事例教示による未整備環境における ロボットの移動操作ルールの自動生成

小方 博之

目次

1 序論

1.1 研究の背景

ロボットを用いることは、製造業などの産業を合理化し企業競争力を高める上で有力な手段となっている。ロボットはその汎用性ゆえに、専用機械に比べて格段にフレキシビリティに富んだ動作を行い、産業の省力化や効率化に大きく貢献してきた。ロボットが専用機械と一線を隔するのは、単一の作業に用途が制限されず、必要に応じて使い回しがきき、作業が変化する毎に機械の入れ替えをしなくて済む点である。それゆえ、ロボットはさまざまな作業をこなす能力を持ち、組立、溶接、塗装などの分野で、いわゆる産業用ロボットとして積極的に導入されてきた。

一般的な産業用ロボットの作業はロボットコマンドをプログラムすることで実現される。教示者はそのためにティーチング・ペンダントを用いて実地に動作を示したり [67]、プログラムをテキストとして編集したり [44]、オフラインでシミュレータを用いたりする [68]。ロボットは手先位置や関節角の具体的な値でプログラミングされた動作を正確に再現することで作業を行ない、整備された工場のラインで行なわれる

単純作業にその効果を発揮している。このように、プログラミングされた動作をロボットが正確に再生する方式はガイディング方式と呼ばれる。

ガイディング方式は長らく実用として用いられる唯一の方式であった。しかし、ロボットの用途が広がるにつれて、ガイディング方式では実現できない作業の実行も必要となってきている。

例えば、自動組立では部品整列やビンピッキングといった作業が必要とされている。工場で産業用ロボットを用いて作業する場合には、部品がロボットに正しく把持されるように、部品を整列しておくか、ロボット自身が正しく把持できる能力を持つ必要がある。現状では小型の部品はパーツフィーダによって振動や回転を利用して方向を揃えることができる。また、大型の部品は整列された状態で工場に搬入されるケースが多い。しかし、中間的な大きさの部品ではパーツフィーダの使用は難しく、自動化の遅れている分野であり、研究課題として盛んに取り組まれている。

また、多品種少量生産もガイディング方式の苦手とするところである。多品種少量生産は時計や乗用車など、ファッション性を重視する製品を製造する現場で長年望まれている。また、近年でも情報技術の発展に伴い、需要に素早く対応するストックレス生産を実現するためにも注目されている生産形態である。多品種少量生産でバリエーションに富んだ製品を効率的に製造するには、部品の共有化や設計の再利用を促進する必要がある。部品の共有化や設計の再利用が進むと、教示コスト削減のため、わずかに異なった状況に対してロボットのプログラムを再利用する類型組立技術も必然的に求められる。また、少ロット生産の場合には部品固定用のジグは用意できない。したがって、同じ部品の組立を行なう場合でも、部品の位置の不確定性を前提としなければならない。

さらに、最近ではペットロボットやアミューズメントロボット、巡回ロボット、清掃ロボットなど、パーソナルユース、サービス業、福祉・医療分野などの非産業用途

を指向したデザインや機能を持つロボットの発表が相次いでいる [64][65]。これらのロボットの機能はまだ限定されたものだが、少子化、高齢化、社会の成熟化などの要因から、今後このような用途を指向したロボットが工場外で活躍することへの期待の高さがうかがえる。市場調査会社の中には、このような非産業用途のロボットの市場が 10 年後には 800 億円の規模に成長すると予測するところもある [62]。

これらの分野でロボットが本領を発揮するには、センサを用いてロボット用に整備されていない環境の不確定性に対処しながら作業を行う能力を簡単に実現できる技術が求められる。

このような作業の実現をめざして、ロボットの能力を拡張しようと、さまざまな研究がなされてきた。一つは、センサ情報を判断条件とするルールによって直接行動を決定する方式である。この方式は決定が速く、環境の変動に対する適応性もよいが、行動決定が近視的である。それゆえ、ロボットがゴールに到達できないおそれがある。また、複雑な作業ほどルールの設計が困難になる。ルールを学習によって自ら獲得するロボットの研究も行われているが、実用的な作業を行うには、学習の効率化など、まだ解決すべき問題が残されている。

別のアプローチとして、センサ情報から作業環境のモデルを構築し、その中でゴールに到達するのに最適な行動を計画する方式も存在する。しかし、これを実現するにはセンサによって環境の十分精度の高いモデルを作成する必要がある上、そこで最適な行動を計画する計算量も爆発的に増大する問題もある。

このように、未整備環境でロボットが効率的に作業を実現するための実用的なルール作成方法はなかなか存在しないのが現状である。

1.2 本研究の目的

上述の点を踏まえて、ここでは未整備環境でロボットが効率的に作業を実現するための実用的なルール作成方法の実現を目的とする。

1.3 本論文の構成

第１章は序論として、本研究の背景と目的を述べた。第２章では本テーマに関連した研究を概観し、それらとの関連を述べることで本研究の位置付けを明確にする。

第３章では本論文で対象とする移動操作作業の定義、未整備環境のモデル化を行なう。また、作業を記述するルールを、操作を行なう形態によって行動選択方式、状態遷移方式に分類し、それぞれの特徴を説明する。

第４章では、ルール記述に行動選択方式を用いた場合に、移動操作作業を実行するためのルールを作成する方法を検討し、実験によりその有効性を確認する。また、第５章では状態遷移方式を用いた場合の方法を検討し、有効性の確認を行なう。

第６章は結論であり、本研究の成果をまとめる。

2　研究の動向と課題

2.1　はじめに

第1章では環境の不確定性に対処しながら作業を実行する方法を確立することがロボットを将来さまざまな作業に適用していく上で重要であることを指摘した。本章では、ロボットが作業を行なうことを想定して整備されていないような環境で、ロボットが作業を実行していく方式を論じた研究の現状と課題について述べる。

第2.2節では上述のような作業を実現するために従来どのような研究が行なわれてきたか概観し、それらの特徴にしたがって方式を分類し、各方式について説明する。

第2.3節では本論文が未整備環境における作業に対して想定する条件を述べ、その作業を実現するときに従来の研究が直面する問題について論じる。更にその解決をめざすための本論文の取り組みについて説明する。

2.2　研究の動向

第1章で述べたように、ロボットは、それに合わせて整備されていない、未整備環境と呼ばれる環境でも、さまざまな作業への使用が期待されている。第1章では、そ

のような作業の例として部品整列、ビンピッキング、多品種少量生産における類型組立、巡回、清掃などを挙げた。これらの作業では、ロボットが初期位置から目標位置に手先や本体を移動する、いわゆる移動操作が多く見られる。そこで、ここでは作業対象として未整備環境における移動操作を中心に論じる。

未整備環境で作業を行う場合は、センサで状況を判断しながら適応的に行動することが必要である。これを実現するために、さまざまな方式の研究が進められている。

センサで得られた知覚から行動を決定するのはルールを設けることで行われる。ここでは、そのルールの適用の深さと作成方法の2点に着目することで、これまでの研究を分類する。ルールの適用の深さに関してには、得られた知覚から一つのルールを適用することで素早く行動を選択する浅い方式から、ルールによって選択した行動の結果を何段階も予測してゴールまでの最適な行動を考えるような深い方式まで存在する。ルールの作成方法に関しては、設計者が手作業で作成する方法と、ロボットに自動的に作成させる方法とがある。これらの観点から、ここではこれまでの研究をプランニング方式、リアクティブ方式、行動学習方式の3種類に分類する (Fig. 2.1)。ルールの適用が深く、かつルールがロボットによって自動作成されるようなシステムの研究例はあまり見られないので、ここでは分類から除外する。

以下ではプランニング方式、リアクティブ方式、行動学習方式を順次解説し、それらを踏まえて、移動操作作業の実現に必要な技術要素について検討する。

2.2.1 プランニング方式

プランニング方式は、ルールによって選択した行動の結果を何段階も予測してゴールまでの最適な行動を考えることを特徴とする。

SMPA モデル [10] はこの方式を実現する典型的な方法である。 SMPA は Sense-Model-Plan-Action の頭文字をとったものである。 SMPA モデルに基づいたシステムは、そ

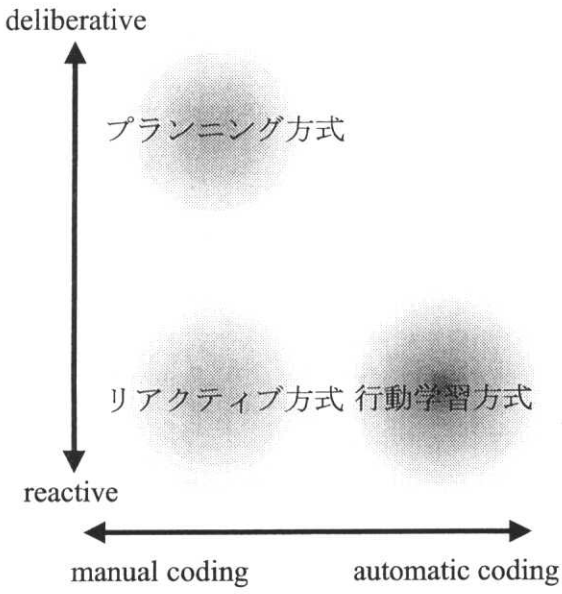


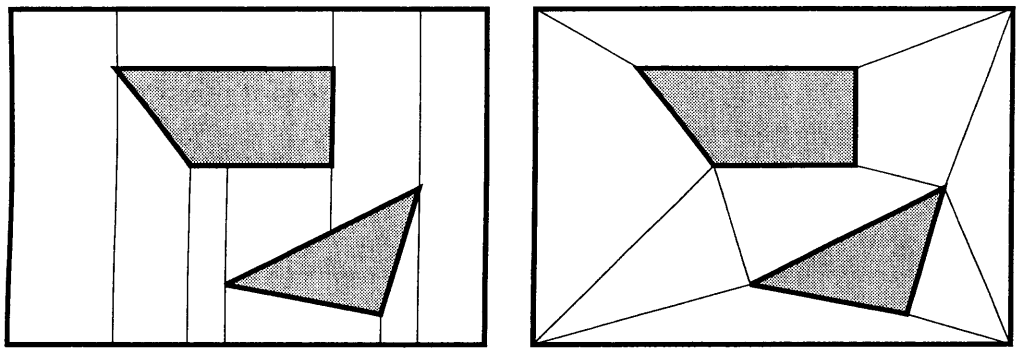
図 2.1: 従来研究の分類

の名前のとおり、作業を開始する前に観察結果から環境全体のモデルを作成し、モデルの中でゴールに効率的に到達する手順を計画し、手順に基づいて作業を実行するというプロセスを踏む。環境全体のモデルを作成するのは行動結果を将来にわたって予測できるようにするためである。

SMPA モデルは自律移動ロボットの研究のはしりとなった SRI の Shakey プロジェクト [46] から盛んに用いられている。 Shakey では天井カメラで得られた画像を環境のモデルとして用いることを想定する。そして、四分木を用いて障害物の存在する部分と存在しない部分を分離し、四分木による分割で生じたセルを、ロボットが障害物と衝突せずに目的地に向かうための経路のプランニングに利用している。

モデルとして四分木のセルを用いることは必ずしも経路の効率的生成に適していないことから、その他にも空間を効率的に構造化するさまざまな手法が考えられている。例えば、 Fig. 2.2 (a) のように頂点に対して垂直に空間を分割して台形または三

角形領域を作成する台形分割法がある [14]。また、 Fig. 2.2 (b) のように頂点と頂点または辺を結んでチャンネルと呼ばれる境界線を作り、互いに重ならない凸多角形に分割するチャンネル分割法も提案されている [19]。これらのように空間を部分空間に分割し、互いの隣接関係を明確にすることで、空間構造をグラフで表現できる。



(a) 台形分割法　　(b) チャンネル分割法

図 2.2: 空間構造化手法の適用例

空間をグラフで表現すると、経路のプランニングにグラフ探索の手法を利用できる。グラフ探索の手法としては、探索のために展開するノードを最小に抑えながら、コスト最小の解を求めることができる A* アルゴリズム [21] などが用いられる。コストとしては距離、ロボットの消費エネルギ、走行時間などを用いることが多い。

このようにして求められた部分空間を経由するような行動を出力することで、ロボットはモデルの上で最適な方法でゴールに向かうことが可能になる。

このように、効率的な行動を生成できるプランニング方式であるが、プランニングに要する計算コストが大きいため、環境の変化に対する即応性がよくないという欠点を持つ。さらに、悪いことに、計算コストはロボットの動作自由度が高くなるにつれて爆発的に増大する。例えば、 Reif は任意の次元のおいて有限個の多面体障害物の存在する環境で多面体から構成されたアームをスタートの姿勢からゴールの姿勢に衝

突せずに移動させる経路を解く問題がPSpace困難であることを示した[51]。同様に、プランニング方式により行動を決定することの困難性を示す報告が多数なされている[11][22][27]。このような問題もあって、マニピュレータのように動作自由度の高いロボットに対するこの手法の適用は未だ実用の域に達していない。また、あらかじめ環境全体のモデルを作成するので、環境情報取得コストが高く、適用できる環境に制限がある。

2.2.2 リアクティブ方式

リアクティブ方式はプランニング方式とは逆に、設計者の用意したルールを用いて、得られた知覚から一つのルールを適用することで素早く行動を選択する。リアクティブ方式は直接ルールを適用する単純な方法に基づいているため、ある程度の水準の作業を比較的簡単に実現できる。例えば、石川はファジー理論を用いたルールを構築し[25]、有人環境で巡回を行うロボットを開発した。しかし、より複雑な作業に適用できるルールの構築には労力を要する。一般にルールの数は知覚の個数乗に比例して増加し、ルールの増加につれて行動の整合性を保つのが困難になる。実際、上述のロボットについても障害物回避だけのために１５５個のルールを要したと報告されている。また、ルールが増加するにつれて整合性の問題のほかにも、処理が複雑なルールを用いるとロボットの応答が遅くなるなどの問題がある。

リアクティブ方式のシステムを構築するもう一つの方法論としてsubsumption architectureが提案されている[9]。アリの歩行動作の様に一見高度な判断力を要すると思われてきた行動でも、単純な反射行動を用いて説明できる[55]という考え方をもとに、知的な行動を簡単にロバストに実現しようとするものである。Subsumption architectureの特徴は、ルールを記述したモジュールが階層構造になっており、処理が並列に行われる点にある。階層構造をとることで、容易な機能拡張が可能になるとし

ている。活性化したルール間に不整合が生じた場合には上の階層のモジュールが優先され、下の階層のモジュールの出力が抑制される。処理を並列にすることで、処理の複雑なモジュールを組み込んでも全体としての応答が遅くならない利点がある。

リアクティブ方式を用いたシステムはプランニング方式と比較して環境に対する適応能力に優れており、障害物回避や脚歩行で歩き回る場合にその効果が示されている。また、その場で得られた知覚に基づいて行動を決定するので、環境情報取得コストも格段に低い。しかし、ゴールをめざすなど、より高次の目的行動については、その可能性や効率性がしばしば問題とされている。これに対して現在の状況と目的の両面からルールの活性化と抑制を行い作業の達成をめざすシステムが提案されている [40]。また、Herbert のように実際にソーダ缶拾いのような作業を行うようなロボットも製作されている [12]。しかし、このような作業を実現するには、ルールの作成に設計者の深い洞察や試行錯誤が必要になり、多大な手間を要するのが実状である。

2.2.3 行動学習方式

上述のプランニング方式やリアクティブ方式とは別に、ロボットが自ら行動することで作業の知識を帰納的に学習していくアプローチも研究されている [18][34]。このアプローチを行動学習方式と呼ぶ。

行動学習方式でルールを自動的に作成するのにしばしば採用される方法に強化学習がある。強化学習は、ロボットが行動で得た評価値から入力知覚と出力行動の間の適切な関係を強化していくことで状況に対してふさわしい行動選択能力をロボットに付与しようというものである。これによって簡単なセンサを備えたロボットに壁沿い走行や、障害物を回避しながらの走行を実現させられることが示されている [28][41][59]。また画像からいくつかの特徴量を抽出してロボットにサッカーボールのシュートを実現した研究もある [5]。

行動学習方式は、リアクティブ方式と同様に、入力知覚から通常一つのルールを適用することで素早く行動を選択するために、環境適応能力に優れる。また、ロボットの実際の行動結果からルールを形成していくので、設計者にはリアクティブ方式のように作業に関する深い知見が必要なく、ハンドコーディングの手間も存在しない。しかし、入力知覚の数や出力行動のバリエーションが増加すると、強化すべき入力知覚と出力行動の関係の数が爆発的に増大するので、学習に時間がかかるようになる。また、学習の繰り返しによるロボットの消耗も問題になる。そのため、より複雑な行動を要する作業を実現することは困難である。上述のシュートロボットの研究では学習を計算機上のシミュレータで行うことでこれらの問題の回避を試みている。しかし、シミュレータを構築するには作業がモデル化されなければならず、設計者に作業に関する深い知見が必要な点でリアクティブ方式と変わらなくなる。

2.2.4 その他の方式

この他にも、上述の方式を組合せることで、互いの欠点を補うことをめざした複合的方式が提案されている。

例えば、Brock and Khatib は未知環境での移動ロボットのゴール誘導について論じた [8]。基本的には dynamic window という概念を用いて移動ロボットの動力学的要因を考慮して局所的な障害物回避をその場で得られたセンサ情報に基づいて行ない、ゴールに向かう。同時に、センサ情報から自由空間の接続関係を取得して環境モデルを作成し、大域的な観点からゴールに向かう手順を決定するので、ロボットの停留を回避できるというものである。

Connell and Mahadevan はロボットの箱押し作業のルール作成を効率的に行なうためにサブタスクに分解して、各々のサブタスクを強化学習させる方法を述べている [13]。

Arkin はロボットのナビゲーションを行なうのに、同様にサブタスク分解を用いている [4]。あらかじめ与えられた環境に関する知識を用いて道順をプランニングし、走行を実現するためのモータやセンサのスキーマに落し込む。

このように複数の方式を組み合わせることで、プランニング方式の環境適応性を改善したり、リアクティブ方式や行動学習方式におけるルールの作成コストを低減できるので、ロボットの移動操作に関する能力を向上可能なことが報告されている。

2.3 本論文のねらい

ここでは本論文で想定する作業条件について述べるとともに、その条件下で適当な作業実現手法のあり方を考察する。

ロボットが未整備環境で移動操作作業を行なう場合、本論文ではその作業条件を以下のように考える。

- **事前知識の有無**

  未整備環境の場合、ロボットはあらかじめ環境モデルを用意しておくことは不可能なので、センサで随時環境の情報を取得して、作業を実行していく必要がある。しかし、類型組立作業や部品整列作業では、環境に関する事前情報が全くないとする必要はなく、部品形状の情報などある程度既知と仮定することが可能である。

- **センサの能力**

  センサを用いて必要なすべての情報をあらかじめ環境から引き出すには膨大なコストが必要であり、現実的ではない。ここではセンサによって取得可能な環境情報には限界があるとする。

- **環境の変動**

  プランニング方式では、作業実行中は環境は完全に静的であることを前提とし

ている。しかし、これは前提としては非常に強いものであり、一般の作業に適用しにくい。一方で、類型組立作業や部品整列作業では、サッカーボールを追う環境のように激しく変動する環境まで考慮する必要はなく、行動学習方式やリアクティブ方式よりも変動の緩やかな環境を想定できる。ここでは、ローカルな範囲でロボットがセンサで情報を取得し行動をしても、そこでは作業に影響するような変化が生じない程度の変動を想定する。その範囲の外では人間による操作などのダイナミックな変動は許容される。

- **作業の複雑性**

  移動ロボットによる作業の場合は2次元3自由度環境を考えれば十分である。一方、類型組立作業や部品整列作業が対象の場合は、3次元6自由度環境で、現実的な作業を行なう能力が必要である。

このように、本論文の想定する作業条件は、プランニング方式と行動学習方式・リアクティブ方式の中間に位置するものといえる (Fig. 2.3)。

このような作業条件に対しては、従来の方式を直接適用して作業を実行するのは困難である。

例えばプランニング方式を適用した場合にはセンサの能力に制限がある場合には環境全体のモデルを作成するのが困難である。また、静的でない環境で、ある程度複雑な作業を実行する場合には、作業動作を生成するのにコストをかけることはできない。

リアクティブ方式は比較的低自由度の移動ロボットで主に用いられているものであり、上述の条件下で適用するには設計者のルールのコーディングの負担を低減する工夫が必要である。また、行動学習方式を適用する場合も必要な試行錯誤を抑制しロボットの消耗を防止する工夫が必要となる。

そこで、本論文ではこのような作業条件下で移動操作作業を実現するための新た

な方式を検討する。

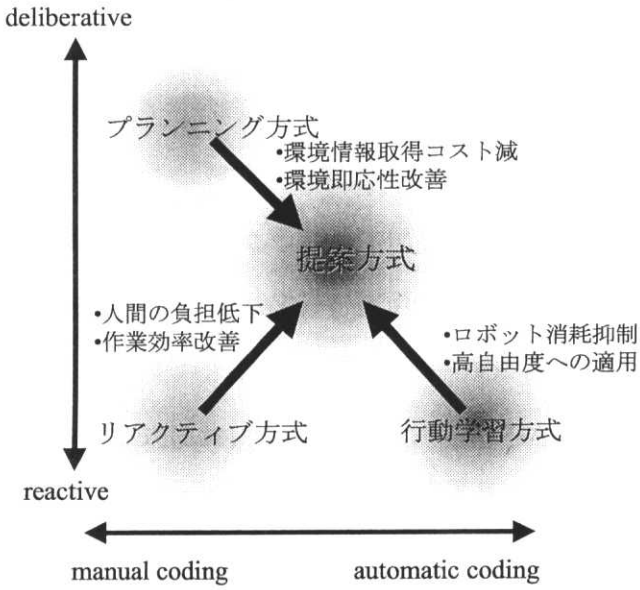


図 2.3: 提案方式の位置付け

そのために、ここでは人間がそのような場面で使用する戦略を真似ることを考える。

類型組立や部品整列等の作業を未整備環境で行う場合は、人間は物体に関するトポロジカルな知識と観測した情報とをうまく組み合わせて作業できる。また、人間は移動操作については何らかの知識を持っており、プランニング方式のように行動決定のための爆発的な量の計算を行うために手を休めたりすることはない。

そこで、ロボットに対しても、上のような人間の方式にならい、物体に関する知識を適当な形で与え、適切な観測によって環境から必要な情報を取得して補うようにし、人間の移動操作に関する知識と同等のルールを用意できれば、ロボットも未整備環境において計算爆発を起こすことなく適切な移動操作を生成できると考えられる。

しかし、そのようなルールはこれまで明示的に書き下された例はなく、リアクティ

ブ方式のようにハンドコーディングで作成する手法は適用できない。環境の次元も大きいので、行動学習方式の手法を流用して試行錯誤によってルールを自動生成することもできない。

また、人間は作業に関する明示的で無矛盾な知識を有するとは限らない [48]。

それでも、人間はそのような知識を持たない作業でも実例を示すことはできる。実例は、知識の一部が環境に依存した形で具体化されたものと考えられる。そこで、さまざまな環境で、知識のさまざまな部分を使用するような実例を収集できれば、その中から共通点を抽出して人間の知識を反映したルールが作成可能と考えられる。

これは、リアクティブ方式において設計者がいろいろな事例を考慮し、整理することでルールを作成するときの手間よりも小さい。また、ロボットが試行錯誤でいろいろな無駄な動きをするよりも消耗が小さい。

最近、データベースに蓄積された大量のデータを処理して、そこに隠された有用なルールを抽出する KDD (Knowledge Discovery in Database) またはデータ発掘 (Data Mining) と呼ばれる一分野が大きな発展を見せている [1][63] [43]。本研究では人間が示した作業の実例を計測し、そのデータをデータベースに蓄積し、 KDD 等で用いられる手法を応用して分析を施すことで移動操作に関する知識をルールとして抽出することを試みる。

第 3章では物体に関する知識や移動操作作業の記述方法を検討する。第 4章と第 5章では、その議論に基づいて 2 つの異なるモデルについてルールの抽出方法を論じる。

2.4 おわりに

本章では、ロボットを未整備環境における作業に適用していくことを目的とした研究の動向と課題を述べるとともに、本研究の取組みの概要を説明した。

第 2.2節では、ルールの適用深さと作成方法に着目して、従来の研究をプランニング方式、リアクティブ方式、行動学習方式に分類した。そして、各方式について説明

を行ない、それらの特徴を明らかにした。

第 2.3節では、類型組立作業や部品整列作業のように、環境変動の度合いが極端でなく、ある程度環境に対する事前知識が得られる作業を行なうためには、プランニング方式、リアクティブ方式、行動学習方式の中間的な方式が望ましいことを指摘した。そして、そのような方式を実現するために、人間の教示した実例から作業の実行ルールを抽出する方式を提案した。

3　移動操作作業のモデル

3.1　はじめに

第2章では、ロボット用に整備されていない環境において作業を実行するための研究の動向と課題について述べ、その課題を解決するための本論文の取組みの概要を説明した。本章では、本論文が対象とする移動操作作業のモデルについて説明し、問題の定式化を行なう。

第3.2節では本論文で考える未整備環境における移動操作作業のモデルについて述べる。

第3.3節では、第3.2節で説明したモデルに基づき、ロボットの状態空間で作業を記述する方法を論じる。

第3.4節では、第3.3節の方法で記述されたモデルをロボットが移動操作に用いる方法を検討する。

第3.5節では、第3.4節の方法を実現するルールを実例教示から作成する方法について述べる。

第 3.6節では作成したルールを有効に再利用する方策を述べる。

第 3.7節では上述のアプローチを実現するための基本的なシステム構成について論じる。

3.2 未整備環境における移動操作作業のモデル

3.2.1 移動操作作業

ここでは、本論文で考える移動操作作業について説明する。第 2章では移動操作を含む作業例として搬送、案内、組立、部品整列等を挙げた。

本論文では、移動操作作業は操作対象となる物体を、操作手段となる装置を用いて、初期状態から目標状態に、環境に存在する他の物体と接触や干渉をさせることなく、移動させる作業と考える (Fig. 3.1)。ここでは、操作対象となる物体を操作物体といい、環境に存在する接触や干渉回避の対象となる物体を環境物体と呼ぶ。操作手段となる装置がロボットである。

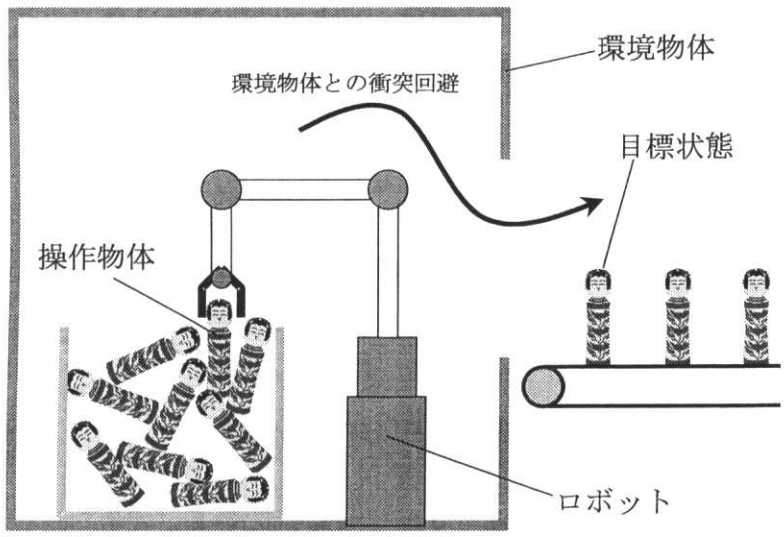


図 3.1: 移動操作作業のイメージ

例えば、移動ロボットが搬送や案内を行なう作業では操作手段となる装置は移動ロボットで、操作物体は移動ロボット自身とその付属物（搬送対象となる荷物など）である。それ以外の物体は環境物体である。また、マニピュレータで組立や部品整列を行なう作業では操作手段となる装置はマニピュレータで、操作物体は組立や整列を行なうパーツである。環境物体はマニピュレータとパーツ以外の物体となる。

本論文では、未整備環境における移動操作作業を考えるので、

- 類型組立等における物体の形状の不確定性
- ビンピッキング、部品整列、ジグレス作業等における物体の位置・姿勢の不確定性

といった環境の不確定性を表現する必要がある。そこで、環境物体には1からnまでの番号を振り、i番目の環境物体に対して特徴ベクトルθ_iを考える。環境における、i番目の環境物体の占有領域はθ_iによって$Shape_i(\theta_i)$で表現されるとする。環境の不確定性は、作業毎のθ_iの値の違いによって表現する。

また、操作物体はロボットが位置制御を行なうことで操作される。ロボットの状態ベクトルを$\mathbf{q}$とし、環境におけるロボットと操作物体の占有領域が$Shape(\mathbf{q})$で表現されるとする。

操作物体とロボットが、1からnまでの環境物体と接触や干渉をしない状態は以下の式で表現される。

$$Shape(\mathbf{q}) \cap (\cup_i Shape_i(\theta_i)) = \phi$$

また、経路パラメータをsによって初期状態と目標状態におけるロボットの特徴ベクトルをそれぞれ$\mathbf{q}(s_0)$, $\mathbf{q}(s_f)$と表現する。s_0からs_fまでロボットが連続的に移動した場合の$Shape(\mathbf{q}(s))$の掃引領域を$Sweep(\mathbf{q}(s)|s = [s_0, s_f])$で表現する。移動操作

作業の定義からその作業動作は以下の式を満たすような s でなければならない。

$$Sweep(\mathbf{q}(s)|s=[s_0,s_f]) \cap (\cup_i Shape_i(\theta_i)) = \phi$$

3.2.2 目標状態

プランニング方式では静的環境を前提とし、作業前の観測によってすべての特徴ベクトル θ_i を測定できると考えるので、動作開始の時点で目標状態 $\mathbf{q}(s_f)$ は定数ベクトルとして扱える。

一方、本論文の場合には、環境は完全に静的ではなく、作業前の観測によって θ_i がすべて観測できることを前提としないので、$\mathbf{q}(s_f)$ は周囲の環境物体に依存した変数となる。そこで、ここでは目標状態 $\mathbf{q}(s_f)$ を θ_i による条件式で表現することになる。ただし、$\mathbf{q}(s_f)$ は θ_i のすべての値に依存するとは限らない。目標状態の具体的な記述方法は第 3.4.1節で述べる。

3.2.3 センシングを考慮したモデル化

ロボットはセンサによって環境から特徴ベクトル θ_i を観測し、環境モデルを作成する。本研究では、環境の不確定性に対処するために、ロボットは観測によってローカルな環境モデルを作成し作業実行することを繰り返すことで移動操作作業を行なう形式を考える。これは以下の考えによる。

- 一度の観測によってすべての θ_{ij}(θ_i の成分) が得られ、環境モデルが確定できると考えるのは現実的ではない。センサの使用にはコストが伴うので、検出する θ_{ij} を最小限にとどめる方がよい。

- ある時点で移動操作に影響する環境物体の占有領域情報はローカルなものに限定される。

- 移動操作作業全体が完了するまでに θ_i に変化が生じない保証はないので、移動の個々の操作に先立って必要な θ_{ij} のみを観測し、ローカルな環境をモデル化する方が現実的である。

ある時点で移動操作に影響する環境物体 i の占有領域情報は $Shape_i(\theta_i)$ 全体とは限らないので、上では θ_i の各成分 θ_{ij} によって表現している。以下では環境物体 i において特徴量 θ_{ij} によって占有領域の確定する部分を $Shape_{ij}(\theta_{ij})$ によって表す (Fig. 3.2)。

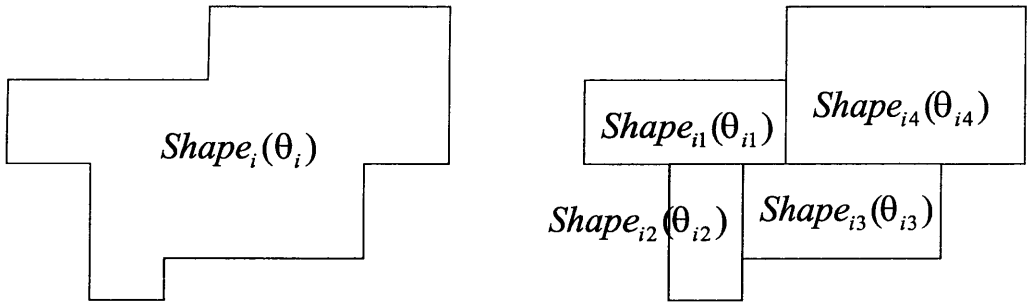


図 3.2: 特徴量による占有領域の分割

上の形式で作業を行なうには、ロボットの状態ベクトル $\mathbf{q}$ に対して、その時点でモデルを確定すべき環境物体の占有領域の範囲を明確にする必要がある。ここでは拘束域という概念を用いてそれを行なう (Fig. 3.3)。拘束域とは、ある特徴量 θ_{ij} で確定する環境物体の占有領域情報が、ロボットの動作に影響を及ぼす $\mathbf{q}$ の範囲を表した領域である。ここでは拘束域を $CArea(\theta_{ij}, \mathbf{q})$ と記述する。拘束域は特徴量 θ_{ij} のほかにも環境の変動の激しさやロボットのセンシング可能範囲を勘案して決められる。環境の変動の度合いが大きいときは拘束域は小さくなり、ロボットのセンサの能力が高く、より多くの特徴量が検出可能であったり精度よく検出できる場合には拘束域は大きく設定できる。

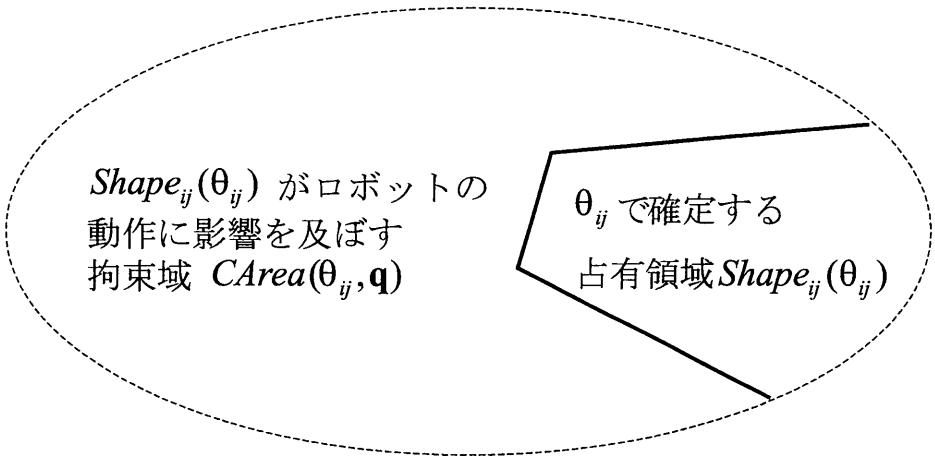


図 3.3: 拘束域の摸式図

ここでは、現在の $\mathbf{q}$ に対して $CArea(\theta_{ij}, \mathbf{q})$ を確定するのに必要な θ_{ij} はすべて観測か推定ができるように拘束域が設定されているものとする。

ロボットの状態ベクトル $\mathbf{q}$ に対して、操作に影響する拘束域は単数の場合もあれば複数の場合もある。拘束域が複数の場合にはそれらを考慮してローカルな環境モデルを作成する必要がある (Fig. 3.4)。また、影響する拘束域の組み合わせが異なる場合には、そこで行なうべき適切な操作も異なってくる。

拘束域の概念を導入した場合、ロボットは以下の手順で移動操作作業を実行する (Fig. 3.5)。

1. センサで環境物体の特徴量 θ_{ij} の検出を行う。

2. θ_{ij} から現在ロボットの存在する拘束域 $CArea(\theta_{ij}, \mathbf{q})$ を列挙する。

3. 列挙した $CArea(\theta_{ij}, \mathbf{q})$ に関わる $Shape_{ij}(\theta_{ij})$ を用いてローカルな環境モデルを作成する。

4. 環境モデルにしたがって移動操作を行う。

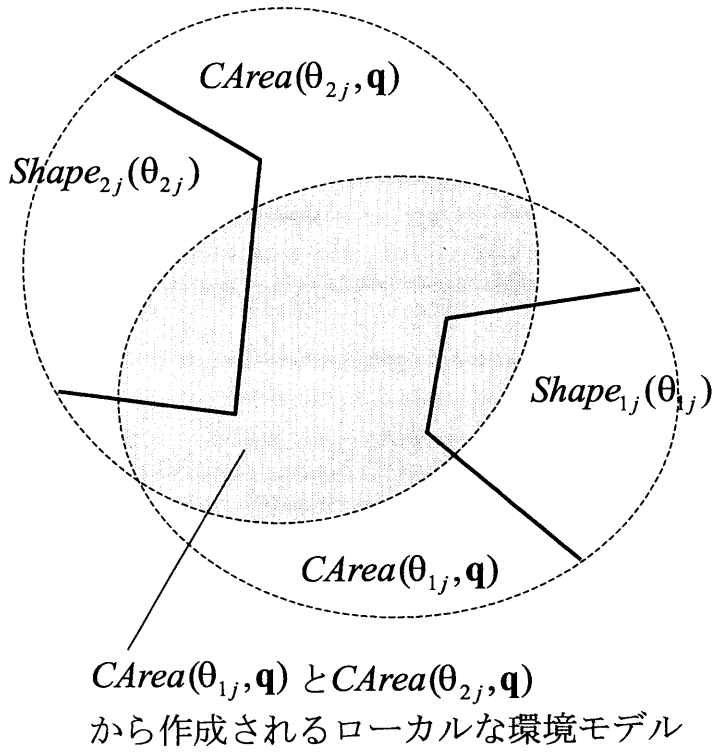


図 3.4: 複数の拘束域から作成されたローカルな環境モデル

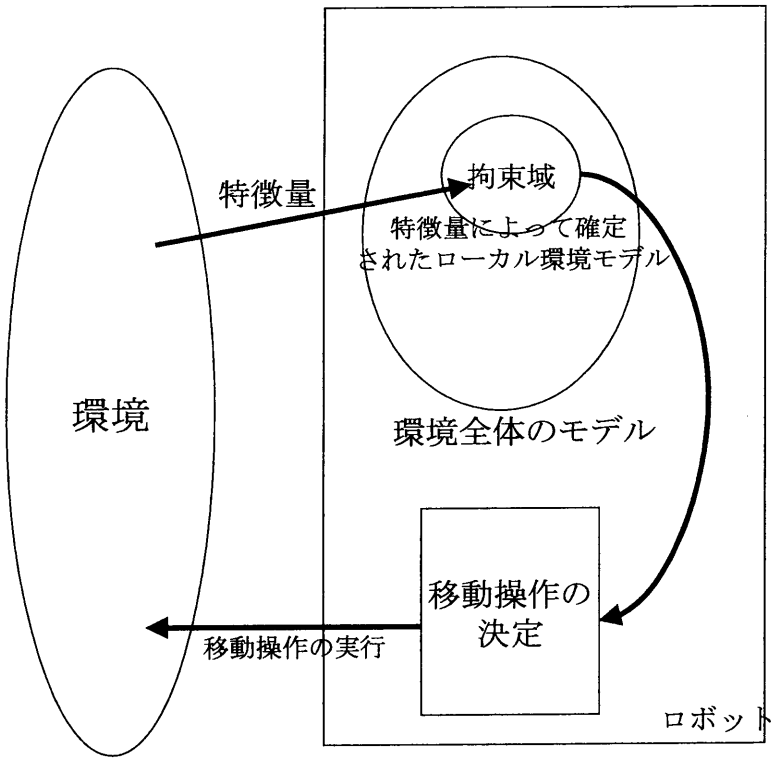


図 3.5: 拘束域と移動操作作業の実行手順

3.2.4 ロボットの操作

ロボットが作業を行なう際にはアクチュエータを介して $\mathbf{q}$ を制御する。その方式は移動ロボットやマニピュレータによってさまざまである。ロボットの動作方式で代表的なものには位置指令方式と速度指令方式がある。

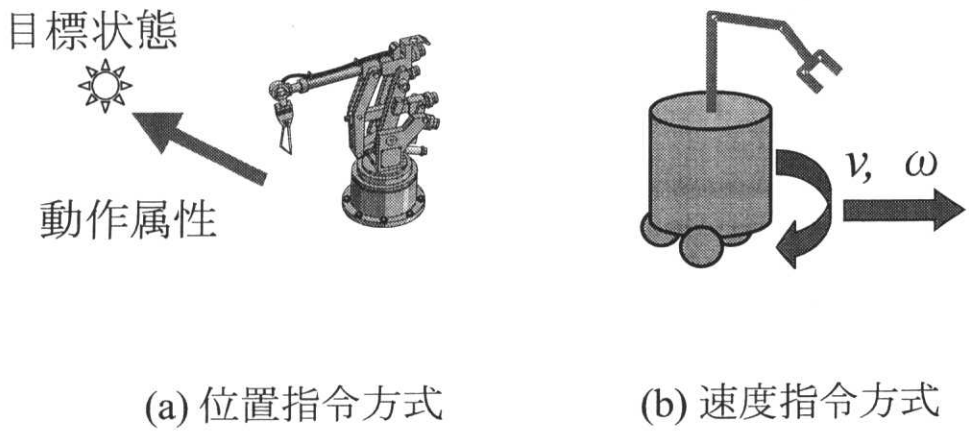


図 3.6: 位置指令方式と速度指令方式

位置指令方式は位置決め精度の良い産業用マニピュレータでよく用いられる方式である (Fig. 3.6 (a))。目標状態と動作属性（手先直線移動、関節等速回転移動など）を与えることでマニピュレータを動作させる。指令はは次の目標状態 $\mathbf{q}(s_{i+1})$ を直接与えることで行なわれる。また、動作属性を与えると $Sweep(\mathbf{q}(s)|s = [s_i, s_{i+1}])$ が確定する。

速度指令方式は外界センサを多用する移動ロボットでしばしば用いられる方式である (Fig. 3.6 (b))。具体的には動作を切替える時点でモータの回転速度を指令として出力し、モータは次の指令を受けるまでその回転速度を継続する。これらは $\mathbf{q}$ の増分の形で与えられるものである。

3.3 コンフィギュレーション・スペースによるモデルの記述

本節ではロボットや環境物体の占有領域を表現するのに、ロボットの状態ベクトル $\mathbf{q}$ の張る、状態空間を用いることを検討する。ロボットの状態空間を用いると、$Shape(\mathbf{q})$ は $\mathbf{q}$ そのものとしてシンプルに表現できる。また、掃引 $Sweep(\mathbf{q})$ も状態空間中の曲線分として表せるので、移動操作作業に適切な動作を求めるのに適している。

ロボットの状態空間の生成にはコンフィギュレーションスペース (configuration space; C-space) の理論が利用できる [38]。C-space の理論を用いると、作業空間における環境物体の形状から状態空間における環境物体の占有領域 $Shape(\theta_i)$ を算出することができる。

また、移動操作作業時にロボットが環境物体から受ける影響もシンプルな形で記述できる。ロボットが移動操作作業を行なう場合に、環境物体から受ける影響は作業空間で考える場合にはロボット・操作物体と環境物体の形状、すなわち $Shape(\mathbf{q})$ と $Shape_{ij}(\theta_{ij})$ の双方を考えなければならない。一方、C-space では、$Shape(\mathbf{q})$ が点 $\mathbf{q}$ に置き換えられるので、$Shape_{ij}(\theta_{ij})$ のみに着目することで移動操作作業に対する影響を論じることができる。

例えば、点 $\mathbf{q} = (x, y)$ を $Shape_{ij}(\theta_{ij})$ に属する面 $S : ax + by + c = 0$ と衝突しないように操作することを考える。S に対して $Shape_{ij}(\theta_{ij})$ の内側を

$$ax + by + c < 0$$

で表した場合、ロボットは操作物体が

$$ax + by + c > 0$$

を保つように移動操作を行なうだろう (Fig. 3.7)。このような、操作に与える影響を表した式を、操作に対する拘束式と呼ぶ。また、この式の左辺を拘束関数と呼ぶことにする。拘束式が有効なのは拘束域 $CArea(\theta_{ij}, \mathbf{q})$ の内部に限られる。

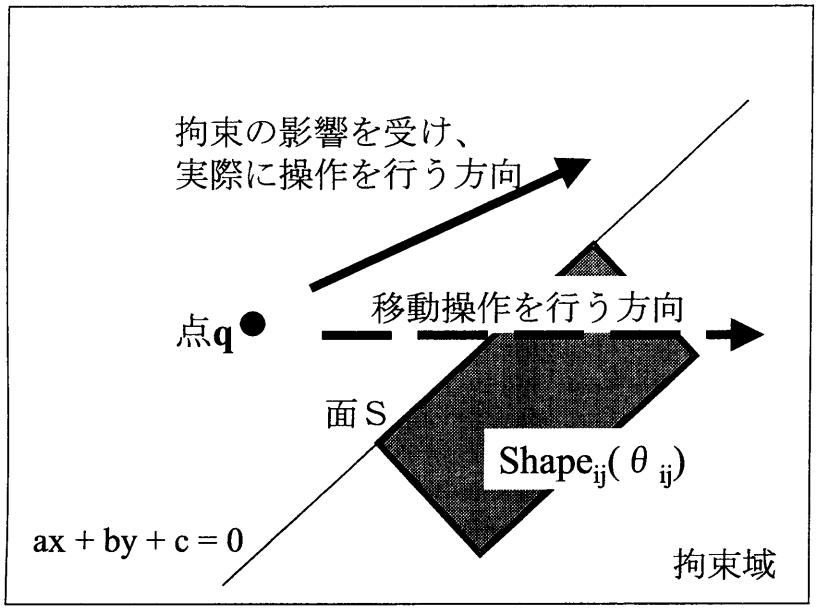


図 3.7: 移動操作とそれに影響を与える拘束

C-space では、環境物体（障害物）の占有領域は C-obstacle と呼ばれる。また、C-obstacle の境界面は C-surface と呼ばれる。拘束関数の値が 0 と等しくなるような面は C-surface の構成要素となる。

ロボット・操作物体と環境物体が作業空間においてともに多面体の場合、作業空間ではロボットや操作物体の頂点、稜、面が環境物体の頂点、稜、面から拘束を受ける。以下では C-space でこれらを表す拘束式の具体的な形を論じる。

3.3.1 多面体の拘束式

2 次元平面の作業空間上で作業を行なう場合は物体間の拘束は頂点、辺の 2 要素の間で生じ、次の 2 種類がある (Fig. 3.9)。

- (ロボット・操作物体の) 辺 - (環境物体の) 頂点
- 頂点 - 辺

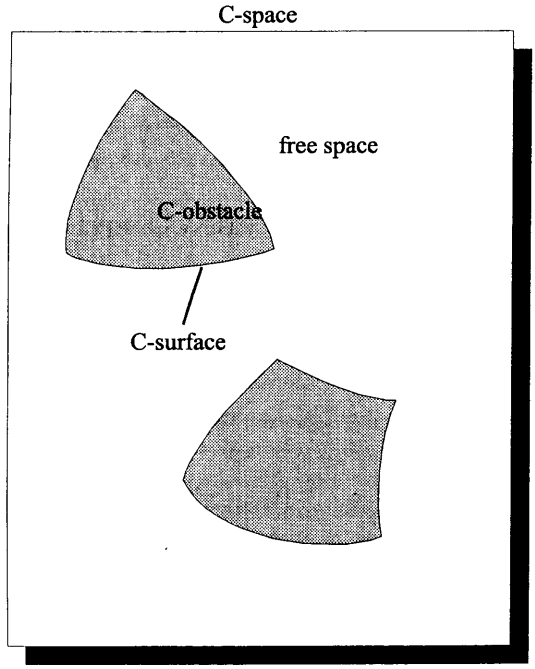


図 3.8: C-space の例

それぞれを type A, type B の拘束と呼ぶ。この他にも拘束は頂点 - 頂点や辺 - 辺の組み合わせが考えられる。しかし、これらは type A, type B の拘束に置き換えることが可能なので、通常は type A, type B の２種類のみを考える。

一方、３次元の作業空間上で作業を行なう場合は、拘束は次の３種類を考えれば十分である [38](Fig. 3.10)。

- 面 - 頂点
- 頂点 - 面
- 稜 - 稜

それぞれを type A, type B, type C の拘束と呼ぶ。

ロボット・操作物体を構成する多面体（移動ロボットの本体、マニピュレータのアームを構成するリンク、操作物体など）の作業空間における位置・姿勢 $\mathbf{r}$ は２次元、３次元の場合にそれぞれ一般に

$$\mathbf{r} = (x, y, \theta)$$
$$\mathbf{r} = (x, y, z, \theta, \phi, \psi)$$

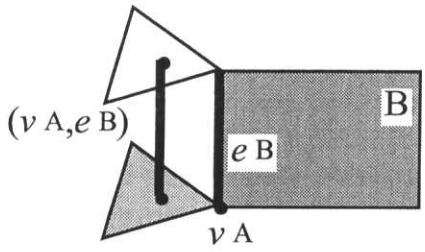
と表現できる。(θ は環境物体の特徴ベクトルの表記に用いた θ_i とは無関係) $\mathbf{r}$ は通常、ロボットの状態ベクトル $\mathbf{q}$ から容易に算出できる。その多面体の辺、頂点、または面、稜、頂点は $\mathbf{r}$ に対して回転行列 R と並進ベクトル T を適用することで求められる。

回転行列 R と並進ベクトル T は具体的には以下の形をしている。

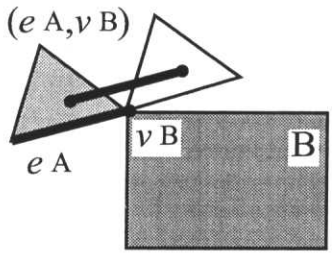
２次元の場合、回転行列は

$$R = \begin{bmatrix} p_0 & -p_1 \\ p_1 & p_0 \end{bmatrix} \tag{3.1}$$

で表される。ここで、 $p_0 = cos\theta$, $p_1 = sin\theta$ である。



a. Edge (v A,e B) of O A(B).



b. Edge (e A,v B) of O A(B).

図 3.9: 2 次元における C-surface

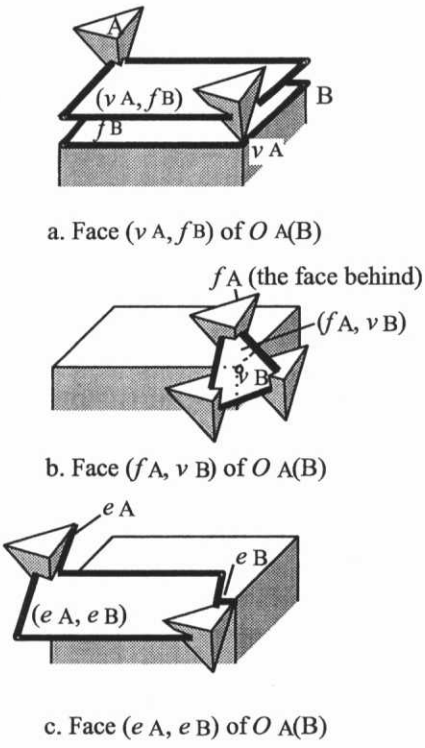


図 3.10: 3 次元における C-surface

3次元の場合、姿勢をオイラー角で表現した場合には

$$R = \begin{bmatrix} cos\theta cos\phi cos\psi - sin\phi sin\psi & -cos\theta cos\phi sin\psi - sin\phi cos\psi & sin\theta cos\phi \\ cos\theta sin\phi cos\psi + cos\phi sin\psi & -cos\theta sin\phi sin\psi + cos\phi cos\psi & sin\theta sin\phi \\ -sin\theta cos\psi & sin\theta \sin\psi & cos\theta \end{bmatrix} \tag{3.2}$$

となる [53]。一般には回転行列は $p_0^2 + p_1^2 + p_2^2 + p_3^2 = 1$ なる p_0, p_1, p_2, p_3 を用いて

$$R = \begin{bmatrix} p_0^2 + p_1^2 - p_2^2 - p_3^2 & 2(p_1p_2 - p_0p_3) & 2(p_1p_3 + p_0p_2) \\ 2(p_2p_1 + p_0p_3) & p_0^2 - p_1^2 + p_2^2 - p_3^2 & 2(p_2p_3 - p_0p_1) \\ 2(p_3p_1 - p_0p_2) & 2(p_3p_2 + p_0p_1) & p_0^2 - p_1^2 - p_2^2 + p_3^2 \end{bmatrix} \tag{3.3}$$

と書けることが知られている [29]。したがって、姿勢の表現がオイラー角以外の場合も拘束関数に関する以下の表現は共通である。

並進ベクトル T は2次元、3次元でそれぞれ次のようになる。

$$T = \begin{bmatrix} x \\ y \end{bmatrix}, \quad T = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

これらを用いて、それぞれのタイプの拘束関数は以下のように記述される。

[2次元の場合]

type A 拘束関数　ロボット・操作物体の辺 $\mathbf{a}_0 \cdot \mathbf{x} + b_0 = 0$ が環境物体の頂点 $\mathbf{p}$ から拘束を受ける場合は、次の拘束関数が得られる。

$$R\mathbf{a}_0 \cdot \mathbf{p} + (b_0 - R\mathbf{a}_0 \cdot T) \tag{3.4}$$

ここで $\mathbf{a} \cdot \mathbf{b}$ はベクトル a と b の内積を表す。

type B 拘束関数　ロボット・操作物体の頂点 $\mathbf{p}_0$ が環境物体の辺 $\mathbf{a}\cdot\mathbf{x}+b=0$ から拘束を受ける場合は、次の拘束関数が得られる。

$$\mathbf{a}\cdot(R\mathbf{p}_0+T)+b \tag{3.5}$$

[3次元の場合]

type A 拘束関数　ロボット・操作物体の面 $\mathbf{a}_0\cdot\mathbf{x}+b_0=0$ が環境物体の頂点 $\mathbf{p}$ から拘束を受ける場合は、次の拘束関数が得られる。

$$R\mathbf{a}_0\cdot\mathbf{p}+(b_0-R\mathbf{a}_0\cdot T) \tag{3.6}$$

type B 拘束関数　ロボット・操作物体の頂点 $\mathbf{p}_0$ が環境物体の面 $\mathbf{a}\cdot\mathbf{x}+b=0$ から拘束を受ける場合は、次の拘束関数が得られる。

$$\mathbf{a}\cdot(R\mathbf{p}_0+T)+b \tag{3.7}$$

type C 拘束関数　ロボット・操作物体の稜 $\mathbf{p}_0+t\mathbf{d}_0$ が環境物体の稜 $\mathbf{p}+u\mathbf{d}$ から拘束を受ける場合は、次の拘束関数が得られる。

$$R(\mathbf{p}_0\times\mathbf{d}_0)\cdot\mathbf{d}+(T-\mathbf{p})\cdot(R\mathbf{d}_0)\times\mathbf{d}$$

いずれのタイプでも回転自由度を固定とみなした場合、つまり R を定数とした場合は拘束関数は線形となる。一方、回転自由度も考慮した場合は、R も変数となるので、拘束関数は非線形の複雑な関数となることがわかる。

2次元、3次元いずれの場合も、幾何的条件によって実際には存在せず、考慮しなくてよい拘束がある。このような拘束は Donald の applicability constraint function を用いることで判別できる [16]。

3.4 移動操作ルールの記述

3.4.1 拘束関数と移動操作ルール

前節では多面体の形状で表わされた環境物体の間で操作物体を移動させる場合の拘束関数の具体的な形を述べた。本節では、これを移動操作に利用する方法を論じる。

拘束関数は多角形の位置・姿勢 $\mathbf{r}$ の関数であるが、$\mathbf{r}$ は $\mathbf{q}$ から求めることができるので、一般に $f(\mathbf{q})$ と記述することが可能である。このとき、拘束は $f(\mathbf{q}) > 0$ または $f(\mathbf{q}) < 0$ の形で表わせる。これらの拘束をそれぞれ $+$ および $-$ の符号で表記する (Fig. 3.11)。

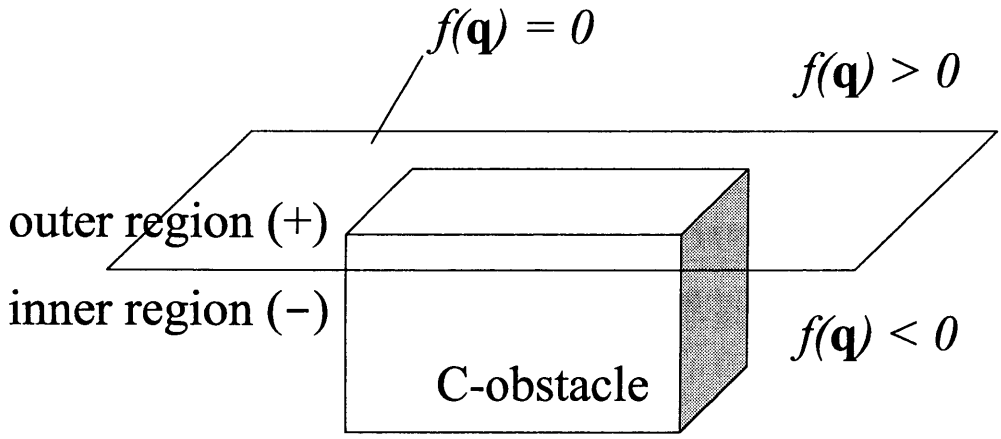


図 3.11: 拘束と符号

移動操作は拘束のうちのいくつかに依存して決定される。拘束からの影響が存在する場合に、影響の強弱に変化がないとすると、これは

拘束 $1 \sim n$ を満たす → 移動操作 M を行なう

のようなルールを用いて記述できる。

具体的には、1からnの拘束を満たすという条件部は

$$\begin{cases} s_1 f_1(\mathbf{q}) & > & 0 \\ s_2 f_2(\mathbf{q}) & > & 0 \\ \quad \vdots & & \\ s_n f_n(\mathbf{q}) & > & 0 \end{cases} \tag{3.8}$$

なる連立不等式で表される。ここで、s_iは符号であり、$s_i f_i$は

$$s_i f_i = \begin{cases} f_i & (when\ s_i\ is\ a\ +\ sign) \\ -f_i & (otherwise) \end{cases}$$

を表す。この連立不等式は状態空間において、ひとつの領域を形成する。

作業の目標状態の具体的な形も上述の連立不等式で表すことが可能である。この場合、ルールの記述は以下のようになる。

拘束1～nを満たす → 作業を完了したと判断し停止する

移動操作は拘束域のすべての拘束から影響を受けるわけではないので、ある状況においてどの拘束が移動操作に影響を与えるのかを知り、適切な条件部を作ることが、ルールを作る上で取り組むべき課題の一つである。移動操作に影響を与える拘束を有効な拘束、与えない拘束を無効な拘束と呼ぶ。

3.4.2 ロボット操作方式と移動操作ルールの実行部

第3.2.4節ではロボットの操作方式として位置指令方式と速度指令方式があると述べた。位置指令方式は位置決め精度のよい産業用マニピュレータでよく用いられるが、目標位置を与える必要がある。速度指令方式は移動ロボットで用いられ、一般に位置決め精度はよくないが、目標位置を与える代わりにモータの回転速度等を与える。このように性質の異なる方式に対してはルールの実行部の記述が異なってくる。

速度指令方式の場合には、操作性を重視して、モータに与える回転速度パターンをいくつかに限定することがしばしば行なわれる。移動ロボットであれば、例えば直

進、後退、右折、左折などの行動を実現するようなパターンを用意し移動操作に用いる。

このような場合には、ルールの実行部は直進、後退、右折、左折などの行動パターンから1つをあてはめることで記述できる。このようにルールを記述する方式を行動選択方式と呼ぶ (Fig. 3.12)。

一方、位置指令方式の場合には、目標位置を与える必要がある。未整備環境を前提とする場合には、目標位置は周辺の環境物体の形状 $Shape_{ij}(\theta_{ij})$ に依存して変化する。例えば、実行部は式 (3.8) と同じ形の

$$\begin{cases} s_1 f_1(\mathbf{q}) & > & 0 \\ s_2 f_2(\mathbf{q}) & > & 0 \\ \quad \vdots & & \\ s_n f_n(\mathbf{q}) & > & 0 \end{cases} \tag{3.9}$$

という連立不等式を満たすような目標位置を出力する形式となる。C-space において、この式の表す領域を一種の状態とみなすと、位置指令方式の場合は条件部も実行部も状態によって記述されるので、状態を次々と移り変わりながら作業を行なうように見える。この観点から、このようにルールを記述する方式を状態遷移方式と呼ぶ (Fig. 3.12)。

3.5 実例教示によるルールの作成

第3.4.1節で指摘したように、ここでの主要な課題の一つは移動操作作業のルールの生成方法を考案することである。第2.3節では、そのために人間の実例教示を通じて、人間の持っている作業に関する知識を推測し、ルールの形でロボットに移植するというアイデアを提案した。

実例教示において人間が教示するのは具体的な環境における具体的な動作である。システムはその動作を観察して、各時点でロボットの満たす拘束や、人間の選択した行動パターンを取得することによって移動操作作業のルールを作成していく。

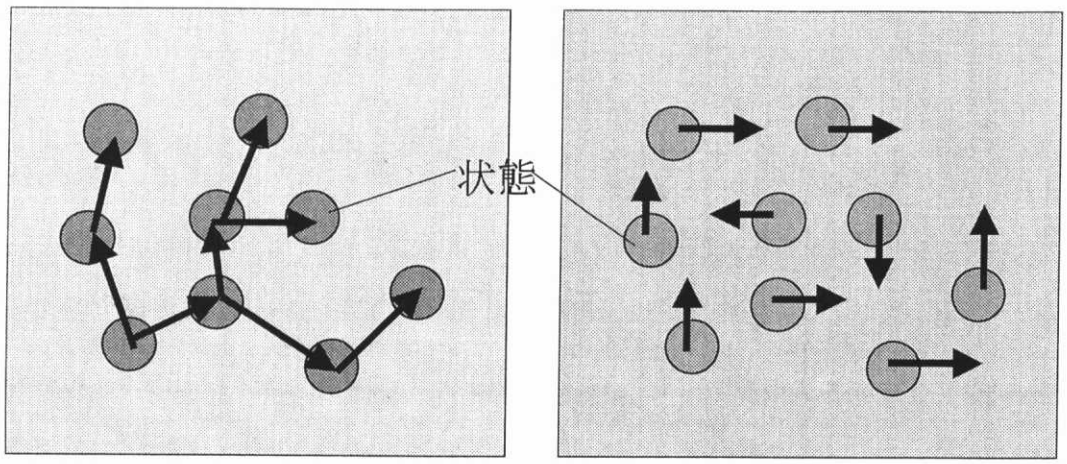


図 3.12: 状態遷移方式と行動選択方式のイメージ

これに類する研究に、実演教示の研究がある。

例えば、國吉らは pick and place 作業に関する実演教示法を検討した [36][37]。 Pick and place 作業はあらかじめ用意された基本操作の組み合わせで記述されると考える。また、それらの基本操作は開始時点と終了時点の状態の差を調べることで区別できるとした。システムは、人間の行なう実演から、操作物体の遷移している状態を調べ、その遷移から人間の行なっている基本操作を推定して、シーケンシャルなコマンド列に変換する。

池内らは接触を伴う組立作業などの実演教示法を検討した [23][30]。接触を伴う組立作業では、操作物体と環境物体の間で接触している面の組み合わせが同一の場合、同じ基本動作を行なうことで同等の作業が実行可能と考え、一つの状態にまとめて表現する。多面体同士の面接触の組み合わせは幾何学的知識を利用して、システマティックに列挙できる。それらから状態遷移グラフを作成し、教示者がどの手順で組立を行なったのか、そのグラフに当てはめることで推定し、シーケンシャルな接触遷移手順を出力する。

本研究では式 (3.8) の表す領域が一種の状態に相当するが、その状態は教示時には

未知であり、教示を通じて推定する必要がある点で、國吉、池内らの研究と異なる。

これに対処するために、本研究では以下のような手順でルールの生成を行なう。

1. 複数の事例について作業を教示する

2. 各時点において影響している可能性のあるすべての拘束と、そのときに行なった操作を組にしてデータベースに保存する。

3. 共通の操作を行なったデータを比較することで、作業に影響を及ぼしていない拘束と及ぼしている拘束を推定し、条件部を作成する。実行部はそれらと組になっている操作となる。

3.6 ルールセットと再利用性

ルールの記述はロボットの属している拘束域にのみ依存しているので、それらの拘束域以外の条件の異なる作業環境においても、それらのルールは再利用可能である。

また、拘束域に関連付けられている $Shape_{ij}(\theta_{ij})$ にローカルな環境を対応付けることができれば、全く異なる場所でもそれらのルールを再利用できる。

このように、再利用を目的としてルールを依存した拘束域毎にまとめておくことが考えられる。これをルールセットと呼ぶ。

例えば、移動ロボットが廊下走行する場合に、廊下が同じような長方形であり、そこに放置されている障害物が同じ形とみなせるならば、その作業について改めてルール作成を行なわずに、以前教示した作業からその部分に相当するルールセットを導入することが考えられる。このようなルールセットの再利用性を活用することで、事例教示によってルールを作成する手間をより節減できる。

3.7 基本的なシステム構成

以上のアプローチを実現するために、本論文ではFig. 3.13のように、以下のモジュールでシステムを構成する。

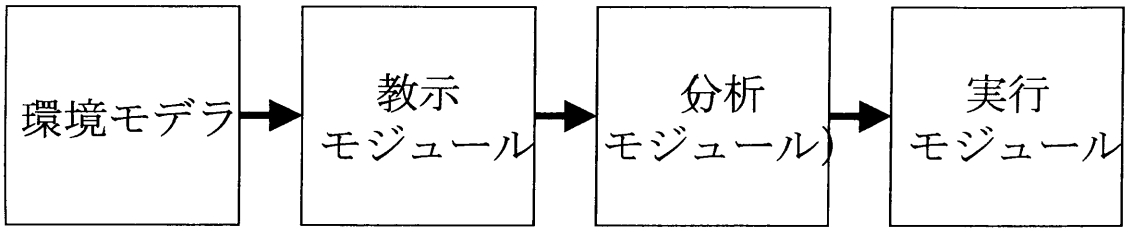


図 3.13: 提案手法を実現するシステムの構成

- 環境モデラ

  環境モデラでは、ロボットが作業を行なう環境のモデリングを行ない以下の３つの機能を備える。

  - 物体の基本的な形状情報の定義。
  - 物体に関連した特長量の設定と、それに依存する形状部分の指定。
  - 各形状部分に対応した拘束域の定義。

- 教示モジュール

  教示モジュールは以下の３つの機能を備える。

  - オペレータが事例教示を行なうためのインタフェース機能。
  - オペレータの教示から得られたデータを、環境モデラで作成した環境情報と関連付ける機能。
  - 関連付けを行なったデータをデータベースに蓄積する機能。

- 分析モジュール

  分析モジュールは教示モジュールでデータベースに蓄積したデータを分析して、移動操作作業を行なうためのルールを作成する。後に述べるように、分析に用いる手法によってシステムは分析モジュールを有する場合と有しない場合とがある。分析モジュールを有しない場合にはロボットは作業実行時に教示モジュールで作成したデータベースを直接用いることで操作方法を決定する。

- 実行モジュール

  実行モジュールは分析モジュールで作成されたルール、または教示モジュールで作成したデータベースに基づいて作業を実行する。

3.8 おわりに

本章では、本論文が対象とする移動操作作業のモデルについて説明し、問題の定式化を行なった。

第3.2節では、移動操作作業を初期状態から目標状態まで、操作対象となる物体をその他の物体と衝突させることなく移動させる作業ととらえ、操作物体、環境物体、およびロボットの関係で記述されることを明らかにした。また、環境の不確定性に対処するために、環境モデルをセンサで特徴量を検出することで部分的に確定しながら作業を進める考えを述べ、拘束域の概念を導入した。さらに、ロボットの動作方式として位置指令方式と速度指令方式が存在することを指摘した。

第3.3節では、第3.2節で説明したモデルに基づき、ロボットの状態空間で作業をシンプルに記述する方法を論じた。状態空間の具体的な形はコンフィギュレーションスペースの理論を用いて知ることができる。また、移動操作時にロボットが環境物体から受ける影響について検討し、その拘束式を具体的な形で示した。

第3.4節では、ロボットが受ける拘束を考慮しながら作業する場合は、ルールによって実行すべき操作を決定できることを述べた。ルールの条件部は拘束の集合で表現さ

れた状態によって記述されることを説明した。ルールの実行部は、速度指令方式の場合はそのときに実行すべき行動パターンで記述される。また、位置指令方式の場合は操作によって次に満たすべき状態によって記述される。

第3.5節では、第3.4節で述べたルールを実例教示から作成する方法について述べた。

第3.6節では作成したルールを有効に再利用するための、ルールセットの考え方について論じた。

第3.7節では事例教示を用いてルールを作成し、作業を実行するシステムを構築する場合の基本的なシステム構成について説明した。

4　行動選択方式における事例教示

4.1　はじめに

本章では行動選択方式で操作を行なうロボットを想定した場合の、移動操作作業の実行ルールを事例教示から作成する方法を検討する。本章の概要は以下の通りである。

第4.2節では、センサで得られた情報と事前にロボットが持っている環境に関する知識からルールを作成する方法の概要を説明する。

第4.3節では、提案手法の有効性を評価する方法について検討する。

第4.4節では、事例教示を行うための教示モジュールの構成を論じる。また、第4.5節では事例教示から移動操作作業を実行するためのルールを作成し、それを用いて作業を行う分析モジュールおよび実行モジュールの構成を述べる。

第4.6節では事例教示により作成されたルールを補うためのデフォルト・ルールの概念について説明する。

第4.7節では、第4.3節の評価方法に基づいて、提案方式の評価を行う。

第4.8節ではルールを作成する手法として有望な方法を比較し、その中から本論文の目的に最も適した手法を検討する。次の第4.9節ではその手法によってルールを作成するためのアルゴリズムの詳細を述べる。

第4.10節ではそのアルゴリズムを計算機に実装し、実際に移動ロボットを用いて移動操作作業の事例教示からのルール作成を行う実験を行い、提案手法の有効性を検証する。

4.2 ルール作成の概要

行動選択方式の場合、ルールの実行部の出力はあらかじめ用意された行動のパターンのうちの一つとなる。ルールの条件部を記述する拘束の候補はセンサによって得られた特徴量から得られる。この前提で、人間の示した作業の実例からルールを作成する方法の概要を述べる。

ルールを作成するためには、候補の中から有効な拘束を選別し、適切な行動と結びつける手段を考える必要がある。

これを考えるために、まずすべての拘束を有効とみなし、状態空間においてそれを満たす部分空間を考える。これは拘束で表わすことの可能な最小の部分空間である。また、この部分空間は求めようとしている状態の部分空間でもある。そこで、これを状態プリミティブと呼ぶ (Fig. 4.1)。

状態プリミティブも一種の状態なので、これを条件部としてもルールを作ることができる。しかし、そのルールの適用可能範囲は非常に限定されるうえ、作業ごとの環境の相違が大きい場合に再利用性も低い。したがって、十分な性能のルールを得るためには教示量を増やさねばならず、オペレータにかかる負担は大きくなり、実用性に劣る。そこで、実例教示により一旦データを収集し、それを分析して無効な拘束を排除することで、より少ない教示量によって効果的にルールを作成することを図る。ここでは、無効な拘束を見出すために、同じ行動が選択されている状態プリミティブ

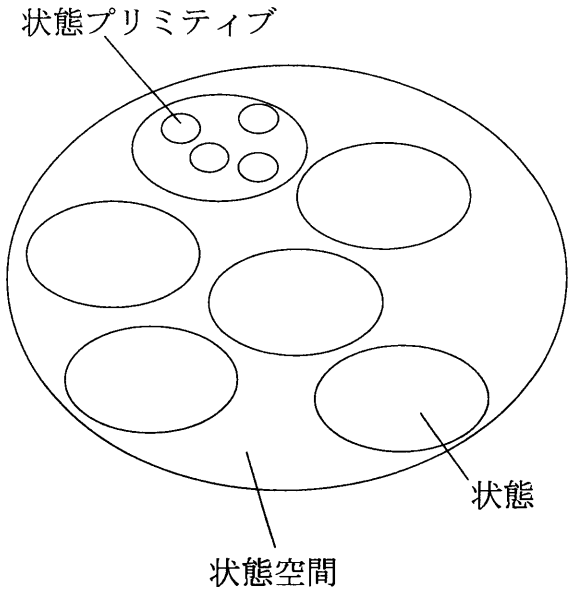


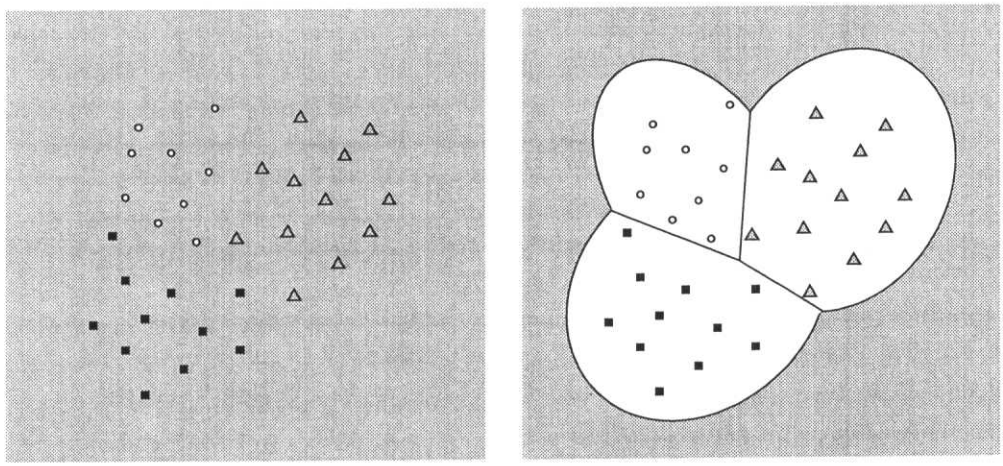
図 4.1: 状態空間と状態プリミティブ

の分布状況に着目する。つまり、それらが状態空間においてクラスタをなしているようであったら、その近傍はひとつの状態を構成していると推定し、無効な拘束を見出せると考える。

この問題は、パターン認識問題の技法を利用して解くことができる。

パターン認識とは、得られたパターンをあらかじめ定められた複数のクラスのうちの一つに対応させる処理である [24]。通常、パターンはベクトルで表わされ、パターンベクトルの張る空間はパターン空間と呼ばれる。パターン空間はクラスに応じて部分空間 $R_1, \cdots, R_n$ に分割できると考える。ここで、n はクラスの数である。観測されたパターン p が部分空間 R_i に属するなら、p は R_i に対応付けられたクラスに属すると決定される。パターン認識問題とは、サンプルの分布状態からこれらの部分空間を決定する問題であり (Fig. 4.2)、これを解くための方法がいくつか存在する。

これを本問題に適用する場合、状態プリミティブを構成する拘束をパターンとみ

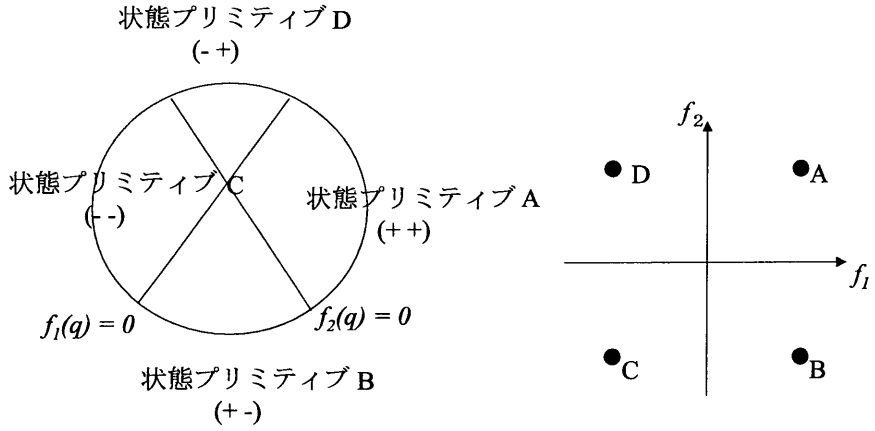


(a) サンプルの分布状態の例

(b) 分布状態から
部分空間を決定した例

図 4.2: パターン空間のクラス分け

なす。そこで、パターン空間に対応する拘束空間を考える。拘束空間では、式 (3.8) で表れる拘束の符号によって状態プリミティブの位置を表現する (Fig. 4.3)。例えば、n 番目の拘束の符号が $+$ または $-$ の場合、拘束空間における座標の n 番目の成分は 1 または -1 になる。ここで、例えば n 番目の拘束が 1 の状態プリミティブと -1 のそれが同じクラスに属するという結果が出たら、n 番目の拘束は無効な拘束であると判別できる。状態プリミティブは現在モデリングされているローカル環境に存在するすべての拘束によって記述されているので、いずれの拘束に対しても必ず符号が存在する。



(a) 状態空間の例　　(b) aに対応した拘束空間

図 4.3: 状態空間と拘束空間の例

以下では、パターン認識技法の適用を検討するとともに、それにより有効なルールが作成可能かどうか検証する。

4.3 有効性の評価方法

移動操作に関するルールは作成するのに用いた技法やそれにかける手間によって所定の性能を満たさなかったり、優劣が生じることが考えられる。そこで、何らかの基準を設けてその有効性を評価する必要がある。

ここでは、以下の方法で有効性を検証することを考える。

方法が満たすべき最低限の条件として以下を考える。

- かけた手間に応じて作業達成率の向上が見られるか。
- 作業達成率が適正な水準まで向上するか。

  作業達成率が低い水準で飽和するようであれば、有効な方法とはいえない。

また、他の有効な手法と比較するときは以下を検証することを考える。

- 同じ手間をかけた場合の作業達成率の差

4.4 教示モジュールの構成

本節では行動選択方式でロボットの事例教示を行なう場合の教示モジュールの構成を示す。

教示モジュールはセンサ部、モデリング部、教示インタフェース部、行動記録部、エフェクタ部、および環境データベース、マナーデータベースから構成される (Fig. 4.4)。

センサ部はカメラなどロボットの具備するセンサを用いてあらかじめ決められた環境の特徴量を観測し、その結果をベクトルとして出力する。

モデリング部はセンサ部から入力された特徴ベクトルから、現在ロボットが属している拘束域を判定し、特徴ベクトルに基づいて、環境データベースのデータからローカルな環境のモデリングを行なう。この手順は第 3.2.3節で述べたとおりである。環境

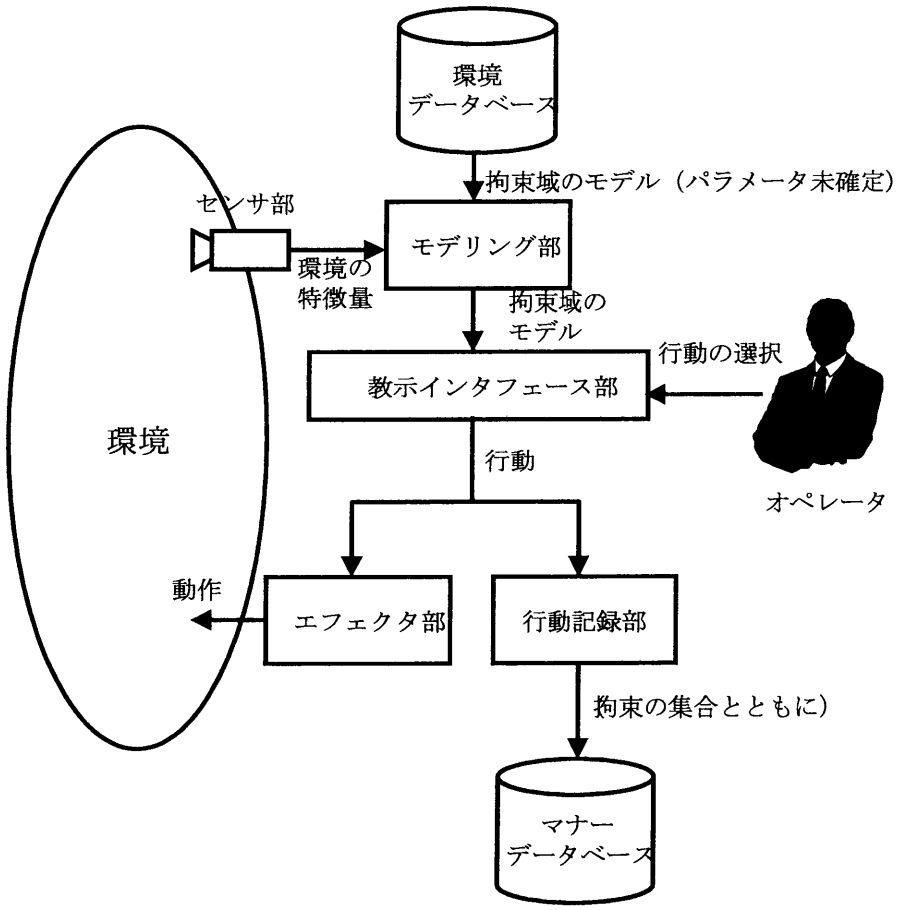


図 4.4: 教示モジュールの構成

データベースには特徴ベクトルから環境をモデリングするための情報が収められている。

教示インタフェース部では、行動リストの中から行動を選択する形式になっている。行動リストの要素あらかじめ用意された行動のパターンで構成されている。オペレータはその中からそのときに最も適切と思われる行動を１つ選択する。

オペレータの選択した行動は行動記録部とエフェクタ部に出力される。行動記録部はオペレータの選択した行動を受けとると、そのときに作業に影響を与えている可能性のある拘束の集合（つまり状態プリミティブを構成する拘束の集合）と関連付けて、マナーデータベースに記録していく。マナーとは拘束の集合とそのときに行なった行動のペアのことである。

エフェクタ部はオペレータの選択した行動に基づいてロボットを動作させる。

4.5 分析・実行モジュールの構成

マナーデータベースに蓄積されたデータからは、パターン認識の手法を用いてルールを作成する。パターン認識問題を解く手法としては最近傍法、決定木、ニューラル・ネットワークが代表的である。適用する手法によって、作業を実行しながらルールを作成選択する場合と、あらかじめルールを作成してから作業を実行する場合がある。

最近傍法を適用する場合は前者に相当する。このときは分析モジュールは存在せず、実行モジュールのみで処理が行なわれる。

このときの実行モジュールはセンサ部、モデリング部、行動決定部、エフェクタ部、マナーデータベースから構成される (Fig. 4.5)。

センサ部、モデリング部、およびマナーデータベースは教示モジュールのものと共通である。センサ部から特徴量ベクトルが出力され、モデリング部がローカルな環境のモデルを作成すると、最近傍法を用いた場合、行動決定部は拘束空間において最も類似のサンプルをマナーデータベースから検索し、行動を選択する。サンプルとの

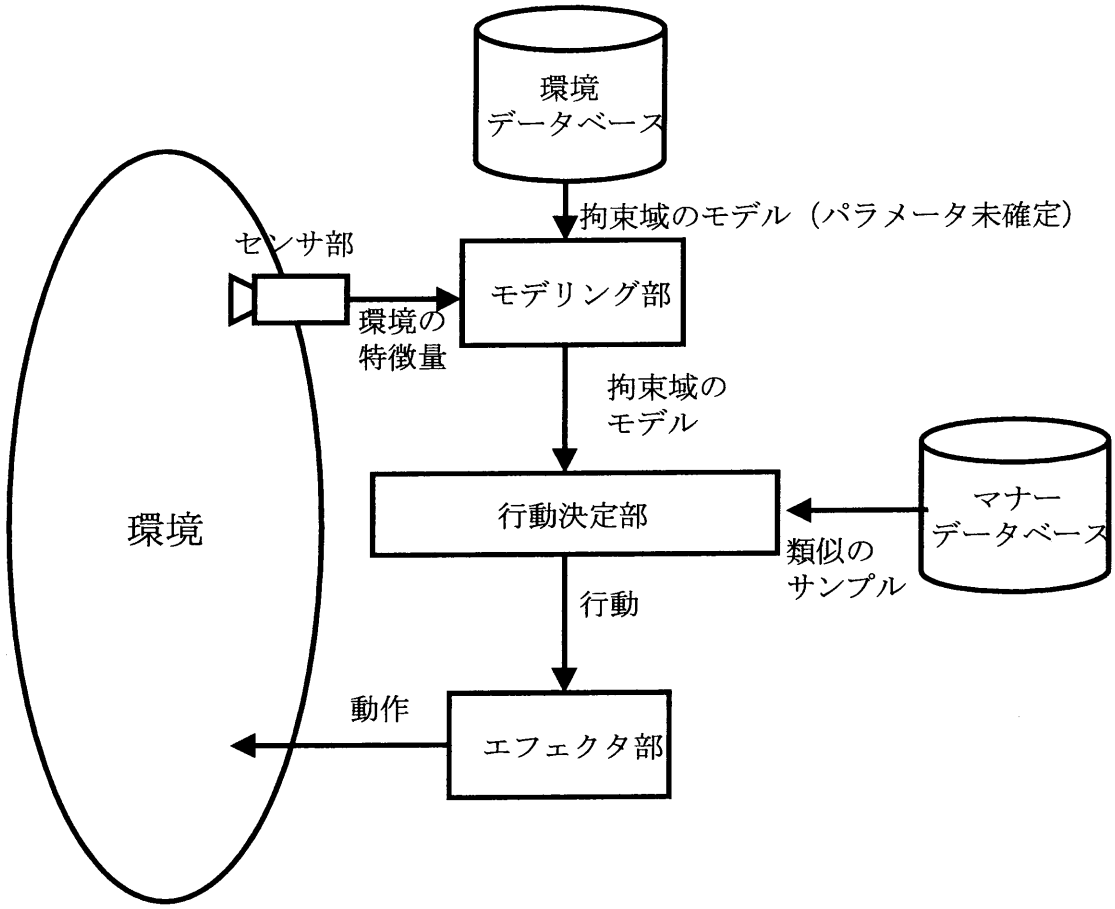


図 4.5: 実行モジュールの構成 (分析モジュールが存在しない場合)

類似性は拘束空間における距離により評価する。拘束空間における距離は例えば拘束の符号の相違数により定義できる。

エフェクタ部は類似性が最も高いと判定されたサンプルで選択した行動が適切な行動と考え、それに基づいてロボットを動作させる。

決定木、ニューラル・ネットワークを利用する場合は後者に相当する。このときは、分析モジュールがデータを分析し、ルールを出力する。出力されるルールの形式は、手法によって決定木や、ニューラル・ネットワークの結合係数の形をとることがあり、必ずしも if-then 形式のような明示的表現とは限らない。実行モジュールはそのルールに基づいて作業を実行する。

このときの分析モジュールはマナーデータベースのデータを入力とし、ルールデータベースに記録する (Fig. 4.6)。

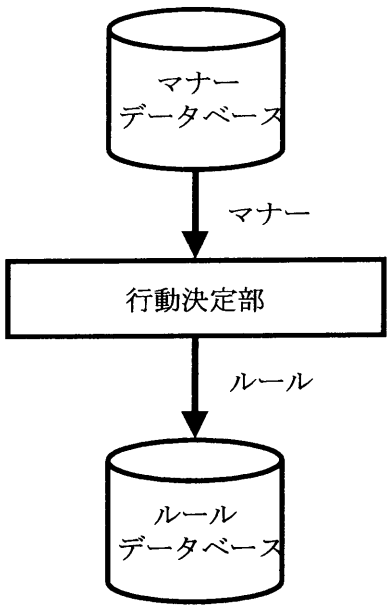


図 4.6: 分析モジュールの構成

一方、実行モジュールはセンサ部、モデリング部、行動決定部、エフェクタ部、ルー

ルデータベースから構成される (Fig. 4.7)。

センサ部の出力した特徴量ベクトルから、モデリング部がローカルな環境モデルを作成する。行動決定部はロボットの現在の状態ベクトルが満たす条件部を有するルールをルールデータベースから検索し、その実行部が出力する行動をエフェクタ部に実行させる。

最近傍法、決定木、ニューラル・ネットワークの手法の比較検討は第 4.8節で行なう。

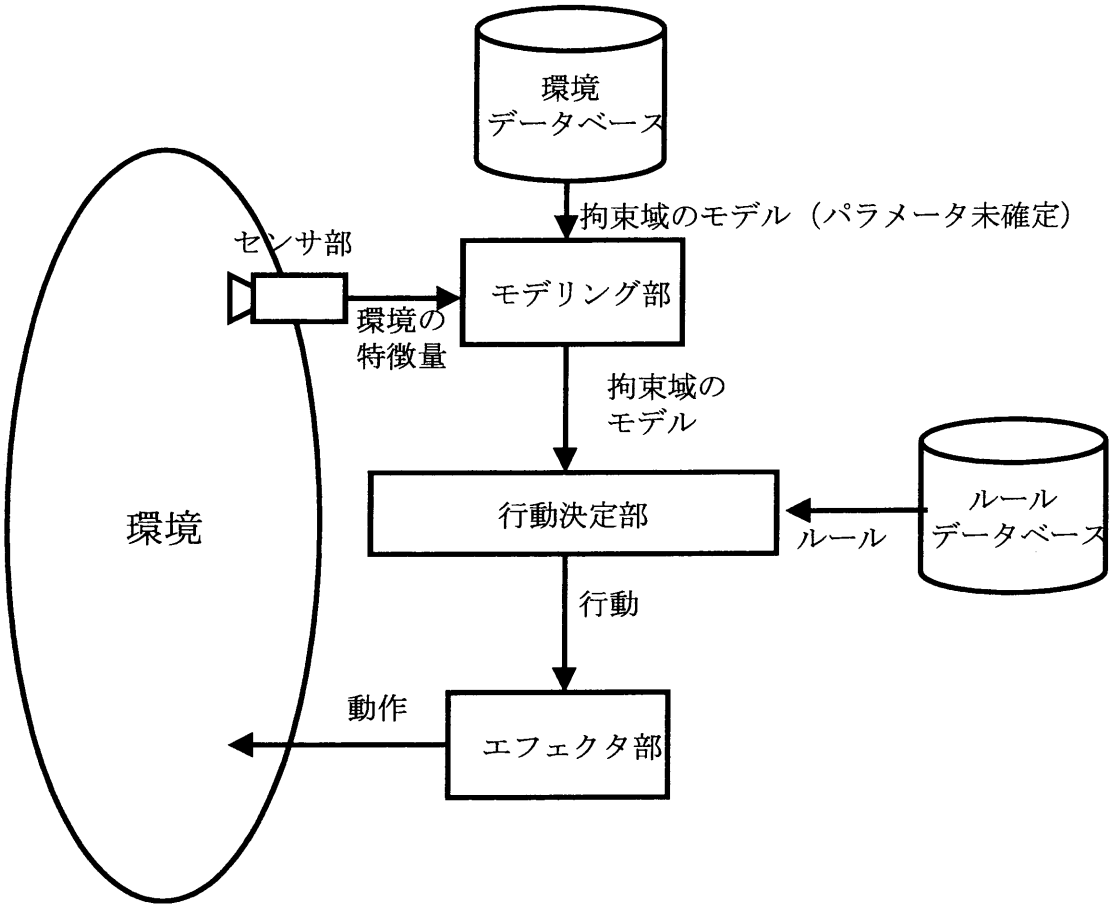


図 4.7: 実行モジュールの構成 (分析モジュールが存在する場合)

4.6 行動失敗時のリカバリ

事例教示が不十分であれば、条件を満たすルールが存在しなかったり、適切なルールが生成されず操作の実行に失敗したりする可能性がある。

これらが生じた場合のリカバリにここでは２段階の方法を用意することで対処する。一段階目では、事例教示で生成したものほど効率的でなくてもよいが、より確実に目標状態に到達できる行動を生成するルールを用意することを考える。このルールをデフォルト・ルールと呼ぶ。

デフォルト・ルールとして利用可能な方法はいくつか考えられる。例えば、ラメルスキー法、アニーリング法、ブラウニアン法がある。

ラメルスキー法は壁沿い法を改良した一般的なルールであり、未知環境で経路が存在すれば確実にゴール点まで到達する方法である [39]。

アニーリング法は直接ゴールに向かう行動に、ランダム要素を加えた方法である [7]。ランダムノイズを加えることで、理論的には局所的なトラップにはまるのを回避してゴールに到達可能である。

ブラウニアン法は、すべての方向に等確率で動く、ブラウン運動をする物体の位置は、初期位置を中心として、分散が時間とともに増大していく性質を利用したものである [57]。運動範囲に障害物が存在する場合は、正確には正規分布とはならない。しかし、ブラウン運動を継続すると、ロボットが近傍の既知のルールが適用できる範囲に到達する確率は高まる。

ラメルスキー法は他の２つに比べて効率的であるが、適用可能なのは状態空間が２次元の場合に限られる。アニーリング法とブラウニアン法は次元による制約はない。しかし、アニーリング法ではその時点でゴールの位置が既知の必要があるので、本手法には向かない。そこで、ここではデフォルト・ルールとしてブラウニアン法を用意する。

ただし、ブラウニアン法を用いるためにはロボットはすべての方向に等確率で向かう運動を近似的に実現する必要がある。

例えば、その場回転と前進の可能な移動ロボットの場合は、これを実現するために、デフォルト・ルールを適用している間は、以下のような手順で動作を行なうことが考えられる。

1. その場でランダムに方向転換を行なう。転換はすべての方向に等確率に行なわれるようにする。

2. 短く前進を行なう。前進を行なう時間は一定とする。

3. リカバリが完了するまで以上を繰り返す。

あるタイムリミットの間にリカバリが完了しなければ、システムはデフォルト・ルールの適用を打ち切る。二段階目ではオペレータに対してその事例に関する教示をリクエストする。オペレータへのリクエストは確実な方法であるが、オペレータの手間の増加も意味するので、最終的な手段となっている。

4.7 事例教示によるルール生成能力の検証

以上、事例教示によってルールを作成する方法を述べた。

本節ではシミュレーションによって事例教示からルールを作成する方式の有効性を確認する。ここでは第4.3節に従って以下の項目を検証する。

- 教示回数と作業達成率の関係
- 作業達成率の妥当性
- 他の手法とのルール作成効率性の比較
- 人間の選択的教示による効果

本節ではこれらの検証について順次説明する。

4.7.1 教示回数と作業達成率の関係

まず、事例教示によって作成されたルールを用いた場合の作業達成率について調査した。

[実験1]

100x100 個のセルからなる拘束域を用意し、ロボットを任意のセルからゴールとして定められたセルまで移動させる作業を考える。環境には大きさ 12x12 の環境物体が 40 個配置され (Fig. 4.8)、ロボットは環境物体の占拠しているセルには進入できないとする。ここで、事例教示の量と作業成功率の関係を調査する。

各セルは状態プリミティブに相当し、ロボットは最近傍法を用いて適当な行動を選択する。ある行動を選択した結果、環境物体の占拠したセルに進入した場合は、その行動は失敗したとみなし、デフォルト・ルールを採用し、ブラウニアン法によりもとのセルから新たな行動を選択して対処する。オペレータへのリクエストは行なわない。

ロボットの選択できる行動は上下左右 4 方向のセルへの移動と、作業終了、作業中断の 6 種類とする。ロボットが作業終了を選択した時点でロボットは作業を達成したとみなす。ロボットが作業中断を選択した場合はロボットが作業達成不可能と判断したとみなす。

ゴールのセルを環境の中心に設定する。事例教示時は、スタートのセルを環境物体の占有していないセルの中からランダムに選択する。

作業効率を最適経路のステップ数と比較することで評価可能にするために、事例教示では A* アルゴリズム [21] で求めた最適経路を与える。

マナーデータベースに蓄積されるマナーは状態プリミティブとそのときに選択した行動のペアで記述される。状態プリミティブはセルの座標 (x, y) で与えた。

求めるルールが単純で自明なものであることを避けるために環境には次の2種類のトラップが用意されている。

1. スタートによっては環境物体を迂回しなければゴールのセルに到達できない場所がある。

2. スタートによってはゴールに到達できない場所がある。

後者のケースでは作業中断を選択したら作業達成とみなす。

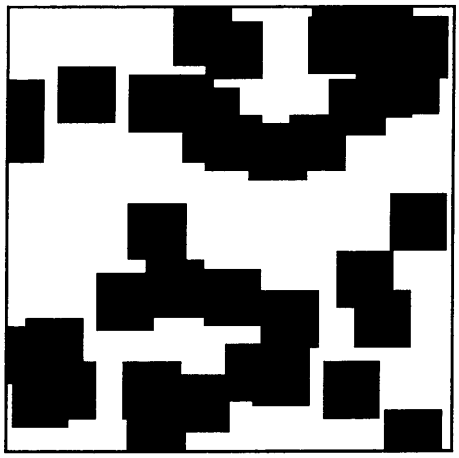


図 4.8: 本シミュレーションで用いた環境

ルールの選択は最近傍法により、現在のセルの位置と距離 $|x_1 - x_2| + |y_1 - y_2|$ が最短となるようなセルを持つものを適用する。10000 ステップ以内に作業終了か作業中断の判断が行われない場合にはタイムアウトする。作業はタイムアウトしたとき、ゴール以外で作業終了したとき、および作業可能 (ゴール到着可能) なのに作業中断したときに達成できなかったと判断する。

以上の設定でDOS/V互換の計算機上にプログラムをC言語で作成し、シミュレーションを行った。シミュレーションでは教示回数はスタートを与えてゴールのセルに到達するまでの期間を1回とカウントしている。

Fig. 4.9に第4.4節に示した教示モジュールの構成と、本シミュレーションの教示時のシステム構成との関係を示す。また、Fig. 4.10に第4.5節に示した実行モジュールの構成と、本シミュレーションの実行時のシステム構成との関係を示す。

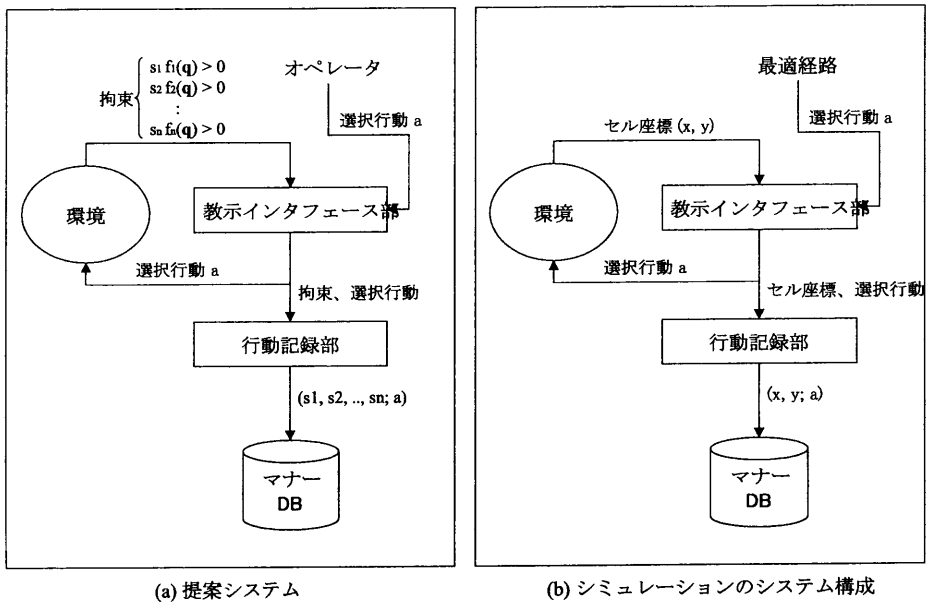


図 4.9: シミュレーションの教示モジュールの対応

この条件で実験を行ない、教示回数と作業達成率との関係を調べたものがFig. 4.11のグラフである。このグラフでは教示回数の増加につれて作業達成率は単調に増加し、教示回数70回（教示回数の軸は10回刻み）においてゴール到達率100 %に達している様子が見られる。

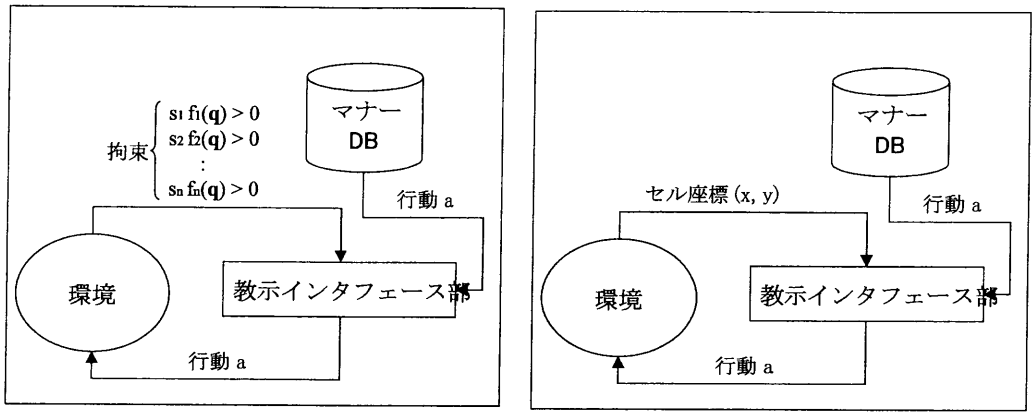


図 4.10: シミュレーションの実行モジュールの対応

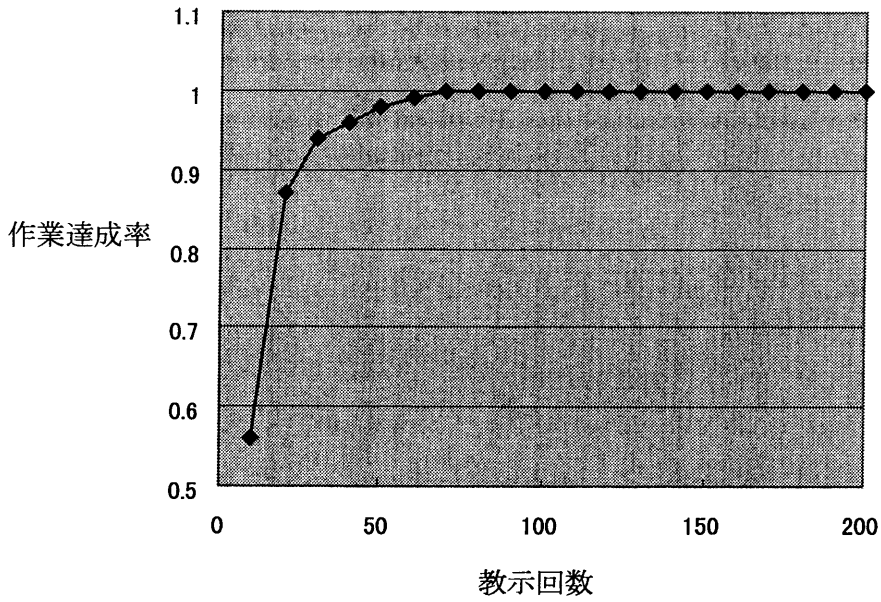


図 4.11: 教示回数と作業達成率の関係

4.7.2 作業効率の妥当性

実験１に対して、同時に作業効率の妥当性について調査した。Fig. 4.12はルールの生成した経路の長さが最適経路の長さと一致した率を示したグラフである。横軸には教示回数をとっている。

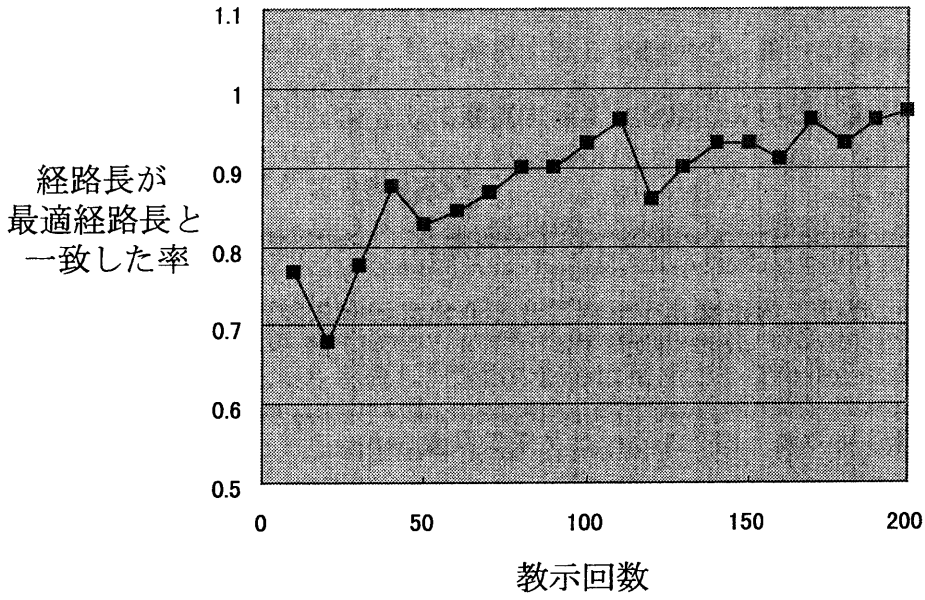


図 4.12: 教示回数と経路長の関係

このグラフでは一致率は単調には増加しないが、全体の傾向としては増加している様子が見られ、教示回数 70 回で一致率がコンスタントに９割以上となっており、適正な水準に至っているとみなせる。

このように、教示回数に応じて作業達成率は向上し、その効率の妥当性も見られたので、本論文の提案手法は少なくともルール作成に対する最低限の条件を満たしていることがわかる。

4.7.3 ルール生成の様子

次に、実験１の環境を 50x50 個のセルに縮小した環境において、同様の実験を行い、ルールの生成する様子を観察した。

Fig. 4.13は教示事例数５～１００の場合に、各セルにおいて選択される行動パターンの変化をプロットした図である。塗りつぶされている領域は、濃い方から順に左、右、下、上への移動を行動パターンとして選択することを示す。真っ白な領域は作業終了または作業中断を行動パターンとして選択することを示している。また、白い点は本実験で教示した事例のスタート点を示す。

Fig. 4.14は Fig. 4.13の行動パターンの選択によってそのセルをスタート点として、ゴールに到達できたセルとできなかったセルとをプロットしたものである。中間色の部分はゴールまで到達できたセルを表し、空白の部分は到達できなかったセルを表す。ただし、ここではデフォルト・ルールの適用効果がわかるように、ロボットが 1000 ステップ以内にゴールに到達できなかった場合はタイムアウトとした。（実験１の場合は 10000 ステップ）

Fig. 4.14を観察すると、教示事例数の増加につれてゴールに到達できたセルの数が順調に増大し、教示事例数 100 ではゴール到達可能なほぼすべてのセルからゴールに到達している様子が見られる。これは実験１で得られた、ゴール到達率が単調増加し、最終的に 100 %近くになるという結果を再確認するものである。

ところで、ゴールに到達できたセルの分布を見ると、 Fig. 4.14(b) における A の部分のように、ゴールに到達できたセルがまばらに分布している部分が存在する。 Fig. 4.13 (b) において対応する部分を見ると、左向きに進む行動パターンのみ選択されて、左の障害物に衝突してしまうようになっており、作業を完遂するための妥当な行動パターンが設定されていないことが判る。 A はロボットが壁にぶつかったときにデフォルト・ルールを適用してそれに対処した部分と推定される。 Fig. 4.13にプロットした

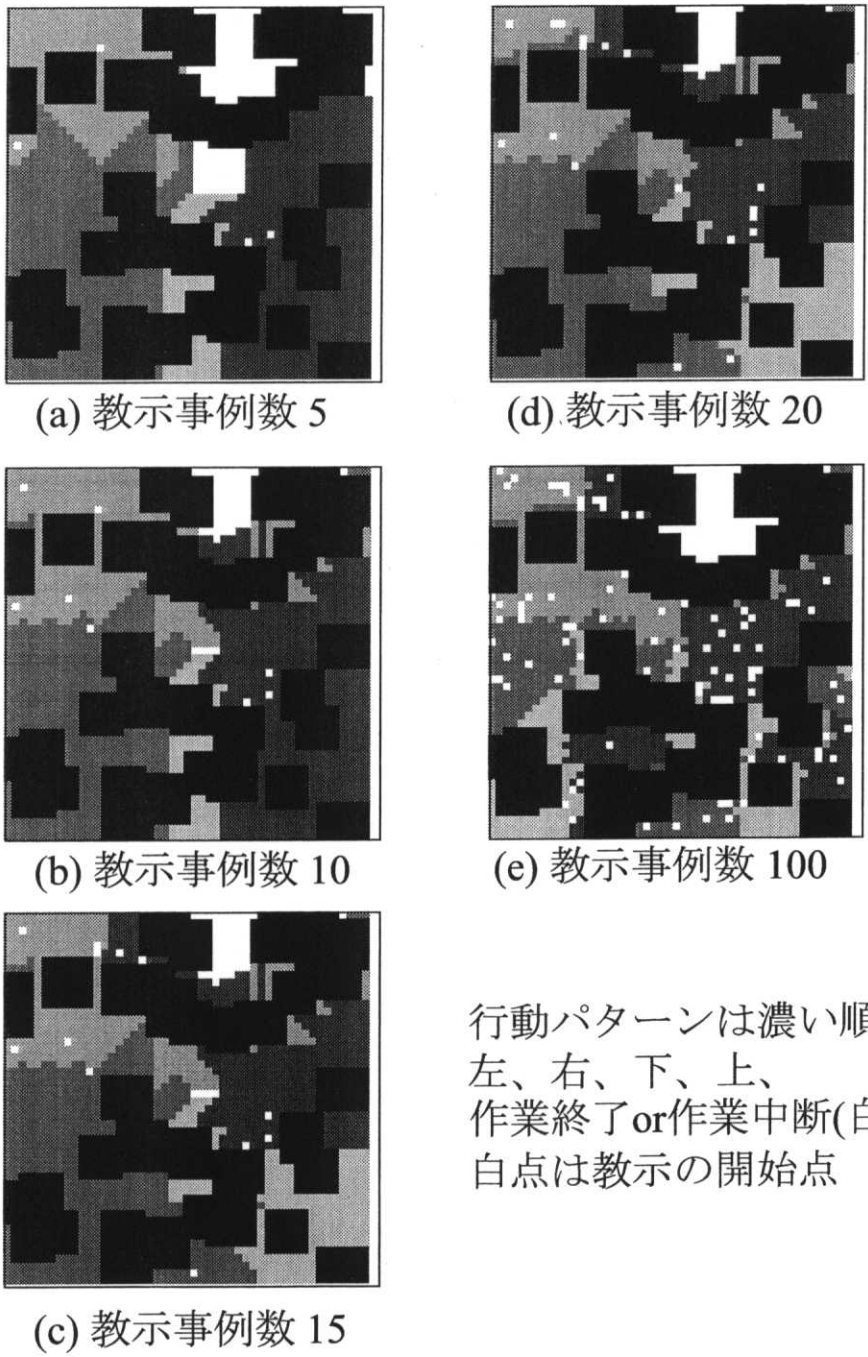


図 4.13: 教示事例数と生成ルールの変化

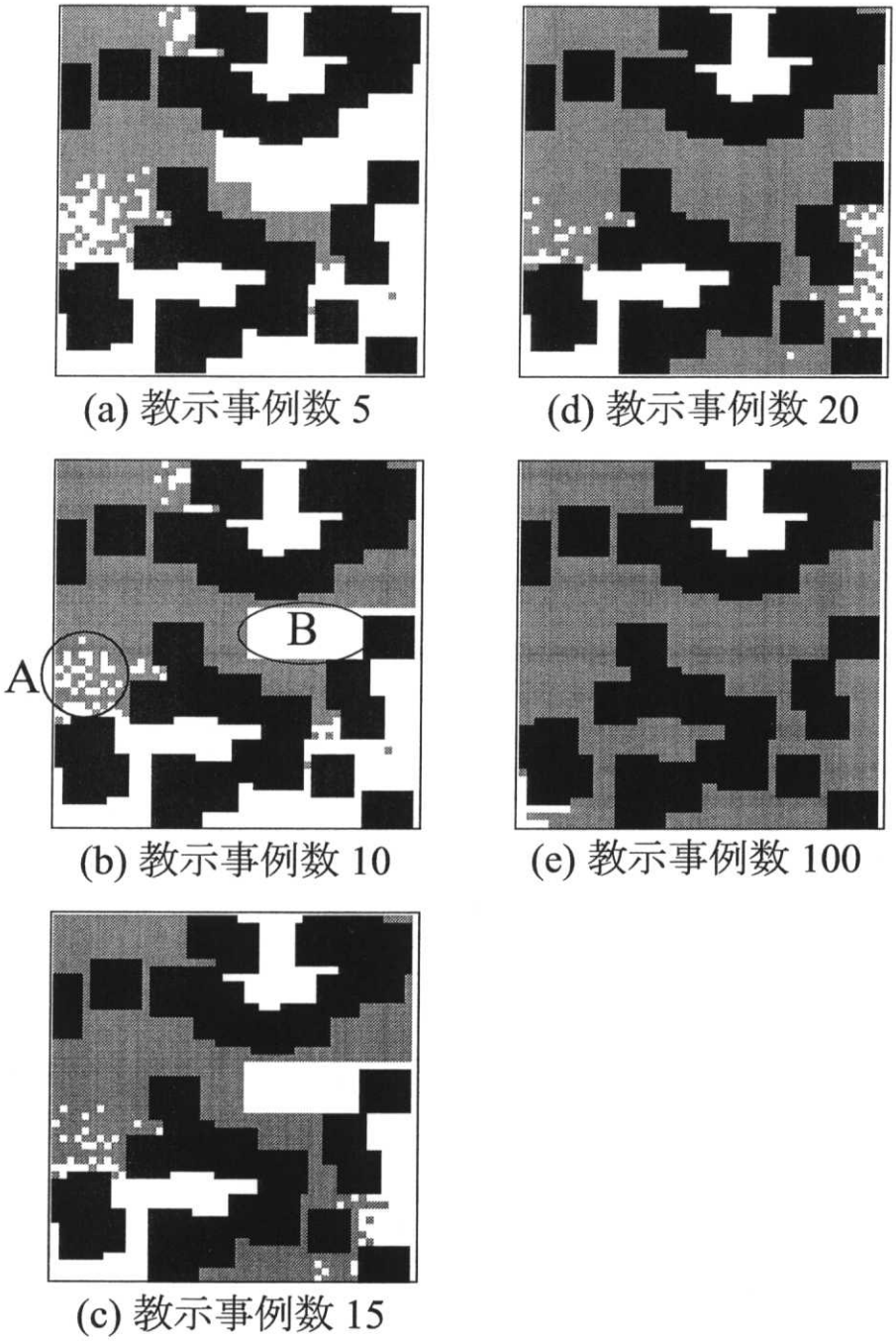


図 4.14: ゴール到達可能セルの分布の変化

スタート点を見ると、(d) までは A 内のセルをスタート点または通過点とする事例がまったく教示されなかったため、この部分には妥当な行動パターンが設定されていない。しかし、教示が進んで (e) になるとそのような事例も教示されたために A においてもゴールに到達できるような行動パターンが設定され、それらのセルからもゴールに到達できている様子がわかる。

Fig. 4.14(b) における B の部分は、Fig. 4.13では一見妥当なルールが生成しているにもかかわらず、ゴールに到達できていないように見受けられる。これはゴール付近に、ゴールと判断して停止するルールを保持するセルが 3 つ存在し、そのうちの不正なセルで停止したためである。作業においてゴールの判断条件がゆるやかな場合は、これらもゴールに到達したと判断してよいケースである。一方、ゴールの判断条件が厳しい場合には、最近傍法適用時にその条件を考慮に入れた対処が必要なことが判る。(付近のセルでは停止をルールとして選択しないようにするなど)

4.7.4 他の手法とのルール作成効率性の比較

本手法では行動学習方式と比較してルール作成効率を上げることで、ロボットの消耗を抑制することが目標の一つであった。そこで、ここでは行動学習方式の手法と作業達成効率の向上速度を比較し、その目標を達成しているか検証する。

[実験 2]

実験 1 と同じ条件で作業を行った場合について比較した。行動学習方式の代表には強化学習で一般的に用いられる Q 学習 [60] を用いた。Q 学習において、報酬は行動を行なう度に -1 だけ与えられるものとし、ロボットができるだけ少ない行動数でゴールに達した時の報酬が最大になるように設定した。また、探査関数は

$$f = \begin{cases} \text{Q 値を最大にする行動} & \text{(現在までの試行回数／最大試行回数 の確率で選択)} \\ \text{ランダムな行動} & \text{(それ以外)} \end{cases}$$

で与えた。ただし、スタートからゴールのセルに到達するまでを1試行とし、最大試行回数はその都度設定し、Q値を学習させる。つまり、教示回数n時のルールを学習させるのに、教示回数$m(m < n)$時の結果に追加して学習させることはしない。これは最大試行回数が異なると、探査関数も異なるためである。

Q学習でさまざまな最大試行回数を設定して作業達成率を調査した。その最大試行回数と作業達成率の関係をプロットして、本手法と比較したのがFig. 4.15である。本手法のプロットは「提案手法(ランダム)」と記載された折れ線である。このグラフから、本手法による作業達成率の増加はQ学習と比較してはるかに速く、ルール作成効率に優れているという結論が得られた。

本手法のルール作成効率性は数値からも見積もれる。

事例教示を行った場合、ゴール到達率が100 %に達した70回時点でマナーデータベースに蓄積されたマナー数は4216個であった。この中には同じセルに重複して教示を行なったものも含まれる。一方、Q学習で、ある程度正確な効用値を見積もるためにQ値1つあたり少なくとも20回の選択操作が必要だと仮定し、環境中で環境物体によって占有されていないセルの総数を5000個と見積もった場合、行動の種類は6種類なので、Q学習に必要な選択操作の総数は$5000 \times 6 \times 20 = 600,000$は必要になる。この選択操作が本手法における事例教示の選択操作(マナーの数)と同等とした場合、必要数は本手法のほぼ150倍にあたる。これから見ても本手法のルール作成効率性がよいことが判断される。

4.7.5 人間の選択的教示による効果

実験1ではスタートするセルを計算機がランダムに与えていたが、実際にはオペレータが作業を行なう事例を選択し、サンプルを満遍なく与えることで効率的に教示することが見られる。ここではその効果を定量的に検証した。

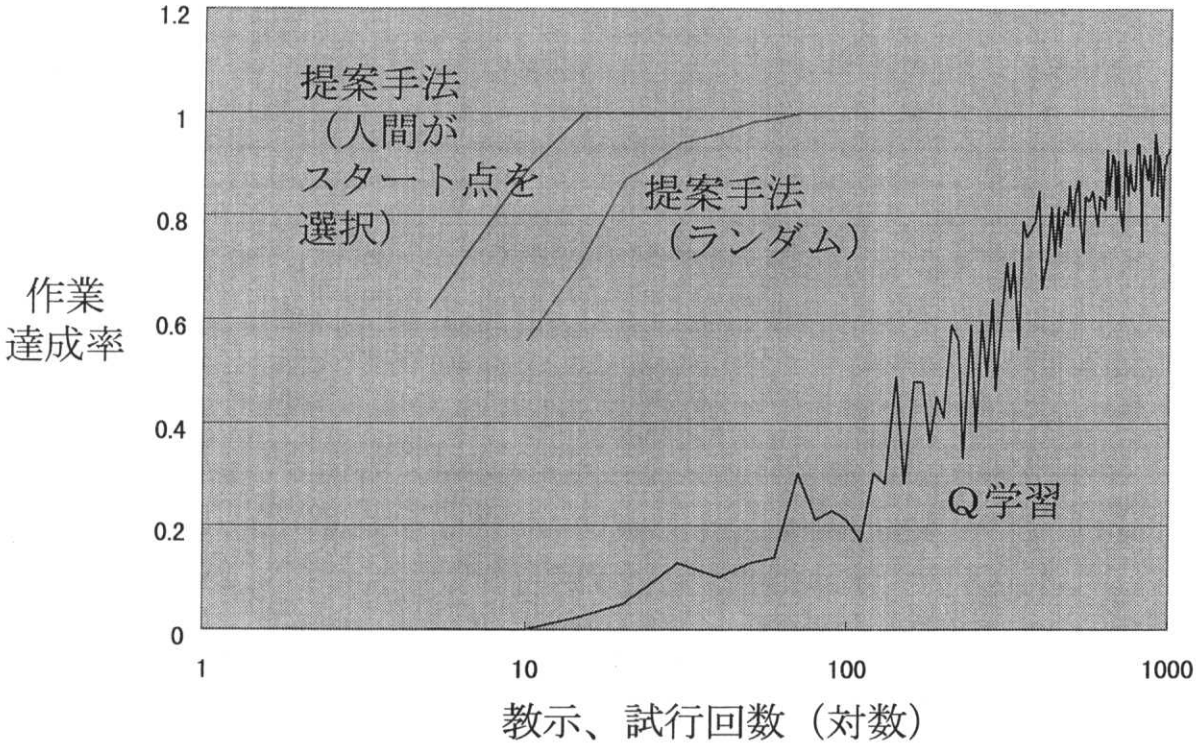


図 4.15: 提案手法とQ学習とのルール作成効率比較

[実験３]

実験１において、計算機でランダムにスタートするセルを与えて教示した場合と、人間がスタートするセルを選んで教示した場合とで、作業達成率の向上速度がどの程度異なるのか調査した。本実験においてFig. 4.16に人間がスタートとして与えた点を掲げる。

実験１同様、教示回数と作業達成率の比較を行った。その結果、人間がスタートするセルを与えた場合は教示回数15回(教示回数の軸は5回刻み)で作業達成率100%となった。その様子をFig. 4.15に併掲する。このように、人間が選択的に事例を教示することで、ルール作成の効率がさらに上げられることが確認された。

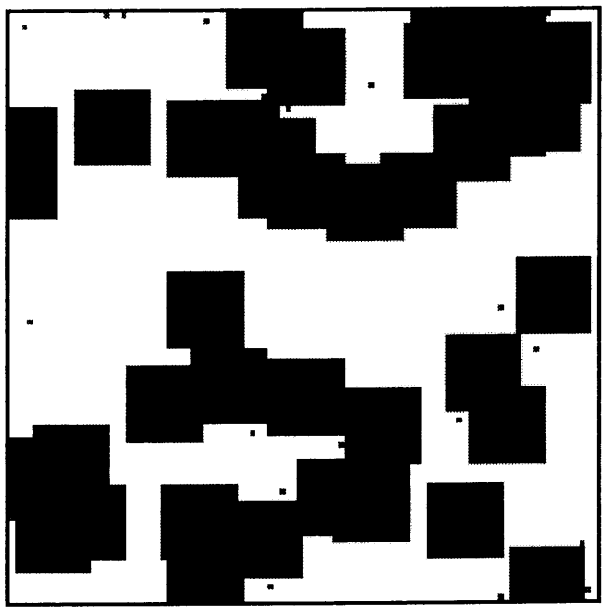


図 4.16: 人間がスタートとして与えた点

4.8 ルール生成手法の検討

第4.5節では、パターン認識に用いられる主要な手法として最近傍法、決定木、ニューラル・ネットワークを挙げた。ここではそれらの方法を比較し、移動操作作業におい

て実例教示からルールを作成するのに適した手法の検討を行なう。

以下に、それぞれの手法の紹介を行なう。

最近傍法 (nearest neighborhood)

最近傍法はパターン認識において用いられる最も基本的な技法のひとつである [15]。データベースにパターンとそれの属するクラスを組にして蓄積しておき、テストパターンが入力された場合に、データベースを検索してパターンの最も類似したものの属するクラスをテストパターンのクラスとするものである。クラスを決定するために、最も類似したパターンをひとつだけ参照する方式を 1- 最近傍法と呼び、最も類似した k 個のパターンを参照するものを k- 最近傍法と呼ぶ。類似性は、例えばパターン空間に距離を定義してその大小を比較することなどで評価する。単純で直感的に理解しやすい方法であるが、パターン認識分野で行われたベンチマークでは最も優秀な性能を示したという報告もある [42]。

ニューラル・ネットワーク (neural network)

ニューラル・ネットワークとは人間の脳の神経組織の構造をモデルとして情報処理に応用しようというものである [2]。ニューラル・ネットワークはニューロンと呼ばれるユニットの間をシナプス結合した構造になっている。ニューロンには外部からの情報入力を受理するものと、外部へ結果を出力するものが存在する。

情報が入力されたニューロンは、その値に対して処理を行ない、シナプス結合された他のニューロンに結果を送出し、それを受理したニューロンが処理を行なうことを繰り返す。最終的にニューロンが外部に出力する値が結果となる。

ニューラル・ネットワークを用いた情報処理はトレーニングデータが豊富に得られる分類問題や関数近似問題の解決に向いているといわれている。パターン認識には、フィードフォワード型ニューラル・ネットワークが一般に用いられる。フィードフォワード型ニューラル・ネットワークはニューロンが層をなし、シナプス結合が前層か

ら後層のみに存在し、フィードバックがないようなものである。

入力層、中間層、出力層の3層構造をしたニューラル・ネットワークが用いられることが多い。

決定木 (decision tree)

決定木は、帰納学習の分野で研究された手法である [50]。決定木は二分木構造をしており、テストパターンは木の根から入力される。木の各非終端節ではテストパターンに対するテストを行ない、その結果によっていずれかの枝を進むか選択される。終端節は、そこに到着したときに返すべきクラスが記載されている。

以下では、本方式に対する各手法の実装方法を述べ、その特徴を比較することで、本方式に適した手法の検討を行う。

最近傍法

最近傍法を用いる場合、第4.5節で述べたように、分析モジュールは存在せず、実行モジュールにおいて直接マナーデータベースを利用する方式となる。

実行モジュールでは現在ロボットの属する状態プリミティブの拘束の符号のリストを入力し、最近傍法ではそれを拘束空間上の点として扱い、それに最も距離の近いサンプルをマナーデータベースから検索する。距離は拘束の符号の相違数により定義する。

拘束空間で最短距離にあるサンプルで選択した行動が現在において選択すべき行動と判断される。

しかし、入力される拘束の中には有効なものと無効なものが存在し、それはあらかじめ区別することは不可能である。最近傍法では、無効な拘束がかく乱要因となって行動決定が正しく行なえない可能性があるという問題がある。すなわち、有効な拘束と無効な拘束の符号をそれぞれ d_i $(i = 1 \cdots m)$, $\bar{d}_j$ $(j = 1 \cdots n)$ とする。このとき

拘束空間の軸を適当に並べ替えて

$$(d_1, d_2, \cdots, d_m, \bar{d}_1, \bar{d}_2, \cdots, \bar{d}_n)$$

と表現しても一般性を失わない。このとき、同じクラスに属するサンプルとは第1成分から第 m 成分までの値が共通なので、それらの m 個の成分を足した値は0である。しかし、それ以降の n 個の成分によって、距離に最大 n の誤差が加わる。そのため、無効な拘束が存在する場合は行動を誤選択する可能性が高い (Fig. 4.17)。

対策としては、マナーデータベースに蓄積するデータの数を増やすことが考えられるが、教示の手間が大幅に増大し、提案手法の利点が失われる。

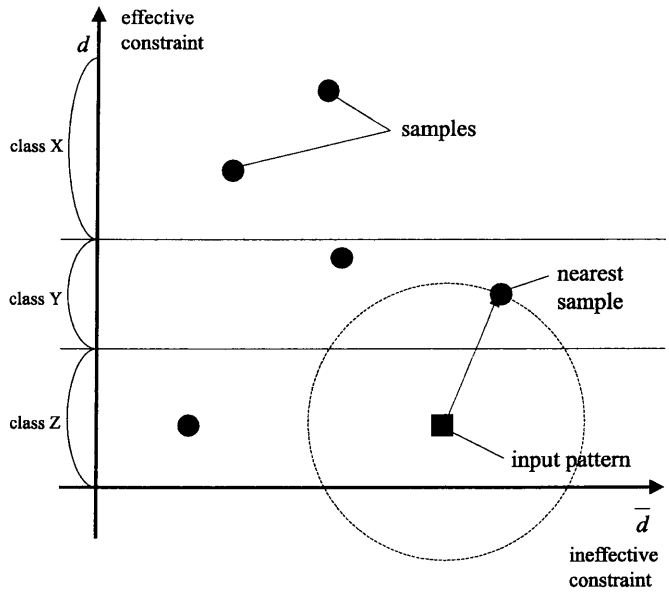


図 4.17: 最近傍法によって行動選択に誤りが生じる例

ニューラル・ネットワーク

本方式にニューラル・ネットワークを適用する場合、トポロジーはパターン認識に一般的に用いられるフィードフォワード型を用いる。その構造はFig. 4.18のようになる。状態プリミティブの拘束の数をmとし、ロボットが選択できる行動の数をnとする。この場合、入力層にはm個のノードが必要である。各入力ノードには1個ずつ拘束を対応させる。また、出力層にはn個のノードを用意し、それぞれをロボットの行動のひとつに対応させる。各入力ノードに拘束の符号を入力したとき、出力値が最も大きいノードに対応した行動が選択すべき行動となる。

適切な行動を出力させるために、分析モジュールにおいて、マナーデータベースに蓄積されたデータを教師データとしてバックプロパゲーション学習[52]を行なわせる。

ニューラル・ネットワークは汎用性の高い手法だが、学習に大きなトレーニングセットを必要とする。

また、学習によって獲得されたルールは人間にとってはブラックボックスである。ニューラル・ネットワークは、設計や学習に洗練されたノウハウが必要な方法でもある。例えば、中間層におけるノード数の設定が少なすぎれば分類能力が低下するし、逆に多すぎれば過学習に陥る[31]。

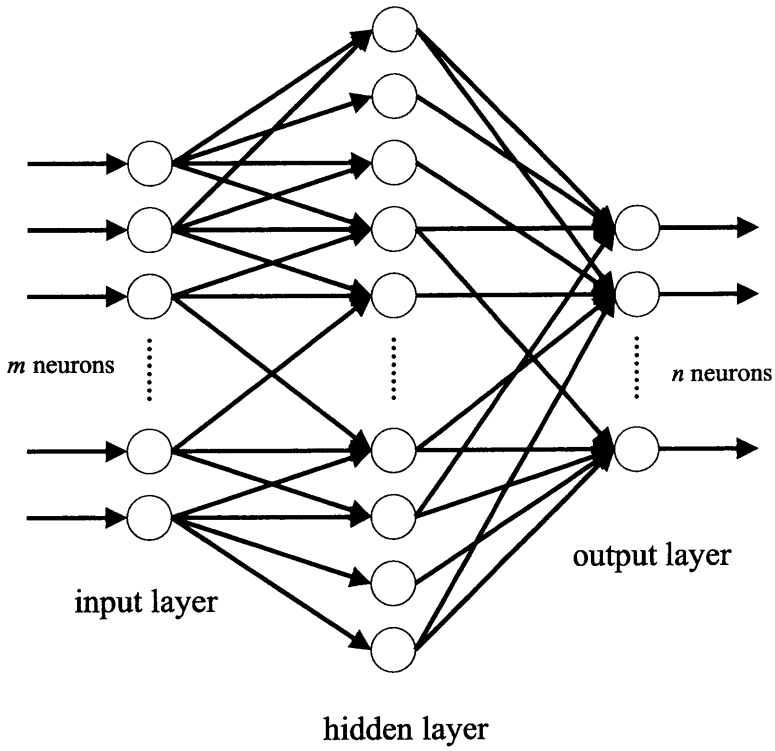


図 4.18: フィードフォワード型ニューラル・ネットワーク

決定木

決定木を使用するには、分析モジュールにおいて、マナーデータベースに蓄積されたデータを用いて決定木を生成する。決定木はアルゴリズムによって機械的に生成でき、ニューラル・ネットワークにおけるトポロジー決定のようなノウハウは必要ない。また、決定木は人間に可読性のよいという優れた性質を持っている。入力に無効な拘束が含まれていても最近傍法のように判別には影響しない。大量のデータの規則性を高速に見出す方法として優れた方法である [3]。このようなことから、本手法に対しては決定木を採用することが適当と判断する。

4.9 決定木生成アルゴリズム

本節では決定木生成アルゴリズムについて説明する。

入力されたパターンベクトルの成分の値から真偽を判別する操作をテストと呼び、テスト全体の集合を T と表す。

テストの数はパターンベクトルの次元数と等しい。決定木において、葉がテストに対応するとき、その葉を活性節と呼ぶ。データベースに蓄積されているパターンベクトルとクラスの組の集合を D、決定木を H で表す。

H は $H = D$ または $H = (t, H_0, H_1)$ という形で記述される。前者は、パターンベクトルとクラスの組の集合 D で、クラスがすべて共通なことを意味する記述である。また、後者は、t が根節におけるテストであり、H_0, H_1 は t の判別結果がそれぞれ真と偽の場合の部分木であるという記述である。部分木もまた決定木の形をしており、H_0, H_1 はそれぞれ t の判別結果がそれぞれ真と偽となったパターンベクトルとクラスの組を持つ。このとき、以下の手順で決定木を生成できる。

1. D が空集合、または D のパターンベクトルとクラスの組がすべて同一クラスに分類されるなら、$H = D$ として停止、H が求める決定木である。

2. T の中からテスト t を選択し、それに対する真偽判別結果によって D を D_0, D_1 の 2 つの集合に分類する。

3. D_0, D_1 に対してテスト集合 $T' = T - \{t\}$ を作成しそれぞれについて決定木 H_0, H_1 を作成する。 $H = (t, H_0, H_1)$ が求める決定木である。

ここで、2 において T の中からテスト t としてどれを選択するかによって生成する決定木の形状が変わってくる。 D の分類効率をよくするにはできるだけ単純な決定木が生成されることが望ましい。そのようなテストの選択基準のひとつに、情報量に基づくものがある [3]。

決定木 H が D を m 個のクラスに分類するものとし、クラス i に属するパターンベクトルとクラスの組の数を d_i とする。このとき、 H が単にパターンベクトルとクラスの組の数に応じて確率的にクラスを出力するだけの機械だとしたら、入力されたパターンベクトルがクラス i に分類される確率は

$$\frac{d_i}{\sum_{j=1}^{m} d_j}$$

である。このような機械の伝達する情報量は

$$E(d_1, \cdots, d_m) = -\sum_{i=1}^{m} \left\{ \frac{d_i}{\sum_{j=1}^{m} d_j} \, log_2 \left(\frac{d_i}{\sum_{j=1}^{m} d_j} \right) \right\} \tag{4.1}$$

で与えられる。

ここで根節においてテスト t が選択され、部分木が H_0, H_1 となった場合の H の情報量は

$$E'(t) = \frac{\sum_{i=1}^{m} d_{0i}}{\sum_{i=1}^{m} d_i} E(d_{01}, \cdots, d_{0m}) + \frac{\sum_{i=1}^{m} d_{1i}}{\sum_{i=1}^{m} d_i} E(d_{11}, \cdots, d_{1m}) \tag{4.2}$$

である。 d_{0i}, d_{1i} はそれぞれ H_0, H_1 においてクラス i に属するパターンベクトルとクラスの組の数を表す。

式 (4.1) と式 (4.2) を比較すると、後者に情報量の増加があった場合に、それだけテスト t の選択が効果的な分類を行なったと判断できる。そこで、

$$\max_t \{E(d_1, \cdots, d_m) - E'(t)\}$$

を与えるような t を採用するというのが情報量を基準とした選択方法である。

この方法はパターンベクトルとクラスの組の数に対して計算オーダが線形なので、大量に教示を行った場合でも効率的な分類が可能である。

4.10 実験

4.10.1 実験システムの概要

本節では、以上で述べた提案手法に基づいて、移動ロボットに対して行動選択方式で移動操作作業を行なうルールを事例教示から作成し、本手法の有効性を確認する。今回製作したロボットシステムの実験系を Fig. 4.19に示す。

環境情報を取得するセンサ部は天井に取り付けた CCD カメラ、富士通社製のトラッキングビジョンボード [58]、および自作のトラッキングプログラムにより構成される。CCD カメラの画像はトラッキングビジョンに NTSC 信号として入力される。トラッキングビジョンはハードウェアを用いた相関演算によって、物体上に指定した特徴点をビデオレートで追跡する。これによって、トラッキングプログラムは物体やロボットの画像上での位置・姿勢などの特徴量を算出できる。

移動ロボットは LEGO 社の Mindstorms というロボット製作キットを用いて製作した (Fig. 4.20)。2 輪はそれぞれ独立なモータによって駆動され、前進、後退、右回り、左回り、停止の 5 種類の行動パターンをプログラムした。これがルールで選択可能な行動となる。ロボットへの行動命令は RS-232C ポートに接続された赤外線タワーを介して計算機から行動の種類を表す 1 バイトの信号を送ることで行なう。

システムの主だった処理を行う計算機にはCPUのクロック数160MHzのDOS/V互換マシンを用いた。

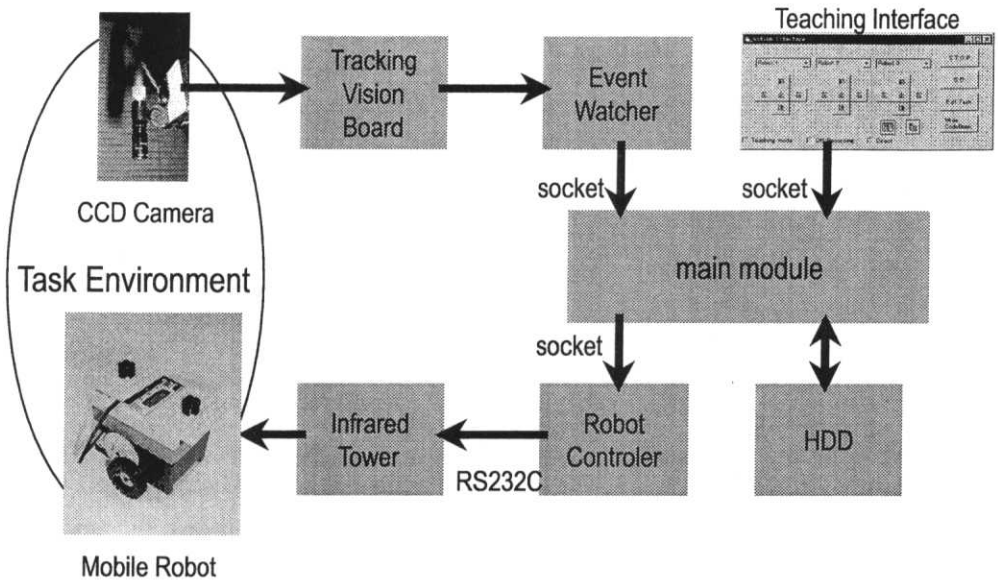


図 4.19: 行動選択方式で作業を行なうロボットの実験系

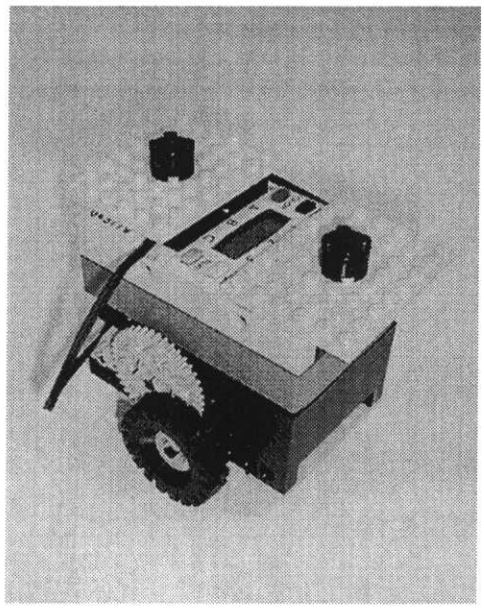


図 4.20: 実験に使用した移動ロボット

4.10.2 環境モデル

ここでは本実験で用いた環境モデルについて述べる。

まず、Fig. 4.21の環境をモデル化し、Fig. 4.22のように５つの拘束域を設けた。Fig. 4.22に示されたAからEの文字は拘束域をアイデンティファイするための便宜的な記号である。また、ロボットも概形を長方形としてモデル化する。

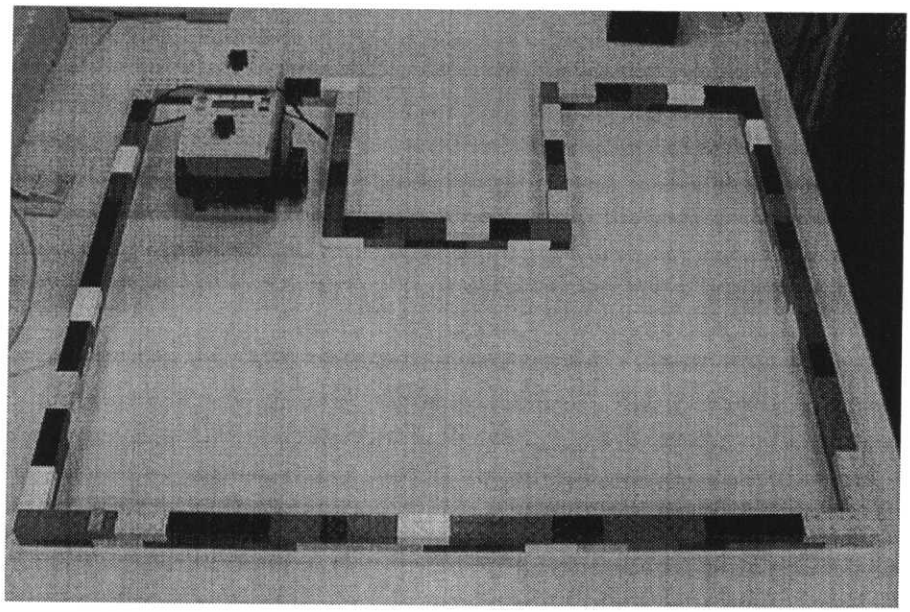


図 4.21: モデル化を行なう環境

拘束域はそれぞれ環境の壁に関する４つの特徴点の２次元座標、およびロボットに関する２つの特徴点の２次元座標によってモデル化する (Fig. 4.23)。すなわち、これらの拘束域はそれぞれ１２個の特徴量で表される。拘束域の間で特徴点が重複しているものも存在するが、環境のモデリングに対する影響はない。

ある拘束域に属しているかどうかの判定は、ロボットの２つの特徴点の中間点が、その拘束域の壁に関する４つの特徴点の形成する４辺形内部に存在するかどうかで判

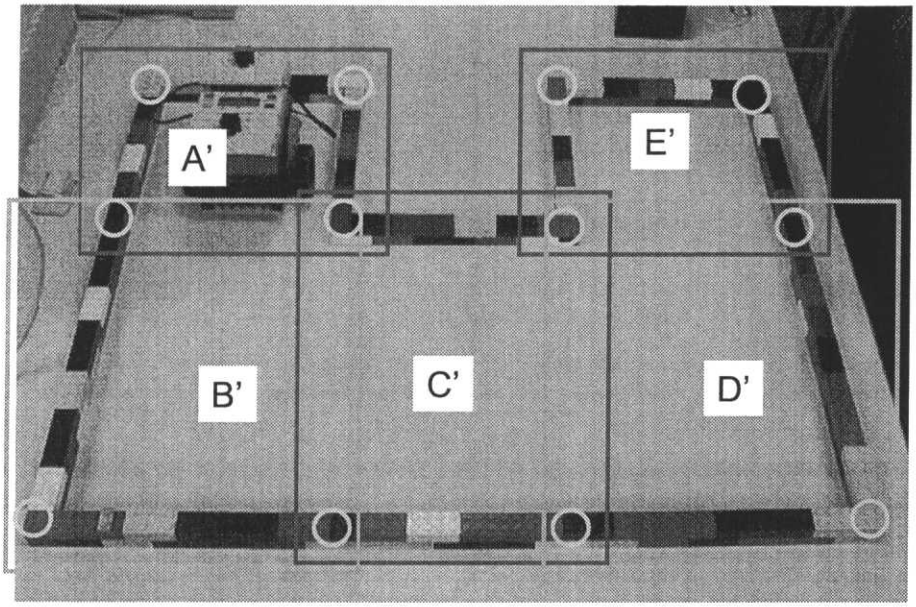


図 4.22: 環境に設けた拘束域

図 4.23: 拘束域で検出する特徴量

断する。もしも内部に存在していればその拘束域に存在していると判断する。

拘束域のうち、AからDは形状と特徴量を互いに対応させることが可能であり、局所的な作業が廊下の通行という共通の内容なので、ルールセットを共有可能である。Eに関しては作業内容がゴールでの停止であり、他と異なるので、ルールセットは別個になる。それぞれのルールセットを便宜的にGO_THROUGH, PARKと呼ぶ。

4.10.3 ルールの作成

前節のようにモデル化した環境において、事例教示を行ないルールを作成する方法を述べる。

事例教示を行なっている間、環境の特徴量は天井に設置されたCCDカメラから取得する。具体的には、作業に先立って画像上で特徴点を指定しておき、トラッキングビジョンで追跡させる。これによって、ロボットが移動したり、環境が変動したりしても、常に特徴量の取得が可能である。

また、ロボットが選択可能な行動として前進、後退、右回り、左回り、停止の5種類のパターンを用意した。オペレータは計算機のキーボードを通じて行動の選択を行ない、計算機はロボットにその行動を通知するとともに、現在の状態プリミティブと選択した行動を関連付けてマナーデータベースに保存できる。

本実験ではGO_THROUGH, PARKという2種類のルールセットを作成するので、マナーデータベースはそれらに対応して2種類存在する。各拘束域では、対応するルールセットに応じたマナーデータベースに記録が行なわれる。

拘束の検出は以下のように行なう。第3.3.1節で述べたように、2次元環境における拘束は2種類あるので、センサ部から検出した特長量を用いて以下の2種類の方法で拘束を求め、その拘束が変化した場合にルールを適用して新たな行動を選択する。

type A

ロボットの辺の式はその特徴量からロボットの位置と姿勢を求め、ロボットの形状モデルに当てはめることで求められる。得られた辺の画面上での式を

$$ax + by + c = 0$$

とする。一方、環境物体の画面上での頂点 (e_x, e_y) は特徴量から直接得られる。このとき、前の辺の式に頂点の座標を代入した

$$ae_x + be_y + c$$

の正負の変化を観測することで、状態プリミティブの変化をその拘束の符号の変化として検出できる。観測のイメージを Fig. 4.24 に示す。

type B

環境物体の 2 つの特徴点から、環境物体の辺の画面上での式が求められる。得られた式を

$$ax + by + c = 0$$

とする。一方、操作物体の画面上での頂点 (r_x, r_y) は特徴量からロボットの位置と姿勢を求め、ロボットの形状モデルにあてはめることで求められる。このとき、

$$ar_x + br_y + c$$

の正負の変化を観測することで、状態プリミティブの変化をその拘束の符号の変化として検出できる。観測のイメージを Fig. 4.25 に示す。

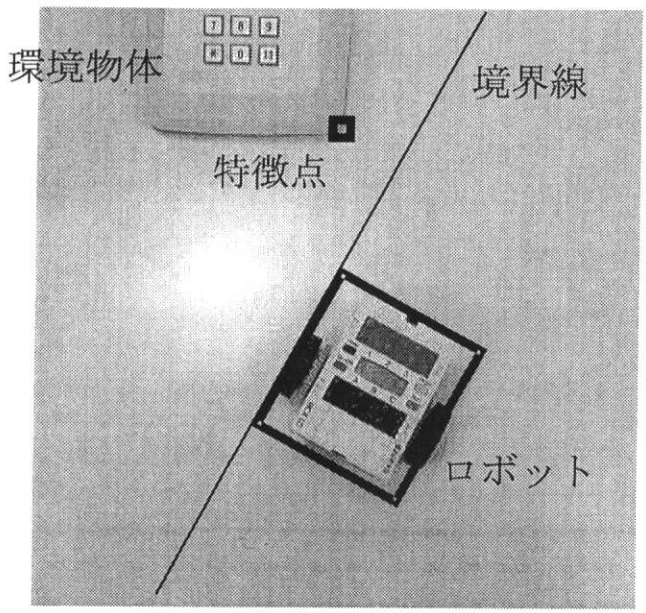


図 4.24: タイプ A の観測イメージ。特徴点が (e_x, e_y)、境界線が $ax + by + c = 0$ に相当する。

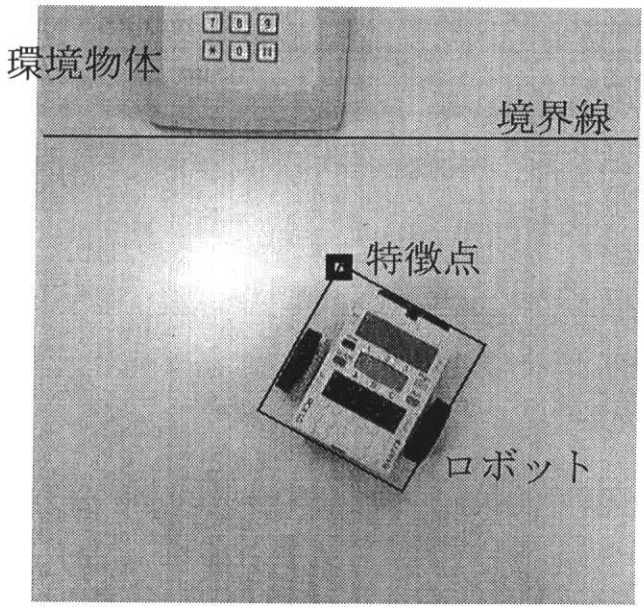


図 4.25: タイプ B の観測イメージ。特徴点が (r_x, r_y)、境界線が $ax + by + c = 0$ に相当する。

本実験ではローカル環境モデルの辺や頂点の個数は4個であり、ロボットの辺や頂点の個数も4個である。したがって、そこにおける拘束の個数は32個であり、存在しえる状態プリミティブの種類は2^{32}となる。それゆえ、各状態プリミティブに対応する行動を教示することでルールを作成することは、このレベルの問題でも現実的ではないことがわかる。そこで、ここでは決定木を作成することで、現実的な量の教示からルールを作成することを図る。

決定木を作成するために実際に事例教示を行ない、GO_THROUGH, PARKのマナーデータベースにそれぞれ451個、93個のデータを蓄積した。それぞれに対して決定木生成アルゴリズムを適用してルールを作成した。決定木アルゴリズムの適用にはOC1というフリーウェアのプログラムを利用した[47]。

生成した決定木をそれぞれFig. 4.28, Fig. 4.29に示す。これらの図では左端のボックスが根節となり、右端のボックスが葉となる。ロボットや環境の頂点にはFig. 4.27のようにラベルをつけている。

例えば、Fig. 4.28の根節で14-Dとあるのはロボットの頂点1と4を通る辺に対して環境の頂点Dがロボットと反対側か同じ側にあるか判定するテストを表す。このテストではDがロボットと反対側にある場合は真を返し、それ以外の場合は偽を返す。このテストが一つの拘束の符号をチェックする手続きに相当する。真の場合は右側の分岐を上に進み、23-Cと書かれた節で同様のテストを行なう。偽の場合は分岐を下に進み、「右折」と書かれた葉に至る。葉に至った場合は、そこに書かれた内容がそのときに選択すべき行動となる。

例えば、テスト14-Dに関する拘束の符号s(14-D)と表した場合、Fig. 4.28の決定木は次のようなif-then型ルールに書き下せる。

if s(14-D) $>$ 0 and s(23-C) $>$ 0 then 前進
if s(14-D) $>$ 0 and s(23-C) $<$ 0 then 左折
if s(14-D) $<$ 0 then 右折

また、 Fig. 4.29の決定木をクラスタリングによって表現したものをFig. 4.26に示す。左側の絵はロボットの姿勢を示す。右側の図はロボットがその姿勢のときに、ローカル環境の各地点において選択される行動を示す。

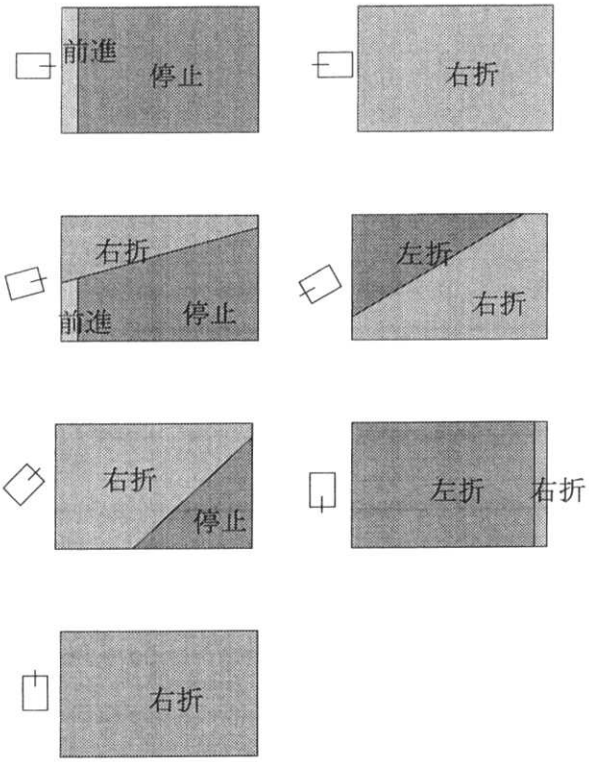


図 4.26: 生成した決定木のクラスタ表現

このルールを見てもわかるように、有効な拘束の数は決定木のテストの総数と同じになる。したがって、 GO_THROUGH, PARK のルールではそれぞれ 32 個の拘束のうち 30 個、 28 個が無効な拘束と判断され、 2 個、 4 個が有効な拘束と判断されていることがわかる。

なお、教示データには実際にはノイズが存在するため、単純に決定木生成アルゴリズムを適用しただけでは、生成した決定木はより複雑な構造になる。これはオペレータの教示のゆらぎ・ミスや、実際の作業に対してモデルが完全には整合していないこ

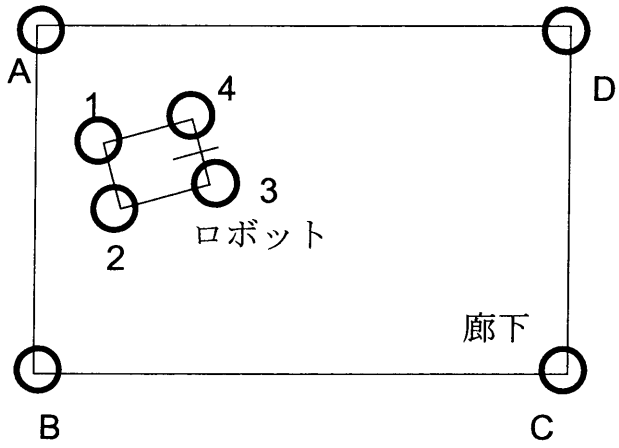


図 4.27: ロボットや環境の頂点と辺

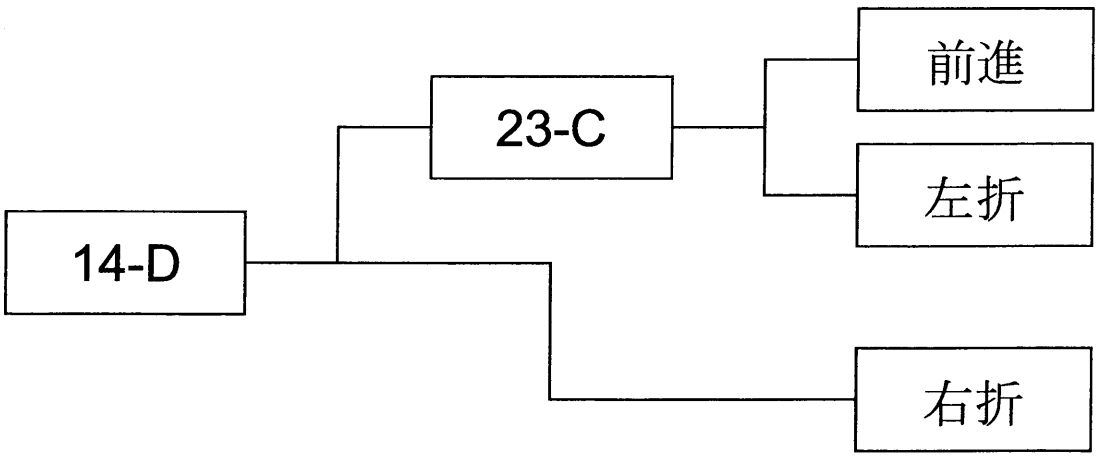


図 4.28: GO_THROUGH の決定木

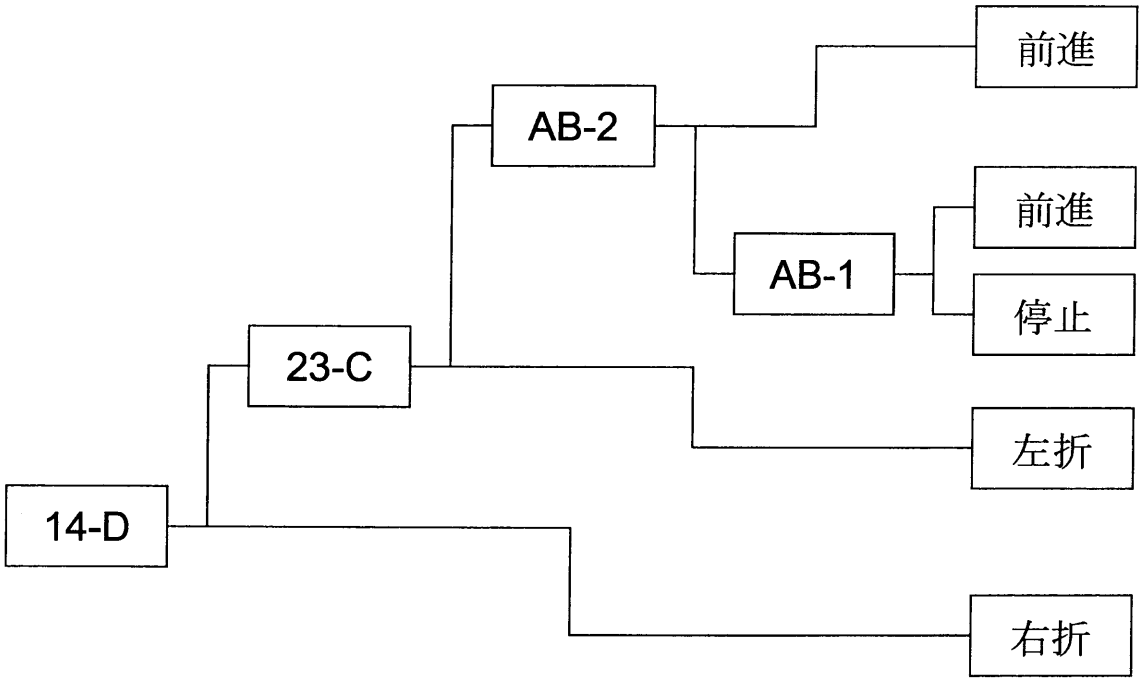


図 4.29: PARK の決定木

となどが原因と考えられる。図に掲げたものは、そのうち、事例数の少ない些細な枝を刈り取った結果である。マナーデータベースに蓄積されたデータが、作成したルールと適合する割合はGO_THROUGH, PARKの場合でそれぞれ94 %、92 %であった。すなわち、残りの6 %, 8 %のデータはノイズと判断された。

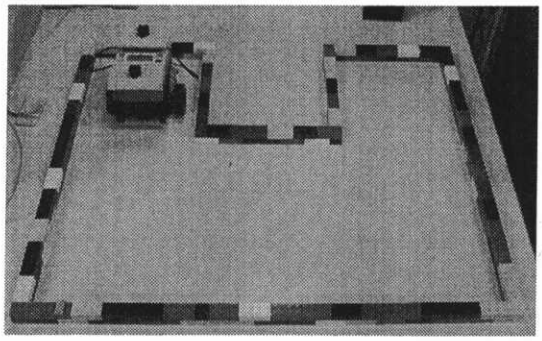
4.10.4 作業の実行

前節で作成されたルールが実際に有効なものか確認するために、そのルールを用いてロボットに実際に移動操作作業を行なわせた。その様子をFig. 4.30に示す。

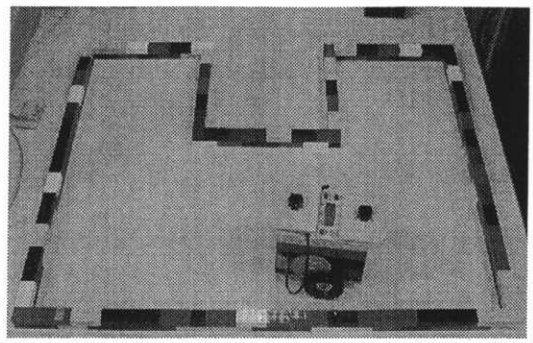
また、上の実験に続けて、ロボットを意図的に別の場所に移し、強制的に作業を続行させた。その様子をFig. 4.31に示す。

Fig. 4.30ではロボットが適切な位置でルールを適用し、適切な行動に切り替えながらゴールに向かい、ゴールの判定も正しく行ない、そこで停止している様子がわかる。また、Fig. 4.31からは、ロボットがシーケンシャルに教示された手順で作業を実行しているのではなく、センサ情報に基づいてそれにマッチするルールを選択して実行している様子がわかる。

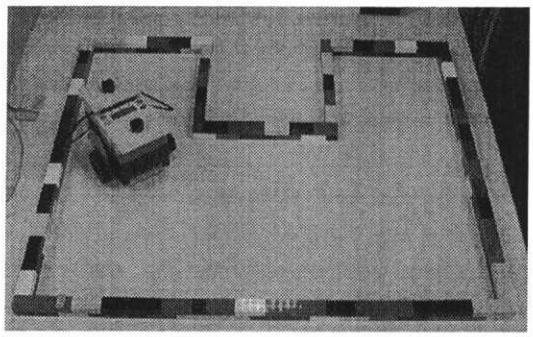
このように、本手法を用いることでオペレータの事例教示からロボットに適切なルールを作成可能なことが確かめられた。



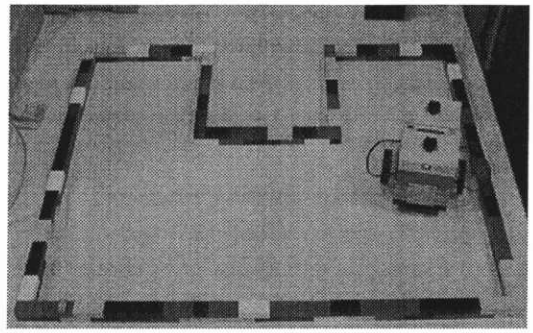
(a)



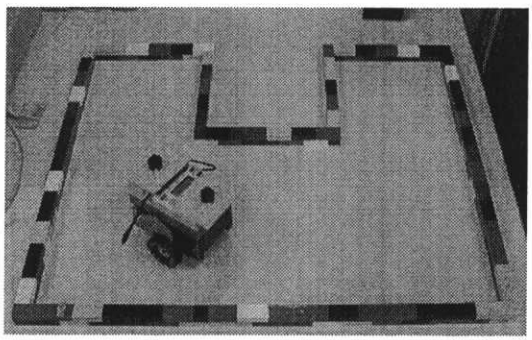
(d)



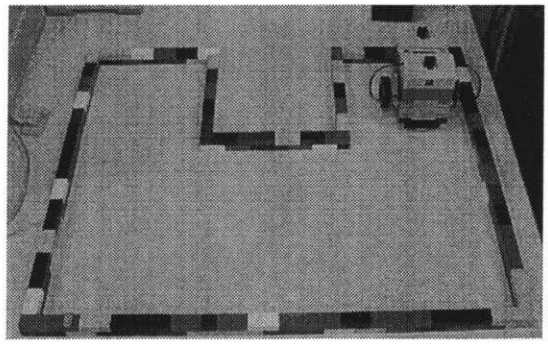
(b)



(e)

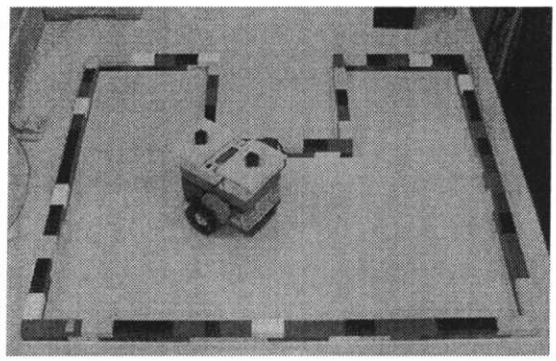


(c)

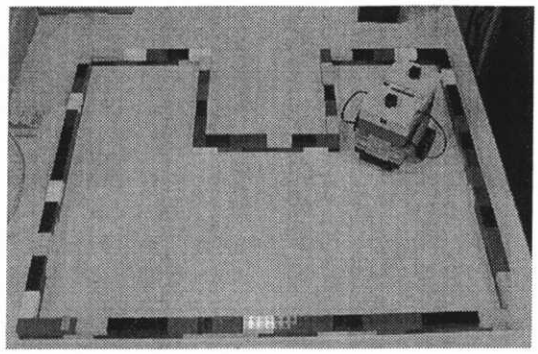


(f)

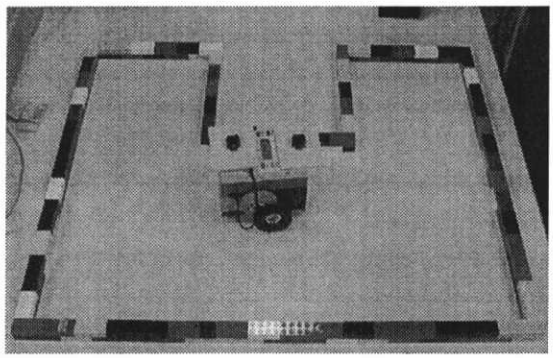
図 4.30: ロボットが作業を実行している様子



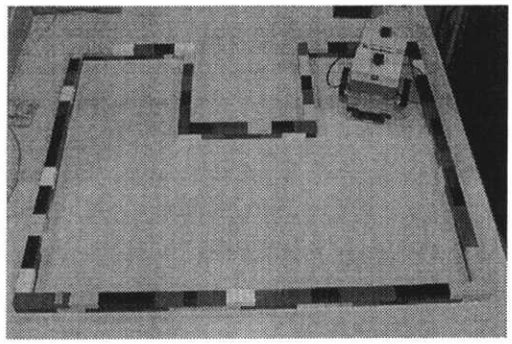
(a)



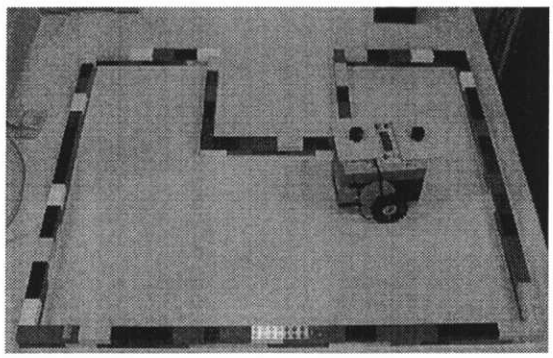
(d)



(b)



(e)



(c)

図 4.31: ロボットが作業を実行している様子 (承前)

4.11 おわりに

本章では、行動選択方式で操作を行うロボットにおいてオペレータの示す事例教示から移動操作作業のルールを自動的に作成する方法を検討した。

第4.2節ではルール作成方法の概要を述べた。行動選択方式の場合は、ルールの条件部は拘束条件を判定するテストで記述され、実行部はあらかじめ用意された行動パターンのうちの一つを出力する仕組みになっている。条件部においては、事例教示で得られた情報を分析して、候補となる拘束の中から有効なものを見出す手法を考案することが中心的課題となる。ここでは、それに対してパターン認識の手法を利用することを考えた。

第4.3節では提案した手法を評価する方法を検討した。

第4.4節では、オペレータの行なった事例教示からデータを収集する教示モジュールのシステム構成について検討した。また続く第4.5節では、収集したデータを分析してルールを作成し実行する分析モジュール、実行モジュールのシステム構成について検討した。

第4.6節では、事例教示が不充分だったり、操作が失敗したときにリカバリを行うデフォルト・ルールについて説明した。

第4.7節ではシミュレーション・プログラムを作成し、提案方式の有効性を第4.3節で提案した評価方法に従って調査した。ルール作成手法に最近傍法を用いて、教示回数と作業達成率の向上の評価、作業達成率が適正な水準に達するかの調査、他の手法とのルール作成効率の比較、人間の選択的教示の効果の調査を行った。提案手法が評価を満足した優れた性質を持った手法であるという結果を得た。

第4.8節では、ルール作成に用いる手法のより具体的な検討を行った。最近傍法、ニューラル・ネットワーク、決定木の３つの手法について実装方法を検討し、特徴の比較を行った。その結果、決定木を用いた方法が本論文の目的に最も好適な特徴を持っ

ていることが確かめられた。第4.9節ではそれを受けて、決定木生成アルゴリズムの詳細を説明した。

第4.10節では、決定木生成アルゴリズムを計算機に実装し、移動ロボットとカメラを用いたシステムで本手法の有効性を確かめる実験を実施した。移動ロボットに対して事例教示を行い、実際に移動操作作業に有効なルールを、決定木を用いて作成可能なことを確認した。