動的環境における自律エージェントによる複数のロボットを用いた協調動作の実現

理学部情報科学科 4 回生 小山 由

平成 23 年 2 月 14 日

概要

近年ロボットは、市場において非常に安価に購入できるようになり、個人でも身近な存在になりつつある。しかしその一方で、ロボットの制御は依然として特定の応用に対するチューニングを主体としたものが多く、汎用的な自律エージェントによる動作制御を目指した研究はあまり行われていない。そこで我々は、BDI エージェントを用いることで、動的環境における複数のロボットの制御システムの実現を目指し、その一環として、複数のロボットの協調動作の実現を行った。例題としては、縦 4 ×横 5 マスのフィールド内でのロボットの制御に成功した。本論文では、我々が作成した制御プログラムにおける、探索・探索続行の機能とその実装結果について示す。

1 はじめに

本研究は、奈良女子大学 4 年隅田との共同研究である。

人はそれぞれに自分の信念を持ち、状況の変化に動的に対応し、意志決定をすることで目標達成に向けた一貫的な行動をとることができる。この考え方を応用し、信念 (**B**elief)、願望 (**D**esire)、意図 (**I**ntention) の三つの心的状態パラメータを用いて、合理的かつ自律的なエージェントを実現したものが BDI エージェントである。BDI エージェントは人間の思考をモデル化し、エージェントとして実現することで、作業を行う際は常に熟考しながら自律的に行動選択を行う。この方法を用いることにより、エージェントは矛盾のない一貫した行動をとることが可能となる。

近年ロボットは市場において、個人でも非常に安価に入手できるようになったことで、我々にとってますます身近な存在になりつつあり、それに伴ってロボットの潜在的な応用範囲はますます拡大している。しかしそれに反し、動的環境において人工知能による知的な動作制御を目指した研究はあまり行われていないのが現状である。人工知能による知的な制御システムを用いた制御の自動化や、自律的な行動によるロボットの独立化を図るためにも、自律エージェントを用いて制御を行うことは、これからのロボット制御に必要な手段であると言える。また、複数のエージェントの協力によらないと解決が難しい問題もあることから、複数ロボットを用いての制御を実現することは必要不可欠であると考える。

そこで我々は、ロボット、LEGO MINDSTORMS NXT を 2 台用いることによって、動的環境におけるロボット同士の協調動作に必要な制御システム構築の実現を目指した。本論文では、ロボット間の協調動作の中での基本動作として行われる探索、および探索続行について、その設計及び実装を記す。ただしここで言う探索続行とは、探索中にロボットがあるゴールを達成するために探索を中断し、その後再び探索を行う際の探索のことを示す (これについては 4.3 節で述べる)。また特に、協調動作の実験において判明したタイミングに関する問題と、その解決については 5.5 節で述べる。

2　関連研究との比較

本研究に関連している研究として、Developing Multi-Agent Lego Robotics [1] がある。これは本研究と同様、複数のロボット NXT を用いて協調作業を行い、設定した問題を解決するということを行っている。しかし我々の研究との相異点として、[1] ではロボットの移動手段として光センサを使用している点、また使用したロボットの機能が全て同じであるという設定で実験を行っているという点がある。

1 つ目の点について、光センサは表面の光の強度を測ることができるため、同じ強度の場所を選択して移動することができる。このことから、テープなどで望ましい場所に道を作っておけば、その道をたどって進むことが可能である。しかし、進行方向に物体が存在した場合、それらを回避するということは困難であるといえる。これに対して我々は、本実験において超音波センサを使用した。超音波センサは、物体との距離を値で計算することが可能である。またこの値は、物体との距離が近ければ近い程小さな値で表示される。これによって、移動中進行方向に壁が存在しても、それを壁と認識できる任意の数値に値を指定しておくことで、壁が存在することを認識することができ、衝突を回避するために的確に対処することができる。

光センサは道をテープなどで作る必要があるのに対し、超音波センサは既定値を設定しておくだけでロボット自身で進むことが可能な道を見つけ出し、フィールドを探索することができるため、実験の場を更に拡大して行う過程において超音波センサを使用する手段を用いたほうがより臨機応変な対応ができると考えられる。また 2 つ目の点については、我々は本研究において、機能の異なる 2 台のロボットを用いて協調を行わせている点が異なっている。本研究では、2 台のロボットが協力し合うことを目的としているため、あえて異なる特徴をもつロボットを選択して行った。

3　基本設計とプラットフォーム

3.1　BDI アーキテクチャ

BDI アーキテクチャとは、動的に変化する環境を知覚し、合理的に問題解決を行うためにプランを選択しながら動作する、熟考型エージェントの内部アーキテクチャである [2]。

BDI アーキテクチャは、エージェントの意図 (Believe)・願望 (Desire)・目的 (Intention) の三つの心的状況やプランライブラリ・イベントキュー、およびそれらを参照・更新するインタプリタ等で構成される。インタプリタは、環境知覚 (信念) と自らの目標から、プランライブラリを参照することによってそのときに実行すべきプランを選択し、これを実行するために適した意図を形成する。ここで形成された意図は、実行されるときが来るまで保持される。意図には一貫性や整合性が要請されるため、今持っている意図と矛盾する行動を取ったり、達成できないと信じていることを意図したりすることはない。また、もともとの目標を達成する必要がなくなったなどで、意図を持続する理由がなくなった場合はそれを破棄する。

3.2　Jason

本研究では、BDI エージェントの実装に Jason を使用した。Jason とは AgentSpeak の拡張版インタプリタであり、多くのユーザーがカスタマイズできる機能を備えたマルチエージェントシステム開発のためのプラットフォームを提供するものである [3]。

Jason は願望を達成するために信念ベースとプランライブラリを用いて意図を生成し、行動を行い、環境と相互作用することにより、新しい知覚を得る。これを繰り返すことによって、エージェントは願望を達成しようとする。この設計の基礎となるものが BDI アーキテクチャである。

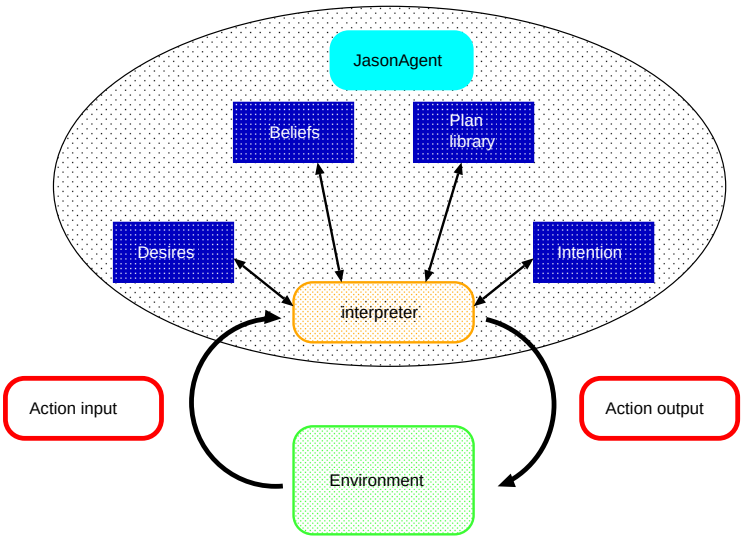

図 1: Jason と環境との相互作用

3.3 LEGO MINDSTORMS NXT

本研究において我々が使用したロボット、「LEGO MINDSTORMS NXT」は、レゴ社が開発した商用教育用ロボットである。

安価で比較的制御しやすいため、ロボットを実際に扱って実験することについては適したものであると言える。

3.4 Bluetooth

本実験では、ロボットが通信を行う手段として Bluetooth を使用した。

Bluetooth は数 m から数十 m 程度の距離の情報機器間で、電波を使い簡単な情報のやりとりを行うのに使用される。ノートパソコンや携帯電話、PDA などをケーブルを使わずに接続し、音声やデータをやりとりすることができるため、幅広く使用されている。

本研究では、一般的な家電量販店で販売されている、100m の通信が可能な Bluetooth アダプタを使用した。

実験開始当初は 1 つの Bluetooth アダプタで 2 台のロボットとの通信を行っていたが、ロボットの動きが芳しくなく、Bluetooth アダプタを 2 つ使用して再度実験を行ったところ、1 つのみの使用に比べ格段にロボットの動きがスムーズになったため、本実験では 2 つの Bluetooth アダプタを使用している。なお、本研究ではこの原因の追求は行っておらず、今後の課題の一つである (7 節)。

4　本研究におけるロボットの探索及び探索続行方法

本章では、本研究のうち著者が担当した部分であるロボットの探索、および探索続行の方法について述べる。

今回我々が採用した例題は、図 3 のような縦 4 ×横 5 マスのフィールド内を特性の異なる 2 台のエージェントが探索を行い、適切な台を見つけ、その上に物体を置くというものである。

探索範囲内はフィールド左上を座標 (0, 0) と固定し、xy 平面上の (0, 0) から (4, 3) までの格子点で表現する。フィールド内にはあらかじめ台を複数配置しておき、それらの中には物体を置くことができる台とできない台がある。本論文では以降、物体を置くことができる台を object、置けない台を obstacle として記述する。このとき、先に述べた object と obstacle とは高さの違いで区別している。

移動の際、ロボットは進みたい方向に回転し、一度の移動で 1 マス移動することができる。このときの角度と座標の変化は以下の表の通りである。

角度	移動方向	座標
0	-y	(X,Y-1)
90	+x	(X+1,Y)
180	+y	(X,Y+1)
270	-x	(X-1,Y)

図 2: 角度ごとの座標変化

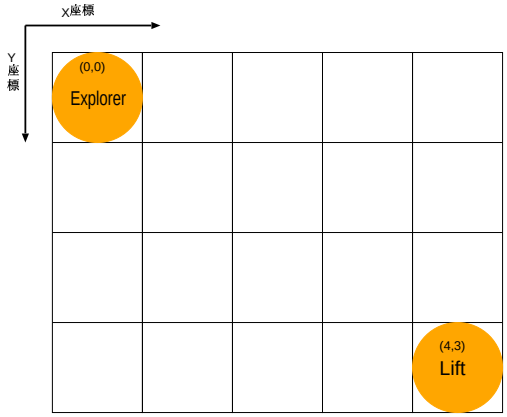

図 3: シミュレーションの図

またこのフィールドは、当初は当研究の先行研究でも使用されたプラスチック板で作成されたものを使用していたが、実験を進めていくうちにプラスチック板の劣化により板に出来た凹凸部分にロボットが車輪を取られ、ロボットの回転方向の誤差が大きくなった。そのため、途中からビニールシートに切替え再度実験を行ったところ、ロボットの動きがスムーズになったことから、フィールドはビニールシートを使用した。

4.1　ロボットの種類と機能

本実験での目的は先で述べたように、ロボットがそれぞれ探索を行うことで object を発見し、object の台の上に物体を置く、ということである。この問題を解決するため、実験では LEGO MINDSTORMS NXT を 2 台用意し、それぞれに異なる役割を持たせた。本節では、実験において使用するロボットの種類、およびその機能について述べる。

4.1.1　Explorer

2 台のうち 1 台は、主に探索の役割を持つロボットとした。これを Explorer と呼ぶ。(図 4)

それぞれロボット上部に超音波センサ、下部にタッチセンサを搭載しており、この 2 つのセンサを使ってフィールド内を探索する。このとき、Explorer は超音波センサの位置が比較的高いため、高低差を認識することができる。

本研究では、この特徴を obstacle と object の識別に活用した。

obstacle および object と Explorer との高さ関係は 図 5 に示す。図からわかるように、Explorer の持つ超音波センサの位置と 2 つの台とを比較すると、obstacle はセンサ前方に存在していることからセンサで知覚できるのに対し、object はセンサの下位置にあり、知覚できない。この関係から、Explorer は 2 つの台を識別することが可能となる。

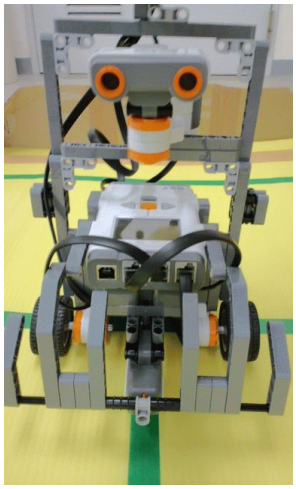

図 4: Explorer

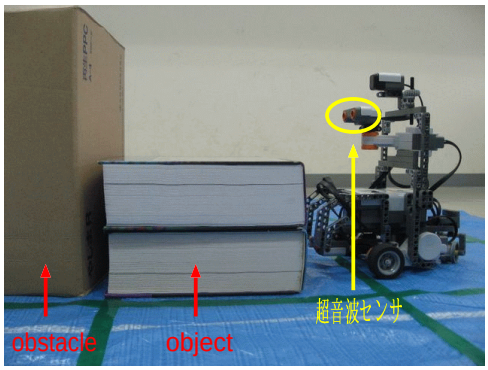

図 5: Explorer と台の高さ関係

4.1.2 Forklift

もう 1 台のロボットには、目的の場所へ物体を運んでいき、置く役割を持たせた。これを ForkLift と呼ぶ。(図 6)

このロボットは、Explorer と同様に超音波センサが搭載されている他、リフト機能を持っているため、物体を持ち上げて置くことができる。実験では物体を object の台の上に置く重要な役割を果たす。

このとき、ForkLift の超音波センサは低い位置に取り付けられているため、高低差を識別することはできない (図 7 参照)。このことから、何か探索中に台を発見しても、Explorer に一度その台が物体の置ける台かどうかを判断してもらう必要がある。

またこれ以降、本論文における説明および図では、Forklift を Lift と省略して表現するものとする。

4.2 探索

探索 (以後、search) は、この研究においてロボットが一番最初に行う重要な作業である。本研究の目的を達成するための課題として設定した問題を解決するためには、search は必要不可欠である。

本節では、本研究で作成した探索プログラムの性能について述べる。

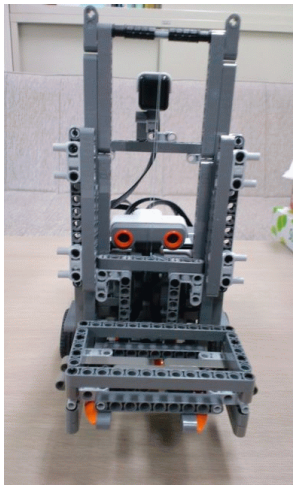

図 6: Lift

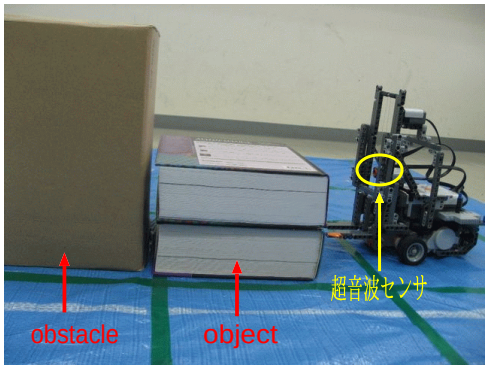

図 7: Lift と台の高さ関係

4.2.1 探索プログラムの機能について

探索は、本実験で使用した 2 台のロボットのどちらもが使用する機能であり、基本的な動作は同じであるが、ロボットの機能がそれぞれ異なることから、ロボットの持つ特色を生かすため Explorer、Lift それぞれ探索方法は異なる形で作成した。

- **探索の基本動作**

フィールド上を探索する際、知覚した座標 ((X, Y) とする) は “look(X, Y)” として信念に追加しながら探索を進める。このようにしておくことで、知覚済みの座標は信念に “look(X, Y)” という形で記憶される。また進みたいマスが既に知覚されているマスなのかどうかは、前に進む前に進みたいマスの座標の look が信念に存在するかどうかを確認することで知ることができる。

探索時は常時この動作を繰り返すことによって、知覚済みのマスの探索が重複して行われることを防ぐことができる。

このようなことを踏まえて、下記に探索の基本動作の手順を記述する。

1. 進みたいマス (座標 (X, Y) であるとする) の look が信念ベースにあるかどうか調べる
2. look がなければ進みたいマスを知覚する (以下、percept)
   look が存在していれば 5 へ進む
3. 知覚したマスを look(X, Y) として信念に追加する
4. look(X,Y) をもう一台のロボットへ送る
5. マス (X,Y) に進む

なおこの基本動作とは、進みたいマスに台等の障害物が何も存在しないときの動作を示す。また、Explorer が obstacle を発見した、もしくは Lift が発見した台が obstacle であった際は、obstacle(X, Y)((X, Y) には obstacle が存在する座標が入る) として 2 台のロボットそれぞれの信念に追加される。

- **Explorer の探索**

Explorer は台の識別を行うことができ、台の識別方法については 4.1.1 節で示した通り、超音波センサで知覚できるかどうかで判断している。しかし超音波センサ方向で知覚できなかった時は、そこに object が存在する可能性の他に、そこに何も存在しないという可能性もあるため、1 マスずつ進んで確認する必要がある。object が存在した場合、Explorer の正面部分に取り付けられているタッチセンサによって Explorer が object に衝突した衝撃を認識し、前方に何か台が存在することを知る。このとき、この状態で発見した台は超音波センサでは認識できていないものであることがわかっているので、本研究の設定から考えて object を発見したと断定してよい。これらの特徴を考慮すると、図 8 のような探索方法が最適な方法であると考えられる。図のようにジグザグに進むことにより、まずは超音波センサで進みたいマスを percept し、そのマスに移動してタッチセンサによって台の有無を確認することができる。

このように探索を進めていく上で、進みたいマス (座標 (X, Y) であるとする) を percept した際に obstacle を発見した場合、Explorer は自分の信念に obstacle(X,Y) を追加し、それと同時に Lift にも同様の情報を送る。object を発見した場合も同様である。ただし object を発見した時点で、Explorer は object の位置を Lift に送ると同時に Lift に物体を object の位置に置いてもらうように依頼しなければならない。それらの方法等については [4] で説明する。

また obstacle を発見した際、Explorer は次のマスを探索するために obstacle を回避しながら進む必要がある。obstacle の回避方法は、図 9〜11 に示す。それぞれ、進行方向に台が存在した場合、存在する台が object であれば Lift に物体を置くように依頼し、obstacle であれば情報を信念に追加した上でそのマスを避け、まだ進んでいない他のマスに進路転換を行う。

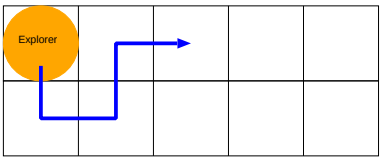

図 8: Explorer の基本探索

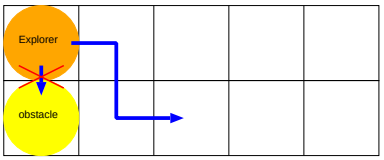

図 9: 進行方向に obstacle が存在する時 (i)

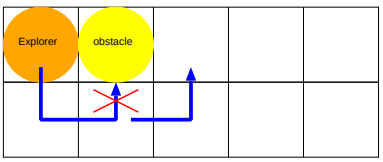

図 10: 進行方向に obstacle が存在する時 (ii)

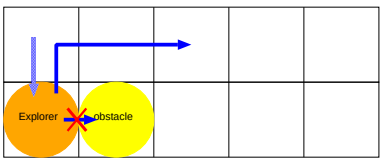

図 11: 進行方向に obstacle が存在する時 (iii)

- **Lift の探索**

Lift は、obstacle と object の区別をつけることができないが、超音波センサによって台の存在を知ることは可能である。従って、Lift の基本探索の方法は図 12 のような方法が最適であると考えられる。このとき、図の赤い矢印は percept を意味する。

Lift は台の識別ができないため、Explorer のように毎回知覚したマスに進むというような動作はせず、その場で 0 度方向、および 270 度方向を知覚しながら、270 度方向へのみ進む。そのため、Lift の探索方法は Explorer よりも比較的単純であるといえる。

次に主な手順を示す。

1. 現在値を座標 (X,Y) とするとき、座標 (X,Y-1) について、信念の中に look が存在するかを調べる
2. look が存在しなければ 0 度方向に回転し、座標 (X, Y-1) を percept
   look が存在すれば 4 へ
3. look(X, Y-1) を信念に追加し、Explorer にも情報を送る
4. 270 度方向に回転し、座標 (X-1, Y) について 1 と同様に look の有無を調べる
5. look が存在しなければ percept
   look が存在すれば 6 へ
6. 座標 (X-1, Y) へ進む (以後、go)

また、探索中に Lift が台を発見した際、それが obstacle か object かを特定するため Explorer に調べてもらう (以後、check) 必要があるので、Explorer を呼び、Explorer からの返答が返ってくるまでその場で待機する。なお、check 動作については [4] で述べる。もしも 5 の過程で発見した台が obstacle であればそれを回避して進む必要があるため、図 13 のように obstacle を回避して進む。

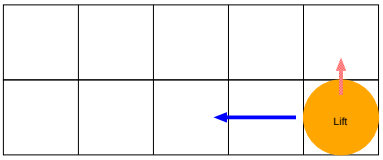

図 12: Lift の基本探索

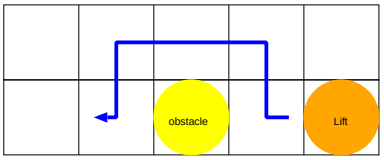

図 13: 進行方向に obstacle が存在する時

4.3 探索続行

ここで言う探索続行とは、Lift が発見した台を Explorer が check した結果、obstacle であった際に再び search を再開するときの動作を言う。以後、探索続行は research として表す。

check が呼ばれると Explorer、Lift 共にその時点での現在値を信念に追加し (以後、mark)、覚えておく。check の結果、その台が obstacle であった時、Explorer がその判定結果を Lift に送り、それと同時に research をするよう Lift に指示する。research を行う際、Explorer、Lift 共に再び探索を開始するが、そのときの現在位置が check 開始時に信念に追加した座標と異なる可能性があるため、research が行われるとまず予め記憶していた座標に戻った (以後、return) 後、探索を再開する。

以上のことを踏まえて、探索続行の動作の流れを図 14 に示す。

(1) Explorer の探索中に Lift が台を発見し、Explorer に判定依頼が送られる。
このとき、ロボットはそれぞれその時点での自分の現在値を “mark(X, Y)” として信念に記録しておく。

(2) Explorer が判定をするために依頼された場所まで移動する。この場合の台の判定結果は obstacle であるので、Explorer はその情報を Lift に送り、自分の信念にも追加する

(3) 探索を再び再開する。このとき、(1) の段階で予め元いた現在値を mark しているため、ロボットはそれぞれ mark された位置まで戻る。

(4) mark された位置まで戻り次第、ロボットは探索の続きを行う。

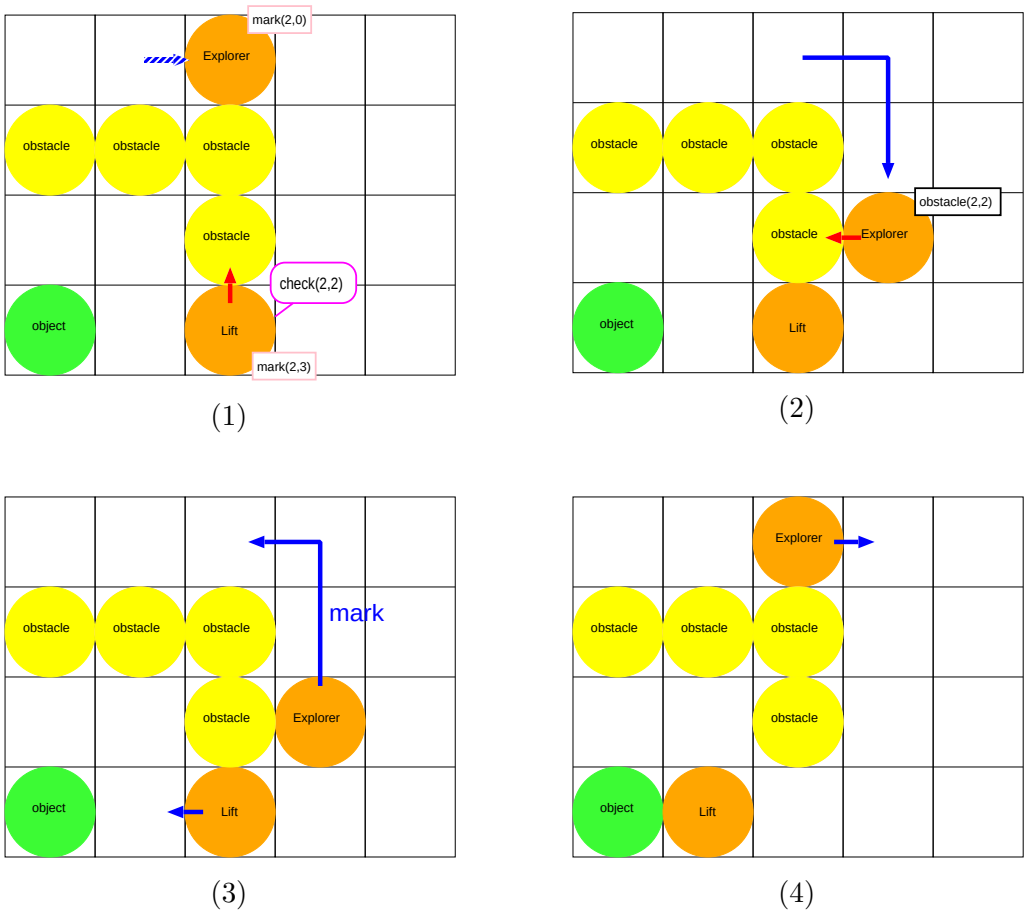

図 14: 探索続行の流れの模式図

5 実験

我々はこの研究において、本論文で紹介した探索及び探索続行のプログラムを実際にロボットに搭載し、動作の実験を行った。

本節ではその際の条件、及び結果について述べる。

5.1 協調動作の概要

この実験は、2 台のロボットを用いて協調動作を行わせていることを目的としている。実験において我々が設定した問題は、**「探索を実施することによって目的に適した台を見つけ出し、その位置に正しく物体を置くこと」**としている。ただしフィールド上に配置した台は複数存在するとし、それらの台のうちどの台に物体を置くことができるのかについては、ロボットが実際にフィールド上を探索し、見つけ出さなければならない。ロボットはそれぞれ Bluetooth によって互いに通信を取り合うことが可能であり、通信によって自分の位置を示すことや、新たなゴールを依頼することも可能である。

この研究のテーマである複数のロボットによる協調作業の実現ということを考慮し、実験における問題設定は、2 台のロボットが協力することによって初めて解決することができるようなものとした。

5.2 初期設定

協調動作の実験にあたり、いくつかの事項を初期設定として定めた。設定内容を以下に示す。

- **各エージェントの初期位置**

  Explorer 及び Lift の初期位置は
  Explorer:(0, 0)　ForkLift:(4, 3)
  とする。ロボットは初期位置から互いに探索を開始する。

- **Explorer と Lift の探索範囲**

  実験では 2 台のエージェントを用いて探索を効率的に行わせるため、探索範囲を 2 つに分け、それぞれロボットに範囲を指定した。範囲の区分については図 15 に示す。このとき、緑の線内が Explorer の探索範囲、ピンク色の線内が Lift の探索範囲とする。

  2 台のエージェントは探索中それぞれに与えられた範囲内のみを探索し、相手の範囲には入って行かない (check の場合を除く)。

- **探索範囲内に設置する台の数、及び位置**

  この実験において範囲内に存在する台は、object: 1 つ obstacle: 4 つの合計 5 つとして固定した。これらを範囲内に配置する。このとき、obstacle については

  - Explorer および Lift のスタート位置
  - object の配置場所
  - ロボットの行く手が遮られると考えられる位置

  を除き、様々な形に配置可能であるとする。これに対し、object については本研究において配置パターンをある程度絞るため、座標 (0, 3) として位置を固定した。

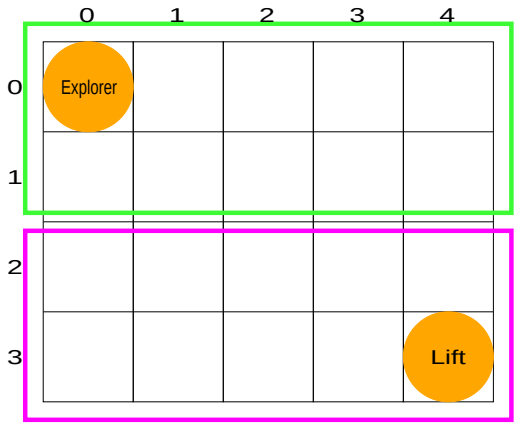

図 15: Explorer と Lift の探索範囲の区分

- **ロボットにはそれぞれ、台の位置情報は与えない**
  実験にあたり、ロボットにはあらかじめ台の位置情報は与えず、それぞれロボットが探索の中で自ら追加していくものとする。その際、台が object と obstacle とのどちらか、という情報は、常にロボット同士が通信し合い、共有しているものとする。

5.3 実験における制約条件

本研究で探索プログラムを作成するにあたり、いくつかの制約条件を指定した。制約条件の内容は以下の通りである。

- **各ロボットは 2 行分の探索が可能であるとする**
  探索プログラムは、2 行分の探索が可能なプログラムの作成を目指した。このときプログラムとしては、列の拡張は可能であるとして作成した。しかし 5.2 節で示したように、本実験では探索範囲を区分する (図 15) と設定したため、2 行分の探索が可能なプログラムの作成を目指した。このとき Explorer は (0, 0) スタート、Lift は (4, 3) スタートとするため (5.2 節参照)、ロボットそれぞれのスタート位置の Y 座標を Y とおくと、Explorer は Y〜$Y+1$ 行の 2 行分、Lift は Y〜$Y-1$ 行の 2 行分の探索が可能であるとする。

- **ロボット同士の探索範囲が重複するような配置方法は考慮しない**
  上記で述べたように、各ロボットは 2 行分の探索が可能であるが、2 台のロボットが同じ範囲の探索を行うことは効率が悪いことから、あらかじめ探索範囲が重複するとわかっているような配置方法は考えないものとする。

- **探索範囲が塞がれた場合の回避方法は考慮しない**
  探索範囲が塞がれた形の例を図 16 に示す。
  本研究において作成したプログラムでは、図のように各ロボットそれぞれが指定された探索範囲において、行く手を obstacle で遮られてしまっているということが進んで

いった結果わかった場合、それ以上の探索は不可能であることから、ロボットはスタート時の位置に戻るようにプログラムした。

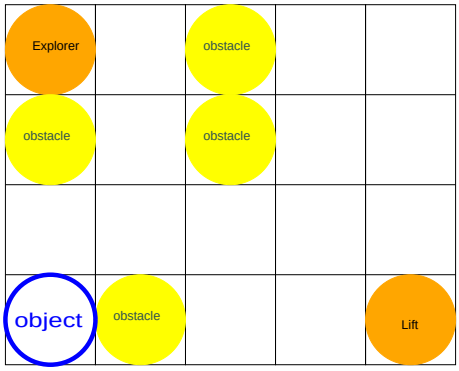

図 16: Explorer の探索範囲が塞がれた場合の例

5.4 実験結果

5.3 節の条件の元で実世界において実験を行った結果、本研究で指定したフィールド上での探索において、ロボットはフィールド上を着実に探索し、object を発見することに成功した。また、知覚したマスを随時 look(X, Y) として信念に残しておくようにすることで、進みたいマスが知覚済みであるかどうかを、信念中の look の有無を調べることによって簡単に確かめることができる。こうすることにより、ロボットは自分で知覚すべきかどうかを判断し、知覚済みのマスを重複して知覚することなく、必要最低限の動作で目的を達成することができた。

しかし実験を繰り返していくうち、フィールド上に置かれた obstacle の配置場所によってはうまく動作が行われないことがあるという問題が生じた。この問題とその解決策について、5.5 節で述べる。

5.5 問題と解決方法

実験で生じた問題を、以下に示す。

Explorer の探索中に Lift から check がきた場合、Explorer はそれまで行っていた search を取り消し、check に切替える。その後、依頼された台を判別するため、その場所までの経路選択を行う。しかしタイミング次第では、search の動作の一環として go が実行されている途中で check が呼ばれ、go が実行されている間に経路選択が始まってしまう。(図 17)

ロボットは常に現在値がどこかという座標情報を更新しており、座標の更新は go の実行が完了した後に行われている。このため、go の実行中に現在値から check の位置までの経路選択を行うと、現在値が更新されないままにスタート位置が指定されてしまう。よって目的地までの経路がずれ、結果として目的地にたどり着くことができない。

この問題点を解決するため、check が依頼されてから経路選択の実行を行うまでに待機時間を設けることにした。この問題の原因は、search は取り消されても実行途中である go は継続して実行され、その動作が完了する前に経路選択が実行されることにある。よって、経路選択を go が完了した後に実行できるように改良すれば、解決するはずである。

待機させるための方法としては、Jason の internal action である “wait(A)” を用いた。A には待機させたい時間を入力し、A は 1/1000 秒として計算される。例えば、一秒待機させたければ A=1000 として呼べば良い。

これを経路選択を行う前に実行し、go の実行後まで待機させるように改良した。

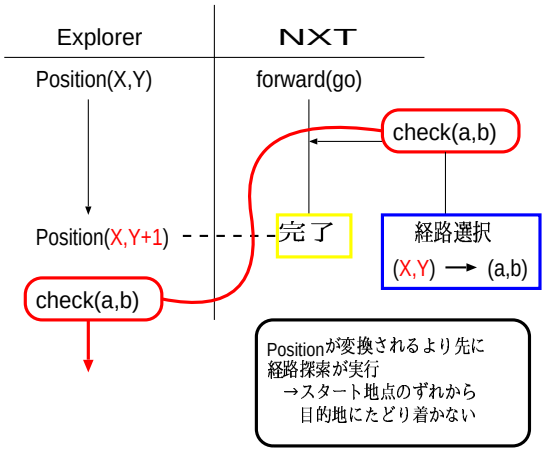

図 17: 問題発生の模式図

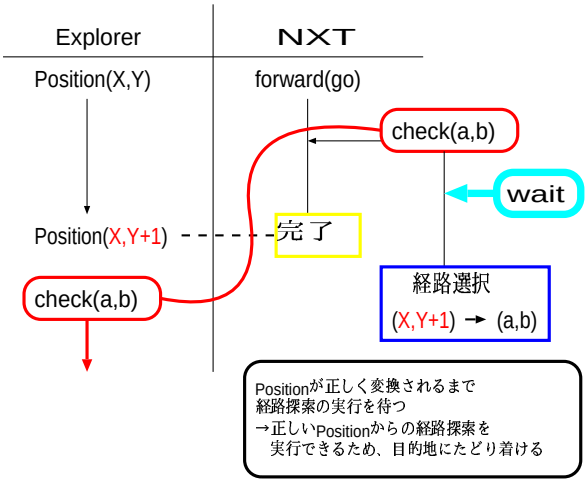

図 18: 問題解決方法の模式図

5.6 改良後の実験結果

プログラムを改良した後再び実験を行った結果、待機時間によって成功する obstacle の配置パターンと失敗する配置パターンがあるということがわかった。また、ロボットの移動には誤差を伴うため、ロボットの充電状態によって動きの善し悪しにも影響が出てくる。このことから、実験を進めていくうち初期段階では成功していた待機秒数でも、何度も実験を繰り返していくと後半で失敗する、というような状況に陥った。これらを考慮した上で実験を繰り返し、適当な秒数を計測した結果、現時点で 8 秒が失敗の起こらない最小の待機時間であることがわかった。よって、プログラムでは 8 秒待機するように指定している。このように wait を追加することによって問題点は解決され、Explorer はほぼ確実に check を依頼された位置までたどり着くことができるようになった。(図 18)

5.7 考察

実験を行った結果、シミュレーションでは問題なく成功したからといって、実世界でも同じように成功するとは限らないということがわかった。シミュレーションとは違い、実世界ではロボットの充電状態や実験している環境によってロボットの動きが大きく変わる。本実験で見つかった問題についても、タイミングによっては待機時間を設けなくても問題なく成功する。

これらのことから、実世界でのロボット制御を行う際は実世界での実験を繰り返し行い、どのような条件下でも動作がうまく実行できているかどうかを確かめる必要があるといえる。また、動作の設計段階でタイミングの影響を受けにくいような頑強な設計にすることも有効であることから、今後実世界においてロボット同士の協調動作を目標とするプログラ

ムプログラムを作成する際には、予めタイミングの重要性を考慮した設計を考えておくと、後々の実験段階において、大きな問題が起きることを軽減することができると思われる。

6　まとめ

本研究では、実世界においてロボットを 2 台用いて行う協調動作を実現することを目標においてプログラムを作成した結果、ロボットは本研究で指定した範囲内での動作制御に成功した。その際、シミュレーションでの動作と実世界の動作との差異には注意が必要であるということがわかった。シミュレーションとは違い、実世界では実験環境やロボットの充電状態などの関係でスピードが一定ではないこと、またロボットの回転移動の制御がまだ完全ではないため、回転不足が起きるとロボットが指定された通り正しい方向に回転できず、その結果そのマスには存在しない物体を誤って認識してしまうことなど、予想外なことが多々起きる。このことから、実世界での実験は非常に重要であるといえる。

7　将来の展望

本研究の将来の展望については、以下の 4 つが考えられる。

1. フィールドの拡張、および改良
2. Bluetooth の使用方法の検討
3. 5.6 節で固定指定した待機時間の改良
4. ロボットの回転動作の制御

1 について、本研究では探索範囲を縦 4 マス×横 5 マスに指定したフィールド内でのロボット制御に成功した。しかし、探索範囲を更に拡大してロボット制御を目指す場合、本研究では考慮しなかった、2 つ以上の obstacle が縦に並んだ際 (図 16 参照) の回避プログラムの追加は重要であると考えられる。また、この問題を解決するにあたっては、ロボット同士の衝突回避も同時に関わってくる問題であることから、[4] で述べられる衝突回避の方法との関連を考えなければならない。しかし、この問題の解決方法を考える場合、本研究で指定した 4 × 5 マスの範囲内では obstacle の配置方法に限界があり、様々な配置条件で実験を行うことが非常に難しい。従って今後、今よりも更にフィールドを広げた場合での探索方法の改良、及び obstacle の回避方法について考えたいと思う。

また 2 について、本実験ではロボット同士の通信手段として Bluetooth を使用したが、Bluetooth アダプタを 1 つ使用するよりも 2 つ使用したほうがよりロボットの動きがスムーズになったため、今回は 2 つの Bluetooth アダプタを使用した。しかし、その原因については検討していない。今後ロボットの台数を増やした際に、ロボットと同数の Bluetooth アダプタを用意するのでは効率が悪いため、今後 Bluetooth の使用方法について検討する必要があると考える。

3 について、現時点では待機時間を 8 秒として固定指定している。しかしこれはあくまでロボットの移動中に台の判定依頼がきた際に生じる問題であり、タイミング次第では待つ必要がない状況も存在する。このことから、移動中に呼ばれた時にのみ一定時間待機する、というような場合分けが可能であれば、よりロボットの動作効率の向上を望むことができると

考えられる。待機時間についても、移動した時点で wait を止めることができればより効率的であると言える。

最後に 4 についてであるが、これはロボットにとって非常に致命的な問題であるといえる。しかし、これはロボット本体の性能にも大きく関わってくることであるので、ある程度の限界が存在することは否めない。しかし本実験において、ロボット本体の重さが回転度合やスピードに大きく影響し、本体が重ければ重いほどロボットの動き自体が鈍くなることや、左右の重さが異なることでロボットは正しい方向に回転することができないということがわかった。従って、もう少しロボット本体の重さを軽減すること等、ロボットの重さを考慮することによって今以上のロボットの制御を望むことができるのではないかと考える。

よって、今後はこれらの問題について取り組みたいと考える。

8 謝辞

本論文の執筆及び研究にあたり、とても親身に御指導してくださった指導教官の新出尚之准教授に深く感謝し、厚く御礼申し上げます。また、普段から丁寧かつ的確なアドバイスをくださっている藤田恵先輩、そして鴨浩靖准教授、更に新出、鴨研究室のみなさまに、感謝の意を表します。

本当に、ありがとうございました。

参考文献

[1] Anders Lemke, Johan Laidlaw, and Lars Zilmer- Pedersen. Developing multi-agent lego robotics. 2007.

[2] Munindar P. Singh, Anand S. Rao, and Michael P. Georgeff. Formal method in DAI. Multiagent systems: amodern approach to distributed artificial intelligence, 1999.

[3] Rafael H. Bordini, Jomi Fred Hubner, Michael Wooldridge. *Programing multi-agent systems in AgentSpeak using Jason*. WILEY, 2007.

[4] 隅田麻由. 「BDI エージェントを用いた実世界での協調作業に伴う衝突回避について」, 奈良女子大学理学部情報科学科 2010 年度卒業論文, 2011.