学術・技術論文

経路の変形と再探索を併用したオンライン動作再計画

吉田英一[*1] 金広文男[*1*2] 横井一仁[*1*2] Pierre Gergondet[*1]

Online motion planning using path deformation and replanning

Eiichi YOSHIDA[*1], Fumio KANEHIRO[*1*2], Kazuhito YOKOI[*1*2] *and* Pierre Gergondet[*1]

We present a reactive method for online robot motion replanning in dynamically changing environments by combining path replanning and deformation. Path deformation is integrated in a replanning method featured by parallel planning and execution. The proposed reactive planner can handle dynamic environments including continuously moving obstacles by smoothly deforming the path during execution. If the collisions cannot be removed by path deformation, alternative paths can be replanned efficiently by using continuously updated roadmaps. Simulation results are shown to validate the effectiveness of the proposed method.

Key Words: Motion planning, Online replanning, Path deformation, Collision Avoidance

1. はじめに

サンプリング運動計画手法は，計算機能力の発達に従い，ここ 10 年間急速な発展を遂げている [1] [2]．RRT (Rapidly-exploring Random Tree) や PRM (Probabilistic RoadMap) に代表されるこの手法では，コンフィグレーション空間（C-空間）でランダムにサンプリングし，これらをノードとするロードマップと呼ばれる自由経路のネットワークを構成することにより，グラフ探索を行って初期位置から目的位置への経路を導出することを基本とする．

既知で静的な環境やオフラインの計画を前提としている点が実際のシステムへの応用の障害であったが，最近ではこれらの手法を，動的に変化する環境でオンラインで実際のロボットに適用する事例も報告されるようになってきている．

最初のアプローチとしてあげられるのは，環境の変化があった場合に「再計画」を行う手法である．Leven らは，障害物との干渉がないロードマップをセル分割された作業空間にマッピングし，環境変化が生じた場合にこれを部分的に修正して再計画する方法を提案した [3]．Ferguson らは，RRT において環境変化により有効でなくなった木の部分を更新することにより，動的な環境で再計画を行う手法 [4]，また任意の時間における問い合わせに対して，改善が保証された解を返す “anytime” RRT 計画法 [5] を提案した．Jaillet らは，切り替え可能な複数の経路を計算しておき，ドアの開閉などにより環境が変化した場合に再計画を行う手法を提案している [6]．RRT に基づくオンラインの経路再計画手法としては，サンプリングへのバイアス付加，距離計算のヒューリスティック，遅延した干渉検出などを用いて都市環境での車両運動計画に用いた例 [7] や，視覚により Voxel で表現された環境においてロードマップを更新して再計画を行う手法 [8] も報告されている．筆者らも，動作計画と軌道の実行を並列に行うことにより，環境が変化した場合にオンラインで再計画を行う方法を提案した [9]．

これらの「再計画」法は，ロードマップを更新し，異なるホモトピー（トポロジーを変えずに連続的に変形可能な経路の集合）に属する別の大域的経路を探索するもので，突然出現した障害物や，離散的あるいは低速で移動する障害物が存在する環境では有効である．しかしながら，計算機の効率が向上したとしても，再計画が終了する前に障害物がロボットとの安全距離以下に入ってきたりした場合には，ロボットの停止が頻繁に起こる可能性がある．計算量を要する再計画への問い合わせが繰り返される点で，この「再計画」法は，連続的に移動する障害物のオンラインの対処には一般的には向かない．

これらに対し，第 2 のアプローチとして，ロードマップ上の探索をせずに，経路を局所的に修正することで，オンラインで経路を適応させる経路の「変形」がある．Fraichard らは，車輪型ロボットへのリアクティブなナビゲーション法を提案した [10]．

原稿受付
[*1] 独立行政法人 産業技術総合研究所 知能システム研究部門 AIST-CNRS ロボット工学連携研究体
[*2] 独立行政法人 産業技術総合研究所 知能システム研究部門 ヒューマノイド研究グループ
[*1] CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/CRT, National Institute of Advanced Industrial Science and Technology (AIST)
[*2] Humanoid Research Group, AIST

「Elastic band」法もこのような経路変形法の一つであり，移動障害物に対して，一旦計算された経路を，エネルギーを最小化するように適応的に修正する [11] [12]．筆者らも，ロボットの動的な運動を実現する経路を計画し，障害物との干渉回避のために変形する手法を提案した [13] が，これはオフライン計画である．これらの経路変形においては，経路のホモトピーの大きな変化は仮定されておらず，環境変化により実行中の経路が実行不能になった場合には適用は難しい．

以上の背景から，さまざまな環境変化に適応するため，オンラインで経路を実行しつつ，局所的な経路の変形と大域的な再計画の両方を統合する動作計画システムが望ましいと考えられるが，これに類する研究はこれまで少数の例が報告されているのみである．Brock らは，作業空間において遷移可能な経路を表現し，ノードの移動を許す “Elastic roadmap” を提案している [14]．サンプリング計画手法とは異なる可変の経路ネットワークの考え方を取っており，環境変化を反映するポテンシャル関数と，それにより駆動される制御器の適切な設計が必要となる．また，Xiao らは，複数の経路の集合を保持し，遺伝アルゴリズム的な経路の更新を適用して，適合値により評価した経路間で切り替えを行うことで動的な環境に適応して経路計画と実行を行う “Real-Time Adaptive Motion Planning (RAMP)” を提案している [15]．しかしながら，計画周期より長い制御周期を想定しており，また場合によっては大幅な経路の切り替えが必要となる場合も考えられ，局所的な環境変化に対する応答性が低下する可能性がある．

本研究では，経路の再計画と変形の効率的な統合により，従来研究では困難であった，ロードマップを用いた再計画手法による大域的な経路計画能力と経路変形の高い応答性の両方の利点を活用できるオンライン経路再計画手法について述べる．大域的な計画手法のみによる不必要な探索や頻繁な経路の切り替え，また逆に，局所的な変形のみでは解決できない環境の大幅な変化など，移動障害物が存在する環境におけるこれまでの手法の問題点を解決し，状況に応じて適切な手法を適用して滑らかな干渉回避を行う枠組みが，本研究における主要な新規提案である．具体的には，干渉のない経路を計画し，その実行を開始した後，障害物が接近してきた場合にはまず経路変形を適用する．変形した経路が実行可能な限りは，計算負荷の高い再計画なしに連続的に障害物を 回避することができる．障害物が経路をふさぐことなどにより，実行中の経路が実行不可能となった場合には，それまでに蓄積した環境情報を効率的に利用できる 2 種類の学習・作業ロードマップを用いた再計画を行って，干渉のない経路を再探索する．経路変形の計算は，干渉回避方向をロボットのヤコビ行列の逆行列を用いずに求めることで高速に行うことができる．ロードマップを用いる経路の再計画は必要がない限り呼び出されないため，ロードマップもコンパクトに維持することができ，再計画の一層の効率化にも有効である．

本稿では，2 章でオンライン経路再計画システムの概要について述べた後，3 章，4 章でそれぞれロードマップの再利用，経路変形による再計画手法を導入する．提案する再計画システムの有効性を 5 章で検証した後，6 で手法の制約や適用範囲等の課題について考察し，7 章で結論を述べる．

2. オンライン経路再計画システムの構成

2.1 経路計画・実行を並列に行う再計画システムの概要

動作中に環境の変化が検出された場合に，オンラインで適応的に再計画を行う機能を実現する計画器には，以下の仕様が要求される．

- 計画された経路の実行中に障害物との干渉が予測される場合，ただちに再計画を開始し，再び干渉のない経路が得られたらその実行に移る．
- 経路の再計画中は，設定した安全距離以下に障害物に接近しない限り，動作を継続する．安全距離以下に接近したときに再計画が終了していなければ動作を中止する．
- また，再計画中に予測される干渉が排除された場合には再計画を中止し，元の経路を継続することができる．

2.2 計画器の構成

Fig. 1 に，上記の仕様を満たす再計画機能を実現する計画器の状態遷移図を示す．これは全体として一つの枠組みであるが，ロボットが動作している状態で計画を行うことを想定し，スタートした後，Planning と Execution の 2 つの並列なスレッドに分かれて動作することが特徴である．

Fig. 1 で，囲い文字は状態を表す．状態遷移は，1) 「信号」の受け取り（Fig. 1 実線），2) 内部処理の結果（Fig. 1 点線）によって生じる．状態遷移を生じさせる信号，内部処理を，それぞれ矢印付近のイタリック，下線のテキストで示す．信号の発信は，網掛け文字で示す．

信号には，計画器内部でスレッド間で送受信するものと，外部から受信するものを設定する．信号 “Query”, “Finished”, “Path found” は，内部で送受信する．外部信号の “Start”, “Canceled” はユーザから，“Geo. change” は Fig. 2 で想定するセンサ情報を管理する機構から送信される（Fig. 1 には発信は示されて

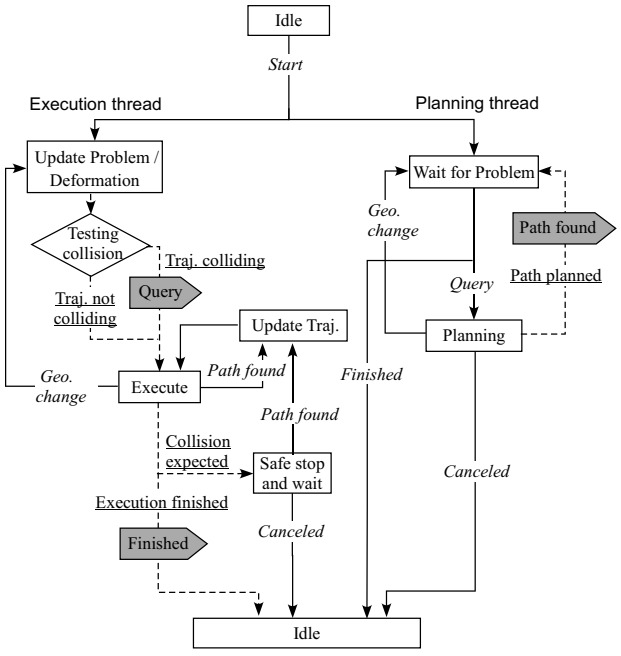


Fig. 1 State transition diagram of replanning.

いない）．環境変化に関しては，アーム装着カメラや距離センサなどの本体搭載センサ，あるいは環境に装備されたカメラなどの外部センサを利用して獲得できると仮定する．

Execution スレッドは，経路の実行中は “Execute” 状態であり，環境変化により “Geo. change” 信号を受け取ると，動作を停止せずに “Update problem” 状態に移行し，再計画が必要かどうかを確認する．経路と障害物との干渉が検出されなければ，即座に “Execute” 状態に戻り経路の実行を継続する．干渉が予測される場合には，Execution スレッドは目標位置までの残りの経路から，障害物からの安全距離で減速・停止する安全停止経路を導出し，これを実行する．その後，Planning スレッドに 再計画を依頼する Query 信号を送る．

Planning スレッドによる経路計画が成功すれば “Path found” 信号を受信し，Execution スレッドは “Update Traj.” に移行して軌道を更新した後，経路実行中であれば停止せず，すぐに新しい経路の実行に移行する．もし安全停止経路の実行が終了する前に再計画が終了しない場合には，その終点で停止し，Planning スレッドが新たに干渉のない経路を導出するまで，“Safe stop and wait” 状態に移行して待機する．安全停止経路の終端に達するまでの間は実行を継続することで，再計画中もロボットの動作を不必要に中断させることはない．これにより，もし障害物が排除された場合は Geo. change 信号で再計画を中止して元の軌道を続行できる．計画時間の上限を設けて，それまでに計画が終了しなければ，ユーザが “Canceled” 信号を各スレッドに送って再計画を中止することも可能である．

Planning スレッドでは，最初 “Wait for Problem” の状態で待機している．Execution スレッドより， “Query” 信号を受け取ると再計画を開始して “Planning” 状態に移行する．計画が成功した場合，計画中に “Canceled” あるいは “Geo. change” 信号を受け取った場合には計画器の実行を終了する．“Canceled” により再計画自体の終了が要求された場合，また目標位置に到達して Execution による経路実行が終了した場合を除き，終了後は “Wait for Problem” 状態に戻って，新たな計画問題の解決を要求されるまで待機する．

再計画の際には，3 章で示すように動作計画中に逐次的に更新される 2 種類の学習・作業ロードマップを導入し，それまでに行った環境探索の知識を有効に利用する．

2.3 連続的経路変形の導入

前節に示した再計画の枠組みにより，ロボットの動作中に突然障害物の移動があったり，センサの測定可能範囲外にあった障害物に近づいて検出されたりした場合でも，本章冒頭の要求仕様を満たす再計画を行うことができる．しかしながら，連続して移動する障害物に対しては，微小な変化に対しても毎回ロードマップを利用した再計画処理が呼び出されることとなり，計算時間の点で非効率的となる場合が多い．そこで本研究では，そのような場合に再計画を行わず，その経路が属するホモトピーを変更せずに，局所的な経路変形を適用する仕組みを併用することで，全体として効率的な再計画を行うことを目指す．

経路変形は，Fig. 1 に示した構成の計画器に容易に統合することができる．再計画は行わないので，Execution スレッドにこの機能を追加する．ロボットが経路を実行中であれば，環境変化により “Geo. change” 信号が受け取られた後， Execution スレッドは “Update Problem” 状態に移行し，4 章で示す変形操作を適用することにより経路が改善されるかを調べる (Fig. 1)．もし変形された経路が障害物と干渉していなければ，現在実行中の経路を変形した経路で置き換え，“Execution” 状態に戻ってそれまでと同様に経路の実行を継続する．後ほど 4.3 節で述べるように，経路の改善度は指定した計算時間内で経路長をどれだけ短縮できるかによって評価する．したがって，変形操作が成功している間には，ロードマップによる経路探索は行われない．もし変形操作が失敗した場合には，更新された 作業ロードマップを用いて Planning スレッドによる経路再計画が行われる．

2.4 ロボット制御器とのインタフェース

提案する再計画システムでは，運動計画と軌道実行を並列に行うことから，ロボットの制御器とのインタフェースの定義も重要となる．ロボット制御器の機能は，運動計画部分で計画した経路を関節角など制御指令に変換し，ロボットのシミュレータや実機の制御を行うことである．さまざまなロボットの制御器との接続を可能とするため，インタフェースの定義をできるだけ一般的に行うことが望ましい．

再計画システムとロボット制御器の一般的なインタフェースとして，Table 1 のように設定した．軌道の実行と停止，現在の実行状況（実行中かどうかの確認と現在のコンフィグレーション取得）などの基本的な機能を定義しており，これらは状況に応じた再計画と実行軌道の変更に必要となる．また，移動距離，環境変化の検出機能は，外部・内部の知覚システムとのインタフェースであり，それぞれ衝突位置の計算，再計画機能開始のトリガとなる．Fig.2 には典型的なロボットシステムにおける，これらのインタフェースを解したロボットの制御器やセンサシステムとの接続関係を示す．

3. ロードマップ再利用による再計画

3.1 継続的なロードマップの更新

リアクティブな再計画のためには，環境の動的な変化を反映するためにロードマップを効率的に管理する必要がある．従来研究においても，ロードマップをオンラインで更新する手法が提案されている [3] [5] [7] [16].

障害物が移動したことによりロードマップ内のノードやエッジを消去しようとすると，ロードマップの接続性の管理の際に煩雑な処理が必要となる．また，毎回新たにロードマップをゼロから作り直すと，それまでの探索結果を活用できないうえ，再

Table 1 Planning interface with robot controller

`bool execute(Path)`	Execute the path
`void stop()`	Stop the execution safely with deceleration
`bool isMoving()`	Return true if moving
`config getCrntConfig()`	Get the current config.
`double getCrntDist()`	Get the traveled distance on the path
`bool getGeoChange()`	Return true in case of environmental changes

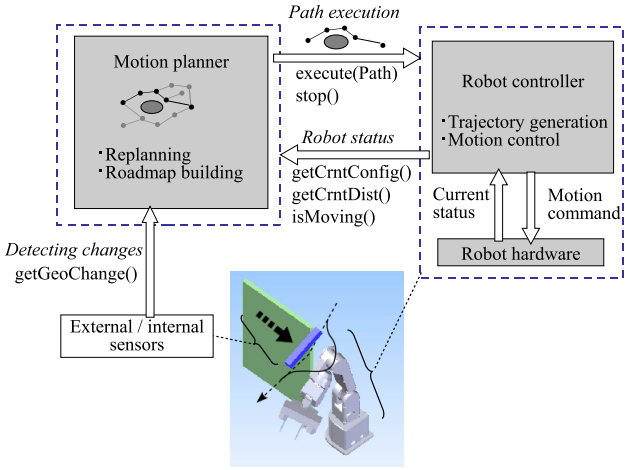


Fig. 2 The functionality of the interface defined in Table 1 in the robotic systems including motion controller.

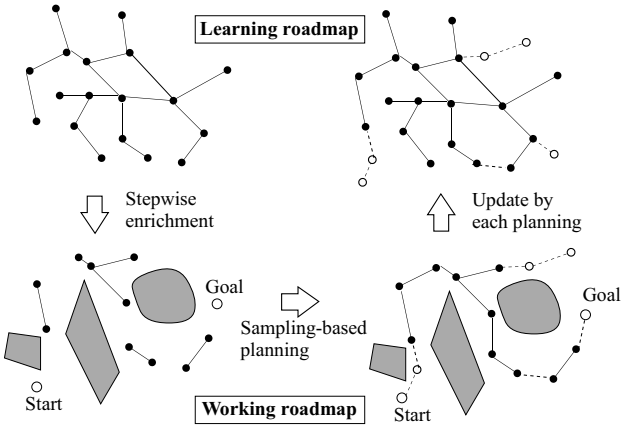


Fig. 3 Learning and working roadmaps.

探索に時間も必要となる．そこで，経験を蓄積する学習ロードマップと，変化した環境でその都度使用する作業ロードマップ２種類のロードマップを導入して，再計画の効率化を図る．

学習ロードマップは，計画・実行期間全てを通じて更新する．これに対し，作業ロードマップは，環境の変化により再計画が必要となった際に，サンプリング計画手法による目的位置・姿勢までの経路探索に使用するもので，再計画ごとの探索開始時にクリアして初期化する．ここで通常のサンプリング計画手法と異なるのは，学習ロードマップの有効な部分を作業ロードマップで利用することで，すでに実行可能な部分経路を再度探索しなくてもよくなり，効率化が期待できるという点である．また，環境情報の更新を C-空間において２種類のロードマップを用いて行う点で，作業空間において大域的ロードマップのノード位置の移動により環境変化を反映する Elastic Roadmap [14] とはアプローチが異なる．

PRM や RRT といったサンプリング計画手法では，コンフィグレーション空間でサンプリングした結果を用いてロードマップを成長させ，経路探索を進めていく．したがって，動作計画は干渉のない姿勢と局所経路に対応するノードとエッジを逐次的にロードマップに追加していくという，単位処理の繰り返しととらえることができる．

Fig. 3 に，２種類のロードマップの機能を示す．図の上部が学習ロードマップ，下部が作業ロードマップである．

環境が変化が検出された場合，まず作業ロードマップをクリアし，目標位置までの経路計画のためのサンプリング動作計画を開始する．動作計画の単位処理を行う際に，学習ロードマップから，干渉のない局所経路に相当する m 本のエッジ（実線）をランダムに選択し，作業ロードマップに追加する．これが，Fig. 3 左側の “Stepwise enrichment” に相当する処理である．これにより，再計画時には，有効なエッジのみを学習ロードマップから加えることで，作業ロードマップをコンパクトに保ちつつ，これに環境の最新状態を反映させることが可能となる．サンプリング計画において単位処理は繰り返し呼ばれるので，m は少ない数で良く，以下では $m = 1$ とする．

反対に，再計画の過程で作業ロードマップに新たに追加されたエッジ（点線）は，学習ロードマップに追加され，これまでの探索経験として蓄積して次回以降の再計画に活用する．この処理は，Fig. 3 右側の “Update by each planning” に示される部分である．

ここでのロードマップ処理では，ロードマップ内に複数の連結要素が含まれることを許容する．すなわち，探索が進むにしたがって将来的に結合される可能性を考慮し，互いに接続していない局所経路のグラフがロードマップ内に複数存在しても良い．また，サンプリング計画手法としては，RRT, PRM など任意のアルゴリズムを適用可能である．

以上により，計画においては，作業ロードマップをコンパクトに保ちつつ，これに環境の最新状態を反映させることが可能となる．

3.2 ロードマップを用いたオンライン再計画

２章で見たように，提案する再計画システムでは，ロボットの経路実行中にも動作計画は並列して行われており，経路が見つかるとすぐにそれを実行する．本節では，継続的に更新されるロードマップを用いた再計画手法について述べる．Fig. 4 にその手法を示す．

環境変化が検出されて Execution スレッドが “Geo. change” 信号を受け取ると，干渉が生じる経路 $P(s)$ 上の点 s_2 と，安全な停止動作を開始する点 s_1 を推定する (Fig. 4a)．次に，経由点 $P(s_2)$ から現在の目標位置までの干渉のない経路を，3.1 節で示したように，学習ロードマップを利用して探索を効率化した作業ロードマップに基づいて再計画する (Fig. 4b)．

経路上の s_1 に到達する前に再計画が終了し，干渉のない新たな経路 $P'(s)$ が導出されれば，Execution スレッドは “Path found” 信号を Planning スレッドから受け取り，“Update Traj.” 状態に移行する．さらに，Execution スレッドは $P(s)$ 上で s_1 より前の点と，$P'(s)$ 上の別の点を結ぶ短い局所経路を $P'(s)$ に追加する．現在実行中の経路から新しく計画した経路にスムーズに移行するため，経路平滑化（例えば，Adaptive Shortcut 法 [17] など）を，再計画された経路全体に適用する (Fig. 4c)．平滑化が終了した後，Execution スレッドはすぐに再計画された経路の実行を開始する (Fig. 4d)．

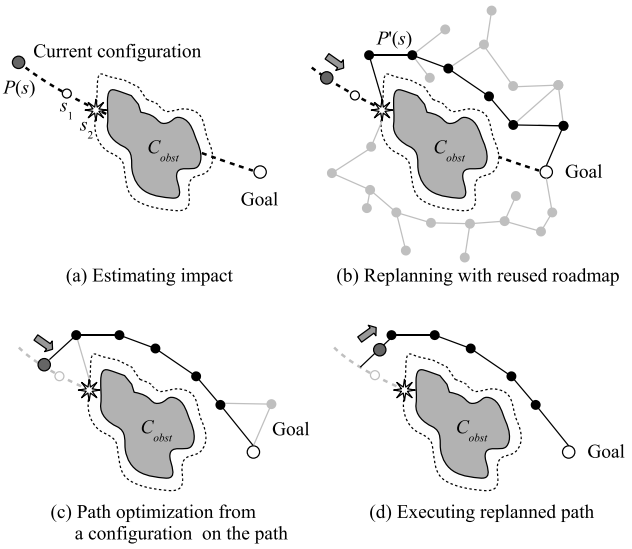


Fig. 4 Replanning the path with the anticipated collision.

再計画が s_1 に到達する前に終了しない場合は，ロボットは s_2 で動作を停止し，再計画により新たな経路が求められるまで，経路実行を中止する．計画に対して制限時間が設定されている場合には，一旦再計画を中止し，別途計画が要求されるまで待機する．

このようにして，継続的に更新されるロードマップを用いて，それまでに蓄積された環境に関する知識を利用して効率的に経路の再計画を行うことができる．

4. 経路変形による再計画

4.1 連続的経路変形

前節までに示した枠組みでは，再計画時に必ずロードマップを用いた経路の再探索が行われていた．しかしながら，経路と障害物の干渉が軽微かつ連続的な場合，ロードマップ更新とグラフ探索の頻度が高くなり，計算時間の点で非効率的となる．

特に，障害物が連続的に動くような場合には，最悪の場合，環境変化が検出されるたびに再計画処理が呼び出されことになる．この際，ロボットが障害物と設定された安全距離以下に近づく前に再計画が終了しない場合には，頻繁に停止する可能性がある．

この問題を解決するため，ロードマップを用いた経路再探索を行うのではなく，経路のトポロジーは変更せずに経由点を動かして連続的に変形することにより，再計画をより効率的に行うことが考えられる．そこで，Elastic band [11] [12] を代表的な例とするこの経路変形を導入することとする．

4.2 変形方向の計算

Fig. 5 に示すように,「変形開始距離」d_i を導入し，ロボットと障害物の最短距離が d_i 以下になったとき，軌道変形を開始する．ロボット・障害物上の最近接点をそれぞれ P, Q とし，これらを結ぶベクトル $\overrightarrow{QP}$ は，PQ 間の距離 d と単位ベクトル $\boldsymbol{n}$ を用いて $\overrightarrow{QP} = d\boldsymbol{n}$ と表すことができる．d が d_i よりも大きくなる方向に点 P を移動させるため，微小変位 ΔP は

$$\Delta P \cdot \boldsymbol{n} \geq d_i - d.$$

を満たす必要がある．C-空間における微小変位 $\Delta\boldsymbol{q}$ と ΔP は，点 P における ヤコビ行列 $\boldsymbol{J}_P$ を用いて以下のように関係付けられる．

$$\Delta P = \boldsymbol{J}_P \Delta\boldsymbol{q}.$$

したがって，以下が成り立つ．

$$\boldsymbol{J}_P \Delta\boldsymbol{q} \cdot \boldsymbol{n} = \Delta\boldsymbol{q} \cdot \boldsymbol{J}_P^T \boldsymbol{n} \geq d_i - d$$

ロボットの移動距離の下限値 $d_i - d$ を用い，$\Delta\boldsymbol{q}$ のノルム最小解は，$\boldsymbol{J}_P$ の擬似逆行列を用いずに以下のように求められる．

$$\Delta\boldsymbol{q} = (d_i - d)\frac{\boldsymbol{J}_P^T \boldsymbol{n}}{||\boldsymbol{J}_P^T \boldsymbol{n}||^2} \tag{1}$$

ロボットが $\Delta\boldsymbol{q}$ の方向に $||\Delta\boldsymbol{q}||$ よりも大きな移動を行えば，変形開始距離 d_i よりも距離が大きくなる安全な方向に移動することができる．

この経路変形は，式 (1) に示す微小移動方向への移動が実現可能なロボットに適用できる．非ホロノミック制約等によりこれが直接実現できない場合には，目標回避方向に近づく移動を導出するとともに，制限された移動可能範囲を考慮して，変形開始距離 d_i を大きく取るなどの対処が必要となる．

4.3 変形アルゴリズム

Fig. 6 に変形アルゴリズムを示す．実行される経路は，C-空間上の経由点を局所移動法により接続した局所経路列として表される．関数 Deform(*path*, timeLimit, improveThresh) は経路中の経由点を順番にたどり，i 番目の経由点 $\boldsymbol{q}_i$ の前後の局所経路 LP_1 と LP_2 に変形操作を適用する．この操作は，経由点 $\boldsymbol{q}_{i-1}$ と $\boldsymbol{q}_{i+1}$ を直接結ぶ局所経路の方向に点 $\boldsymbol{q}_i$ を移動させる関数 tighten() により行われる．もしロボットの周囲に障害物が存在しなければ，この関数は Fig. 7 に示す $\boldsymbol{q}_{i-1}$，$\boldsymbol{q}_{i+1}$ の短絡経路上を LP_1 と LP_2 の経路長の比で内分した点 $\boldsymbol{q}_0$ を返し，LP_1，LP_2 はそれぞれ $\boldsymbol{q}_{i-1}$ と $\boldsymbol{q}_0$，$\boldsymbol{q}_0$ と $\boldsymbol{q}_{i+1}$ を結ぶ局所経路 LP_1^{new} and LP_2^{new} で置き換えられることになる．

これに対し，もし最近接の障害物が変形開始距離内にある場合には，tighten() は Fig. 5 で求めた $\boldsymbol{n}$ 方向に $\boldsymbol{q}_0$ を障害物からの距離 d_i にある変形開始境界上の超平面上に射影した $\boldsymbol{q}_{\text{new}}$ を返す（Fig. 7）．もし $\boldsymbol{q}_{i-1}$ と $\boldsymbol{q}_{\text{new}}$，また $\boldsymbol{q}_{\text{new}}$ と $\boldsymbol{q}_{i+1}$ をそれぞれ結ぶ局所経路 LP_2^{new} と LP_1^{new} の両方が障害物と干渉しない場合，関数 Deform() は局所経路 LP_1 と LP_2 をそれぞ

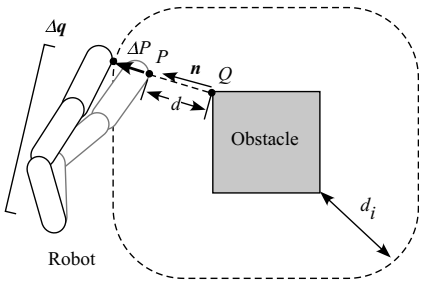


Fig. 5 Direction of path deformation through small displacement for obstacle avoidance

Deform(*path*, timeLimit, improveThresh)

```
1: timeout ← false, changed ← false
2: while !timeout AND !changed do
3:   lenOld ← path.length()
4:   for i=1 to path.numConfigs() do
5:     LP1 ← localPath(q_{i-1}, q_i)
6:     LP2 ← localPath(q_i, q_{i+1})
7:     r ← LP1.length() / (LP1.length() + LP2.length())
8:     LP← localPath(q_{i-1}, q_{i+1})
9:     q0 ← LP.getConfigAt(r)
10:    qnew ← tighten(q_i, q0)
11:    // local path is "tightened" towards q0
12:    newSeg ← ∅
13:    LP1^new ← localPath(q_{i-1}, qnew)
14:    LP2^new ← localPath(qnew, q_{i-1})
15:    if LP1^new.valid() AND LP2^new.valid() then
16:      newSeg ← makePath(q_{i-1}, qnew, q_{i+1})
17:      path.replace(i-1, i+1, newSeg)
18:      changed ← true
19:    else
20:      if LP1.valid() then
21:        newSeg.add(LP1)
22:      else
23:        // cut the local path into two
24:        qhalf ← LP1.getConfigAt(0.5)
25:        newSeg.newPath(q_{i-1}, qhalf, q_i)
26:      end if
27:      // do the same thing for LP2 and update newSeg
28:      //...
29:      path.replace(i-1, i+1, newSeg)
30:      changed ← true
31:    end if
32:    timeout ← checkOutOfTime(timeLimit)
33:  end for
34:  if changed = true then
35:    if (lenOld - path.length()) / path.length() < improveThresh then
36:      changed ← false
37:    end if
38:  end if
39: end while
```

Fig. 6 Path deformation procedure

れ LP_1^{new} と LP_2^{new} で置き換える．逆に，新しく生成された 2 つの局所経路 LP_n^{new} (n=1, 2) のうち一つでも障害物と干渉する場合には，より細分化した経路への変形が必要と考えられるので，もとの局所経路 LP_n を中点で分割して，次の **for** ループでの処理の際にこれまでと同様の tighten() 操作を適用する．

関数 Deform(*path*, timeLimit, improveThresh) は，計算時間 “timeLimit” と変形された経路の長さによって評価される改善率と “improveThresh” の制限が満たされる限り繰り返される．Fig. 6 に示す処理は，“improveThresh” 以上の改善が見られなくなった時点で終了する．

この変形操作は経路実行時に適用されることから，2.3 節で述べたように Fig. 1 の Execution スレッドに統合する．環境変化が検出されて “Update Problem” 状態に移るときに経路変形が適用される．障害物との干渉がない場合も含め，経路変形が成功した場合には，現在実行中の経路が変形されたものに置き換えられる．成功しなかった場合には，Planning スレッドによる経路の再計画を適用する．このようにして，経路変形を自然な形で経路計画・実行を並列で行う枠組みに統合することができる．

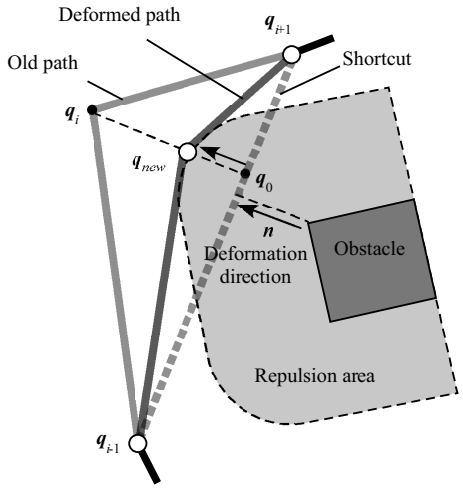


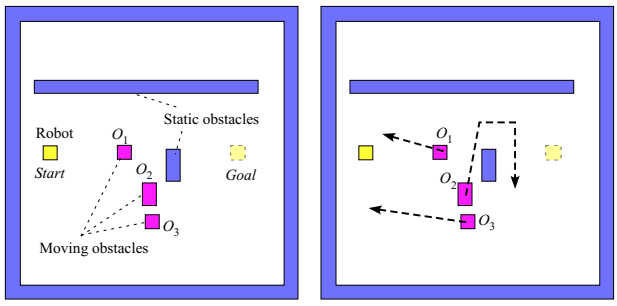
Fig. 7 Local deformation of the path by the function Deform(). The local paths connecting the configurations q_{i-1}, q_i, q_{i+1} is deformed using q_{new} that is the projection of the shortcut configuration q_0 onto the hyperplane representing the boundary of repulsion area.

5. 計画シミュレーション

提案する再計画手法を，連続して移動する障害物が複数存在し，再計画だけでは取り扱いが困難な環境に適用した．

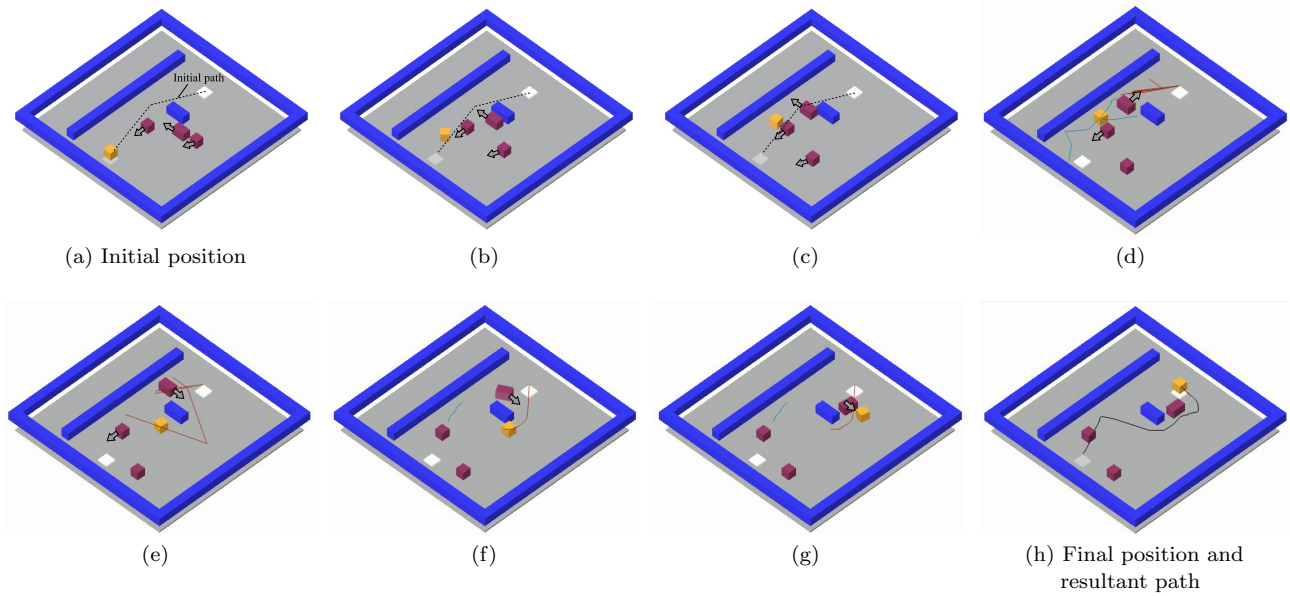
Fig. 8 は，異なる速度で連続的に移動する障害物 $O_1 \sim O_3$ と静止障害物が存在する平面上の環境を示しており，ロボットは初期位置（Start）から目標位置（Goal）に移動する．ロボットの制御器としては，Table 1 のインタフェースに従い，定められた最大速度内で与えられた軌道を追従する簡易的な線形フィードバック制御器を用いる．また，障害物は Fig. 8b に示す軌道を，ロボットの最大速度以下の速度で連続的に動くとする．

提案する計画手法による計画結果を Fig. 9 に示す．まず，ロボットは最初に計画された干渉のない目標位置までの経路の実行を開始する．障害物 O_1 がロボットに向かって移動するにしたがって，当初計画された経路が変形されており，変形が可能



(a) Simulation environment (b) Movement of obstacles

Fig. 8 Planning with continuously moving obstacles



(a) Initial position (b) (c) (d)

(e) (f) (g) (h) Final position and resultant path

Fig. 9 Results of reactive motion planning. The initial path is deformed at (b-d). Alternative path is searched when obstacle fills the gap (d) and is executed (e-g) which is also deformed at (g). The resultant path is shown in (h).

な限りはロードマップを用いた再計画を行わずに，同じホモトピーによる経路が実行されている (Fig. 9b-c)．その後，障害物 O_3 がロボットが当初通過しようとしていた経路をふさぐ形で移動してきたことにより，経路変形では対処できない状況となるため，別のホモトピーの経路を逐次的に更新されるロードマップを用いて再計画する必要性が生じる (Fig. 9d)．Fig. 9e-f からわかるように，新しい経路が再計画された後，すぐにその経路の実行に移り，ロボットは移動方向を変えている．さらに障害物 O_3 は進路を変更し，回転しながら目標位置付近で再びロボットに接近する．ここでも，目標位置に到達する前に衝突を回避するため経路の変形が行われている (Fig. 9g)．ここでは，経路の改善率の閾値と制限時間（Fig. 6 の improveThresh, timeLimit）は，1% と 0.1s に設定した．

同様の障害物環境で，経路変形なしの再計画についても検証を行った．乱数の種を変えて 10 回計画シミュレーションを行った結果，経路変形を行わない再計画では，10 回中 7 回でロボットと障害物の干渉が生じて目標位置に到達できなかったのに対し，経路変形を併用した場合にはすべての場合で目標位置まで到達した．失敗の理由としては，環境変化が各サンプリングごとに検出されてロードマップを用いた再計画が行われるため，ロードマップ再利用による効率的な手法を用いても次の呼び出しまでに再計画が終了しない場合があることがあげられる．このため，再計画中に障害物が接近し，これを避ける新しい経路の計画が終了し実行される前に，障害物との干渉が生じてしまっていた．

さらに，経路変形の導入により，成功率に加えて以下のように計画の面でも効率が改善されることが期待される．学習ロードマップの大きさ（ノード数，エッジ数）は，環境変化の際に Planning スレッドに対して再計画が呼び出されたときに，新たなに探索された部分が追加されるため増加する (Fig. 3)．これに対し，特に連続的に移動する障害物が存在する環境で，Execution スレッドにおいて経路変形を継続的に適用して再計画なしに干渉回避を行うことで，ロードマップを用いた探索の呼び出しが減り，その大きさの増大を抑えることができると考えられる．Fig. 10 は，計画が成功した場合の学習ロードマップのノード数を，経路変形を併用した場合としない場合で比較したものである．横軸は，計画・実行を開始してからの経過時間である．図からわかるように，経路変形を併用することでロードマップの大きさは約 1/4 に減少しており，再計画における探索空間の減少に役立っている．

Fig. 11 は通算の計算時間の比較である．予想されるように，経路変形を用いた場合には，再計画に費やした時間（Planning in Deform.）は低い値になっている．これに対し，経路変形自体の計算時間（Deform.）は，経路変形を併用しない場合の通算の計画時間（Replanning only）と同等の値となっている．図から，20s から 60s の間，再計画が行われていない間にも経路変形の計算時間は増加していることが観察される．これは，経路変形が Execution スレッドに統合され，連続して移動する障害物の微小変異が検出されるたびに適用されるからである．Fig. 1 で提案する枠組みでは，障害物がロボットに接近していなくても，実行予定の残りの経路に生じる可能性のある障害物との干渉に対して予備的に経路変形処理を適用する．結果的には，合計の計算時間（Deform. + Planning in Deform.）は経路変形を用いない場合（Replanning only）と同程度となるが，目標位置到達成功率が大きく改善されたことを考慮すると，これは許容できる範囲といえる．

別の例として，冗長マニピュレータに提案手法を適用した例を Fig. 12 に示す．7 自由度マニピュレータは，Fig. 12a の初期位置から，Fig. 12h に示す目標位置に，静止・移動障害物のある環境に移動する動作を計画する．まず，Fig. 12b, c で，マ

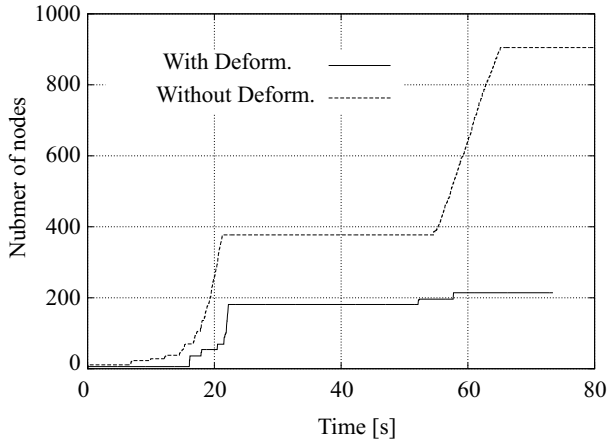


Fig. 10 Transition of learning roadmap size during execution with and without deformation

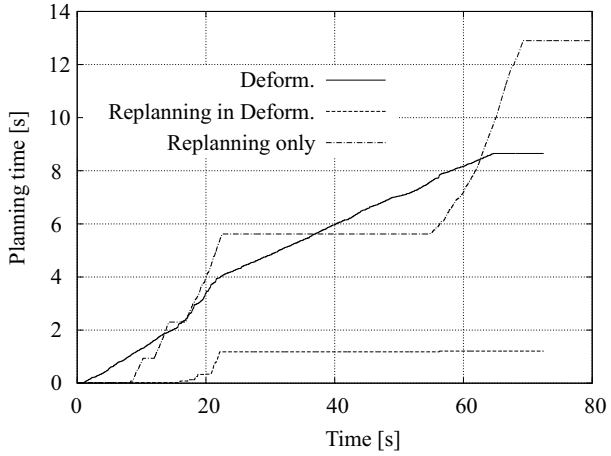


Fig. 11 Cumulative planning time with and without deformation. “Deform.” and “Replanning in Deform.” mean computation time for deformation and replanning with the proposed planner. “Replanning only” means the planner without deformation.

ニピュレータが奥の２つの板状の静止障害物の間から手先を上方に動かす経路を実行するときに，立方体の障害物が手先に接近し，経路が奥行き方向に変形されている．その後，マニピュレータは手前の２つの板状の障害物に挟まれた下方の目標位置に向かって動作を継続する（Fig. 12d）．今度は別の障害物が接近し，経路変形のみでは対応しきれなくなる（Fig. 12e）ため，ロードマップを用いた再計画を行って，板上障害物の外側を回り込んで目標位置に到達する経路が再計画される（Fig. 12f-h）．

以上のシミュレーションで，提案手法により，ロボットは環境の変化に応じて経路の変形と再計画を使い分けながら適応的に経路計画を行っており，提案手法の有効性が示された．

6. 考 察 と 課 題

6.1 実時間計画・実行における制約

提案する計画手法では，実時間での経路の再計画を想定していることに起因する制約が生じる．Fig. 4 において s_1 は，ロボットが s_2 で停止するために，安全停止距離を考えて減速を開始する点である．再計画においては，s_1 に到達するまでの時間を，経路実行中に並列に行う計画に使うことができる．現実には考えにくいが，s_1 よりも近くに障害物が発見されるなどした場合には対応はできない．上記の制限時間内に計画を行って解を得るため，再計画アルゴリズムの一層の効率化も今後の課題である．現在は環境中で障害物が移動するたびに，将来の干渉に対して予防的に適用している経路変形操作を，予測される衝突位置が現在より遠い場合などには省略する，また学習ロードマップにおいて時間が経過した部分を破棄する，などの対策が考えられる．

また，移動障害物の速度を v_{obs} とすると，経路変形のための Fig. 6 の計算に必要な時間の上限 timeLimit の間に障害物は $v_{\mathrm{obs}} \cdot \mathrm{timeLimit}$ だけ移動する．したがって，経路変形においては，4.2 節における単位時間の関節移動の最大値 $\Delta \boldsymbol{q}_{\max}$ としたときの最近接点 P の最大変位 $\Delta P_{\max} = \boldsymbol{J}_P \Delta \boldsymbol{q}_{\max}$ の値が，安全率を $\alpha(>1)$ を考慮した $\alpha \cdot v_{\mathrm{obs}} \cdot \mathrm{timeLimit}$ よりも大きいことが干渉を回避できる条件となる．

提案する再計画手法では，障害物が新たに発見されるなどした場合に，まず s_1 に達するまでの時間で大域的に経路を探索する．また連続的に移動する障害物がある場合には，Fig. 5 に示す変形開始距離 d_i を s_1 の位置より遠くに取ることで，なるべく減速しないで回避が可能なように経路変形を行う．このようにして，移動障害物が以上の制約を満たす範囲で，よりスムーズに環境に適応することが期待される．

6.2 計画手法の適用範囲と改善点

サンプリング計画手法については，サンプル数が無限に近づくにしたがって，目的地までの自由経路が存在すれば，それが求まる確率が１に近づく確率的完全性（Probabilistic completeness）が保証されている [1] [2]．しかし，本研究で扱う動的な環境では，これは保障されない．特に厳しい条件では，複数の動的な障害物によってデッドロックが生じたり，障害物により環境の辺縁や特異点付近に追い込まれたりする状況も考えられる．

障害物の動きが完全に予測できない動的環境で，これらの状況を完全に回避することは困難であるが，提案する計画手法の対処能力を向上させる方策はいくつか考えられる．まず，大域的な計画においては，複数の障害物や特異点からなるべく遠くを通る経路を計画したり [18]，障害物の挙動が予測可能と仮定し，将来の動作予測に基づく計画を行ったりして [16] [19]，デッドロック状況を未然に回避することが考えられる．

また本手法では，ロボットが動作するために，経路の計画が行われていることを前提としている．ロボットが停止して経路計画を行っている間に障害物が近づいてきて衝突したりする状況を避けるため，より下位レベルの制御器にて，経路が計画されていなくても反射的に回避する [20] などの動作を導入することも改善策の一つである．

7. お わ り に

本稿では，経路の変形と再計画を含むリアクティブなロボット動作計画手法を提案した．障害物との距離に基づく局所的な経路変形手法を導入し，経路の実行と計画を並列に行う動作計画の枠組みに統合した．これにより，実行中の経路のホモトピーを

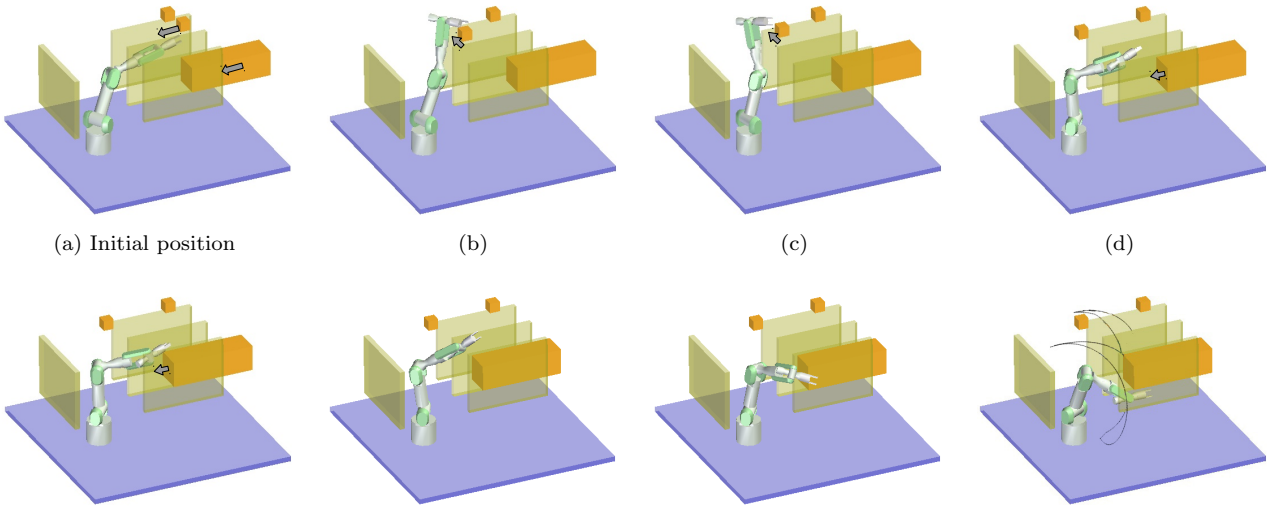


Fig. 12 Planning of a redundant manipulator. The deformation is performed with the small moving obstacle (b-c). When the planned path is obstructed (d-e), a new path is replanned (f-h). The resultant end-effector path is shown in (h).

変えずに局所的に変形させることにより，連続的に移動する障害物の回避を行うことができる．経路変形により予測される障害物との干渉が取り除けない場合には，継続に更新されるロードマップを再利用して，効率的な再計画を適用する．複数の障害物が移動する環境における計画シミュレーションを行い，提案手法により経路の変形と再計画が状況に応じて適用され，目標位置に到達する再計画が行われたことを確認し，その有効性を検証した．

今後は，6 章で議論した手法の効率や確実性の向上に加え，動力学シミュレーションや実際のロボットを用いた本手法の有効性の検証にも取り組んでいく予定である．また，本研究で扱う動的環境での動作計画とは異なるが，作業計画の分野では順序関係のあるロボット作業に対して，作業時間の不確実性を考慮してオンラインで再計画を行う手法も提案されている [21]．これらを本研究のオンライン動作再計画手法と組み合わせることで，より応答性の高い作業システムの構築も将来の展開の可能性として考えられる．

謝辞 本研究は，NEDO 次世代ロボット知能化技術開発プロジェクト「ロボット知能ソフトウェアプラットフォームの開発」，科学研究費補助金・基盤研究 B(21300078) の一部として実施された．本研究の手法の実装にあたり有益な助言をいただいた Kineo CAM 社の Ambroise Confetti 氏と Etienne Ferré 氏，また EU Vulcanus in Japan 奨学生として本研究での開発に携わった Clément Petit 氏に謝意を表する．

参 考 文 献

[1] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2006.
[2] S. LaValle, *Planning Algorithm*. Cambridge University Press, 2006.
[3] P. Leven and S. Hutchinson, “A framework for real-time path planning in changing environments,” *Int. J. of Robotics Research*, vol. 21, no. 12, pp. 999–1030, 2002.
[4] D. Ferguson, N. Kalra, and A. Stentz, “Replanning with RRTs,” in *Proc. 2006 IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 1243 – 1248.
[5] D. Ferguson and A. Stent, “Anytime RRTs,” in *Proc. 2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 5369 – 5375.
[6] L. Jaillet and T. Simeon, “Path deformation roadmaps: Compact graphs with useful cycles for motion planning,” *Int. J. of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008.
[7] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, “Motion planning for urban driving using RRT,” in *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 1681–1686.
[8] A. Nakhaei and F. Lamiraux, “Motion planning for humanoid robots in environments modeled by vision,” in *Proc. 8th IEEE-RAS Int. Conf. on Humanoid Robots*, 2008, pp. 197–204.
[9] E. Yoshida, K. Yokoi, and P. Gergondet, “Online replanning for reactive robot motion: Practical aspects,” in *Proc. 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 5927–5933.
[10] T. Fraichard, M. Hassoun, and C. Laugier, “Reactive motion planning in a dynamic world,” in *Proc. of the IEEE Int. Conf. on Advanced Robotics*. IEEE, 1991, pp. 1028–1032.
[11] S. Quinlan and O. Khatib, “Elastic bands: Connecting path planning and control,” in *Proc. 1993 IEEE Int. Conf. on Robotics and Automation*, 1993, pp. 802–807.
[12] O. Brock and O. Khatib, “Elastic strips: A framework for motion generation in human environments,” *Int. J. of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.
[13] E. Yoshida, C. Esteves, I. Belousov, J.-P. Laumond, T. Sakaguchi, and K. Yokoi, “Planning 3D collision-free dynamic robotic motion through iterative reshaping,” *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1186–1198, 2008.
[14] Y. Yang and O. Brock, “Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments,” in *Proc. Robotics: Science and Systems,*, 2007.
[15] J. Vannoy and J. Xiao, “Real-time adaptive motion planning

(RAMP) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Trans. on Robotics*, vol. 24, pp. 1199–1212, 2008.
[16] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. 2006 IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 2366–2371.
[17] D. Hsu, J.-C. Latombe, and S. Sorkin, "Placing a robot manipulator amid obstacles for optimized execution," in *Proc. 1999 Int. Symp. on Assembly and Task Planning*, 1999, pp. 280–285.
[18] S. Guha, D. Suri, and I. Suzuki, "Random probing to approximate medial axes and plan safe motion," in *Proc. 8th International Conference on Advanced Robotics*, 1997, pp. 353–358.
[19] 坂原洋人, 升谷保博, 宮崎文夫, "時空間 RRT による複数移動障害物を考慮したリアルタイム軌道生成," 計測自動制御学会論文集, vol. 43, no. 4, pp. 277–284, 2007.
[20] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios," *IEEE Trans. on Robotics*, vol. 20, no. 1, pp. 45–59, 2004.
[21] J. Ota, "Goal state optimization algorithm considering computational resource constraints and uncertainty in task execution time," *Robotics and Autonomous Systems*, vol. 57, pp. 403–410, 2009.

吉田 英一（Eiichi YOSHIDA）

1996 年東京大学大学院工学系研究科博士課程修了，博士（工学）．同年工業技術院機械技術研究所入所，2001 年 4 月より産業技術総合研究所知能システム研究部門主任研究員．2004〜09 年フランス LAAS-CNRS にて日仏ロボット工学共同研究ラボラトリー（JRL-France）共同ディレクター．現在，産業技術総合研究所知能システム研究部門 AIST-CNRS ロボット工学連携研究体長として，フランス CNRS との共同研究に従事．作業・動作計画，モジュール型ロボット，人間型ロボットの制御に興味を持つ．IEEE，日本機械学会などの会員．（日本ロボット学会正会員）

金広 文男（Fumio KANEHIRO）

1999 年東京大学大学院工学系研究科情報工学専攻博士課程修了．博士（工学）．1998 年より日本学術振興会特別研究員．2000 年電子技術総合研究所入所．現在産業技術総合研究所知能システム研究部門主任研究員．2007 年より 1 年間 LAAS-CNRS 客員研究員．ヒューマノイドロボットのシステム構成法，全身行動制御に興味がある．IEEE の会員．
（日本ロボット学会正会員）

横井 一仁（Kazuhito YOKOI）

1986 年東京工業大学大学院機械物理工学専攻修了．博士（工学）．1986 年工業技術院機械技術研究所に入所．1995〜1996 年スタンフォード大学客員研究員．2001 年産業技術総合研究所知能システム研究部門主任研究員，現在知能システム研究部門副研究部門長．ヒューマノイド研究グループ長，筑波大学連携大学院教授を兼任．ヒューマノイド，知的システムに興味を持つ．IEEE，日本機械学会会員．（日本ロボット学会正会員）

Pierre Gergondet

Pierre Gergondet graduated from Ecole Centrale de Paris with a major in embedded systems. He is now a PhD student in CNRS-AIST JRL, UMI3218/CRT, focusing on the usage of brain-computer interfaces to control a humanoid robot.