



**University of
Nottingham**

UK | CHINA | MALAYSIA

Validating Matches between Spatial Objects

Guangdi Hu

20029929

scygh1@nottingham.edu.cn

Computer Science with Artificial Intelligence

Supervised by Heshan Du


heshan.du@nottingham.edu.cn

COMP3050 Academic Year 2020 - 2021

Department of Computer Science University of Nottingham Ningbo China

Acknowledgements

I hereby declare that this dissertation is all my own work, except as indicated in the text.

Signature: 
Date: 24 / 04 / 2021

Abstract

The increasing application of geospatial information in different industries raised the requirement of integrating geospatial data from multiple sources. The key step of the integration process is identifying the correspondence between objects in different datasets, or called spatial object matching. Many matching algorithms and methods have been proposed to solve this task using different measures such as geometry, topology and context. Since different geospatial data sources have different focuses and contain different types of information, the performance of one matching method may vary in different use cases. However, few techniques can help people pick up the best-performing matching method for a certain use case, and the common way may involve manual check and comparison of matching results. This project aims to design a validation algorithm which can automatically check correctness of the matching results. The desired algorithm can be used as a unified standard to evaluate the performance of matching methods, helping people identify the most appropriate matching method or the optimal parameter configuration.

Contents

1	Introduction	1
1.1	Background and Motivation	1
1.2	Aim and Objectives	3
1.3	Dissertation Outline	3
2	Background and Related Works	4
2.1	Geographic Data Representation	4
2.1.1	Primary Geographic Data Formats	4
2.1.2	Geographic File Formats	5
2.2	Geographic Information System	5
2.3	Geographic Object Matching Methods	6
3	Methodology	9
3.1	Algorithm Overview	9
3.2	Context Similarity	10
3.2.1	Defining the Context	11
3.2.2	Angle Difference Measure	11
3.2.3	Sequence Order Measure	12
3.3	Object Similarity	13
3.4	Validating Input Matches	13
3.4.1	Forward Traversal	13
3.4.2	Backward Rectification	14
3.5	Detecting Missing Matches	14
3.6	Pseudo Code of Validation Algorithm	15
4	Implementation	19
4.1	Choosing GIS Platform	19
4.1.1	About OpenJump	19
4.2	Implementation Details	20
4.2.1	Matching Plug-in	20
4.2.2	Validation Plug-in	22
4.2.3	Scrutinize Plug-in	24
5	Experiments and Evaluation	25
5.1	Experiment Setting	25
5.2	Context Scope	26
5.3	Angle Tolerance	28
5.4	Context Similarity Weight	30
5.5	Validation Threshold	32
5.6	Conclusion	33
6	Project Management	34
6.1	Project Management	34
6.2	Reflection	36

7	Summary	38
7.1	Contribution	38
7.2	Future Works	39
A	Terminology	42
B	Experiment Data	44
B.1	Context Scope	44
B.2	Angle Tolerance	46
B.3	Context Similarity Weight	47
B.4	Validation Threshold	49
C	Meeting Records	50

List of Figures

1.1	Map of lake Pontchartrain area (data from tapiquen-sig.com)	2
1.2	Map of lake Pontchartrain area (data from statsilk.com)	2
1.3	Overlay the two map of lake Pontchartrain area	2
2.1	Vector map	4
2.2	Raster map (https://desktop.arcgis.com/zh-cn/arcmap/10.3/manage-data/raster-and-images/what-is-raster-data.htm)	4
2.3	Comparison between vector map and raster map (https://vimeo.com/53016715)	5
2.4	Attributes of a spatial object	5
2.5	Geometry information of a spatial object	5
2.6	Visualization of the spatial object	5
2.7	Single layer of buildings in GIS	6
2.8	Multiple layers overlapped in GIS	6
2.9	Use context to evaluate similarity [27]	7
2.10	Voronoi diagrams used in object matching [12]	7
3.1	Correct match M-M': M and M' are in the same context	10
3.2	Incorrect match M-N: the contexts of M and N have significant discrepancy	10
3.3	Connecting the objects involved in candidate match and its supporting match	12
3.4	Overlay the connecting lines to check the angle difference θ	12
3.5	The different orders of appearance between contextual objects	12
4.1	workbench-properties.xml	20
4.2	Class diagram of the implemented software	21
4.3	Angle difference	23
4.4	Code for calculating degree using coordinates	23
4.5	Sort supporting objects in anti-clockwise order	23
4.6	The code of object similarity calculator	24
4.7	Information printed by the scrutinize plug-in	24
4.8	Highlighted supporting matches	24
5.1	The variation of Precision and Recall of the three output sets according to the change of context scope	27
5.2	Smaller angle difference for remote match	27
5.3	Spotting using 3 supporting matches	28
5.4	Loose restriction using 1 supporting match	28
5.5	The variation of Precision and Recall of the three output sets according to the change of angle tolerance	29
5.6	Example of the part-of matching relation	30
5.7	The variation of Precision and Recall of the three output sets according to the change of context similarity weight	31
5.8	The same offset causes different influence on overlap proportion	31
5.9	The variation of Precision and Recall of the three output sets according to the change of validation threshold	32
6.1	Comparison between plan and actual progress	35

List of Tables

4.1	Several investigated GIS platforms	19
5.1	Two datasets used in experiments	25
5.2	Validation result using default parameter configuration: validation threshold is 0.8, context similarity weight is 0.8, context scope is 5, angle tolerance is 5 degrees	25
5.3	The evaluation of three output sets	25
B.1	Influence of context scope on performance of VIM for Arlington dataset	44
B.2	Influence of context scope on performance of DMM for Arlington dataset	45
B.3	Influence of context scope on performance of VIM for Santa Clara dataset	45
B.4	Influence of context scope on performance of DMM for Santa Clara dataset	46
B.5	Influence of angle tolerance on performance of VIM for Arlington dataset	46
B.6	Influence of angle tolerance on performance of DMM for Arlington dataset	46
B.7	Influence of angle tolerance on performance of VIM for Santa Clara dataset	47
B.8	Influence of angle tolerance on performance of DMM for Santa Clara dataset	47
B.9	Influence of context similarity weight on performance of VIM for Arlington dataset	47
B.10	Influence of context similarity weight on performance of DMM for Arlington dataset	48
B.11	Influence of context similarity weight on performance of VIM for Santa Clara dataset	48
B.12	Influence of context similarity weight on performance of DMM for Santa Clara dataset	48
B.13	Influence of validation threshold on performance of VIM for Arlington dataset	49
B.14	Influence of validation threshold on performance of DMM for Arlington dataset	49
B.15	Influence of validation threshold on performance of VIM for Santa Clara dataset	49
B.16	Influence of validation threshold on performance of DMM for Santa Clara dataset	49

Chapter 1

Introduction

This chapter introduces the general motivation and idea of this project, and the structure of this dissertation.

1.1 Background and Motivation

Geospatial information typically refers to the data which represents two-dimensional geographic features of the real world in the form of map, attached with different thematic information for different purposes (e.g. temperature and precipitation for weather analysis, land-use types for urban planning, names of buildings and roads for navigation) [8]. This kind of data provides users useful location-dependent knowledge, helping them linking lexical concepts with corresponding geographic positions or regions in the real world. With the development of techniques like Remote Sensing and Global Position System, geospatial information can be produced more easily and be utilized more widely [24]. Today, this kind of information plays crucial role in many areas, including military, tourism, logistics and business [17]. In daily life, people can also benefit from it, especially in some location-based services like online take-out platform, car-hailing apps, express delivery service, navigation service, etc.

The prevalent application and rising availability of geospatial information also indicates that this kind of data can be fetched from diverse sources, such as governments, commercial geographical industries and even normal citizens [24]. Different data sources may have different emphasises and strengths, but also have some limitations and drawbacks, which makes it necessary to integrate data from multiple sources or databases. During data integration, the lacked or inaccurate data of one database can be detected and supplied by other databases, which improves the quality and accuracy of these databases and facilitates the mutual enrichment of information [2]. For example, map conflation is a method to merge the data from two sources to generate a superior dataset which absorbs the strengths of both databases (e.g. higher spatial accuracy, detailed attribute information) and thus achieve a higher accuracy level [21]. Furthermore, geospatial data integration can also facilitate the propagation of updates from one database to others [2]. For example, it is convenient to update an authoritative database by integrating it with crowded-sourced or volunteered data (where end-users can contribute to the data edition and update), since the latter one is often rich of recent and user-based information [11].

It is worth noticing that the integration process is more than simply overlaying data in two datasets, as the relations between individual spatial objects should be explicitly identified [2]. Here, a spatial object is an entry in the geospatial database, representing a tangible geographic entity in real world (e.g. buildings, roads, mountains and rivers). A crucial step of data integration is spatial object matching, where corresponding spatial objects of different databases are identified and matched, with the consideration of their semantic and geometric similarities [2]. Shape comparison is a fundamental method for comparing geometric similarities of spatial objects [30]. Besides the shape, the position of a spatial object is also informative. Jun et.al. designed a spatial object matching method based on geographic context. To be more precise, it can identify corresponding spatial objects by considering the similarities of their relative spatial relation with neighboring objects [12]. Shan et.al. proposed a learning model for spatial object address matching, which can match

corresponding objects based on their semantic similarities calculated from their addresses [23].

One difficulty in spatial object matching is the non-uniform formats and standards for geospatial data representation in different institutions. In different datasets, the corresponding spatial objects might be represented by different geometries and at different levels of granularity and accuracy. This kind of discrepancy may prevent the matching methods from identifying the correspondence, influencing the quality of the matching result [7]. Figure 1.1 and Figure 1.2 represent the county map of the same geographical region in US with the geospatial data from two different sources. The county boundaries in Figure 1.1 contains more curves than the corresponding ones in Figure 1.2, which means the data of Figure 1.1 is more expressive and precision, as it records more detailed geospatial information in the real world. A more significant difference is that, the lake Pontchartrain, which is represented by the white hollow in Figure 1.1, disappears from the map in Figure 1.2. This discrepancy is caused by the different spatial object representing formats applied in these two datasets: while the area of the lake is excluded from the counties in the first dataset, the second dataset split the lake into pieces and merge them into the surrounding counties. These inconsistent representing standards, levels of granularity and precision would even influence human judgement, not to mention spatial object matching methods.

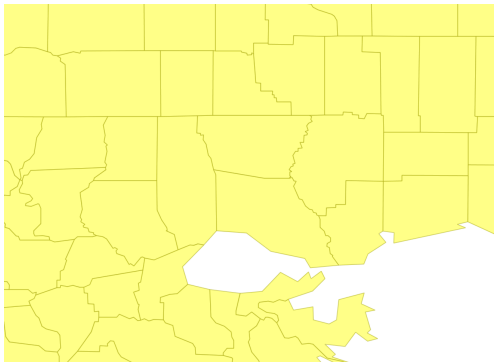


Figure 1.1: Map of lake Pontchartrain area
(data from tapiquen-sig.com)

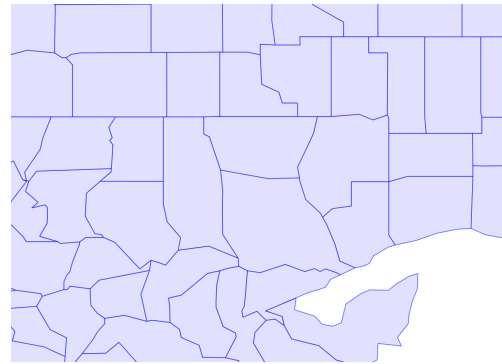


Figure 1.2: Map of lake Pontchartrain area
(data from statsilk.com)

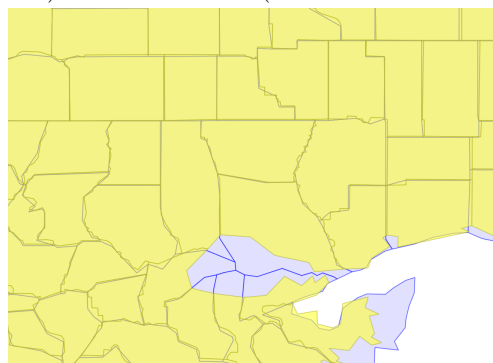


Figure 1.3: Overlay the two map of lake
Pontchartrain area

Object matching is the most important step of data integration, since its outcome will be used through further operations [24]. Thus, it is necessary to validate the correctness of generated matches and use the matching algorithm with the highest accuracy. However, in most articles about matching algorithms, some extent of ambiguity is left in the description of their validation processes: they only mentioned the precision and recall values of their algorithms, without explaining how these values are calculated [12, 23, 24]. This situation makes it difficult to compare the performance of different matching algorithms, as their accuracy evaluations may be carried out using different methods with different standards, which makes the results incomparable.

Thus, it is worthwhile to design a new validation method which can be applied to different matching algorithms and evaluate the quality of the generated matches, allowing a fair comparison of

their performance. At the same time, this validation method may also be used in production environment, lessening the human workload in verifying generated matches. The desired method should be versatile to be applied on different datasets and different matching results. In other words, the validation process should utilize the ubiquitous attributes over most of the geographic datasets. For example, the position attribute appears in all the geographic datasets, but the address information may be included in less datasets; if the validation methods utilize the latter type of information, its range of application becomes more limited. In addition, the validation method should assess both the correctness (i.e. how many of the generated matches are correct) and completeness (i.e. how many of the matches have been identified) of the matching results, since these two properties are both important in real use cases. Finally, the validation method should be free from human interference. Since human judgment is subjective and empirical, different people may make different decisions toward the same problem. If too much human operation is involved in the validation method, the evaluation result will significantly depend on the human operator, which involves more randomness into this process.

1.2 Aim and Objectives

The aim of this project is to develop a new validation method which can act as a unified accuracy evaluation standard for spatial object matching algorithms. By using this validation method, different matching algorithms can be evaluated with the same standard, making the comparison of their performance easier and fairer. Also, since the desired method is applicable to different matching algorithms, it can be used in the development of matching algorithms, liberating human from doing manual check and designing specific validation method for every new matching algorithm. In addition, the validation result can be used as a quality control mechanism in geographic object matching tasks.

The validation method should be able to check the correctness of the input matches and detect missing matches. In addition, the method should record and report the incorrect matches detected in the input set, improving the transparency and understandability of the validation process. An extra requirement for the expected validation method is automation. The method should be as automatic as possible, so that human influence on the evaluation result can be minimized and the efficiency of validation process can be improved.

The key objectives of this research project are:

1. Investigate existing geospatial matching algorithms and validation methods.
2. Design a validation algorithm, which inspects both correctness and completeness of a set of matches over two geographic datasets.
3. Implement the validation algorithm, use a graphical interface to visualize the spatial objects and matching relations.
4. Test and evaluate the validation method with data from different geographical regions, sources, and matching tools.

1.3 Dissertation Outline

This chapter has introduced the motivation of this project, as well as the aim and objectives. Chapter 2 supplies more background knowledge and related works in the task domain. Chapter 3 introduces the methodology to solve the problem, providing the detailed operations and pseudo code of the validation algorithm. Chapter 4 shows the implementation steps of the algorithm, where a Java program is developed according to the algorithm design. In chapter 5, the performance of the proposed algorithm is tested and evaluated using several sets of experiments. Chapter 6 outlines the time management and reflections on this project. At the end, chapter 7 summarizes the project outcomes and future works.

Chapter 2

Background and Related Works

This chapter outlines some background knowledge of the task domain from the shallow to the deeper. First, the format of geographic data and common geographic information processing tools are outlined. Then, the common methods of identifying spatial object correspondence and some matching algorithms with reference value are introduced.

2.1 Geographic Data Representation

2.1.1 Primary Geographic Data Formats

There are two primary geographic data formats: vector data and raster data. Vector data is made up of points, lines and polygons. A vector point is a simple coordinate which is generally represented by a pair of longitude and latitude with a spatial reference frame. Points are commonly used to represent features that are too small to be represented by polygons, for example, the cities whose boundaries are too small to be represented in a national map. Lines, which are constructed by connecting several points, are used to represent linear features, such as rivers, roads and railways. A polygon is formed by joining a set of points in order and close it. The area and boundary of a spatial object can be represented using a polygon.

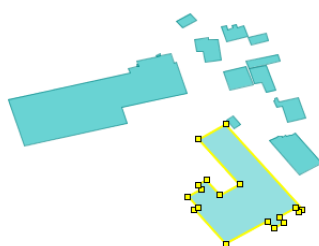


Figure 2.1: Vector map

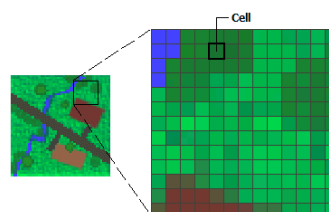


Figure 2.2: Raster map
(<https://desktop.arcgis.com/zh-cn/arcmap/10.3/manage-data/raster-and-images/what-is-raster-data.htm>)

Raster format is a data model for satellite data and remote sensing (RS) data. As Figure 2.2 shows, raster data is made up of pixels (also called grid cells), each of which has its own value or class for storing data. Thus, raster data is useful for storing continuous data, such as land use map, rainfall map and temperature map. However, since the cell is the smallest information representing unit, the accuracy and level of details of a raster map is limited by its grid size. The polygons in raster data are filled with pixels, even their boundaries are indicated by a set of adjacent pixels. If the grid size is too big for a small polygon, its outline would seem ambiguous.

Generally speaking, while vector data records each spatial object as a single entry in the file, raster data may split a spatial object into many grids (as Figure 2.3 shows). Since in most of matching

tools, each the spatial object is treated as the smallest indivisible unit, vector data fits better with the task requirement.

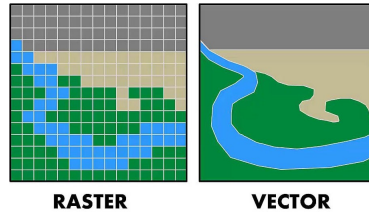


Figure 2.3: Comparison between vector map and raster map (<https://vimeo.com/53016715>)

2.1.2 Geographic File Formats

Depending on the data format, platform and use case, many different file formats could be used to store geographic data. Some vector data file formats are: Esri Shapefile (.SHP, .DBF, .SHX), OpenStreetMap OSM XML (.OSM) and Geography Markup Language (.GML). The common raster file formats include JPEG, PNG, IMG, TIF... The following paragraph will introduce one file format called shapefile.

Shapefile is a format for storing the nontopological geometry and attribute information of geospatial objects. To store a set of data, three different files with the same filename prefixes but different suffixes are used: the .shp file and the .dbf file separately stores the geometric information and any other attributes for each spatial object, and the .shx file records the positional indices of these objects in the corresponding order.

Figure 2.4 shows the attribute types and contents of one record from a .dbf file. The record of each spatial object starts with a header indicates the shape type, followed by a set of coordinates of points used to compose the geometry. The geometry information in a .shp file is in vector format, comprising points, lines and polygons. The .shp is written in binary code, and it can be translated it into Well-Known Text (WKT) format to improve readability. Figure 2.5 shows one record from a .shp file in the WKT format; Figure 2.6 illustrates the corresponding spatial object, which is composed by the set of points in that record.

...	FID	osm_id	code	fclass	name
■	227920	656857...	7210	nature_reserve	Cedar Swamp Rocks Tract

Figure 2.4: Attributes of a spatial object

POLYGON ((-75.6062365 39.441032, -75.6044877 39.4419269, -75.6029213 39.4427471, -75.6008184 39.4435425, -75.600121 39.4437828, -75.5996382 39.444313, -75.5990589 39.444719, -75.5978358 39.4453404, -75.596602 39.4460198, -75.5965161 39.4456801, -75.596248 39.4453487, -75.595851 39.4450836, -75.5953789 39.4449841, -75.594585 39.4449179, -75.5934262 39.4450173, -75.5928147 39.4451001, -75.5920529 39.4450587, -75.5915165 39.4451001, -75.5910874 39.4452079, -75.5904114 39.4455475, -75.5898858 39.4460695, -75.5897999 39.4464009, -75.5897999 39.4467323, -75.5899072 39.4471962, -75.590154 39.4476188, -75.5893171 39.4476685, -75.5887592 39.4477016, -75.5876327 39.4478839, -75.5869353 39.4480662, -75.5861736 39.4483727, -75.5859161 39.4482153, -75.5857229 39.4477265, -75.5852187 39.4473122, -75.5846715 39.446724, -75.5840707 39.4463926, -75.5837274 39.4462269, -75.5833197 39.4461192, -75.582397 39.4462021, -75.5816031 39.4465003, -75.5810344 39.4467654, -75.5803585 39.4473619, -75.5796183 39.4477514, -75.579291 39.4477783, -75.5790148 39.4476851, -75.5789638 39.4473122, -75.5790067 39.4469228, -75.5788243 39.4464341, -75.5787706 39.4461192, -75.5784702 39.4458375, -75.5774296 39.4455227, -75.5770862 39.4456138, -75.5768824 39.4457464, -75.5766893 39.4460032, -75.5765498 39.4460446, -75.5763781 39.4458127, -75.5760885 39.445647, -75.5752838 39.4452576, -75.5747151 39.4453238, -75.5743075 39.4455641, -75.5739266 39.4456904, -75.5736208 39.4456056, -75.5734277 39.4455807, -75.5730951

Figure 2.5: Geometry information of a spatial object

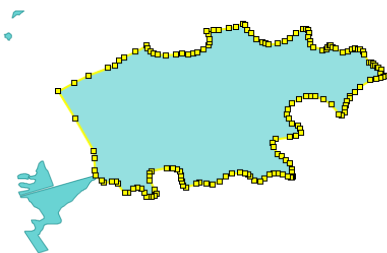


Figure 2.6: Visualization of the spatial object

2.2 Geographic Information System

Geographic Information System (GIS) is a kind of software for capturing, visualizing, editing and analyzing spatial and geographic data. Some famous GIS applications are: ArcGIS¹, OpenJump²,

¹<https://www.arcgis.com/index.html>

²<http://www.openjump.org>

QGIS³, GRASS GIS⁴ and uDig⁵. In a more general scope, some mapping software, such as Google Map, can also be seen as GISs⁶. Since GIS is used in the implementation stage of this project, it is necessary to introduce the common features and basic functionalities of GISs in advance.

Geographic data for a certain area is commonly divided into different datasets, each of which usually focus on one thematic information (e.g. buildings, roads, landuse). In GIS, these datasets are visualized as different layers of maps in the same window. By stacking and hiding these layers, different themes can be combined and displayed together, illustrating the relations between different attributes. Figure 2.7 and Figure 2.8 are two snapshots of a GIS, where the former figure only displays the layer of buildings, and in the later one, five different layers are overlapped.



Figure 2.7: Single layer of buildings in GIS

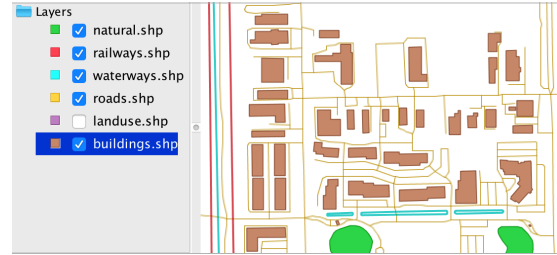


Figure 2.8: Multiple layers overlapped in GIS

Except for data visualization, GISs also provide many useful tools for data processing and analyzing. For example, some measuring tools can be used to calculate the distance or area on the map. Clustering and filtering tools allow the users to extract spatial objects with required attributes. Buffering tools can generate buffers surrounding the target object with specified radiuses, which is commonly used for analyzing the spatial objects' relations with their neighbours.

2.3 Geographic Object Matching Methods

As previously mentioned, the wide application of geographic information create the demand of transmission and integration of geographic information between different databases. During these processes, a crucial step is matching the corresponding geospatial objects in different datasets, which is also termed as data matching. Accurate matching result can improve the accuracy in the later conflation, support the data quality control [25], and even automate the data update process [1]. Thus, efforts have been taken in this area for decades. The following part of this chapter summarizes some related works about geographic object matching. However, the related validation methods are absent, since no researches or studies focus on this task has been found.

According to the supported degree of correspondence, matching methods can be classified as one-to-one matching, one-to-many matching and many-to-many matching [27]. One-to-one matching is the simplest approach for matching geospatial objects. In this approach, a spatial object may be matched with either one object from the other dataset, or holds no matching relation. Due to the different levels of details applied in object representation, sometimes one spatial object might be split into several more detailed parts and be recorded as multiple objects in another database. This kind of asymmetric matches in multi-scale datasets can be identified in one-to-many approach, which considers all the components as a composite during the matching process. Similarly, the many-to-many approach would check the potential matches between groups of object, and build a 'relation set' to describe their correspondence [2].

Though there are three types of matching approaches, their underlying criteria for judging the correspondence between (sets of) objects are common. According to Zhang and Meng [28], the

³<https://qgis.org/en/site/>

⁴<https://grass.osgeo.org/>

⁵<http://udig.refractory.net/>

⁶<https://www.geo-tel.com/gis-map-google-maps/>

essence of the matching process is gathering the information with pronounced geospatial and semantic similarities. Thus, the task of geospatial object matching could be understood as finding out the pairs of geographic objects with compelling similarity. In a survey, Xavier et al. [27] have categorized the existing matching methods, and listed the common types of similarity measures, which includes geometric measure, context measure, topologic measure, attribute measure and semantic measure.

Geometric measure focuses on the shape and position of spatial objects, evaluating the similarity using their properties like distance, area overlap, perimeter, inertial axis, orientation, etc [27]. For example, a shorter distance or larger area overlap between two objects from different datasets generally lead to higher similarity, implying the high possibility of their matching relation. At the same time, various alternative methods and criterion have been proposed for measuring the same property from different aspects of view, or measuring the properties on different types of geospatial objects. Taking distance as an example, Euclidean distance might be the most famous way to describe interval between points, and this technique is commonly applied to match point data [15]. However, this method is not applicable to polygon objects, since they are represented by sets of points. Thus, Hausdorff distance (HD) may replace Euclidean distance in this case.

HD measures distance between point sets by selecting the maximum value among minimum distances between points of two sets [20]. This ability promotes its wide use in polygon-polygon matching [10]. Also, there is a variation of HD called median HD, which aims to mitigate the influence of outlying part of the polygons by measuring the central disposition of the distance distribution [16]. In addition to points and polygons, lines are also widely used in geographic data representation (e.g. roads, railways, rivers, pipelines). The distance between oriented lines can be measured by Fréchet distance [5], which has already been widely applied in geographic area for dealing with tasks like coastline matching [14] and road matching [3]. To sum up, different properties can be used to measure the geometric similarity; and for each property multiple measuring methods may be available. Thus, the users should choose the measure depending on specific use cases.

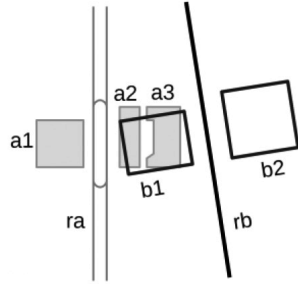


Figure 2.9: Use context to evaluate similarity [27]

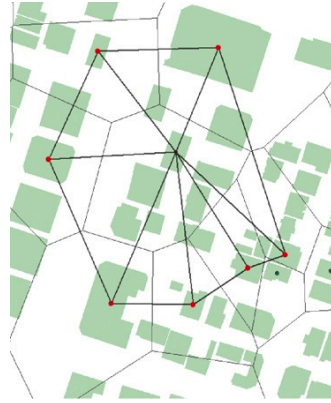


Figure 2.10: Voronoi diagrams used in object matching [12]

Context measure provides another way to evaluate the object similarity. In geographic domain, context refers to the spatial relations between objects [22]. Sometimes the geometric features may cause some ambiguity, and the context may evaluate the similarity more accurately [29]. In Figure 2.9, two geospatial object sets are overlaid (i.e. one is colored, another is represented by thick black lines). When only individual attributes are considered, $b1$ has high similarity with the compound of $a2$ and $a3$, since their area overlap and position are both similar. However, the similar context pattern between the two datasets (i.e. buildings separated by a road) indicates a potential rotation between their coordinate systems. Thus, it might be more fair to match $a1$ with $b1$, ra with rb , and $a2$ and $a3$ with $b2$.

The comparison of contexts is commonly carried out with the help of landmarks. Landmarks are pairs of matched spatial objects that are pre-selected to act as the reference system. In the

matching process, the similarity between a pair of objects is determined by the similarity of their relative spatial relations (e.g. direction, distance) to surrounding landmarks. Kim et al. proposed a context-based matching method which absorbs the concepts of Voronoi diagrams and triangulation [12]. In this method, the geographic space is split into small areas with landmarks in their center. To find the corresponding object of one spatial object, the area where it exists will be mapped to the other dataset to select all the objects in the corresponding area as the candidate counterparts. Then, for each candidate object, connecting lines will be drawn between the its centroids to the landmarks in adjacent cells, generating Voronoi diagrams constructed by triangles (As Figure 2.10 illustrates). Comparing with the Voronoi diagram of the being checked object, the candidate object whose Voronoi diagram has the highest similarity measured by triangulation (i.e. the sum of ratios of the areas and perimeters of triangles) will be selected as the matched object.

In addition to the previous two measure, other similarity measures also have their own strengths. Topologic measures evaluates objects using their topological relations. For example, the node with the same node index (i.e. the number of edges attached to the node) generally have high similarity [19]. The nature of the topologic measure determines it will have higher performance when dealing with net structure. As a result, this measure is mainly used for matching networks such as systems of rivers, roads and railways [18].

The last two similarity measures both utilize the non-geometric properties, or literal information of the geospatial objects. Attribute measure exploits the information provided by the labels of the spatial object. Depending on the type of the data, this measure can be further classified as numeric measures, list measures and text measures [27]. For numeric fields, the similarity can be measured using the natural metric in the form of $|m - n|$ [6], where m and n are the data value from the two being compared objects. As for text fields, the similarity (or distance) can be measured by edit-distance functions, which computes the minimal number of operations required to make two strings consistent [13, 4]. Comparing with attribute measure, the semantic measure exploits the literal information in a more conceptual and abstract way. In semantic measure, similarity is calculated from the “meaning” distance of information about descriptive concepts in real-world (e.g. classes and attributes of spatial objects). Usually, different concepts and attributes are organized as a taxonomy tree, where the similar concepts are close to each other, and the concept held by a parent node includes the concepts in its children nodes[9]. In this way, the similarity calculation can be implemented by counting the path length between two nodes, or by counting the depth of their least common parent node [26].

Chapter 3

Methodology

This chapter introduces the methodology to solve the validation task. The first section generalizes the procedure and criteria of the validation algorithm. The following two sections explain the new terms context similarity and object similarity separately. Then the detailed processes of validating input matches and detecting missing matches are expatiated in the next two sections. The last section contains the pseudo code of the whole validation algorithm.

3.1 Algorithm Overview

The validation algorithm receives two geographic object sets and a set of matches over them. All the input matches will be classified as either valid (i.e. the algorithm considers that this match correctly identified the corresponding objects) or invalid (i.e. the algorithm thinks the two matched objects represent different objects in the real world) depending on their evaluation results. Then, the algorithm tries to match each of the alone objects, which are not captured by any valid matches, with all their potentially matched objects. If a potential match shows enough similarity between its two objects, this match will be considered as a missing match. Finally, a valid match set, an invalid match set and a missing match set are output as the validation result.

All the unverified input matches or potential matches are called candidate matches. To check the validity of a candidate match, it will be evaluated from two aspects: the context similarity S_c inspects whether the objects involved in the candidate match have similar relations to their contexts; the object similarity S_o checks the internal correspondence between these two objects. These two similarities will be added up according to their predefined weights w_c and w_o . The resulted sum is called confidence level CL , which comprehensively quantifies the extent to which this candidate match can be proved as correct. If the confidence level of a match reaches the predefined threshold, this match will be evaluated as valid (or be considered as a missing match).

$$\begin{aligned} S_c, S_o &\in [0, 1] \\ CL &= w_c S_c + w_o S_o, \text{ where } w_c + w_o = 1 \\ match &\begin{cases} \text{valid,} & CL \geq \text{threshold} \\ \text{invalid,} & CL < \text{threshold} \end{cases} \end{aligned}$$

To validate the set of input matches, the above process should be applied on each single match. Starting from a random match, a forward traversal will validate all candidate matches in the manner of a width-first search. Since the calculation of context similarity involves information about other matches, including some assumptions of unprocessed matches (more details in next section), a context similarity may become inaccurate if the assumption it utilized is proved as incorrect later. To solve this problem, after the validation of each match, a backward rectification process will retrospect and remedy the influence of incorrect assumptions on this match in previous validation process.

After validating the input matches, the algorithm moves to the stage of detecting missing matches. The potential matches on alone objects (i.e. the spatial objects which are not included in any

match) will be inspected in the same way as above. In this process, the previously classified valid matches could be exploited as reference materials for the context similarity computation.

3.2 Context Similarity

The concept of context similarity derives from the context measure in geographic object matching algorithms. As previously mentioned, Kim et al. proposed a context-based spatial object matching algorithm. In that algorithm, several pairs of matches in ground truth are used as the benchmark; and two objects (i.e. landmarks) will be matched if they form similar network structures with their nearby benchmark objects. The similar idea can be applied in validating matches. Instead of assigning limited number of landmarks, all the input matches can act as the landmarks for each other, providing more detailed and accurate contextual information. In this case, when inspecting a candidate match, its nearby matches (i.e. valid matches) will be used as the landmarks (or called supporting matches of the candidate match), and the objects in these matches will be considered as the contexts of the involved objects in this candidate match. Finally, context similarity will be calculated to quantify the similarity between these two contexts.

In context similarity calculation, all objects are abstracted as their centroids. In Figure 3.1 and Figure 3.2, the dotted lines represent matches between spatial object (represented by points) from two datasets (represented by yellow and blue planes). Image that the algorithm is verifying match $M-M'$ and $M-N$, and all the remaining matches in the figures are included in the contexts of the candidate match. In Figure 3.1, the spatial relations between M to its context objects and N to its context objects are consistent, this correspondence between contexts demonstrates the correctness of match $M-M'$. On the contrary, in Figure 3.2 the match $M-N$ is judged as invalid since the contexts of M and N have significant difference. For example, the two objects in the supporting match $C-C'$ are expected to appear at the same direction of the candidate match $M-N$, however, C is at the left of M and C' is at the right of N . It is worth noting that, this example only shows some intuitive and qualitative comparison and analysis; in the validation algorithm, the context similarities will be quantified as concrete values.

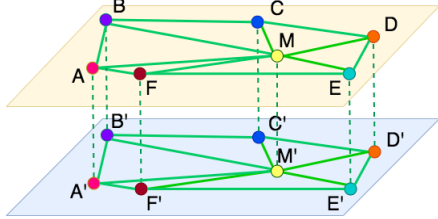


Figure 3.1: Correct match $M-M'$: M and M' are in the same context

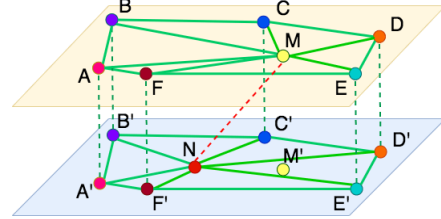


Figure 3.2: Incorrect match $M-N$: the contexts of M and N have significant discrepancy

The decision of applying context similarity in the validation algorithm is based on the following reasons: Firstly, context similarity theoretically provides a robust validation result, as the matches mutually prove the correctness of each other. Under this situation, when the context indicates an input match is invalid, this judgment can hardly be refuted since a wrong judgment means most supporting matches are also incorrect, which is less possible. In addition, inspecting matches in their contexts can resist the influence of offsets between coordinate systems of the two datasets. Even if the coordinate system error leads to the offsets on positions (i.e. described by longitude and latitude) of all the objects in a dataset, the relative spatial relations among the objects are still kept, which means the context similarity practicable.

Secondly, inspecting the geographic objects in context accords to human intuition. In daily life, people tends to remember and describe buildings use their relative relations to their surroundings, since empirically the combination of spatial relations from multiple surrounding entities and specify a unique building. Thus, applying context similarity calculation can be seen as a simulation of manual check.

Thirdly, the calculation of context similarity promotes a full exploitation of the input information. The algorithms of matching spatial objects and validating matching results are slightly homogeneous, as they are both answering the questions of whether two objects represent the same entity in the real world. Thus, when conflicting opinions appears, in order to make the validation result more persuasive than the input matches, the validation algorithm has to show its superiority over the matching algorithm. This can be achieved by exploiting the input matches. The matching process can only see the individual attributes of the geographic objects, but the validation process can also make use of the input matches. These input matches are informative and referable, since the current matching algorithms can generally have accuracy over 90%. Though some wrong matches are mixed among these landmarks, they may not mislead the validation results for the following two reasons: (1) the majority of input matches should be correct and they will dominate the validation result; (2) once a wrong match is detected, all the previous validation processes involve it will be re-computed by the backtrack system to eliminate its influence. As a result, exploiting this extra information makes the validation algorithm steps to a higher starting point than matching algorithms, which improves the persuasion of the validation result.

After explaining the reason of applying context similarity, the rest part of this section introduces the detailed processes of its calculation. First, the supporting matches are selected depending on their distances to the being checked candidate match; then, two alternative measures are provided to calculate the context similarity using the selected contexts.

3.2.1 Defining the Context

The first step of context similarity calculation is defining the range of context and finding the supporting matches within it. In Kim's algorithm, the context of an object consists of the landmarks from the Voronoi cells adjacent to the cell of that being checked object. In this validation algorithm this method is no longer appropriate. Instead, a parameter called context scope is defined to indicate the number of supporting matches required in context similarity calculation. Then for each candidate matches, the specified number of nearest valid or candidate matches will be selected as its supporting matches.

In the searching process, the distance between spatial objects is defined as the minimal Euclidean distance between two arbitrary points on their own outer contours. Between the two involved objects of the candidate match, the algorithm will select the one from the source dataset as the starting point, searching the nearest objects to it one by one. During the searching, the discovered alone objects (i.e. the objects which are not involved in any match) and objects involved in invalid matches are ignored; only the objects involved in valid matches or other unprocessed matches are counted into the context. The invalid matches are discarded since they contains no useful information. The unprocessed matches are assumed as valid in this process, otherwise at the beginning of the validation the algorithm may be terminated due to the lack of valid matches (more details in the Validating Input Matches section). After the context of the source set object is formed, a corresponding context in reference object set can be mapped using the matches on them.

3.2.2 Angle Difference Measure

Once the contexts of objects involved in the being validated match are defined, the context similarity can be quantified using either angle difference measure or sequence order measure. The context similarity evaluates the correspondence between the two candidate objects to their contexts. The spatial relation between two geographic objects can be represented by the combination of direction and distance information. But when measuring the relation between the candidate match and supporting matches, only the direction information is used. When multiple supporting objects are selected, an object in this reference system can be described using the combination of its directions to the reference objects. Thus, by checking whether the candidate matched objects have similar directional relations to each pairs supporting object in the two corresponding contexts, the correspondence between these two objects to their contexts can be determined.

Whether the directions between two supporting objects to the candidate objects could be considered as similar is determined by their angle difference. As Figure 3.3 and Figure 3.4 illustrate, two connecting lines between the centroids of the supporting objects and candidate objects are overlaid to generate an angle difference θ . If an angle difference θ is within the range of a predefined angle tolerance, these two directions are considered as similar, and this supporting match is said to acknowledge the validity of the candidate match. Finally, the context similarity is calculated as the number of supporting matches which acknowledges the candidate match divided by the total number of supporting matches.

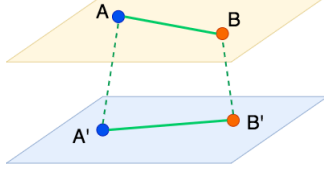


Figure 3.3: Connecting the objects involved in candidate match and its supporting match

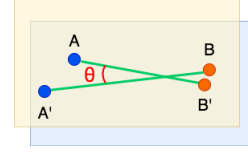


Figure 3.4: Overlay the connecting lines to check the angle difference θ

3.2.3 Sequence Order Measure

The angle difference measure evaluates the spatial relation between candidate objects to each of its contextual object, and then combine the individual results. On the contrary, this sequence order measure considers the context as a whole, analyzing the spatial relation between candidate objects and all contextual objects at the same time. In this measure, the algorithm simulates two viewers at the locations of candidate objects. These viewers will turn around in anti-clockwise direction to check whether all contextual objects appears in the same order in the two contexts.

For example, Figure 3.5 shows the contexts of the candidate matched objects in the two datasets. In the first dataset, the contextual objects come into the view of the viewer in the sequence of A, B, C, D, E, F, G. However, an offset on the candidate object in the second dataset leads to a slightly difference sequence, where the order of C' and D' is reversed to the order of their corresponding objects C and D. Then the algorithm tries to ignore the minimal number of contextual objects, in order to make all the remaining pairs of objects appear in the same order. The final context similarity can be calculated as the maximum number of in order supporting matches divided by the total number of supporting matches. In this case, either ignoring C-C' or D-D' can satisfy the requirement, so the context similarity is $(7 - 1)/7 * 100\% = 85.71\%$.

Comparing with angle difference measure, the sequence order measure may be more sensitive to the inconsistency in orderly contexts, where the spatial objects are frequently organized in lines. For example, sequence order measure may exploits the context better than angle difference measure in the case shown in Figure 3.5. Here supporting matches C-C' and D-D' are at very close directions to the candidate objects, so the directional information they provided are slightly redundant. And the offset between might be ignored if the resulted angle difference is covered by the angle tolerance. However, the sequence order measure will notice that the candidate objects appear at different sides of the line of C and D, which may convey a sign of non-correspondence between the candidate objects.

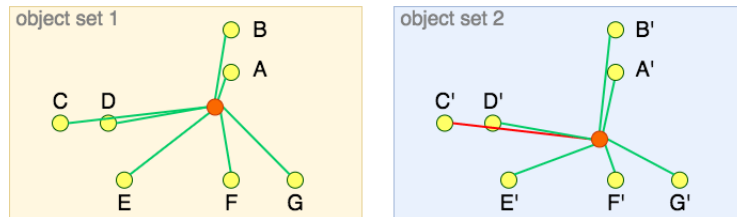


Figure 3.5: The different orders of appearance between contextual objects

3.3 Object Similarity

The context similarity focus on the mutual relation between geographic objects. However, other object-level attributes (e.g. area, perimeter, shape, type, label) also contains useful information for validation. These previously ignored features will be evaluated in the object similarity, providing a more comprehensive inspection on the match.

Another reason of employing object similarity is to mitigate the potential instability of the accuracy of object similarity. When abstracting two matched spatial objects into points, the difference of their shapes may result in an offset between the centroids. This error will propagate every time the relative spatial relation between the centroids and neighboring objects are considered, leading to an underestimated context similarity. In this case, the object similarity can inspect the match from other points of view, mitigating the errors in processing.

In the proposed validation algorithm, the object similarity is measured by the proportion of overlap area. First, the area of overlap between the two objects is calculated. Then, as the following equation shows, the object similarity is calculated as dividing the twofold overlap area by the sum of the areas of the two objects.

$$\text{object similarity} = \text{overlap area} * 2 / (\text{area1} + \text{area2})$$

It is worth noting that the reason of choosing area overlap proportion as the object similarity is to support the universality of the validation algorithm. Since shape is a basic attribute of geographic objects, measuring it instead of other attributes allows the practicability of the validation algorithm over most geographic datasets. However, this design does not restrict the use of other measures. Other measures (e.g. label, distance, shape) can also be used, as long as they are the common attributes of the two input datasets. Applying other measures, or even multiple weighted measures, may improve the accuracy of the object similarity evaluation.

3.4 Validating Input Matches

This section introduces the main logic of validating input matches. A width-first search is applied to process all input matches, classifying them as valid or invalid depending on their confidence levels. When selecting the context matches in the stage, all the invalid matches will be ignored, as the incorrect matches can provide no credible information about the context. The unprocessed matches will be assumed as correct, and be utilized together with other valid matches in context similarity calculation. Sometimes the assumptions (i.e. an unprocessed match is correct) may be overturned in the following validation process. In this case, a backward rectification procedure will be executed to recalculate the context similarities where the wrong assumption is involved.

3.4.1 Forward Traversal

In the forward traversal stage, the confidence levels of all the input matches will be calculated to estimate their correctness. While the calculation of object similarity is straightforward for all the matches, contradictions may occur in context similarity calculation. The accuracy of context similarity is based on the correctness of the selected supporting matches, which means to validate a candidate match, its surrounding matches should be validated first. However, this candidate match itself also act as part of the contexts of all its surrounding matches, so itself should be validated before its supporting matches.

To jump out of this trap, the forward traversal is designed as a loop of assume-prove processes: to facilitate the context similarity calculation, all the unprocessed matches will be assumed as correct; and these assumptions will be validated later. Starting from one match, the algorithm will ignore the surrounding invalid matches, and assumes the remaining matches are all correct. Based on these assumed-as-correct matches the context similarity can be calculated. After the validation process of one match, the next step is proving the correctness of the previous assumptions, i.e. validating the correctness of surrounding matches assumed as correct in previous validation processes. These matches are sequenced in a queue structure, being popped out to be validated one

by one. And during the validation of these matches, new assumptions will be made and relative matches will be pushed onto this queue, forming a loop until all the matches are checked. Since the elements in a queue structure are processed in a first-in-first-out manner, the assumptions which are made earlier (i.e. the matches discovered earlier) will be validated earlier as well, building up a width-first search. Seeing through the whole forward traversal, the validation process starts from one core and annexes surrounding matches layer by layer. In this processing order, each single validation operation can make best use of previously proved matches, since the matching relations at the inner side are all proved.

A leak of this forward traversal is that, the remote objects will never be added into the queue, since they are so far away from other objects that they can never be discovered in other objects' contexts. Taking the map of US counties as an example, if the forward traversal starts from the mainland, the Alaska and Hawaiian islands will never be reached. More specifically, due to the middle empty spaces, spatially segregated clusters of objects can hardly discover each other, which prevents the validation algorithm from processing all input matches. Thus, to fix this problem, after each round of forward traversal, the algorithm checks whether there is unprocessed object left. New forward traversals will be started on a randomly selected unprocessed matches, until all input matches are processed.

3.4.2 Backward Rectification

While the forward traversal runs smoothly if all the matches are correct, more complex situation may occur if some incorrect matches are discovered. Since these wrong matches had been assumed as correct during the calculation of context similarity of some previously inspected matches, the incorrectness of this match indicates some fake assumptions which may have caused mistakes in previous confidence level calculation and classification of matches. To remedy this defect, the backward rectification process is introduced to dynamically modify the confidence levels of previous matches in response to the appearance of every invalid match.

Match B is said to be dependent on match A, if A composed part of the context of B. If match A is determined as incorrect, the backward rectification will be activated and a mistake signal will be broadcasted to all the matches which depend on A. Once a match receives such a mistake signal, its context and supporting matches will be selected again and its context similarity will be recalculated. In this round, match A will be ignored and removed from the context, as it has already been marked as invalid. The signal receivers should extend the searching space to find an extra match to compose their new contexts. After all the receivers update their contexts, A is no longer involved in any context similarity calculation, and its influence is erased.

However, the backward rectification has not finished yet, as the re-check on previous matches may incur chain effect. If some matches are diagnosed as invalid in the backward rectification, they will broadcast the mistake signal further, since their correctness is the assumption of some other matches. At the end, if no new invalid match appears and the system come back into tranquility, the backward rectification will be terminated and the control will be moved back to the forward traversal.

3.5 Detecting Missing Matches

After the validation of input matches, the second stage of the validation algorithm is detecting the missing matches, which refers to the correct matches that are not included in the input match set. Empirically, these missing matches can only exists on alone objects (i.e. the objects without matches on them). In other words, for any missing match, at least of one side of it is a alone object either from the source or target object set. This assertion can be proved as following: (a) When considering the one-to-one matches the assertion can be proved by contradictions: if both sides (e.g. A, B') of a new match have already been contained in other matches (e.g. A-A', B-B'), this new match will build up an equivalence among four instances A, A', B, B'. According to this, A and B, which are two objects from the same dataset, represent the same real world object. However, this conclusion conflicts with the rule of geographic dataset that each real world instance

can only be represented once in a database. Thus, it is impossible to detect new matches upon two objects which are both involved in other matches. (b) When considering one-to-many matches (i.e. the part-of matching relations), though the parent object may have already been involved in other matches with its corresponding child objects, the other side of the new match, which represents a component of the parent object, must be alone object.

Thus, the first step of the detection process is identifying the sets of alone objects, since all the potential matches are hidden behind them. This target can be achieved by intersection operations between the source object set and the set of matched source objects, and between the target object set and the set of matched target objects. Then, each alone object (either from the source or target set) will be projected onto the map of the other geographic dataset. Any objects that overlaps with this projection are listed as its potentially matched objects. All the potential matches between the alone object and its candidate matched objects will be inspected in the same way of validating input matches. In this stage, backward rectification process is no longer needed, as all the supporting matches are selected from the previously validated input matches.

3.6 Pseudo Code of Validation Algorithm

The following pseudo code shows the detailed steps and operations in the validation algorithm. For some steps several alternative options are proposed, the pseudo code only expresses one possibility.

Algorithm 1 shows the pseudo code for validating input matches. All the input matches are traversed in the nested while loops. The inner loop (line 14) is in charge of the forward traversal. Starting with a queue contains a random unprocessed match, the inner loop repeatedly validate the head of the queue and push the discovered neighboring matches into the queue. The purpose of the outer loop (line 8) is to cope with spatially segregated datasets. If the input objects are split into several clusters, a single forward traversal (i.e. the inner while loop) may fail to reach all matches. Under this situation, the outer while loop will re-initialize the queue with an unprocessed match from the other clusters to activate the forward traversal again, until all input matches are processed.

Algorithm 1 This algorithm in charge of the validation of input matches. It executes a forward traversal on each spatially segregated clusters of matches to validate all the input matches in sequence.

Input: M match set; S : source spatial object set; T : target spatial object set

Output: a set of valid matches and a set of invalid matches

```

1: function FORWARDTRAVERSAL( $M, S, R$ )
2:    $threshold \leftarrow$  predefined value  $\in [0, 1]$ 
3:    $Wc \leftarrow$  predefined value  $\in [0, 1]$ 
4:    $Wo \leftarrow 1 - Wc$ 
5:   for  $match \in M$  do
6:      $match.discovered \leftarrow False$ 
7:      $match.res \leftarrow unknown$ 
8:   end for
9:   while  $M.undiscovered > 0$  do
10:     $startingMatch \leftarrow$  a random match from  $M.undiscovered$ 
11:     $startingMatch.discovered \leftarrow True$ 
12:     $queue \leftarrow$  empty queue
13:     $queue.push(startingMatch)$ 
14:    while  $queue.length > 0$  do
15:       $m \leftarrow queue.pop()$ 
16:       $context \leftarrow FINDCONTEXT(m, M)$ 
17:       $m.contextSim \leftarrow CONTEXTSIMILARITY(m, context)$ 
18:       $m.objectSim \leftarrow OBJECTSIMILARITY(m)$ 
19:       $m.confidenceLevel \leftarrow Wc * m.contextSim + Wo * m.objectSim$ 
20:      if  $m.confidenceLevel \geq threshold$  then
21:         $m.res \leftarrow valid$ 
22:      else
```

```

23:          $m.res \leftarrow invalid$ 
24:         BACKTRACK( $m, M$ )
25:     end if
26:     for  $sm \in context$  do
27:         if  $sm.discovered == False$  then
28:              $queue.push(sm)$ 
29:              $sm.discovered \leftarrow True$ 
30:         end if
31:     end for
32: end while
33: end while
34:  $validMatches, invalidMatches \leftarrow$  empty sets
35: for  $match \in M$  do
36:     if  $match.res == valid$  then
37:          $validMatches.add(match)$ 
38:     else
39:          $invalidMatches.add(match)$ 
40:     end if
41: end for
42: return ( $validMatches, invalidMatches$ )
43: end function

```

As previous sections mentioned, every time a match is judged as invalid, a backtrack rectification process should be activated (as line 23 in Algorithm 1). Algorithm 2 shows the detailed steps in this process. First, the contexts of all influenced matches are refreshed to exclude the newly discovered invalid matches. Then their context similarities will be re-calculated to update their confidence levels (line 5 to 6). If new incorrect matches are discovered during this process, this algorithm will be executed in a recursive way to deal with the chain effect (line 9).

Algorithm 2 Backtrack from an invalid input match, identify and re-check all the previous matches whose context similarity calculation involves this match

Input: m : a discovered incorrect match; M : set of input matches

Output: none

```

1: function BACKTRACK( $m, M$ )
2:      $dependencies \leftarrow$  all matches that depend on  $m$ 
3:     for  $m' \in dependencies$  do
4:          $context \leftarrow FINDCONTEXT(m', M)$ 
5:          $m'.contextSim \leftarrow CONTEXTSIMILARITY(m, context)$ 
6:          $m'.confidenceLevel \leftarrow Wc * m.contextSim + Wo * m.objectSim$ 
7:         if  $m'.confidenceLevel < threshold$  then
8:              $m'.correct \leftarrow False$ 
9:             BACKTRACK( $m', M$ )
10:        end if
11:    end for
12: end function

```

After the validation of input matches, the second stage of the validation task is detecting the missing matches, whose logic is explained in the following Algorithm 3. For all the alone objects from source and reference object, each of them will be projected to the opposite object set to filter all their potential matched object who overlap with their projection (line 6 to 10). Then, for each pair of alone object and its potential matched target, their context and object similarities will be calculated in the same way as that in the first stage. Finally, all the matches whose confidence level reaches the threshold will be marked as missing match and be output.

Algorithm 4 elucidates the steps to find out the context of a match. Here the context is defined as the K-nearest neighbouring matches to the input match. However, sorting the distances to all the matches from the input match set is time-consuming ($O(n^2)$). Instead, the nearest neighbour are filtered out by an increasing buffer. The radius of the buffer is initialized as the radius of

Algorithm 3 Detect missing matches which are not included in the input match set

Input: M match set; S : source spatial object set; R : reference spatial object set
Output: *missingMatches*: a set of detected missing matches

```
1: function DETECTINGMISSING( $M, S, R$ )
2:   missingMatches  $\leftarrow$  empty set
3:   for  $obj \in S \cup R$  do
4:     if  $obj$  is not involved in any match then
5:       candidates  $\leftarrow$  empty set
6:       for  $obj' \in$  the opposite object set of  $obj'$  do
7:         if  $overlap(obj, obj') == True$  then
8:           candidates  $\leftarrow obj'$ 
9:         end if
10:      end for
11:      for  $can \in candidates$  do
12:        match  $\leftarrow makeMatch(obj, can)$ 
13:        context  $\leftarrow FINDCONTEXT(match, M)$ 
14:        contextSim  $\leftarrow CONTEXTSIMILARITY(match, context)$ 
15:        objectSim  $\leftarrow OBJECTSIMILARITY(match)$ 
16:        if  $Wc * m.contextSim + Wo * m.objectSim \geq threshold$  then
17:          missingMatches.add(match)
18:          break
19:        end if
20:      end for
21:    end if
22:  end for
23:  return missingMatches
24: end function
```

Algorithm 4 Define the context of an input match using K-nearest neighbors approach

Input: cm : center match; M : input match set; S : source spatial object set
Output: *context*: a set of matches in the context of cm

```
1: function FINDSURROUNDINGMATCHES( $cm, M$ )
2:   min  $\leftarrow$  minimal number of required surrounding matches
3:   context  $\leftarrow$  empty set
4:    $obj \leftarrow$  spatial object of  $cm$  in  $S$ 
5:    $r \leftarrow m.radius$ 
6:   while Surr.size  $< min$  do
7:     Surr.clear()
8:     for  $m \in M$  do
9:        $obj' \leftarrow$  spatial object of  $m$  in  $S$ 
10:      if  $distance(obj, obj') \leq r$  AND  $m.res \neq invalid$  then
11:        context.add(m)
12:      end if
13:    end for
14:     $r \leftarrow 1.1 * r$ 
15:  end while
16:  return context
17: end function
```

the spatial object of the match (line 5). If the requested numbers of neighbours are reached by this buffer, these neighbours will be used as the context. Otherwise, the radius of the buffer will continuously increase (line 14), until it includes enough neighbours. If the buffer capture more objects than needed, the farthest ones will be removed. Still, identifying the farthest objects in such a small set is much easier than sorting the whole object set. In this way, for every buffer radius, all the matches will be processed once; and the buffer will get big enough within a limited numbers of rounds. Finally, the time efficiency of this algorithm is reduced to $O(n)$.

Algorithm 5 explains how context similarity is calculated in the angle difference manner. The x and y coordinates of the centroids can be directly obtained since the points are simply recorded as a coordinates in shapefile. The angle between the connecting lines between two objects and the horizontal can be obtained using *atan* function.

Algorithm 5 Calculate context similarity of input match using angle difference measure

Input: *cm*: center match; *context*: the set of supporting matches

Output: *cs*: context similarity of *cm*

```

1: function CONTEXTSIMILARITY(cm, context)
2:   tolerance  $\leftarrow$  the upper limit of tolerated angle difference
3:   c, c'  $\leftarrow$  the objects of m in source dataset and target dataset
4:   count  $\leftarrow$  0
5:   for match  $\in$  context do
6:     obj, obj'  $\leftarrow$  the objects of match in source dataset and target dataset
7:     xDiff  $\leftarrow$  obj.centroid.x - c.centroid.x
8:     yDiff  $\leftarrow$  obj.centroid.y - c.centroid.y
9:     angle  $\leftarrow$  atan(yDiff/xDiff)
10:    xDiff'  $\leftarrow$  obj'.centroid.x - c'.centroid.x
11:    yDiff'  $\leftarrow$  obj'.centroid.y - c'.centroid.y
12:    angle'  $\leftarrow$  atan(yDiff'/xDiff')
13:    if abs(angle - angle')  $\leq$  tolerance then
14:      count  $\leftarrow$  count + 1
15:    end if
16:  end for
17:  return count/context.size
18: end function

```

Algorithm 6 shows the steps of calculating object similarity using overlap proportion measure. Here the steps of area computation and overlap computation are omitted, since they are too complex and unrelated to topic of this project. In addition, most GIS platforms provide inner APIs for these kinds of operations.

Algorithm 6 Calculate the object similarity of two objects according to the proportion of their overlap area to their total area

Input: *m*: a match

Output: *os*: the object similarity between the spatial objects in *m*

```

1: function OBJECTSIMILARITY(m)
2:   obj, obj'  $\leftarrow$  the objects of m in source dataset and target dataset
3:   overlap  $\leftarrow$  the overlap area between obj and obj'
4:   return overlap * 2 / (obj.area() + obj'.area())
5: end function

```

Chapter 4

Implementation

In this project, the validation algorithm is implemented in Java, as a plug-in of an open-source GIS software called OpenJump. The details of the implementation and the design of the validation plug-in are described in this chapter.

4.1 Choosing GIS Platform

Instead of building up a software from the ground up, the validation algorithm was implemented based on existing open-sourced GIS platform. This decision is made for the following reasons: (1) each GIS platform provides a full kit of basic functions, such as file reading and data visualization, which can accelerate the development; (2) The source code of these GISs is published after thorough test and is maintained by thousands of volunteers, so the robustness of the framework can save much workload in debugging.

Platform	Open Source	Development Language
Mapline	no	–
QGIS	yes	C++, Python
GRASS GIS	yes	C, Python
OpenJump	yes	Java
uDig	yes	Java

Table 4.1: Several investigated GIS platforms

As Table 4.1 shows, several open source GISs have been investigated. Considering the personal familiarity of Java project development, the scope of choice was reduced to the two Java-based frameworks to speed up the development. The functionalities of OpenJump and uDig are about the same, as they both provide the desired tools. Considering the user experience, uDig owns higher popularity¹, and has a more professional user interface than OpenJump. However, the development of uDig is highly depends on the Eclipse framework. Developers have to spend extra time learning how to develop an Eclipse plug-in before starting to develop an uDig plug-in², which is an adverse factor to this time-pressured project. On the contrary, OpenJump provides a series of inner APIs for data manipulation, supporting a lightweight development. After comprehensive consideration, OpenJump was chosen as the framework of this project.

4.1.1 About OpenJump

OpenJump is an open source Java-based GIS. Its core framework provides all the basic functionalities, such as file reading, data visualization, layer style control, distance measurement and geographical object manipulation. At the same time, it provides many useful inner APIs to supporting further development, including: centroid generation, buffering, area calculation, etc.

¹data source: <https://www.saashub.com/compare-udig-vs-openjump-gis>

²information from: <https://gis.stackexchange.com/questions/19537/comparing-java-based-gis-programs>

Similar as most of open source GISs, OpenJump allow users to customize their desired features as plug-ins, without changing the core framework. In development environment, all the scripts of the new functionality should be placed under the same folder, whose path is recorded in a configuration file called workbench-properties.xml (as Figure 4.1 shows). When OpenJump is launched, it will scan the configurations and automatically load the new plug-in. Once the development of a plug-in is finished, it can be packed and published as a single JAR file. In production environment, to use a new plug-in, users can simply place its JAR file under a specific directory of their installed OpenJump, then the OpenJump will scan that folder and load all plug-ins in it.

```
<workbench>
  <plug-in>org.openjump.core.ui.plugin.validation.validationMethod</plug-in>
</workbench>
```

Figure 4.1: workbench-properties.xml

4.2 Implementation Details

The Java program is divided into three plug-ins, each of them in charge of one task: (1) A matching plug-in for generating and recording the matching relations from the input geographic object sets; (2) A validation plug-in for validating the generated matches and detecting missing matches; (3) A scrutinize plug-in for showing the detailed decision making process of validation process, explaining the reason behind each validation result.

The class diagram of the developed program is shown in Figure 4.2. In the framework of OpenJump, each geographic object is represented by an instance of the Feature class, and each object set is stored in a FeatureCollection object. As the middle-upper part of the diagram illustrates, the classes of the three plug-ins inherit an abstract class called AbstractUIPlugIn, which provides basic methods for adding the plug-in into OpenJump menu and visualizing user interfaces. Since these three plug-ins are segregated from each other, the class SharedSpace is developed for data transmission among them. The singleton design pattern is applied on SharedSpace, ensuring only a unique instance is used through the lifetime of the program. Once a plug-in generates some data which is required to be shared with other plug-ins (e.g. the set of matches), it will store the reference to the data in the shared space; later, other plug-ins can get the same SharedSpace instance using its static getInstance() method, and then access to the stored data.

More details of the class diagram is introduced in the following sections, each of which focuses on the function and implementation of one plug-in.

4.2.1 Matching Plug-in

Since the geographic object sets fetched from the Internet are not attached with existing matches, the methods for matching objects should also be implemented in order to generate the raw material for the validation process. In the original plan, though it is time-consuming, a separate object matching program should be developed to generate matches and store them into local files, which will then be read by the validation plug-in. However, after a search in the OpenJump plug-in market, an open-source matching plug-in³ was found to be an alternative way of completing this task.

³https://sourceforge.net/projects/jump-pilot/files/OpenJUMP_plugins/More%20Plugins/Matching%20PlugIn/

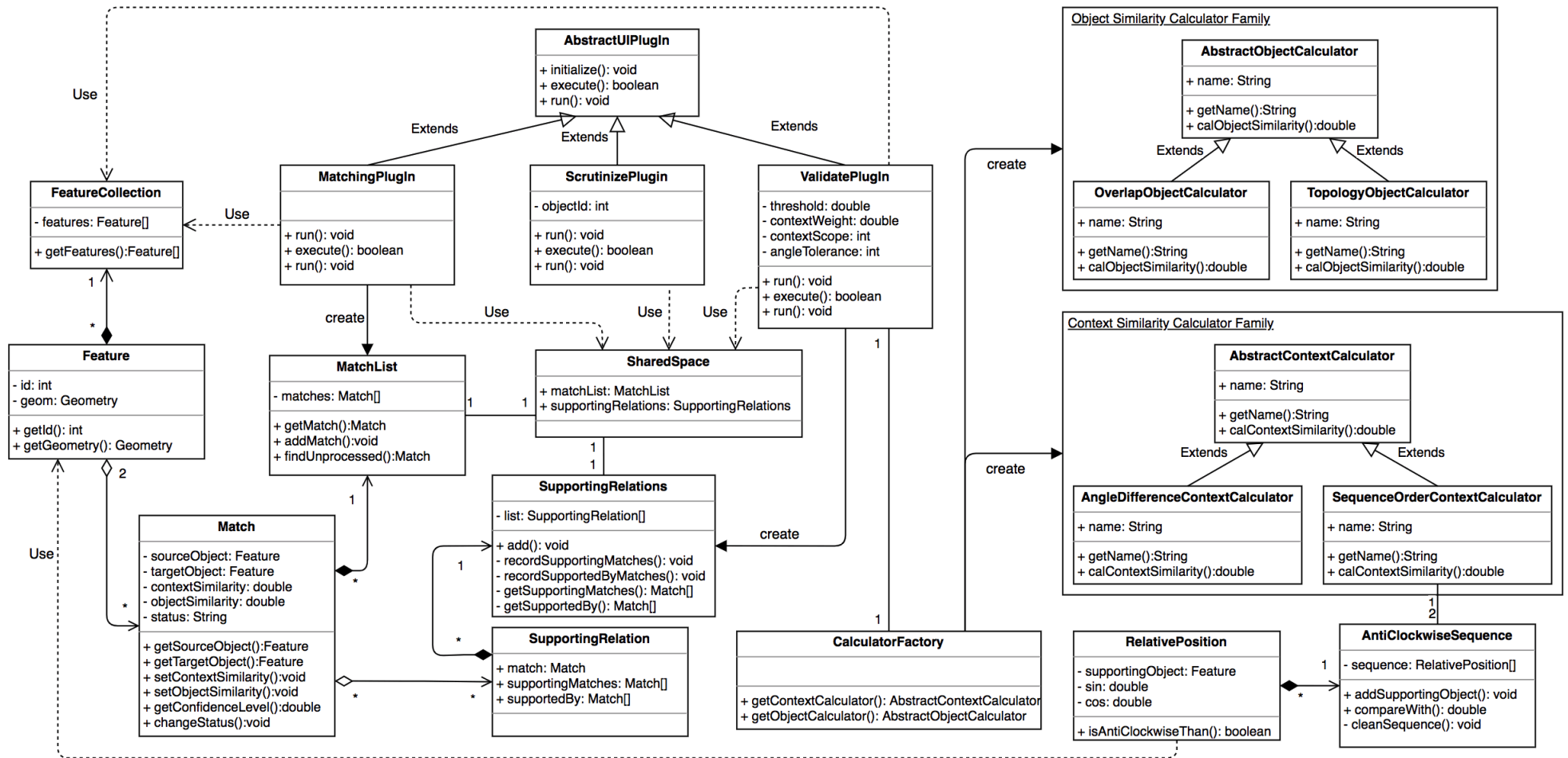


Figure 4.2: Class diagram of the implemented software

This matching plug-in provides several common tools for matching geographic objects, such as overlap matcher, intersects matcher, centroid distance matcher and topological matcher. When the matching plug-in is executed, in addition to the matching method, the users are also asked to select the source and target layers (i.e. visualized geographic object sets in GISs) for the matching. The later matching process is based on the assigned source layer. The plug-in will go through all the objects in source layer, checking all their potential matches and recording the matches it considers as correct. Finally, the plug-in will generate a new layer contains all matched source layer objects, and another layer for the unmatched ones. To visualize the matching relations more clearly, the matching plug-in can also generate a third layer, where each match is visualized as a line segment starting from a source layer object and ending at the corresponding target layer object.

In the matching plug-in, the matches are stored in a data structure called MatchMap. However, the MatchMap is incompatible with the later validation processes, since the relative information (e.g. context similarity, object similarity) for matches can not be recorded in it. In addition, modifying the MatchMap to adapt to requirement of validation process conflicts with both Open-Closed Principle and Single Responsibility Principle. To solve this problem, the class MatchList is created. Each match stored in this new data structure is attached with data fields of context similarity, object similarity and status. When a match is generated in the matching plug-in, a corresponding element in MatchList is created. The MatchList instance is stored in the shared space, so the validation plug-in can access to and manipulate these matches easily. Finally, it is worth noting that this modification is permitted by the open source licence of the matching plug-in.

4.2.2 Validation Plug-in

The whole validation algorithm is implemented in this plug-in. The main body of validation algorithm is implemented in ValidatePlugIn class. The context and object similarity calculation methods are developed in individual classes.

When the validation plug-in is executed, a dialogue window will be displayed to allow users configure the parameters like validation threshold, context scope and angle tolerance. There is no need to import the datasets manually, since the plug-in automatically fetch the required datasets and information from the shared space. After evaluating each input match, the plug-in records its context similarity, object similarity in corresponding fields attached to the match. The status field for all matches are initialized as UNDISCOVERED. After a match is discovered as a supporting match of other candidate matches, it will be pushed onto the tail of a queue, and its status field turns into INQUEUE. After the evaluation of a match, depending on the result the match will be marked as VALID or INVALID. Finally, in the process of detecting missing matches, all the detected matches will be added into the MatchList, with statuses of NEW.

The process of supporting match selection is set with the ValidatePlugIn, since the selected supporting matches will be considered as the candidate matches to be evaluated in the following turns. In addition to pushing these supporting matches onto the queue, the plug-in also records the dependence between each candidate match and its supporting matches in the data structure called SupportingRelations. In later backward rectification process, by looking up this record the plug-in can identify all the matches which are influenced by a discovered invalid match.

Context Similarity Calculators

The candidate matches and their supporting matches are passed into the context similarity calculators for context similarity computation. Though the two proposed context similarity measures have different inner logic, they share the same input and output parameter patterns. The validation framework only need to call the method of the selected context calculator, without the necessity of distinguishing which calculator is used. Thus, the two context similarity calculators can be organized using the Factory pattern.

As the top right corner of Figure 4.2 shows, the two context calculators both inherit an abstract super class, which specifies the common interface and methods. The factory class CalculatorFactory is in charge of the creation of them. Once the validation plug-in receives the user selected type of context similarity measure, it will pass the type to CalculatorFactory and get a calculator

of corresponding type. Since the interface of the two calculators are the same, the plug-in can control the received calculator using the same script. In this way, the code of context similarity computation in the main validation framework is separated from the code and logic of specific context calculators. Modifying a existing context calculator and adding new ones will not influence the main framework, which accords with the Open-Closed Principle.

In the angle-difference context similarity calculator, the angle difference is measured by the degree of angle between two connecting lines formed by the candidate and supporting objects in source and target layer. As Figure 4.3 shows, instead of calculating the intersection angle directly, the angle difference is calculated by subtracting the two angles between the connecting lines and positive x-axis (i.e. east direction). In each object set, according to the x- and y-differences between the candidate object and supporting object, the angle between their connecting link and positive x-axis can be calculated (As the code in Figure 4.4 shows). If the subtraction result of the angles is smaller than the predefined angle tolerance, this supporting match shows its acknowledgement to the validness of the candidate match. At the end, the context similarity is calculated as: number of acknowledgements / total number of supporting matches.

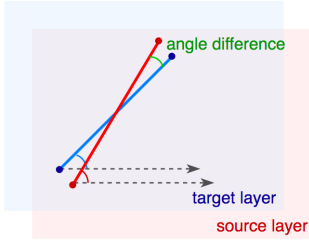


Figure 4.3: Angle difference

```
Point centerCentroid = centerFeature.getGeometry().getCentroid();
Double cX = centerCentroid.getX();
Double cY = centerCentroid.getY();
Point supportingCentroid = supportingFeature.getGeometry().getCentroid();
Double fX = supportingCentroid.getX();
Double fY = supportingCentroid.getY();
Double xDiff = fX - cX;
Double yDiff = fY - cY;
if (xDiff == 0 && yDiff == 0) {
    angle = -1; // points overlay
} else {
    angle = Math.toDegrees(Math.atan2(yDiff, xDiff)); // degree: [-180, 180]
}
```

Figure 4.4: Code for calculating degree using coordinates

In the sequence-order context similarity calculator, the angle between the connecting lines of object centroids and the positive x-axis will be calculated in the same way as Figure 4.4 shows. According to these angles, the supporting objects in each object set are sorted in ascending order separately. Since increasing the angle from 0 degree to 360 degree make the angle extend in the anti-clockwise direction, this kind of sort also arranges the objects in an anti-clockwise order. As Figure 4.5 illustrates, an increasing angle will push the object toward the anti-clockwise direction, the order of the angles of A, B and C is also their order of occurrence in anti-clockwise direction. After the sorting, some head elements in the resulted sequence will be moved to the tail, making the two sequences aligned with matched object at the head. Then, a nested for-loop will search the optimal way to remove the minimal number of objects, keeping the matched objects appear at the same location in the two sequences. At the end, the length of the remaining sequence is divided by the number of input supporting matches to get the context similarity.

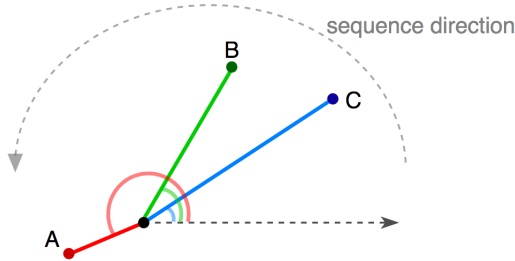


Figure 4.5: Sort supporting objects in anti-clockwise order

Object Similarity Calculators

Same as context similarity calculators, object similarity calculators can also be implemented using factory method. All the concrete object similarity calculators inherit the class of `AbstractObjectCalculator`. The validation plug-in can get a required object similarity calculator through the

CalculatorFactory.

Since the object similarity is an auxiliary tool for supporting the validation result, it is not the focus of this project. Some simple object similarity calculator were developed for applying some geometry-based similarity checking. For example, in the `OverlapObjectCalculator` (code is shown in Figure 4.6), object similarity is measured as the proportion of the twofold overlap area to the sum of the two objects' areas.

```
public class OverlapObjectCalculator extends AbstractObjectCalculator{
    @Override
    public double calObjectSimilarity(Feature f1, Feature f2) {
        Geometry f1Geom = f1.getGeometry();
        Geometry f2Geom = f2.getGeometry();
        Geometry overlap = f1Geom.intersection(f2Geom);
        double baseArea = f1Geom.getArea() + f2Geom.getArea();
        return (2 * overlap.getArea()) / baseArea;
    }
}
```

Figure 4.6: The code of object similarity calculator

4.2.3 Scrutinize Plug-in

The scrutinize plug-in is an auxiliary tool for visualizing the validation process. Sometimes the user may wonder why an input match is classified as valid/invalid. Under this situation, the user may execute the scrutinize plug-in and input the id of the suspected match. The scrutinize plug-in will print the context similarity and object similarity of this match, as well as the parameter configuration applied in the validation (As Figure 4.7 shows). In addition, the plug-in will create a new layer to highlight the supporting matches of the being checked match. As Figure 4.8 shows, above the source object layer (in blue), the scrutinize plug-in overlays a new layer with objects cut from the target layer. Here, the target layer object involved in candidate match is visualized in orange; the green objects are the target layer objects in supporting matches which acknowledge the candidate match; and the red object indicates the angle different of supporting match exceeds the angle tolerance. This kind of visualization can give the users intuitionistic feeling about the discrepancy between the contexts.

```
----- Scrutinizing the match id = 970 -----
Validation Result: Valid
Confidence Level: 0.8339 ( threshold: 0.8 )
Context Similarity: 0.8571 ( weight: 0.8 )
Object Similarity: 0.7409 ( weight: 0.2 )

Context similarity calculated using Angle Difference Measure
Angle tolerance: 5
Source layer objects: 171 326 639 806 1230 1531 1718
Source layer angles: 74.8667 175.5166 16.6909 194.3921 122.9871 93.2667 194.9110
Target layer angles: 86.0953 175.4524 18.2619 192.6359 125.0260 96.8613 193.1364

Within angle tolerance: 6 / 7 = 0.8571428571428571
The following surrounding match(es) exceed(s) the angle tolerance range:
171-3793: degree difference: 11.2286
```

Figure 4.7: Information printed by the scrutinize plug-in

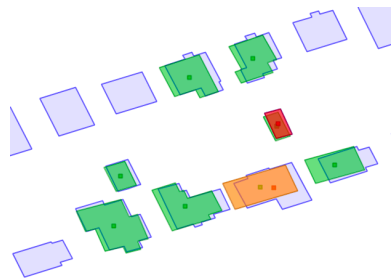


Figure 4.8: Highlighted supporting matches

Chapter 5

Experiments and Evaluation

This chapter introduces how the implemented program was tested. Several sets of experiments are applied to evaluate the influence of validation threshold, context similarity weight, context scope and angle tolerance.

5.1 Experiment Setting

The implemented program has been tested over several different datasets, within which two representative datasets are evaluated in this chapter. Each of the testing datasets composes of two geographic object sets from different data sources ¹, and a list of matches over them. Table 5.1 shows the size of object sets and match set in these two datasets. The last two columns of it display the ground truth of how many correct matches are included in the input, and how many matches are absent from the input match set.

Table 5.1: Two datasets used in experiments

Dataset	Object Set A	Object Set B	Input Matches	Correct	Missing
Arlington, US	2276	2017	1685	1679	266
Santa Clara, US	3263	3200	2349	2335	494

Table 5.2: Validation result using default parameter configuration: validation threshold is 0.8, context similarity weight is 0.8, context scope is 5, angle tolerance is 5 degrees

Dataset	Task	TP	FP	TN	FN
Arlington	VIM	1648	1	5	31
	DMM	151	1	65	110
Santa Clara	VIM	2228	1	13	107
	DMM	419	0	420	75

Table 5.3: The evaluation of three output sets

Dataset	Valid Match		Invalid Match		Missing Match	
	Precision	Recall	Precision	Recall	Precision	Recall
Arlington	99.94%	98.15%	13.89%	83.33%	99.36%	58.65%
Santa Clara	99.96%	95.42%	10.83%	92.86%	100.00%	84.82%

¹Data sources: <https://hub.arcgis.com> and <https://extract.bbbike.org>

The experiments in this chapter all use the angle-difference context similarity measure and overlap-proportion object similarity measure, and tests the influence of 4 parameters (i.e. validation threshold, context similarity weight, context scope, angle tolerance) to the algorithm performance. Table 5.2 shows the validation result on these two datasets under the default parameter configuration, where the terms VIM and DMM are the acronyms of the two sub-tasks: validating input matches and detecting missing matches.

In the task of validating input matches, Positive and Negative represent the validation results on the input matches (i.e. whether the validation algorithm consider an input match as correct); True and False represent the correctness of these judgements. For example, FP is the number of matches the program identified as correct, but are actually incorrect.

In the task of detecting missing matches, TP and FP separately represents the number of correctly and incorrectly detected matches. After the detection process, the objects that remain alone are grouped under Negative label, meaning that the validation algorithm fails to detect new matches on them. Thus, FN can be understood as the number of omitted matches that the detection process fails to identify.

According to Algorithm 1 and Algorithm 3, the whole validation algorithm outputs three datasets: (1) a valid match set of all the valid matches from input match set; (2) an invalid match set for the remaining matches from the input, which are considered as invalid; (3) a missing match set which contains all the newly detected matches. In other words, all the positive predictions of the task of validating input matches are recorded in the valid match set; all the negative predictions from this task are stored in invalid match set. As for the task of detecting missing matches, its positive results (i.e. the detected new matches) form the missing match set.

$$\begin{aligned}\text{Valid Match Set} &= TP_{VIM} + FP_{VIM} \\ \text{Invalid Match Set} &= TN_{VIM} + FN_{VIM} \\ \text{Missing Match Set} &= TP_{DMM} + FP_{DMM}\end{aligned}$$

In this chapter the algorithm performance will be evaluated from these three output sets. For example, Table 5.3 shows the example evaluation on the experiment result in Table 5.2. For each output set, its precision and recall will be used as the assessment criteria. The precision measures how many matches of each output set are correctly classified. The recall of each output set expresses among all the matches which should be classified into this set, how many of them are correctly classified into this set.

$$\begin{aligned}\text{ValidMatchSet} &\begin{cases} \text{Precision} &= TP_{VIM}/(TP_{VIM} + FP_{VIM}) \\ \text{Recall} &= TP_{VIM}/(TP_{VIM} + FN_{VIM}) \end{cases} \\ \text{InvalidMatchSet} &\begin{cases} \text{Precision} &= TN_{VIM}/(TN_{VIM} + FN_{VIM}) \\ \text{Recall} &= TN_{VIM}/(TN_{VIM} + FP_{VIM}) \end{cases} \\ \text{MissingMatchSet} &\begin{cases} \text{Precision} &= TP_{DMM}/(TP_{DMM} + FP_{DMM}) \\ \text{Recall} &= TP_{DMM}/(TP_{DMM} + FN_{DMM}) \end{cases}\end{aligned}$$

5.2 Context Scope

The context scope is measured as the number of matches involved in the context similarity calculation for each being validated match. Generally, a small context scope means only the adjacent or nearby matches are picked up to support the context similarity calculation; and a large context scope implies the involved supporting matches are scattered in a wider area. In this experiment, the context scope is tuned from 1 to 49, with step of 2. The collected data is visualized in Figure 5.1. The raw data of this experiments and the following experiments is attached in appendix.

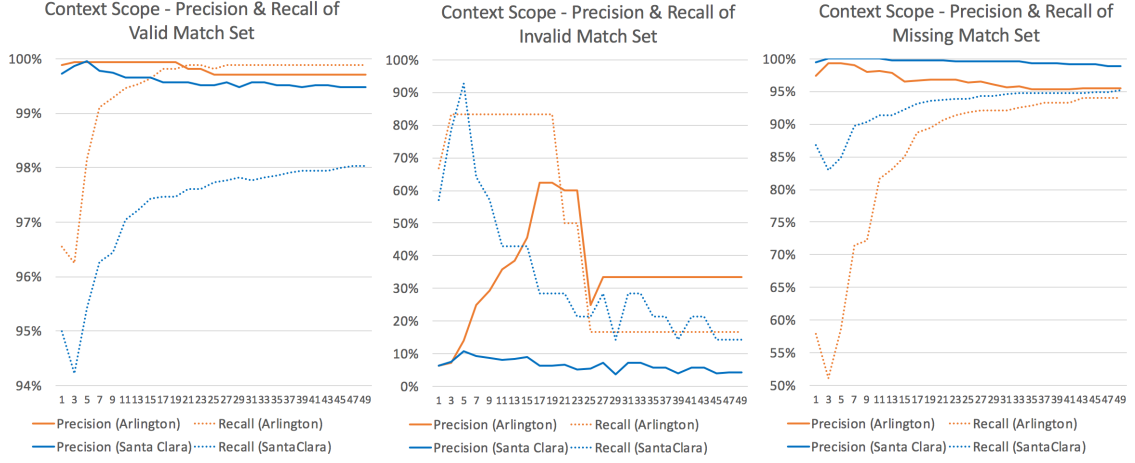


Figure 5.1: The variation of Precision and Recall of the three output sets according to the change of context scope

Generally speaking, for identifying valid matches and missing matches, extending the context scope benefits the recall, but damages the precision. When the context scope gets bigger, increasing number of remote matches are involved in the context similarity calculation. As Figure 5.2 illustrates, when viewing from a remote points, the offset between candidate objects can be reflected in a smaller angle difference than viewing from a nearby point. The difference between the objects in the candidate match will be blurred by their distance to the supporting matches. In other words, remote supporting matches are more tolerant to the discrepancy between candidate objects, and thus tends to over-estimate their similarity. As a result, the inflating context similarity helps more matches step across the validation threshold, being classified as valid matches or missing matches. Among these new matches, the actually valid ones help improve the precision of these two output sets, and the incorrect matches pull down their recalls.

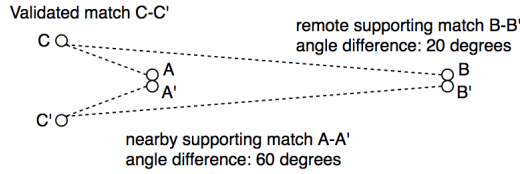


Figure 5.2: Smaller angle difference for remote match

Though the changes on the validation results of valid matches and missing matches accord to the same pattern, their scale of variation are different. In response to the same amount of context scope change, the precision and recall of missing match set vary more rapidly, which indicates the result of detecting missing matches is more sensitive to the setting of context scope. This difference might be explained by the nature of the datasets. The missing matches have not been identified by the matching algorithm since the matched objects have relatively large discrepancy on their shape or position. Thus, the similarities of objects in the input matches should generally be higher than those in missing matches. Under the same criteria, most of the missing matches should have lower context similarity than the input matches, which means the missing matches are harder to be acknowledged by the algorithm, resulting in lower recalls of detecting missing matches lower than identifying valid input matches. However, when context scope grows bigger, the influence of geometry discrepancy will be diluted. Except for some extreme cases of matches, most of the valid matches and missing matches can all be correctly identified. Thus, eventually the recall of these two sets both stabilise at similar upper limits.

At the same time, the recall of invalid match set changes in nearly the reverse way to those of the other two sets. As the extending context scope makes the validation criteria looser, the context similarity of invalid matches will be inflated and deceive the algorithm to classify them as valid,

damaging the recall of invalid match set. While the recalls follow a clear downtrend, the variation of precision is more irregular. As the context scope grows, both valid and invalid matches tend to be removed from the invalid match set. The change of precision is determined by the ratio between correct and incorrect matches in the removed matches, which depends on the concrete features of the objects. If in a scope extension more correct matches are removed from the invalid match set, the precision of invalid match set may increase; on the contrary, if more incorrect matches are removed by mistake, the precision will drop.

In the charts of valid match set and missing match set, there is also an anomaly worth noting. When context scope is 1, all lines in these two charts have a significant grow at the direction reverse to their main trends. The reason is that, in a 2D plane one point can be accurately described by its relative direction to at least two other points. As Figure 5.3 shows, when two or more surrounding matches are used, their composite angle restriction can limit the position of the candidate object in a small area. In this algorithm, this means the candidate objects have to appear in narrow area in their own contexts to obtain a higher assessment of context similarity, which requires a more strict context correspondence. But if the context similarity is calculated using only one surrounding match (as Figure 5.4 illustrates), all the points within a sector specified by the angle tolerance satisfy the condition well. Under this situation, larger geometry difference is allowed under the same angle tolerance restriction, and matches can obtain higher assessments of context similarity more easily. As a result, more matches are classified as correct, and more missing matches are discovered. Among these matches, the matches which are correct in ground truth contribute to the improvement of recall; and the actually incorrect matches lead to the drop of precision.

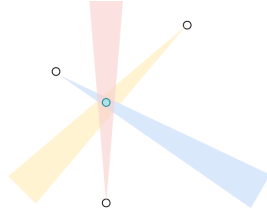


Figure 5.3: Spotting using 3 supporting matches

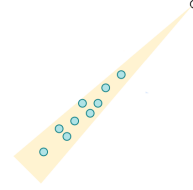


Figure 5.4: Loose restriction using 1 supporting match

Finally, the suggested range of context scope is approximately 15 to 20. The precision and recall of valid match set and missing match set all keep stable at relatively high values since the context scope gets larger than 15. The performance of invalid match set varies severely and should be paid more attention to. For the dataset of Santa Clara, the peaks of precision and recall of the invalid match set overlap from 15 to 25. For the Arlington dataset, both precision and recalls fluctuate in a downtrend, but they are generally acceptable before 20. Overall, context scope between 15 to 20 may help the validation algorithm achieve good performance in all these three aspects.

5.3 Angle Tolerance

In addition to context scope, angle tolerance also plays an important role in context similarity calculation. A supporting match will vote for the candidate match if the angle difference between is within the angle tolerance. Then the context similarity is calculated as the proportion of votes in all supporting matches. In this experiment, angle tolerance is tuned from 1 to 29, which should be large enough in real use cases. Figure 5.5 visualizes the change of precision and recall in the three output sets according to angle tolerance.

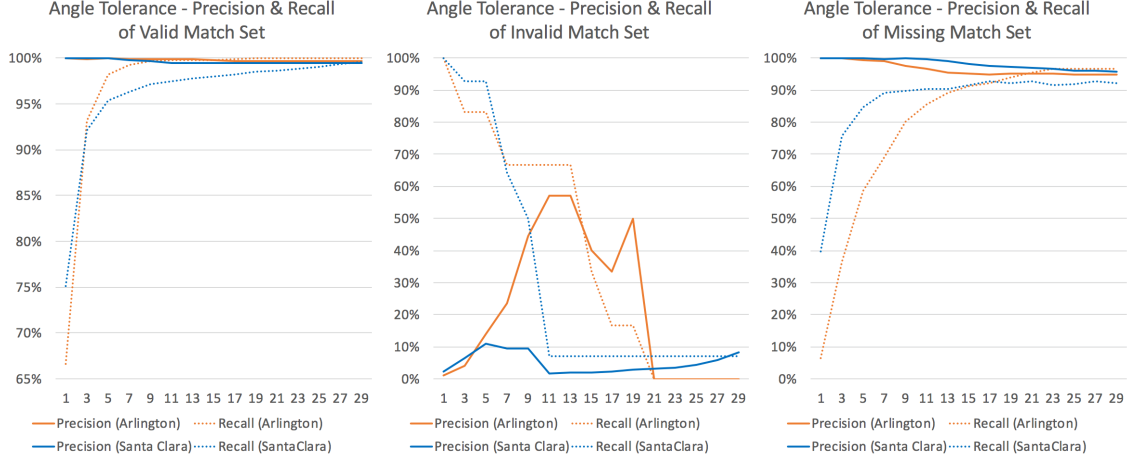


Figure 5.5: The variation of Precision and Recall of the three output sets according to the change of angle tolerance

As angle tolerance grows, after a short stable stage at 100%, the precision of valid match set and missing match set gets into gentle downtrends. The recalls of these two output sets both get into plateau after a rapid soar. The precision of invalid match set has peak in the middle, whose range and height vary between the two datasets. The recalls of the invalid set plummet from 100% to less than 10%, and stabilise at 7% and 0% separately.

Correspondence could be discovered between this experiment result and the previous one. The variation patterns of recall and precision in all the three output sets are similar with the corresponding ones in previous experiment, which indicates the algorithm performance changes in the similar way in response context scope change and angle tolerance change. The reason behind this similarity is the homogeneous influence of context scope and angle tolerance to context similarity calculation. With higher angle tolerance, the supporting matches will acknowledge the candidate matches even if they have larger deviation in centroid locations. Similar as context scope, the increasing angle tolerance applies more tolerant validation criteria on matches, resulting in similar variation patterns.

However, though the patterns are similar, in the valid match set and missing match set, the scale of variation is larger when reacting to the change of angle tolerance than to context scope. Regardless of the change on context scope, the recalls of these two sets always stay above 94% and 50%. But when angle tolerance is altered, the recall of valid match set and missing match set can drops under 70% and 10%, owning a large range of variation. This larger variation scale might be explained by the hierarchy of these two parameters. The angle tolerance effects at a deeper level in the context similarity calculation. The angle tolerance determines the lower boundary at which the supporting matches can acknowledge a candidate match, which influences each supporting match's judgement directly. The context scope can only decide the number of supporting matches used in context similarity calculation, make subtle adjustment on the result by involves more or less supporting matches. Thus, no matter how context scope changes, the floor level of its influence has already been determined by the angle tolerance.

One limitation of the validation algorithm is also exposed in this experiment. Theoretically, if two objects are matched according to ground truth, even if they have significant shape difference or errors from their data sources, the resulted deviation on centroids should not be exorbitantly large. However, even the angle tolerance of 29 degrees fails to help the validation algorithm identify all the missing matches. As the angle tolerance increases, the recalls of missing match set rise but then become stable at their upper limits which are both less than 100%. A manual scrutinize after the validation process shows that these undetected missing matches belong to part-of matching relations. As Figure 5.6 illustrates, a real world building might be represented as a single object in one database, but in another database be split and recorded as two separate objects. In this case, two part-of matches exist between 12390 and 7195, and between 12696 and 7195. However, both the centroid deviation and overlap proportion between a component object and the whole

object exceeds the normal range for verifying one-to-one matches. Thus, in the two validation criteria, this kind of matches both performs poorly. As a result, the validation algorithm is unable to identify this kind of matching relations, and always incorrectly classifies the part-of matches as invalid.

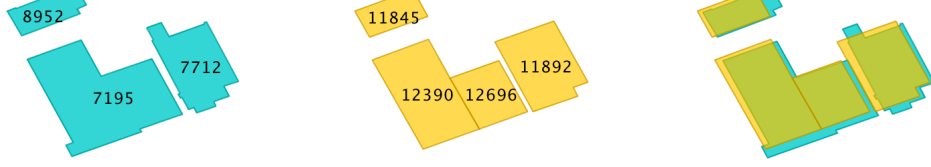


Figure 5.6: Example of the part-of matching relation

Further, the inefficiency in identifying part-of matches is a primary cause of the low precision of the invalid match set. Among all incorrect classification in the invalid match set, most of them are part-of matches. In the experiment on Santa Clara dataset with angle tolerance of 5 degrees, among the total 107 false classifications in invalid match set, 80 of them are part-of matches. If the dataset contains large proportion of part-of matches, these part-of matches will concentrate in the invalid match set and overwhelm the small number of actual invalid matches, pulling down the precision significantly. For example, in the dataset of Santa Clara with angle tolerance of 5, only one invalid match is omitted from the invalid match set, but the large number of part-of matches determines its precision can never reach to 10%. However, though the precision is not high in this dataset, it is also reasonable to believe that the validation algorithm can achieve better performance on datasets with less part-of matches.

Considering the above analysis, the selection of the optimal angle tolerance can be made. The algorithm performance on valid match set and missing match set are both stable and satisfying when angle tolerance increases. However, its performance on invalid match set changes rapidly. In invalid match set, the precision of the two datasets have different peaks in the middle, and happens to overlap with the plateaus of their recalls. Thus, the optimal angle tolerance depends on the specific dataset on which the algorithm is applied. Here for Arlington dataset, the optimum is between 10 to 15 degrees; for Santa Clara, the optimum is about 5 degrees.

5.4 Context Similarity Weight

After the configuration for context similarity achieves the optimum, an additional step of setting the weight between context and object similarity is required. If the context similarity weight gets too big, the individual attributes will be overlooked and some discrepancy between the matched objects might be omitted. On the contrary, if the context weight is set too small, the contextual information and the information of input matches cannot be fully exploited. Thus, a balance between these two measures should be kept to support a comprehensive and fair inspection without bias on any specific feature. As Figure 5.7 shows, in this experiment the context similarity is tested from 0 to 1.

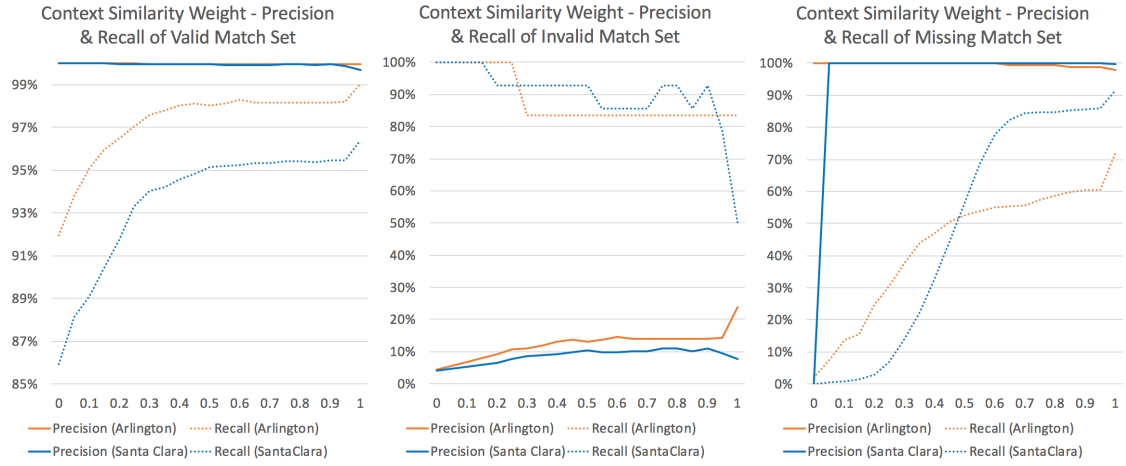


Figure 5.7: The variation of Precision and Recall of the three output sets according to the change of context similarity weight

As the first and third charts in Figure 5.7 illustrate, for valid match set and missing match set, their recalls grow up to the upper limits as the context similarity weight increases, but the precision is almost unaffected. As for the invalid match set, its precision rises gently with context similarity weight; and the recalls are maintained at 100% at small context weights, and is kept at relatively high level in most of the remaining time. An anomaly is the zero value of precision in Santa Clara dataset at weight = 0. This is a concomitant phenomenon of the zero value of the corresponding recall, since no missing match is correctly identified under this configuration.

In the task of validating input matches, when the context weight approaching 0, 100% precision appears in both valid match set and invalid match set; but at the same time their recalls are relatively low. Under this case the object similarity calculated using objects' overlap proportion is considered as the only criterion for verification. Thus, the low recall means that many matches in ground truth have relatively significant geometry discrepancies on their shapes and positions. This insight seems counterintuitive, since the correspondence of geometry features is considered as a prerequisite of geographic object matching. A deeper scrutinize indicates that high proportion of these omitted matches have acceptable similarity in shapes of their objects, and they are mis-classified due to the offset between their positions. For example, in the two object sets of Santa Clara dataset, the offset between their coordinate systems leads to subtle offsets between all matched objects. As Figure 5.8 illustrates, under this uniform offset, the object similarity on the match of object 5128 is 87%, but the object similarity for object 4730 is 74%. While the big objects are robust toward subtle offsets, the same offsets may separate half of the small objects, leading to low overlap proportion. For this type of matches, object similarity become useless; but context similarity can still effectively identify the correspondence, since the uniform offset on the whole dataset not influence the inner spatial relations between objects, which means the correspondence between the matched objects' contexts can still prove their matching relation.

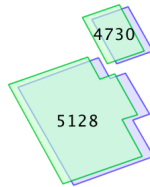


Figure 5.8: The same offset causes different influence on overlap proportion

Comparing the first and third charts in Figure 5.7, it can be found that the recalls of missing match sets change more rapidly than those of valid match sets. Since most of the input matches have both high context similarity and object similarity, they can get high confidence level regardless how the weight changes. However, the objects in missing matches may have significant discrepancies on either shape or position (which essentially is the reason why they are not identified by

the matching algorithm). When the weight of object similarity dominates, these differences will be amplified and pull down the whole confidence level, resulting in low starting points of recalls. As the weight of context similarity rises, the shape differences, most of which are caused by the different standards and levels of details applied in object sets, will be blurred and tolerated. Also, the influence of coordinate system offsets is mitigated in this process. As a result, most of the missing matches can be discovered eventually. The low starting point and the high end position make the variation range of recalls in missing match larger than that in the valid match set.

In summary, a general guidance is setting the context similarity weight over half. Seeing from Figure 5.7, when context weight exceeds 0.5, recalls stabilise at relatively high value, and the precision is also acceptable.

5.5 Validation Threshold

Validation threshold is the last parameter in the validation algorithm. The validness of a match will be acknowledged only if its confidence level is larger than or equal to the threshold. In this experiment, the validation threshold is tuned from 0 to 0.5, and the result is visualized in Figure 5.9.

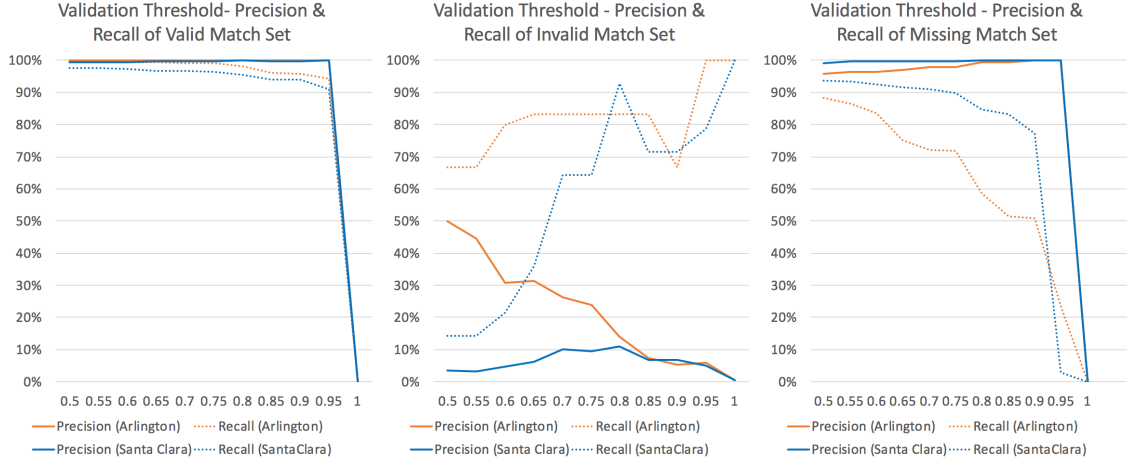


Figure 5.9: The variation of Precision and Recall of the three output sets according to the change of validation threshold

Similar patterns appear in valid match set and missing match set: as the threshold rises, precision gradually rises and the recalls gently drop, but they both plummet to 0 when threshold is 1. For invalid match set, its precision goes down when threshold approaches to 1; the recalls are in up-trend, with short stable phases in the middle.

When identifying valid matches and missing matches, a higher threshold implements a stricter criteria and only picks the matches whose objects are highly corresponding. Empirically, the matches with the top confidence levels are scarcely possible to be incorrect, so the precision of this cherry-picking is almost 100%. However, since some matches are obsoleted in this process, the recalls would inevitably drop. This law is valid until the threshold become 1, where slumps occur on all the curves.

When the validation threshold is set as 1, a match can be classified as valid if and only if its context similarity and object similarity both reach to 100%. This is nearly impossible. Due to the errors between different data sources, the matched objects from the two object sets can hardly have exactly the same centroid position and outer contour. This limitation prevents the object similarity and also the confidence level from reaching to 100%. As a result, the threshold of 1 will force the algorithm to classify all input matches as invalid.

With regard to the invalid match set, the precision reduces with the rising of threshold, since an increasing number of valid matches drop behind the threshold and be classified into the invalid match set by mistake. The extending base in the calculation makes the precision drop.

Considering the above analysis, an appropriate range of the validation threshold might be from 0.6 to 0.8. A lower threshold may lead to the low recall in invalid match set; a higher threshold may encounter the plummet of performance in valid match set and missing match set.

5.6 Conclusion

In summary, the validation algorithms could achieve the desired performance. The precision and recall of identifying valid input matches stay above 99.5% and 95% separately. Seeing from the aspect of identifying invalid input matches, 5 of the total 6 invalid matches and 13 of the total 14 invalid matches in the two datasets are successfully identified, demonstrating its practicability in real use cases. As for the task of detecting missing matches, the appropriate configuration of parameters could help the algorithm identify around 95% of all the missing matches, with the precision over 95%. At the same time, the performance of the algorithm is stable. Though the starting point of validation is selected randomly, the backtracking mechanism keeps trimming the inappropriate neighbour selection in context similarity calculation, making the final validation result converge.

However, the validation algorithm also has limitations. First, as previously mentioned, the algorithm fails to understand the part-of matching relations. This kind of matches become the primary reason of its false predictions. Second, the efficiency of the algorithm drops significantly when exorbitantly strict validation criteria are applied, which makes it less cost-efficient in the tasks pursuing extreme precision. In the algorithm design, for each invalid match, the backtrack rectification mechanism will be executed to apply extra context similarity calculation for all supporting matches in its scope. When strict validation is implemented (e.g. by improving the validation threshold or setting a small tolerance), most of the input matches are recognized as invalid and this mechanism will be activated frequently, amplifying the workload of computation. For example, when context scope is 10 and a strict validation rules out 80% of valid matches since they are less credible, the workload of context similarity computation will be amplified to at least $100\% + 80\% * 10 = 9$ times. However, though this drawback limits the algorithm’s ability to cherry-pick the most creditable matches, it will not slow down the efficiency in general use cases, where the backtrack processes can be controlled within 3 rounds when the parameter configuration approaching optimum.

The previous experiments show that, comparing to invalid match set and missing match set, the performance on valid match set is more stable in response to the variation of parameter configuration. As for valid match set and missing match set, context scope, angle tolerance and context similarity weight have similar effect on algorithm performance. The increments of these parameters generally lead to looser validation criteria, which rise the recall and slightly sacrifice the precision. On the contrary, the rising validation threshold will improve their precision and decrease their recalls. As for invalid match set, the changes on all parameters lead to reverse effects. In summary, selecting the parameters from the middle parts of the tested ranges can make a tradeoff between precision and recalls of the output sets.

Chapter 6

Project Management

This chapter reviews the whole process of this project. Based on the comparison between the actual progress and the original plan, some experience and reflection are presented.

6.1 Project Management

The work plan and actual progress of this project are visualized in the Gantt charts displayed in Figure 6.1. Generally speaking, though some tasks had some delay in the middle of the project, eventually all the tasks are finished, and all the objectives are achieved on time.

In the first stage of the project, which is the first semester, the main work is background research and validation algorithm design. The milestones of this phase are the pseudo code of the validation algorithm and a framework of software, both of which were achieved on time. Seeing from the Gantt charts, an extra week was spent on background research, since some relative research papers derived from the collected papers were also explored. The extra reading took me more time to digest, but also provided useful guidance for my algorithm design. Another difference between the plan and progress is that the development of software framework was started one week in advance. At that time, some GIS software had been installed to visualize the collected geographic data. A glance into these software frameworks made me realize that their complexity exceeded my expectation, and more time should be spent on getting familiar with these open source frameworks. Eventually, the extra week made this task be finished on time, which indicated the necessity of the temporary change on plan.

The second stage of this projects is between the end of the exam weeks and the start of the Chinese Spring Festive. The main task in this stage is developing the main body of the validation algorithm, as well as the context similarity calculator, since these two parts tangled with each other. However, a temporary decision of participating in a winter camp activity disrupted the plan. The unexpected activity occupied too much time and the project development almost stagnated. As a result, the delayed tasks significantly increased the workload in the next stage.

The third stage starts at the middle of February and ends at the deadline of the project. Due to the previous task delay, the working hours assigned to this project was increased twofold. Generally speaking, the adjusted work plan efficiently controlled and mitigated the negative influence of the task delay. With the cost of extra workload, the progress of the whole project eventually caught up the plan. In the later experiment phase, some extra time was spent on collecting geographic data, since the original data was too few to support a comprehensive evaluation on the implemented algorithm. The loss of time was worthy, as a new website which provides high-quality geographic datasets was discovered in the search. Finally, the analysis of experiment result was more time-consuming than expectation. Due to the inexperience in this task, before making the optimal choice, unexpected long time was spent on trying different evaluation criteria and charts. Also, during this period, more working hours were taken to keep the progress stay on schedule.



Figure 6.1: Comparision between plan and actual progress

In addition to the project schedule, some extra measures were taken to support the time management. Firstly, the weekly meeting with supervisor is kept through the whole year, which can be proved by the brief list of meeting records attached in the appendix. The requirement to regularly report work progress would spur me to keep at least a minimal workload on this project even in the coursework season. In addition to the supervision on progress, the communication with supervisor provides valuable feedback about the works, which may timely prevent the project from going on a wrong direction. Secondly, fixed working hours are scheduled in every week. Placing the project onto the timetable can frequently remind me to regularly carry on the project when I was busy for other courses. In addition, integrating the work in fixed long time slot, instead of scatter it randomly through the whole week, can keep the coherence of thinking and improve the efficiency.

6.2 Reflection

The first and the most important experience from this project is that, the practice is not as ideal as imagination. Emergencies and unexpected difficulties may occur frequently, postponing the progress behind schedule. For example, at beginning the data collection was considered as a easy task, but the practice made me realized that the geospatial resources are scattered in different data sources, and it difficult to find the desired pairs of dataset on the same geographic region. In addition to the external factors, some internal factors may also influence the execution of the plan, within which my personal preference is the most significant one. I prefer to start with the easy and straightforward tasks, and leave the tricky ones to the last minute. In the first stage of the project, after finding the minimal sample data collection for testing the Java framework, I shelved the further search for better data resources and turned to other tasks which were ahead of schedule.

In general, the risks of task delay is controllable by setting a more flexible schedule and adjusting the workload depending on concrete situation. Firstly, the work plan should be changed dynamically to address the uncertainty in practice. For example, in the previous work, I found OpenJump can be directly used as the software framework, so I gave up the plans about developing framework by myself and assigned more time on getting familiar with OpenJump. In addition, I underestimated the difficulty of collecting sample data and have spent extra time in this task. These accidents all require me to frequently compare the progress with the plan, and adjust the plan if necessary. Secondly, the plan should be concrete about the current stage and abstract about the future. This long-term project should be split and carried out in short sprints of one or two weeks. The long-term plan only need to describe the general stages and structure of the project; more detailed tasks should be organized at the beginning of each sprint. For example, only a ambiguous task of ‘developing context similarity calculator’ is listed on the work plan; but in the practice, more minute sub-tasks were listed and scheduled before the development. Thirdly, some buffer time should be reserved and interspersed among the tasks on the schedule. If the completion of a task is delayed, the next buffer slot can be used to catch up the progress. If everything follow the schedule, these buffers can be used to bring forward the following tasks.

Another reflection is about software quality control. Before the software development, the pseudo code of the validation algorithm is written to specify the logic behind each step of operation; and a class diagram is developed to ensure the robustness and maintainability of software hierarchy. However, during the software development, these two documents were rarely referred to. Most scripts of the software is developed by impromptu coding, which accords with the general direction of the project but may conflict with original software design in subtle places. For example, some methods and data structures might be created by temporary decision. These unplanned code blocks may introduce new bugs or incompatibility to other parts of the software. As a chain effect, more arbitrary modifications and patches occurs. As a result, on one hand, the readability of source code is damaged, making the future maintenance more difficult; one the other hand, this impromptu development may make the software deviate from the original design. In both of these cases, extra work is required to clean the source code, slowing down the progress of the project.

To avoid messing up the source code and control the software quality, several measures have been taken: (1) Print out the relative documents and stick them on the wall of the workplace. The frequent glance on them can remind me of the original design of the software. (2) Develop the

whole skeleton of software before implementing any concrete logic. For example, all the classes and the interfaces of their methods can be developed first. The function calls and parameter list of methods can be fixed in this step. Then the following development is just filling the logic into these empty methods. Since the skeleton has already been established, the remaining development would be restricted in the framework of the original design, with less possibility to cause divergence. (3) Split the development of the whole software into small units. After the development of each unit (e.g. an object similarity calculator or a context similarity calculator), compare the implementation with pseudo code, and run some unit tests to validate its correctness. Before turning into the development of the next unit, review its pseudo code and structure. This strategy is especially useful in this kind of long-term software development, where the memory of software design may fade away after intensive work or after working on other parts for a long time. By developing in units, the robustness of the whole software can be established at the beginning; in addition, the mapping relation between each unit and corresponding pseudo code can clarify the purpose and logic of each part of the software, facilitating the inspection and maintenance.

Chapter 7

Summary

This chapter summarizes the main outcomes of the project. The contribution of the proposed algorithm is discussed in a wider context relating to other existing methods and tools. The future works are also listed to introduce the potential improvements of the validation algorithm and the implemented OpenJump plug-in.

7.1 Contribution

The validation algorithm has shown its high performance in validating input matches and detecting missing matches, which demonstrates its practicability in real world use cases. As the introduction chapter declares, the first and most primary usage of this algorithm is supporting the inspection of matching results of geographic datasets. In most workplaces, manual double-check is required to inspect the matches generated by matching methods. Now this process can be replaced by, or at least be accelerated by, software of this validation algorithm. Though the current validation algorithm fails to achieve 100% of precision and recall at the same time, users can focus on the high recall of identifying invalid matches to filter all potential invalid matches. Under this configuration, though the sacrifice on precision means some correct matches may be mixed into the output invalid match set, most of the truly incorrect matches can be identified correctly. The concentration of incorrect match means staffs can search the errors of matching result from a small match set instead of all the generated matches. Considering that the normal size of the output invalid set is less than 5% matches of the whole input match set, the search space can be significantly reduced, and the workload can be lessened.

Another contribution is that, the validation algorithm can be applied cooperatively with existing matching methods to improve their performance. Generally, the matching methods have to balance the tradeoff between precision and recall. While a strict matching criterion helps the matching method match corresponding objects accurately, some less recognizable matches will be inevitably omitted. This shortage can be efficiently covered by this validation algorithm. Given two partly matched geographic object sets, the validation algorithm shows high performance in detecting the remaining missing matches based on the existing information. Thus, it is possible to apply strict criteria in matching methods to generate relatively less but accurate matches; then use the validation algorithm's ability of detecting missing matches to find the remaining matches. In other words, let the matching method pursue high precision, and use the implemented validation algorithm to supplement the recall of discovered matches. Finally, the combination of these two tools can generate the matching result with both high precision and recall.

The validation algorithm may also be used as an evaluation tool to compare the performance between different matching methods. The various types of measures in different matching methods implies their performance may also varies depending on the feature of the datasets they are applied on. Thus, how to choose the most appropriate matching method for a specific use case sometimes remain as a problem. The validation algorithm can simplify the decision making process. The validation result (e.g. the rate of valid matches and the number of detected missing matches) matching results can clearly indicates whether each of the matching method can efficiently deal with the task in a specific dataset. Furthermore, the visualized incorrect matches and missing

matches may expose the weaknesses of these matching methods, indicating why some of them performs less efficient in this use case. In summary, by comparing the validation results, the optimal matching method in a specific use case can be identified.

In addition to optimizing the choice of matching methods, the validation algorithm also contributes to the parameter optimization for each matching method. After choosing a matching method, the next step is tuning its parameters to improve its performance, which can also be facilitated by the validation algorithm. The matching results under different parameter configurations can be fed into the validation algorithm. Instead of comparing the performance manually, the validation algorithm can automatically assess the precision and recall of each set of experiment, displaying the accuracy and completeness of the matching result. The user can choose the set of parameters under which the largest proportion of match are correctly identified.

7.2 Future Works

Limited by time and professional knowledge in task domain, the proposed validation algorithm is practicable but not perfect. Many future works are required to further evaluate and improve its performance.

Firstly, the evaluation on the validation algorithm may still not thorough enough. Only one type of context similarity calculation method and one object similarity calculation method is tested in experiments. In the future work, more experiments should be applied on other alternative measures and other combination of context and object similarity measures. At the same time, applying more experiments over a wider range of datasets is necessary, since some potential problem might only be exposed in specific use cases.

Secondly, the limitation on identifying part-of matches needs to be optimized. Due to the different standard of recording geographic objects, this kind of matching relation widely appears between different databases. If the validation algorithm is unable to deal with the part-of matches, its practicability will be significantly restricted. A possible way to identify these matches is introducing an extra inspection step. For example, after the validation process, each cluster of adjacent alone objects can be combined as an entity, being checked again in an extra round.

Thirdly, the modules in the validation algorithm can be improved independently. The modular construction of the validation algorithm ensures the flexibility of its components. Provided the input and output interface are kept the same, the inner logic of each module can be changed freely. For example, in the proposed validation algorithm, the context of a match is defined as the K-nearest neighbours; however, the context can also be defined in the form of donuts, where the closest matches are ignored and the relatively nearby matches as used as the supporting matches. This modification may improve the fairness of the context similarity evaluation, since in previous experiments, the close supporting matches appear to be over-sensitive to the difference in objects' shapes. Moreover, the measure of context similarity computation may also be improved by comparing the correspondence of distances between the candidate objects and their contextual objects. Though the current measures, which only utilize the direction relations, have shown acceptable precision and recall, it is still possible to improve the performance further by introducing the an extra dimension of inspection.

Finally, it is possible to adjust the validation strategy based on extra information provided by the user. Generally, after a overview on the datasets or matching result, the users may have some impression and advanced knowledge about the two being matched datasets. For example, some experts may deduce that the being matched datasets have obvious offset between coordinate systems, or they may contain large proportion of part-of matches. These kinds of pre-knowledge may help the validation algorithm establish a correct cognitive system about what is normal and what is anomalous in the being processed datasets. As a result, the improved understanding may help the algorithm eschew mistakes caused by special and unexpected features in different datasets.

References

- [1] KH Anders and J Bobrich. “MRDB approach for automatic incremental update”. In: *ICA workshop on generalisation and multiple representation, Leicester*. Vol. 2004. 2004.
- [2] Matthias Butenuth et al. “Integration of heterogeneous geospatial data in a federated database”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 62.5 (Oct. 2007), pp. 328–346. DOI: 10.1016/j.isprsjprs.2007.04.003.
- [3] Daniel Chen et al. “Approximate Map Matching with respect to the Fréchet Distance”. In: *2011 Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, Jan. 2011, pp. 75–83. DOI: 10.1137/1.9781611972917.8.
- [4] Fred J. Damerau. “A technique for computer detection and correction of spelling errors”. In: *Communications of the ACM* 7.3 (Mar. 1964), pp. 171–176. DOI: 10.1145/363958.363994.
- [5] Thomas Devogele. “A New Merging Process for Data Integration Based on the Discrete Fréchet Distance”. In: *Advances in Spatial Data Handling*. Springer Berlin Heidelberg, 2002, pp. 167–181. DOI: 10.1007/978-3-642-56094-1_13.
- [6] Elena Deza. *Dictionary of distances*. Amsterdam, the Netherlands Boston: Elsevier, 2006. ISBN: 9780080465548.
- [7] Heshan Du and Natasha Alechina. “Qualitative Spatial Logics for Buffered Geometries”. In: *Journal of Artificial Intelligence Research* 56 (Aug. 2016), pp. 693–745. DOI: 10.1613/jair.5140.
- [8] Heshan Du et al. “A Method for Matching Crowd-sourced and Authoritative Geospatial Data”. In: *Transactions in GIS* 21.2 (May 2016), pp. 406–427. DOI: 10.1111/tgis.12210.
- [9] Anna Formica and Elahé Pourabbas. “Content based similarity of geographic classes organized as partition hierarchies”. In: *Knowledge and Information Systems* 20.2 (Nov. 2008), pp. 221–241. DOI: 10.1007/s10115-008-0177-8.
- [10] Yong Huh, Kiyun Yu, and Joon Heo. “Detecting conjugate-point pairs for map alignment between two polygon datasets”. In: *Computers, Environment and Urban Systems* 35.3 (May 2011), pp. 250–262. DOI: 10.1016/j.compenvurbsys.2010.08.001.
- [11] Mike Jackson, Rahemtulla A, and Jeremy Morley. “The synergistic use of authenticated and crowd-sourced data for emergency response”. In: Jan. 2010.
- [12] Jung Ok Kim et al. “A new method for matching objects in two different geospatial datasets based on the geographic context”. In: *Computers & Geosciences* 36.9 (Sept. 2010), pp. 1115–1122. DOI: 10.1016/j.cageo.2010.04.003.
- [13] Vladimir I Levenshtein. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966, pp. 707–710.
- [14] Ariane Mascaret et al. “Coastline Matching Process Based on the Discrete Fréchet Distance”. In: *Progress in Spatial Data Handling*. Springer Berlin Heidelberg, 2006, pp. 383–400. DOI: 10.1007/3-540-35589-8_25.
- [15] Grant McKenzie, Krzysztof Janowicz, and Benjamin Adams. “A weighted multi-attribute method for matching user-generated Points of Interest”. In: *Cartography and Geographic Information Science* 41.2 (Jan. 2014), pp. 125–137. DOI: 10.1080/15230406.2014.880327.
- [16] D. Min, L. Zhilin, and C. Xiaoyong. “Extended Hausdorff distance for spatial objects in GIS”. In: *International Journal of Geographical Information Science* 21.4 (Apr. 2007), pp. 459–475. DOI: 10.1080/13658810601073315.

- [17] Oxera. *What Is the Economic Impact of Geoservices?* Jan. 2013. URL: http://www.oxera.com/Oxera/media/Oxera/downloads/reports/What-is-the-economic-impact-of-Geo-services_1.pdf.
- [18] Ana-Maria Olteanu Raimond and Sébastien Mustière. “Data Matching – a Matter of Belief”. In: *Headway in Spatial Data Handling*. Springer Berlin Heidelberg, 2008, pp. 501–519. DOI: 10.1007/978-3-540-68566-1_29.
- [19] Sanjay Ranade and Azriel Rosenfeld. “Point pattern matching by relaxation”. In: *Pattern Recognition* 12.4 (Jan. 1980), pp. 269–275. DOI: 10.1016/0031-3203(80)90067-9.
- [20] J J Ruiz-Lendinez, F J Ariza-López, and M A Ureña-Cámara. “Automatic positional accuracy assessment of geospatial databases using line-based methods”. In: *Survey Review* 45.332 (Sept. 2013), pp. 332–342. DOI: 10.1179/1752270613y.0000000044.
- [21] Juan Ruiz-Lendinez, Manuel Ureña-Cámara, and Francisco Ariza-López. “A Polygon and Point-Based Approach to Matching Geospatial Features”. In: *ISPRS International Journal of Geo-Information* 6.12 (Dec. 2017), p. 399. DOI: 10.3390/ijgi6120399.
- [22] Ashok Samal, Sharad Seth, and Kevin Cueto1. “A feature-based approach to conflation of geospatial sources”. In: *International Journal of Geographical Information Science* 18.5 (July 2004), pp. 459–489. DOI: 10.1080/13658810410001658076.
- [23] Shuangli Shan et al. “Geographical address representation learning for address matching”. In: *World Wide Web* 23.3 (Feb. 2020), pp. 2005–2022. DOI: 10.1007/s11280-020-00782-2.
- [24] Wenbo Song et al. “Relaxation-Based Point Feature Matching for Vector Map Conflation”. In: *Transactions in GIS* 15.1 (Feb. 2011), pp. 43–60. DOI: 10.1111/j.1467-9671.2010.01243.x.
- [25] Volker Walter. *Zuordnung von raumbezogenen Daten: am Beispiel der Datenmodelle ATKIS und GDF*. Beck, 1997.
- [26] Zhibiao Wu and Martha Palmer. “Verb semantics and lexical selection”. In: *arXiv preprint cmp-lg/9406033* (1994).
- [27] Emerson M. A. Xavier, Francisco J. Ariza-López, and Manuel A. Ureña-Cámara. “A Survey of Measures and Methods for Matching Geospatial Vector Datasets”. In: *ACM Computing Surveys* 49.2 (Nov. 2016), pp. 1–34. DOI: 10.1145/2963147.
- [28] Meng Zhang and Liqui Meng. “An iterative road-matching approach for the integration of postal data”. In: *Computers, Environment and Urban Systems* 31.5 (Sept. 2007), pp. 597–615. DOI: 10.1016/j.compenvurbsys.2007.08.008.
- [29] Xiang Zhang et al. “Data matching of building polygons at multiple map scales improved by contextual information and relaxation”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 92 (June 2014), pp. 147–163. DOI: 10.1016/j.isprsjprs.2014.03.010.
- [30] Zuoquan Zhao and Roger R. Stough. “Measuring Similarity Among Various Shapes Based on Geometric Matching”. In: *Geographical Analysis* 37.4 (Oct. 2005), pp. 410–422. DOI: 10.1111/j.0016-7363.2005.03704001.x.

Appendix A

Terminology

This appendix summarizes the new terminologies used in this report.

1. **Source set:** the main geographic object set used in a matching task.
2. **Reference set:** another geographic object set used in a matching task. In conflation task some object attributes in reference set will be merged onto corresponding objects in the source set.
3. **Match set:** the output of the matching algorithms; and is also the input of the validation algorithm. This dataset contains the matches to be validated.
4. **Valid match:** an input match that is classified as correct by the validation algorithm. While correctness describes the essence of a match from the view of ground truth, validness describes the validation result on a match.
5. **Invalid match:** an input match that is classified as incorrect by the validation algorithm.
6. **Missing match:** a match that is not included in the input matches, but it should be correct according to the ground truth.
7. **Alone Object:** a geospatial object which is not involved in any match.
8. **Candidate match:** an input match that the validation algorithm is processing, but has not been classified as valid or invalid yet.
9. **Supporting match / contextual match:** each match has a set of supporting matches, which build up the context for it. The supporting matches will support the validation of the candidate match (i.e. the relative spatial relations between the candidate objects and supporting objects are the material for context similarity calculation).
10. **Context of a match:** a set of surrounding matches, which are used to support the context similarity computation.
11. **Context similarity of a match:** used to evaluate the similarity between spatial relations of two objects to their contextual objects. A high context similarity indicates the two objects involved in this match locate at the corresponding places in their contexts.
12. **Object similarity of a match:** used to evaluate the correspondence between the individual attributes of two objects.
13. **Confidence level of a match:** a weighted sum of context similarity and object similarity; comprehensively evaluates the similarity between two objects. If the confidence level of a match reaches to the validation threshold, this match will be classified as valid.
14. **Valid match set:** one of the three output datasets of the validation algorithm, contains all input matches which are classified as valid.

15. **Invalid match set:** one of the three output datasets of the validation algorithm, contains all input matches which are classified as invalid.
16. **Missing match set:** one of the three output datasets of the validation algorithm, contains all detected missing matches.

Appendix B

Experiment Data

B.1 Context Scope

Table B.1: Influence of context scope on performance of VIM for Arlington dataset

Dataset: Arlington; Task: Validating Inpt Matches							Valid Match Set		Invalid Match Set	
Context Scope	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
1	1623	62	1621	2	4	58	99.88%	96.55%	6.45%	66.67%
3	1617	68	1616	1	5	63	99.94%	96.25%	7.35%	83.33%
5	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
7	1665	20	1664	1	5	15	99.94%	99.11%	25.00%	83.33%
9	1668	17	1667	1	5	12	99.94%	99.29%	29.41%	83.33%
11	1671	14	1670	1	5	9	99.94%	99.46%	35.71%	83.33%
13	1672	13	1671	1	5	8	99.94%	99.52%	38.46%	83.33%
15	1674	11	1673	1	5	6	99.94%	99.64%	45.45%	83.33%
17	1677	8	1676	1	5	3	99.94%	99.82%	62.50%	83.33%
19	1677	8	1676	1	5	3	99.94%	99.82%	62.50%	83.33%
21	1680	5	1677	3	3	2	99.82%	99.88%	60.00%	50.00%
23	1680	5	1677	3	3	2	99.82%	99.88%	60.00%	50.00%
25	1681	4	1676	5	1	3	99.70%	99.82%	25.00%	16.67%
27	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
29	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
31	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
33	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
35	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
37	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
39	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
41	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
43	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
45	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
47	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
49	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%

Table B.2: Influence of context scope on performance of DMM for Arlington dataset

Dataset: Arlington; Task: Detecting Missing Matches						Missing Match Set	
Context Scope	Detected	TP	FP	TN	FN	Precision	Recall
1	158	154	4	62	112	97.47%	57.89%
3	137	136	1	65	130	99.27%	51.13%
5	157	156	1	65	110	99.36%	58.65%
7	192	190	2	64	76	98.96%	71.43%
9	196	192	4	62	74	97.96%	72.18%
11	221	217	4	62	49	98.19%	81.58%
13	226	221	5	61	45	97.79%	83.08%
15	234	226	8	58	40	96.58%	84.96%
17	244	236	8	58	30	96.72%	88.72%
19	246	238	8	58	28	96.75%	89.47%
21	249	241	8	58	25	96.79%	90.60%
23	251	243	8	58	23	96.81%	91.35%
25	253	244	9	57	22	96.44%	91.73%
27	254	245	9	57	21	96.46%	92.11%
29	255	245	10	56	21	96.08%	92.11%
31	256	245	11	55	21	95.70%	92.11%
33	257	246	11	55	20	95.72%	92.48%
35	259	247	12	54	19	95.37%	92.86%
37	260	248	12	54	18	95.38%	93.23%
39	260	248	12	54	18	95.38%	93.23%
41	260	248	12	54	18	95.38%	93.23%
43	262	250	12	54	16	95.42%	93.98%
45	262	250	12	54	16	95.42%	93.98%
47	262	250	12	54	16	95.42%	93.98%
49	262	250	12	54	16	95.42%	93.98%

Table B.3: Influence of context scope on performance of VIM for Santa Clara dataset

Dataset: Santa Clara; Task: Validating Input Matches							Valid Match Set		Invalid Match Set	
Context Scope	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
1	2224	125	2218	6	8	117	99.73%	94.99%	6.40%	57.14%
3	2203	146	2200	3	11	135	99.86%	94.22%	7.53%	78.57%
5	2229	120	2228	1	13	107	99.96%	95.42%	10.83%	92.86%
7	2253	96	2248	5	9	87	99.78%	96.27%	9.38%	64.29%
9	2258	91	2252	6	8	83	99.73%	96.45%	8.79%	57.14%
11	2274	75	2266	8	6	69	99.65%	97.04%	8.00%	42.86%
13	2278	71	2270	8	6	65	99.65%	97.22%	8.45%	42.86%
15	2283	66	2275	8	6	60	99.65%	97.43%	9.09%	42.86%
17	2286	63	2276	10	4	59	99.56%	97.47%	6.35%	28.57%
19	2286	63	2276	10	4	59	99.56%	97.47%	6.35%	28.57%
21	2289	60	2279	10	4	56	99.56%	97.60%	6.67%	28.57%
23	2290	59	2279	11	3	56	99.52%	97.60%	5.08%	21.43%
25	2293	56	2282	11	3	53	99.52%	97.73%	5.36%	21.43%
27	2293	56	2283	10	4	52	99.56%	97.77%	7.14%	28.57%
29	2296	53	2284	12	2	51	99.48%	97.82%	3.77%	14.29%
31	2293	56	2283	10	4	52	99.56%	97.77%	7.14%	28.57%
33	2294	55	2284	10	4	51	99.56%	97.82%	7.27%	28.57%
35	2296	53	2285	11	3	50	99.52%	97.86%	5.66%	21.43%
37	2297	52	2286	11	3	49	99.52%	97.90%	5.77%	21.43%
39	2299	50	2287	12	2	48	99.48%	97.94%	4.00%	14.29%
41	2298	51	2287	11	3	48	99.52%	97.94%	5.88%	21.43%
43	2298	51	2287	11	3	48	99.52%	97.94%	5.88%	21.43%
45	2300	49	2288	12	2	47	99.48%	97.99%	4.08%	14.29%
47	2301	48	2289	12	2	46	99.48%	98.03%	4.17%	14.29%
49	2301	48	2289	12	2	46	99.48%	98.03%	4.17%	14.29%

Table B.4: Influence of context scope on performance of DMM for Santa Clara dataset

Dataset: Santa Clara; Task: Detecting Missing Matches						Missing Match Set	
Context Scope	Detected	TP	FP	TN	FN	Precision	Recall
1	431	429	2	418	65	99.54%	86.84%
3	410	410	0	420	84	100.00%	83.00%
5	419	419	0	420	75	100.00%	84.82%
7	443	443	0	420	51	100.00%	89.68%
9	446	446	0	420	48	100.00%	90.28%
11	451	451	0	420	43	100.00%	91.30%
13	452	451	1	419	43	99.78%	91.30%
15	457	456	1	419	38	99.78%	92.31%
17	461	460	1	419	34	99.78%	93.12%
19	463	462	1	419	32	99.78%	93.52%
21	464	463	1	419	31	99.78%	93.72%
23	466	464	2	418	30	99.57%	93.93%
25	466	464	2	418	30	99.57%	93.93%
27	468	466	2	418	28	99.57%	94.33%
29	468	466	2	418	28	99.57%	94.33%
31	469	467	2	418	27	99.57%	94.53%
33	470	468	2	418	26	99.57%	94.74%
35	471	468	3	417	26	99.36%	94.74%
37	471	468	3	417	26	99.36%	94.74%
39	471	468	3	417	26	99.36%	94.74%
41	472	468	4	416	26	99.15%	94.74%
43	472	468	4	416	26	99.15%	94.74%
45	473	469	4	416	25	99.15%	94.94%
47	474	469	5	415	25	98.95%	94.94%
49	475	470	5	415	24	98.95%	95.14%

B.2 Angle Tolerance

Table B.5: Influence of angle tolerance on performance of VIM for Arlington dataset

Dataset: Arlington; Task: Validating Input Matches							Valid Match Set		Invalid Match Set	
Angle Tolerance	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
1	1118	567	1118	0	6	561	100.00%	66.59%	1.06%	100.00%
3	1566	119	1565	1	5	114	99.94%	93.21%	4.20%	83.33%
5	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
7	1668	17	1666	2	4	13	99.88%	99.23%	23.53%	66.67%
9	1676	9	1674	2	4	5	99.88%	99.70%	44.44%	66.67%
11	1678	7	1676	2	4	3	99.88%	99.82%	57.14%	66.67%
13	1678	7	1676	2	4	3	99.88%	99.82%	57.14%	66.67%
15	1680	5	1676	4	2	3	99.76%	99.82%	40.00%	33.33%
17	1682	3	1677	5	1	2	99.70%	99.88%	33.33%	16.67%
19	1683	2	1678	5	1	1	99.70%	99.94%	50.00%	16.67%
21	1684	1	1678	6	0	1	99.64%	99.94%	0.00%	0.00%
23	1684	1	1678	6	0	1	99.64%	99.94%	0.00%	0.00%
25	1685	0	1679	6	0	0	99.64%	100.00%	N/A	0.00%
27	1685	0	1679	6	0	0	99.64%	100.00%	N/A	0.00%
29	1685	0	1679	6	0	0	99.64%	100.00%	N/A	0.00%

Table B.6: Influence of angle tolerance on performance of DMM for Arlington dataset

Dataset: Arlington; Task: Detecting Missing Matches						Missing Match Set	
Angle Tolerance	Detected	TP	FP	TN	FN	Precision	Recall
1	17	17	0	66	249	100.00%	6.39%
3	97	97	0	66	169	100.00%	36.47%
5	157	156	1	65	110	99.36%	58.65%
7	186	184	2	64	82	98.92%	69.17%
9	218	213	5	61	53	97.71%	80.08%
11	236	228	8	58	38	96.61%	85.71%
13	248	237	11	55	29	95.56%	89.10%
15	255	243	12	54	23	95.29%	91.35%
17	258	245	13	53	21	94.96%	92.11%
19	263	250	13	53	16	95.06%	93.98%
21	267	254	13	53	12	95.13%	95.49%
23	270	257	13	53	9	95.19%	96.62%
25	271	257	14	52	9	94.83%	96.62%
27	271	257	14	52	9	94.83%	96.62%
29	271	257	14	52	9	94.83%	96.62%

Table B.7: Influence of angle tolerance on performance of VIM for Santa Clara dataset

Dataset: Santa Clara; Task: Validating Input Matches							Valid Match Set		Invalid Match Set	
Angle Tolerance	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
1	1753	596	1753	0	14	582	100.00%	75.07%	2.35%	100.00%
3	2151	198	2150	1	13	185	99.95%	92.08%	6.57%	92.86%
5	2229	120	2228	1	13	107	99.96%	95.42%	10.83%	92.86%
7	2254	95	2249	5	9	86	99.78%	96.32%	9.47%	64.29%
9	2275	74	2268	7	7	67	99.69%	97.13%	9.46%	50.00%
11	2290	59	2277	13	1	58	99.43%	97.52%	1.69%	7.14%
13	2297	52	2284	13	1	51	99.43%	97.82%	1.92%	7.14%
15	2301	48	2288	13	1	47	99.44%	97.99%	2.08%	7.14%
17	2307	42	2294	13	1	41	99.44%	98.24%	2.38%	7.14%
19	2313	36	2300	13	1	35	99.44%	98.50%	2.78%	7.14%
21	2317	32	2304	13	1	31	99.44%	98.67%	3.13%	7.14%
23	2321	28	2308	13	1	27	99.44%	98.84%	3.57%	7.14%
25	2326	23	2313	13	1	22	99.44%	99.06%	4.35%	7.14%
27	2332	17	2319	13	1	16	99.44%	99.31%	5.88%	7.14%
29	2337	12	2324	13	1	11	99.44%	99.53%	8.33%	7.14%

Table B.8: Influence of angle tolerance on performance of DMM for Santa Clara dataset

Dataset: Santa Clara; Task: Detecting Missing Matches							Missing Match Set	
Angle Tolerance	Detected	TP	FP	TN	FN		Precision	Recall
1	196	196	0	420	298		100.00%	39.68%
3	374	374	0	420	120		100.00%	75.71%
5	419	419	0	420	75		100.00%	84.82%
7	441	440	1	419	54		99.77%	89.07%
9	444	444	0	420	50		100.00%	89.88%
11	448	447	1	419	47		99.78%	90.49%
13	451	447	4	416	47		99.11%	90.49%
15	461	453	8	412	41		98.26%	91.70%
17	470	458	12	408	36		97.45%	92.71%
19	468	455	13	407	39		97.22%	92.11%
21	473	459	14	406	35		97.04%	92.91%
23	469	453	16	404	41		96.59%	91.70%
25	472	454	18	402	40		96.19%	91.90%
27	477	458	19	401	36		96.02%	92.71%
29	475	455	20	400	39		95.79%	92.11%

B.3 Context Similarity Weight

Table B.9: Influence of context similarity weight on performance of VIM for Arlington dataset

Dataset: Arlington; Task: Validating Input Matches							Valid Match Set		Invalid Match Set	
Context Similarity Weight	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
0.00	1544	141	1544	0	6	135	100.00%	91.96%	4.26%	100.00%
0.05	1575	110	1575	0	6	104	100.00%	93.81%	5.45%	100.00%
0.10	1597	88	1597	0	6	82	100.00%	95.12%	6.82%	100.00%
0.15	1611	74	1611	0	6	68	100.00%	95.95%	8.11%	100.00%
0.20	1620	65	1620	0	6	59	100.00%	96.49%	9.23%	100.00%
0.25	1629	56	1629	0	6	50	100.00%	97.02%	10.71%	100.00%
0.30	1639	46	1638	1	5	41	99.94%	97.56%	10.87%	83.33%
0.35	1643	42	1642	1	5	37	99.94%	97.80%	11.90%	83.33%
0.40	1647	38	1646	1	5	33	99.94%	98.03%	13.16%	83.33%
0.45	1648	37	1647	1	5	32	99.94%	98.09%	13.51%	83.33%
0.50	1647	38	1646	1	5	33	99.94%	98.03%	13.16%	83.33%
0.55	1648	37	1647	1	5	32	99.94%	98.09%	13.51%	83.33%
0.60	1651	34	1650	1	5	29	99.94%	98.27%	14.71%	83.33%
0.65	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
0.70	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
0.75	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
0.80	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
0.85	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
0.90	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
0.95	1650	35	1649	1	5	30	99.94%	98.21%	14.29%	83.33%
1.00	1664	21	1663	1	5	16	99.94%	99.05%	23.81%	83.33%

Table B.10: Influence of context similarity weight on performance of DMM for Arlington dataset

Dataset: Arlington; Task: Detecting Missing Matches						Missing Match Set	
Context Similarity Weight	Detected	TP	FP	TN	FN	Precision	Recall
0.00	5	5	0	66	261	100.00%	1.88%
0.05	20	20	0	66	246	100.00%	7.52%
0.10	36	36	0	66	230	100.00%	13.53%
0.15	41	41	0	66	225	100.00%	15.41%
0.20	66	66	0	66	200	100.00%	24.81%
0.25	81	81	0	66	185	100.00%	30.45%
0.30	100	100	0	66	166	100.00%	37.59%
0.35	117	117	0	66	149	100.00%	43.98%
0.40	125	125	0	66	141	100.00%	46.99%
0.45	134	134	0	66	132	100.00%	50.38%
0.50	140	140	0	66	126	100.00%	52.63%
0.55	143	143	0	66	123	100.00%	53.76%
0.60	146	146	0	66	120	100.00%	54.89%
0.65	148	147	1	65	119	99.32%	55.26%
0.70	149	148	1	65	118	99.33%	55.64%
0.75	154	153	1	65	113	99.35%	57.52%
0.80	157	156	1	65	110	99.36%	58.65%
0.85	161	159	2	64	107	98.76%	59.77%
0.90	163	161	2	64	105	98.77%	60.53%
0.95	163	161	2	64	105	98.77%	60.53%
1.00	196	192	4	62	74	97.96%	72.18%

Table B.11: Influence of context similarity weight on performance of VIM for Santa Clara dataset

Dataset: Santa Clara; Task: Validating Input Matches							Valid Match Set		Invalid Match Set	
Context Similarity Weight	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
0.00	2006	343	2006	0	14	329	100.00%	85.91%	4.08%	100.00%
0.05	2058	291	2058	0	14	277	100.00%	88.14%	4.81%	100.00%
0.10	2081	268	2081	0	14	254	100.00%	89.12%	5.22%	100.00%
0.15	2111	238	2111	0	14	224	100.00%	90.41%	5.88%	100.00%
0.20	2144	205	2143	1	13	192	99.95%	91.78%	6.34%	92.86%
0.25	2180	169	2179	1	13	156	99.95%	93.32%	7.69%	92.86%
0.30	2196	153	2195	1	13	140	99.95%	94.00%	8.50%	92.86%
0.35	2201	148	2200	1	13	135	99.95%	94.22%	8.78%	92.86%
0.40	2209	140	2208	1	13	127	99.95%	94.56%	9.29%	92.86%
0.45	2215	134	2214	1	13	121	99.95%	94.82%	9.70%	92.86%
0.50	2223	126	2222	1	13	113	99.96%	95.16%	10.32%	92.86%
0.55	2225	124	2223	2	12	112	99.91%	95.20%	9.68%	85.71%
0.60	2226	123	2224	2	12	111	99.91%	95.25%	9.76%	85.71%
0.65	2228	121	2226	2	12	109	99.91%	95.33%	9.92%	85.71%
0.70	2228	121	2226	2	12	109	99.91%	95.33%	9.92%	85.71%
0.75	2229	120	2228	1	13	107	99.96%	95.42%	10.83%	92.86%
0.80	2229	120	2228	1	13	107	99.96%	95.42%	10.83%	92.86%
0.85	2229	120	2227	2	12	108	99.91%	95.37%	10.00%	85.71%
0.90	2230	119	2229	1	13	106	99.96%	95.46%	10.92%	92.86%
0.95	2232	117	2229	3	11	106	99.87%	95.46%	9.40%	78.57%
1.00	2258	91	2251	7	7	84	99.69%	96.40%	7.69%	50.00%

Table B.12: Influence of context similarity weight on performance of DMM for Santa Clara dataset

Dataset: Santa Clara; Task: Detecting Missing Matches						Missing Match Set	
Context Similarity Weight	Detected	TP	FP	TN	FN	Precision	Recall
0.00	0	0	0	420	494	N/A	0.00%
0.05	3	3	0	420	491	100.00%	0.61%
0.10	4	4	0	420	490	100.00%	0.81%
0.15	7	7	0	420	487	100.00%	1.42%
0.20	14	14	0	420	480	100.00%	2.83%
0.25	34	34	0	420	460	100.00%	6.88%
0.30	69	69	0	420	425	100.00%	13.97%
0.35	110	110	0	420	384	100.00%	22.27%
0.40	161	161	0	420	333	100.00%	32.59%
0.45	220	220	0	420	274	100.00%	44.53%
0.50	281	281	0	420	213	100.00%	56.88%
0.55	339	339	0	420	155	100.00%	68.62%
0.60	385	385	0	420	109	100.00%	77.94%
0.65	407	407	0	420	87	100.00%	82.39%
0.70	417	417	0	420	77	100.00%	84.41%
0.75	418	418	0	420	76	100.00%	84.62%
0.80	419	419	0	420	75	100.00%	84.82%
0.85	422	422	0	420	72	100.00%	85.43%
0.90	423	423	0	420	71	100.00%	85.63%
0.95	424	424	0	420	70	100.00%	85.83%
1.00	454	452	2	418	42	99.56%	91.50%

B.4 Validation Threshold

Table B.13: Influence of validation threshold on performance of VIM for Arlington dataset

Dataset: Arlington; Task: Validating Input Matches							Valid Match Set		Invalid Match Set	
Validation Threshold	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
0.50	1677	8	1675	2	4	4	99.88%	99.76%	50.00%	66.67%
0.55	1676	9	1674	2	4	5	99.88%	99.70%	44.44%	66.67%
0.60	1672	13	1671	1	4	9	99.94%	99.46%	30.77%	80.00%
0.65	1669	16	1668	1	5	11	99.94%	99.34%	31.25%	83.33%
0.70	1666	19	1665	1	5	14	99.94%	99.17%	26.32%	83.33%
0.75	1664	21	1663	1	5	16	99.94%	99.05%	23.81%	83.33%
0.80	1649	36	1648	1	5	31	99.94%	98.15%	13.89%	83.33%
0.85	1616	69	1615	1	5	64	99.94%	96.19%	7.25%	83.33%
0.90	1610	75	1608	2	4	71	99.88%	95.77%	5.33%	66.67%
0.95	1583	102	1583	0	6	96	100.00%	94.28%	5.88%	100.00%
1.00	0	1685	0	0	6	1679	N/A	0.00%	0.36%	100.00%

Table B.14: Influence of validation threshold on performance of DMM for Arlington dataset

Dataset: Arlington; Task: Detecting Missing Matches							Missing Match Set	
Validation Threshold	Detected	TP	FP	TN	FN		Precision	Recall
0.50	245	235	10	56	31		95.92%	88.35%
0.55	239	230	9	57	36		96.23%	86.47%
0.60	230	222	8	58	44		96.52%	83.46%
0.65	206	200	6	60	66		97.09%	75.19%
0.70	196	192	4	62	74		97.96%	72.18%
0.75	195	191	4	62	75		97.95%	71.80%
0.80	157	156	1	65	110		99.36%	58.65%
0.85	138	137	1	65	129		99.28%	51.50%
0.90	135	135	0	66	131		100.00%	50.75%
0.95	63	63	0	66	203		100.00%	23.68%
1.00	0	0	0	66	266		N/A	0.00%

Table B.15: Influence of validation threshold on performance of VIM for Santa Clara dataset

Dataset: Santa Clara; Task: Validating Input Matches							Valid Match Set		Invalid Match Set	
Validation Threshold	Valid	Invalid	TP	FP	TN	FN	Precision	Recall	Precision	Recall
0.50	2293	56	2281	12	2	54	99.48%	97.69%	3.57%	14.29%
0.55	2288	61	2276	12	2	59	99.48%	97.47%	3.28%	14.29%
0.60	2284	65	2273	11	3	62	99.52%	97.34%	4.62%	21.43%
0.65	2269	80	2260	9	5	75	99.60%	96.79%	6.25%	35.71%
0.70	2259	90	2254	5	9	81	99.78%	96.53%	10.00%	64.29%
0.75	2254	95	2249	5	9	86	99.78%	96.32%	9.47%	64.29%
0.80	2229	120	2228	1	13	107	99.96%	95.42%	10.83%	92.86%
0.85	2201	148	2197	4	10	138	99.82%	94.09%	6.76%	71.43%
0.90	2200	149	2196	4	10	139	99.82%	94.05%	6.71%	71.43%
0.95	2126	223	2123	3	11	212	99.86%	90.92%	4.93%	78.57%
1.00	0	2349	0	0	14	2335	N/A	0.00%	0.60%	100.00%

Table B.16: Influence of validation threshold on performance of DMM for Santa Clara dataset

Dataset: Santa Clara; Task: Detecting Missing Matches							Missing Match Set	
Validation Threshold	Detected	TP	FP	TN	FN		Precision	Recall
0.50	466	462	4	416	32		99.14%	93.52%
0.55	463	461	2	418	33		99.57%	93.32%
0.60	459	457	2	418	37		99.56%	92.51%
0.65	455	453	2	418	41		99.56%	91.70%
0.70	450	449	1	419	45		99.78%	90.89%
0.75	444	443	1	419	51		99.77%	89.68%
0.80	419	419	0	420	75		100.00%	84.82%
0.85	411	411	0	420	83		100.00%	83.20%
0.90	381	381	0	420	113		100.00%	77.13%
0.95	14	14	0	420	480		100.00%	2.83%
1.00	0	0	0	420	494		N/A	0.00%

Appendix C

Meeting Records

This appendix includes a summary of the topics and content of each weekly supervision meeting. This record provides some auxiliary material for the project timeline.

Oct 5th

- Showed a draft of proposal, discussed about the structure and requirement of the proposal
- Proposed an initial idea about object-based validation using context measure
- Filled in the preliminary ethics form

Oct 12th

- Discussed about writing reference and citation in Latex using .bib file
- Looked through the proposal draft and marked some places for improvement

Oct 19th

- Showed the modified proposal
- Discussed some problems in setting the article format in Latex
- Discussed the framework of the Java program

Oct 26th

- Discussed about choosing OpenJump as the framework
- Asked for some websites for collecting geospatial data

Nov 2nd

- Showed some collected sample data
- Showed the progress on running OpenJump source code, and adding own plugin into it
- Discussed about the difficulties met in running OpenJump using Eclipse and IntelliJ

Nov 9th

- Showed the progress on installing and running a matching plugin in OpenJump
- Discussed about the design of the final Java program, marked some useful features of OpenJump
- Discussed about the structure of interim report, and the its difference with proposal

Nov 16th

- Discussed some detailed assignment of content in interim report
- Had a short review on the current progress, made a work plan for the following weeks

Nov 23rd

- Checked the progress on interim report

- Discussed about a new idea of matching-relation-based validation algorithm
- Discussed the difference between matching task and validation task

Nov 30th

- Introduced the modified validation algorithm: topologic network validation with strict checker
- Discussed about the flaws and limitations of the validation algorithm

Dec 7th

- Had a quick viewing on the interim report
- Discussed about the modified validation algorithm: main validator and auxiliary validator
- Discussed about the tricky part of the algorithm, proposed the idea of dynamic validation

Dec 14th

- Showed the latest change on validation algorithm, e.g. the expression of confidence level, stepped-waterfall backtracking system
- Discussed about schedule and work plan after exams

Mar 4th

- Showed the progress and modification on validation algorithm, rename the two parts of evaluation as context similarity and object similarity
- Proposed a new idea of using star calculus to calculate context similarity
- Demonstrates the developed software, and how the matching plug-in cooperates with the new validation plug-in and scrutinize plug-in
- Discussed about the experiment settings, planned to test the performance of validating input matches and detecting missing matches separately

Mar 11th

- Discussed about some drawbacks of star calculus discovered in implementation stage; modify the star calculus by adding the mechanism of angle tolerance
- Showed a draft about final report structure
- Discussed about experiment settings further, including the order of tests on each parameter, number of datasets, evaluation criteria toward collected experiment data

Mar 18th

- Demonstrated the new version of implemented context similarity calculator, where the star calculus measure is updated as angle difference measure
- Displayed the collected pairs of datasets, discussed about how to select datasets for experiments

Mar 25th

- Got some feedback about the modification on background, methodology and implementation chapters in final report
- Consulted about how to define the ground truth about geographic object matching

Apr 1st

- Showed the progress on final report writing, e.g. new class diagram of implemented software, the pseudo code in small functions
- Displayed the experiment result, explained how the experiments are done, and the criteria used in the analysis)
- Discussed about the strength and some potential drawbacks of the validation algorithm discovered in experiments

Apr 8th

- Displayed the new version of experiment data diagrams, confirmed the experiment setting (i.e. each set of experiments focus on one parameter, instead of one functionality)
- Discussed the aspects of analyzing experiment result
- Discussed about the problem that few TN cases occurs in data set, and how to implement a fair evaluation on unbalance data set; proposed the new evaluation criteria of negative predict value and specificity

Apr 15th

- Discarded negative predict value and specificity, apply uniform criteria (i.e. precision and recall) in the evaluation of each output dataset of the validation algorithm
- Got some feedback about the final report, e.g. strengthening the explanation in background research part, adding an overview section at the beginning of the experiment chapter

Apr 22th

- Got some feedback about the final report, i.e. the deficiency in project management chapter
- Confirmed the requirement about code submission, discussed about the content and structure of the presentation video