2016 年度 卒業論文

自律ロボットの目的達成における行動の柔軟化について

奈良女子大学 理学部 情報科学科 ４回生 新出研究室
13251401 石井あすか

平成２９年２月

目 次

概 要

近年、動的に変化する環境下においても、自律的に目的を達成するロボットの実現が望まれている。このようなロボットの実現を目指し、先行研究において ROS とカメラを用いた目標物の認識、到達行動の実装が行われた。しかし、実世界では動的な環境変化による障害の発生や知覚の誤りなどにより、目的達成までにプランが失敗することが起こり得るが、これまでには失敗に対する考慮は行われていなかった。そこで我々は、目的達成までに起こる環境の変化や様々な障害を想定し、失敗の概念の取得や環境に適した失敗後の行動を選択する自律ロボットの実装を行った。本論文では失敗を認識した後の、回復行動の実現について述べる。

1 はじめに

1.1 研究背景

近年自律ロボットの普及が進んでいる。このようなロボットがもつべき能力として「目標物に到達する」というような目的達成行動が挙げられる。そこで、先行研究では ROS やカメラを用いた目標物へ到達行動をとるロボットが実現された [1, 2]。

しかしロボットを取り巻く環境は常に一定ではなく、またセンサによる知覚やモータによる移動も正確ではないため、目標物が見つからない場合や遠方にあって直接には到達できない場合なども考えられる。そのような場合においても自律ロボットが目的を達成するために、起きた事柄に応じて適切な行動プランを選択することが望まれる。その一環として、実世界で行為に失敗した際に状況に適した回復行動をとることが求められる。

そこでロボットに行為が失敗したことを認識させその後の適切な行動選択を実現するため、我々は目標物到達における行為の失敗の概念を導入し、失敗の認識を効率よく行わせ、かつ環境に応じた失敗後の回復処理を行うロボットの実装を行った。なお本研究は、奈良女子大学理学部 4 回生小谷、山本との共同研究である。失敗の概念の取得については [3]、失敗認識の効率化については [4] で述べられている。本論文では失敗後の回復行動について述べる。

2 Jason と ROS

本節では我々のロボットの実現に用いた技術について述べるため、Jason と ROS について述べる。

2.1 Jason

2.1.1 概要

Jason とは BDI エージェントの構築プラットフォームであり [5]、トリガリングイベント、コンテキスト、ボディで構成されるプランを用いるため、様々な条件を組み合わせることでエージェントは状況に応じたプランを選択して実行することができる。また、プランが失敗したのちに選択されるプランを作成することも可能であるため、その失敗に対する回復処理を設定することができる。我々はこれを用いて、ロボットの目標物への到達行動をとるプランが失敗した際に、あらかじめ追加したプランが選択されることで他の目的達成方法へ変更するような失敗の回復行動を実現した。

2.1.2 プラン

Jason で動作するエージェントプログラム内のプランは以下のようになっている。上記のように、プランはトリガリングイベント、コンテキスト、ボディからなる。

```
+!arrive_at(X): connectable(1)
<-              .print("Arriving at ", X);
                tcp_write(arrive);

                      ⋮
```

このプランのトリガリングイベントは`+!arrive_at(X)` となり、`connectable(1)` がコンテキスト、`<-` 以下がボディと呼ばれている。ゴール `!arrive_at(X)` が新たに生まれたというイベント `+!arrive_at(X)` が発生し、そのイベントにマッチするトリガリングイベントを持つプランがある。そのプランのコンテキストにある条件が満たされている場合に、そのプランによってそのゴールが達成しようとされ、ボディの中身を上から順に実行していく。この実行がすべて成功したとき、このプランが成功したと言え、実行が 1 つでも失敗した場合にはプランが失敗したこととなる。

また`+!arrive_at(X)` のプランが失敗した際に実行したいプランはトリガリングイベントを

```
-!arrive_at(X)
```

とすることにより作成できる。

2.1.3 実行画面

Jason で構築したエージェントの実行中はコンソール画面のウィンドウが生成され、エージェントからの出力を表示することができる。以下はコンソール画面の例であり、本論文で述べるロボットのエージェントが動作中の出力の例である。このエージェントは、動作状況の把握のため、下記のそれぞれの時点で以下のようなメッセージをコンソールに出力するようになっている。

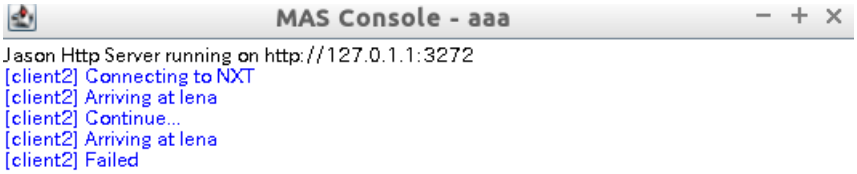

図 1: 実行画面

<Connecting to NXT>
NXT（ロボット）との接続。行動開始。
<Arriving at X>
X は目標物名であり、X へ到達しようと探索または到達行動を行っている。
<Continue... >
到達行動が未完了であるという旨の通信がロボットから返り、目標物の探索または到達行動を続ける。
<Failed>
ロボットが現在の目標物に到達することが出来ず、目的達成行動の失敗。

2.2 ROS

我々はロボットの制御に ROS(Robot Operating System) を用いた。本節では ROS について述べる。

2.2.1 概要

ROS とはロボット用ソフトウェアフレームワークであり、世界中で広く使用されている。ROS でプログラムを動かすためのパッケージがあり、その内部にノードという実行ファイルが必要となる。ノードを組み合わせてシステムをつくることにより一部分の修正の際でもすべての修正を行うことなく、そのノード内の修正のみで終えることができる利点がある [6]。

2.2.2 通信方法

ROS の通信方法としてトピックによるノード間の通信がある。このトピックを用いた通信では一方向の通信となる。また他の通信方法として、サービスを介するノード間の通信が存在する。この通信はトピックを用いた通信とは異なり、ノード間の双方向の通信を行う。そのため通信によって受け取った情報に対して処理を行い、その処理に応じた結果を返すことが可能である[2]。

本研究ではノードによる通信、サービスによる通信のどちらも用いているが、本論文で述べるノード間の通信は処理に対する値を受け取る必要があるため、サービスを介する双方向の通信が用いられている。

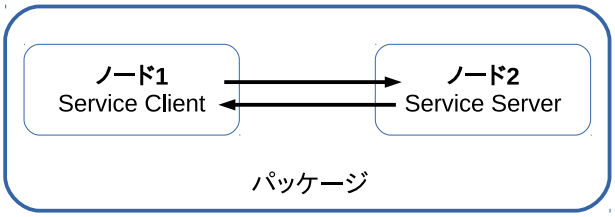

図 2: ノード間のサービス通信

3　実装

3.1　実装方針

ロボットが目標物への到達行動に失敗した際、目標物探索時の状況に応じて行動選択をさせることとした。本研究では、目標物を認識したが到達に失敗した場合と、目標物を認識せず到達に失敗した場合の 2 通りの場合を考慮し、それぞれについて別々の回復行動をとらせる。

初めに、目標物を認識したが到達に失敗した場合、その地点で一定時間待機したのちに再探索を行うよう実装した。この場合は目標物を認識後、ロボットと目標物の間になんらかの障害物が発生したことが失敗の原因として考えられるが、一定時間後にはその障害物が取り除かれる、または移動、消滅することも可能性として考えられる。それを期待してこのような回復行動をとることとした。なお、障害物が消失しない場合は、その障害物を避けて進行する回避行動が必要なことも考えられるが、今回はこれについては考慮していない。

次に目標物を認識せずに到達に失敗した場合、別地点へ移動し再探索を行うよう実装した。これは最初に探索を行った地点付近に目標物が存在する可能性は低いと考えられるからである。

また、以上のような再探索を 2 回行っても現在の目標物に到達できない場合は、次の目標物へと目標を変更し探索を開始することとした。これは、人間でも中間目的地にたどり着けない場合にその中間目標を放棄して次の目標を目指すようプランを変更するのと同様の考えによる。

3.2　構成

ロボットの制御は ROS を用いて機能別に分けて実装されており、Jason と ROS を仲介する tcp ノード、物体を検出する haar_like ノード、ロボットと通信を行う nxt ノードの 3 つのノードからなる。これらは従来研究で実装されたものである。3 つのノードはそれぞれ互いに通信しているが、本論文で述べる周囲の状況に応じた回復行動を実装する際に変更を加えたのは nxt ノード、tcp ノードであり、物体検出に関しては変更を行っていないため、本論文では haar_like ノードの説明は省略するものとする。これらと Jason での制御を行う client エージェントは、図 3 のように繋がっている。nxt ノードは ROS サービスサーバ、tcp ノードは ROS サービスクライアントとして実装され、これらの間は ROS サービスを介した通信を行う。client エージェントと tcp ノードは TCP/IP で通信する。

また、ロボットが各行動をとるまでの流れは次のようになっている。

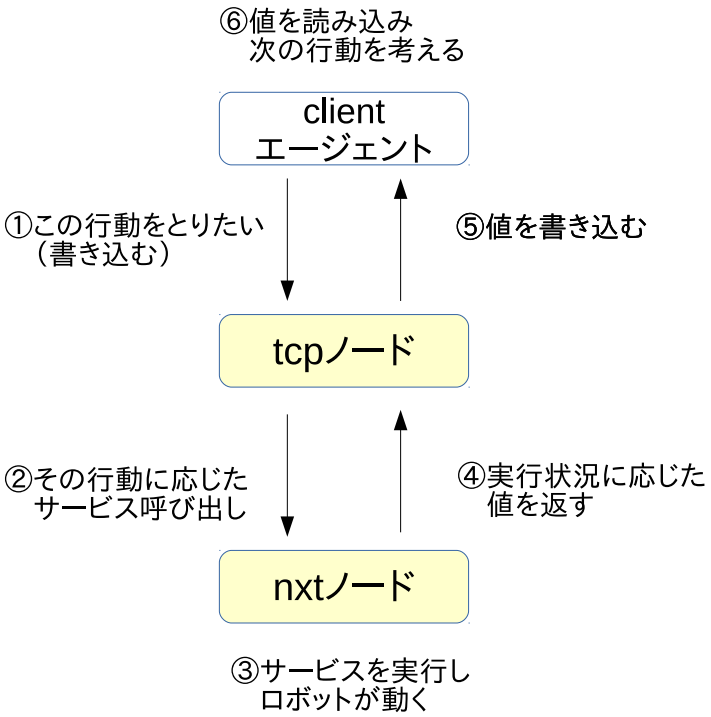

図 3: 通信状況

① client エージェント内で決めた、次にとりたい行動を tcp ノードへ書き込む。
② tcp ノードはそれを読み込み、その行動に応じた nxt ノード内の ROS サービスを呼び出す。
③ nxt ノード内ではそのサービスを実行し、ロボットが動く。
④ 実行状況に応じて 0 または 1 などの値を tcp ノードへ返す。
⑤ tcp ノードはその値を client エージェントへ書き込む。
⑥ client エージェントではその値を読み、状況にあった次にとるべき行動を考える。

この一連の流れを繰り返すことでロボットは通信、探索、到達、再探索などの行動をとる。

3.2.1 client エージェント

client エージェントは Jason で制御を行っており、ロボットの行動プランはこのエージェント内に組み込まれている。従来はロボットと通信するプラン、目標物を発見し到達しようとするプランで構成されていたが、目標物への到達失敗後に再び探索を開始するよう実装した。その際に追加したプランは、select プラン、 move_to プラン、wait プランである。

- select ゴール
  ロボットが目標物への到達に失敗した際の周囲の状況から、次にどのような回復処理行動をとるのかを決定し、実行に移すという目標。このゴールに対するプランは 1 つあって、これは到達に失敗すると実行されるプランであり、そのプランの本体内で周囲の状況を判断した nxt ノードから 0 または 1 の値を受け取って、その値に応じて以下のサブゴールを達成する。
  - 0 (探索時に目標物を認識せず到達に失敗) — 別地点への移動を行う move_to プランへ
  - 1 (探索時に目標物を認識したが到達に失敗) — 一定時間その場で待機する wait プランへ
- move_to ゴール
  再探索のため、ロボットが別地点へ移動するという目標。ある角度だけロボットが右回転を行い、向いた方向へ前進を繰り返すことで移動する。本研究では回転角度を引数で与えており、現在は 90 度に設定している。select ゴールと同様、移動を行って状況を判断した nxt ノードから 0 または 1 の値を受け取って、その値に応じて以下のサブゴールを達成する。
  - 0 (移動中) — 移動未完了のため、move_to プラン再帰
  - 1 (移動完了) — 別地点へ移動が完了したため、再探索開始
- wait ゴール
  ロボットがその地点で一定時間待機するという目標。待機後、再探索を行う。このゴールを達成するため、本体内で以下の内部アクションを用いた。また、本研究では待機時間を 15 秒に設定している。
  - .wait(15000) — 15 秒待機

また、再探索を 2 回行っても現在の目標物に到達できない場合に次の目標物へと変更するようにした。これについては、あらかじめ再探索時に回数をカウントし、その回数に応じて再探索を行うか目標物を変更するかを判断するよう実装した。

3.2.2 nxt ノード

実際にロボットと繋がっており、このノード内の Python 言語で書かれた関数によってロボットが動いている。本研究では、探索時の周囲の状況を判断する select 関数と、別地点へ移動する move_to 関数を新たに作成し、それらをそれぞれ Select と MoveTo という名の ROS サービスとして提供している。

- select 関数
  探索時に目標物を認識したか否かを判断し、それに応じて値を返す関数。変数 exist をおいて初期値を 0 とする。目標物を探索するための search 関数内において、目標物を認識したときに exist を 1 とし、これにより認識の有無を判断する。
  - exist = 0 のとき (目標物を認識しなかった)— 0 を返す
  - exist = 1 のとき (目標物を認識した)— 1 を返す

- move_to 関数
  別地点へ移動する関数。実行される回数によってロボットの動きは異なる。この関数で 0 を返すと client エージェントの move_to プランが再帰され、繰り返し MovoTo サービスが呼ばれることとなり、10 回目に 1 が返って繰り返しをやめることで 10 回の前進を行っている。そのため初期値が 0 である変数 count をおいて、1 回前進する度に値を 1 つずつ増やし、回数を判断している。また、count の値が 0 のときのみロボットは右回転を行い、移動する方向を定める。これによってロボットの別地点への移動を実現している。
  - count = 0 のとき — 指定された角度だけ右回転後 1 回前進、0 を返す
  - 0 < count < 10 のとき — 1 回前進後、0 を返す
  - count = 10 のとき — 1 を返す

3.2.3 tcp ノード

tcp ノードは Jason と ROS の仲介を行っている。client エージェントからとりたい動きを書き込まれた際に、それに合った nxt ノードの適切なサービスを呼び出す働きをもつ。また、呼び出したサービスから返ってきた値を client エージェントに書き込む役割もある。

以下は tcp ノード内での実装である。reqSelect() は Select サービスの、reqMoveTo() は MoveTo サービスのサービスハンドラであり、これらの関数を呼ぶことで対応するサービスが呼び出される。それらの関数の返り値はサービスからのレスポンスで、サービスからの返り値は reqSelect() の返り値の selectness メンバ、および reqMovoTo() の返り値の movement メンバから取り出せる。それらを string 型に変換し、client エージェントへ書き込んでいる。

select 関数

```
def select():
    res = reqSelect()
    ch.writeLine(str(res.selectness))
```

move_to 関数

```
def move_to(direction):
    res = reqMoveTo(direction)
    ch.writeLine(str(res.movement))
```

3.3 実行結果

以上のような実装により、各プランや関数は図 4 のように通信を行い周囲の環境に応じた失敗からの回復行動が可能となった。

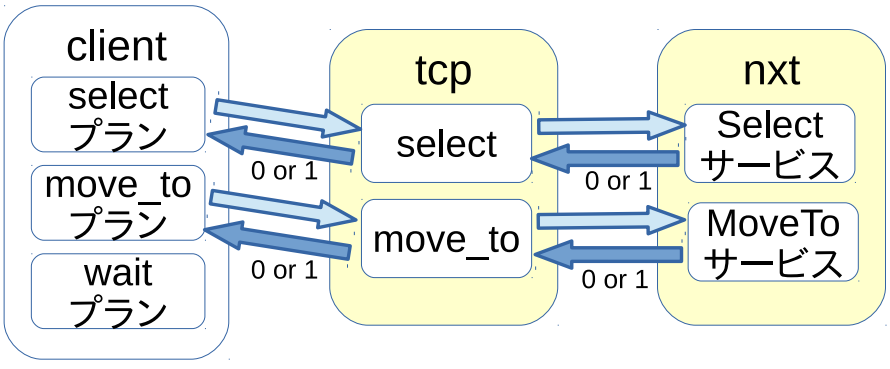

図 4: 通信

目標物を認識せず到達に失敗したとき

目標物への到達失敗後、client エージェント内の move_to プランで指定した角度だけロボットが右回転し、その方向へ向けて前進を 10 回繰り返した。移動完了後にはその地点で始めと同様の探索、到達行動を開始した。(図 5)

目標物を認識したが到達に失敗したとき

到達失敗後に現在いる地点でロボットが 15 秒静止した。15 秒後、目標物を探索するためにロボットは再び動き始めた。(図 6)

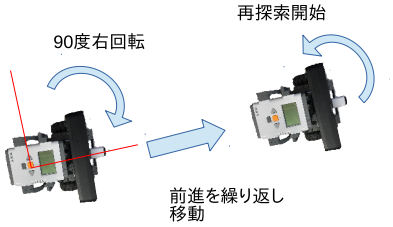

図 5: 目標物を認識しなかったとき

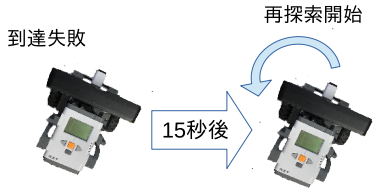

図 6: 目標物を認識したとき

また、第一目標物への探索行動を 2 回行っても到達出来なかった場合には、Jason 側の命令として第二目標物へと目標物を変更した。以下は Jason の実行画面である。
ただし、第一目標物を'lena'、第二目標物を'rilakkuma' とする。

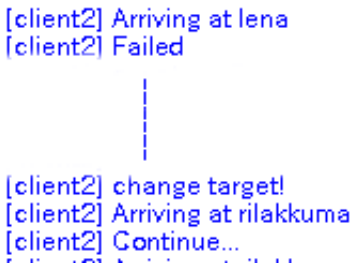

図 7: 目標物変更画面

以上の画面からもわかるように、第一目標物への到達の失敗を繰り返したのちに「change target!」の表示とともに現在の目標物が第二目標物に変更されている。

4　まとめ

ROS ノードとの通信を用いた client エージェント内で様々なプランを作成したことによって、失敗に対する回復処理行動を周囲の環境に応じて選択をするようになった。また、今までは失敗の概念がなく、第一目標物に到達しなければ次の第二目標物の探索を開始せず、第一目標物が存在しない場合でも到達するまで目的達成行動をとっていたが、失敗を繰り返すことで第二目標物へと目標を変更するようになり、人間の思考や行動に近い動きが簡単にではあるが可能となった。今後の研究では、プランを追加する分だけより複雑な目的達成行動ができるため、様々なロボットの行動パターンを想定しプランの作成を行い、より複雑な人間の思考に近い行動をとる自律ロボットの実現が望まれる。例えば今回は別地点へ移動する際の回転角度を client エージェント内のプランの引数としてあらかじめこちらから与えているが、将来的にはエージェントが探索時や過去の知識を利用して目標物がありうる方向を自ら考え、回転角度を決定できることを目指したい。待機時間についても同様に、探索時の知識やカメラ映像を用いた知覚的な知識を上手く取り入れ、自律的に時間を設定できることが必要となる。また道の状況などから通行可能か否かを判断し、それに応じた探索行動や目標物の設定をエージェント自身が行うようにすることも将来課題である。

5　謝辞

本論文の執筆および研究にあたり、いつも親身にご指導してくださった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、貴重な時間を割いてまで多くの知識や示唆を頂きました樽井先輩や兼松先輩をはじめ、新出研究室の皆様に感謝の意を表します。ありがとうございました。

参考文献

[1] 兼松明未. ROS による自律ロボットの目標地点への到達行動の実現について. 2015 年度卒業論文, 奈良女子大学理学部情報科学科, 2016.

[2] 柿原玲奈. 自律ロボットの実現に向けた ROS による環境情報の取得. 2015 年度卒業論文, 奈良女子大学理学部情報科学科, 2016.

[3] 山本千尋. 実世界における行為の失敗の概念を考慮した自律ロボットの実装について. 2016 年度卒業論文, 奈良女子大学理学部情報科学科, 2017.

[4] 小谷麻緒. 方位情報を用いた自律ロボットの行為の失敗認識の効率化について. 2016 年度卒業論文, 奈良女子大学理学部情報科学科, 2017.

[5] Rafael H. Bordini, Jomi Fred Hübner, Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason.* John Wiley & Sons, 2007.

[6] Aaron Martinez and Enrique Fernandez. *Learning ROS for Robotics Programming.* Packt Publishing, 2013.