未知迷路探索ロボットにおける動的経路計画アルゴリズムの設計と評価

五十嵐柊司*1
*1 東京都立小石川中等教育学校 5年

1．研究の背景・目的

RoboCupJunior Rescue Maze競技[1]での活用を目的とし、**図1**のような有限範囲の未知迷路を効率的に全探索するロボットのアルゴリズムを設計した。実機のロボットを用いて限られた時間内に探索を行うために、安定性を備えながら効率的に全探索することを目標とし、マッピングを用いた全探索アルゴリズムと、ずれが生じた際の補正アルゴリズムを中心に研究を行っている。

研究においては、RoboCupJunior Rescue Mazeのルールに基づいて実機環境でロボットを動作させることを想定し、以下の条件を前提としている。

・フィールドは正方形のタイルによって構成された格子状の迷路である。
・ロボットの直進移動と回転移動にはそれぞれ異なるコスト（所要時間）がかかる。
・フィールド上のタイルにはいくつかの種類があり、タイルによって移動コストが異なる。
・フィールド情報の取得にはオドメトリやLiDARセンサを使用する。

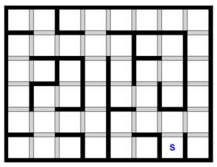


図1
迷路の例

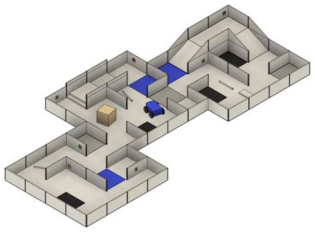


図2
RoboCupJunior Rescue Maze
フィールドのイメージ[2]

2．全探索アルゴリズムの設計と評価

2-1. 提案手法

既存の格子状迷路の全探索手法として、壁に右手をつけて進む「右手法」を基本としつつ、訪問済みタイルを記憶しておき訪問回数の少ないタイルが接している方向へ向かうことですべてのタイルへの到達を可能とする「拡張右手法」が広く知られている。

しかし、拡張右手法では常に右壁をつたうことを優先するため、**図3**のように全探索をする際に遠回りしてしまうことが多くある。
（左下がスタート。青い移動が無駄である。）

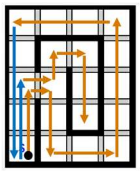


図3
拡張右手法による探索

ここでは、未到達タイルを保存しておき、現在地から最も近い未到達タイルへ最短経路で移動する探索を繰り返すことで、効率的な全探索を実現するアルゴリズムを提案する。アルゴリズムの詳細を以下の疑似コード(Algorithm 1, 2)に示す。

ダイクストラ法を用いて現在地から最も近い未到達タイルを求めるが、それぞれのタイルにつき北・南・東・西の4方向の状態のノードを用意することにより、旋回と回転のコストを考慮した最短経路算出を行うことを可能とした。

Algorithm 1: Full Maze Exploration with Directional Dijkstra

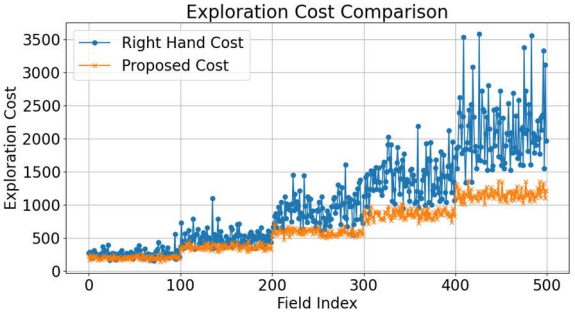
```
Initialize map and position p;
Initialize direction d;
while ExistsUnvisitedTile() do
    info ← GetSensorInfo();
    UpdateMap(p, info);
    path ← DijkstraToNearestUnvisited(p, d);
    if path = ∅ then
        return Finished;
    Follow(path);
return Finished
```

Algorithm 2: DijkstraToNearestUnvisited(p, d)

```
Initialize priority queue Q;
push node (p, d) with cost 0 into Q;
Initialize dist[·] ← +∞;
dist[p, d] ← 0;
while Q not empty do
    (pos, dir, cost) ← ExtractMin(Q);
    if TileUnvisited(pos) then
        return ReconstructPath(pos, dir);
    for each nextDir ∈ {0, 90, 180, 270} do
        if NoWall(pos, nextDir) then
            nextPos ← Neighbor(pos, nextDir);
            turnAngle ← |(nextDir − dir) mod 360|;
            if turnAngle = 0 then
                moveCost ← c_move
            else
                if turnAngle = 180 then
                    moveCost ← 2c_turn + c_move
                else
                    moveCost ← c_turn + c_move
            newCost ← cost + moveCost;
            if newCost < dist[nextPos, nextDir] then
                dist[nextPos, nextDir] ← newCost;
                parent[nextPos, nextDir] ← (pos, dir);
                push (nextPos, nextDir, newCost) into Q;
return ∅
```

2-2. 評価実験

穴掘り法で生成した木構造迷路に対し、一定割合で壁を破壊することでランダムな迷路500個を作成し、拡張右手法および提案手法で探索コストを比較した。



Field Index		Field Size
0 – 99	:	6x6
100 – 199	:	8x8
200 – 299	:	10x10
300 – 399	:	12x12
400 – 499	:	14x14

図4
拡張右手法（青）と提案手法（橙）の比較

2-3. 考察

実験結果から、提案手法は拡張右手法に比べてコストを削減することができることが分かった。
また、提案手法は同一サイズの迷路においてコストの分散が小さく、安定した探索性能を示した。

3．ずれ補正アルゴリズムの設計

実機環境で動かす際には、マップの情報を走行モータのオドメトリとLiDAR、ジャイロセンサなどによって取得するが、障害物やバンプの多い険しい地形ではその際に何らかのずれが発生する可能性がある。その際にマッピングと探索計画を適切に修正し、全探索できるように復帰できるアルゴリズムを備えておくことは重要である。
現在は、発生する可能性が高いと考えられる以下の2つのずれに対応した補正アルゴリズムを開発しているところである。

- 前進している際の1マスずれ　- 回転する際の90度ずれ

発見済みのタイルの座標に達した時、周囲の壁情報と保存されたマッピングデータに矛盾が無いかを確認することでずれを検出し、ずれが起きた位置を全探索で探すことでマッピングと矛盾しないずれ発生位置を特定する。
この際、必ずしも特定できるとは限らないので、ずれ位置の特定に失敗した場合はマッピングデータを部分的に消去して探索をやり直す。

4．結論・今後の研究計画

提案アルゴリズムは未知迷路の全探索において、走行コストおよび安定性の点で拡張右手法より優れていることを確認した。

今後は、Raspberry PiやLiDARを搭載した開発中のロボット（**図5**）にて本探索アルゴリズムを動作させ、実機応用に向けて精度を高めていく。
特に、シミュレーションでは検証することが難しい測定ずれについて、実機で動かした際にどのようなずれが問題となるのか、どの程度の補正が必要なのかを、実際にロボットを動かすなかで調査・検証していく。

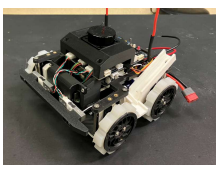


図5
開発中のロボット

[1] RoboCupJunior Rescue Maze：https://junior.robocup.org/rcj-rescue-maze/
[2] RoboCupJunior Rescue Maze Rules 2025 (RoboCupJunior Rescue Committee) より引用

本研究で作成したプログラムはこちらのGitHubリポジトリにて公開しています：