「マルチメディア，分散，協調とモバイル
(DICOMO2019)シンポジウム」 令和元年7月

交差ストロークを用いた道なり優先経路探索の高速化手法

日浦 勇貴[1]　山本 大介[1]　高橋 直久[1]

概要：近年，Google Maps や Yahoo! 地図などの web マップサービスが普及しており，これらやカーナビなどの経路探索システムは距離優先や道幅優先といった様々な条件で経路探索できる．また，我々はストロークという道なりに続く道路を用いた研究を行っている．先行研究ではストロークを用いた道なり優先経路探索を行ったが，道路リンク単位の経路探索を行うため，計算に時間がかかってしまうという問題点がある．そこで，本研究では上記の問題を解決するためにストローク単位の経路探索を行うことで道なり優先経路探索を高速化する手法を提案する．また，探索の高速化のために，交差ストロークと呼ばれるストロークに交差するストロークを事前に求め，データベース化することによりオンデマンドで行う探索時間を削減した．本論文では交差ストロークを用いた道なり優先経路探索システムを提案，実装した．そして，提案システムと従来システムを用いて評価実験を行った結果，提案システムは従来手法に比べ直線距離が 1km の場合約 140 分の 1，直線距離が 10km の場合約 40 分の 1 の探索時間で探索できることが分かった．また，名古屋市内の全てのストロークから，交差ストロークテーブルを作成した．この交差ストロークテーブルを利用し，名古屋市内ならば右左折が 16 回以内で任意の場所まで行くことができるということが分かった．

A Fast Routing Method Preferred Following Paths Using Cross Strokes

YUKI HIURA[1]　DAISUKE YAMAMOTO[1]　NAOHISA TAKAHASHI[1]

1. はじめに

近年，Google Maps[1] や Yahoo!地図 [2] などの web マップサービスが普及しており，これらはルート検索や所要時間を容易に調べることができる．またカーナビなどの経路探索システムでは，一般道優先や道幅優先といった様々な条件で経路探索できるようになっている．しかしながら「道なり」を優先的に案内することがあまりなく，ある目的地までを最短経路を探索することが主である．最短経路は経路長が短くなり，目的地まで早く到着しやすいが，右左折が多くなり複雑な経路になる場合がある．道なり優先経路は右左折が少ないので道を間違えにくく，また道案内が容易であるという利点がある．

また，我々は道路ネットワークとストロークについての研究を行っている．道路ネットワークは，道路の交差点を表す「交差点ノード」と隣接する交差点ノード同士を接続した「道路リンク」で構成されている．ストロークとは，認知心理学に基づいて道路ネットワークを連続性が認められる場合にグルーピングしたもので，いわゆる道なりの道路である．図 2 は，図 1 において交差点ノードと道路リンクで構成された道路ネットワークに対して知覚群化の考えを適用してストロークを構成したときの道路ネットワークである．例えばストローク S_1 は，道路リンク L_1，L_2，L_3 をグルーピングしている．先行研究では，ストロークを用いた道なり優先経路探索を行ったが，道路リンク単位の経路探索であり，交差点ノードを接続する道路リンクの数だけ分割し，分割した各子ノードに対して経路探索を行うため，計算に時間がかかってしまうという問題点があった [8]．

そこで，本研究ではストローク単位の経路探索を行うことで道なり優先経路探索を高速化する手法を提案する．ここで言う道なり優先経路は，右左折の最も少ない経路の中で経路長が最も短い経路のことを指す．ストロークは右左折のない道なりに続く道路であるから，ストローク数が最少の経路は右左折の回数が最も少ない経路であるといえる．つまり最少ストローク数最短経路を求めることで，厳

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

密な道なり優先経路を求めることができる．本論文では以下の特徴を有したシステムを提案する．

- ストロークに交差するストロークを次に探索するストロークの候補として，ストローク単位の幅優先探索を行う．
- あるストロークに交差するストロークを「交差ストローク」と定義し，交差ストロークを事前にデータベース化することでオンデマンドの計算量を削減する．
- ストローク単位の幅優先探索をすると，探索対象のストロークが指数関数的に増加してデータ量が膨大になるため，出発点と到着点の双方向からの探索を行うことで計算量を削減する．

また，後述する先行研究の道なり経路探索システムとの計算時間の比較を行う．

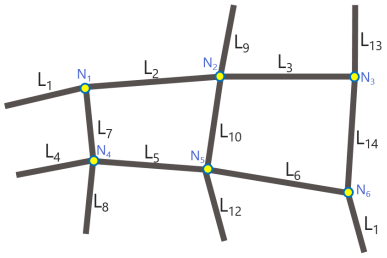

図 1　リンクとノードで構成された道路ネットワーク

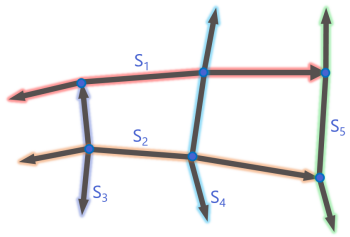

図 2　ストロークネットワーク

2. 関連研究

本研究に関連のある研究を以下に挙げる．

2.1 幅優先探索

幅優先探索 [3] は，始点となるノードから隣接するノードを探索し，そこからさらに隣接するノードに対して探索を繰り返して目的のノードを見つける．始点から近い順に探索をしていくため，探索するのノードはキュー (FIFO) を使って管理することになる．

2.2 深さ優先探索

深さ優先探索は，始点となるノードから目的のノードが見つかるか，子のないノードにたどり着くまで深く伸びていくような探索を繰り返し，そのあとは探索の終わってないノードまで戻って再度探索を繰り返す．探索するのノードはスタック (FILO) を使って管理することになる．

2.3 最良優先探索

最良優先探索は，何らかの関数を評価関数として定義し，評価関数の値 (コスト) に従って次に探索する最も望ましいノードを選択するアルゴリズムである．最良優先探索の例としてはダイクストラ法 [4] や A*アルゴリズム，均一コスト探索が挙げられる．経路探索に関しては，道路の特徴をコストにして渋滞を回避する経路を探索する研究 [5]，ダイクストラの最短経路探索から高速化アルゴリズムを開発する研究 [6] などがある．

2.4 先行研究の道なり優先経路探索

先行研究の道なり優先経路探索 [7] はリンク単位で探索する手法である．例えば，図 3 のような道路ネットワークで経路探索を行う場合，出発点から交差点ノード N_4 や N_5 までの最少ストローク数を求めるとき，N_3 は N_1 と N_2 の合計ストローク数を覚えておかなければならない．そこで，各ノードは接続するリンクの数だけ子ノードを持ち，その子ノードに対し接続リンク ID，ストローク ID，ストローク数，経路長を持たせた構造を持つ，道なり経路探索のアルゴリズムは下記の手法を用いて全ての子ノードに対して道なり経路探索を行う．

(1) ストローク数算出手法
着目するノードの子ノードのストローク ID とその直前のノードの子ノードのストローク ID よりストローク数を計算する．着目するノードの子ノードのストローク ID と直前のノードの子ノードのストローク ID が同じ場合，着目するノードの子ノードがもつストローク数は直前のノードの子ノードがもつストローク数となる．子ノードのストローク ID が異なる場合は，着目するノードの子ノードが持つストローク数は直前のノードの子ノードがもつストローク数に 1 加えたものとなる．

(2) ストローク数最適解算出手法
直前のノードの子ノードの数だけコスト比較を行い，着目するノードの子ノードに以下のコスト比較条件を満たす最少ストローク数とその時の経路長を登録する．

(a) ストローク数が最少
(b) ストローク数が同数なら，経路長が最小

このアルゴリズムで行う道なり優先経路探索はノードを接続するリンクの数だけ分割し，分割した子ノードに対して経路探索を行うため，計算に時間がかかってしまうという問題点がある．

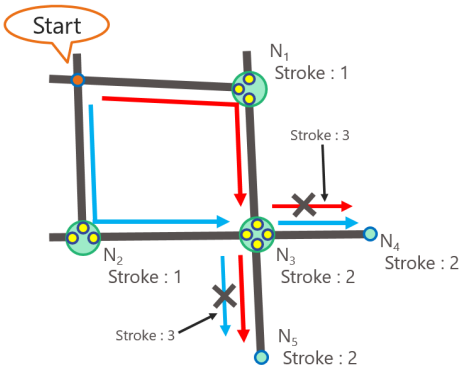

図 3　先行研究の道なり優先経路探索

3.　提案システムの概要

我々は，探索の高速化のために「交差ストローク」と呼ばれるストロークに交差するストロークを事前に求め，交差ストロークテーブルを構築する．そして交差ストロークテーブルを用いて経路探索をストローク単位で行なう最少ストローク数最短経路探索システムを提案する．

3.1　提案システムの特徴

本論文では，以下の特徴を有したシステムを提案する．

3.1.1　特徴 1 ストローク単位の幅優先探索

図 4 のような道路ネットワークが存在し，ストロークを用いずに経路探索をする場合，一般的に図 5 の幅優先探索やダイクストラ法などのように，交差点ごとに枝分かれする道路をリンク単位で探索する．また，先行研究の従来システムにおいてもストロークを使用して道なり優先経路探索を行っているが，道路リンク単位で探索を行うためデータ量がリンク数に比例して多くなり，計算に時間がかかってしまう．それに対し提案システムの探索アルゴリズムは，図 6 のようにストロークに交差するストロークを探索していき，ストローク単位の幅優先探索を行っている．探索し終えたストロークは訪問済みストロークリストに追加され，この訪問済みストロークリストは出発点からの到達に必要なストローク数によって分けられている．これにより到着点から出発点まで戻るように訪問済みストロークリストをたどることによって経路を効率的に求めることができる．

3.1.2　特徴 2 交差ストロークテーブルを用いた高速化

提案システムでは，探索中のストロークに交差するストロークを次に探索する．探索を進める毎にデータベースから SQL の Intersects 演算子を用いて交差するストロークを取得すると探索時間が長くなってしまう．そこで，事前に交差ストロークテーブルを構築することで交差するストロークを一括で求めることができるため，オンデマンドの計算量を削減できると考えた．

3.1.3　特徴 3 出発点と到着点の双方向からの探索による高速化

最少ストローク数を求める探索はストローク単位の幅優先探索を行なうため，出発点から到着点までの最少ストローク数が多いと探索対象のストロークが指数関数的に増えてデータが膨大になってしまい，計算時間が長くなる．そこで，出発点と到着点の双方向からの探索を行うことで訪問済みのストロークが少なくなり，探索の計算量の削減に繋がると考えた．

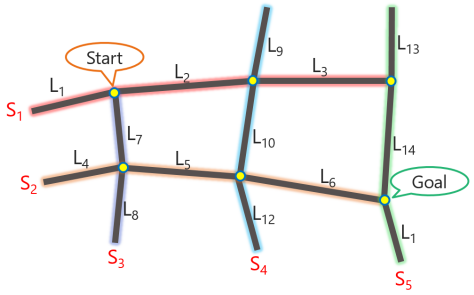

図 4　道路ネットワーク

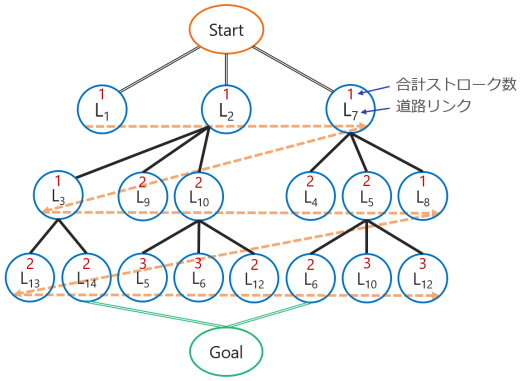

図 5　道路リンク単位の幅優先探索

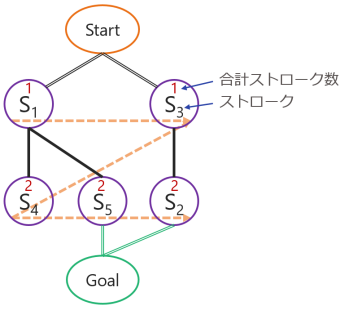

図 6　ストローク単位の幅優先探索

3.2　提案システムで使用するデータ

提案システムにおいて使用するデータは，道路データとストロークデータ，交差ストロークのデータである．

3.2.1 道路データ

道路のデータ構造を図 7 に示す．道路データには，ポイント，ノード，アークの 3 つのデータ がある．ポイントは，道路上の点を表し，位置座標 (緯度，経度) からなる．道路の形状は図のように道路上の隣接ポイントを順番につないでできる折れ線で近似的に表現される．ノードとは，道路ネットワークの交差点である．アークは始点ノードから終点ノードに至る道路であり，道路上のポイントの系列で表す．リンクはアークとノードをまとめたものであり，始点ノードと終点ノードの組をエッジと呼ぶ．リンクは表 1 に示すデータ構造を持つ．また，アークは線を表現するジオメトリ型 LineString の形式で表現する．

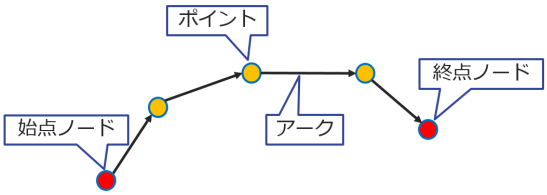

図 7 道路データ構造

表 1 リンクテーブル

カラム名	データ型	説明
ID	int	リンクを一意に識別するリンク ID
clazz	int	道路のクラス
source	int	リンクの始点ノード ID
target	int	リンクの終点ノード ID
x1	double	リンクの始点ノードの経度
y1	double	リンクの始点ノードの緯度
x2	double	リンクの終点ノードの経度
y2	double	リンクの終点ノードの緯度
km	double	ノード間の道路の長さ (単位は km)
geom_way	geometry	リンクの形状を表す

3.2.2 ストローク

ストロークとは，認知心理学に基づいて道路ネットワークをグルーピングしたものであり，道なりに続く道路に対応する [9]．しかし，道なりという概念は形状や人の感覚によって異なり，何をストロークと定めるかにより違いが出る [10]．本研究では，道路の形状を表現するアークを道なりの判定に用いる．以下のルールに従って，連続したリンクを同一のストロークと定める．

ルール 1 図 8 のリンク L_1 の端点に接続するリンクがリンク L_2 の 1 つのみ の場合，リンク L_1 とリンク L_2 は同じストロークに属する．

ルール 2 図 8 のリンク L_3 の端点に接続するリンクが複数の場合，注目するリンクと同じ道路のクラスで，隣接するアークの成す角 θ_1 が 45 度以下であり，かつ最小と なるアークを持つリンク L_4 をリンク L_3 の系列の対象とする．

ルール 3 大きな交差点内において交差点内リンクが存在する場合，交差点内リンクを挟むリンク を全て接続するものとみなし，ルール 2 を満たすリンクを系列の対象とする．

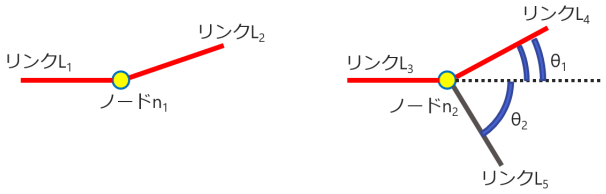

図 8 ストロークルール

ストロークデータベースにはストロークテーブルが存在する．ストロークテーブルのデータ構造を表 2 に示す．

表 2 ストロークテーブル

カラム名	データ型	説明
stroke_id	int	ストロークの ID
link_ids	int	ストロークに含まれるリンクの ID
stroke_length	double	ストロークの長さ
stroke_arc	geometry	ストロークの形状

3.2.3 交差ストローク

交差ストロークとは図 9 のように，ある 1 本のストロークに交差するストロークのことである．ストロークの全体集合 S_0 と交差ストローク判定関数 intersect(s_i, s_j) を定義すると，次のように議論を展開して整理できる $(s_i, s_j \in S_0)$.

$$\text{intersect}(s_i, s_j) = \begin{cases} \text{True } (si, sj \text{ が交差している場合}) \\ \text{False}(\text{その他の場合}) \end{cases}$$

そして $s_i \in S_0$ の交差ストローク集合 CS(s_i) は次のように定義される．

$$\text{CS}(s_i) = \{s \mid \text{intersect}(s_i, s)\text{=True}, s \in S_0\}$$

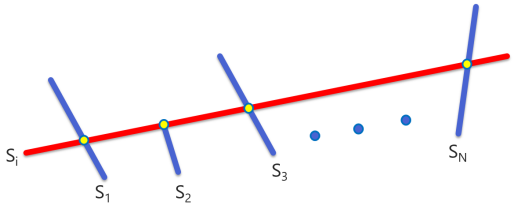

図 9 交差ストローク

交差ストロークテーブルには，ストローク ID とそれに交差する交差ストローク ID および交差ノード ID がある．また，交差ストロークテーブルのデータ構造を表 3 に示す．

表 3　交差ストロークテーブル

カラム名	データ型	説明
stroke_id	int	ストロークの ID
intersect_stroke_id	int	交差するストロークの ID
intersect_node_id	int	交差点ノードの ID

3.3　提案システムの構成

提案システムの構成を図 10 に示す．提案システムでは事前に交差ストロークテーブル生成機能により，道路データとストロークデータから交差ストロークテーブルを生成する．ユーザが指定した地図の範囲の道路データと交差ストロークデータを取得し，最少ストローク数経路探索機能により最少ストローク数で行くことができる経路候補を探索する．そして，最少ストローク数の最短経路決定機能により経路候補の中から最短経路を求める．これにより求まった最少ストローク数最短経路を経路描画機能により地図上に描画する．

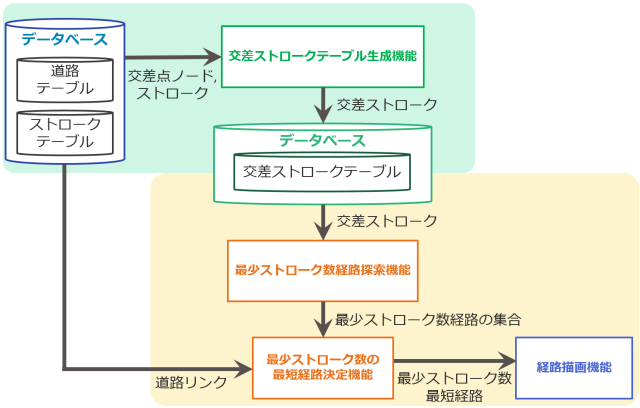

図 10　システムの構成図

4.　提案システムの実現法

提案システムは，交差ストロークテーブル生成機能，最少ストローク数経路探索機能，最少ストローク数最短経路決定機能からなる．それぞれの実現法を以下に示す．

4.1　交差ストロークテーブル生成機能

交差ストロークテーブル生成機能では，データベースから道路データとストロークデータを取得し，交差ストロークテーブルを作成する．

はじめに，交差ストロークテーブルを構築したい範囲の必要なストロークデータと道路データを取得する．ストロークデータからは，ストローク ID とストロークのアークを取得する．ストロークのアークは折れ線を表現するジオメトリ型 LineString 型の形式で表現している．道路データからはノード ID を取得し，ストローク ID と対応付けをしてストロークに含まれるノードが分かるようにする．取得したデータから新たに作成したストロークテーブル (以降，ストロークテーブル 2 と呼ぶ) を表 4 に示す．

表 4　ストロークテーブル 2

カラム名	データ型	説明
stroke_id	int	ストローク ID
node_ids	text	ストロークに含まれる全ノード ID
flatted_arc_series	geometry	ストロークの形状

ストロークテーブル 2 に含まれるすべてのストロークに対し，以下の SQL を実行する．INTERSECT 演算子は 2 つのクエリの出力を積結合する．すなわち，2 つのクエリの結果から同じものだけを出力する．ストロークとその交差ストロークが分かればそれぞれに共通するノードがあり，それが交差点ノードである．これらのデータから交差ストロークテーブルを作成する．

```
select stroke_id , node_ids from ”ストロークテーブル 2”
where st_intersects(flatted_arc_series,
(select flatted_arc_series from ”ストロークテーブル 2”
where stroke_id= ”ストローク ID”))
```

4.2　最少ストローク数経路探索機能

最少ストローク数経路探索機能では，右左折の回数が最も少ない経路を探索する．提案システムでは，一度に最少ストローク数とその経路を求めるのではなく，はじめに最少ストローク数を求めてその後に経路探索を行う．

4.2.1　最少ストローク数探索

出発点からストローク単位で到着点のストロークを探索する．以下に探索アルゴリズムの疑似コードを示す．

——最少ストローク数探索の疑似コード——

S_a：始点ノードを含むストローク
S_b：終点ノードを含むストローク
S_i：i 番目のストローク
$L(n)$：出発点から n 本で到達できる訪問済みストロークの集合
$CS(s)$：ストローク s の交差ストローク集合

$n = 1$; $L(n) = S_a$;
while(S_b の全要素が $L(n)$ に含まれない){
　for($S_i \in L(n)$){
　　for($S_j \in CS(S_i)$){
　　　if($Sj \notin L(n)$){
　　　　S_j を $L(n+1)$ に追加;
　　　}

```
    }
  }
n++
}
```

以上の探索により，出発点から到着点まで最少 n 本のストローク，つまり $n-1$ 回の右左折で行くことができる．

4.2.2 出発点と到着点の双方向からの最少ストローク数探索

上記の出発点からストロークを探索する手法は，最少ストローク数が多ければ多いほど図 11 と図 12 のように探索するストロークが多くなり，経路候補にならないストロークを多く探索してしまうため計算量の増加に繋がる．そこで，出発点と到着点の双方向から最少ストローク数探索をすることで，経路を求めるときに必要のないストロークの探索を減らすことができ，計算量を削減する．

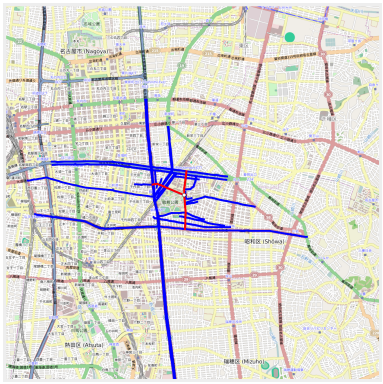

図 11 ストローク数 2 本

図 12 ストローク数 4 本

4.3 最少ストローク数最短経路決定機能

4.2 で求めた訪問済みストロークリストは，出発点からストローク何本で到達できるかによって分けられている．これを用いて最少ストローク数の経路候補を以下の手順で求める．

(1) 到着点に接続するストロークが含まれる訪問済みストロークリストを L_n とする．訪問済みストロークリスト L_n のストロークの交差ストロークのうち、訪問済みストロークリスト L_{n-1} に含まれるストロークのみを訪問する。

(2) 訪問済みストロークリスト L_{n-1} に交差するストロークのうち、訪問済みストロークリスト L_{n-2} に含まれるストロークを訪問する。

(3) (2) を n の値を 1 ずつ減らして続けていき、訪問済みストロークリスト L_1 まで訪問したら探索を終了する。

以上の方法により、最少ストローク数経路が全て求まる。全ての経路候補に対してリンク単位で経路をたどり，経路長を求める手順を以下に示す．

(1) 出発点ノードから出発点に接続する経路のストロークをリンク単位でたどる．

(2) 次に探索するストロークとの交差点ノードを発見した場合，その交差点ノードから次のストロークをリンク単位でたどる．

(3) (2) を繰り返し行い，到着点ノードを発見したとき，探索を終了する．

5. プロトタイプシステム

我々は提案システムを実現するために，以下のリソースを用いてプロトタイプシステムを実装した．

- Java 言語
- Eclipse
- PostgreSQL
- pgAdmin4

5.1 プロトタイプシステムの概要

プロトタイプシステムは，提案システムの最少ストローク数最短経路探索とダイクストラ法を用いた最短経路探索を実装した。そして最少ストローク数探索においては，出発点からのみの探索と，出発点と到着点の双方向からの探索の 2 通りの探索方法を選択可能になっている．地図データは OpenStreetMap の地図データが利用された地図サーバから取得する．入力した緯度経度より地図サーバから地図画像を取得し，研究室サーバ内にあるデータベースから範囲内に存在する道路データと，ローカルデータベースから交差ストロークデータを取得する．

5.2 プロトタイプシステムの動作

プロトタイプシステムの操作画面を図 13 と図 14 に示す．上部に入力インターフェースを，下部に地図画像を配

置している．入力インターフェースで探索方法を選択し，地図画像内から出発点，到着点の順にクリックすると選択した探索方法で経路探索を行い，経路が描画される．探索した経路の経路長，ストローク数，探索時間は Eclipse のコンソールに出力される．また，「移動」ボタンの上部にある「座標」,「地名」,「ランドマーク」の中から 1 つ選択し，左部にあるスケールから表示したい縮尺を選択した状態でボタンを押すことにより，中心としたい緯度経度で指定した縮尺の地図に変更する．「座標」を選択した場合，地図画像の中心としたい位置の日本測地系の緯度経度を入力することによりその座標に移動することができる．「地名」または「ランドマーク」を選択した場合，その右部にあるテキストボックスに地名または施設名を入力することによりその座標に移動することができる．また，移動は地図画像内で中心としたい位置を右クリックすることで右クリックされた位置を中心にした地図に変更することもできる．

プロトタイプシステムの使い方を以下に示す．

(1) 地図画像内を右クリックして地図の中心を移動するか，入力インターフェースで地図画像の中心としたい位置を設定する．

(2) 入力インターフェースから探索方法を”最短経路”，”最少ストローク数最短経路”，”双方向最少ストローク数最短経路”の 3 つの中から選択する．

(3) 地図画像内から出発点，到着点の順に 2 回クリックすると，(2) で選択した経路が描画される．

図 13 プロトタイプシステムの起動画面

図 14 最少ストローク数最短経路探索の実行画面

6. 評価実験

提案手法を元に 2 つの評価実験を行った．

6.1 評価実験 1

提案システムと従来システムの探索時間の比較実験を行う．

6.1.1 評価実験 1 の方法

プロトタイプシステムを用いて，名古屋市内で 15km 四方の矩形範囲の中から任意の 2 点の組を出発点と到着点として 200 組選定する．提案システムの最少ストローク数最短経路探索は，出発点の片方向からの探索と出発点と到着点の双方向からの 2 通りの探索があるため，それぞれに対し探索時間を求める．そして直線距離別・最少ストローク数別に探索時間の平均を計算して比較を行う．また，先行研究の道なり優先経路探索を行うことができる従来システムを用いて同様に名古屋市内で 15km 四方の矩形範囲の中から直線距離が 1km から 1km おきに 10km まで任意の 2 点の組を出発点と到着点としてそれぞれ 10 組選定し，ストローク数および平均探索時間を求める．

6.1.2 評価実験 1 の結果と考察

3 つの手法の探索時間の評価実験の結果を表 5 と図 15 および図 16 に示す．なお，提案システム 2 種および従来システムを以下のように呼んでいる．

- 手法 1：提案システム (出発点からのみの探索)
- 手法 2：提案システム (出発点と到着点の双方向からの探索)
- 従来手法：従来システム

図 15 より，探索時間は提案システムと従来システムともに直線距離が長くなるほど探索時間が長くなっているが，提

案システムは従来システムの 10 分の 1 以下であることが分かる．直線距離が 1km のとき提案システムは従来システムのおよそ 140 分の 1，直線距離が 10km のとき提案システムは従来システムのおよそ 40 分の 1 である．また図 16 より，提案システムは最少ストローク数に比例して探索時間が長くなるが，従来システムは最少ストローク数と探索時間に相関は無かった．したがって提案システムは，探索距離が 10km 以下であれば従来システムよりも短い時間で探索ができることが分かった．この理由として，従来システムはノードの中に子ノードを持つという構造であるから，例えば十字路の交差点の場合は通常の 4 倍の 4 つの子ノードに分けて探索を行う．それに対し提案システムはストローク数単位で探索を行い，経路候補を絞った後にリンク単位で経路長を計算し最短経路を求めるため，従来システムよりも短い探索時間で経路探索できたと考えられる．また，手法 1 と手法 2 を比較すると，探索時間に大きな差はないことが分かる．これは，ストローク数が多くなるほど最少ストローク数を求める計算時間は手法 2 の方が短くなるが，最少ストローク数の経路候補の中から最短経路を求める計算時間が同じであり，最少ストローク数を求める計算時間よりも長いためだと考えられる．

表 5　提案システムと SPMS 法の経路探索時間

直線距離 (km)	提案システム (s)		従来手法 (s)
	手法 1	手法 2	
1	4.19	3.98	563.36
2	4.81	4.80	576.21
3	7.96	7.70	596.63
4	8.53	8.22	604.24
5	9.80	8.46	667.44
6	11.61	9.36	735.96
7	13.91	11.35	744.92
8	18.09	15.44	806.42
9	20.14	16.68	1036.54
10	33.47	29.60	1133.70

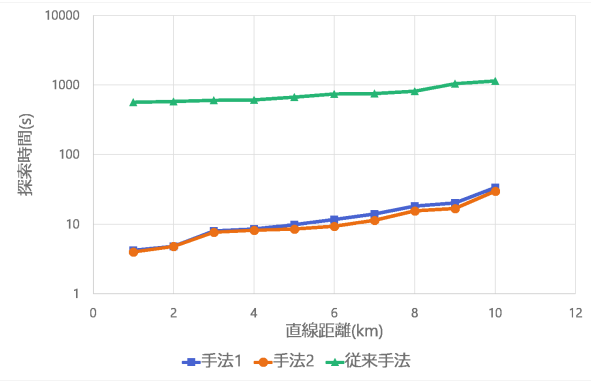

図 15　提案システムと従来システムの探索時間 (直線距離別)

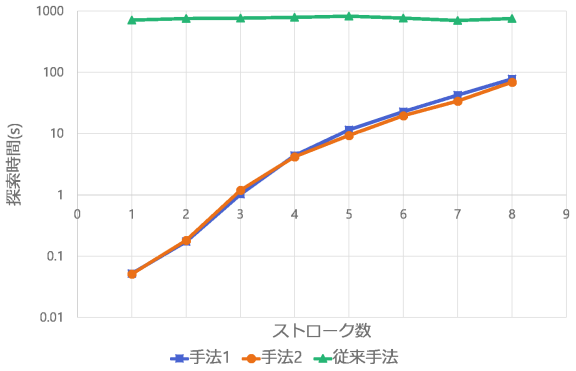

図 16　提案システムと従来システムの探索時間 (ストローク数別)

6.2　評価実験 2

名古屋市内で最も右左折の多い最少ストローク数経路を求める．

6.2.1　評価実験 2 の方法

国土交通省の国土数値情報ダウンロードサービス [12] から名古屋市のポリゴンデータを取得し，名古屋市内にある全てのストロークから交差ストロークテーブルを構築する．図 17 は名古屋市内の全ストロークである．提案システムの最少ストローク数探索を応用し，以下の疑似コードのアルゴリズムを名古屋市内で未訪問のストロークが無くなるまで探索を行う．

——ストローク探索の疑似コード——

S_a：名古屋市内の 1 本のストローク

S_i：i 番目のストローク

$L(n)$：最初のストロークから n 本で到達できる訪問済みストロークの集合

$CS(s)$：ストローク s の交差ストローク集合

$n = 1$; $L(n) = S_a$;
while$(L(n) \neq \phi)$\{
　for$(S_i \in L(n))$\{
　　for$(S_j \in CS(S_i)$\{
　　　if$(Sj \notin L(n))$\{
　　　　S_j を $L(n+1)$ に追加;
　　　\}
　　\}
　\}
$n++$
\}
return n

名古屋市内の全てのストロークに対してこれを行い，総探索時間と $L(n)$ の n の最大値を求める．

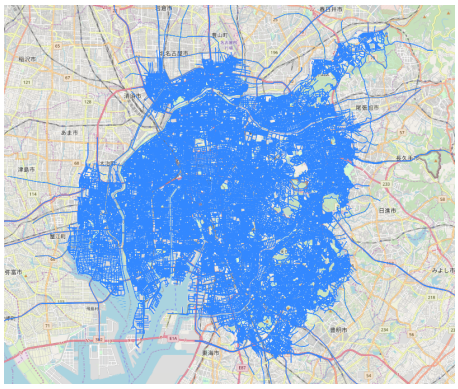

図 17　名古屋市内のストローク

6.2.2　評価実験 2 の結果と考察

総探索時間は 28 時間 12 分であった．また，n の最大値は 17 であった．このことから，名古屋市内ならば右左折が $n-1$ の 16 回以内で任意の場所まで行くことができるということが分かった．道なり優先経路で 16 回の右左折が必要な経路の最初のストロークは 7 本あり，そのほとんどが名古屋市内の郊外や田舎であった．これは，市街地は格子状に交差する道路が多いが，郊外などでは比較的入り組んだ形状の道路が多いためだと考えられる．

7.　おわりに

本論文では，道なり優先経路の探索方法として，交差ストロークテーブルに基づく最少ストローク数最短経路探索システムを提案し，その実現方法について述べた．また，提案システムに基づいてプロトタイプシステムを実装し，評価実験を行った．

評価実験では，提案システムと従来システムの探索時間について比較を行った．その結果，提案システムの経路探索時間は従来システムに比べ直線距離が 1km の場合約 140 分の 1，直線距離が 10km の場合約 40 分の 1 の探索時間で最少ストローク数最短経路を探索できることが分かった．

また，名古屋市内の全てのストロークから，交差ストロークテーブルを作成した．この交差ストロークテーブルを利用し，名古屋市内ならば右左折が 16 回以内で任意の場所まで行くことができるということが分かった．

今後の課題としては，最少ストローク数経路候補から最短経路を求める探索時間の削減のため，探索アルゴリズムの改善が挙げられる．本研究では最少ストローク数経路探索の計算時間の削減について主に取り組んできたが，その経路候補の中から最短経路を求める探索の計算時間については改善の余地があると考えている．

参考文献

[1] Google Maps
入手先 ⟨https://www.google.co.jp/maps/⟩（参照 2019-04-26）.

[2] Yahoo! 地図
入手先 ⟨https://map.yahoo.co.jp/maps/⟩（参照 2019-04-26）.

[3] Judea Pearl, HEURISTICS Intelligent search, Strategies for computer, Problem solving, pp.36-52, 1984.

[4] Dijkstra
入手先 ⟨http://www.deqnotes.net/acmicpc/dijkstra/⟩（参照 2019-04-26）.

[5] 原，塚原，狩野，多目的遺伝的アルゴリズムを用いたカーナビゲーションのための予測経路探索，情報処理学会高度交通システム研究会，ITS-24(5)，pp.31-38，2006.

[6] P.Sanders, D.Schultes, Engineering Fast Route Planning Algorithms, Experimental Algorithms(WEA), pp.23-36, 2007.

[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] 新帯里奈，山本大介，高橋直久，多層ストロークネットワークの構築と道なり優先経路探索への応用，第 9 回データ工学と情報マネジメントに関するフォーラム，E4-4，Mar.2017.

[9] Masaki Murase, Daisuke Yamamoto, Naohisa Takahashi, On-Demand Generalization of Guide Maps with Road Networks and Category-Based Web Search Results, Proceedings of Web and Wireless Geographical Information System 2015 (W2GIS2015), pp.53-70, Grenoble, France, May 2015.

[10] Robert C.Thomson, Dianne E.Richardson, The ‘Good Continuation’ Principle of Perceptual Organization applied to the Generalization of Road Networks, the 19th International Cartographic Conference, Ottawa, Canada, pp.1215-1223, 1999.

[11] Pablo MARTINEZ LERIN, Daisuke YAMAMOTO, Naohisa TAKAHASH, Encoding network-constrained travel trajectories using routing algorithms, International Journal of Knowledge and Web Intelligence, Vol.4, No.1, pp.34-49, May.2013.

[12] 国土数値情報　ダウンロードサービス
入手先 ⟨http://nlftp.mlit.go.jp/ksj/⟩（参照 2019-04-26）.