J-055

移動ロボットの経路探索の高速化のためのマップリンク法
Map-link Method for Efficient Path Planning

藤本敬介 * **守屋俊夫** * **中山泰一** †
Keisuke Fujimoto Toshio Moriya Yasuichi Nakayama

1. まえがき

移動ロボットの経路計画とは，ロボットの形状や移動モデルを考慮し，初期姿勢から目標姿勢までの動作の手順を探索することである．移動ロボットにとって，経路計画問題は基本的な問題の一つであり，古くから多くの研究がされてきた．経路計画問題は大きく分けて，ロボットがホロノミックな性質を持つかどうかに分けられる．ホロノミックな性質を持つ場合，ロボットは任意の方向へ自由に移動する事ができる．そこで，ロボットが障害物に衝突しない姿勢空間（コンフィギュレーション空間）を計算し，その空間内での探索により目的地までの最短経路を決定できる．これに対し，ホロノミックな性質を持たない移動ロボットとして，例えば自動車などが挙げられる．この場合は，瞬間的には左右に動く事ができないという特徴があり，任意の方向に進むためには，切り返しを行う必要がある．そのようなロボットが障害物に衝突しないような目的地までの経路を求める際には，移動方向を限定しつつ経路上に障害物が存在しないように衝突判定を行うことで，最短の経路を求める．

最短の経路を効率よく求めるための具体的な手法のひとつとして，ロボットの移動可能な経路の集合をグラフとして表し，そのグラフを探索するという手法がある．上記グラフは，ロボットの姿勢を示すノードを予め複数設置し，そのノード間をロボットが通行できる事を示すエッジで接続することで作成する．多くの手法では，障害物に衝突しないような経路から成るグラフを，障害物の形状に応じて事前に作成しておく．しかし，環境の形状が変わった際にはグラフを作り直す必要があり，それに伴いロボットの移動できる経路の再計算が必要となる．また，グラフの事前作成を行わず，経路探索時に衝突判定をすることで，通行可能な経路を辿る手法もある．この手法は周辺形状が変化しても再探索が可能であるが，探索時の衝突判定計算が必要となる．どちらの手法も，環境の形状が変化するような場合に，実行時に周辺形状に対しての衝突判定を行う必要があり，多くの計算時間がかかるという問題がある．

これに対し，本稿では障害物との衝突判定を行うことなく高速に経路探索を実現するためのマップリンク法を提案する．マップリンク法では，障害物が無い空間内の各地点と，そこを通過する経路の対応リストを事前に作成しておく．そして，経路探索の実行時に，衝突判定計算を行う代わりに，障害物が存在する地点に対応するエッジをリストから読み込み，エッジを通行禁止にする．これにより，探索時に衝突判定の計算をせずとも，上記リストの参照のみでロボットが移動可能な経路の探索が可能となる．また障害物の形状が変わったとしても，変更部におけるリストを参照するのみで，衝突判定を行うこと無しにグラフを更新可能である．さらに本手法では，ロボットの移動可能な経路は，どの地点を基準にしても同じになることを利用し，まず小さな空間内でマップリンクを作成し，これをタイルのように再帰的に並べ接続することで，大規模な地図にも適用可能である．実験によって，従来必要であった衝突計算をせずに経路の高速な探索が可能となる事を確認し，静的な環境に関しては同程度の速度に保ちつつ，環境が変化した場合においては従来手法と比較して約 10 倍程度の高速化を実現した．

2. 関連研究

本節では車と同等の操舵機構を持つ移動ロボットのための既存の経路計画について述べる．上記移動ロボットの経路計画としては，まず障害物が無い場合の 2 姿勢間を最短の動作で接続する経路を見つける手法が研究された．Dubins はロボットが一定速度で前進のみを行う場合について 2 姿勢間の最短経路は直線と最小旋回半径の円弧の有限個の組み合わせで構成される事を示した [2]．また Reeds らは前後方向にロボットが移動する条件下で，最大 5 個の直線と円弧の組み合わせで最短経路が構成できる事を示した [8]．

しかしロボットが実環境上で動作する場合は障害物を考慮して適切な経路を動く必要がある．障害物があった場合はロボットと衝突しない経路を探索する必要がある．その際にロボットと障害物との衝突判定計算を行うが，その計算を効率的に行うために障害物を多角形に近似して経路計画を行う手法が提案された [5]．

しかし，複雑な環境では適切に多角形として近似する事は難しくなるため，自由な形状を扱える必要がある．そこで占有格子地図を用い，空間を量子化してビットマップとして障害物を表現した環境下での経路計画手法も多く提案されている．Barraquand らはコンフィギュレーション空間内において，空間内をロボットの移動モデルに従って操舵回数が最小になるようにダイクストラ法を用いて探索を行う手法を提案した [1]．Guang らの提案した Customizable-PRM(C-PRM) 法を用いた計画法では，障害物に衝突しない経路を大まかに計算しておき，探索した経路に対して詳細に衝突判定を行う手法を提案した [9]．

また，近年では Rapidly-Exploring Random Trees(RRT) を用いたグラフベースの手法が提案されている [6][7]．RRT では，初期姿勢・目標姿勢から空間内にランダムに作成したノード間を，ロボットのモデルに従い逐次的に衝突判定計算をしながら繋げてゆく手法であり，動的環境や複数のロボットが存在する

*日立製作所, Hitachi Ltd.
†電気通信大学, The Univ. of Electro-Communications

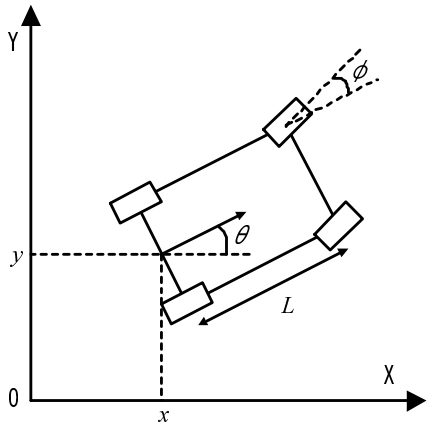

図 1: ロボットモデル

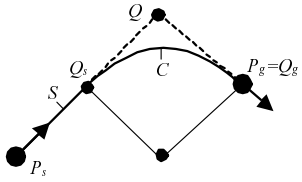

図 2: SC(直線＋円弧)

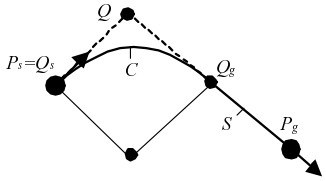

図 3: CS(円弧＋直線)

環境でも用いられている．

しかしながら，それらの障害物の存在する環境下での経路計画手法には，実行時の衝突判定計算量が大きくなるという問題がある．それらに対し本手法では，前計算によって各領域からロボットの通る経路への対応を求めておき，実行時には障害物に衝突するエッジを対応に従い選択することで，実行時に衝突判定の計算を行わずに経路を探索する手法を提案する．

3. ロボットモデル

3.1. ロボットの動作モデル

本稿で扱うロボットモデルは図 1 に示すような前輪操舵駆動型の 4 輪ロボットとする [1]．ロボットの姿勢は後輪の中心点を (x, y) とし，ロボットの角度 θ として 2 次元平面状に表される．前輪操舵駆動型のロボットは前輪の操舵角によって移動方向の変化量が決定され，ここでは操舵角を ϕ とする．前輪と後輪の距離を L とする．その時，ロボットの位置の変化量 $(\dot{x},\dot{y},\dot{\theta})$ は前輪の回転駆動速度 ω，車輪径 r を用いて以下のように表すことができる．

$$\begin{cases} \dot{x} = r\omega\cos(\theta+\phi) \\ \dot{y} = r\omega\sin(\theta+\phi) \\ \dot{\theta} = r\omega\frac{\sin(\phi)}{L} \end{cases} \tag{1}$$

3.2. ノード間の接続方法

本手法は経路探索のためにロボットの通行可能な部分的な経路をエッジとしたグラフを作成し，グラフ上の最短経路を探索する．ここで，3.1 節に示したようにロボットは自由方向に進む事ができないため，2 ノード間をエッジで接続する際に，ロボットが通るのに適した部分経路を作成する必要がある．移動ロボットのための，2 姿勢間の接続方法として，円弧と直線の組み合わせによる手法が広く用いられている．本稿でも円弧と直線の組み合わせを用いたノード接続を適用する．ただし本手法では，一般的に用いられている Reeds らの手法を用いず，より単純な手法を用いる．以下に，本手法での 2 姿勢間の接続手法について述べる．

本稿では，2 姿勢間の繋ぎ方を，1 つの直線と 1 つの円弧の組み合わせによって表す．ここでは直線を S として，円弧を C として表し，その組み合わせとして SC(図 2) または CS(図 3) の 2 つのパターンで表す．

始点の位置と方向ベクトルを (P_s, N_s)，同様に終点を (P_g, N_g) とし，2 姿勢ベクトルの交点を Q として，

$$Q = P_s + t_s N_s = P_s + t_g N_g \tag{2}$$

を満たす係数 t_s, t_g を求め t_s と t_g の積が 0 未満の場合は，Q が 2 つのベクトルの同一方向上の交点に作られ，始点から終点を結ぶ直線から大きく逸脱するため，ここでは経路の作成を行わない事とする．

また $|P_sQ| \geq |P_gQ|$ の場合は作成する経路を SC とし，$|P_sQ| < |P_gQ|$ では CS とする．以下，SC の場合について述べる．

SC の場合，直線経路は P_s と Q_s 間を接続する．Q_s および Q_g は，

$$\begin{cases} Q_s = P_s + (|t_s| - |t_g|)N_s \\ Q_g = P_g \end{cases} \tag{3}$$

となり，Q_s と Q_g 間を円弧によって接続する．円弧の中心は Q_s と Q_g から出る，それぞれ N_s, N_g と直交する法線ベクトルの交点とする．このとき曲率半径 $1/\kappa$ は $|t_g| - |t_s|$ となり，曲率 κ は $1/(|t_g| - |t_s|)$ となる．これにより円弧によって Q_s と Q_g 接続し，直線と繋げることで部分的な経路とする．CS の場合も同様に求める．

本稿では円弧を用いてノード間の接続を行うが，この場合 2 次の微分値が連続とならないため，操舵角を急激に変える必要がある．ただし，本手法はノード間の経路の接続法は自由に設定可能であるため，円弧の代わりにクロソイド曲線を用いればより走行に適した経路を作成することも可能である．

4. マップリンク法に基づくタイルの作成

本節では，実行時に衝突判定計算を行わずに経路探索を実現するための，マップリンク法について述べる．本手法では，占有格子地図上の各領域（本稿ではピク

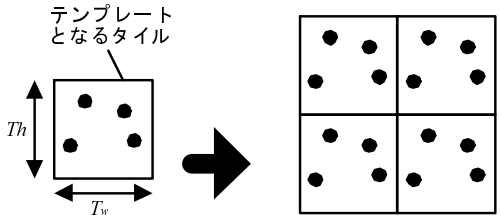

図 4: タイルを敷き詰めてグラフを作成

セル単位での分割とする）がグラフ上において，その位置と重なるエッジのリストを保持しておき，実行時に障害物の位置に応じて上記リストに従いエッジを通行止めにする．さらにこれを，広い環境に適用させるために，前計算した少領域内の上記リストをテンプレートとして，タイルのように再帰的に並べることにより（図 4），広い範囲にも適用可能となる．本稿では，上記小領域をタイル，ピクセルとエッジの対応表をマップリンクと呼ぶこととする．本節ではマップリンクの計算に基づくタイルの作成の手順について述べる．

4.1. 準モンテカルロ法によるノード作成

はじめに，タイル上にノードを作成する．ノードの位置をランダムに配置することによる経路計画法は多く提案されているが，特に複雑な領域や狭い領域において，本来通ることのできる領域であっても，図 5 のようにサンプルの偏りによって通れないと判定される場合がある．一方，動的な環境など，障害物の形状を事前に限定しない場合は，偏りの少ないサンプル法が望ましい．そこで本手法では，サンプルの偏りが少ない準モンテカルロ法を用いて図 6 に示すようにノード位置を決定する．本手法では準モンテカルロ法として，Hammersley 点群を利用した [10]．Hammersley 点群はサンプルする点数を予め決める必要があるが，他の手法と比較して一様性が高いという利点がある．

ノードの生成では，タイルの大きさを $[0, T_w] \times [0, T_h]$ としたときに，Hammersley 点群を用いてタイルの範囲内にノード位置を設置する．また，ノードの示す向きについてもランダムに作成する．各ノードはインデックスを持ち (x_p, y_p, θ_p) として表す．

4.2. エッジによるノード間の接続

ここでは，ノード間のエッジの接続方法について述べる．エッジの接続法としては，タイル内のノードの接続と，隣接するタイル間でのノードの接続に分けられる．

4.2.1. タイル内のノード間でのエッジ接続

タイル内のノード間でのエッジ接続の際は，タイル内に存在する全てのノードの組み合わせについて，エッジを作成するかどうかの判定を行う．まず，直線距離が遠い場合は組み合わせを除去する．ここで，4.1 節において，ひとつの位置に複数の方向を持つノードを重複して作成すれば，1 回の直線距離の判定で複数のノードの組み合わせの判定を一括して行うこともできる．続いて，直線距離が一定以下となる 2 グループ間の各ノードの組み合わせについて，3.2 の手法で曲率を求め，閾値以上となった場合に接続を行う．

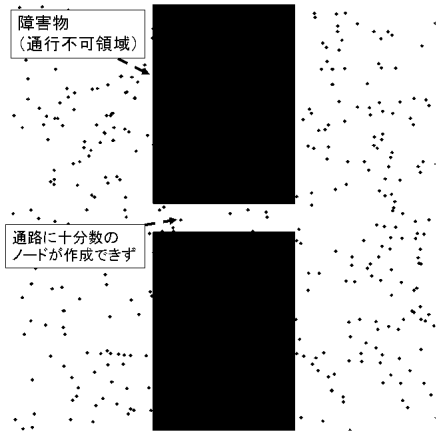

図 5: 乱数によるノード作成

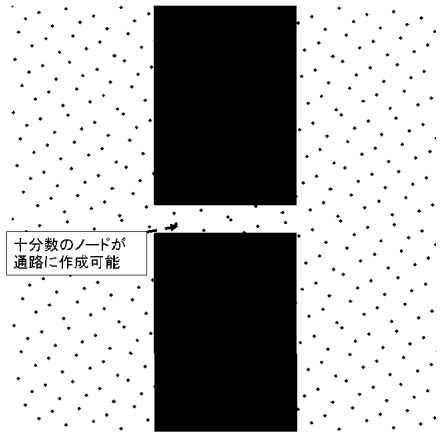

図 6: 準乱数によるノード作成

4.2.2. 隣接タイルへのエッジ接続

タイルを並べて全域を覆うのに十分な大きさのグラフを作成するため，タイル内のグラフと，隣接するタイルのグラフを接続する必要がある．そこで，図 7 のように隣接するタイル内のノードに接続を行う．隣のタイルのグラフへのエッジを作成する際に，各エッジはそのタイルの方向を保持しておく．仮定する隣接タイルは周辺 8 方向 (自タイルを含めて 9 個となる) に作成し，自タイルを含めた各タイルへの方向を $d_i \times d_j = [-1, 0, 1] \times [-1, 0, 1]$ とする．

接続の判定は 4.2.1 節と同様に直線距離によるノードグループの選択および，直線と円弧で接続した場合の曲率によって行う．タイルの位置関係を考慮したノード (p, q) 間の位置関係は，タイルの大きさを $T_w \times T_h$ として始点 (x_p, y_p, θ_p)，終点の姿勢を $(x_q + d_i T_w, y_q + d_j T_h, \theta_q)$ としてエッジの接続判定を行う．ただし，ノード p から見たノード q のタイルの方向を $d_i \times d_j$ とし

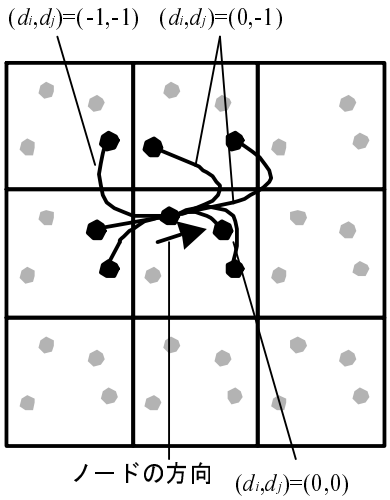

図 7: 隣接するタイルへの接続例

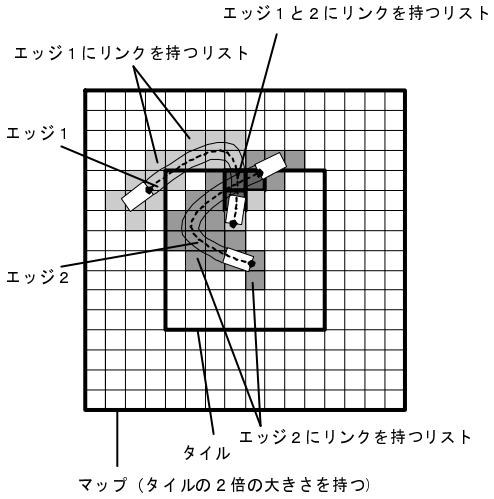

図 8: マップにおけるエッジへのリンク

た．接続可能と判定した場合，エッジは接続元と接続先のノードインデックスと，接続先のタイルの方向を保持する．

4.3. マップリンクの作成

マップリンクを作成するために，どのピクセルにどのエッジが衝突するかの対応関係を計算する．まず前節までに作成した各エッジについてロボットが通る場合に，どのピクセル上と重なるかを求める．続いて，図 8 のように通過する各ピクセル上に対応するリストにエッジのインデクスを追加する．

ここで，隣接するタイル間でのノード接続や，大きく曲がった経路を作成した際に，エッジが通過するピクセルが自タイルの外となる場合が存在する．そこで，図 8 のように，リンクを保持する領域がタイルの外側を覆うように作成することとする．ここでは，上記領域をタイルの 2 倍とし，タイル上のピクセル (x_m, y_m) からマップリンクへの座標 (x'_m, y'_m) を，

$$\begin{cases} x'_m = x_m + T_w/2 \\ y'_m = y_m + T_h/2 \end{cases} \tag{4}$$

のようにして変換することとした．図 8 の例では，タイル内の 2 つのエッジに関するマップリンクを示す．2 つの薄い灰色部のピクセルはそれぞれ 1 種類のエッジが通過するために，通過するエッジのみのリストを保持する．また，濃い灰色部は 2 つのエッジが通過し，リスト内に 2 つのエッジを保持する．

4.4. タイルの保存

本手法では，タイルの情報 (ノード，エッジ，マップリンク) はノード数およびエッジ作成の際の閾値に，およびロボットの形状によって決まる．そこで，あらかじめ作成しておいて使用時に読み込む事により高速な起動が可能となる．また，マップリンクを参照しながらエッジを選択することによって，障害物の形状が変化した場合にも，一度起動してしまえば高速にグラフを更新する事が可能である．

5. 有効なエッジの選択によるグラフ作成

単一のタイルのみを用いた場合，大きさに制限があるためにタイル以上の大きさとなる地図に対して適用する事ができない．しかし，大きなタイルを用いた場合，様々な大きさの地図へ対応は可能であるがタイルのデータ量が大きくなってしまう問題がある．

そこで，本手法ではタイルを再帰的に並べ，タイル間を接続することで十分な大きさのグラフを作成する．それにより，地図の大きさが変化した場合にも柔軟に対応する事ができ，タイルのデータ量も少なくて済む．本手法では，地図に対して十分な大きさになるまでタイルを並べ，それぞれのタイル間でのエッジ接続を行い，各タイルのマップリンクと地図上の障害物の比較を行い，通行することが不可能なエッジを選択する．以下に詳細について述べる．

5.1. タイル間のグラフ接続

作成した小さなタイルを，地図を囲むのに十分な大きさとなるまで並べ，隣接するタイル間でエッジの接続を行う．なお，必要なタイル数 $N_w \times N_h$ は

$$\begin{cases} N_w = \lceil \frac{M_w}{T_w} \rceil \\ N_h = \lceil \frac{M_h}{T_h} \rceil \end{cases} \tag{5}$$

となる．4.1 節においてタイルに登録されたノード (x_p, y_p, θ_p) の位置は，タイルのインデックスが (a_p, b_p) となる場合，$(x_p + a_p T_w, y_p + b_p T_h, \theta_p)$ となる．

4.2.2 節において，隣接するタイル間のエッジについて，接続方向 (d_i, d_j) と接続元および接続先のノードのインデックスをそれぞれ (p, q) として決定した．タイルを並べる際に，隣接するタイルに接続されるエッジについて，接続元のタイルのインデックスを (a_p, b_p)，接続先のタイルの方向 (d_i, d_j)，ノードのインデックスを q としたとき，接続先の位置 (x, y, θ) は以下のよう

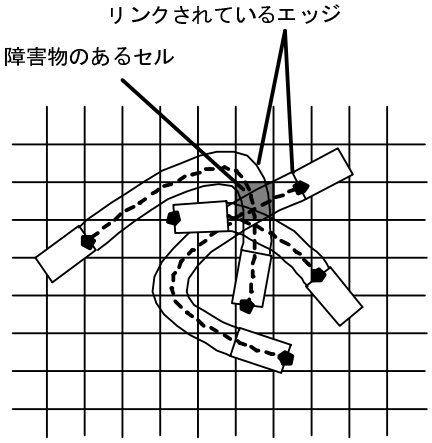

図 9: 障害物からエッジへの対応付け

に展開される．

$$\begin{cases} x = x_q + (a_p + d_i)T_w \\ y = y_q + (b_p + d_j)T_h \\ \theta = \theta_q \end{cases} \quad (6)$$

5.2. 通行不可能となるエッジの選択

作成されたマップリンクとの照らし合わせを行い，有効な辺の選択を行う．図 9 のように，入力された障害物の座標と衝突するエッジのリストを，対応するマップリンクを用いて選択する．よって，ここで選択されたエッジは内部に障害物の座標を含むために通行不可能とする．

まず，本稿では各マップリンクはタイルの大きさの 2 倍としたため，障害物の座標 (o_x, o_y) を含むマップは，以下の範囲のタイルに含まれる．

$$\begin{cases} (a_{\min}, a_{\max}) = (\lfloor \frac{o_x}{T_w} - \frac{1}{2} \rfloor, \lfloor \frac{o_x}{T_w} + \frac{1}{2} \rfloor) \\ (b_{\min}, b_{\max}) = (\lfloor \frac{o_y}{T_h} - \frac{1}{2} \rfloor, \lfloor \frac{o_y}{T_h} + \frac{1}{2} \rfloor) \end{cases} \quad (7)$$

上記範囲内の各マップに対し，障害物の座標 (o_x, o_y) に対応するエッジを通行不可能とする．タイルのインデックス (a_p, b_p) におけるマップおいては，

$$\begin{cases} x = o_x + (\frac{1}{2} - a_p)T_w \\ y = o_y + (\frac{1}{2} - b_p)T_h \end{cases} \quad (8)$$

のピクセルからリンクされたエッジが対象となる．本計算は，障害物上のピクセルに対して実行するため，エッジの選択に必要な計算時間は障害物の大きさに比例する．本手法では，本処理を高速に行うために，障害物の外周のみについてのみエッジの選択を行う．

5.3. 任意点のグラフへの追加

作成したノードはランダムにサンプルするため，サンプル密度が低い場合，ユーザが指定した任意位置から十分に近いノードが存在しない場合がある．そこで，経路の探索の際に，指定した初期姿勢と目標姿勢から，それぞれ周辺のノードに対して新たにエッジを作成する必要がある．本節はその作成手法について述べる．

まず指定された初期（目標）姿勢から直線距離の十分近いノードの選択を行う．ここでの距離の閾値は 4.2.1 節の値を使用した．続いて，初期（目標）姿勢から選択したノードグループ内のそれぞれの方向のノードに対し，部分的に経路を作成する．ここでも 4.2.1 節と同様に，曲率によってエッジを作成するかどうかの判定を行い，さらに各経路上をロボットが走行した場合に通過する領域を求め，障害物と衝突しないエッジを加える．

6. 実験

本手法は，障害物との衝突判定計算を行う代わりに，障害物位置とマップリンクの照合を行うことで経路探索用のグラフを作成するものである．そこで本実験では，マップリンクの導入によって，上記衝突判定計算を，障害物の位置情報に関わらず事前に行えることの効果を調査する．具体的には，Barraquand らの手法 [1] のように探索実行時に毎回通行経路との衝突判定を行った場合，および RRT ベースの手法 [6][7] のように障害物の情報が得られた後に，衝突判定を行いつつグラフを作成する場合との比較を行う．前者の手法では，グラフ上を探索する際に衝突判定を行い通行可否を調べる．これを手法 1 とする．また，後者の手法では，障害物が得られた際にグラフ上の経路の通行可否を調べることで行う．これを手法 2 とする．これらの手法に対し，マップリンクを導入した際の計算時間を比較する．

実験は，CPU が Intel Pentium M CPU 1.70GHz，主記憶容量が 1.2GB の計算機で行った．まずノード数を変化させた場合のタイル作成時間およびタイルあたりのデータ量の比較を行った．続いて，600×600 の大きさの地図と 2000×2000 の地図に対して，それぞれ経路探索を行った．以下に実験の結果を示す．

6.1. タイルの作成

表 1 にタイル作成に必要な計算時間およびデータ量を示す．本稿ではロボットの大きさを 40×20 とした．また，タイルの大きさを，50×50，100×100，200×200 とした．なお，エッジ作成のための曲率のパラメータを 0.3 とした．

エッジの計算時間が大きいのは，各ノード対についての接続を行い，曲率や長さに応じた判定計算を行っているからである．ノード数が 2 倍になるとノード対の数は 4 倍になり，計算時間・データ量もそれに従い増加している事が確認できる．

なお，タイルの構成は内部に持つノード数と，ロボットモデル，通行可能な部分経路を決定するパラメータによって決定される．そのため，一度タイルを作成してしまえば，環境が変わったとしても，上記パラメータを変えない限りは，同じタイルを使えばよい．

6.2. 経路生成

本節では，従来行われていた障害物とエッジの衝突判定に基づく経路探索法との比較を行う．本実験では，(1) エッジと障害物との衝突判定を探索時に行い，障害物と衝突しないエッジのみを辿る手法と，(2) 障害物の形状が入力された時点で，全てのエッジの衝突判定を行い，障害物と衝突するエッジをあらかじめ抽出する

表 1: タイル作成時間

ノード数	50	100	200
ノード作成 (s)	0.04	0.04	0.04
エッジ作成 (s)	3.415	13.5	53.5
マップリンク作成 (s)	1.82	4.07	11.546
合計 (s)	5.27	17.6	65.1
データ量 (MB)	23	86.0	333.2

手法と，それぞれ探索時間について比較を行った．なお，グラフ内での経路探索法は A^* 探索 [4] を用いた．また，本節ではタイル内のノードグループ数を 100 として実験を行った．

まず地図の大きさが 600×600 の環境下での経路の探索を行った．各手法とも同じグラフを用いているため，どの手法を用いても作成される経路は図 10 および図 11 となった．また障害物と衝突するエッジの選択および経路探索に関する計算時間を表 2 および表 3 に示す．なお，各表における比較手法として，手法 1 は探索時に障害物とエッジとの衝突判定を行う手法であり，手法 2 は障害物と衝突しないエッジを予め作成する手法である．

実験の結果，手法 1 について図 10 では 1.67s，図 11 では 3.15s となった．手法 1 での衝突判定では，ノード間を繋ぐエッジを通る際にロボットが通過する領域を計算し，障害物の領域を参照する事によって判定を行うために，前計算がない一方，探索に多くの時間を必要とした．

手法 2 は，障害物に衝突しないエッジのみのグラフを事前に作成する手法である．探索に必要な時間は図 10 では 17ms，図 11 では 35ms となった．ただし，グラフを作成するために 14.6s の計算時間が必要であった．手法 2 は，環境の形状が一定ならば一度グラフを作れば以降は高速な探索が可能であるが，環境の形状が変わる場合にグラフを再作成する必要があり，その計算時間が多くかかることが分かる．

それらに対し，本手法はマップリンクの参照のみで，衝突判定の計算をせずに，障害物と衝突しない経路を探索可能である．まず，グラフ作成のために，通行可能なエッジを調べるための時間は，図 10 では 690ms，図 11 では 270ms となった．エッジ選択ではリンクリストを辿るだけでよく，衝突判定の複雑な計算をする必要が無く，手法 2 の衝突判定計算に基づくグラフ作成と比較して高速である．また，探索のみに必要な時間は図 10 では 17ms，図 11 では 35ms となった．本手法は通行可能なエッジのみから成るグラフを作成済みであり，手法 1 のように探索時に衝突判定を行う手法と比較して高速な探索が可能である．

続いて 2000×2000 の大きさの地図を用いて経路の探索を行った．経路の探索結果を図 12 に，計算に要した時間を 4 に示す．環境が大きくなったため，手法 1 を用いた場合，探索には 49s と長い計算時間を必要とした．さらに手法 2 では通行可能なエッジのみから成るグラフを作成するのに 192s を要し，探索は 780ms となった．一方，本手法を用いた場合，エッジの選択は 3.43s で済み，また探索時間は手法 2 と同様に 780ms となった．

表 2: 実行時間の比較

手法	手法 1	手法 2	提案手法
衝突エッジ選択	—	14.6s	690ms
探索	1.67s	17.0ms	17.0ms
合計	1.67s	14.7s	707ms

表 3: 実行時間の比較

手法	手法 1	手法 2	提案手法
衝突エッジ選択	—	14.6s	270ms
探索	3.15s	35.0ms	35.0ms
合計	3.15s	14.6s	300ms

これらの実験により，本手法では，マップリンクを用いることにより，従来経路探索時に必要であった障害物との衝突判定を不要にし，経路探索の計算時間を，従来手法と比較して 10 倍程度高速化できることが確認できた．

7. おわりに

本稿では障害物との衝突判定を行うことなく高速に移動ロボットの経路探索を実現するためのマップリンク法を提案した．従来手法は，特に障害物の形状が変化する場合などに，探索時に障害物との衝突判定を行う必要があり，多くの計算時間がかかるという問題があった．本手法では空間内の各地点と，ロボットの移動経路候補との対応付けを行っておき，探索実行時に上記対応を参照することで，衝突判定を行うことなく経路探索が可能となる．本稿では，上記計算を行うための，タイルベースでの対応作成手法および，再帰的にタイルを作成することによる大規模空間に対応可能な手法を提案した．本手法によって，従来手法と比較して約 10 倍の高速化を実現した．

本手法は，地図上の各ピクセルに衝突するエッジを高速に決定する事ができる．そこで，経路上を動作中にも環境が動的に変化する場合にも，通行するエッジが通行不可能に変化するかどうかを高速に決定する事ができる．そこで，今後動的なグラフに対しての探索手法である $D*$[3] などを適用し，動的環境下での経路作成を目指す．

参考文献

[1] J. Barraquand, and J.-C. Latombe, “On Nonholonomic Mobile Robots and Optimal Maneuvering,” Proc. of Int. Conf. on Intelligent Control, pp.340–347, 1989.

表 4: 実行時間の比較

手法	手法 1	手法 2	提案手法
衝突エッジ選択	—	192s	3.43s
探索	49.6s	780ms	780ms
合計	49.6s	192s	4.21s

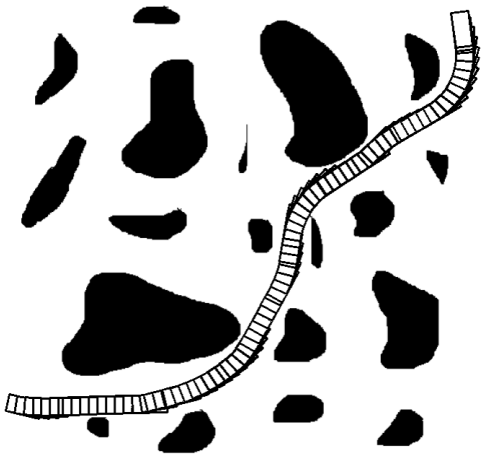

図 10: 環境 1

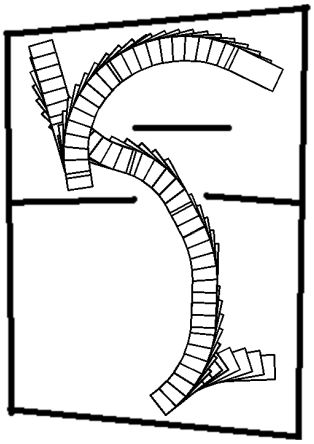

図 11: 環境 2

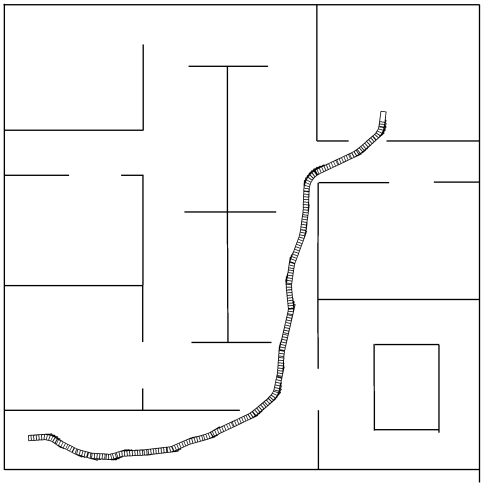

図 12: 環境 3

[2] L. E. Dubins, “On Curves of Minimal Length with a Constraint on Average Curvature and with Prescribed Initial and Terminal Positions and Tangents,” Amer. J. Math., Vol.79, pp.497–516, 1957.

[3] D. Ferguson and A. Stentz, “The Field D* Algorithm for Improved Path Planning and Replanning in Uniform and Non-Uniform Cost Environments,” Robotics Institute, CMU-RI-TR-05-19, 2005.

[4] P. E. Hart,N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” IEEE Trans. on Systems Science and Cybernetics, Vol.4, No.2, pp.100–107, 1968.

[5] J.-P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray, “A Motion Planner for Nonholonomic Mobile Robots,” IEEE Trans. on Robotics and Automation, Vol.10, No.5, pp.577–593, 1994.

[6] S. M. LaValle, and J. J. Kuffner Jr., “Randomized Kinodynamic Planning,” IEEE Int. Conf. on Robotics and Automation, Vol.1, pp.473–479, 1999.

[7] K. Macek, M. Becked, and R. Siegwart, “Motion Planning for Car-Like Vehicles in Dynamic Urban Scenarios,” IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp.4375–4380, 2006.

[8] J. A. Reeds, and R. A. Shepp, “Optimal Paths for a Car That Goes both Forward and Backwards,” Pacijic J. Math., Vol.145, No.2, pp.367–393, 1990.

[9] G. Song, and N. M. Amato, “Randomized Motion Planning for Car-like Robots with C-PRM,” Proc. of Int. Conf. on Intelligent Robots and Systems, pp.37–42, 2001.

[10] T.Wong, W. Luk, P. Heng, “Sampling with Hammersley and Halton Points,” Journal of Graphics Tools, Vol. 2, No. 2, pp. 9–24, 1997.