作業への優先度割り当てを用いたコマンド駆動型マニピュレータの
オンライン衝突回避手法

(指定する作業の優先と途中回避不可能な作業への対応)

佐藤 大輔^{*1}, 相山 康道^{*2}**Online collision avoidance method for command-based manipulators
by allocating priority to job consists of PTP commands
(Prioritization of specify job and correspondence to job
that manipulator can't avoid halfway)**Daisuke SATO^{*1} and Yasumichi AIYAMA^{*2}^{*1} Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki 305-8573, Japan^{*2} Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki 305-8573, Japan**Received: 8 February 2018; Revised: 28 May 2018; Accepted: 12 August 2018****Abstract**

In this paper, we introduce an online collision avoidance algorithm for multiple command based manipulators with a concept of job and command priority. Considering application to factory use, we adopt a robot controller which is close to actual industrial controller with a sequence of point-to-point motion command. Previously, our colleagues have developed online collision avoidance methods with interlock algorithm and with advanced collision map. But these two methods have some disadvantages; for example, they cannot restrict to have an collision avoidance motion during commands which must be done continuously such as insertion of a peg into a hole. We introduce a concept of priority with continuity to P-to-P command and job which is a group of some commands; pick an object from a certain position, place an object on a certain position etc. When deadlock is occurred, one robot must get out of the others way. To determine a getting out robot, priority has an important role. When a robot must get out, its priority changes to the same high priority of the other to let other robots on the getting out path move to avoid new collision. With some simulations, we have verified that our new algorithm works well in the case of mutual collision of multiple robot manipulators.

Keywords : Manipulator, Collision avoidance, Deadlock avoidance, Priority allocation, Mutual collision**1. 序 論**

近年、工場の生産ラインの自動化が進み、多くの産業用ロボットが導入されている。中でも、複数のロボットを近距離に配置する場合、ロボット同士の衝突回避を考慮した動作計画が必要である。動作計画を立てるにはティーチングペンダントやオフラインティーチングソフトを用いた方法が一般的であるが、衝突回避を考慮した動作計画には多大な計算量・時間・労力が必要となり、また作業内容を変更する際はこの動作計画を立て直す必要がある。そこで、これらの手間を省くために、オンライン環境での衝突回避が求められている。

オンライン環境での衝突回避の先行研究として、楊らによる複数台マニピュレータの幾何学的関係により形成したポテンシャル場を用いて衝突回避する手法(楊他, 1994)や、Akella らによる動力学に基づいたタイムスケールング則を用いて実行時間の最適化及び始動時間のタイミング調整を行い衝突回避する手法(Akella and Peng, 2004)

No.18-00078 [DOI:10.1299/transjsme.18-00078], J-STAGE Advance Publication date : 23 August, 2018

本論文は、第23回ロボティクスシンポジウム 講演論文集(2018), No.1C1の掲載内容に基づいた論文である。

^{*1} 筑波大学大学院システム情報工学研究科 (〒305-8573 茨城県つくば市天王台 1-1-1)^{*2} 正員, 筑波大学システム情報系

E-mail of corresponding author: aiayama@esys.tsukuba.ac.jp

が存在する。しかし、これらの手法はロボットコントローラ内部の軌道生成アルゴリズムの把握または変更を必要とする。一般に市販されている産業用ロボットコントローラの内部はブラックボックス化され、内部プログラムを把握・変更できない場合が多く、研究室のロボットコントローラとは条件が大きく異なることから、これらの手法を現場に適用するのは難しい。

産業用ロボットコントローラへの適用を考慮した研究として、白土はロボット間の接近速度に基づく干渉チェック方式を提案し、現場適用についても効果を確認した(白土, 2013)。しかし、衝突回避手段がロボットの停止のみであり、ロボット同士が進路を塞ぎ合う場合に互いに身動きが取れなくなってしまう状態(デッドロック)に対応できない問題点が挙げられる。

そこで、本研究グループでは現場への導入性を確保するために実際の産業ロボットコントローラに近い形式のロボットコントローラと Point-to-Point(PTP) コマンド方式を用いることを前提とし、複数台のロボットがそれぞれ独立して作業を行う環境を想定して、オンライン環境でも動作可能なロボット同士の衝突及びデッドロック回避手法について検討してきた(Zhou and Aiyama, 2015), (Afaghani, 2015, Afaghani and Aiyama, 2015)。

Zhou はロボットが適時必要な分の作業空間を占有し、その内部への他ロボットの侵入を禁止する衝突回避手法を提案した(Zhou and Aiyama, 2015)。この手法はコマンド送信のタイミング調整のみで衝突回避が可能であり、軌道生成アルゴリズムを必要としないため高い実用性を示したが、3 台以上の複数台環境において対応せず、また特定の作業に対応しない問題がある。

Afaghani は各ロボットの軌道情報を用いて、衝突しないタイミングでコマンド送信をする衝突回避手法を提案した(Afaghani, 2015, Afaghani and Aiyama, 2015)。この手法はコマンド送信のタイミング調整のみで衝突回避が可能であり、非常に高い効率とともに3 台以上の環境での対応を示したが、Zhou の手法と同様に特定の作業に対応せず、また実装においてロボットコントローラ内部でブラックボックス化される軌道生成アルゴリズムを必要とする問題がある。

本稿では、ロボットが行う作業に優先度の概念を付与し、以上の問題に対応可能な衝突回避手法を提案する。挿入等の作業の途中で衝突回避動作を行っては作業が破綻してしまう状況や、できるだけ先に遂行する必要がある作業に対応できるように、連続性と優先性を有する優先度を定義し、その概念に基づいた衝突回避アルゴリズムを提案する。2 章において、本研究で想定するロボットや作業環境の設定を行う。3 章において、従来の Zhou および Afaghani の手法を紹介し、本研究で検討すべき問題点を指摘する。4 章において優先度の提案および、それに対応した衝突回避のアルゴリズムを提案し、5 章では、いくつかの事例により提案アルゴリズムが機能することを示す。最後に 6 章でまとめと、今後に残された課題について紹介する。

2. 問 題 設 定

本章では、本研究において想定する作業環境と、前提とするコントローラシステムを説明する。

2.1 作業環境

現在、実際の現場で最も用いられているオンラインでの衝突回避手法として、共通ワークスペースのインターロック法が挙げられる。これは、複数のロボットの作業空間が重なった場合、動作内容は変更せずにタイミング調整により衝突が発生しうる空間(共通ワークスペース)に存在するロボットを1 台以下に保つ手法である。共通ワークスペースは事前に設定が必要であるが、このとき空間を大きく設定する必要があるため、他ロボットの作業効率を損なう。一方、作業シーケンスがオンラインで変動するようなシステムでは前述の空間の問題に加え、タイミング調整の設計が困難である。本研究では特にオンライン環境でも動作可能な衝突回避手法を目標とし、PTP コマンドにより動作するロボットを複数台配置し、それぞれが独立して作業を行う環境を想定する。

2.2 ロボットコントローラ

本研究では、NEDO 知能化プロジェクト(次世代ロボット知能化技術開発プロジェクト, 2008)の作業サブワーキンググループにより提案された、高・中・低レベルに分割されたモジュールを持つロボットコントローラを用いる(次世代ロボット知能化技術開発プロジェクト, 2010)。表1に各モジュールの役割を、図1にシステムの構

造を示す。アプリケーションにおけるジョブとは、作業内容によりまとめられた PTP コマンド群を指す。例えば物体をピックアップする場合、(1) 物体直上に移動し、(2) 把持する高さまで降下し、(3) グリッパを閉じ、(4) 直上に移動するといった 4 つのコマンドで作業が構成され、このまとまりをジョブとする。実際の産業用ロボットコントローラ役割はこのロボットコントローラの中～低レベルモジュール部分で表現することができ、単純に中レベルが上位から制御コマンドを受け取る方式であれば、本ロボットコントローラは実際の産業用ロボットコントローラに近似できると考えられる。よって、本研究では、各レベルモジュール内部のプログラムを変更しないこと、PTP コマンド制御方式を用いること、中レベルが単純に上位から制御コマンドを受け取ることを前提として、衝突回避を実現する。

Table 1 Role of each level module

Module	Role
Application	Order job to each robot based on motion plan with order of job
High Level	Provide PTP commands according to job
Middle Level	Generate Trajectory using PTP command and provide joint trajectory derived by Inverse Kinematic
Low Level	Control robot using each joint coordinate value

Fig. 1 System architecture of robot controller developed by NEDO project. This controller is three independent module with common interface. Role of middle and low level module is close to industrial robot controller. We approximate this robot controller to industrial robot controller by assuming the use of simple PTP command control method and not to change each level module's program

3. 先行研究

本章では本研究グループで行われた先行研究として、インターロック法を拡張した分割ワークスペース割り当て法と、複数台環境において検討を行った拡張衝突地図法を紹介する。

3.1 分割ワークスペース割り当て法 (Zhou and Aiyama, 2015)

本研究グループの Zhou により提案された分割ワークスペース割り当て法は、作業空間をキューブ上の領域要素 (リソース) に分割し、適時必要なリソースを占有して他ロボットの侵入を禁止し、衝突回避する手法である。

3.1.1 ロボットのモデリング

分割ワークスペース割り当て法では、計算負荷やモデリングの簡便さを考慮し、図 2 右図に示すように、ロボット全体を球の複合体で近似する。近似球は Link Modeling Sphere(LMS) と呼称し、ロボットのリソース占有判定に利用される。LMS の個数、中心位置、半径を変更することにより、多様なロボットのモデリングが可能である。

3.1.2 リソースの占有による衝突回避

図 2 にリソース占有判定の概要を示す。全体の作業空間は図 2 左図の様にキューブ上に分割され、各キューブの外接球を Workspace Cube Modeling Sphere(WCMS) と呼称する。ロボットの LMS の半径を R 、キューブの WCMS の半径を r 、LMS と WCMS の中心間の距離を d としたとき、以下の式を満たすとリソースは占有され、他ロボットの侵入が禁止される。

$$d \leq R + r \quad (1)$$

コマンド実行前に逆運動学から全 LMS の経路を導出し、その経路上で上式を満たすリソースを占有することで、ロボットは衝突の発生しない空間で作業を行うことができる。なお、コマンド実行に必要なリソースが既に

他ロボットに占有されて確保できなかった場合、目標位置への経路上でリソースの確保できる最も近い点に臨時の待機位置を設定し、リソースの開放を待つ。図3に例を示す。Robot1は目標地点 G_1 に至るためのリソースがRobot2に占有されているため、臨時の目標地点として W_1 を設定し、必要なリソースが開放されるまで待機する。Robot2が移動してリソースが開放された場合、Robot1は改めてリソースを確保し、 G_1 に移動する。

ここで、この手法のロボットコントローラを図4に示す。この手法では、図1の高レベルと中レベルの間に新たに開発した Collision Avoidance Planner(衝突回避計画器)を挿入している。衝突回避やリソース管理の処理は衝突回避計画器内で完結し、上位アプリケーション内で衝突回避を考慮して動作計画を組む必要がないため、アプリケーションプログラマの動作計画の負担を緩和できる。この手法は2台の環境までシミュレーションされ、高い実用性と効率を示しているが、特定の作業に対応せず、また3台以上の複数台環境に対応しない問題がある。以降、これら問題点について説明する。

Fig. 2 Robot is approximated as a set of spheres and Workspace is partitioned on cube. Robot's sphere and Cube's circumscribing sphere are used for collision detection.

Fig. 3 Robots occupy resource that is workspace partitioned on cube as much as they need. Robot starts moving after occupying all necessary resource.

Fig. 4 In previous research, collision avoidance planner with collision and deadlock avoidance algorithm is insert between middle level module and low level module.

3.1.3 途中回避が不可能な作業に対する問題

この手法では、両方のロボットがリソースの開放待ちをするときにデッドロックに陥るため、デッドロック回避を行う必要がある。このとき、回避を行うロボットは手先と目標値との距離がより大きいロボットに決定され、相手の必要とするリソースから最短距離で抜け出せる方向へ回避行動を行う。しかし、この手法ではペグインホールなど途中で回避行動を取ると把持物体やロボットが破損してしまう作業の最中でも回避をする危険性がある。図5にペグを穴に挿入した直後にデッドロック回避の指令を受け取った場合について例を示す。現在のシステムでは、回避するロボット・方向・タイミングはロボットの作業内容に依らず決定されるので、(a)のようにペグを挿入するコマンドの終了直後に回避を行う。しかし、ロボットと把持物体の安全性を確保するには(b)のようにペグインホールの終了後、ロボットが把持物体及び環境に拘束されない位置姿勢から回避を行うことが望ましい。しかし、衝突回避計画器内で PTP コマンドがどのような作業に属するものかは判別はできず、現状のシステムでは(b)のような回避動作は難しい。

3.1.4 3台以上への拡張において想定される問題

この手法はデッドロック回避先が他ロボットに塞がれる等、デッドロック回避先が見つからない場合について対応しておらず、システム全体が停止してしまう問題がある。ロボットが増えるほど障害物が増え、デッドロック回避先が見つからない可能性は高いと考えられる。また、多数台環境では他ロボット直近での待機行動はデッドロックを誘発したりデッドロック回避先を塞いでしまう可能性が高いと考えられる。よって、待機行動を削除し、デッドロック回避先が見つからない場合にも対応できるアルゴリズムが必要である。

Fig. 5 Deadlock Avoidance During Peg-in-Hole

3.2 拡張衝突地図法 (Afaghani, 2015, Afaghani and Aiyama, 2015)

本研究グループの Afaghani により提案された拡張衝突地図法では、PTP コマンドを中レベルコントローラに送信する前に、コマンドを送信した場合にいつ・どの時点で衝突するかを各ロボットの経路・軌道情報から割り出し、コマンド送信のタイミング調整を行うことで衝突を回避する手法である。タイミング調整による衝突回避が不可能な場合にはデッドロックが発生するが、この手法ではこの際衝突するロボットが動作する空間を立方体状に掃き出し、その掃き出し空間から最短距離で抜け出せる方向にデッドロック回避を行う。本手法はロボットの経路・軌道情報によりタイミング調整を行うために非常に効率が良い、また図 4 と同様のシステムで実装が可能であるが、ロボットコントローラ内部の軌道生成アルゴリズムを事前に把握する必要があり、実用化への問題が残る。

3.2.1 複数台環境への対応

この手法は分割ワークスペース割り当て法と同様に途中回避不可能な作業などロボットの作業内容に関して検討はなされていないが、複数台環境への対応が可能であり、4 台環境までシミュレーションがなされている。具体的には 3 台環境では図 6 のような三角配置、4 台の環境においては図 7 のような平行配置のもと手先を左右に振るようなピックアッププレイス動作を行い、このとき 3 台環境では全てのロボットの作業空間が、4 台環境では隣り合うロボット同士の作業空間が重複しているが、いずれも衝突・デッドロック回避を適切に行い、作業を完遂した。しかし、複数台環境においていずれかのロボット同士がデッドロックを起こした場合、デッドロック回避を開始するまでに衝突・デッドロックを起こしていない機体も含め全てのロボットが停止する必要があるため、デッドロックが発生した場合に全体の作業効率を大きく損なう危険性がある。

Fig. 6 Previous research's simulation with 3 robots (Afaghani, 2015)

Fig. 7 Previous research's simulation with 4 robots (Afaghani, 2015)

4. 提案手法

4.1 設計仕様

前節の問題点から必要とされる設計仕様について以下にまとめる。

- (1) 現場への適用性を高めるため、ブラックボックスとされる軌道生成アルゴリズムは利用せず衝突回避を行う
- (2) 作業途中でデッドロック回避を行えない場合のために、回避を行うロボット・タイミングを制御可能とする
- (3) 複数台環境において、他ロボット直近での待機動作は不要なデッドロックが発生しやすくと考えられるので、目標値途中での待機動作を削除する
- (4) 一部ロボットがデッドロック回避中でも、関与しない他ロボットは独自に作業を続ける

以上に対応するため、提案手法では軌道生成アルゴリズムが不要であり、現場への導入性の高い分割ワークスペース割り当て法を拡張し、優先度を利用した新たなリソース占有ルール及びデッドロック回避アルゴリズムを提案する。

4.2 優先度の仕様

提案手法では、作業に優先度を付与し、これにより優先する作業やデッドロック回避を行うロボット・タイミングを制御することを目的とする。この優先度の仕様は ART-Linux(加々美, 2013) のようなリアルタイム OS におけるリアルタイムスケジューリングを参考に、リアルタイムスケジューリングにおいて実時間処理が必要なタスクを、本稿では途中回避ができない作業と近似して優先度の仕様決定に利用している。本節では、後述するリソース占有ルール及びデッドロック回避アルゴリズムにおいて利用される、優先度の仕様について説明する。以下に、優先度の仕様を示す。

- (1) 優先度は『優先性』と『連続性』のパラメータを有し、優先性は『優先されやすさ』、連続性は『ジョブ途中で回避動作可能か』を表す。これらはジョブごとに設定され、アプリケーションプログラム作成時にプログラム中に記述されるか、動作計画アプリケーションから高レベルコントローラへジョブを送る際にアプリケーション内で設定される。連続性については高レベルコントローラ内で規定することも可能である
- (2) 優先性は『作業の優先されやすさ』を表し、数値によって設定され大小が比較できる
- (3) 連続性は『作業の途中でデッドロック回避を行ってよいか』を表し、連続・不連続のいずれかを設定する。不連続の場合はコマンドが、連続の場合はジョブが切り替わるタイミングで回避が可能となる
- (4) 衝突回避計画器のキュー内部には 1 種類のジョブに属するコマンドのみが存在し、コマンドはジョブの優先度を継承する。ロボットはリソース占有判定時とコマンド実行中に現在のコマンドの優先度を継承し、待機中などその他の場合は優先性は最低値に、連続性は不連続に設定される。

以上の仕様をもって優先度は後述するリソース占有ルール及びデッドロック回避アルゴリズムに利用され、これにより前節の問題に対応する。

4.3 ロボットコントローラ

図 8 に、提案するロボットコントローラの構成を示す。アプリケーションと衝突回避計画器の間に新たに PTP コマンドに優先度を付与する機能を備える優先度管理者 (Priority Manager) を挿入した。本モジュールは高レベルモジュールと接続されるが、高レベルモジュールが送受信する情報は従来と比較して変更はなく、先行研究の衝突回避計画器と同様に高・中・低レベルモジュール内部を変更することなく導入が可能である。

Fig. 8 In proposed controller, algorithm in collision avoidance planner is updated and priority manager that is new module for giving priority to PTP command is inserted between application and collision avoidance planner.

4.4 コマンド・キューの設定

優先度管理者と衝突回避計画器内にはロボットの台数分コマンドを格納するキューが存在し、上位コントローラからコマンドを受け取り、処理を行う。コマンドは目標値の他に、優先度と作業の種類 (PTP 指令, グリッパ開閉, 回避 PTP 指令) の属性を保持し、これらの属性を参照してリソース占有判定を行う。コマンドはジョブ単位で優先度管理者及び衝突回避計画器のキューに格納され、ジョブの全てのコマンドが格納された場合、それら全てを実行するまで新たなコマンドを受け取ることはできない。全てのコマンドが実行されるとキューはリセットされ、コマンドの格納が可能となる。以上より、各キューはジョブを格納する上限 1 のキューとみなすこともできる。

4.5 アプリケーションと新規モジュールの役割

本節では、アプリケーション、優先度管理者、衝突回避計画器の各モジュールについて説明する。これらのモジュールは内部でロボット 1 台ずつ処理を行い、完了後は別ロボットの処理を行う。

4.5.1 アプリケーション

図9に、アプリケーションの動作の概要を示す。アプリケーションにはジョブとその優先度、実行するロボットを記した動作計画が格納され、動作開始後はこれに則りジョブとその優先度、実行するロボットを優先度管理器に入力する。このとき、優先度管理器の対応するロボットのキュー内に未送信のコマンドがある場合、ジョブは入力されず、別のロボットの処理へと移行する。なお、アプリケーションプログラムは動作計画時にロボットごとにジョブの内容、作業順序とともに、ジョブに優先度を設定する必要がある。優先度の設定のない場合、ジョブの優先性は最低値に、連続性は不連続に設定される。

4.5.2 優先度管理器

図10に、優先度管理器の動作の概要を示す。優先度管理器では、アプリケーションからジョブを入力されると、まずそのジョブを実行するロボットに対応した高レベルモジュールから、ジョブを構成するコマンド群を要求する。高レベルモジュールは従来通りの機能を持つため、コマンド群は目標値と作業の種類のみ保持しており、優先度は記録されていない。そこでコマンドを自キュー内に保存後、アプリケーションから入力された優先度を先程受け取った全てのコマンドに継承し、衝突回避計画器内の対応するロボットのキューへ入力する。キュー内の全てのコマンドを衝突回避計画器に入力後、優先度管理器のキューをリセットし、アプリケーションから新たなジョブが入力可能となる。なお、衝突回避計画器内の対応するロボットのキュー内に未実行のコマンドがある場合、コマンドは入力せず、別のロボットの処理へと移行する。

4.5.3 衝突回避計画器

衝突回避計画器では、後述する新たなリソース占有ルールを用いて衝突・デッドロック回避及び中レベルコントローラへのコマンド送信のタイミング調整を行う。計画器内部でロボットがコマンドを実行する前に必要なリソースを逆運動学により導出し、他ロボットのリソースと競合したらタイミング調整やデッドロック回避を行う。なお、衝突回避計画器の動作の概要は次節に示すものとする。

4.6 優先度によるリソース占有ルールとデッドロック回避アルゴリズム

本節では、衝突回避計画器内で処理されるリソースの占有とデッドロック回避について説明する。このフローチャートを図11に示す。

STEP1 リソース占有判定

提案手法では、分割ワークスペース割り当て法と同様の方法でリソースの占有判定を行う。リソース占有判定方法の詳細は3.1節を参照するものとする。ロボットはコマンドを実行する前に逆運動学を解き、目標値までの必要なリソースを占有し、全てのリソースを確保できたら動作を開始する。なお、実行するコマンドの連続性が連続の場合、現在のコマンド単体ではなく、キュー内のそのジョブに属する全てのコマンドに必要なリソースを占有する。これは、連続なジョブを持つロボット同士が、リソースの競合から回避してはいけない状況でデッドロック回避するのを防ぐためである。

STEP2 デッドロック回避アルゴリズム

本節では、図11の赤枠内の動作を説明する。必要なリソースが既に他ロボットに占有されていた場合、衝突及びデッドロック発生とし、優先度を用いてデッドロック回避するロボットを決定する。ここで、最も優先されるのがジョブの連続性である。相手が連続なジョブの作業途中である場合、自分の優先度に関わらず相手に回避を命じることはできないので、リソース占有処理を終了し、現在位置で待機したまま次のロボットの処理へと移行する。次に、相手のジョブが不連続の場合について説明する。この場合、自分と相手のジョブの優先性を比較し、相手の優先性が自分より低かった場合、相手のキューに割り込む形で回避コマンドを指令できる。この回避コマンドは、必要なリソースから最短で抜け出すPTPコマンドとし、回避の原因となった優先性を引き継ぎ、連続性は不連続とする。この優先度の継承により、デッドロック回避先が他ロボットにふさがれた場合にも、高い優先性で広い範囲にデッドロック回避を伝播させ、回避先を探索することを期待する。なお、自分の優先度が相手より低かった場合、リソース占有判定を終了し、次のロボットの処理へと移行する。

STEP3 回避コマンドの採用判定

本節では、図11の赤枠上部の、実行するコマンドが回避コマンドだった場合について説明する。実行するコマンドが回避コマンドだった場合、まず自分のキューの回避コマンドの次に実行するコマンドを参照する。これが存

Fig. 9 Flowchart(Application)

Fig. 10 Flowchart(Priority Manager)

Fig. 11 Flowchart(Collision Avoidance Planner)

在しない場合はそのまま回避コマンドの実行に移るが、存在した場合、そのコマンドと優先度を比較し、回避コマンドを採用するかを判定する。次のコマンドが連続なジョブの2つ目以降のコマンドの場合、また不連続だが回避コマンドより優先性が高い場合、回避コマンドは破棄される。それ以外の場合、回避コマンドは通常のコマンドと同様に処理が行われ、必要なリソースが競合した場合、デッドロック回避するロボットの判定が行われる。

STEP4 回避できないデッドロックの判定

本節では、図 11 の赤枠下部の、自身が待機動作に入った際の動作について説明する。本アルゴリズムでは対応できないデッドロック回避が発生し、全てのロボットが停止した場合、ユーザに警告を出し、システムを停止するものとする。ロボットは自身が待機動作に入った際、他ロボットが待機状態か参照し、全てのロボットが待機状態ならカウントを1つ進める。これが規定値以上になったとき、ユーザに警告を出すものとする。

以上により、ロボットの作業内容に応じて優先度やデッドロック回避のタイミング調節を可能とし、回避先が他ロボットに塞がれる場合にも高い優先度を継承して広く回避先を探索することを期待する。

5. シミュレーション

本章では、前章にて説明した提案手法を用いてロボットが複数台配置された環境を想定してシミュレーションを行い、その性能を評価する。シミュレーションに用いるロボットの寸法はおおよそ以下のとおりである。ロボットは標準的な6軸垂直多関節型(R-P-P-R-P-R)で、手首部までの最大リーチは約530mm、上腕の長さ260mm、前腕の長さ270mmである。腕の太さは、上腕が約200mm、前腕が約140mmとなっている。手首部は長さ90mm、太さは約100mmである。可動範囲の制限は今回のシミュレーションでは特に設けてはおらず、リーチの届く範囲で運動が可能としている。ロボットの各リンクは、半径 $R=80\text{mm}$ の近似球(LMS)で覆うように設定されている。また、ワークスペースは25mm角のキューブ(外接球WCMSの半径 $r \approx 21.7\text{mm}$)で区切られている。

各ロボットの動作はあらかじめアプリケーションプログラムに記載されており、実行すべきジョブや優先度はあらかじめ決められているものとする。デッドロック回避時のコマンドの優先度については4章のアルゴリズムに従い、自動的に決定される。

5.1 2台環境のシミュレーション

本実験では、2台環境でのシミュレーションを行い、ジョブ及びそのコマンドに設定した優先度によってデッドロック回避するロボットの選択や、回避を行うタイミングの調整が可能かを評価する。

5.1.1 実験内容

同様の6軸マニピュレータを平行に500mmの間隔で図12左図のように配置し、青いロボットをUPJ、白いロボットをHP3Jと呼称する。各ロボットは独立して2回ずつピックアップアンドプレースを行い、作業終了後に初期姿勢へと戻る。作業はピックとプレースの2種類のジョブに分けられ、その内容を表5.1.1に、各ロボットの作業順序を表5.1.1に示す。ピック時、ロボットはオブジェクトに対し直上から真下に接近しなければ、把持は失敗するものとする。次に、作業順序と各座標値を図12の右図に示す。矢印はジョブに応じたロボット手先の移動を示す。なお、図12のピック位置①、③、⑤、⑦で同時に作業する場合、全ての組み合わせにおいて衝突が発生する。提案手法では全てのジョブの優先性を $UPJ < HP3J$ とし、全てのジョブの連続性を(a)不連続、(b)連続にした場合についてシミュレーションを行う。また、(c)分割ワークスペース割り当て法についても同様のシミュレーションを行い、結果を比較する。

Table 2 Content of Job

	P1, P2(Pick)	P'1, P'2(Place)
Command1	Move above object	Move to above place position
Command2	Down to grasp	Down to place
Command3	Close Gripper	Open Gripper
Command4	Rise	Rise

Fig. 12 Simulation with Two Robots

5.1.2 結果・考察

図13は各手法を用いた2台環境のシミュレーションのタイムチャートである。(a)¹、(b)²より、提案手法では必要なリソースが競合すると優先性の低いロボット(UPJ)がデッドロック回避を行っており、優先性により回避するロボットを制御できていることが分かる。なお、優先性の低いUPJから先に動作を開始しているのは、HP3Jのコマンドが格納される前にUPJのコマンドが格納され、処理されたためである。

次に、回避を行うタイミングに注目する。(b)の場合はジョブP1が終了してから回避を行っているが、(a)の場合はジョブP1の作業途中で回避を行っている。この回避はジョブP1の最初のコマンド1が終わった直後に行われ、HP3Jの退去後はコマンド2から再開している。この様子を図14に示す。このとき、UPJは回避位置からコマンド2を実行するので把持姿勢に直上から近づけず、把持に失敗している。また、同様の実験ではグリッパを閉じる際に回避を行い、把持物体をはじく様子も見られた。(c)の場合では、目標値途中の待機地点で回避を行ったために復帰できたが、同様の実験では(b)と同じく把持物体を正しく扱えない様子が見られた。

¹動画: <https://drive.google.com/file/d/1bGbHZ4DEmJFJYVmeC6XCaO1Z8Qzd8lZQ/view?usp=sharing>

²動画: <https://drive.google.com/file/d/1bOa3CPtOoQx-Dhz7W3VHxBz85iX0H949/view?usp=sharing>

Table 3 Application Order in Simulation with Two Robots

	UPJ	HP3J
(1)	Pick (Target①)	Pick (Target⑤)
(2)	Place (Target②)	Place (Target⑥)
(3)	Pick (Target③)	Pick (Target⑦)
(4)	Place (Target④)	Place (Target⑧)
(5)	Go back to initial pose	Go back to initial pose

Fig. 13 Timechart of Simulation with Two Robots

以上より、提案手法では優先度によって回避するタイミングを調整し、途中回避不可能なジョブにも対応できることを示した。また、全体の作業時間は (b)>(a)、優先性の高い UPJ の作業完了時間は (a)>(b) となり、作業の連続性を保つと全体の作業時間は短くなるが高優先性の作業は遅れるので、高優先性の作業を迅速に完了したい場合は周囲の作業が不連続の方が都合がよいことが確認された。作業時間に関しては、従来の手法 (c) が最も短くなっているが、これは連続性などを考慮していないためであり、本提案手法は、ある程度作業時間は延びるが、優先性や連続性を考慮できる手法となっている。

Fig. 15 Simulation with Three Robots in Parallel Arrangement

Fig. 14 Snapshot of Deadlock Avoidance in the Middle of Job

Fig. 16 Timechart of Simulation in Parallel Arrangement with Three Robots

5.2 3台環境のシミュレーション (平行配置)

本実験では3台のロボットを平行に配置し、隣り合うロボット同士のみ作業空間が重複する環境でシミュレーションを行う。拡張衝突地図法では4台環境において同様の実験がなされているが、一部のロボットがデッドロック回避を行う場合、回避コマンドの実行開始まで衝突・デッドロックに関与しない他のロボットも全て動作を停止する必要があった。提案手法では、デッドロックに関与していないロボットは停止しておらず動作を続けていても良い、としてアルゴリズムを構築しており、このアルゴリズムが有効に働くか、シミュレーションにより確認を行う。

5.2.1 実験内容

図 15 左図のように3台のロボットを平行に配置し、右図に表す作業点でピックアップ・プレースを行う。新たに配置した赤いロボットは THIRD と呼称する。このとき、3台環境においても2台環境と同様のパフォーマンスが

発揮できるか、また一部のロボットがデッドロック回避を行う場合、回避に関与しない他ロボットがどのような挙動をとるか観察する。ピック位置①と④、⑤と⑦はそれぞれ衝突の発生しうる領域であり、HP3Jは作業点⑥と⑦、THIRDは作業点①と②で作業を行い、UPJは作業点③と④、④と⑤を交互に行う。各ロボットの作業順序を表4に示す。また、全てのジョブの連続性は連続とし、優先性はUPJ>HP3J>THIRDとする。この条件で各ロボットが独立して2回ずつピックアンドプレースを行う。

Table 4 Application Order in Simulation with Three Robots

	UPJ	HP3J	THIRD
(1)	Pick(Target Value①)	Pick(Target Value⑦)	Pick(Target Value④)
(2)	Place(Target Value②)	Place(Target Value⑥)	Place(Target Value③)
(3)	Pick(Target Value①)	Pick(Target Value⑤)	Pick(Target Value⑦)
(4)	Place(Target Value②)	Place(Target Value③)	Place(Target Value⑥)
(5)	Go back to initial pose	Go back to initial pose	Go back to initial pose

5.2.2 結果・考察

図16に本実験での各ロボットのタイムチャートを、図17に作業の様子を示す³。今回の実験では、UPJとTHIRD間で衝突及びデッドロックが発生したが、デッドロック回避を行っている間、HP3Jは独自にリソースの占有や作業を行っていることが確認できる。以上より、過去の衝突地図法と比較すると、一部のロボットでデッドロック回避の処理を行ってもそれに関与しない他のロボットは停止することなく独自に作業を進められており、アルゴリズムは意図通りに動いていることが確認できた。

この問題は、より多数台、例えば10数台が連なったライン型の場合などに顕著で、1箇所2台の間でデッドロックが起きた際に、すべてのロボットが停止しなければならないとすると効率は非常に悪くなるが、本提案手法により、そのような問題を避けられる。

Fig. 17 In simulation, UPJ and THIRD deadlocked but HP3J continues and complete its job regardless of deadlock

Fig. 18 Simulation with Three Robots in Triangle Arrangement

Fig. 19 UPJ makes THIRD avoid with priority as same as UPJ's it. Next THIRD makes HP3J avoid with priority as same as UPJ's it.

5.3 3台環境のシミュレーション (三角配置)

本実験では三角形にロボットを3台配置し、最も低い優先度のジョブを持つデッドロック回避先をより優先性の高いジョブを持つロボットで塞ぐ。これにより、デッドロック回避先を塞いだ状態でより優先度の高いロボットからデッドロック回避コマンドを割り込まれた場合の各ロボットの挙動について観察する。

³動画：<https://drive.google.com/file/d/1bPlnrBJOt6hpwv71St3A9mBkcElZ5h73/view?usp=sharing>

5.3.1 実験内容

3台のマニピュレータを図18左図のように配置し、図18右図に表す作業点でピックアンドプレースを行う。ピック位置①, ②, ③, ④と③, ⑤, ⑥, ⑦はそれぞれ衝突の発生しうる領域であり、HP3Jは作業点⑥と⑦, THIRDは作業点①と②で作業を行い、UPJは作業点③と④, ④と⑤を交互に行う。ジョブの作業順序は表4と同様であるが、各目標値は図18に対応する。また、全てのジョブの連続性は連続とし、優先性はUPJ>HP3J>THIRDとする。UPJ, HP3Jは1回、THIRDは作業点④と⑥を1回ずつ計2回作業を行う。

5.3.2 結果・考察

図19に動作開始直後のデッドロック回避の様子を、図20にタイムチャートを示す⁴。優先性最高であるUPJの作業を優先するために、HP3JとTHIRDがデッドロック回避を行っている様子が見られるが、UPJの作業位置はHP3Jの回避前のリソースにまでは及んでいないことから、HP3Jに回避コマンドを割り込ませたのはTHIRDであると分かる。本来自身より優先性の高いロボットに回避を行わせることはできないが、これはUPJがTHIRDに回避コマンドを割り込ませた際に回避コマンドにUPJの優先性が継承され、このコマンドの優先性をTHIRDが継承することで一時的にTHIRDがHP3Jより高い優先性を有していたために本来優先性の低いTHIRDでもHP3Jに回避コマンドを割り込ませることができたと考えられる。

以上より、提案手法により、周囲を他ロボットに塞がれてデッドロック回避先が存在しない場合でも、より高い優先性を持って広い範囲にデッドロック回避を伝播させ、優先性の高い作業を実行可能とできることを示した。これにより、他ロボットによってデッドロック回避先が塞がれやすい多数台環境でも、安定してデッドロック回避が可能であり、優先性の高い作業が実行できると考えられる。

Fig. 20 Timechart in Triangle Arrangement Simulation with Three Robots

5.4 考察

これまでの3つの節の3通りの他、いくつかの状況でシミュレーションを行い、本手法により、連続性を維持しつつ衝突回避、デッドロック回避が可能であることを確認した。ただし、本手法は絶対にデッドロックが起きないことを保証する手法ではない。また、効率の面でも必ずしも優れた手法ではないと考えられる。

例えば、デッドロック回避できる方向が狭く、ロボット自身の幅に対しあまり余裕がなく、ワークスペースを分割した際に通れないと判断される場合や、袋小路の先に待機しているロボットがいる場合など、回避できる可能性があるにも関わらず回避できない状況が存在する。袋小路の場合などは一旦自身が回避し、先に待機しているロボットをどかせれば作業を遂行できるが、そのような計画能力は本手法では組み込まれていない。

同様に、数個先のジョブやタスクを勘案すれば、スケジューリングによって効率的な作業手順を構築できるような場合でも、本手法ではその時に与えられているジョブ・タスク情報のみで動作手順を決めるため、そのような結果的に効率的な計画は立てられない。これは動作指令（ジョブ・タスク）がオンラインでリアルタイムに生成されるかも知れない、という本研究の前提に基づくものであり、ここを変えることはその前提を大きく変えてしまうことになる。このように、本手法は効率（作業遂行時間）よりもリアルタイムな動作指示への適応性を重視した手法となっており、シミュレーション結果からもそのことが確認できる。

また、ワークスペースを有限のサイズで分割するという手法ゆえ、ロボットが密集している場合やロボットの形状によっては分割されたワークスペースを確保しづらく、動作をなかなか実行できない、という場合も存在するが、これは上記のような通れないと判断されてしまう場合を除き、時間はかかるが実行はできるだろうと考えられる。しかし、この点については検証を行っておらず、今後、性能評価を進めていきたい。

⁴動画：<https://drive.google.com/file/d/1bScDt9z8DD6VoSyUwxu1vBXHQF6DbEj3/view?usp=sharing>

6. 結論と今後の展望

本稿では、複数台のコマンド駆動型マニピュレータの衝突回避手法として分割ワークスペース割り当て法を拡張し、新たに優先度の概念を付与した提案手法を示した。また、2, 3 台の環境でのピックアンドプレイスについてシミュレーションを行い、その結果から優先する作業の設定及び途中回避不可能な作業への対応が可能で、一部のロボットがデッドロック回避を行っている最中にも関与しない他ロボットは独立に作業を行うことが可能であることを示した。また、デッドロック回避先が他ロボットに塞がれている場合に、より高優先度をもってデッドロック回避を伝播させ、広い範囲にデッドロック回避先を探索できることを示した。しかし、デッドロック回避指令が自分に循環する、つまり自分が作業を行うために自分が回避を行う必要がある場合には回避できないデッドロックが発生する。これはロボット間の距離を広くとり、回避先を広く用意することで抑制は可能だが、根本的な解決とはいえない。よって、今後はより複雑な作業環境や多数台のシミュレーションを行い、提案手法の性能を評価するとともに、多数台環境に対応したデッドロック回避アルゴリズムを開発する必要がある。また、今回はロボット間の相互干渉のみに着目してアルゴリズム開発、シミュレーションを行ったが、ワークスペースを分割する本手法では、障害物をそこへ配置することは容易に行えるので、今後は障害物を考慮したアルゴリズムの開発も進めていく。

文 献

- Afaghani, A. Y., On-line collision and deadlock avoidance of PTP command-based industrial manipulators using advanced collision map, University of Tsukuba (2015), Ph.D. Thesis.
- Afaghani, A. Y. and Aiyama, Y., On-line collision detection of n-robot industrial manipulators using advanced collision map, Proceeding of International Conference on Advanced Robotics (ICAR), Istanbul (2015), pp. 422-427.
- Akella, S. and Peng, J., Time-scaled coordination of multiple manipulators, Proceeding of the 2004 IEEE International Conference on Robotics & Automation (ICRA), New Orleans (2004), pp. 3377-3344.
- 加々美聡, ロボットシステム開発のための実時間 OS “ART-Linux”, 日本ロボット学会誌, Vol.31, No.3 (2013), pp.222-225.
- NEDO 次世代ロボット知能化技術開発プロジェクト, 「次世代ロボット知能化技術開発プロジェクト」基本計画 (2008).
- NEDO 次世代ロボット知能化技術開発プロジェクト, ACT 共通インターフェース仕様書 (2010).
- 白土浩司, 複数台の産業ロボット間における接近速度に基づく干渉チェック方式開発, 日本ロボット学会誌, Vol.31, No.7 (2013), pp.697-702.
- 楊向東, 山本元司, 毛利彰, ポテンシャル関数を用いる複数台マニピュレータの衝突回避動作計画, 日本機械学会論文集 C 編, Vol.59, No.568 (1993), pp.213-217.
- Zhou, J. and Aiyama, Y., On-line collision avoidance system for two PTP command-based manipulators with distributed controller, Advanced Robotics (2015), pp.239-251.

References

- Afaghani, A. Y., On-line collision and deadlock avoidance of PTP command-based industrial manipulators using advanced collision map, University of Tsukuba (2015), Ph.D. Thesis.
- Afaghani, A. Y. and Aiyama, Y., On-line collision detection of n-robot industrial manipulators using advanced collision map, International Conference on Advanced Robotics (ICAR), Istanbul (2015), pp. 422-427.
- Akella, S. and Peng, J., Time-scaled coordination of multiple manipulators, Proceeding of the 2004 IEEE International Conference on Robotics & Automation (ICRA), New Orleans (2004), pp. 3377-3344.
- Kagami, S., A realtime operating system “ART-Linux” for robot system development, Journal of the Robotics Society of Japan, Vol.31, No.3 (2013), pp.222-225 (in Japanese).
- NEDO Advanced Intelligent Robot Technology Development Project, Basic plan (2008) (in Japanese).
- NEDO Advanced Intelligent Robot Technology Development Project, ACT common interface specification (2010) (in Japanese).

- Shiratsuchi, K., The development of the collision check method for FA robots by using each approach speed, Journal of the Robotics Society of Japan, Vol.31, No.7 (2013), pp.697-702 (in Japanese).
- Yang, X., Yamamoto, M. and Mohri, A., Collision-free motion planning for multiple manipulator system using a potential function, Transactions of the Japan Society of Mechanical Engineers, Series C, Vol.59, No.568 (1993), pp.213-217 (in Japanese).
- Zhou, J. and Aiyama, Y., On-line collision avoidance system for two PTP command-based manipulators with distributed controller, Advanced Robotics (2015), pp.239-251.