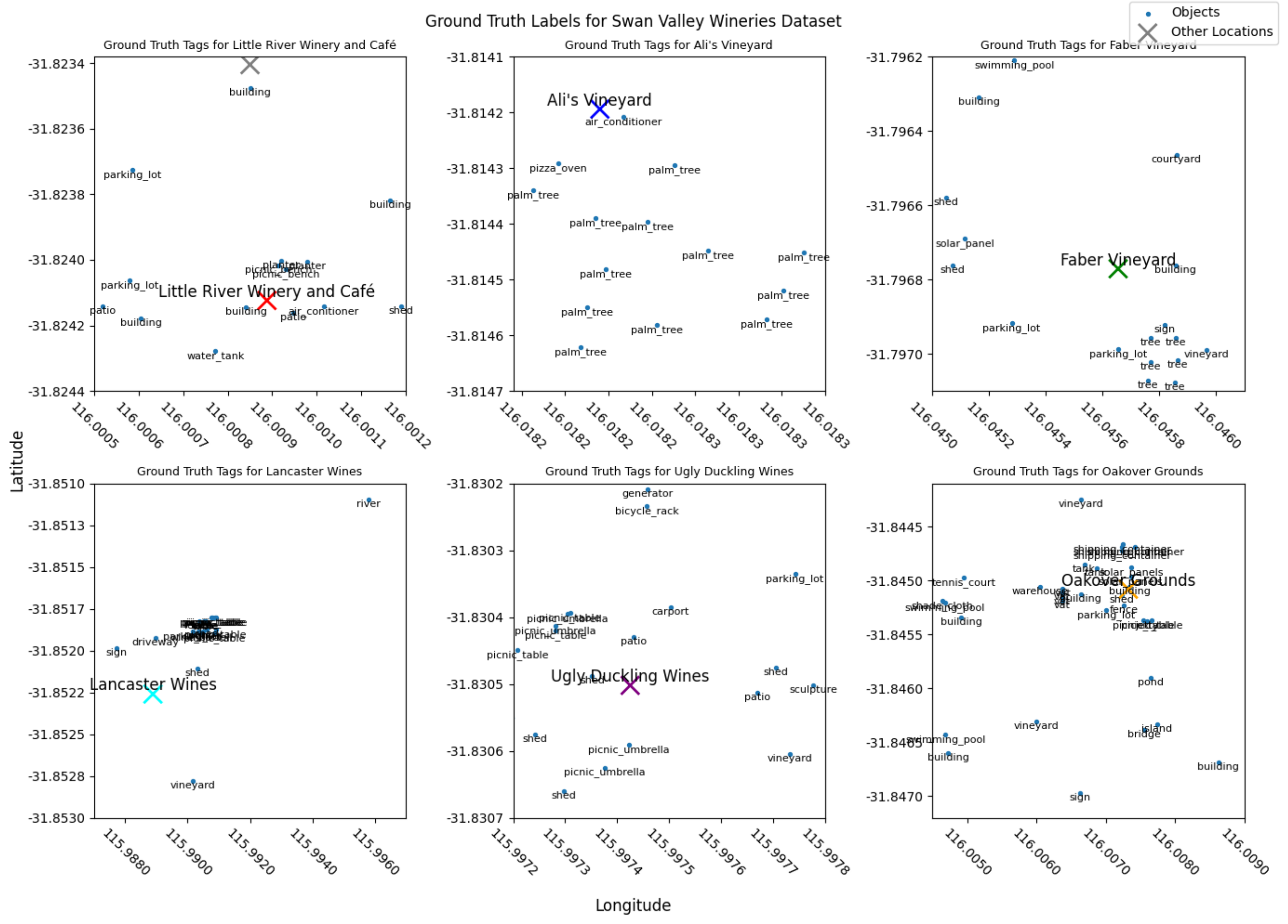


Figure 3: A representative view of the six wineries in the dataset shows the geospatial heterogeneity of object positions between locations.

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 its 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 the 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 helpful 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 increasing the parameter value, we can allow for objects to be assigned to multiple clusters if they are close to the centroid (within c% of the largest object-centroid distance in the dataset, the same value used to normalize the confidence scores).

We run all Swan Valley Wineries experiments with DBSCAN parameter $\epsilon = \frac{0.05}{6371}$ and DC experiments with $\epsilon = \frac{0.01}{6371}$. For both the MinClusterSize=3, choosing these values using the knowledge that the environment of interest (Swan Valley region) is semi-rural. The epsilon value is set to the equivalent of roughly 50m , a reasonable cluster neighborhood size for such an environment, and the minClusterSize is not critical given that fuzzy assignment allows objects to be added to nearby clusters, which overcomes too fine a clustering outcome.

4.3 Results

We show the Object-to-location assignment results for the Swan Valley Wineries and the *Combined* datasets in Figure 2. Increasing the fuzziness parameter c increases the chance that objects are assigned to multiple clusters, which improves recall on the noisier *Combined* dataset. As expected, increasing the likelihood that objects are assigned to multiple clusters in the less noisy Swan Valley Wineries dataset reduces the precision but, interestingly, does

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.l the predicted location for a cluster
O.c confidence score that object was correctly assigned
- - - - - - - - - -
procedure ObjectOwnershipAssignment(Locs, Objs)
    Clusters ← DBSCAN(Objs)
    for All C in Clusters except NULL cluster do
        C.x ← Calculate Cluster Centroid
        for O in C do
            $O.c \leftarrow 1 - normalize(dist(O, C.x)$
        end for
    end for
    for All O in NULL Cluster do
        $O.c \leftarrow 0.5$
    end for
    for All C in Clusters Except NULL Cluster do
        C.l ← Closest L in Locs to C.x
    end for
end procedure
```

Dataset	Method	Fuzzy Param	Precision	Recall
Swan Valley Wineries	Exact	$c = 0$	1.0	0.89
	Fuzzy	$c = 2$	0.85	0.89
	Fuzzy	$c = 4$	0.85	0.89
	Fuzzy	$c = 6$	0.85	0.89
	Fuzzy	$c = 8$	0.85	0.89
	Fuzzy	$c = 10$	0.85	0.89
Combined	Exact	$c = 0$	-	0.88
	Fuzzy	$c = 2$	-	0.88
	Fuzzy	$c = 4$	-	0.91
	Fuzzy	$c = 6$	-	0.94
	Fuzzy	$c = 8$	-	0.97
	Fuzzy	$c = 10$	-	0.98

Table 2: Object-to-location assignment results for Swan Valley Wineries and Combined datasets. Increasing fuzziness improves recall on the noisier *Combined* dataset.

not show the benefit in recall that it did for the noisier *Combined* dataset.

Overall, the object ownership results show that our method, while straightforward, handles the noisy object tags well, especially on the larger dataset. By using DBSCAN for the initial clustering, we observe a noise reduction effect as objects which do not belong to any locations are relegated to the NULL cluster (i.e., a mistagged singular object with nothing else around it for miles in any direction), and a similar recall on the ground-truth labels is achieved despite adding a considerable quantity of noisy object tags (in the *Combined* dataset).

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 while retaining the noise reduction effects of DBSCAN. We leave a detailed comparison of Object Ownership Assignment techniques as an interesting avenue of future study for the last-mile search problem.

5 CONCEPT MAPPING

TThe core of *GESTALT*'s geospatial search capability resides in the *Concept Mapping* component. Concept mapping extracts and explicitly encodes 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 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.

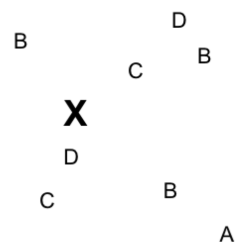
5.1 Object-Location relations

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 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 relative position to the location coordinates (4b), and then how queries are processed (4c).

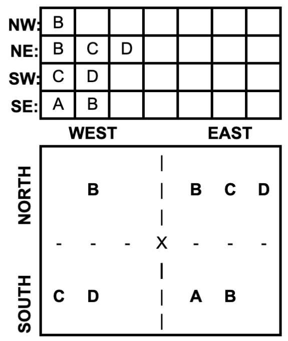
The query returns the candidate locations ranked by the number of query terms that match the appropriate quadrant with respect to the location.

5.2 Object-Object relations

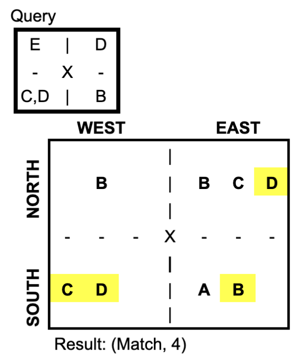
The other type of spatial relationships *GESTALT* supports searching are object-object relations, which are typically more complex than object-location relations. Object-object relations refer to a configuration of objects that relate to one another spatially. For example, a tree, a pond, and a sign 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 a 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 snowballs. Object-object relations lend themselves particularly well to pictorial query specification [15]. This method of specifying the spatial positioning of objects aligns nicely with how humans think about and describe landmarks in the world- by drawing a map. Figure 5 shows how we encode a location (5a) as a matrix of relative object positions (5b) and then how queries are processed (5c). The object-object approach to defining spatial configurations of objects is also more flexible than the object-location approach, which requires the user to know where each object lies in relation to the location and the general cardinal bearings.



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

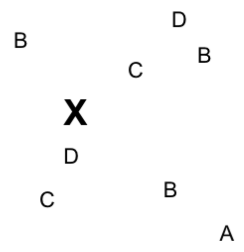


(b) The objects are binned into spatial quadrants based on their relative position to the location coordinates, X.

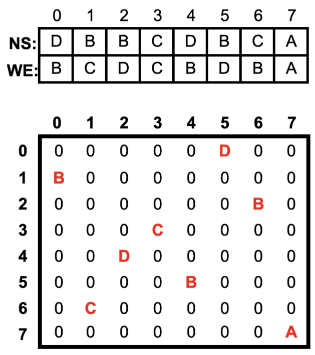


(c) Rank the locations by the number of query terms that are found in the correct quadrant with respect to the location.

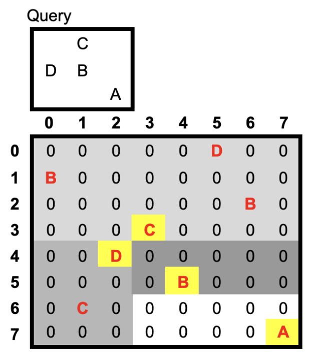
Figure 4: Generate and Query an Object-Location Concept Map.



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



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



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

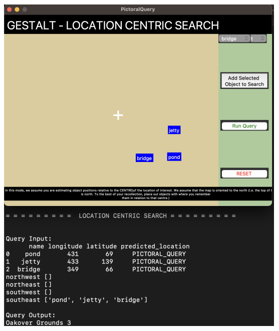
Figure 5: Generate and Query an Object-Object Concept Map.

5.3 Encoding the Spatial Relationships of objects and locations

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 location coordinates. The division of space aims to emulate how people relate the world to their position in it geospatially. **Object-object** concept maps encode objects in sparse NxN matrices, 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 (depicted in Figure 5b).

6 SEARCH

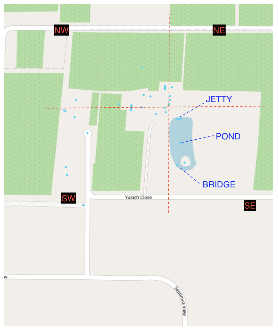
The core function of *GESTALT* is to perform last-mile search given partial or uncertain information. The user is assumed to know the general region of interest and some information about the objects at the location they seek. Under these conditions, the search problem can be framed in several ways, which we describe below in increasing order of complexity and utility.


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

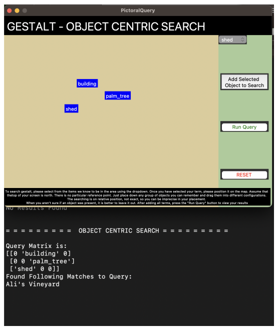
In this mode, we assume you are estimating object positions relative to the CENTRE of the location of interest. We assume that the map is oriented to the north (i.e. the top of t is north. To the best of your recollection, place out objects with where you remember them in relation to that centre.)

```
= = = = = = = = =  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 center of the location "+" to divide the search space into quadrants.



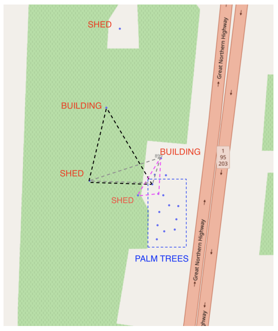
(b) The location matches the query since each object is Southeast of the location coordinates.


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 however they see fit.



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

Figure 6: Pictorial Query interface, where users position objects from the drop-down menu on the canvas by dragging the labels. Beneath each window is the encoded query and the search result returned by *GESTALT*. The interface allows users to mimic how they would explain features of the location to another person, in this case by drawing a quick map.

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
```

Algorithm 3 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
```

6.1 Membership Search

6.1.1 ***Exact membership search***. The simplest search function in *GESTALT* (Algorithm 2) takes a set of query terms representing objects the user knows are at a location, and performs the appropriate look-ups and set intersections to determine which locations are a match for containing *all* those objects.

6.1.2 ***Ranked membership search***. When the exact membership search returns a large number of hits, such as for a broad query (i.e. Which locations have a tree and a bench?) the ranking of those locations can help narrow the results. Using Algorithm 5, we aggregate the confidence scores from the object tagging and ownership assignment stages to determine the overall likelihood that a given location contains the object of interest. These scores are then aggregated per location for the relevant query objects, and the final scores determine the ranking of the results.

6.1.3 ***Fuzzy Membership Search***. When the exact membership search returns no matching locations, we use a fuzzy search procedure (Algorithm 3) to find a set of partial match locations based on the most discriminative object(s) in the list of query terms provided. The most discriminative terms are emphasized since rare objects are memorable and more uniquely identify locations than common objects do.

6.2 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 execute directly.

6.2.1 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.

6.2.2 Specifying the Query Pictorially. Humans naturally conceive of and express geospatial relationships visually. In keeping with a human-centric approach to spatial search, *GESTALT* employs a pictorial query interface that encodes the relative positioning of objects and locations to enable search over those relations. Figure 6a shows the location-object query interface, where the location being sought is represented at the center of the canvas and a user issues a query by positioning objects around that center point. Figure 6c demonstrates the object-object query interface, where users define the spatial arrangement of objects they want to query by placing the object names on the canvas space.

6.2.3 Searching the Concept Maps. To avoid unnecessary computation, the spatial searches are only performed after the initial membership check has been done to narrow the pool of locations to those that contain the right set of objects. Searching the object-location concept maps involves a simple iteration through the candidate locations, searching each one's NW/NE/SW/SE quadrants for the relevant query objects, and ranking the locations based on how many objects were found in the correct quadrant. Searching the object-object concept maps involves using the recursive grid search (Algorithm 4) to identify which locations have objects matching the configuration specified in the pictorial query, and returning the relevant set of results. The pictorial query is first converted into a concept map matrix using the process outlined in Appendix A.2 and the search terms in the matrix are ordered into a queue with the northernmost enqueued first, followed by the westernmost, then the second northernmost, and so on until all search terms are in the queue. Then, the grid search is employed to recursively prune the search space and quickly eliminate candidate locations that do not match the query configuration. This process is depicted visually in Figure 5.

Query Method	Metric	Results
Loc-Obj	Mean Precision	0.854
	Mean Recall	0.917
Obj-Obj	Mean Precision	0.721
	Mean Recall	0.875

Table 3: Spatial results across 24 ground-truth pictorial queries run on the *combined swan valley wineries* dataset.

6.2.4 Results on Ground Truth Spatial Queries. We hand labeled the ground truth responses for 24 spatial queries on object configurations for objects belonging to the 6 wineries in the Swan Valley Wineries dataset. To test the overall recall of *GESTALT*'s spatial search process in a noisy environment, we test those queries on the *Combined* dataset. The results (Table 3) show that *GESTALT* has a high recall on both methods of query specification, object-location and object-object. We expect precision to be low on these query results when compared to the ground-truth, since the *Combined* dataset includes many additional locations that were not considered during the hand-labeling. However, the precision we recorded is reasonable, which points to the discriminative power of spatial configurations of objects as a search constraint.

Algorithm 4 Recursive Grid Search

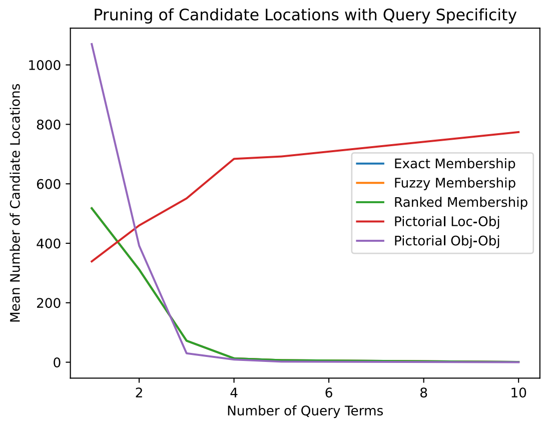
```
M A ConceptMap Matrix with objects or 0s; [0,0] is NW most point
L A NW to SE ordered list of objects to search for
D The alternating direction we prune from, 'north' (default) or 'west'
- - - - -
procedure RECURSIVEGRIDSEARCH($M$,$L$,$D$)
    if $L$ has only 1 item then                          ▷ Base Case
        if pop($L$) in $M$ then
            return $True$
        else
            return $False$
        end if
    end if
    $PruningPoint$ ← pop($L$)
    if $PruningPoint$ not in $M$ then
        return $False$
    end if
    $M'$ ← Prune all objects $D$ of $PruningPoint$
    return recursiveGridSearch($M'$,$L$,$D$.changeDirection())
end procedure
```

6.3 Scalability

Figure 7 shows the query response times on the Washington D.C. dataset (with over 91,000 objects and 12,000 locations) as the number of query terms increases. The queries tested were constructed using the 10 most common object classes: (e.g. crossing, traffic signals, street lamp, bus stop) to simulate worst case conditions where the user recalls no highly distinctive features of their target location. For the three membership searches (exact, ranked, and fuzzy), the response times decrease as the number of objects specified in the query increases (i.e. as the pool of possible locations meeting the query specification narrows), following the trend in Figure 7 which shows the aggressive pruning that takes place as the queries become more specific. Critically, this same effect is achieved for the object-object spatial search, where the recursive pruning of the search space quickly eliminates any candidates that are not viable matches to the pictorial query specification. The location-object search does not show this effect because it counts the number of objects matching the query configuration and uses that to rank candidates, so no early stopping is done when the right object is found to the wrong cardinal direction of the location.

7 RELATED WORK

GESTALT is designed to take a human-centric approach to geospatial search, borrowing principles from human spatial cognition and studies of the way humans navigate the physical world and conceive of objects on a map. We address related work in human spatial reasoning and pictorial querying, and



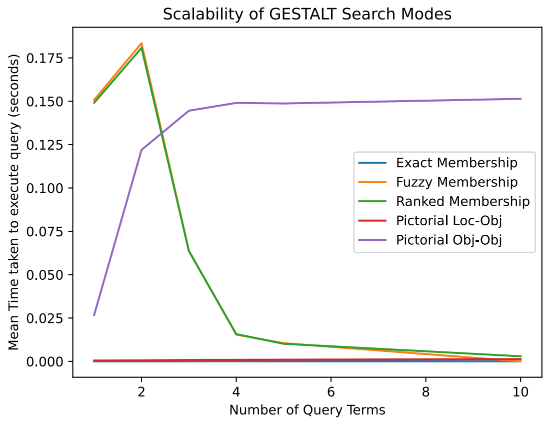


Figure 7: Number of candidate locations and query response times for queries on the Washington D.C. Dataset (12,179 Locations, 91,188 objects).

then compare *GESTALT* with the closest geospatial search tool available currently.

7.1 Human Spatial Cognition and Reasoning

According to criminology, psychology and neuroscience, people tend to anchor memories on objects they see while experiencing a location [5], and use them to find it again. Even robots we design to do perception need visual anchoring [11]. These principles underpin *GESTALT*'s human-centric approach to last-mile spatial search, wherein the goal is to retrieve locations within a region given information about their objects. This forms the reciprocal problem to the online game *Geoguessr* [8], which challenges users to determine the broad region of a photograph based on the visible objects. The success of *GeoGuessr* shows that enough objects, even common ones, have the power to substantially prune a large search space down to just a few, or even one candidate region. *GESTALT*'s approach to geospatial search closely follows the *Winthrop Method* [6] which was developed in Northern Ireland in the 1970s to detect clandestine weapons caches. The core principle behind this method is that humans form mental models of terrain based on objects in the environment, and then use those to navigate. Other research has show that When revisiting locations, the objects can even serve as keys to other memories of that location [8].

[8]Geoguessr Website

7.2 Pictorial Querying

The core spatial search functionality of *GESTALT* leverages a pictorial query interface to bridge the gap between the user's mental model of objects, and the geospatial data available to search. The idea of pictorial querying was developed by Soffer and Samet [15]. 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 Pictorial 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 [13] 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 [12] 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, which is NP-Complete when formulated as a subgraph matching problem [2]. *GESTALT* addresses the complexity issues by successively pruning the search space, beginning with the last-mile search region, pruning out any locations that fail to contain the query terms, and then commencing the spatial search, which involves recursively pruning the search space even further, until a a candidate location is determine to match or not match the configuration specified in the pictorial query.

7.3 Automated Geospatial Search

In recent months the Open Source Intelligence platform *Bellingcat*[9] released *Bellingcat OSM Search*[10] [18], a tool also aimed at pruning search space using tagged objects. Using drop-downs and sliding bars, a user can specify a text-based query for OSM tagged objects that appear within an adjustable distance from each other inside a region of interest. Although aimed 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) Their tool only accounts for the distance between objects, not their geographic configuration. They key contribution of *GESTALT* is allowing users to leverage information they remember about how the objects relate to each other spatially to improve the search result.
(2) Their tool is limited to data tagged in OSM, which is sparse for object tags that give *GESTALT* much of its pruning power.
(3) Their tool does not rank results, which is critical when object tags are noisy and users are uncertain about the partial information they recall about a location.
(4) Their tool returns 0 results if a set of search terms fails to match on every term in the window of interest, which is problematic when dealing with noisy, incomplete data where the likelihood of any given object being tagged is not very high. On the other hand, *GESTALT* uses a fuzzy matching procedure to return a partial match based on the most discriminative search term provided when the entire set yields no results, taking the approach that a partial answer is better than no answer.

8 CONCLUSION

The last-mile search problem refers to the bottleneck faced by users who want to find a location that matches some set of criteria that is not easily query-able with a GIS tool. This typically includes visual landmarks or terrain features that comprise a partial or uncertain set of information about the location of interest. Solving the last-mile search problem without manual inspection of images requires spatial search techniques beyond

[9]Bellingcat Website
[10]Bellingcat OSM Search Tool

just accommodating queries for collections of objects in close geographic proximity to each other. Taking inspiration from the way humans recall and search for information, we developed *GESTALT*, a pipeline that enables querying for locations based on the spatial configuration of nearby objects given partial information about them. Our method includes automatically detecting objects from geotagged images, clustering object tags to fuzzily assign objects to (possibly multiple) nearby locations, mapping objects into a format that enables pictorial querying on spatial relations between them, and performing probabilistic search over locations to return partial matches when the search constraints are too narrow, and ranked results when they are too broad. *GESTALT* shows high recall on the ground truth benchmark dataset we contribute and easily scales to the larger, noisier datasets we tested it on. This work invites several new avenues of research in improving human-centric spatial search. Some examples include developing techniques to handle querying across additional dimensions, like time, object attribute values, and object size, scale, quantity. Advances in computer vision can also be leveraged to quantify and adjust for uncertainty in object position within an image scene using the camera's bearing information [4, 7, 9, 14], which we already incorporate into some of our datasets.

REFERENCES

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Portland, Oregon) *(KDD'96)*. AAAI Press, 226–231.

[2] Andre Folkers, Hanan Samet, and Aya Soffer. 2000. Processing pictorial queries with multiple instances using isomorphic subgraphs. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, Vol. 4. IEEE, 51–54. https://ieeexplore.ieee.org/iel5/7237/19521/00902863.pdf

[3] Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive computing* 7, 4 (2008), 12–18. https://ieeexplore.ieee.org/iel5/7756/4653458/04653466.pdf

[4] James Hays and Alexei A Efros. 2008. IM2GPS: estimating geographic information from a single image. In *2008 ieee conference on computer vision and pattern recognition*. IEEE, 1–8. https://ieeexplore.ieee.org/iel5/4558014/4587335/04587784.pdf

[5] Jason Helbing, Dejan Draschkow, and Melissa L.-H. Võ. 2020. Search superiority: Goal-directed attentional allocation creates more reliable incidental identity and location memory than explicit encoding in naturalistic virtual environments. *Cognition* 196 (2020), 104147. https://doi.org/10.1016/j.cognition.2019.104147

[6] David Keatley, Chris O'Donnell, Brendan Chapman, and David D Clarke. 2021. The psycho-criminology of burial sites: developing the winthropping method for locating clandestine burial sites. *Journal of Police and Criminal Psychology* (2021), 1–10.

[7] Yang Liu, Jie Jiang, Jiahao Sun, Liang Bai, and Qi Wang. 2020. A survey of depth estimation based on computer vision. In *2020 IEEE Fifth international conference on data science in cyberspace (DSC)*. IEEE, 135–141. https://ieeexplore.ieee.org/iel7/9169739/9172417/09172861.pdf

[8] Jonathan F Miller, Markus Neufang, Alec Solway, Armin Brandt, Michael Trippel, Irina Mader, Stefan Hefft, Max Merkow, Sean M Polyn, Joshua Jacobs, et al. 2013. Neural activity in human hippocampal formation reveals the spatial context of retrieved memories. *Science* 342, 6162 (2013), 1111–1114. https://www.science.org/doi/pdf/10.1126/science.1244056

[9] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. 2021. Deep learning for monocular depth estimation: A review. *Neurocomputing* 438 (2021), 14–33. https://www.sciencedirect.com/science/article/pii/S0925231220320014

[10] Peter Mooney, Marco Minghini, et al. 2017. A review of OpenStreetMap data. *Mapping and the citizen sensor* (2017), 37–59. https://library.oapen.org/bitstream/handle/20.500.12657/31138/1/637890.pdf#page=46

[11] Miguel Oliveira, Luís Seabra Lopes, Gi Hyun Lim, S. Hamidreza Kasaei, Ana Maria Tomé, and Aneesh Chauhan. 2016. 3D object perception and perceptual learning in the RACE project. *Robotics and Autonomous Systems* 75 (2016), 614–626. https://doi.org/10.1016/j.robot.2015.09.019

[12] Hanan Samet and Aya Soffer. 1996. Marco: Map retrieval by content. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18, 8 (1996), 783–798. https://ieeexplore.ieee.org/iel1/34/11231/00531799.pdf

[13] Hanan Samet and Aya Soffer. 1998. Magellan: Map acquisition of geographic labels by legend analysis. *International journal on document analysis and recognition* 1 (1998), 89–101. https://link.springer.com/content/pdf/10.1007/s100320050009.pdf

[14] Noah Snavely. 2011. Scene reconstruction and visualization from internet photo collections: A survey. *IPSJ Transactions on Computer Vision and Applications* 3 (2011), 44–66. https://www.jstage.jst.go.jp/article/ipsjtcva/3/0/3_0_44/_pdf

[15] Aya Soffer and Hanan Samet. 1997. Pictorial query specification for browsing through image databasess. In *Proceedings of the Second International Conference on Visual Information Systems*. 117–124. https://www.cs.umd.edu/~hjs/pubs/soffervlc.pdf

[16] John E Vargas-Munoz, Shivangi Srivastava, Devis Tuia, and Alexandre X Falcao. 2020. OpenStreetMap: Challenges and opportunities in machine learning and remote sensing. *IEEE Geoscience and Remote Sensing Magazine* 9, 1 (2020), 184–199. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9119753

[17] Steven M Weisberg and Nora S Newcombe. 2016. How do (some) people make a cognitive map? Routes, places, and working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 42, 5 (2016), 768.

[18] Logan Williams. 2023. Finding Geolocation Leads with Bellingcat's OpenStreetMap Search Tool. *Bellingcat Blog* (May 2023). https://www.bellingcat.com/resources/how-tos/2023/05/08/finding-geolocation-leads-with-bellingcats-openstreetmap-search-tool/

A APPENDIX

A.1 Membership Search Algorithms

Algorithm 5 Rank

```
QT a list of query terms to search for
II an inverted index with objects as keys and locations as values
- - - - -
procedure RANK($S$,$QT$,$II$)            ▷ $S$ results from $Q$ query terms in $II$
    $LR$ = Empty Ordered Dictionary                  ▷ Location Rank
    for Each $L$ in $S$ do                                 ▷ Location
        $p$ = 1
        for $O$ in $QT$ do
            $p$ = $p$x$OP$                   ▷ $OP$ = Object Probability
        end for
        $LR[L]$ = $p$
    end for
    return $LR$ ordered most to least likely
end procedure
```

A.2 Spatial Search Algorithms

Algorithm 6 Creating a Concept Map

```
T a Location Table with ID, Name, Lat & Long of its objects
- - - - -
procedure CREATECONCEPTMAP($T$)
    LonList, LatList = []                                    ▷ Lists
    $M$ = [[][]]                                    ▷ A matrix of 0s
    for $R$ in $T$ do                  ▷ Where $T$ sorted North to South
        Add R.$ID$ to $LatList$           ▷ $R.ID$ is the obj unique ID
    end for
    for $R$ in $T$ do                   ▷ Where $T$ sorted West to East
        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
```

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
    $Traversed$ = []
    for $r$ in range ($R$+$C$)-1 do        ▷ $R$, $C$ = Number of rows and columns
        for $i$ in range $r$+1 do
            if $i < R$ and $r$-$i < C$ and $M[i][r-i]$ != 0 then
                Add $M[i][r-i]$ to $Traversed$
            end if
        end for
    end for
    return $Traversed$
end procedure
```

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
```