Automatic Generation of Geographically Accurate Bus Route Maps and its Evaluation

Sogo Mizutani, Yonghwan Kim, and Daisuke Yamamoto
Nagoya Institute of Technology, Nagoya, Japan.
emails: s.mizutani.814@stn.nitech.ac.jp, kim@nitech.ac.jp, daisuke@nitech.ac.jp

***Abstract*—There have been many studies on the automatic generation of deformed route maps, but there have only been a few studies on the automatic generation of geographically accurate route maps. This is because mapping route data and bus route data and drawing them on a map are difficult tasks due to various constraints, such as route placement problems. In this study we estimate bus routes that use bus stop coordinate series and strokes and propose an automatic bus route map generation method based on this estimation. In the proposed method, bus stop nodes are first generated on the road network from the bus stop coordinate series. Then, the route between two adjacent bus stop nodes is estimated using the road priority search method, and this is set as the bus route. In the road priority search method, the route with the fewest number of left and right turns between bus stop nodes is estimated as the bus route. Additionally, when drawing multiple routes, the placement order of overlapping sections is dynamically calculated so that intersections when turning left or right are reduced. The experimental results applying the proposed method to 30 bus routes show that the geographically accurate route maps with few intersections among these routes can be generated correctly.**

Keywords-network; bus route map; stroke.

I. INTRODUCTION

The preliminary version of this paper is presented in [1]. This paper includes more detailed evaluations and additional experiments to help understand this work and provides a more detailed discussion about the contribution of this study.

The advancement of public transportation has received considerable attention recently, as typified by Mobility as a Service (MaaS). Among these advancements, buses and bus routes are one of the means of transportation that are at the core of public transportation, and they are of high importance. In a metropolitan area, these transportation systems can be very complex with over hundreds of bus routes. Generally, when using public transportation such as trains and buses, users think about how to get to their destination by looking at route maps. Therefore, there is a need for more understandable and accurate bus route maps.

There are two types of route maps: deformed route maps and geographically accurate route maps. Deformed route maps schematically show the locations and connections of stations and bus stops. As shown in Figure 1, a deformed route map does not necessarily need to be geographically accurate in terms of direction and distance, and knowing the relative positions of stations on a route and their connections is sufficient. Geographically accurate route maps are drawn on a route map based on accurate location information, as shown in Figure 2. As a result, there is an advantage in that it is possible to obtain information about the bus stop and the nearby amenities in addition to its position with respect to the bus line. Moreover, when multiple routes are drawn on one route, the routes overlap each other, thereby reducing visibility. Improving visibility requires arranging routes to minimize overlap between routes. This is called the route placement problem. The route placement problem is a type of combinatorial optimization problem which is known as NP-hard.

Furthermore, online map systems such as Google Maps [2] and OpenStreetMap [3] have become popular in recent years. The online map system allows users to freely change the scale and position of the map and view the desired location. Some APIs, such as Leaflet [4], can control the online map system and permits the drawing of lines and objects on the online map. The use of these technologies enables the expression of geographically accurate route maps by drawing route maps on online maps. Furthermore, the popularization of the General Transit Feed Specification (GTFS) [5] standard has led to transportation system data such as route buses and subways being open to the public. GTFS includes not only timetable data but also bus stop coordinates (expressed by latitude and longitude) and route connection data. Moreover, there has been a problem where the route coordinate series is an optional item and is not necessarily included.

The purpose of this research is to propose a method that generates highly visible and geographically accurate bus routes by estimating routes on road networks from bus stop coordinates and route data included in GTFS and minimizing the overlap between routes.

The remainder of this paper is organized as follows. Section 2 describes the problems that are to be addressed in this study. Section 3 describes the related work. Section 4 outlines the proposed system. The details of the proposed method are described in Section 5. Section 6 reports the experimental results, and Section 7 provides a summary.

Figure 1. Example of deformed route map (cited from Nagoya City Transportation Bureau.)

Figure 2. Example of geographically accurate route map (cited from Nagoya City Transportation Bureau.)

II. PROBLEMS

There have been many previous studies on the automatic generation of deformed route maps [6-12], but there have been few studies on the automatic generation of geographically accurate route maps associated with road networks [13]. In practice, most geographically accurate route maps are created manually, but it is very difficult to draw understandable route maps on a road map by associating the road and bus route coordinates. In particular, the number of bus routes is greater than that of railway networks, and there are many routes that overlap each other, so creating these maps require considerable time.

The automatic generation of geographically accurate bus route maps can raise the following issues:

Issue 1) GTFS includes bus stop coordinates, but the coordinates of the routes that connect them are optional items and not necessarily included. Therefore, when trying to draw a geographically accurate route on a road network, the path of the route needs to be estimated from the bus stop coordinates and road network. Additionally, estimating the path requires a bus stop node on the road network. However, the bus stop coordinates indicate the position of the boarding point, and they do not necessarily exist on roads.

Issue 2) A method of determining the placement order between routes by brute force for each road link can be used to minimize the overlap between routes. However, this is an NP-hard problem, and this method has the drawback of exponentially increasing the computational load.

Issue 3) Improving the visibility of the route map requires drawing the routes by shifting them, but the placement order of the routes results in a route map with many intersections. The placement order of routes with fewer intersections needs to be automatically obtained.

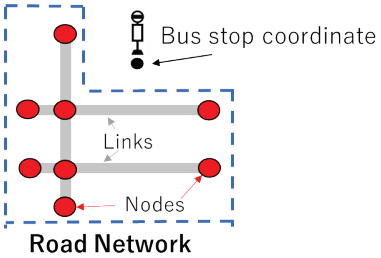

Figure 3. Bus stop coordinates and road network.

Therefore, in this research, we propose a system with the following features:

Feature 1) Stroke-based route generation function

A bus stop node to the nearest point on the road link closest to the bus stop is added from the road network and bus stop coordinates, as shown in Figure 3. Furthermore, the path between the adjacent bus stop nodes is estimated with the road priority search, and that path is set as the bus route path.

Feature 2) Bus stroke (BS)/bus stroke fragment (BSF) generation function

First, a path composed of road links that are estimated by the route path generation function is converted into a BS set composed of strokes. Then, we create a function that generates a BSF set from the BS set that considers bus route overlap. The BS/BSF model can be used to aggregate many road links into the minimum necessary number of BSFs. As a result, the number of combinations can be minimized, and the placement order can be determined efficiently.

Feature 3) Route placement/drawing function

The route placement order is decided from the route map composed of the BSF set to reduce the number of intersections. The routes are drawn on the online map based on these results.

III. RELATED WORK

Previously, research on route maps was mainly research on deformed maps, as shown below, unlike the proposed method. Hong et al. [6] proposed a method for the problem of the automatic generation of deformed route maps of subways. Onda et al. [7] proposed a method for automatically generating railway route maps for Tokyo subway routes. The directions between stations were limited to eight directions (in 45°-increments), and a mixed-integer programming problem (MIP) was used to automatically place route maps while preserving the geographical network topology. There are also many studies on the automatic generation of deformed route maps for subway route maps. Stott et al. [8][10] proposed the automatic generation of railway route maps using a multi-criteria optimization algorithm for appropriate route placement. A clustering method was applied, in which multiple evaluation criteria were set for rendering, and the sum of those results was used as the evaluation value. They proposed a route diagram generation mechanism that thus avoided the local minimum problem and found routes efficiently. Fink et al. [9] proposed a method of drawing routes using Bézier curves for railway route diagrams, which are often drawn linearly. Routes were expressed with the fewest number of Bézier curves using a graph-drawing algorithm based on a dynamic model. Furthermore, Wang and Peng [11] proposed a system that could interactively edit the layout of subway route maps. Route maps are usually drawn with a finite number of colored lines. Lloyd et al. [12] proposed a color-coding method for subway route maps.

An example of research on geographically accurate route maps includes Bast et al. [13] proposed a method of automatically generating geographically accurate route maps by using the connection relationships between stations and route position coordinates included in GTFS as inputs. In this research, they focused on the number of intersections between routes, they improved integer linear programming (ILP) and applied it to the optimization problem of the placement order of routes running in parallel in order to obtain a placement order with few intersections between routes at high speed. Furthermore, as the number of subway routes is small, there was no mention of a stroke reduction method like that of the proposed method, and because subways run underground, there was no need to associate routes with roads, like in bus routes.

In the present study, the concept of a stroke [14][15] is used. A stroke is a grouping of a road network based on cognitive psychology, representing a road that follows a path.

Research using road strokes includes those on road generalization. Zhang et al. [16] achieved road generalization by selecting characteristic roads based on the road connection relationships. Road generalization is a method that draws only major roads in a road network based on the length of the road stroke, and methods that achieve road generalization from facility search results [17][18] and methods that achieve road generalization in a Fisheye view format [19] have been proposed. A path search method using strokes [20] has also been proposed. However, there has been no research that attempts to apply strokes to the drawing of route maps.

IV. PROPOSED SYSTEM

In this section, we describe the configuration of the proposed system, data format and terminology definitions.

A. Configuration of proposed system

Figure 4 shows the configuration of the proposed system. The proposed system consists of four functions: a stroke-based route path generation function, BS/BSF generation function, route placement function, and route drawing function. The route path generation function generates bus stop nodes from the bus information and road data published in GTFS and generates route paths by searching for routes between adjacent bus stop nodes. The BS/BSF generation function generates a BS by grouping the route data in units of strokes and also generates a BSF by dividing the BS into overlapping sections of multiple BSs. Section 4 describes the details of the definition of BS/BSF. The route placement function sorts based on the rule that sets the placement order of BSF. Finally, the route drawing function draws the route on the online map and presents it to the user.

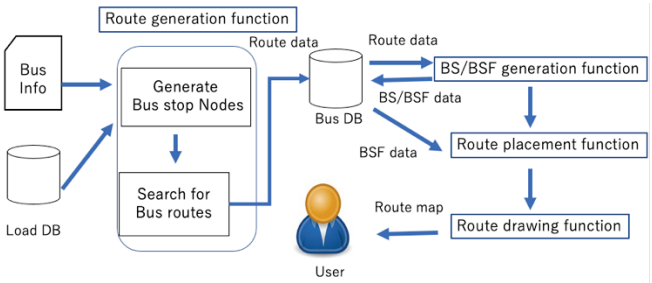

Figure 4. System configuration.

B. Data format and terminology definitions

The data formats and terminology used in this study are described as follows:

1) Definition of road data

We used OpenStreetMap as a road database. Because "road" is an ambiguous term, this paper defines road data by road links, nodes, and arcs, as shown in Figure 5. Nodes represent intersections and turns on the road network. A link indicates a path that connects nodes. A link has a start node and end node, and it becomes a directed graph given the direction of the road. The shape of the road is represented by a geometry-type arc format that is represented by a point sequence. Table 1 shows the data structure of the road link table. Additionally, OpenStreetMap stores the types of roads, such as highways, national roads, and pedestrian roads, such as road classes. Looped road links where the start node is the same as the end node are not addressed in this study. However, road links with a loop construct can be divided into

road links with a non-loop construct by adding a node at the midpoint of the link.

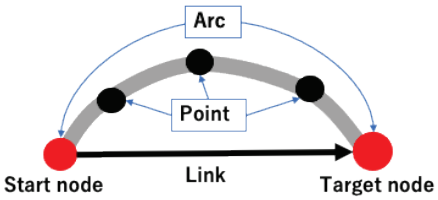

Figure 5. Configuration of road data.

TABLE I. ROAD LINK DATA FORMAT.

Column name	*Data type*	*Explanation*
Id	Integer	Link ID
Clazz	Integer	Road class
Source	Integer	Start node ID
Target	Integer	End node ID
x1	Double	Start node longitude
y1	Double	Start node latitude
x2	Double	End node longitude
y2	Double	End node latitude
Km	Double	Link length
geom_way	Geometry (LineString)	Link shape

2) Definition of stroke

A stroke [14][15] represents a series of road links that follow a path. A road network composed of strokes is called a stroke network. An example of a stroke network is shown in Figure 6 (right). In this example, based on stroke generation rules, the road links on the road network in Figure 6 (left) are grouped to generate a stroke network composed of colored strokes in Figure 6 (right).

Table 2 shows the stroke data format. A stroke consists of an ID indicating the stroke, a series of road link IDs included in the stroke, and its length and shape.

TABLE II. STROKE DATA FORMAT.

Column name	*Data type*	*Explanation*
id	Integer	Stroke ID
link_ids	Text	Included link ID series
stroke_length	Double	Stroke length
arc_series	Geometry (LineString)	Stroke shape

3) Bus data

In this study, we used the GTFS-JP format bus data that was released as open data in 2017 by the Nagoya City Transportation Bureau. We used three items: bus stop data, system data, and bus stop series.

Bus stop data is information that indicates the position and ID of a bus stop. Table 3 shows the data format. Bus stop data includes the bus stop ID, bus stop name, latitude and longitude of the bus stop, etc.

TABLE III. BUS STOP DATA FORMAT.

Column name	*Data type*	*Explanation*
Id	Integer	Bus stop ID
busstop_name	Varchar	Bus stop name
Lat	Double	Bus stop latitude
Lng	Double	Bus stop longitude
noriba_info	Varchar	Additional information
geom_way	Geometry (Point)	Latitude/longitude coordinates

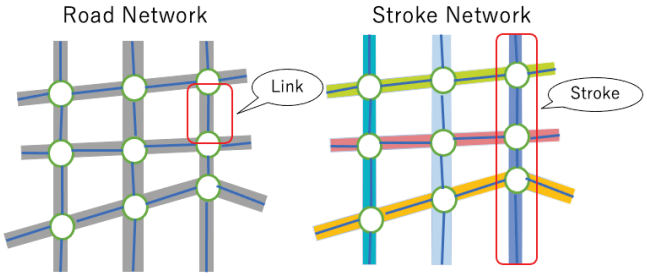

Figure 6. Stroke network.

System data is the data in which operation data is stored for each system of a bus route. The system data is stored in the system table, and information on the start and end points for the operation sections in the system, system code, route code, and direction code are stored. System data is uniquely identified by three items: system code, route code, and direction code.

The bus stop series stores the series of bus stop data that passes from the start point to the endpoint in the operation section of each system data.

V. PROPOSED METHOD

In this section, we describe the details of each proposed method.

A. Stroke-based route path generation function

The route path generation function generates a path on the road network that the bus will actually pass through as a road link series from the bus stop series. The main flow is to generate bus stop nodes from the road network and bus stop series. Stroke-based path search is then conducted on the generated network.

1) Bus stop node generation

As mentioned in Issue 1, a bus stop node needs to be generated from the bus stop coordinates. Specifically, the bus stop coordinates, road link, and road class are set as inputs, and the bus stop node is generated on the road link that is closest to the bus stop in the specified road class.

The bus stop node generation method is shown below. Note that functions starting with ST_ are functions provided by PostGIS [21].

- $L = (l_1, l_2, \cdots, l_i)$: Set of links
- $Class$: Road class
- Bus_{point} : Bus stop coordinates
- Bus_{node} : Bus stop node

Step 1) Among Bus_{point} and L, $Class$ obtains a set of neighboring links other than highways or connecting roads to expressways. Simultaneously, the ST_DWthin is used to obtain a set of neighboring links within 20 m.

Step 2) The link with the shortest distance in the set of nearby links is found using the ST_Distance function and is obtained as the nearest neighbor link.

Step 3) Bus_{point} is used to find the nearest point among the neighboring links, and the ratio r of the nearest point to the start and end points of the link is obtained using the ST_LineLocatePoint function. It is stored in the table as Bus_{node} using the ST_LineLocatePoint function from the obtained ratio r. The data format of the bus stop node is shown in Table 4.

The reason for limiting the road class to those other than expressways in Step 1 is as follows. In urban areas, general roads are often laid under elevated expressways, and it is impossible to determine which bus stop is based on latitude and longitude coordinates alone. Therefore, we used the property that there are almost no route bus stops on expressways and did not generate bus stop nodes on expressways. If the neighboring links are obtained only from the latitude and longitude, then there is a possibility that the wrong links, such as expressways, are obtained.

TABLE IV. BUS STOP NODE TABLE.

Column name	*Data type*	*Explanation*
id	Integer	Bus stop ID
link_id	Integer	Nearest neighbor road link ID
node_lat	Double	Bus stop node latitude
node_lng	Double	Bus stop node longitude
ratio	Double	Ratio on link

2) Creation of split link

Nodes on a link require splitting the road links and regenerating the road network to search for a path between bus stop nodes using the created bus stop nodes. In this study, a link that is obtained by splitting a road link at a bus stop node is termed a split link. The procedure for generating a split link is demonstrated as follows:

Step 1) Obtain the nearest neighbor link from the bus stop node table, check how many bus stop nodes there are on the link in the road database, and find the number of splits.

Step 2) If there are multiple bus stop nodes, then they are sorted based on Ratio, which is the bus stop node table ratio, and the links are split, in order, from the starting point.

Step 3) A new ID is allocated to the split link and stored in the split link table.

In the example of Figure 7, Link2, which is the road link of Figure 7 (top), is split at the point of the bus stop node. This increases the number of links from 3 to 4.

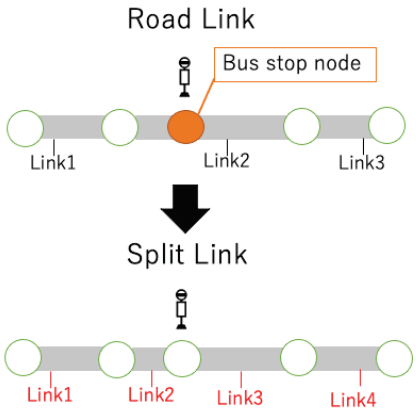

Figure 7. Split link.

3) Stroke-based route path search function

The stroke-based route path search function is a method of finding the distance between adjacent bus stop nodes by the stroke-based path search function. The specific steps are as follows:

- ID $= (id_1, id_2, \cdots, id_n)$: Bus stop ID list of the obtained route
- $Node = (N_1, N_2, \cdots, N_n)$: Node ID list
- $V \ni (s, g)$: Combination of start and end points
- $R = (r_1, r_2, \cdots, r_n)$: Path data list

Step 1) Obtain the ID for the specified system, route, and direction code from the bus stop order table.

Step 2) Obtain the bus stop node corresponding to the ID from the bus stop node table and add it to the $Node$.

Step 3) In v $= (N_i, N_{i+1}) \in V$, check if there is a record $(s, g) = (id_i, id_{i+1})$ *or* (id_{i+1}, id_i) in the path table.

Step 4) If it exists in Step 3, the acquired path data is assumed to be r_i.

Step 5) If it does not exist in Step 3, a path search in v is performed to find r_i.
Step 6) If all paths are found, R is stored in the path table.

In Step 3, the search time can be reduced by checking whether a path is already in the table and finding sections where a path search is not needed.

In Step 5, a road priority search is performed. A road priority search is a method that adopts the path with the shortest distance among paths with the smallest number of passing strokes.

The route bus has a long body, so the costs of turning left or right are high. Therefore, there is a tendency to select paths with as few right and left turns as possible as the bus route. Therefore, it was thought that selecting wide roads as the route path would be better. Thus, we chose road priority search instead of shortest path search, the latter of which is generally used in path search. Table 5 shows the generated path table format.

TABLE V. ROUTE PATH TABLE.

Column name	*Data type*	*Explanation*
route_code	Integer	System code
line_code	Integer	Route code
dir_code	Integer	Direction code
route_name	Varchar	System symbol
s_order	Integer	Start point order number
g_order	Integer	End point order number
s_gid	Integer	Start point bus stop ID
s_name	Varchar	Start point bus stop name
g_gid	Integer	End point bus stop ID
g_name	Varchar	End point bus stop name
geom_way	geometry(LineString)	Path shape
link_ids	Varchar	Link ID series

B. BS/BSF generation function

This subsection describes the definitions and algorithms of BS and BSF.

1) Bus stroke (BS)

BS is a representation of a bus route not as a series of road links but as a series of strokes. Bus routes often pass along roads, so it was thought that expressing a route as a set of strokes instead of as a set of road links could express it with a smaller number of links. Because the number of combinations can be reduced, this is expected to contribute to the speeding up of the combinatorial optimization problem. Figure 8 shows an example of converting a route path (number of links is 6) represented by a series of road links into a BS series (number of links is 3).

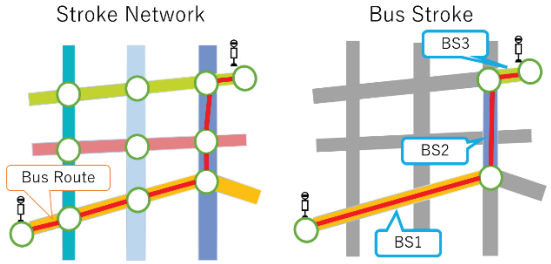

Figure 8. Example of bus stroke generation.

The BS generation procedure is shown as follows:
Step 1) The road link series is obtained from the path data of the desired route.
Step 2) The stroke that contains the acquired link series is obtained from the stroke table.
Step 3) A stroke series is generated in the order of the route paths and stored in the BS table.
The BS table that is generated by the above procedure is shown in Table 6 below.

TABLE VI. BS TABLE.

Column name	*Data type*	*Explanation*
route_code	Integer	System code
line_code	Integer	Route code
dir_code	Integer	Direction code
num	Integer	Order number
stroke_id	Integer	Stroke ID
link_ids	Varchar	Link series
geom_way	geometry(LineString)	Path shape

2) Bus stroke fragment (BSF)

A decomposition of the BS into shorter strokes in consideration of the overlapping of multiple routes is defined as the BSF. Specifically, this is a network in which a path where multiple paths overlap is split at the breakpoints, as shown in Figure 9. In this example, a red route BS1 is split into BSF1 and BSF2 after considering the overlap with the blue route. The BSF generation procedure is shown as follows:

- $R(l_i) = \{r_a, r_b, r_c\}$: Set of routes passing link l_i
- $r.BS = (bs_1, bs_2, \cdots, bs_n)$: BS data list that configures route r
- $r.BSF = (bsf_1, bsf_2, \cdots, bsf_n)$: BSF list that configures route r

Step 1) The $r.BS$ of the desired multiple routes is obtained.

Step 2) A route list $R(l_i)$ that passes through the road links is generated from the bus route data.
Step 3) The BS with multiple routes is divided into overlapping and non-overlapping sections from R, and the overlapping sections are split to generate the BSF.
Step 4) The BSF series $r.BSF$ that passes through each route is obtained and stored in the route BSF table. The BSF table and route BSF table are shown in Tables 7 and 8, respectively, below.

In Step 3, the BS is split based on the obtained R. The BS splitting procedure is shown as follows:

Step 1) $R(l_i)$ and $R(l_{i-1})$ are compared based on the route order. If the included routes are the same, then they are left as they are, and if they are not the same, then l_{i-1} is specified as a splitting position.
Step 2) The BS link series is cut from the first link to the splitting position, thereby splitting the BS.
Step 3) The BSF is generated by splitting the BS and is stored in the BSF table.

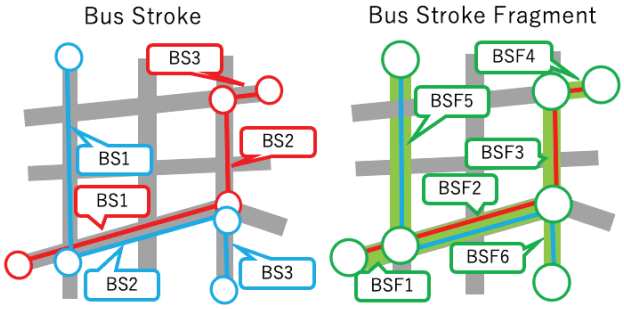

Figure 9. Example of BSF generation.

TABLE VII. BSF TABLE.

Column name	*Data type*	*Explanation*
bsf_id	Integer	BSF ID
stroke_id	Integer	Stroke ID
geom_way	geometry(LineString)	Path shape
link_ids	Varchar	Link series
route_codes	Varchar	Route series

TABLE VIII. ROUTE BSF TABLE.

Column name	*Data type*	*Explanation*
route_code	Integer	System code
line_code	Integer	Route code
dir_code	Integer	Direction code
num	Integer	Order number
bsf_id	Integer	BSF ID
bsf_front	Integer	Front BSF ID
bsf_behind	Varchar	Behind BSF ID
geom_way	geometry(LineString)	Path shape

C. *Route placement/route drawing function*

Details of the route placement function and route drawing function are described below.

1) *Route placement function*

We describe a method of determining the placement order of parallel sections of multiple routes using BSF as the input for each route. The procedure is shown as follows:

Step 1) The BSF series data for two routes among the input routes is obtained.
Step 2) A target BSF list for which the placement order needs to be determined is created.
Step 3) The placement order of the target BSF is obtained in order from the starting point of the route, and if necessary, the previous placement order is used.
Step 4) One route is added to the current result, and the target BSF list for which the placement order needs to be calculated is obtained, as in the case of the two routes.
Step 5) Steps 3 and 4 are repeated for each input route.

In the above procedure, the following two rules are set to determine the placement order.

Rule 1) The placement order of the target BSF is based on the angle formed by the target BSF and the previous BSF.
Rule 2) For routes where the placement order is not uniquely determined, the BSFs are determined backward until the placement order is determined.

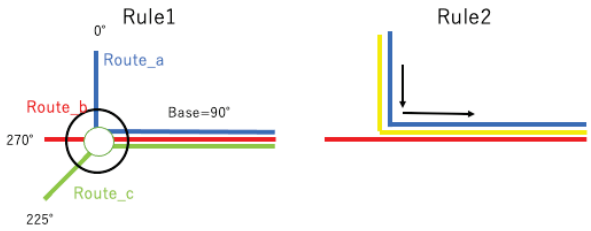

Figure 10. Route placement rules.

Figure 10 shows an example of route placement.

In Rule 1, the order of placement of routes is determined by the angle between the previous BSF and the target BSF at the start point. The PostGIS functions ST_Azimuth and degree were used to calculate the angles: the ST_Azimuth function returns the rightward radians with respect to the north, and the degree function converts radians to degrees.

These are used to calculate the angle between the previous BSF and the target BSF (Base). In the example in Figure 10, (a, b, c, Base) = (0°, 270°, 225°, 90°). The order where these are rotated from the beginning to end until the Base value comes to the beginning, (90°, 0°, 270°, 225°) = (Base, a, b, c), is the placement order arranged in order from the north.

In Rule 2, the angle formed by the BSF connected to the blue and yellow starting points and the target BSF is the same, so the placement order of the front BSF is inherited as is to the placement order of the target BSF.

2) Route drawing function

We describe a method that draws a route map whose route placement order is dynamically changed by generating a GeoJSON file as drawing data based on the route placement order results and reading this file.

The generated GeoJSON file is read and drawn onto an online map using Leaflet. At this time, the BSF is drawn using the Leaflet Polyline Offset [22] plug-in, which can give an offset to a polyline to draw routes by shifting them so that the routes in the same section do not overlap. It also has the function of displaying the bus stop position from bus stop data as a marker and displaying the route name in a popup, as well as the function of drawing in one color without giving the route an offset when the scale is small.

Figure 11 shows an example of a Nagoya city route bus drawn using the route drawing function. It can be seen in this example that the three routes and four bus stops (markers) on the map are displayed correctly.

Figure 11. Route drawing example.

VI. EVALUATION EXPERIMENT

We conducted the following two experiments to verify the effectiveness of the proposed method.

A. Evaluation of stroke-based route estimation function

We evaluate the stroke-based route estimation function, which is one of the proposed methods. The route estimation function takes the coordinates of adjacent bus stops as input and estimates the bus route between those bus stops.

As evaluation methods, we compared three methods: road priority search method using strokes (proposed method), shortest path search (conventional method 1), and shortest path search that considers road classes (conventional method 2). Conventional method 2 weights the cost (distance) according to road class when applying the shortest path algorithm.

The evaluation targets were the 50 bus routes of the Nagoya City Transportation Bureau. The evaluation scale was the matching ratio M of the road links between the estimated route and actual route, and equation (1) is used.

$$M = \frac{Number\ of\ matched\ road\ links}{Number\ of\ actual\ road\ links} \times 100 \quad (1)$$

Table 9 shows the results of the evaluation experiment. The matching ratio M was 92.0% for conventional method 1, whereas the ratios were high at 94.0% and 96.1% for conventional method 2 and the proposed method, respectively. The proposed method was superior to conventional methods 1 and 2 at a significance level of 5%. It was shown that considering the road path, that is, stroke, was effective for bus routes. This was thought to be because the bus tends to run on straight paths as much as possible as turning left or right comes at a high cost.

Figure 12 shows an example of an actual generation. It can be seen that the path generated by the proposed method (Following path) is closer to the actual bus route (Actual line) when compared to conventional method 1 (Shortest Path).

Meanwhile, there were cases where the bus route could not be estimated correctly at some points. In particular, there were many estimation errors near bus terminals. As an example, the green estimated route on the left side was estimated differently on the right side due to an error in estimating the entrance/exit of the bus terminal, as shown in Figure 13. Bus terminals have many bus stops, and the entrances and exits are different, so the road network is also more complicated. Therefore, it is thought that the cause was the generation of a bus stop node on the wrong road link.

TABLE IX. EVALUATION OF ROUTE ESTIMATION FUNCTION.

	Matching ratio M (%)
Conventional method 1	92.0
Conventional method 2	94.0
Proposed method	96.1

Figure 12. Comparison of route generation methods.

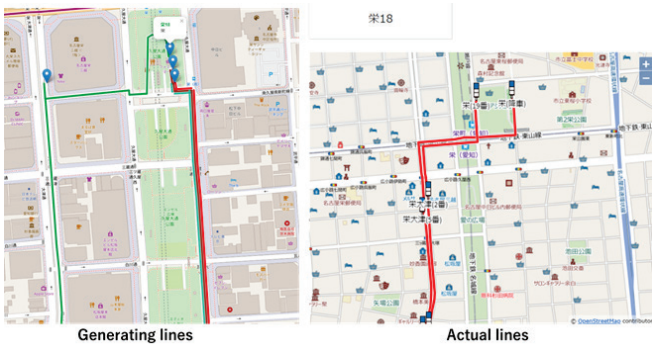

Figure 13. Example of route generation failure.

B. *Verification of link reduction effect by BS/BSF model*

Next, we verified the reduction of the number of links by the BS/BSF model. The BS/BSF model is a method for reducing the number of links by treating bus routes as a set of BSFs rather than as a set of road links. If the number of links can be reduced, then the number of combinations in the route placement problem can be reduced, and the computation time is expected to be reduced.

The evaluation targets were the 661 routes of the Nagoya City route buses. The total number of road links that make up the routes was compared with the number of BSF links that make up the route.

Table 10 shows the experimental results. Each number and reduction rate are shown. The number of road links is 16,433, whereas the number of BSFs is 2,776. As a result, we were able to reduce the number of links by 83.1%.

These results showed that the proposed system can reduce the number of links to minimize the overlap between routes, and that route placement order could be obtained efficiently.

TABLE X. COMPARISON BETWEEN NUMBER OF ROAD LINKS AND NUMBER OF BSFS.

Number of road links	Number of BSFs	Reduction rate (%)
16433	2776	83.1

C. *Qualitative evaluation in automatic generation of 30 routes*

Finally, we conducted a qualitative evaluation on automatically generated routes in this evaluation. The evaluation item here is the visibility of the route map. Figure 14 shows the rendering result of 30 routes. At this scale, 20 routes are displayed on the screen. It was confirmed that the rendering was mostly correct and that there were no problems in actual use.

However, there were several issues. For example, in the route drawing function, routes are distinguished by arbitrary color coding, but visibility is reduced as a result of displaying different routes with similar colors. Additionally, there was the issue of visibility decreasing in drawings of BSFs with three or more routes overlapping or BSFs near bus terminals at positions away from the actual road. Therefore, the realization of a drawing function that maintains visibility near bus terminals and when routes overlap is a topic for future study.

VII. CONCLUSION

In this study, we proposed an automatic estimation method for geographically accurate bus route maps using bus stop coordinate series and strokes. Specifically, the path between adjacent bus stops is estimated using the road

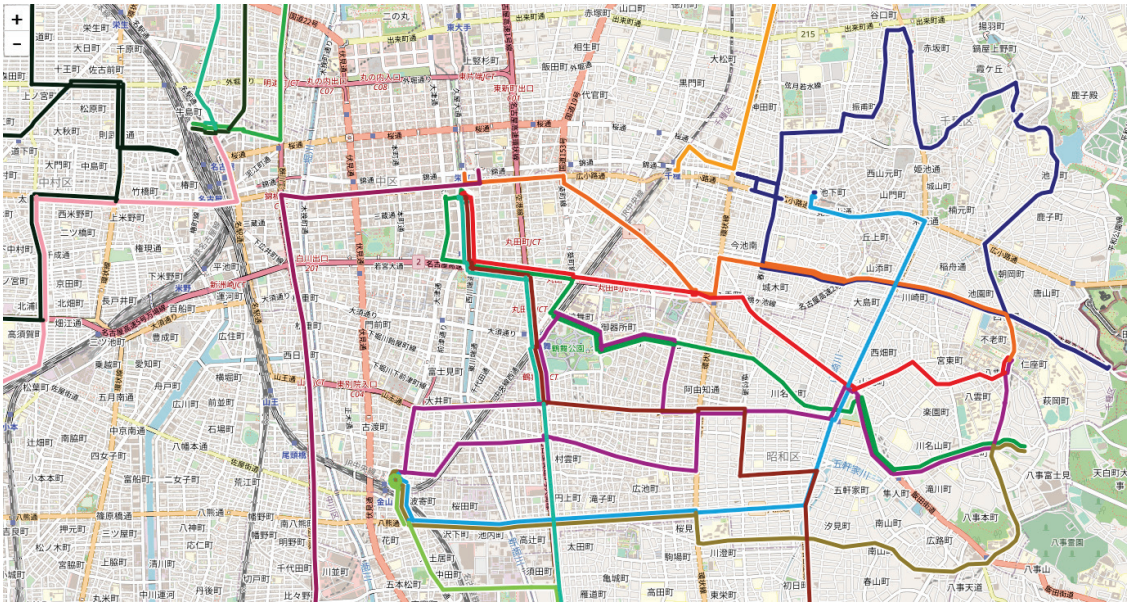

Figure 14. Drawing result of 30 routes

priority search method. The estimated path has the fewest number of right and left turns, so it is thought to be a suitable path for the bus route. As a result, we were able to estimate bus routes with significantly higher accuracy (96.1%) than the conventional method (92.0% and 94.0%).

Additionally, there was the issue where bus routes would intersect each other and become difficult to see when multiple bus routes are drawn on a map. Solving this issue is a type of combinatorial optimization problem, and there is the issue that the computational costs increase exponentially as the number of combinations increases. Therefore, we proposed the BS/BSF model for the purpose of reducing the number of combinations. We minimized the number of links by grouping paths based on the road network to the extent possible. As a result, we were able to reduce the number of links by 83.1% when compared to the conventional road network model.

Furthermore, we confirmed by drawing 30 routes on OpenStreetMap that they could be drawn within a practically acceptable range.

Future issues are as follows. Currently, only 30 routes can be drawn, but we would like to improve this so that it could be applied to more routes. Additionally, we would like to solve the issue of poor visibility at locations where the bus routes intersect in a complicated manner, such as bus terminals. Finally, we would like to investigate the issue of color coding of routes with high visibility.

VIII. ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Numbers JP19H04115 and JP21K19766.

IX. REFERENCES

[1] S. Mizutani, Y. Kim, D. Yamamoto, and N. Takahashi, "Automatic generation method for geographically accurate bus route maps from bus stops," In Proceedings of the GEOProcessing 2022, The Fourteenth International Conference on Advanced Geographic Information Systems, Applications, and Services, Porto, Portugal, ISBN:978-1-61208-983-6, https://www.thinkmind.org/index.php?view=article&articleid=geoprocessing_2022_1_50_30037, pp. 19-24, 2022.

[2] Google Maps, (Comment: Example of popular web maps), https://www.google.com/maps/. Accessed 5 Dec. 2022.

[3] OpenStreetMap, (Comment: Data and model used with this research), https://www.openstreetmap.org/. Accessed 5 Dec. 2022.

[4] Leaflet, (Comment: Map library used with this research), http://www.leafletjs.com/. Accessed 5 Dec. 2022.

[5] General Transit Feed Specification, https://gtfs.org/. Accessed 5 Dec. 2022.

[6] S. H. Hong, D. Merrick, and H. A. Do Nascimento, "The metro map layout problem," In Proceedings of the International Symposium on Graph Drawing 2004, pp. 482-491, Springer, 2004. DOI: /10.1007/978-3-540-31843-9_50.

[7] M. Onda, M. Moriguchi, and K. Imai, "Automatic Drawing for Metro Maps in Tokyo," IEICE-COMP / IPSJ-AL, 2017-AL163, Vol.13, pp. 1-8, 2017.

[8] J. Stott, P. Rodgers, J. C. Martinez-Ovando, and S. G. Walker, "Automatic metro map layout using multicriteria optimization," IEEE Transactions on Visualization and Computer Graphics, Vol. 17, No. 1, pp. 101-114, 2010. DOI: 10.1109/TVCG.2010.24.

[9] M. Fink, H. Haverkort, M. Nollenburg, M. Roberts, J. Schuhmann, and A. Wolff, "Drawing Metro Maps Using Bezier Curves," 20th International Symposium on Graph Drawing, pp. 463-474, 2012.

[10] J. M. Stott and P. Rodgers, "Metro map layout using multicriteria optimization," In Proceedings of the Eighth International Conference on Information Visualization, pp. 355-362, 2004. DOI: 10.1109/IV.2004.1320168

[11] Y. S. Wang and W. Y Peng, "Interactive metro map editing," IEEE Transactions on Visualization and Computer Graphics, Vol. 22, No. 2, pp. 1115-1126, 2016. DOI: 10.1109/TVCG.2015.2430290

[12] P. B. Lloyd, P. Rodgers, and M. J. Roberts, "Metro map colour-coding: Effect on usability in route tracing," In Proceedings of the International conference on theory and application of diagrams, pp. 411-428, Springer, 2018. DOI: 10.1007/978-3-319-91376-6_38

[13] H. Bast, P. Brosi, and S. Storandt, "Efficient Generation of Geographically Accurate Transit Maps," In Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information, pp. 13-22, 2018. DOI: 10.1145/3337790

[14] R. Thomson and R. Brooks, "Efficient generalization and abstraction of network data using perceptual grouping," In Proceedings of the 5th International Conference on GeoComputation, pp. 23–25, 2000.

[15] R. Thomson and D. Richardson, "'Good continuation' principle of perceptual organization applied to the generalization of road networks," In Proceedings of the 19th International Cartographic Conference, pp. 1215–1223, 1999.

[16] Q. Zhang, "Road network generalization based on connection analysis," In Proceedings of the 11th International Symposium on Spatial Data Handling, pp. 343–353, 2005. DOI: 10.1007/3-540-26772-7_26

[17] D. Yamamoto, M. Murase, and N. Takahashi, "On-Demand Generalization of Road Networks based on Facility Search Results," IEICE Transactions on Information and System, Vol. E102-D, No. 1, pp. 99-103, 2019. DOI: 10.1587/transinf.2017EDP7405

[18] M. Murase, D. Yamamoto, and N. Takahashi, "On-demand Generalization of Guide Maps with Road Networks and Category-based Web Search, Results," In Proceedings of the 14th International Symposium on Web and Wireless Geographical Information Systems, Vol. 19, pp. 53-70, 2015. DOI: 10.1007/978-3-319-18251-3_4

[19] D. Yamamoto, S. Ozeki, and N. Takahashi, "Focus+Glue+Context: An Improved Fisheye Approach for Web Map Services," In Proceedings of the ACM SIGSPATIAL GIS 2009, pp. 101-110, 2009. DOI: 10.1145/1653771.1653788

[20] Y. Hiura, (supervisor: D. Yamamoto), "Proposal of an efficient nth min stroke shortest path search method," Master's thesis, Nagoya Institute of Technology, 2020. (In Japanese)

[21] PostGIS, (Comment: Database used with this research), https://postgis.net. Accessed 5 Dec. 2022.

[22] Leaflet Polyline Offset, (Comment: Map library used with this research), https://github.com/bbecquet/Leaflet.PolylineOffset. Accessed 5 Dec. 2022.