オープンデータを利用したバス路線生成システム

服部 真由[1]　山本 大介[1]　高橋 直久[1]

概要：本論文では，オープデータを用いたバス路線の自動経路生成に関して述べる．バスの路線を位相関係に基づいた概略図ではなく実地図上に描画する場合，バス路線経路と道路データとの対応付けが必要となるが，路線経路は路線数や経路変更の観点からデータ整備には多くの労力を要する．そこで我々は交通事業者が公開するオープンデータに着目し，停留所の座標と路線の停留所系列のみから路線経路を自動生成するシステムを提案し，その実現法について述べる．提案システムでは停留所に対する道路ネットワーク上のノードとしてバス停ノードを生成したのち，バス停ノード間の経路探索を行い，データベース化した．プロトタイプシステムを用いた評価実験を行った結果，交差点などの近傍ノードに比べバス停ノードを生成することで距離を約 78%縮められ，有用性を示すことができた．また，5 路線に対する生成経路と実経路を比較検証した結果，最短経路探索を用いた経路では一致率が 85%以上であり，道なりを考慮に入れた経路探索を行うことで最短経路による余分な右左折を減らし，より実経路に近い経路生成を行えることが分かった．

Bus Route Generation System Using Open Data

MAYU HATTORI[1]　DAISUKE YAMAMOTO[1]　NAOHISA TAKAHASHI[1]

1. はじめに

一般的に鉄道やバスの路線図は地理的情報ではなく，位相関係に基づいた概略図で作成されていることが多い．このような路線図は人の手によって描かれていると考えられ，図 1 のように複数の路線間でなるべく交差が少なくなるよう路線が配置されている．これまでに鉄道路線における路線図の自動生成 [1] [2] やバス路線における重複を避けた自動描画手法 [3] [4] など，路線図の自動生成手法に関する研究が多く行われてきた．一方で，実地図上におけるバス路線図の自動描画に関する研究は少ない．これは，概略図とは違い，実際の道路上に沿ってバス路線を描画するためには，バス路線の経路と実際の道路データ (位置情報) との対応付けが必要となり，より問題設定が複雑となるためであると考えられる．また，バス路線の経路データは路線数が多いことや道路整備等における経路変更の可能性があるため編集の手間が大きい．

このような課題に対して，我々は交通事業者が公開するオープンデータに着目した．オープンデータとは機関や企業が誰でも入手し，利用・再配布できるよう公開しているデータのことである．近年では，国土交通省 [5] が GTFS 形式に基づいて定めた「標準的なバス情報フォーマット」の取り組みが進んでいるが，このフォーマットではデータの提出条件として停留所の緯度経度座標は必須項目 (stops) である事に対し，路線経路の描画情報の項目 (shapes) は任意項目であるため路線の通過経路データは公開されていない場合がある．そこで，我々は停留所の情報のみからバス路線の経路を推定するシステムを提案する．

図 1　名古屋市営バス路線図

[1] 名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology

本研究の目的は，名古屋市交通局のオープンデータを用いて停留所座標と停留所系列からバス路線経路を生成し，プロトタイプシステムを用いて実経路との差異を比較検証することである．今回使用するオープンデータは以下の 3 つである．

- 停留所データ：停留所の位置情報を示すデータ．
- 系統データ：系統の路線に対する起点と終点を示すデータ．
- 停留所並順データ：路線に対する起点から終点までに通過する停留所を示すデータ．

提案システムでは，入力された停留所系列に従い各停留所間の経路探索を行って求めた経路データを統合したものをバス路線経路として扱うものとする．ここで，提案システム実現にあたり以下の課題が考えられる．

課題 1　使用する停留所の位置座標は実際の道路上の点ではないため，出発点，終着点となる停留所に対する道路ネットワーク上の点を求める必要がある．

課題 2　停留所が集約されているバスターミナルなどでは，全ての乗り場において正しい出入り口からバスが通過するよう経路生成を行う必要がある．

課題 3　複数路線において停留所系列が重複することがあり，再度同じ区間の経路探索を行うのは余分な計算量がかかる．

課題 1 に関して，道路ネットワークを構成するグラフは図 2 の様に交差点や曲がり角を表す「ノード」とノード間を結ぶ「リンク」で表現される．この時，2 点間の経路探索において始点終点となるノードを停留所からの最近傍ノードとした場合，実際の停留所との位置の誤差が大きくなる可能性がある．理由として，交差点間の距離が長いような道路沿いに設置される停留所の場合，設置場所が交差点付近か道路の中央付近かでノードとの位置の誤差は大きく変わる．また課題 2 において，駅前のロータリーやバスターミナルなどの広い敷地内には複数の乗り場が集約されている．このような施設内に設置される乗り場からの経路探索を行う場合，バスは必ずターミナルの出入り口を通過して公道に出るものとする．この場合，乗り場の位置によっては正しい出入り口ではないノードから出発する可能性が考えられる．課題 3 に関しては，提案システムは停留所間ごとに経路探索を行うため，始点終点となる停留所が同一であるならば路線が異なる場合でも同じ経路を通過すると仮定する．これらの課題に対して提案システムでは次のアプローチをとる．

アプローチ 1　停留所に対する道路上の最も近い点としてバス停ノードを生成する．

アプローチ 2　バスターミナルにおける乗り場に対するノードを 1 点に集約する．

アプローチ 3　生成した経路をデータベース上で管理し，区間ごとの経路データと全体の経路データを分けて生成する．

アプローチ 1 において，図 3 の様に道路ネットワーク上に既存のノードよりも，より近い位置に新規ノード (バス停ノード) を追加することで，経路探索時の始点終点の位置の誤差を減少させる．また，バス停ノード生成に伴い，対象となる道路データを編集する必要がある．そこで，生成したバス停ノード及び編集された道路データをデータベース化し，経路探索時に道路ネットワーク上の対象となるリンクのデータを書き換える手法を提案する．アプローチ 2 では，複数の乗り場に対するバス停ノードの集約化を行うことでターミナル内の乗り場における出発位置の修正を図る．例として図 4 ではターミナル内の乗り場 6 つに対してそれぞれバス停ノード BN1 から BN6 がある．これらの乗り場に対する経路を正しい出入り口から出発させるため，B1 から 4 及び BN6 に対する停留所のノードは全て BN5 のデータに書き換えることで集約化するものとする．また，アプローチ 3 では，停留所間ごとの経路データと路線全体の経路データを分けて管理する．これにより，別路線における停留所区間で既に経路データが存在する場合は，そのまま経路データを適応することが可能となる．

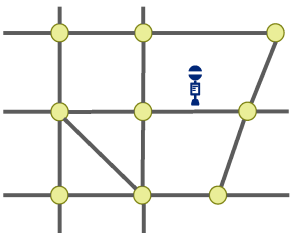

図 2　道路ネットワークグラフ

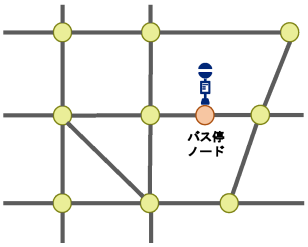

図 3　バス停ノード

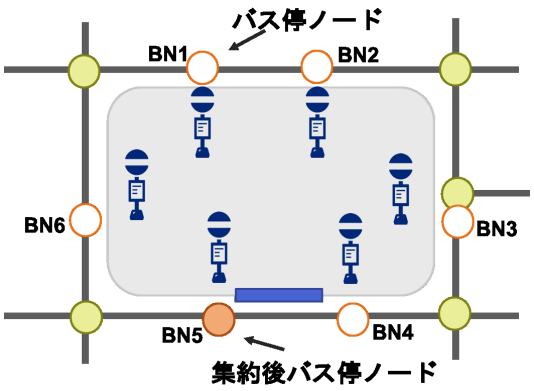

図 4　バス停ノード集約

2. 提案システムの概要

この章では，使用する道路データ及びバスデータについて説明し，提案システムを構成する機能の特徴について述べる．

2.1 使用データ

提案手法で使用するデータは，道路データと名古屋市交通局のオープンデータに基づいたバスデータである．

2.1.1 道路データ

道路ネットワークは図 5 のようにポイント，ノード，リンクによって構成されている．ポイントは地図上の緯度経度を有する点であり，ノードは道路ネットワーク上の交差点を指す．リンクはポイント間を結んだ線分であり，ノードとポイントによって形状 (アーク) を構成する．使用する道路テーブルを表 1 に示す．リンクの形状はジオメトリ型 (LineString) で表されている．

表 1 道路テーブル

カラム名	型	説明
id	Integer	リンク ID
source	Integer	始点ノード側の ID
target	Integer	終点ノード側の ID
x1	double	始点ノード側の経度
y1	double	始点ノード側の緯度
x2	double	終点ノード側の経度
y2	double	終点ノード側の緯度
km	double	リンクの長さ (単位：km)
geom_way	LineString	リンクの形状 (アーク)

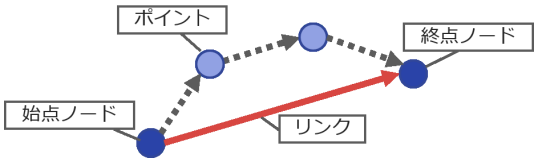

図 5 道路ネットワーク

2.1.2 バスデータ

バスデータは名古屋市交通局が公開する平成 29 年度時点における 3 つのバスデータを使用する．また，以下の※はオープンデータ以外に追加したカラムであり，停留所を一意に識別する番号となっている．

停留所データ

停留所テーブルを以下の表 2 に示す．停留所データには追加した停留所番号 (id) と停留所及び乗り場別のコードが割り振られており，停留所の位置座標が格納されている．id もしくは停留所コード+乗り場コードを指定することで各乗り場のレコードを一意に識別することができる．

表 2 停留所テーブル

カラム名	型	説明
id ※	Integer	停留所番号
busstop_name	varchar	停留所名称
busstop_code	Integer	停留所コード
noriba_code	varchar	乗降場所コード
busstop_noriba_code	varchar	停留所コード+乗降場所コード
lat	double	停留所の経度
lng	double	停留所の緯度

系統データ

系統テーブルを表 3 に示す．系統データには各系統・路線・方向に対しコードが割り振られており，この 3 つのコードを指定することで系統情報が一意に識別できる．

表 3 系統テーブル

カラム名	型	説明
route_code	Integer	系統コード
line_code	Integer	路線コード
dir_code	Integer	方向コード
route_name	varchar	系統記号
start_point	varchar	起点の停留所名
end_point	varchar	終点の停留所名

停留所順序データ

停留所順序テーブルを表 4 に示す．停留所順序データには系統の路線に対する起点から終点までに経由する停留所情報が格納されている．4 章で述べるプロトタイプシステムではこのデータから停留所系列を取得する．

表 4 停留所順序テーブル

カラム名	型	説明
route_code	Integer	系統コード
line_code	Integer	路線コード
dir_code	Integer	方向コード
route_order	Integer	順序番号
busstop_noriba_code	varchar	停留所コード+乗降場所コード
id ※	Integer	停留所番号

2.2 提案システムの特徴

提案システムは以下の機能を持つ．

バス停ノード生成機能

停留所に対する最も近い道路上のポイントと交差点を頂点とし，頂点を結ぶ道路を辺としたグラフ G(バス・道路ネットワーク) を作成する．これにより，バスの運行経路と道路ネットワークの対応付けが可能となり，実際の停留所位置とグラフ G の頂点の位置との誤差を減少する．

バス停ノード集約機能

施設内などの対象となる乗り場のバス停ノードを一つのバス停ノードデータに書き換える．これにより，経路探索時に対象の乗り場において，実際のバス停ノードではなく集約後のバス停ノードデータを取得することで位置の誤差による経路の誤りを防ぐ．

バス路線生成機能

任意の路線に対する停留所系列をバスデータベースから取得し，作成したグラフ G において隣接する停留所頂点間の経路の推定及び経路データの統合によりバス路線経路を生成する．この際に各停留所間の経路データと統合後の経路データを分けて管理することで，別路線に対する経路データへの利用及び停留所系列の変更に対する路線編集の簡単化を図る．

また，経路探索法として道なり経路を考える． バスのような大型車両は経路としてなるべく右左折が少なくなるような道なりとなる経路を通過すると考えられる．そこでグラフ G においてストロークデータベースを用いた道なり優先探索を提案する．ストロークとは認知心理学に基づきグループ化された道路ネットワークの集合であり，リンク列を表す．例として，図 6 は道路リンクをストロークで表現した様子である．文献 [7] では，ストロークネットワークを構築した上でストローク数が最小となる最短経路探索法 (SPMS 法) を実現している．

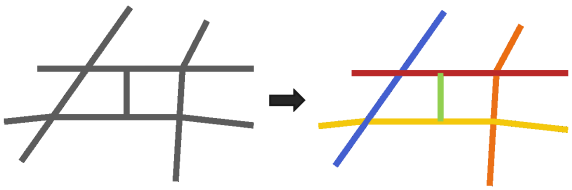

図 6 ストローク例

2.3 提案システムの構成

提案システムの構成図を図 7 に示す．バス停ノード生成機能では停留所座標と道路データを入力とし，バス停ノード及び分割リンクデータをバスデータベースに格納する．バス停ノード集約機能では対象となる停留所のすべてのノード ID を集約するノード ID に更新する．バス路線生成機能では取得した停留所系列を入力として経路探索を行い，生成した経路をデータベースに格納している．

また，新たに構築したバスデータベース内のテーブルを表 5 に示す．表に対して一段目の三つのテーブルがオープンデータに基づいた市バスデータ，二段目のつがバス停ノード生成機能，三段めの二つがバス路線生成機能において生成されたデータを示している．これらのデータ構造については次章で述べる．

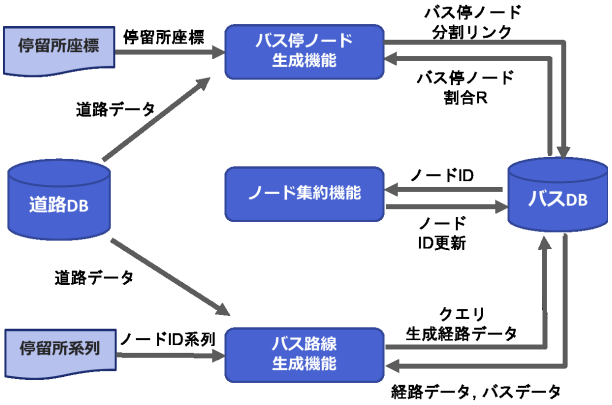

図 7 提案システム構成図

表 5 バスデータベース

テーブル名	説明
停留所テーブル	停留所データ
系統テーブル	系統データ
停留所順序テーブル	系統に対する停留所順序データ
バス停ノードテーブル	停留所に対する最近点データ
分割リンクテーブル	リンク分割後の新規リンクデータ
id 対応テーブル	新規リンク ID と停留所番号の対応表
停留所間経路テーブル	停留所間の経路データ
バス路線経路テーブル	路線の全体の経路データ

3. 提案システムの実現法

提案システムは 2.2 節で述べたように三つの機能で構成されている．はじめにバス停ノード生成機能で全ての停留所に対するバス停ノードの生成を行なってから，バス停ノード集約機能において集約されるノードデータを更新し，これらのデータを用いることでバス路線生成機能を実装する．この章では，それぞれの機能に対する実現法を述べている．

3.1 バス停ノード生成機能

3.1.1 バス停ノードの算出

以下に示す手順でバス停ノードを生成する．

手順 1 停留所座標 p と道路 DB から近傍リンク集合 L を取得する．

手順 2 $L = L_1, L_2, ..., L_i$ から最近傍リンク L' を求める．

手順 3 点 p から最近傍リンク上の最近点を取得する．

この 3 つの手順を実現するために，PostGIS [11] の関数を使用する．まず，手順 1 において，ST_DWithin によって道路 DB から停留所付近のリンク集合 L を取得後，手順 2 で ST_Distance により停留所と各リンクの距離を算出し，最も距離が短い最近傍リンク L' を求める．手順 3 では, 関数 ST_LineLocatePoint と ST_LineInterpolatePoint を用い

る．各関数の使用例を図 8, 9 に示す．ST_LineLocatePoint では，与えたラインストリングとポイントから最近点が位置するラインストリング上の割合 $R(0 \leq R \leq 1)$ を求めることができ，図 8 における最近点はラインストリング上の始点 (左側) から 60%の位置に存在することが分かる．R 取得後，ST_LineInterpolatePoint によってラインストリング上の座標値を算出する．求められたバス停ノードデータは表 6 に示す形式で格納される．

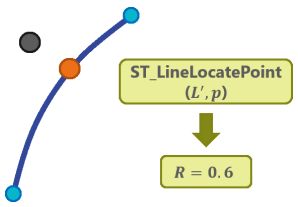

図 8 ST_LineLocatePoint

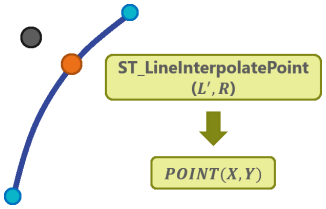

図 9 ST_LineInterpolatePoint

表 6 バス停ノードテーブル

カラム名	型	説明
id	Integer	停留所 ID
link_id	Integer	最近傍リンク ID
node_lat	double	バス停ノードの経度
node_lng	double	バス停ノードの経度
ratio	double	割合

3.1.2 リンクデータの分割

関数 ST_LineSubstring を用いて対象となるリンクデータをバス停ノード部分で分割する．この関数はラインストリング上の指定した開始 (及び終了) 点の割合 R 間の部分ラインストリングを返す．関数の使用例を図 10 に示す．この時，$R = 0.6$ となるポイントでは始点 $(R_s = 0)$ から R までと，R から終点 $(R_g = 1)$ までの 2 回行うことでそれぞれのラインストリングが求まり，結果リンクを分割することが出来る．また，各停留所はそれぞれ独立にバス停ノードが生成されているため，バス停ノードにおける最近傍リンクが重複することがある．このような場合は，各停留所のバス停ノードの割合の小さいものから順に分割するものとする．

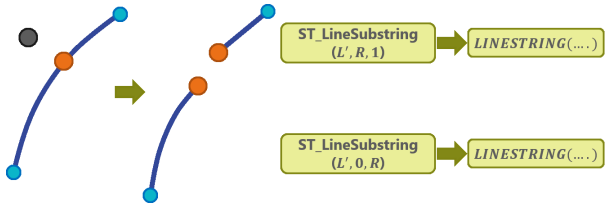

図 10 ST_LineSubstring($R = 0.6$)

図 11 のように，分割されたリンク及び構成される始点と終点のノード ID を新たに割り振り，表 7 の形式でデータベースに格納する．なお，道路テーブルのリンク ID，ノード ID との重複を避けるため，新規の ID は共に 100000000 から割り振られている．また，表 7 は停留所データとは独立であるため，分割リンクと停留所データの関係を示すテーブルを表 8 の形式で示す．カラム id_source はバス停ノードがリンクの始点側，終点側のどちらに位置するかを示しており，始点側の時に true とする．例として，図 11 では linkID=100000000 が false，linkID=100000001 が true となる．

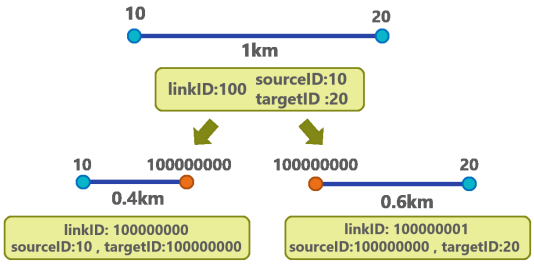

図 11 リンク分割

表 7 分割リンクテーブル

カラム名	型	説明
link_id	Integer	リンク ID
seg_link_id	Integer	分割後のリンク ID
seg_source	Integer	分割後の始点ノード ID
seg_target	Integer	分割後の終点ノード ID
seg_km	double	分割後のリンク長さ (単位：km)
seg_geom	LineString	分割後のリンク形状

表 8 id 対応テーブル

カラム名	型	説明
id	Integer	停留所番号
link_id	Integer	最近傍リンク ID
seg_link_id	Integer	分割後のリンク ID
seg_source	Integer	分割後の始点ノード ID
seg_target	Integer	分割後の終点ノード ID
id_source	boolean	バス停ノードが始点側なら true

3.2 バス停ノード集約機能

バス停ノード生成機能では，すべての停留所に対して最近傍リンク上にノードを生成した．しかし，1 章で述べたように複数の乗り場が集約されているバスターミナルなどでは，乗り場の位置によって最近傍リンクが異なり，バス停ノードの位置にもばらつきが生じる．この際に，ターミナル内の構内道路データが存在する場合は停留所に対するバス停ノードは構内道路上に生成されるため，構内道路と公道を繋ぐリンク (ターミナルの出入り口) を通過すると考えられる．しかし，そのような道路データが取得できない場合，上記の様な問題から停留所が必ずしも出入り口付

近を通過するとは限らないため対象となるバス停ノードを出入り口に最も近い停留所のバス停ノードに集約化する．手法としては，id 対応テーブル内の対象となる乗り場のレコードを，統一する乗り場の情報に書き換えており，これにより 3.3 節で述べるバス路線生成機能において，系列取得時に実際のバス停ノードではなく集約後のノード ID を取得する．例として，名古屋駅のバスターミナルにおいて集約化を行なった結果を図 12，13 に示す．名古屋駅では 1 番から 11 番までの乗り場 (青点) のバス停ノードデータを降車 (id:2413) のバス停ノード (緑点) に書き換えている．

図 12 名古屋駅 (集約前)

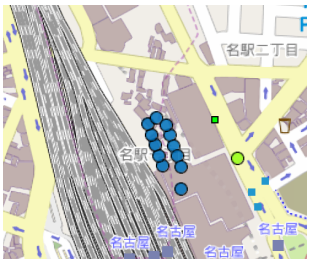

図 13 名古屋駅 (集約後)

3.3 バス路線生成機能

以下のように記号を定義し，経路データ生成の手順を示す．

- $ID = id_1, id_2, ...id_i$：停留所番号リスト
- $N = N_1, N_2, ..., N_i$：ノード ID リスト
- $V \ni (s, g)$：始点 s と終点 g の組
- $R = r_1, r_2, .., r_j, .., r_{i-1}$：経路データ (LineString)

手順 1 停留所系列データを取得する．

1-1 停留所順序データから指定した系統・路線・方向コードに対する停留所番号リスト ID を取得する．

1-2 id 対応テーブルから各停留所番号に対するノード ID リスト N を取得する．

手順 2 $v = (id_i, id_{i+1}) \in V$ における経路生成を行う．

2-1 停留所間経路テーブルで (s_id,g_id)=(id_i, id_{i+1}) または (s_id,g_id)=(id_{i+1}, id_1) となるレコードが存在するか検索する．

2-2 レコードが存在する場合，取得したレコードの経路データを r_j とする．

2-3 レコードが存在しない場合，v における経路探索を行い r_j を求める．

2-4 求めた経路データを停留所間経路テーブルに格納する．

手順 3 経路データの統合を行う．

3-1 指定した系統・路線・方向コードに対する経路データ集合 R を取得する．

3-2 集合 R のデータを MultilLineString 型にしてバス路線経路テーブルに格納する．

バス路線経路はデータベース上で部分的な停留所間の経路と，全体の路線経路に分けて管理する．理由として，複数路線において通過する停留所が重複する可能性があるためである．例として，図 1 でも示されている栄 18 号と昭和巡回は名大病院から曙町二丁目までは通過する停留所が同一である．経路探索は停留所間ごとに行うため，始点終点となる停留所が同一であるならば路線が異なる場合でも同じ経路を通過すると考えることができる．したがって，手順 2-2 のように各停留所間の経路データとしてデータベース化しておくことでその都度同じ区間の経路探索を行う必要がなくなり計算量の削減に繋がる．手順 2-3 では経路探索前に取得したリンクデータのリストのうち，分割リンクテーブル内の link_id と一致するリンクはリストから削除し，代わりに対応する分割リンクデータを追加する．

表 9 停留所間経路テーブル

カラム名	型	説明
route_code	Integer	系統コード
line_code	Integer	路線コード
dir_code	Integer	方向コード
s_order	Integer	始点側の順序番号
s_id	Integer	始点側の停留所番号
g_order	Integer	終点側の順序番号
g_id	Integer	終点側の停留所番号
geom_way	LineString	始点から終点までの経路

表 10 バス路線経路テーブル

カラム名	型	説明
route_code	Integer	系統コード
line_code	Integer	路線コード
dir_code	Integer	方向コード
route_name	varchar	系統記号
s_gid	Integer	起点側の停留所番号
g_gid	Integer	終点側の停留所番号
geom_way	MultiLineString	起点から終点までの経路

4. プロトタイプシステム

提案システムを実現するため，Java [8] 言語を用いたプロトタイプシステムを実装した．地図データは OpenStreetMap [9] を用いたサーバから，バスデータは構築したバスデータベースから取得するものとし，データベースにおいて PostgreSQL [10] と PostGIS [11] を用いる．今回，プロトタイプシステムでは一般的な経路探索法として挙げられるダイクストラ法 [12] による最短経路探索を用いた実装となっている．また，名古屋駅 (図 12,13) と栄の二箇所のバスターミナルにおいてバス停ノードの集約化を行なった．栄における集約後のバス停ノードを図 14,15 に示す．栄では乗り場 2 番から 7 番 (青点) のバス停ノードデータを 1 番乗り場 (id:1100) のバス停ノード (緑点) に書き換えている．

図 14 栄 (集約前)

図 15 栄集約後)

4.1 プロトタイプシステムの動作

システムの実行画面を図 16 に示す．画面上部に入力パネル，下部に地図画像が配置されており，起動時は中心座標 (35.15642247131321,136.92293150601253)，スケール値 14(縮尺 1/35,000) とした地図画像を表示している．入力パネルにおいて道路データ及びバスデータを取得後系統・路線・方向のコードを指定することで任意の停留所系列をバスデータベースから取得し経路探索を行う．また，バスデータ取得時にリンク対応テーブルから地図画像範囲内に置けるリンクデータの書き換えを行うものとする．経路探索後，各経路データを統合し，バス路線経路テーブルに格納する．格納されたデータは「バス路線表示」ボタンをクリックすることで地図上に赤線で表示される．プロトタイプシステムにおいて生成されたバス路線の表示結果を図 17 に示す．

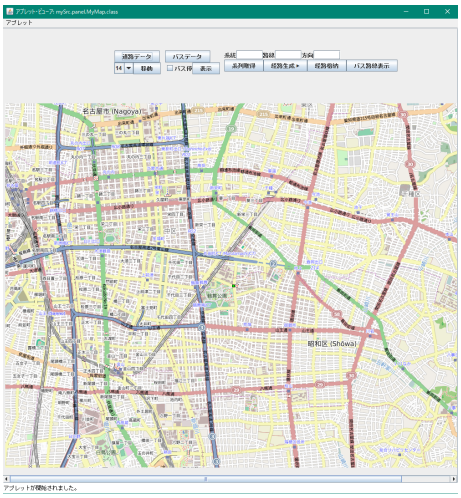

図 16 プロトタイプシステム起動画面

図 17 昭和巡回

5. 評価実験

5.1 バス停ノードの評価

5.1.1 評価方法

停留所に対するバス停ノード及び道路テーブルを用いて取得した最近傍ノードを評価対象とし，停留所座標から各ノードまでの距離を測定する．また，求めた距離について以下の記号を定義し式 1 を用いて最近傍ノードに対するバス停ノードの距離の減少率 R を計算する．

- $dist\_nns$：停留所と最近傍ノードの距離
- $dist\_busnode$：停留所とバス停ノードの距離

$$R = \frac{dist\_nns - dist\_busnode}{dist\_nns} \times 100 \tag{1}$$

5.1.2 結果と考察

各ノードと停留所間の距離のグラフを図 18 に示す．また，平均減少率 $\overline{R}$ を求めた結果，$\overline{R} = 77.758\%$ となり，最近傍ノードに比べてバス停ノードを生成することで停留所との位置の誤差を約 77%縮めることができた．この結果から，バス停ノードの有用性が示せた．

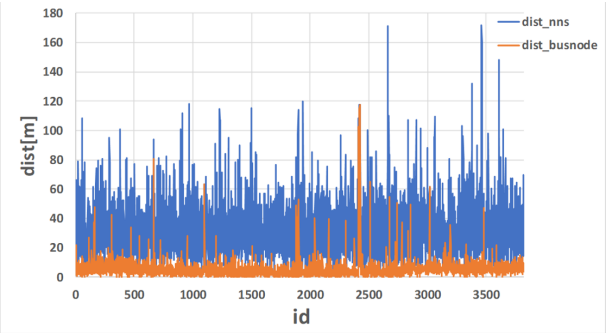

図 18 停留所とノード間の距離

5.2 生成経路の評価

最短経路探索において生成したバス路線経路と，名古屋市交通局が公開する路線の地図画像を比較し，実経路に対する生成経路の一致度を計算する．また，ストロークを考慮に入れたバス路線経路を用意し，最短経路探索にとより生成した経路との差異を比較する．ただし，ストロークデータを用いた経路探索は未実装のため，道なりを考慮に入れた手書きのバス路線経路となっている．

5.2.1 評価対象

下記に示す名古屋市営バスの 5 つの路線に対する生成経路，実経路，ストロークを用いた経路を評価対象とする．

- 基幹 1：栄 - 星崎
- 栄 18：栄 - 妙見町
- 幹名駅 1：名古屋駅 - 上飯田
- 幹名駅 2：名古屋駅 - 野跡駅
- 昭和巡回：御器所通 - 名古屋大学

5.2.2 評価方法

実経路のリンク本数 ($route_T$) のうち，生成経路で通過しているリンクと同一であるリンクの本数 ($route_M$) を求める．ここで以下のように定めた道路を 1 本とみなし，これらの本数から実経路に対する生成経路の一致度 $M = route_M/route_T$ を算出する．例として，図 19 に示す停留所 A から B までの経路において (赤:実経路，青:生成経路)，リンクの定義に従うと実経路は 6 本であることが分かる．このうち生成経路で同一のリンクを通過するのは 3 本となるため，一致度 M は $3/6 \times 100 = 50\%$ となる．

リンクの定義

- 交差点から交差点までの道路
- 交差点から曲がり角，行き止まりまでの道路
- 停留所から交差点までの道路

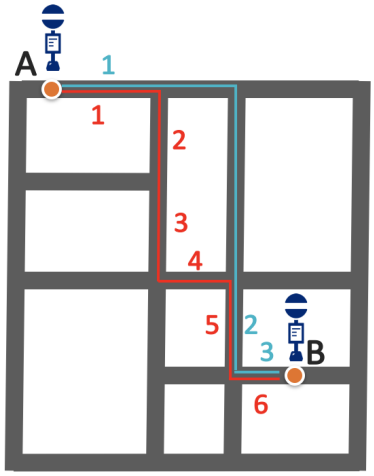

図 19 経路内のリンク本数

5.2.3 結果と考察

各路線に対するリンクの本数及び実経路に対する生成経路・ストローク経路の一致度の結果を表 11 に示す．まず生成経路において，5 つの路線全てにおいて一致度 M が 85%以上と実経路に非常に近い路線を生成できている．実経路と比較すると，どの路線においても停留所間が直進となるような経路は必ず最短経路となるため実経路と一致しており，右左折時における交差点の位置の違いにより経路に差異が生じていることが分かった．また，ストローク経路に置ける一致度 M' を算出した結果，全ての路線に対する M と比較して，ストロークを用いた経路の方が実経路との一致度が高いという結果が得られた．生成経路と比較すると，ストローク経路においても停留所間が直進であるような場合は，道なりであり，かつ最短であることから実経路，生成経路共に一致していた．また，一致度 M から M' において数値が上昇した理由として，生成経路における右左折時の経路の差異がストローク経路を用いることで縮まったためであることが分かった．従って，バス路線の経路生成においてストロークを用いるという概念は非常に重要であると考えられる．

表 11 路線のリンク本数及び一致度

系統名	リンク本数			M(%)	M'(%)
	実経路	生成経路	ストローク		
基幹 1	118	116	118	98.305	100
栄 18	110	94	107	85.455	97.273
幹名駅 1	114	111	114	97.368	100
幹名駅 2	90	87	89	96.667	98.889
昭和巡回	204	185	200	90.686	98.039

6. おわりに

本稿では従来の人の手により作成されていた路線経路に対し，名古屋市交通局が公開するオープンデータに着目しバスデータと道路データを対応づけることで，実地図上において道路上に沿ったバス路線の経路データを停留所座標と系列のみから自動生成するシステムを提案し，実現法について述べた．また，提案システムに基づいたプロトタイプシステムを実装し，比較評価実験を行なった．実験結果より，バス停ノードを生成することで交差点などの近傍ノード時に比べて距離を約 78%縮めれたことからバス停ノードの有用性を示すことができた．また，最短経路探索における生成経路と実経路を比較した結果，一致率は 85%以上となり，ストロークを考慮に入れた経路との比較において最短経路による余分な右左折を減らした経路生成を行えることが分かった．

今後の課題として，ストロークデータを用いた道なり優先による経路探索を実装し評価実験を行うことが挙げられる．また，路線の停留所系列を取得する際に系列のコード指定ではなく，系統や路線名などの入力から取得する様にするなどの UI の改善が挙げられる．

7. 謝辞

本研究は JSPS 科研費 25700009，および，総務省 SCOPE の助成を受けたものです．この場を借りて，感謝の意を表します．

参考文献

[1] Martin Nollenburg, Alexander Wolff, Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming, IEEE Transactions on Visualization and Computer Graphics, Vol.17, No. 5, pp. 626-641, 2011.
[2] 恩田雅大, 森口昌樹, 今井桂子, 東京の路線網に対する鉄道路線図生成手法, 信学技報, Vol.117, No.28, COMP2017-10, pp.69-76, May.2017.
[3] 篠原卓, バス路線図に対する自動描画手法, 中央大学大学院研究年報, 理工学研究科篇, 第 46 号, 2016.
[4] 加藤りか, 道路ストロークを用いたバス路線マップの描画手法, 名古屋工業大学大学院工学研究科修士論文, 2018.
[5] 国土交通省, 入手先 <http://www.mlit.go.jp/index.

html>(2018.4.30 参照)
[6] 名古屋市交通局，入手先 <https://www.kotsu.city.nagoya.jp/jp/pc/>(2018.4.30 参照)
[7] 新帯里奈, 山本大介, 高橋直久, ストロークネットワークを用いた道なり優先経路探索システムの実現，マルチメディア, 分散協調とモバイルシンポジウム論文集, pp.396-403, Jul.2016.
[8] Java，入手先 <https://www.java.com/ja/>(2018.4.30 参照)
[9] OpenStreetMap, 入手先 <http://www.openstreetmap.org/>(2018.4.30 参照)
[10] PostgreSQL，入手先 <https://www.postgresql.org/>(2018.4.30 参照)
[11] PostGIS，入手先 <http://postgis.refractions.net/>(2018.4.30 参照)
[12] Dijkstra，入手先 <http://www.deqnotes.net/acmicpc/dijkstra/>(2018.4.30 参照)