(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2017-131973
(P2017-131973A)
(43)公開日　平成29年8月3日(2017.8.3)

(51)Int.Cl.			F I			テーマコード（参考）
B25J	9/10	(2006.01)	B25J	9/10	A	3C269
B25J	19/06	(2006.01)	B25J	19/06		3C707
G05B	19/4069	(2006.01)	G05B	19/4069		

審査請求 未請求　請求項の数 16　ＯＬ　（全 31 頁）

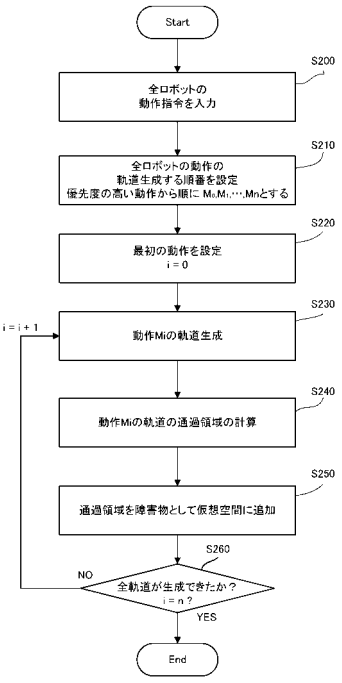
(21)出願番号　特願2016-11688(P2016-11688)
(22)出願日　平成28年1月25日(2016.1.25)

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100082337
弁理士　近島　一夫
(74)代理人　100141508
弁理士　大田　隆史
(72)発明者　前田　泰晴
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

Ｆターム(参考)　3C269 AB33 BB14 CC09 KK13 MN40
3C707 AS06 AS13 BS10 LS20 LV02
LV14 MS08

(54)【発明の名称】ロボット軌道生成方法、およびロボット軌道生成装置

(57)【要約】
【課題】実環境で複数台のロボットアームを動作させる時、その通過領域が互いに交差する可能性を著しく低減できるようロボット軌道生成を行う。
【解決手段】複数のロボットアームの軌道の始点および終点を含む動作指令リストを生成する（軌道定義データ生成行程：Ｓ２００）。動作指令リストに基づき各々の軌道生成を行う順序を決定する（生成順序決定行程：Ｓ２１０）。動作指令リスト中の特定のロボットアームに関し、始点および終点に基づき、障害物メモリに他のロボットアームの軌道生成に関して登録された障害物空間を回避するよう、軌道生成を行う（軌道生成行程：Ｓ２３０）。生成軌道で当該ロボットアームを動作させた際、当該アームの躯体によって掃引される掃引空間を、他のロボットアームが回避すべき障害物空間として障害物メモリに追加する（障害物登録行程：Ｓ２４０、Ｓ２５０）。
【選択図】図５



【特許請求の範囲】
【請求項１】
制御装置によって、同じ作業領域に配置されて動作する複数のロボットアームを動作させる軌道を生成するロボット軌道生成方法において、
前記制御装置が、生成すべき複数のロボットアームの軌道について、各軌道の始点および終点を含む軌道定義データを格納した動作指令リストを生成する軌道定義データ生成行程と、
前記制御装置が、前記軌道定義データ生成行程で生成された前記動作指令リストに含まれる複数の前記軌道定義データに基づき各々の軌道生成を行う軌道生成順序を決定する生成順序決定行程と、
前記制御装置が、前記生成順序決定行程で決定された軌道生成順序に基づき、前記動作指令リストに含まれる特定のロボットアームに関する前記軌道定義データに基づき、障害物メモリに登録された障害物空間のうち、当該ロボットアームに関して登録された障害物空間を除く障害物空間を回避するよう、当該ロボットアームを動作させる軌道を生成する軌道生成行程と、
前記制御装置が、前記軌道生成行程で生成された軌道で当該ロボットアームを動作させた際、当該ロボットアームの躯体によって掃引される掃引空間を、当該ロボットアームを除く他のロボットアームが前記軌道生成行程において回避すべき障害物空間として前記障害物メモリに追加登録する障害物登録行程と、
を含むロボット軌道生成方法。
【請求項２】
請求項１に記載のロボット軌道生成方法において、前記制御装置は、前記軌道生成行程および前記障害物登録行程を繰り返し実行することにより、前記動作指令リストに含まれる複数の前記軌道定義データに基づき軌道生成を行うに際し、前記障害物メモリに登録された障害物空間を回避する軌道が生成不可能となった場合は、前記動作指令リストに含まれる前記軌道定義データを、その時点で生成済みの軌道に対応する軌道定義データから成る第１の動作指令と、未生成の軌道に対応する軌道定義データから成る第２の動作指令と、に分割し、前記障害物メモリに登録されている障害物空間のデータをクリアしてから前記第２の動作指令に含まれる軌道定義データについて前記軌道生成行程および前記障害物登録行程を繰り返し実行し、生成した全部の軌道データに基づき、各々のロボットアームを動作させる場合に、前記第１の動作指令に対応するロボットアームの動作と、前記第２の動作指令に対応するロボットアームの動作と、を同期的に実行させるための同期指令を生成するロボット軌道生成方法。
【請求項３】
請求項１または２に記載のロボット軌道生成方法において、前記制御装置は、前記軌道生成行程および前記障害物登録行程を繰り返し実行することにより、前記動作指令リストに含まれる複数の前記軌道定義データに基づき軌道生成を行うのに先立ち、前記軌道定義データの各々の始点および終点において当該ロボットアームの躯体の占める空間を前記障害物メモリに登録しておくロボット軌道生成方法。
【請求項４】
請求項１から３のいずれか１項に記載のロボット軌道生成方法において、前記制御装置が、前記軌道定義データ生成行程において、作業者の入力操作に基づき、生成すべき複数のロボットアームの軌道について、各軌道の始点および終点を含む軌道定義データを格納した動作指令リストを生成するロボット軌道生成方法。
【請求項５】
請求項１から４のいずれか１項に記載のロボット軌道生成方法において、前記制御装置が、生成順序決定行程において、作業者の入力操作に基づき、前記動作指令リストに含まれる複数の前記軌道定義データに基づき各々の軌道生成を行う軌道生成順序を決定するロボット軌道生成方法。
【請求項６】

請求項１から４のいずれか１項に記載のロボット軌道生成方法において、前記制御装置が、生成順序決定行程において、前記動作指令リストに含まれる複数の前記軌道定義データに基づき各々の軌道生成を行う軌道生成順序を決定するロボット軌道生成方法。

【請求項７】

請求項６に記載のロボット軌道生成方法において、前記制御装置が、生成順序決定行程において、前記動作指令リストでより多数の前記軌道定義データが含まれるロボットアームの軌道生成が、前記動作指令リストでより少数の前記軌道定義データが含まれるロボットアームの軌道生成よりも先に実行されるよう、前記動作指令リストに含まれる複数の前記軌道定義データに基づき各々の軌道生成を行う軌道生成順序を決定するロボット軌道生成方法。

【請求項８】

前記制御装置に、請求項１から７のいずれか１項に記載の前記各行程を実行させるロボット軌道生成プログラム。

【請求項９】

請求項８に記載の前記ロボット軌道生成プログラムを格納したコンピュータ読み取り可能な記録媒体。

【請求項１０】

制御装置によって、同じ作業領域に配置されて動作する複数のロボットアームを動作させる軌道を生成するロボット軌道生成装置において、

前記軌道を生成する場合に、前記ロボットアームの躯体を通過させることができない障害物空間を登録する障害物メモリを有し、

前記制御装置は、

生成すべき複数のロボットアームの軌道について、各軌道の始点および終点を含む軌道定義データを格納した動作指令リストを生成し、

生成された前記動作指令リストに含まれる複数の前記軌道定義データに基づき各々の軌道生成を行う軌道生成順序を決定し、

決定した軌道生成順序に基づき、前記動作指令リストに含まれる前記軌道定義データに基づき、障害物メモリに登録された障害物空間のうち、当該ロボットアームに関して登録された障害物空間を除く障害物空間を回避するよう、当該ロボットアームを動作させる軌道を生成するとともに、

生成された軌道で当該ロボットアームを動作させた際、当該ロボットアームの躯体によって掃引される掃引空間を、当該ロボットアームを除く他のロボットアームが軌道生成において回避すべき障害物空間として前記障害物メモリに追加登録する、

ロボット軌道生成処理を実行するロボット軌道生成装置。

【請求項１１】

請求項１０に記載のロボット軌道生成装置において、作業者の入力操作に基づき、生成すべき複数のロボットアームの軌道について、各軌道の始点および終点を入力する軌道定義データ入力装置を備えたロボット軌道生成装置。

【請求項１２】

請求項１０または１１に記載のロボット軌道生成装置において、生成すべき複数のロボットアームの軌道について、各軌道の始点および終点を作業者に入力させる軌道定義データ入力手段を備えたロボット軌道生成装置。

【請求項１３】

請求項１０から１２のいずれか１項に記載のロボット軌道生成装置において、前記動作指令リストに含まれる複数の前記軌道定義データに基づき各々の軌道生成を行う軌道生成順序を作業者に入力させる入力手段を備えたロボット軌道生成装置。

【請求項１４】

請求項１０から１３のいずれか１項に記載のロボット軌道生成装置において、前記複数のロボットアームが配置された仮想空間を表示する表示装置を備えたロボット軌道生成装置。

【請求項１５】
請求項１４に記載のロボット軌道生成装置において、前記表示装置により、前記複数のロボットアームに関して実行される前記ロボット軌道生成処理を表示するロボット軌道生成装置。
【請求項１６】
請求項１４または１５に記載のロボット軌道生成装置において、前記ロボット軌道生成処理で生成された軌道で動作する前記ロボットアームを前記表示装置の前記仮想空間に表示するロボット軌道生成装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、制御装置によって、共通の作業領域に配置されて動作する複数のロボットアームを動作させる軌道を生成するロボット軌道生成方法、およびロボット軌道生成装置に関する。
【背景技術】
【０００２】
近年、工業製品の生産ラインなどにおいて、組み立て、加工、搬送、塗布、塗装などの作業に産業用ロボットのようなロボット装置が多用されるようになってきた。この種の生産ラインでは、生産能力の向上や省スペース化のために、複数台のロボット装置を近接して配置し、また、共通の作業領域の内部において、同時に、あるいは協働して作業を行うことがある。このように、共通の作業領域で複数のロボット装置を動作させる場合、ロボットアーム間で干渉（接触や衝突）が起きないよう制御する必要がある。
【０００３】
ロボット動作の教示には、ティーチングペンダントなどと呼ばれる操作装置で、実際の作業環境中でロボット装置を動かして確認しつつ、ロボットアームの位置、姿勢を変化させるロボット軌道を入力する作業が行われることがある。また、ロボット制御端末の表示装置でロボット装置の作業領域の３Ｄ仮想表示を行い、この仮想環境中でロボット装置を動作させつつ軌道入力を行うロボットプログラミング方式が用いられることもある。このような仮想環境を利用したロボット教示では、仮想環境中のロボット装置の制御、ティーチングペンダントに替えて、キーボード、ポインティングデバイス、タッチパネルなどの入力手段が教示点の入力に用いられる。
【０００４】
上記のような教示者の手動操作過程を含む従来のロボット教示（プログラミング）では、教示者がロボットの通過する経路や軌道、各ロボットを動作させるタイミングを考え、ロボット間で干渉し合わないように試行錯誤的に教示作業を行う必要がある。従って、このような教示者の手動操作の比重が大きい従来方式ではロボットの台数や作業の総数が増加するほど、ロボット同士の干渉のない軌道を確保することが難しく、また、教示作業に多大な時間がかかるという問題がある。当然ながら、生産ライン内の周辺の設備やロボットの配置に変更が生じた場合には、同様の教示作業を再度行わなければならない。
【０００５】
そこで、近年では、ロボットが干渉を起こさない経路（軌道）を自動的に生成する技術が研究されている。このような経路生成の手法として、例えばＲＲＴ（Ｒａｐｉｄｌｙ ｅｘｐｌｏｒｉｎｇ Ｒａｎｄｏｍ Ｔｒｅｅ）やＰＲＭ（Ｐｒｏｂａｂｉｌｉｓｔｉｃ ＲｏａｄＭａｐ Ｍｅｔｈｏｄ）といった経路生成アルゴリズムなどが提案されている。このような軌道／経路生成技術を用いることにより、教示者はロボットの動作の始点と終点を指定するだけで済むため教示作業の負担を軽減できる。また、経路生成は作業環境をシミュレートする仮想空間上のコンピュータ演算によって実行できるため、生産ライン内の周辺の設備やロボットの配置の変更が発生しても、少ない時間で干渉を起こさない経路を再計算することができる。
【０００６】

また、１台のロボットについて、作業環境中の障害物と干渉を起こさない経路を生成するのみならず、作業環境中に位置姿勢が刻々と変化する複数台のロボットが存在する場合に、各ロボット間で干渉を起こさないロボット軌道を生成する技術も提案されている。

【０００７】

例えば、下記の特許文献１では、ロボットの動作を開始するタイミングを調整することでロボット間の干渉を回避している。特許文献１の複数台ロボットの干渉回避方法では、各ロボットの動作の優先順位を設定し、各ロボットの所定時間内の動作によるロボットの通過予定領域、またはそれを含む拡大領域である占有領域を順次演算する。そして演算された占有領域間での重なりの有無を判定し、重なりが有る場合には、上記優先順位の低い一方のロボットを待機させると共に、優先順位の高い他方のロボットを動作させる。また、他方のロボットの占有領域は、該他方のロボットの動作中に既に動作が終った部分について順次解除される。そして上記他方のロボットの占有領域の変化に基づいて、重なりが無くなって以降、一方のロボットの動作を開始する。

【０００８】

また、下記の非特許文献１では、ロボットの動作時の速度を調整することでロボット間の干渉を回避している。非特許文献１の軌道生成では、各ロボットの始点から終点までの経路を他のロボットが存在しないとみなして計算し、次に各ロボットの経路上の位置をパラメータとしたグラフを生成する。そして、生成したグラフ上で干渉のない各ロボットの経路上の位置の組み合わせを探索し、最初に求めた経路と、各ロボットの経路上の位置の組み合わせを結合し、速度を調整しながら動作する互いに干渉し合わない軌道を得ている。

【先行技術文献】

【特許文献】

【０００９】

【特許文献１】特開平８－１６６８０９号公報

【非特許文献】

【００１０】

【非特許文献１】G. Sanchez and J.-C. Latombe, "On delaying collision checking in PRM planning: Application to multi-robot coordination" International Journal of Robotics Research, vol. 21, no. 1, pp. 5-26, 2002

【発明の概要】

【発明が解決しようとする課題】

【００１１】

上記の特許文献１および非特許文献１に記載の軌道制御は、いずれも同期的な制御を前提としている。例えば、いずれの軌道制御においても、各ロボットの経路・軌道を確定した後に、各ロボットの動作させるタイミングや各ロボットの動作の速度を同期的に制御、調整することでロボット間で干渉のない軌道を生成する。

【００１２】

しかしながら、上記のような従来の軌道制御を実施したとしても、ある時刻におけるロボットの通過領域は、他の時刻においては他のロボットの通過領域と交差している可能性がある。即ち、上記の従来技術では、時間軸上でロボット同士の干渉が起きないように制御しているために、実環境において、もしロボットの動作速度や動作タイミングにずれが生じると、ロボットアーム同士の干渉が生じる可能性がある。例えば、複数のロボットのうち片方のロボットが共通の通過領域上で異常停止していたり、ロボットの実際の動作の速度が理想の速度から大きくかけ離れていたりする場合に、ロボット同士で干渉する可能性がある。

【００１３】

また、上記の従来の軌道制御は、複数のロボットが特定の時刻において同じ空間を占めることがないよう制御する技術であり、空間軸および時間軸の双方に係わる演算が必要であり、一般に処理系の演算負荷が大きく、多大な計算資源が必要となる可能性がある。

【００１４】
本発明の課題は、上記の問題に鑑み、ロボット軌道の生成において、実環境のロボットアームの動作速度やタイミング制御の精度に拘らず確実に複数台のロボットアームの通過領域が互いに交差する可能性を著しく低減できるようにすることにある。また、本発明の他の課題は、低速な制御装置でも、高速に軌道生成を行えるようにすることにある。
【課題を解決するための手段】
【００１５】
上記課題を解決するため、本発明の、制御装置によって、同じ作業領域に配置されて動作する複数のロボットアームを動作させる軌道を生成するロボット軌道生成方法においては、前記制御装置が、生成すべき複数のロボットアームの軌道について、各軌道の始点および終点を含む軌道定義データを格納した動作指令リストを生成する軌道定義データ生成行程と、前記制御装置が、前記軌道定義データ生成行程で生成された前記動作指令リストに含まれる複数の前記軌道定義データに基づき各々の軌道生成を行う軌道生成順序を決定する生成順序決定行程と、前記制御装置が、前記生成順序決定行程で決定された軌道生成順序に基づき、前記動作指令リストに含まれる特定のロボットアームに関する前記軌道定義データに基づき、障害物メモリに登録された障害物空間のうち、当該ロボットアームに関して登録された障害物空間を除く障害物空間を回避するよう、当該ロボットアームを動作させる軌道を生成する軌道生成行程と、前記制御装置が、前記軌道生成行程で生成された軌道で当該ロボットアームを動作させた際、当該ロボットアームの躯体によって掃引される掃引空間を、当該ロボットアームを除く他のロボットアームが前記軌道生成行程において回避すべき障害物空間として前記障害物メモリに追加登録する障害物登録行程と、を含む構成を採用した。
【発明の効果】
【００１６】
上記構成によれば、ロボット軌道の生成において、実環境のロボットアームの動作速度やタイミング制御の精度に拘らず確実に複数台のロボットアームの通過領域が互いに交差する可能性を著しく低下させることができる。また、上記構成によれば、ロボットアームの障害物回避に時間軸に係る演算や制御を利用しないため、低速な制御装置を用いる場合でも高速な軌道生成が可能である。
【図面の簡単な説明】
【００１７】
【図１】本発明を実施可能なロボット軌道生成システムの構成を示した説明図である。
【図２】実施例１に係る軌道生成装置の制御系の構成を示したブロック図である。
【図３】実施例１～３の軌道生成制御の流れの概略を示したフローチャート図である。
【図４】実施例１に係る２つのロボットアームを示した説明図である。
【図５】実施例１に係る軌道生成制御の流れを示したフローチャート図である。
【図６】（ａ）～（ｃ）は実施例１に係る２つのロボットアームの軌道とその通過領域を示した説明図である。
【図７】実施例２に係るに係る軌道生成制御の流れを示したフローチャート図である。
【図８】（ａ）～（ｄ）は実施例２に係る２つのロボットアームの軌道とその通過領域を示した説明図である。
【図９】（ａ）～（ｄ）は実施例２に係る２つのロボットアームの軌道とその通過領域を示した説明図である。
【図１０】実施例２に係る２つのロボットアーム動作のタイミングを示す図である。
【図１１】（ａ）～（ｆ）は実施例３に係る２つのロボットアームの軌道とその通過領域を示した説明図である。
【図１２】実施例３に係る２つのロボットアーム動作のタイミングを示す図である。
【図１３】実施例３に係る軌道生成制御の流れを示したフローチャート図である。
【図１４】図１のロボット軌道生成システムにおいて実施可能な動作指令入力のためのユーザーインターフェースの説明図である。

【図１５】（ａ）～（ｃ）は実施例１における障害物メモリ中のデータ内容の遷移を示した説明図である。
【図１６】（ａ）～（ｆ）は実施例２における障害物メモリ中のデータ内容の遷移を示した説明図である。
【図１７】（ａ）～（ｆ）は実施例３における障害物メモリ中のデータ内容の遷移を示した説明図である。
【図１８】実施例１～３に係る軌道生成制御の要部の制御例を示したフローチャート図である。
【発明を実施するための形態】
【００１８】
以下、添付図面に示す実施例を参照して本発明を実施するための形態につき説明する。なお、以下に示す実施例はあくまでも一例であり、例えば細部の構成については本発明の趣旨を逸脱しない範囲において当業者が適宜変更することができる。また、以下の実施形態で取り上げる数値は、参考数値であって、本発明を限定するものではない。
【００１９】
なお、本明細書でいう「ロボット」とは、モータ等の動力源によって動作し、制御されるものであればその構成はどのようなものであってもよく、その基本構成によって本発明は限定されるものではない。以下の実施例では、回転関節を有する垂直シリアルリンク型の多関節ロボットアームを例に本発明に係るロボットの軌道生成方法および軌道生成装置につき説明する。しかしながら、例えば、パラレルリンク側の多関節ロボットや、平面上を移動するロボットなどを複数、作業環境に配置してその軌道生成を行う場合などにおいても、本発明は実施できる。
【００２０】
＜実施例１＞
図１、図２は本発明の実施例１に係るロボット軌道生成システムの構成を示している。図１は、ロボット軌道生成システム全体の構成を示している。図１において、例えば演算処理部１はＰＣ（パーソナルコンピュータ）のようなハードウェアを利用して構成することができる。演算処理部１には、作業者（ロボット教示者）が動作指令を入力するための入力部としてキーボード１１、マウス１２（あるいはトラックパッドのようなポインティングデバイス）が接続されている。また、作業者が入力操作した動作指令や、それに基づき生成されたロボット軌道などを確認するための表示装置としてディスプレイ１３が接続されている。
【００２１】
演算処理部１には、外部記憶装置１４と、ネットワークインターフェース３１などを接続することができる。図１の外部記憶装置１４は、主に外付けのＨＤＤやＳＳＤのようなディスク装置、あるいは各種の光ディスク装置などから構成される。ネットワークインターフェース３１は、実機のロボットアームに対して動作指令データなどを転送するために用いられる。
【００２２】
また、外部記憶装置１４、ネットワークインターフェース３１は、後述のロボット軌道生成プログラムを演算処理部１にインストールしたり、あるいはインストール済みの同プログラムを更新したりするためのインストール／更新手段として利用できる。その場合、例えば外部記憶装置１４は、メディア交換の可能は光ディスク装置などから構成することができる。本発明を実施するための制御プログラムは、上記の光ディスクを介して演算処理部１にインストールでき、またその更新が可能である。同様に、本発明を実施するための制御プログラムは、ネットワークインターフェース３１を介して、所定のネットワークプロトコルを利用して演算処理部１にインストールでき、またその更新が可能である。
【００２３】
図１のロボット軌道生成装置では、例えば、同じ作業環境に配置された複数（この場合２台）のロボットアーム５、６のための軌道生成を行う。ロボットアーム５、６の作業環

境中には、ロボットアーム５、６との干渉や衝突を回避すべき障害物７が配置される場合がある。この障害物７は、他の部品供給装置、換気ダクト、あるいは演算処理部１の筐体など、各々のロボットアーム５、６以外に作業環境中に配置されたあらゆる物体が対象となる。そして、後述の軌道生成制御では、ロボットアーム５、６が障害物７と干渉しないように、またロボットアーム５、６同士が相互に干渉しないように、ロボット軌道を生成する。

【００２４】

作業者（ロボット管理者、ロボットプログラマなど）は、キーボード１１、マウス１２などを用いて、ディスプレイ１３上に表示されるユーザーインターフェースを介してロボットアーム５、６のための動作指令を入力する（軌道定義データ入力）。ディスプレイ１３上のグラフィックス表示によって構成されるユーザーインターフェースとしては例えば後述の図１４のようなダイアログ表示、あるいは下記の３Ｄ仮想表示によるＧＵＩが含まれる。本実施例の動作指令入力では、作業者が、例えば生成すべき軌道の始点、および終点（の３次元座標）を含む軌道定義データを入力する。

【００２５】

演算処理部１は、入力された動作指令入力に基づき、軌道定義データから成る動作指令リストを生成し、この動作指令リストに基づき、ロボットアーム５、６の軌道生成を行う。

【００２６】

なお、図１のロボット軌道生成装置は、軌道生成の対象となる実機のロボットアームと接続されていてもよく、また、ロボットアームとの接続の無いオフライン環境においてもロボット軌道を生成することができるよう構成される。オフライン環境のロボット軌道生成では、ディスプレイ１３にプログラム対象であるロボットアーム５、６の３Ｄ仮想表示を行う。このような３Ｄ仮想表示ＧＵＩは、キーボード１１、マウス１２などを用いて、ディスプレイ１３上に仮想表示されたアームやその関節を操作できるよう構成することができる。

【００２７】

このような仮想環境を用いることにより、後述のようなダイアログ表示（例えば後述の図１４）の他、仮想表示されたロボットアーム５、６を操作することによっても、ロボット軌道の始点や終点（における位置、姿勢）を入力することができる。また、このような仮想環境によれば、必要に応じて、仮想空間上のロボットアームや障害物の配置の変更、といった操作も可能である。

【００２８】

図２は、演算処理部１の制御系の構成の一例をブロック図として示している。図示の構成はシステムバス２９上に、ＣＰＵ２０、ＲＯＭ２１、ＲＡＭ２２、ＨＤＤ２３（例えば演算処理部１の筐体内に内蔵のもの）などの各部を接続して成る。

【００２９】

図１の外部記憶装置１４はインターフェース２８を介してシステムバス２９に接続されている。図１では、特に外部記憶装置１４を構成する部材として光ディスク装置を例示したが、このようなディスクデバイスは、図２では記録ディスクドライブ２４として示してある。記録ディスクドライブ２４は、各種の光ディスクなどフォーマットで構成された記録ディスク１５のデータを読み取り、あるいはさらに記録ディスク１５に対してデータを書き込むことができる。従って、記録ディスク１５は、例えば本発明を実施するための制御プログラムの入出力に利用できる。その場合、記録ディスク１５は同制御プログラムのコンピュータ読み取り可能な記録媒体を構成する。

【００３０】

図１のキーボード１１、マウス１２、ディスプレイ１３は、それぞれインターフェース２５、２６、２７を介してシステムバス２９に接続されている。また、ネットワークインターフェース３１は、外部のネットワーク３２とシステムバス２９の間の通信を制御する。この通信には、ＴＣＰ／ＩＰ、あるいはその上で実行される各種の通信プロトコルを利

用することができる。
【００３１】
後述の軌道生成プログラムは、例えば上記の演算処理部１で動作するＯＳ上で実行することができる。このＯＳ（オペレーティングシステム）の構成によっては、ＨＤＤ２３などの記憶領域を利用した仮想記憶部を利用することができる。その場合、後述の障害物メモリのような領域は、ＲＡＭのような主記憶デバイスのみならず、このような（ＨＤＤ２３などによる）仮想記憶部上に確保される可能性がある。
【００３２】
図４は、本実施例の軌道生成対象であるロボットアーム５、６と、障害物７から含む作業環境の概略構成を示している。ここでは説明を容易にするため、ロボットアームの構成として、ごく簡略化された構成を示している。例えば、図４のロボットアーム５、６は３関節のロボットアームである。なお、以下では、制御の説明の都合上、ロボットアーム５、６をそれぞれＡ、Ｂのアルファベットによって参照することがある。
【００３３】
ロボットアーム５は、３つのリンク５５、５６、５７と、それらを連結する関節軸５１、５２、５３を備え、リンク５７の先端には、例えばロボットハンドのようなエンドエフェクタ５４が設けられている。また、ロボットアーム６は、リンク６５、６６、６７と、関節軸６１、６２、６３と、エンドエフェクタ６４を備える。各ロボットアーム５、６の３つの関節軸は回転駆動機構を持ち、回転の角度によってロボットアーム５、６の先端に位置するエンドエフェクタ５４、６４の位置が決定される。
【００３４】
各ロボットアーム５、６の関節軸の回転駆動機構は、不図示のサーボモータや減速機などの駆動系を備え、ロボット制御データに基づく回転角度に制御できるものとする。例えばロボットハンドのようなエンドエフェクタ５４、６４についても、同様のサーボモータや減速機などの駆動系を介してフィンガの開度のような制御条件を制御することができる。このような構成によって、各ロボットアーム５、６にロボット制御データを送信することにより、各アームを特定の位置姿勢に制御し、またエンドエフェクタ５４、６４で特定の操作を行わせることができる。なお、以上ではサーボモータのような駆動源を例示したが、関節やエンドエフェクタは必ずしも電気的に駆動制御されるものである必要はない。例えば、ロボットアーム５、６の関節やエンドエフェクタは、ロボット制御データによって制御される空気圧、油圧アクチュエータの空気圧や油圧によって駆動されるものであってよい。
【００３５】
図４に示すように、作業環境中においてロボットアーム５とロボットアーム６が近接して配置され、また、作業環境中には障害物７が存在する。このため、２つのロボットアームの関節軸の角度によっては、ロボットアームのリンクや関節軸、エンドエフェクタの間で干渉が発生する可能性がある。後述の軌道生成制御では、ロボットアーム５、６同士が相互に干渉しないように、ロボット軌道を生成する。その場合、もちろん、ロボットアーム５、６は、障害物７と干渉しないように、ロボット軌道が生成される。
【００３６】
以下、図３、図５、図６、図１４、図１５を参照して、本実施例のロボット軌道生成、特に複数台のロボットの通過領域が互いに交差しないようにロボット軌道を生成する手法につき説明する。
【００３７】
図３は、ロボット軌道生成制御の全体の流れを、図５は本実施例の複数台のロボットの通過領域が互いに交差しないようにロボット軌道を生成する制御手順を示している。図６（ａ）～（ｃ）は図４の様式に基づき軌道生成の様子を示している。また、図１４は、動作指令入力のためにディスプレイ１３に表示するダイアログを、図１５（ａ）～（ｃ）は実施例１における障害物メモリ中のデータ内容の遷移を示している。
【００３８】

図３のステップＳ１００では、作業者がキーボード１１、マウス１２などを用いて、ディスプレイ１３上に表示されるユーザーインターフェースを介してロボットアーム５、６のための動作指令を入力する。例えば、図１４に示すようなダイアログ１３１をディスプレイ１３に表示し、作業者にこのダイアログを介して動作指令を入力させる。

【００３９】

図１４のダイアログ１３１の上部には、例えばタイトル表示（この例では「動作指令の入力」）を含むタイトルバー１３１１を表示する。ダイアログ１３１は、ロボットアーム５（Ａ）のための入力を行うセクション１３１５とロボットアーム６（Ｂ）のための入力を行うセクション１３１６に分割されている。

【００４０】

各セクション１３１５、１３１６の上部は、各１行を１つの動作指令に対応する１レコードに対応させた入力部となっており、例えば１３２１～１３２４の各入力フィールドに区画されている。フィールド１３２１は、当該の動作指令の名称のフィールドで、このフィールドには作業者に任意の動作指令の名称を入力しても良く、また、この動作指令の名称はＣＰＵ２０によって自動的に順次、生成してもよい。この例では、ロボットアーム５（Ａ）側では動作指令Ｍａ１、Ｍａ２、Ｍａ３が、ロボットアーム５（Ｂ）側では動作指令Ｍｂ１、Ｍｂ２が入力されている。

【００４１】

本実施例では、これら動作指令は軌道定義データとして、動作指令に対応する軌道の始点、および終点のデータを入力する。これら始点、および終点は、ロボットアーム５、６の先端、例えばエンドエフェクタ５４、６４の基部周辺などの位置に設定される各アームの基準部位を移動させる３次元（ＸＹＺ）座標である。

【００４２】

これらの軌道の始点、および終点のデータは、フィールド１３２２、１３２３にそれぞれ入力される。図１４中では、ロボットアーム５（ａ）の動作指令Ｍａ１、Ｍａ２、Ｍａ３に対応する始点、終点データとして、Ｓａ１とＧａ１、Ｓａ２とＧａ２、Ｓａ３とＧａ３が入力されている。また、ロボットアーム６（ｂ）の動作指令Ｍｂ１、Ｍｂ２に対応する始点、終点データとして、Ｓｂ１とＧｂ１、Ｓｂ２とＧｂ２が入力されている。

【００４３】

なお、ここでは、始点、終点データの各フィールド中には上記のＳａ１、Ｇａ１…のような文字列表現を図示しているが、実際には３次元座標データのような数値データ表現を用いることができる。その場合、始点、終点データの各数値は、キーボード１１などから手動入力する他、仮想表示中のロボットアーム５、６を操作して決定させるようにしてもよい。例えば、作業者がディスプレイ１３の仮想表示中の空間をマウス１２でポイントして所望のアームの基準部位の座標を入力する。あるいは、ディスプレイ１３の仮想表示中のロボットアーム５、６をマウス１２などにより操作し、所望のアームの基準部位の座標に対応する位置、姿勢を取らせ、不図示のダイアログを用いて基準部位の座標を確定してもよい。いずれの場合も、確定された基準部位の座標に対応する数値データを始点、終点データとしてフィールド１３２２、１３２３に入力、表示する。

【００４４】

また、ロボットアーム５、６に対応する各セクション１３１５、１３１６には、直前の動作に対応する動作指令を指定できるようフィールド１３２４が配置されている。フィールド１３２４は、当該の動作指令の直前の動作指令（例えばフィールド１３２１の名称により参照される）を指定するのに用いられる。このフィールド１３２４を設けることにより、ロボットアーム５、６に対応する各セクション１３１５、１３１６は、各動作指令の前後の関係を関係づけたリンクトリストのような構造となる。図１４のように、ユーザーインターフェースにおいて、フィールド１３２４のような動作指令の前後関係を関係付け可能な入力手段を配置することにより、作業者は、入力済みの動作指令の前後を交換するような操作を行える。

【００４５】

実際に、図１４のように動作指令が入力された場合、ＣＰＵ２０は、ＣＰＵ２０が管理する主記憶ないし仮想記憶上に、入力された動作指令に対応する動作指令データを動作指令リストとして格納する。従って、図１４はユーザーインターフェースの画面表示と考えてもいいが、演算処理部１の主記憶ないし仮想記憶上の動作指令リストの構成に対応するメモリマップと考えてもよい。その場合、動作指令データは例えばリンクトリストのような記憶形式で主記憶ないし仮想記憶上の動作指令リストに格納され、上記のフィールド１３２４と同様のポインタデータによって、各動作指令の前後関係が関係づけられる。

【００４６】

各セクション１３１５、１３１６の下部には、その上部で入力された各アーム（Ａ、Ｂ）の軌道定義を確定するため、マウス１２などにより操作されるボタン１３１７、１３１７が配置される。

【００４７】

図３のステップＳ１００では、例えば上述のようなユーザーインターフェースを用いて、作業者が各ロボットアーム５、６（Ａ、Ｂ）に対して所望の動作指令を入力する。これに応じて、ＣＰＵ２０は動作指令の入力に対応する動作指令リストのデータ構造を主記憶ないし仮想記憶上に生成する。

【００４８】

図３のステップＳ１０１では、動作の軌道生成する順番を決定する優先度設定を行う。この優先度には、例えばロボットアーム５、６（Ａ、Ｂ）のいずれを優先するか、というロボットアーム単位の優先度が考えられる。また、図１４のように入力された動作指令Ｍａ１、Ｍａ２、Ｍａ３…、Ｍｂ１、Ｍｂ２…のどの軌道定義データから順に生成する、という優先度も考えられる。このように、（単位の異なる）優先度設定をいくつか可能としておくことにより、ロボットアーム５、６（Ａ、Ｂ）に実行させたい動作の特性に応じて好適な制御結果を得られる可能性が高くなる。このロボットアームないし動作指令を単位とする優先度に係る制御例については、図１８によって後述する。

【００４９】

図３のステップＳ１０２では、ステップＳ１０１で決定された優先度に従った順番で動作指令リスト中の１つの軌道定義データである始点、終点データに基づき、ロボットアーム５（Ａ）、ないし６（Ｂ）の軌道を１つ生成する。この場合、ＣＰＵ２０は、後述のように、障害物メモリに登録された障害物（の空間）を回避するように始点から終点までの軌道を計算する。この始点から終点までの軌道の計算には、上述のＲＲＴ、ＲＰＭといった公知の軌道生成アルゴリズムを利用することができる。

【００５０】

ステップＳ１０３では、ステップＳ１０２で生成した軌道で当該のロボットアーム５（Ａ）、ないし６（Ｂ）を動作させた場合に、そのロボットアームの躯体が掃引（通過）する空間を、他のロボットアームが進入できない障害物として障害物メモリに登録する。実際には、図３のステップＳ１０２、およびＳ１０３は、後述の図５に示すようなループによって逐一処理される。これにより、動作指令リスト中の全部の軌道定義データ（始点、終点データ）に基づき軌道生成が行われる。

【００５１】

図４、図６の例では、各ロボットアーム５（Ａ）、ないし６（Ｂ）の図示の平面内に係る動きしか図示していない。しかし、実際にロボットアームを３次元の作業領域で動作させた場合には、そのロボットアームの躯体が姿勢を変化させつつ掃引（通過）する空間は、特定の形状および体積を有する空間となる。このようなロボットアームの躯体が掃引（通過）する空間は、掃引容積（Ｓｗｅｅｐ（Ｓｗｅｐｔ）　Ｖｏｌｕｍｅ）などと呼ばれることがある。そこで、図６（ａ）～（ｃ）の例では、ロボットアームがある軌道で動作した場合にその躯体が掃引（通過）する空間をＳＶａのような参照符号によって示している。

【００５２】

図６（ａ）～（ｃ）は、ロボットアーム５（Ａ）、６（Ｂ）の動作指令Ｍａ、Ｍｂに対

応する軌道生成の様子を示している。ここで動作指令Ｍａに対応する軌道の（始点、終点）は（Ｓａ、Ｇａ）、動作指令Ｍｂに対応する軌道の（始点、終点）は（Ｓｂ、Ｇｂ）で、図６（ａ）のように配置されている。また、動作指令Ｍａ、Ｍｂに基づき、ＣＰＵ２０の軌道計算によって生成される軌道はＴｒａ、Ｔｒｂによって示してある。

【００５３】

図６の例では、設定された優先度によって、ロボットアーム５（Ａ）の軌道が先に生成されるものとする。ここでロボットアーム５（Ａ）の軌道Ｔｒａは、既に障害物メモリに登録されている障害物７を回避するため図示のような形状で生成される。この生成軌道のロボット動作によってロボットアーム５（Ａ）の躯体が掃引する掃引領域はＳＶａである。このＳＶａの空間は、ロボットアーム５（Ａ）の躯体の形状および姿勢、軌道Ｔｒａなどに応じた形状と（体積）を有する空間である。

【００５４】

本実施例では、１つのロボットアームの軌道を生成すると、その掃引空間ＳＶを（障害物７と同様の）障害物メモリに登録する。図１５（ａ）～（ｃ）は、図６（ａ）～（ｃ）の軌道生成に応じて、障害物メモリ２２０１に障害物（ＳＶｘ）が登録されていく様子を示している。

【００５５】

図１５（ａ）～（ｃ）の障害物メモリ２２０１には、例えばＲＡＭ２２のようなＣＰＵ２０の管理する主記憶上に確保され、概ね２つのフィールド２２０３、２２０４から成る。なお、図１５では障害物メモリ２２０１の記憶先としてＲＡＭ２２に相当する「２２」の参照符号を記憶領域全体に対して用いている。しかしながら、これはあくまでも便宜上のもので、障害物メモリ２２０１は、先の動作指令リストの場合と同様にＣＰＵ２０が管理する主記憶ないし仮想記憶（例えばＲＡＭ２２あるいはさらにＨＤＤ２３）上に確保されるものであってよい。この点は後述の図１６、図１７などにおいても同様である。

【００５６】

障害物メモリ２２０１のフィールド２２０３には、障害物の識別データ（名称や識別番号など）を格納する。また、フィールド２２０４には障害物の立体空間を定義する形状データ（の実体データ、または別途確保された不図示の格納空間へのポインタデータ）を格納する。

【００５７】

フィールド２２０４の形状データの形式は任意である。たとえば作業空間を単位ヴォクセル（Ｖｏｘｅｌ）の集合として考える。その場合、障害物メモリ２２０１に格納する障害物の形状データは、例えば単純な例では障害物の占めるヴォクセル（Ｖｏｘｅｌ）の座標データへのポインタリストなどとすることが考えられる。また、参照利用の容易なデータ形式としては、作業空間に対応するヴォクセル空間に３次元ヴォクセルデータテーブル（不図示）を主記憶ないし仮想記憶上に確保してもよい。そして、このヴォクセルデータテーブル中の障害物の占めているヴォクセルに対応するメモリセルに障害物フラグ（あるいはさらに、躯体掃引によってその障害物を生成したロボットの識別データ）を配置する。ヴォクセルデータテーブルを併用することによって、作業空間中の特定の座標が障害物の領域に含まれているか、といった参照を容易に行える障害物メモリを実現することができる。

【００５８】

図６、図１５の場合、まず、障害物７を回避するように図６（ｂ）のようにロボットアーム５（Ａ）の軌道Ｔｒａが生成される。この処理に先立ち、障害物７の形状データは、図１５（ａ）のように既にフィールド２２０３、２２０４に登録済みである。

【００５９】

図６（ｂ）のようにロボットアーム５（Ａ）の軌道Ｔｒａが生成されると、その掃引空間（ＳＶａ）のデータが図１５（ｂ）に示すように障害物メモリ２２０１に追加登録される。この場合、フィールド２２０３の識別データ（図の例ではＳＶａのような名称）を参照することにより、対応する躯体掃引によってその障害物を生成したロボットの識別を識

別できるものとする。

【００６０】

続いて、ロボットアーム６（Ｂ）に関して、（Ｓｂ、Ｇｂ）の始点、終点データから軌道（図６（ｃ）のＴｒｂ）を生成する。この時、障害物メモリ２２０１の内容を参照すると、既にロボットアーム５（Ａ）の掃引空間（ＳＶａ）に対応する障害物が障害物メモリ２２０１に登録されている（図１５（ｂ））。このため、ＣＰＵ２０は、図６（ｃ）に示すように、ロボットアーム５（Ａ）の掃引空間（ＳＶａ）を回避するよう、始点、終点データ（Ｓｂ、Ｇｂ）からロボットアーム６（Ｂ）の軌道Ｔｒｂを生成する。

【００６１】

このロボットアーム６（Ｂ）の軌道Ｔｒｂの生成によって、ロボットアーム６（Ｂ）の躯体が掃引する空間（ＳＶｂ）は、上記同様に障害物メモリ２２０１に追加登録される。図１５（ｃ）は、軌道Ｔｒｂに対応するロボットアーム６（Ｂ）の掃引空間（ＳＶｂ）が追加登録された直後の障害物メモリ２２０１の状態を示している。

【００６２】

以上のように、図３～図６、図１４、図１５によって概略を説明した軌道生成および障害物メモリ管理によって、ロボットアーム５、６に関して、相互のアーム同士が干渉せず、また、作業環境中の障害物７とも干渉しない軌道を生成することができる。

【００６３】

本実施例においては、障害物の回避戦略としては、時間軸や制御タイミングを利用しない。本実施例では、特定の軌道でロボットアーム躯体が掃引する空間を、そのロボットアームが占める可能性がある障害物空間として障害物メモリ２２０１に登録する。このため、実機環境でロボット動作の遅延などのタイミングのずれが生じている場合でも、ロボットアーム同士あるいはロボットアームと他の障害物が確実に干渉することのない軌道を生成することができる。即ち、本実施例によれば、よりフェイルセーフなロボット軌道の生成が可能である。

【００６４】

ここで、図３のステップＳ１０２、Ｓ１０３で示したロボット軌道生成、および障害物の追加登録処理のより詳細な制御例を図５に示す。

【００６５】

図５は、ロボットの通過領域が互いに交差しない軌道を生成する本実施例の制御手順の一例を示す。図３で説明した通り、制御手順全体の流れは、まず、動作指令の入力および動作指令リストの生成（Ｓ１０１）、優先度設定（Ｓ１０２）から始まる。その後、軌道生成（Ｓ１０３）、および障害物追加（Ｓ１０４）が行われる。図５はこの制御手順のより詳細な構成の一例を示している。

【００６６】

図５は次のような制御行程を含む。即ち、複数のロボットアームの軌道の始点および終点を含む動作指令リストを生成する（軌道定義データ生成行程：Ｓ２００）。動作指令リストに基づき各々の軌道生成を行う順序を決定する（生成順序決定行程：Ｓ２１０）。動作指令リスト中の特定のロボットアームに関し、始点および終点に基づき、障害物メモリに他のロボットアームの軌道生成に関して登録された障害物空間を回避するよう、軌道生成を行う（軌道生成行程：Ｓ２３０）。生成軌道で当該ロボットアームを動作させた際、当該アームの躯体によって掃引される掃引空間を、他のロボットアームが回避すべき障害物空間として障害物メモリに追加する（障害物登録行程：Ｓ２４０、Ｓ２５０）。

【００６７】

図５において、図３の動作指令の入力および動作指令リストの生成（Ｓ１０１）はステップＳ２００に、また、優先度設定（Ｓ１０２）はステップＳ２１０に相当する。ステップＳ２２０は、ステップＳ２３０～Ｓ２５０のループの初期化処理に相当する。また、図３の軌道生成（Ｓ１０３）はステップＳ２３０に、障害物追加（Ｓ１０４）はステップＳ２５０に相当する。ステップＳ２６０は、ステップＳ２３０～Ｓ２５０の各処理を動作指令リストに含まれる軌道定義データ（始点、終点）につき実行させるための制御ステップ

である。
【００６８】
図５のステップＳ２００では、図１４で説明したようなユーザーインターフェースを介して全ロボットアームに対して軌道を生成したい動作の指令を作業者（教示者）が入力する。ここで、例えばロボットアーム５について１つの動作指令Ｍａ１が始点と終点（Ｓａ、Ｇａ）の軌道定義データによって与えられたものとする。また、ロボットアーム６については１つの動作指令Ｍｂ１が始点と終点（Ｓｂ、Ｇｂ）の軌道定義データ動作指令Ｏｐ１として与えられたものとする。
【００６９】
この動作指令とは、各ロボットアームに対して指定される複数の動作の集合を指し、その動作指令データは上述のような動作指令リストとして、ＣＰＵ２０が管理する主記憶ないし仮想記憶上に格納される。
【００７０】
また、動作（Ｍ）は、ロボットアーム（５、６）のエンドエフェクタ（５４、６４）の基部などに相当する基準部位の始点と終点の座標の組合せを含む軌道定義データによって定義される。
【００７１】
多関節ロボットのように、エンドエフェクタ（５４、６４）の位置について、ロボットアーム（５、６）の姿勢が複数、存在する場合は、各関節の角度の組み合わせを始点や終点として指定する。
【００７２】
上記のように与えられた動作指令Ｏｐ１は、
Ｏｐ１　＝　｛　Ｍａ１　、　Ｍｂ１　｝
から成り、各動作（Ｍａ１、Ｍｂ１）は
Ｍａ１　＝　（　Ｓａ　、　Ｇａ　）　、　Ｍｂ１　＝　（　Ｓｂ　、　Ｇｂ　）
から成る。
【００７３】
次に、図５のステップＳ２１０では、軌道生成順序の決定、即ち、動作指令リストの中から、どの順番で動作の軌道生成を行っていくかを決定する。順番の決定方法はロボット全体での動作時間に影響するため様々な方法が考えられる。
【００７４】
優先度の制御例については、後で詳細に説明するが、ここではロボットアーム５（Ａ）の方がロボットアーム６（Ｂ）よりも動作の優先度が作業者によって高く設定されたものとする。実際、各ロボットアームに実行させる動作の重要度に差があるようなケースは稀ではないと考えられ、このようなロボットアーム単位の優先度設定をユーザ（作業者）が行えるようにしておくことで、より現実的な軌道生成が可能になる。
【００７５】
もし、ロボットアーム５（Ａ）の方がロボットアーム６（Ｂ）よりも動作の優先度が高く設定されている場合には、ロボットアーム５の動作指令Ｍａ１を先に軌道生成し、ロボットアーム６の動作指令Ｍｂ１を後に軌道生成するよう制御する。また、本実施例の軌道生成および障害物登録では時間軸に係る制御を行わないため、動作指令リスト中の動作（Ｍ）は、例えば入力順などに関係なく、シャッフルして特定の優先順位で軌道生成（および障害物登録）するようにしてもよい。ステップＳ２１０の処理は、生成順位を格納したテーブルメモリなどを確保し、動作データ指令リスト中の動作（Ｍ）を指すポインタデータなどを上記の優先順位で格納するような処理によって実現される。
【００７６】
図５のループ制御では、軌道生成（障害物登録）順位はポインタ（あるいはカウンタ）ｉによって制御される。ここでｉ＝０は最優先の処理の優先順位に相当し、ｉは０、１、２…とインクリメントされて参照されるが、以後、順に優先順位が下っていくものとする。そこで、図５のステップＳ２２０では、順位はポインタ（あるいはカウンタ）ｉはｉ＝

０に初期化する。

【００７７】

以後、動作指令Ｍｉの軌道生成（Ｓ２３０：ロボット軌道生成処理）、動作指令Ｍｉに対応するロボットアームの通過（掃引）領域の生成（Ｓ２４０）、通過（掃引）領域の追加登録（Ｓ２５０：障害物登録行程）を繰り返し実行する。この制御ループは、ステップＳ２６０で全部の動作指令を処理することが確認されるまで続けられる。動作指令を全て処理していない間は、制御はポインタ（あるいはカウンタ）ｉをインクリメント（ｉ＝ｉ＋１）しつつ、ステップＳ２６０からＳ２３０にループする。

【００７８】

ステップＳ２３０では、ロボットの１動作分の軌道生成を行う。このステップＳ２３０は、例えば経路計画行程、経路短縮行程、軌道計算行程のような３ステップで実行できる。

【００７９】

このうち、経路計画行程では、始点から終点まで向かう間の障害物を回避する経路を求める。経路を求めるには、上記のＲＲＴ、ＰＲＭのような公知の各種の手法によりＣＰＵ２０がランダム探索により自動で計算する方式を用いることができる。

【００８０】

次に、経路短縮行程で、経路計画行程により生成された経路の修正を行うことができる。経路計画行程でランダム探索を用いた方法を選択した場合、生成される経路が回り道をしていたり、角度変化が急になっていたりするため、経路の修正を行う必要がある。この経路短縮行程で、経路の短縮も、経路中の点同士で干渉なく結べる点を探す方法や、スプライン曲線で近似することにより経路を滑らかにする方法など多くの方法が考えられるが、ここではどのような方法を用いてもよい。

【００８１】

また、経路計画行程と、経路短縮行程では、３Ｄの仮想空間上で、ロボットアーム（５、６）同士や障害物（７）の干渉がないか確認する処理を行う。ここでは、ロボットアーム（５、６）の形状データと、障害物メモリ２２０１に保持された障害物データ（ＳＶ）は、例えば上述のヴォクセル（やポリゴン）の集合として取り扱うことができる。そして、ヴォクセル（ポリゴン）の集合同士が幾何学的に接触しているかを判定することによりロボットアーム（５、６）同士や障害物（７）の干渉状態がないかを判断する。

【００８２】

なお、障害物メモリ２２０１には、軌道生成に伴なうアームの通過（掃引）領域が障害物データ（ＳＶ）として格納されるが、一般には（当然ながら）、自身のアームの通過（掃引）によって生成された障害物データ（ＳＶ）は回避しなくてもよい。例えば、上記構成におけるロボットアーム５（Ａ）の動作（Ｍａ）の軌道（経路）生成においては、ロボットアーム５（Ｂ）の動作（Ｍｂ）によって通過（掃引）によって生成された障害物（ＳＶｂ）を回避する。しかしながら、ロボットアーム５（Ａ）の動作（Ｍａ）の軌道（経路）生成においては、ロボットアーム５（Ａ）の動作（Ｍａ）によって通過（掃引）によって生成された障害物（ＳＶａ）は回避する必要はない。

【００８３】

ステップＳ２４０の軌道計算行程では、上記の経路計画／経路短縮行程によって生成された経路（軌道）を、最短の時間で動作できるように、ロボットアーム（５、６）の各関節の回転速度や加速度などを計算する。ただし、この軌道計算行程では、各関節の回転速度や加速度は記憶部３に保持された制約条件を超えない範囲で計算する。しかしながら、本実施例の軌道計算では、最もフェイルセーフな障害物回避条件を用いるため、ロボットアーム間の同期を考慮する必要がなく、単体のロボットで最短の時間で動作できるような軌道を計算すれば良い。

【００８４】

なお、この軌道計算行程によって生成された軌道データ群（ロボットアームの制御データ）は、主記憶ないし仮想記憶上、あるいはＨＤＤ２３などに確保された不図示の軌道デ

ータの記憶領域に格納する。

【００８５】

ステップＳ２４０では、ステップで求めた軌道で動作させた場合にロボットアームの躯体（リンクや関節軸）が通過（掃引）する領域（空間）を計算する。各ロボットアーム（５、６）の形状や寸法などのジオメトリのデータをＨＤＤ２３やＲＯＭ２１などに予め用意しておくことにより、このジオメトリに基づき、ＣＰＵ２０は上記の通過（掃引）領域（空間）を計算することができる。実際の演算では、公知の各種の移動物体の通過した領域を求めるＳｗｅｐｔ　Ｖｏｌｕｍｅ生成手法や、通過領域を立方体や球体などの簡易な形状へ近似する手法などを用いることができる。また、形状データをポリゴンの集合として表現し、干渉の確認処理に利用するようにしてもよい。なお、ロボットアーム（５、６）の形状や寸法などのジオメトリのデータは、ディスプレイ１３上の仮想表示を制御するためにも用いることができる。

【００８６】

図６（ｂ）の例では、ロボットアーム５に対して、軌道Ｔｒａが生成された場合、上記のＣＰＵ２０による通過（掃引）領域の計算によって、始点Ｓａから終点Ｇａまでその躯体が通過することによって、図中の通過（掃引）領域（ＳＶａ）が生成される。

【００８７】

そして、ステップＳ２５０で、ステップＳ２３０の軌道生成に基づき生成された通過（掃引）領域（ＳＶ）を障害物メモリ２２０１に登録する。障害物メモリ２２０１に登録された障害物は領域を仮想空間上の新たな障害物として追加され、続くロボットアーム（６）の軌道生成では、障害物メモリ２２０１中の全部の障害物データ回避するように軌道計算が行われる。

【００８８】

図６の例では、ロボットアーム５の動作指令Ｍａ１の軌道生成と、通過領域の障害物追加が完了した後に、ロボットアーム６の動作指令Ｍｂ１の軌道生成（と通過領域の障害物追加）を行う。処理の流れは上述と同様であるが、ロボットアーム６の動作指令Ｍｂ１の軌道生成時には、新たな障害物として通過領域ＳＶａが追加されている（図１５（ｂ））。このため、動作指令Ｍｂ１の軌道生成においては、図６（ｃ）に示すように予め配置されている障害物７だけでなく、動作指令Ｍａ１の軌道Ｔｒａの通過領域ＳＶａも回避するように実行される。

【００８９】

こうしてロボットアーム６に関しては、ロボットアーム５の軌道Ｔｒａと交差することのない軌道Ｔｒｂを得ることが出来る。以上の説明では、ロボットアーム５－＞６と軌道生成を行う最も単純な例を示したが、ロボットアーム６の軌道Ｔｒｂに対応する通過（掃引）領域も新たな障害物として障害物メモリ２２０１に追加する。これにより、後続の動作指令に基づいて（干渉状態を生じない）軌道生成を続行することができる。

【００９０】

なお、以上では、最も単純な２つのロボットアームの例で説明したが、３つ以上のロボットアームの軌道生成についても、上述同様の手順により軌道生成およびそれに伴なう障害物登録を行うことができる。

【００９１】

以上のように、本実施例によれば、作業環境中の障害物（７）はもちろんロボットアーム（５、６）同士で互いに交差しない軌道を生成できる。このため、干渉する恐れのない軌道で複数のロボットアームを動作させることができる。本実施例の障害物の生成および障害物メモリへの登録を基本とする回避制御は、時間軸や制御タイミングに依存しない。即ち、ロボットアームの掃引（通過）空間（領域）を、時間軸上でそのアームが取り得る位置姿勢の最大範囲として障害物メモリに登録する。このため、本実施例によれば、実環境でアーム制御の遅延などのタイミングのずれが生じても確実にロボットアーム（５、６）同士の干渉を回避可能な軌道を生成することができる。また、本実施例で示した基本的な軌道生成（＋障害物登録）制御は、従来方式などで行なわれている時間軸方向の排他空

間制御を利用しないため、比較的高速に実行でき、過大な計算資源を必要とせず容易に実行することができる。

【００９２】

なお、本実施例（以下の実施例も同様）では、説明を容易にするため、主記憶ないし仮想記憶上に１つの障害物メモリが存在するような構成を示した。その場合、障害物データにはロボットアーム（５、６…）の識別データを含み、あるアームの軌道生成では、そのアーム自身の掃引領域に相当する障害物以外の他のアームの掃引領域に相当する障害物を回避するものとした。しかしながら、主記憶ないし仮想記憶上に１つの障害物メモリの実装には種々の形態が考えられ、当業者は障害物メモリの実装、およびそれを用いた制御の一部に関して種々の設計変更を行うことができる。

【００９３】

例えば、ロボットアーム（５、６…）ごとに障害物メモリを配置して、軌道生成の際、アームの掃引領域に相当する障害物を、当該のアーム以外の障害物メモリに追加していくような制御も考えられる。このような障害物メモリ実装によっても、当該ロボットアームに関して登録された障害物空間を除く障害物空間を回避するよう、当該ロボットアームを動作させる軌道を生成する軌道生成行程を実現することができる。

【００９４】

＜実施例２＞

上記実施例１では、ロボットアーム５、６の２つの動作指令から軌道生成を行う最も単純な例で、ロボットアーム（５、６）同士の干渉のない軌道を生成する手法につき説明した。実施例１の図５に示したようなステップ構成のみによって、より多数の軌道定義データについて軌道生成および障害物登録を繰り返す場合、登録された多数の障害物によって新たな軌道を生成できなくなる可能性がある。

【００９５】

本実施形態の基本的な制御では、生成軌道によってアーム躯体が掃引（通過）する領域全体を障害物として登録するために、軌道生成すべき動作指令が多いほど早期に作業空間の多くの部分が障害物で埋め尽される可能性がある。

【００９６】

通常、生産ラインのような現実の環境で動作するロボットにはより多数の作業が割り当てられる。例えば、単純な搬送動作を行う場合にしても、搬送物を取りに行く動作と置きに行く動作の２つに分かれる。つまり、作業者が入力する指令には、各ロボットに複数の動作が含まれることが想定される。そして、実施例１に示した基本的な制御では、複数台のロボットの通過領域が互いに交差しないような軌道を生成するために、ロボットが行う全動作の通過領域が、他のロボットの全動作の通過領域と交差しないように制御される。しかし、そのような軌道生成は、を作ることは、ロボットに割り当てられた動作が多いほど、また作業空間が狭いほど、困難な問題となる。

【００９７】

例えば図８（ｃ）は、ロボットアーム６の軌道Ｔｒｂ１（始点Ｓｂ１～終点Ｇｂ１）が生成され、対応する掃引（通過）領域に対応する障害物（ＳＶｂ１）が障害物メモリ２２０１に登録された状態を示している。一方、それ以降、ロボットアーム５に対して生成すべき（始点、終点）が（Ｓａ２、Ｇａ２）の動作指令が動作指令リスト中に存在しているとする。この動作指令の軌道定義、特に終点（Ｇａ２）は、図８（ｄ）に示すように既に障害物（ＳＶｂ１）の空間内に存在する。このままでは、動作指令中の（始点、終点）が（Ｓａ２、Ｇａ２）である軌道定義に対応するロボット軌道を生成することはできない。

【００９８】

以下、本実施例では、より多数の動作指令に応じて軌道生成を行う際、図８（ｄ）のような事象が生じた場合に、軌道生成および障害物登録を続行可能な制御手法を示す。また、その場合、生成した軌道によって、実環境のロボットアームを相互に干渉することなく動作させることができるような制御手法を示す。

【００９９】

本実施例２では、与えられた指令に対してロボット同士で互いに交差しない軌道を生成できない場合は、指令を２つに分割し、分割後の指令内ではロボット同士で互いに交差しないような軌道を生成する。

【０１００】

即ち、動作指令リスト中の複数の軌道定義データに基づき軌道生成を行うに際し、障害物メモリに登録された障害物空間を回避する軌道が生成不可能となる場合がある。その場合、動作指令リスト中の軌道定義データを、その時点で生成済みの軌道に対応する軌道定義データから成る第１の動作指令と、未生成の軌道に対応する軌道定義データから成る第２の動作指令と、に分割する。そして、障害物メモリに登録されている障害物空間のデータをクリアしてから第２の動作指令に含まれる軌道定義データについて軌道生成行程および障害物登録行程を繰り返し実行する。また、生成した全部の軌道データに基づき、各々のロボットアームを動作させる場合に第１の動作指令に対応するロボットアームの動作と、第２の動作指令に対応するロボットアームの動作と、を同期的に実行させるための同期指令を生成する。

【０１０１】

以下、図７～図１０、図１６（ａ）～（ｆ）を参照し、本実施例の制御につき説明する。以下では、上述の実施例１の図１、図２、図４などに示した軌道生成システムの全体構成は本実施例でも同様であるものとする。ロボットアーム５、６と障害物７の配置については、図４と同様である。

【０１０２】

図７は、実施例１の図５と同様な詳細な形式で本実施例の軌道生成および障害物登録手順の流れを示している。図８（ａ）～（ｄ）、図９（ａ）～（ｄ）は、軌道生成の様子を図４、図６と同様な詳細な形式で示している。図１０は、本実施例において軌道生成された各動作指令に対応する動作の同期的な制御の様子を示している。図１６（ａ）～（ｆ）は、図１５と同等の形式で本実施例の障害物メモリ２２０１のデータ内容の遷移を示している。以下では、実施例１で説明済みの部材や、処理ステップについては特に必要な場合を除き、重複した説明は省略するものとする。

【０１０３】

本実施例において、軌道生成および障害物登録手順の全体の流れは実施例１の図３と同様で、制御手順全体の流れは、動作指令の入力および動作指令リストの生成（Ｓ１０１）、優先度設定（Ｓ１０２）から始まる。その後、軌道生成（Ｓ１０３）、および障害物追加（Ｓ１０４）を行う。

【０１０４】

図７のステップＳ３００（動作指令の入力および動作指令リストの生成）、Ｓ３１０（優先度設定）、Ｓ３２０（ポインタ／カウンタの初期化）、Ｓ３３０（動作指令Ｍｉの軌道生成）は、図５のＳ２００、Ｓ２１０、Ｓ２２０、Ｓ２３０と同じ処理である。図７のＳ３５０（動作指令Ｍｉに対応するロボットアームの通過（掃引）領域の生成）、およびＳ３６０（通過（掃引）領域の追加登録）は、図５のステップＳ２４０、Ｓ２５０と同じ処理である。また、図７のＳ３７０（ループ終了判定）は、図５のステップＳ２６０と同じ処理である。

【０１０５】

図７で図５と大きく異なるのは、Ｓ３３０（動作指令Ｍｉの軌道生成）と、Ｓ３５０（動作指令Ｍｉに対応するロボットアームの通過（掃引）領域の生成）の間に、ステップＳ３４０～Ｓ３８０を追加している点である。

【０１０６】

ステップＳ３４０では、ステップＳ３３０において動作指令Ｍｉに対応する軌道が生成できたか否かを判定する。もし、障害物メモリ２２０１に登録済みの障害物の掃引空間によって、動作指令に対応する軌道が生成できない場合には、ステップＳ３８０に移行する。

【０１０７】

ステップＳ３８０では、生成済みの軌道に対応する軌道定義データから成る第１の動作指令と、未生成の軌道に対応する軌道定義データから成る第２の動作指令と、に分割する。そして、障害物メモリに登録されている障害物空間のデータをクリアしてから、ステップＳ３００に復帰する。この場合、ステップＳ３００、Ｓ３１０では、以後、動作指令リスト中の未生成の軌道に対応する軌道定義データから成る（第２の動作指令の部分を構成する）各動作指令の処理の優先順位を決定するよう制御する。

【０１０８】

以下、図８～図１０に示すような動作指令を例に図７の処理を説明する。

【０１０９】

図７のステップＳ３００では、ステップＳ１００（図３）ステップＳ２００（図５）と同様に動作指令の入力と、動作指令リスト生成が行われる。ここでは、作業者の入力に基づき生成された動作指令リストには、ロボットアーム５については３つの動作指令Ｍａ１、Ｍａ２、Ｍａ３が、ロボットアーム６については２つの動作指令Ｍｂ１、Ｍｂ２が含まれているものとする。

【０１１０】

図８（ａ）は、上記動作指令に対応する軌道定義（始点、終点）を示している。同図において、動作指令Ｍａ１、Ｍａ２、Ｍａ３は、それぞれ始点Ｓａ１、Ｓａ２、Ｓａ３、終点Ｇａ１、Ｇａ２、Ｇａ３の軌道定義に、また、動作指令Ｍｂ１、Ｍｂ２は、始点Ｓｂ１、Ｓｂ２、終点Ｇｂ１、Ｇｂ２の軌道定義に対応する。

【０１１１】

即ち、この軌道定義のうち、動作指令Ｍａ１は図８（ａ）の始点Ｓａ１から終点Ｇａ１（Ｓａ２）へ、Ｍａ２は始点Ｓａ２から終点Ｇａ２（Ｓａ３）へ、Ｍａ３は始点Ｓａ３から終点Ｇａ３へアームの基準部位を移動させるものである。ロボットアーム５はＳａ１の位置からスタートし、Ｇａ１、Ｇａ２の各位置を経由して、Ｇａ３の位置に向かう。

【０１１２】

また、ロボットアーム６の動作指令Ｍｂ１は図８の（ａ）の始点Ｓｂ１から終点Ｇｂ１へ、また、Ｍｂ２は始点Ｓｂ２から終点Ｇｂ２へアームの基準部位を移動させる動作指令である。この場合、ロボットアーム６はＳｂ１の位置からスタートし、Ｇｂ１の位置を経由して、Ｇｂ２の位置に向かう。この指令Ｏｐ１に対応する動作指令リストは、以下のような構成となる。

Ｏｐ１　＝　｛　Ｍａ１　、　Ｍａ２　、　Ｍａ３　、　Ｍｂ１　、　Ｍｂ２　｝

Ｍａ１　＝　（　Ｓａ１、　Ｇａ１　）、　Ｍｂ１　＝　（　Ｓｂ１　、　Ｇｂ１　）

Ｍａ２　＝　（　Ｓａ２、　Ｇａ２　）、　Ｍｂ２　＝　（　Ｓｂ２　、　Ｇｂ２　）

Ｍａ３　＝　（　Ｓａ３　、　Ｇａ３　）

図７のステップＳ３１０では、設定されている優先順位に基づき、軌道生成の順位を決定する。この軌道生成の優先順位（ロボット単位あるいは動作指令単位）の制御に関しては、上述のように種々の方式が考えられ、例えば、作業者の優先順位指定を許容することが考えられる。

【０１１３】

しかしながら、本実施例では、ＣＰＵ２０の自動制御によって、軌道生成の優先順位（ロボット単位あるいは動作指令単位）を設定するものとする。ここでは、動作指令リストでより多数の前記軌道定義データが含まれるロボットアームの軌道生成が、動作指令リストでより少数の前記軌道定義データが含まれるロボットアームの軌道生成よりも先に実行されるよう、優先度を設定する。この場合、上記の動作指令リストの構成では、動作数が多く、優先度の高いロボットはロボットアーム５となる。

【０１１４】

動作指令単位の優先順に関しては、例えば作業者の入力指定に応じて各動作指令の処理順を任意の順番に並べ換えることが考えられる。しかしながら、本実施例では、説明を容易にするため、上記の動作指令リストの順序（Ｍａ１、Ｍａ２…，あるいはＭｂ１、Ｍｂ２…）とする。

【０１１５】
ただし、本実施例では、複数のロボットアームのうち、軌道生成されていない動作指令が多数あるアームが生じないように、全てのアームを順に１つずつ軌道生成（＋障害物登録）する処理順制御も適用する。これは、１つのロボットだけ集中して軌道生成し、その通過領域によって他のロボットが軌道生成できなくなる問題を回避するためである。
【０１１６】
なお、以上のような本実施例の軌道生成（＋障害物登録）の処理順序の制御の一例は、図１８のフローチャートを用いて本実施例の最後で説明する。
【０１１７】
以上のような本実施例の軌道生成（＋障害物登録）の処理順序の制御によれば、各動作指令につき軌道生成を行う順番はＭａ１、Ｍｂ１、Ｍａ２、Ｍｂ２、Ｍａ３のような順序になる。
【０１１８】
図７のステップＳ３２０（ポインタ／カウンタの初期化）に続き、ステップ３３０では、ステップＳ３１０で定めた順番に従って軌道生成を行う。ステップ３３０の軌道生成は図５のステップＳ２３０で説明した処理と同様である。
【０１１９】
図７のステップ３４０では、ステップＳ３３０で軌道生成が成功したか否かを判定する。ここで軌道生成が成功した場合にはステップＳ３５０へ、軌道生成が生成できなかった場合にはステップＳ３８０へ進む。ここで、軌道が生成できない場合としては、例えば他のロボットアームの通過領域によって、終点までの道のりが塞がれてしまう場合や、他のロボットアームの通過領域内に始点や終点が含まれている場合がある。
【０１２０】
図８の例では、まず、ロボットアーム５の動作指令Ｍａ１が処理される。この場合、始点Ｓａ１から終点Ｇａ１に向かい、かつ障害物７を回避する軌道Ｔｒａ１が生成出来るため、処理ステップはＳ３５０となる。
【０１２１】
ステップＳ３５０、Ｓ３６０では、軌道による通過（掃引）領域（ＳＶａ１）から障害物データを生成し、障害物メモリ２２０１に登録する。これにより、障害物メモリ２２０１の状態は、図１６（ａ）の障害物７のみが存在する状態から図１６（ｂ）の登録状態となる。即ち動作指令Ｍａ１の軌道Ｔｒａ１によって、ロボットアーム５の通過（掃引）領域（ＳＶａ１）が新たに障害物として追加される。
【０１２２】
図８（ｃ）に示すように、上記同様の手順で、次の動作指令Ｍｂ１に関しても、軌道生成と障害物追加を行うことができる。動作指令Ｍｂ１もステップＳ３３０、Ｓ３４０で軌道が生成可能であり、図１６（ｃ）に示すように軌道Ｔｒｂ１に対応する通過（掃引）領域（ＳＶｂ１）が新たに障害物メモリ２２０１に追加される。
【０１２３】
さらに処理は進み、ステップＳ３３０でロボットアーム５に関して動作指令Ｍａ２の軌道生成が試行される。この動作指令Ｍａ２は図８（ａ）に示すように始点Ｓａ２から終点Ｇａ２へ向かう軌道定義を含む。しかしながら、図８（ｄ）に示すように、動作指令Ｍａ２の終点Ｇａ２の位置は、上記の手順で追加された障害物（ＳＶｂ１）の内部にあって障害物との干渉を生じるため、軌道生成を行えない。この場合、処理は図７のステップＳ３８０に進む。
【０１２４】
ステップＳ３８０で、ＣＰＵ２０は軌道が生成出来なかった場合の処理を行う。このステップＳ３８０では、まず、初めに、動作指令（群）を、軌道生成（＋障害物登録）済みの第１の動作指令（群：前半）と、軌道生成（＋障害物登録）が未処理である第２の動作指令（群：後半）に分割する。
【０１２５】

即ち、現在までに軌道が生成できた全動作指令を一つの第１の動作指令（群：前半）に、軌道生成（＋障害物登録）が未処理である以降の全動作を第２の動作指令（群：後半）に分割する。実施例２の例では、動作指令Ｍａ１、Ｍｂ１を第１（前半：Ｏｐ１－１）の動作指令に、動作指令Ｍａ２、Ｍｂ２、Ｍａ３を第２（後半：Ｏｐ１－２）の動作指令として分割する。

【０１２６】

なお、第１、第２の動作指令の分割は、１度のみならず、後半（残り）の第２の動作指令の処理で軌道生成が不可能になった場合は、さらに第２の動作指令を（処理済みの）第１の動作指令と、（未処理の）第２の動作指令に分割する。この繰り返しで、全動作指令データを処理していく。

【０１２７】

また、第１、第２の動作指令への分割が生じた場合、分割の前後で第１、第２の動作指令に対応するロボットアーム（５、６）の動作は同期的に実行する。即ち、第１の動作指令に対応する動作が全て終了するのを待って、第２の動作指令に対応する動作を開始させるような同期制御を行う。このような同期制御を行うため、動作指令の分割を行った場合、ＣＰＵ２０は、実環境でのロボットの動作を上記のように同期させるための同期指令（図１０のＳｐ１）を生成する。この同期指令（図１０のＳｐ１）は、例えば生成した軌道データ群の分割の起きた位置に挿入する。

【０１２８】

そして、ＣＰＵ２０は、図１６（ｄ）に示すように、軌道生成に伴なって生成されたロボットアームの掃引（通過）領域に対応する障害物データを障害物メモリ２２０１からクリアする。この場合、動作指令Ｍａ１、Ｍｂ１の軌道生成で追加された障害物ＳＶａ１、ＳＶｂ１が削除される。これにより、次の指令内の動作の軌道生成が可能になる。

【０１２９】

その後、ステップＳ３００に復帰して、上記同様の処理を繰り返し、残りの未処理の第２の動作指令（群：後半）に関して軌道生成（＋障害物登録）を実行する。なお、第２の動作指令（後半：Ｏｐ１－２）に相当する軌道定義から処理するための措置として、ＣＰＵ２０が、ステップＳ３８０において動作指令リストの処理済みの部分を消去するような処理を行うものとしてもよい。

【０１３０】

上記のように動作指令Ｏｐ１を分割した第１の動作指令（前半：Ｏｐ１－１）、第２の動作指令（後半：Ｏｐ１－２）は、

Ｏｐ１－１　＝　｛　Ｍａ１　、　Ｍｂ１　｝　、

Ｏｐ１－２　＝　｛　Ｍａ２　、　Ｍｂ２　、　Ｍａ３　｝

のような構成となる。なお、ＣＰＵ２０は、上記のような動作指令分割を行う場合、実環境において、分割した第１（前半）、第２（後半）の動作指令の各動作を同期的に制御するための同期指令（図１０のＳｐ１）を生成する。この同期指令（図１０のＳｐ１）は、例えば生成した軌道データ群の分割の起きた位置に挿入するものとする。

【０１３１】

その後の処理において、ステップＳ３００では、ステップＳ３８０を経由したことにより、処理対象は第２の動作指令（後半：Ｏｐ１－２）に相当する軌道定義となる。即ち、上記の動作指令Ｍａ２、Ｍｂ２、Ｍａ３を含む動作指令Ｏｐ１－２が入力される。

【０１３２】

図９（ａ）は、第２の動作指令（後半：Ｏｐ１－２）の処理の初頭における現在のロボットアームの位置と動作指令Ｍａ２、Ｍｂ２、Ｍａ３の始点（Ｓａ２、Ｓｂ２、Ｓａ３）と、終点（Ｇａ２、Ｇｂ２、Ｇａ３）の位置を示している。

【０１３３】

続く図７のステップ３１０では、上記同様に軌道生成の順序を決定する。上記同様に、例えば動作数からロボットアームの優先度を設定し、優先度から処理の順番を決定する。第２の動作指令（後半：Ｏｐ１－２）の処理においても、ロボットアーム５の方が動作数

が多いため優先度が高く設定される。従って、軌道生成処理の順序は、動作指令Ｍａ２、Ｍｂ２、Ｍａ３の順となる。

【０１３４】

以下、ＣＰＵ２０は、第２の動作指令（後半：Ｏｐ１－２）に対して、上述同様に図７のステップ３３０～３６０によって軌道生成行程と障害物登録行程を実行する。最初の動作指令Ｍａ２は始点Ｓａ２から終点Ｇａ２へ向かう動作であり、図９の（ｂ）に示すように軌道生成により、ロボットアーム６の現在位置であるＳｂ２の点を回避するようにして、軌道Ｔｒａ２が生成される。また、図１６の（ｅ）に示すように、軌道Ｔｒａ２におけるロボットアーム６の通過（掃引）領域（ＳＶａ２）が障害物として障害物メモリ２２０１に追加される。

【０１３５】

図７のステップ３３０～３６０によって、さらに後続の動作指令Ｍｂ２、Ｍａ３…が処理される。動作指令Ｍｂ２の始点Ｓｂ２、終点Ｇｂ２については、追加された障害物ＳＶａ２を回避するように軌道Ｔｒｂ２が生成される（図９（ｃ））。また、その軌道の通過（掃引）領域に対応する障害物（ＳＶｂ２）が生成され、障害物メモリ２２０１に追加される（図１６（ｆ））。そして、最後に、動作指令Ｍａ３の始点Ｓａ３、終点Ｇａ３から障害物ＳＶｂ２を回避する軌道Ｔｒａ３が生成され（図９（ｄ））、その軌道の通過（掃引）領域に対応する障害物（ＳＶａ３）が生成される。

【０１３６】

以上のようにして、作業者が最初に入力した動作指令リスト中の全動作指令に対応する全軌道の生成が完了する。本実施例の軌道生成方法によれば、アームの通過（掃引）領域に対応する障害物によって軌道生成が不可能となった場合、処理済みの動作指令を含む第１の動作指令と、未処理の動作指令を含む第２の動作指令とに動作指令リストを分割する。その際、障害物メモリの障害物データをクリアして残りの第２の動作指令を処理するため、軌道生成（＋障害物登録）を続行することができる。

【０１３７】

一方、上記のような軌道生成によると、実環境の動作では、分割した動作指令Ｏｐ１－１とＯｐ１－２の間ではロボットアーム同士の干渉の可能性があるため、同期的な実行制御を行う必要がある。例えば、第１の動作指令Ｏｐ１－１内の全動作が終了した後に、第２の動作指令Ｏｐ１－２内の動作を実行する制御を行う。このために、上記のように、動作指令の分割を行った場合、ＣＰＵ２０は同期指令（図１０のＳｐ１）を軌道データ中の同期制御が必要な位置に挿入していく。

【０１３８】

図１０は、上記のように分割した（第１の）動作指令Ｏｐ１－１と、（第２の）Ｏｐ１－２に対応するロボット動作をロボットアーム５、６で同期的に実行させた時のタイミングを示している。この例は、図８、図９で説明した（第１の）動作指令Ｏｐ１－１と（第２の）動作指令Ｏｐ１－２に対応している。このように（第１の）動作指令Ｏｐ１－１と（第２の）動作指令Ｏｐ１－２との分割を行なった場合、ＣＰＵ２０は軌道データ中に同期指令（Ｓｐ１）を挿入する。これにより、（第２の）動作指令Ｏｐ１－２に対応する動作が（第１の）動作指令Ｏｐ１－１の動作が全て終了するのを待って開始されるよう、同期的な制御を行うことができる。

【０１３９】

図１０では、最初に２つのロボットアーム５、６が同時にそれぞれの動作を開始する。この場合、どちらかのロボットアームが先に動作を終了するが、同期指令（Ｓｐ１）が挿入されている場合には、その次の指令の動作は実行せず、もう片方のロボットアームの動作が完了するまで待機する。図１０の例では、ロボットアーム６が先に動作（Ｍｂ１）を完了するが、待機時間８０の間だけ待機し、ロボットアーム５の動作指令Ｍａ１が完了するのを待つよう、同期制御が行われている。そして、動作指令Ｏｐ１－１内の全動作が完了した後に、次の動作指令Ｏｐ１－２に移り、それぞれの動作（Ｍａ２、Ｍｂ２、Ｍａ３…）を同時に実行開始する。

【０１４０】
なお、動作指令Ｏｐ１－２と後続の動作指令（不図示）との間でも動作指令の分割（および障害物メモリ２２０１のクリア）が生じている場合には、図１０に示すように同期指令（Ｓｐ２）が挿入される。ここでは、ロボットアーム６が先に全動作を完了するが、ここでも待機時間９０の間だけロボットアーム５を待つ同期制御が行われる。
【０１４１】
以上のように、本実施例では、分割後の指令の中では複数のロボットアームが干渉し合わない軌道を保証し、指令間の干渉はロボットアームを同期指令に基づき同期的に動作させる。これにより、本実施例の方法によれば、ロボットアームに多数の動作を割り当てられた場合においても、同期指令を発生して実環境における簡易な同期制御を行うことにより、ロボット同士の干渉の生じない軌道を自動的に生成することができる。
【０１４２】
ここで、図１８に図７のステップＳ３１０（図３のステップＳ２１０でも同様）に関して述べた、ロボットアーム単位および動作指令単位の優先度を用いて、軌道生成（＋障害物登録）の処理順を制御するアルゴリズムの一例を示しておく。
【０１４３】
ここでは、ロボットアーム単位の優先度を指すポインタ(カウンタ)Ｒを用いる。このＲは、最大優先度に対応する値０から最大（全）ロボットアーム数に相当するR_Mまで１ずつインクリメントされる。一方、ポインタ(カウンタ)Ｒにより指示される主記憶ないし仮想記憶上に確保されたロボットアーム順位リストには、ユーザ（作業者）指定（あるいはＣＰＵ２０が自動決定）に基づき、各ロボットアームの識別データを優先順位が高い方から順に格納しておく。このロボットアーム順位リストを、ポインタ(カウンタ)Ｒの値に基づき先頭（０）から参照していくことで、ＣＰＵ２０はロボットアーム順位リストに定義されたアームの順で軌道生成（＋障害物登録）を実行できる。なお、ポインタ(カウンタ)Ｒは、図１８の例では循環的に使用され、その値が最大（全）ロボットアーム数に相当するR_Mに達すると、その値は０にリセットされる。
【０１４４】
また、以上では、ポインタ(カウンタ)ｉを用いた動作指令単位の優先度制御については詳細に述べなかったが、この動作指令単位の優先度制御についても、上記と同様の構成を用いることができる。例えばポインタ(カウンタ)ｉにより指示される主記憶ないし仮想記憶上に確保された動作指令順位リストに、ユーザ（作業者）指定（あるいはＣＰＵ２０が自動決定）に基づき、各動作指令の識別データを優先順位が高い方から順に格納しておく。この動作指令順位リストを、ポインタ(カウンタ)ｉ（最も高い優先度は０）の値に基づき参照していくことで、動作指令順位リストの定義順で動作指令を処理していくことができる。
【０１４５】
図１８は図７のステップＳ３２０～Ｓ３７０の部分をより詳細に示す形式のフローチャートで、ステップＳ３２０は図７のものと同じポインタ(カウンタ)ｉの初期化（ｉ＝０）処理である。また、それに続くステップＳ５２０は、上記のポインタ(カウンタ)Ｒの初期化（Ｒ＝０）処理である。
【０１４６】
ステップＳ５３０では、Ｍｉ＝Ｍｉ（Ｒ）のようなポインタ（カウンタ）演算によって、今回の処理対象の動作指令を決定する。つまり、このポインタ（カウンタ）操作は、ポインタ(カウンタ)Ｒの指すロボットアームに関する、現時点で最も優先度の高い（未処理）の動作指令を特定する処理である。
【０１４７】
ステップＳ５４０では、図７のステップＳ３３０～Ｓ３６０の処理に相当する軌道生成および障害物登録処理を行う。ここでは詳細不図示であるが、軌道生成不可能となった場合のステップＳ３４０～Ｓ３８０の動作指令分割処理も同様に実行することができる。
【０１４８】

ステップＳ５５０では、ポインタ（カウンタ）Ｒ、およびポインタ（カウンタ）ｉを１ずつインクリメントする（Ｒ＝Ｒ＋１，ｉ＝ｉ＋１）。これによりロボット指定、および動作指令は、次のロボットアーム、次の動作指令をそれぞれ指すようポインタ（カウンタ）Ｒ、ｉがそれぞれインクリメントされる。

【０１４９】

ステップＳ５６０では、ポインタ（カウンタ）Ｒが一巡したか、即ち、最大（全）ロボットアーム数に相当する$Ｒ_{M}$に達したか否かを判定する。ステップＳ５６０が肯定された場合にはポインタ（カウンタ）Ｒは最初の最も優先度の高いアームを指すように０にリセットされる。

【０１５０】

ステップＳ５８０は、図７のステップＳ３７０と同じ処理で、ここではポインタ（カウンタ）ｉの値の判定によって、動作指令リスト中の全ての動作指令を処理したか否かを判定している。ここで全動作指令を処理していない間はステップＳ５３０に復帰して上記の動作を繰り返す。

【０１５１】

図１８のような優先度処理によって、ポインタ（カウンタ）Ｒをインクリメントすることによって、全部のロボットアームを設定された優先度の順に１つずつ指定し、またポインタ（カウンタ）ｉによって動作指令を優先度の順に取り出して処理できる。従って、ロボットアーム単位の優先順位制御については、優先順位に基づき、まんべんなく全部のロボットアームの動作指令を設定された優先度の順で処理できる。このため、例えばユーザ（作業者）のロボット優先順位の指定に偏りがあって、あるロボットアームの動作ばかり先に処理され、後続のロボットアームの動作の処理で軌道生成が進まないような事態を回避することができる。

【０１５２】

＜実施例３＞

実施例２で軌道生成が不可能となる事態として、例えば図８（ｄ）のように、軌道生成の終点Ｇａ２（始点の場合も同様である）が、他のロボットアームの軌道生成によって生成された障害物（ＳＶｂ１）に最初から干渉している例を示した。本実施例３では、このような事態を回避できるような処理手法につき説明する。本実施例でも、図１～図４、図１４などに示した軌道生成システムの基本構成は上述の実施例１、２と同様であるものとし、既に説明済みの部材や制御については可能な限り重複した説明は省略するものとする。

【０１５３】

本実施例３では、各ロボットアームが動作指令中の軌道定義である始点および終点の位置にいる時の姿勢において各アームの躯体が占める空間を障害物として登録しておくことにより上記の問題を回避する。これにより、動作指令の始点または終点が最初から他のロボットアームの通過領域内に含まれてしまうような状態を回避することができる。

【０１５４】

以下、実施例３での複数台のロボットの通過領域が互いに交差しない軌道を生成する方法について、図１１～図１３、図１７を用いて、本実施例の処理手法につき説明する。図１１（ａ）～（ｆ）は実施例２の図８と同様の形式で各ロボットアーム（５、６）の姿勢および対応する軌道生成（＋障害物登録）の様子を示している。図１２は、実施例２の図１０に相当する実環境における同期的なロボット制御の様子を示し、図１３は、図７と同様の形式で本実施例における軌道生成（＋障害物登録）の制御手順を示している。また、図１７（ａ）～（ｆ）は、図１５、図１６と同等の形式で本実施例の障害物メモリ２２０１のデータ内容の遷移を示している。

【０１５５】

以下では、実施例３の構成や、処理対象の動作指令リスト中の動作指令は実施例２と同様であるものとする。このため、以下では特に実施例２と異なる点に重点を置いて説明する。

【０１５６】
本実施例において、作業者の入力に応じて生成された動作指令リストは、実施例２と同様であり、
Ｏｐ１　＝　｛　Ｍａ１　、　Ｍａ２　、　Ｍａ３　、　Ｍｂ１　、　Ｍｂ２　｝
Ｍａ１　＝　（　Ｓａ１　、　Ｇａ１　）、Ｍｂ１　＝　（　Ｓｂ１　、　Ｇｂ１　）
Ｍａ２　＝　（　Ｓａ２　、　Ｇａ２　）、Ｍｂ２　＝　（　Ｓｂ２　、　Ｇｂ２　）
Ｍａ３　＝　（　Ｓａ３　、　Ｇａ３　）
のような動作指令によって構成されている。上記の各始点と終点の位置は図１１（ａ）に示す通りである。

【０１５７】
図１３は本実施例における軌道生成（＋障害物登録）処理の流れを示しており、ここでは実施例２の図７と同一の処理ステップには図７と同一のステップ番号を用いている。図１３で図７と異なるのは、ステップＳ３００とＳ３１０の間にステップＳ４１０を挿入してある点である。それ以外のステップの処理は、図７で説明したものと同様である。

【０１５８】
この図１３のステップＳ４１０では、全ての軌道生成を開始する前に、各ロボットアームが動作指令中の始点および終点の位置にいる時の姿勢で各アームの躯体が占める空間を障害物として障害物メモリ２２０１に登録する。図１７（ａ）は、このステップＳ４１０が実行された後の状態を示している。即ち、図１７（ａ）では、上記の各動作指令の始点（Ｓａ１、Ｓａ２…、Ｓｂ１、Ｓｂ２…）および終点（Ｇａ１、Ｇａ２…、Ｇｂ１、Ｇｂ２…）における姿勢において各アームの躯体が占有する領域が障害物として障害物メモリ２２０１に登録されている。

【０１５９】
これ以降の処理の流れは、図７で説明したものと同様である。例えばステップＳ３１０の処理の優先順序の決定方法も実施例２と同様に行われ、ここでは上記の動作指令Ｏｐ１の処理順序はＭａ１、Ｍａ２、Ｍａ３、Ｍｂ１、Ｍｂ２であるものとする。

【０１６０】
次に、ステップＳ３２０の初期化に続き、ステップＳ３３０～Ｓ３７０（およびＳ３８０）のループによって、設定された順位に従って、軌道生成および障害物追加処理を行う。この時、図１７（ａ）のような障害物の初期登録状態によって、本実施例の軌道生成（ステップＳ３３０）では、処理対象のアーム以外の他のアームの始点と終点の位置を回避するよう軌道が生成されることになる。

【０１６１】
この軌道生成および障害物の追加登録の様子は、例えば図１１（ａ）～（ｆ）、図１７（ｂ）～（ｆ）に示す通りである。ここでは、まず動作指令Ｍａ１の軌道生成時は予め配置されている障害物７を避けるようにして軌道Ｔｒａ１が生成され、その掃引（通過）領域が障害物（ＳＶａ１）として障害物メモリ２２０１に追加される（図１１（ｂ）、図１７（ｂ））。

【０１６２】
続いて、動作指令Ｍａ２の軌道生成時は、既に障害物登録されているロボットアーム６の終点Ｇｂ１（始点Ｓｂ２）の位置を回避するように軌道Ｔｒａ２が生成される（図１１の（ｃ））。また、その軌道Ｔｒａ２によるロボットアーム６の掃引（通過）領域が障害物（ＳＶａ２）として障害物メモリ２２０１に追加される（図１７（ｃ））。これにより、実施例２の制御による場合と異なり、後続の動作指令Ｍｂ１の軌道が生成不可能となることが防止される。

【０１６３】
さらに、同様にして動作指令Ｍａ３の軌道Ｔｒａ３がロボットアーム６の終点Ｇｂ１（始点Ｓｂ２）の位置を回避するように生成されその通過（掃引）領域が障害物（ＳＶａ３）として障害物メモリ２２０１に追加される（図１１（ｄ）、図１７（ｄ））。

【０１６４】

また、それに続くロボットアーム６の軌道生成時は、実施例２と同様に、先の手順で追加された通過領域を回避するように軌道生成が行われる。これは通過領域内に他のロボットアームの始点と終点が含まれるためである。従って、ロボットアーム６の軌道生成では、まず動作指令Ｍｂ１の軌道Ｔｒｂ１が生成され、その通過（掃引）領域が障害物（ＳＶｂ１）として障害物メモリ２２０１に追加される（図１１（ｅ）、図１７（ｅ））。そして、動作指令Ｍｂ２については、軌道Ｔｒｂ２（図１１の（ｆ））が生成され、対応する通過領域が障害物（ＳＶｂ２）障害物メモリ２２０１に追加される（図１７（ｆ））。

【０１６５】

以上により、動作指令Ｏｐ１として与えられた全動作の軌道生成が完了する。図１２は、以上のようにして軌道生成を行った２つのロボットアーム５、６を動作させた時のタイムチャートである。上記の動作指令Ｏｐ１の場合、本実施例では、動作指令の始点および終点のアーム躯体の姿勢に相当する空間を障害物として予め軌道開始前に登録しているため、図１３のステップＳ３４０～Ｓ３８０の分岐が生じていない。このため、図１０に示したような同期制御を行う必要がなく、同図に示すように、各ロボットアーム５、６が同時に各々の動作を開始し、他のロボットアームの動作に拘らず与えられた動作の軌道を実行してもアーム同士の干渉を生じない。

【０１６６】

即ち、本実施例３によれば、（いわば確定済みの障害物として）動作指令の始点および終点のアーム躯体の姿勢に相当する空間を、予め（先に）障害物登録してしまい、それを回避するように軌道修正を行う。このため、実施例２で説明した動作指令の分割、障害物データのクリアおよび実環境においてアーム動作を同期的に制御するための同期指令の生成、の各過程を発動させる確率が下がり、アーム同士の干渉のない軌道生成をスムーズに行うことができる。そして実環境で生成した軌道データによりロボットアームを動作させた場合でも、実施例２で示したような同期制御が必要となる確率が下がり、高速にロボットアームを動作させることができる。

【０１６７】

以上に示した実施例中の具体的な構成はあくまでも一例に過ぎず、特許請求の範囲を限定するものではない。特許請求の範囲に記載の技術には、以上に例示した具体例を様々に変形、変更したものが含まれ、当業者は特許請求の範囲内で様々な設計変更を行うことができるのはいうまでもない。

【０１６８】

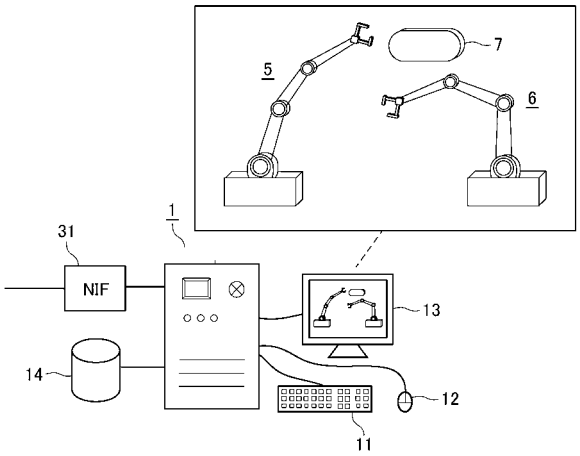
本発明は、上述の実施例の１以上の機能を実現するプログラムを、ネットワーク又は記憶媒体を介してシステムまたは装置に供給し、そのシステムまたは装置のコンピュータにおける１つ以上のプロセッサがプログラムを読出し実行する処理でも実現可能である。また、１以上の機能を実現する回路（例えば、ＡＳＩＣ）によっても実現可能である。

【符号の説明】

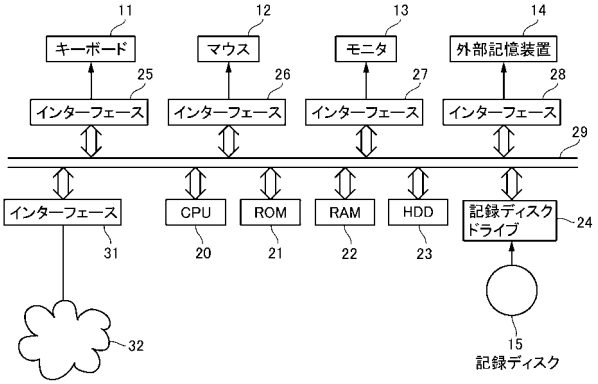
【０１６９】

１…演算処理部、１１…軌道生成部、２…入力部、３…記憶部、５…ロボットアーム（Ａ）、６…ロボットアーム（Ｂ）、１１…キーボード、１２…マウス、１３…ディスプレイ、５１、５２、５３、６１、６２、６３…関節軸、５４、６４…エンドエフェクタ、５５、５６、５７、６５、６６、６７…リンク、７…障害物、８０、９０…待機時間。

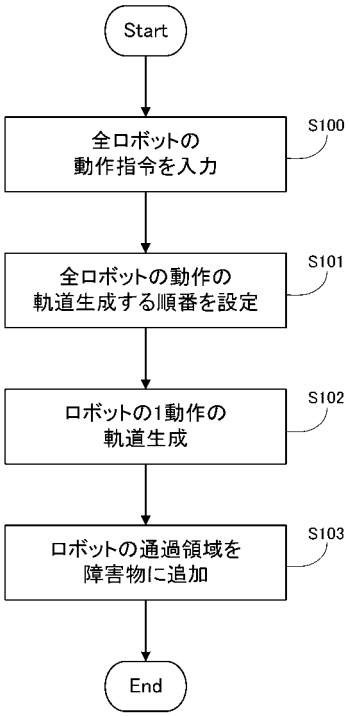
【図１】


5
7
6
1
31
NIF
13
14
12
11

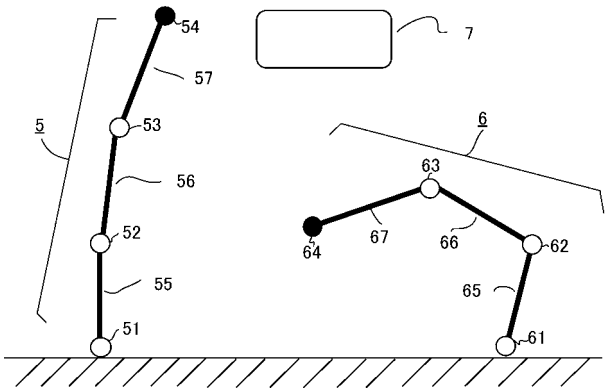
【図２】


11
キーボード
12
マウス
13
モニタ
14
外部記憶装置
25
インターフェース
26
インターフェース
27
インターフェース
28
インターフェース
29
インターフェース
31
CPU
20
ROM
21
RAM
22
HDD
23
記録ディスクドライブ
24
32
15
記録ディスク

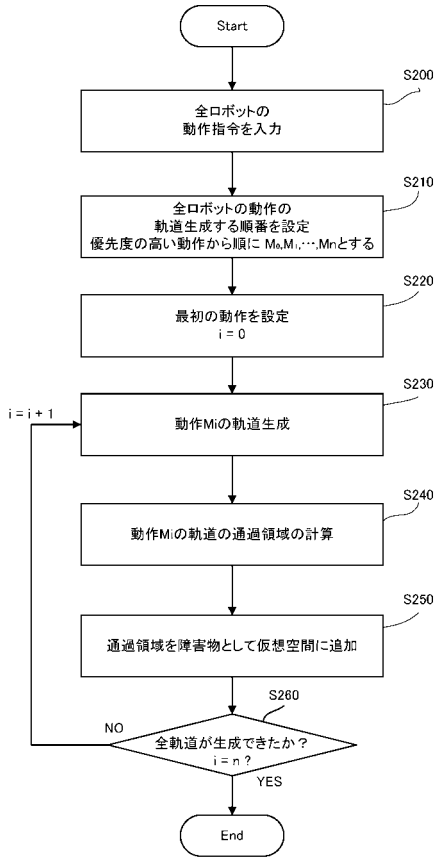
【図３】


Start
全ロボットの
動作指令を入力
S100
全ロボットの動作の
軌道生成する順番を設定
S101
ロボットの1動作の
軌道生成
S102
ロボットの通過領域を
障害物に追加
S103
End

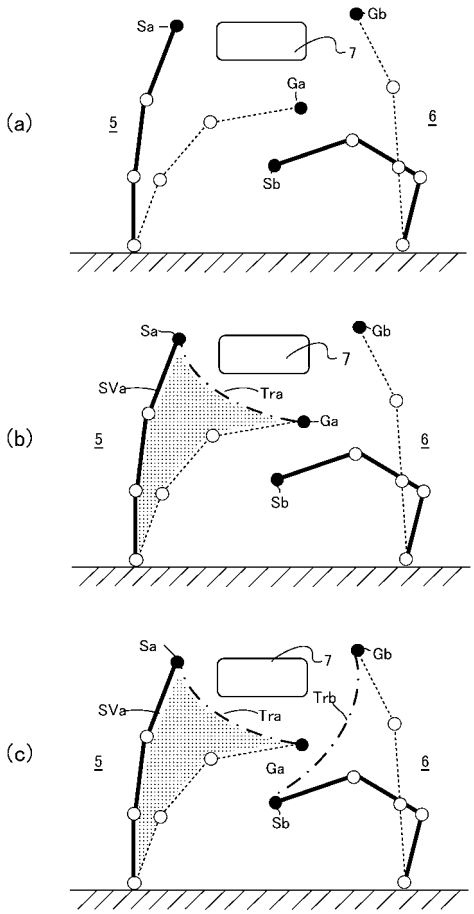
【図４】


54
7
57
5
53
6
56
63
52
64
67
66
62
55
65
51
61

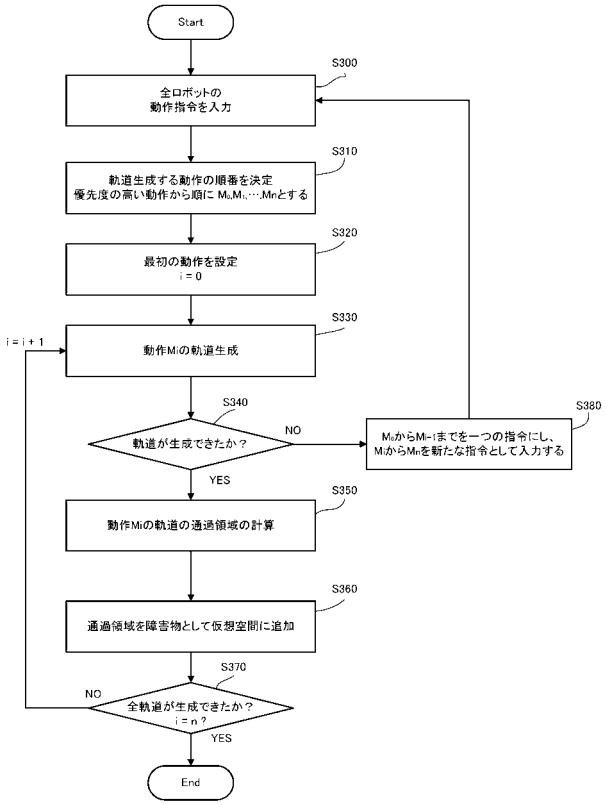
【図５】


Start
S200
全ロボットの
動作指令を入力
S210
全ロボットの動作の
軌道生成する順番を設定
優先度の高い動作から順に M0,M1,…,Mnとする
S220
最初の動作を設定
i = 0
S230
動作Miの軌道生成
i = i + 1
S240
動作Miの軌道の通過領域の計算
S250
通過領域を障害物として仮想空間に追加
S260
NO
全軌道が生成できたか？
i = n ?
YES
End

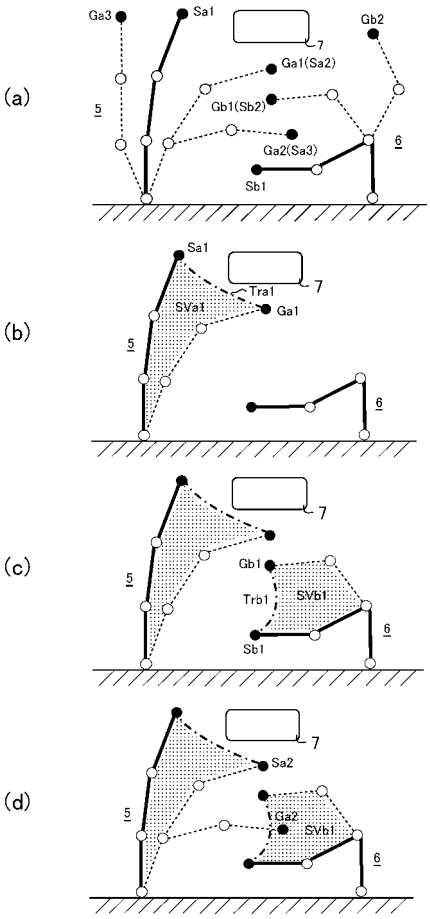
【図６】


(a)
Sa
Gb
7
Ga
5
6
Sb
(b)
Sa
Gb
7
SVa
Tra
Ga
5
6
Sb
(c)
Sa
7
Gb
SVa
Trb
Tra
5
Ga
6
Sb

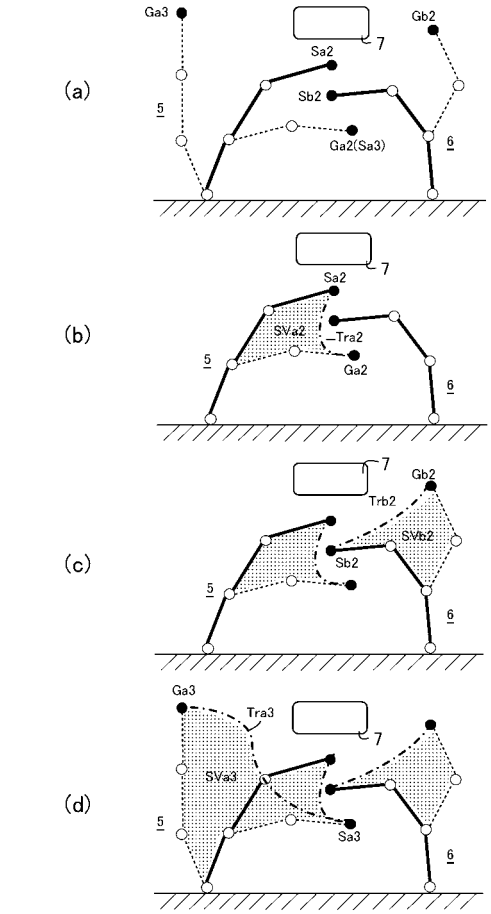
【図７】


Start
S300
全ロボットの
動作指令を入力
S310
軌道生成する動作の順番を決定
優先度の高い動作から順に M0,M1,…,Mnとする
S320
最初の動作を設定
i = 0
S330
動作Miの軌道生成
i = i + 1
S340
軌道が生成できたか？
NO
S380
M0からMi-1までを一つの指令にし、
MiからMnを新たな指令として入力する
YES
S350
動作Miの軌道の通過領域の計算
S360
通過領域を障害物として仮想空間に追加
S370
NO
全軌道が生成できたか？
i = n ?
YES
End

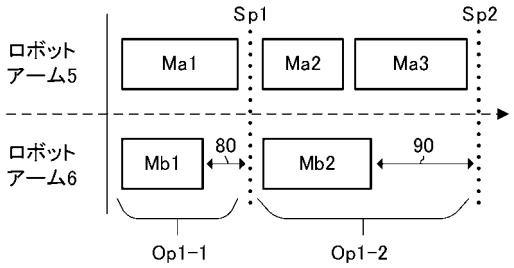
【図８】


(a)
Ga3
Sa1
7
Gb2
Ga1(Sa2)
5
Gb1(Sb2)
6
Ga2(Sa3)
Sb1
(b)
Sa1
7
Tra1
SVa1
Ga1
5
6
(c)
7
Gb1
5
Trb1
SVb1
6
Sb1
(d)
7
Sa2
5
Ga2
SVb1
6

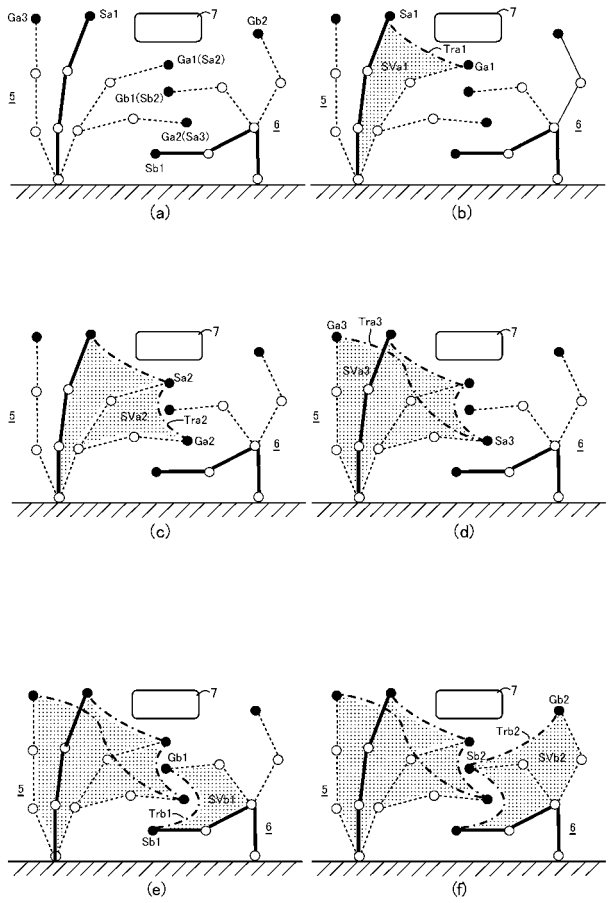
【図９】



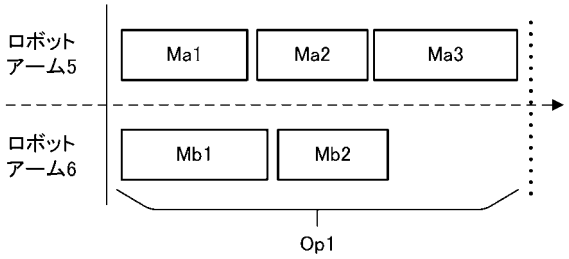
【図１０】



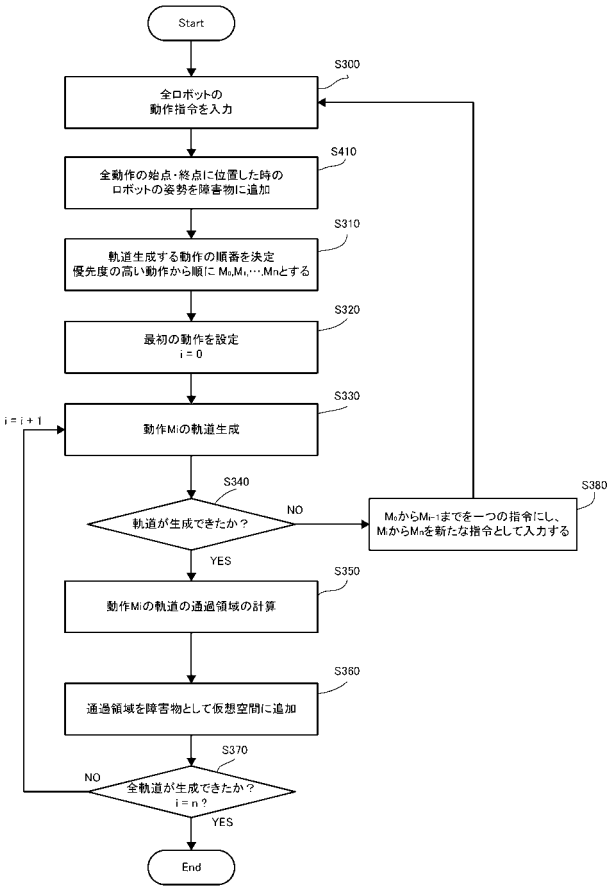
【図１１】



【図１２】

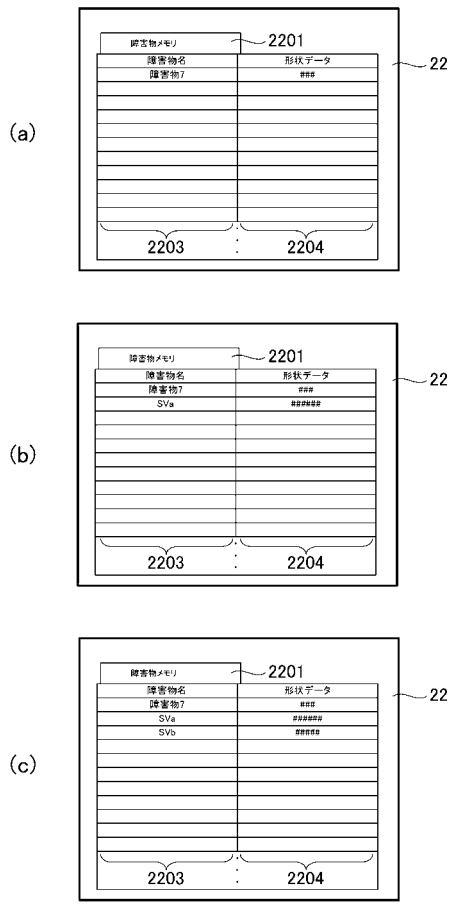


【図１３】

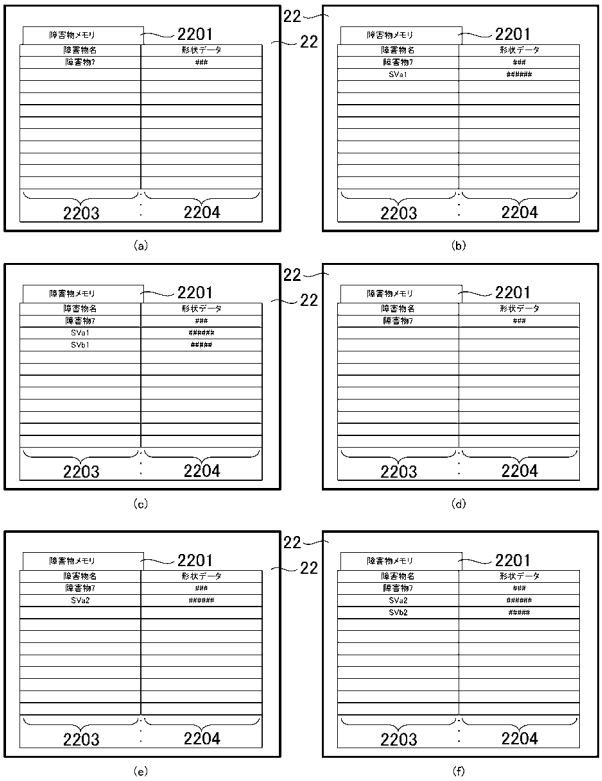


【図１４】

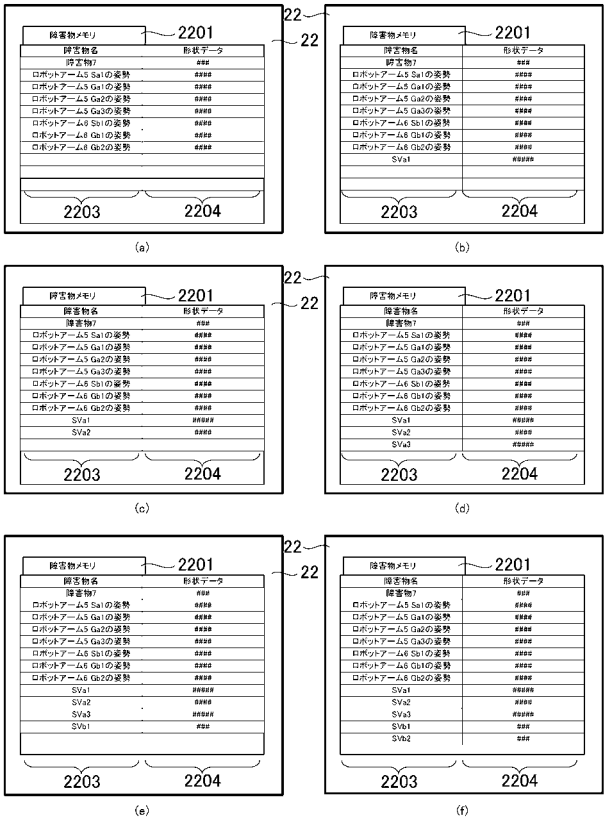
【図１５】



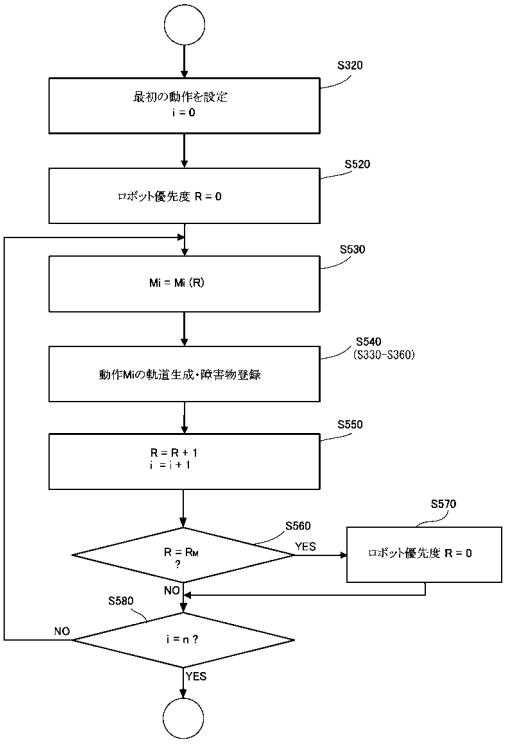
【図１６】



【図１７】


障害物メモリ
2201
22
障害物名
形状データ
2203
2204
(a)
(b)
(c)
(d)
(e)
(f)

【図１８】


最初の動作を設定
i = 0
S320
ロボット優先度 R = 0
S520
Mi = Mi (R)
S530
動作Miの軌道生成・障害物登録
S540
(S330-S360)
R = R + 1
i = i + 1
S550
R = RM ?
S560
YES
ロボット優先度 R = 0
S570
NO
S580
i = n ?
NO
YES