2016年度　卒業論文

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

奈良女子大学　理学部　情報科学科　4回生　新出研究室
13251771　山本千尋

平成29年2月

目 次

概要

近年、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求められている。特に、視覚情報をもとに目標物に到達するという目的を達成するロボットの実現には、ロボットに視覚情報から目標物の認識を行う機能を取り入れることが求められ、先行研究において ROS を用いて RGB-D カメラからの画像取得を行い、ロボットがそれを用いて目標物に到達する能力の実装が行われた。しかし、従来のロボットでは、行為に失敗した際にそのことを認識せず、その後の行動が適切に出来ていなかった。我々は、ロボットが行為に失敗した場合にその失敗を認識し、次の目標に向かって適切な行動をとることを目指し、構築された自律ロボットに新しく失敗を認識する能力の追加を行った。

1　はじめに

1.1　背景

我々は、人間と同様に、自らが考えて行動するコンピュータシステムの実現を目指して研究に取り組んでいる。

何らかの環境においてその環境を感知し、人間と同様に自身の目標を達成するための手段を選んで行動するシステムを「自律エージェント」と呼ぶ。近年、人間の代わりとなるシステムとして盛んに研究されている。それに伴い、「自律ロボット」が世に多く普及してきている。日々の生活の中で一番身近な自動お掃除ロボットから宇宙開発まで多岐に渡る分野で活躍している。

このような自律ロボットは目標達成のための能力の１つとして「目標物を認識してその場所に移動する」という力を要することが多いと考えられる。そこで、先行研究ではそうした行動の汎用的な形での実現を目指し、安価な小型ロボットとオープンソースのソフトウェアを用いて構築を行った。今回、我々は構築された自律ロボットの行動に新しく失敗を認識する能力を加えた。さらにその失敗を認識することの効率化と失敗後の行動の柔軟化を目指した。

本研究は、奈良女子大学理学部 4 回生小谷、石井との共同研究である。本論文では、失敗を認識する能力の追加について述べる。ロボットの失敗認識の効率化については [1]、ロボットの失敗からの回復行動については [2] で述べられている。

1.2　先行研究

先行研究 [3] において、既存の NXT Python ライブラリを基に、より上位レベルの記述が可能になるよう作られた NXT Python+を構築し、ロボットの行動制御が可能になった。また、ROS を用いて RGB-D カメラからの画像

取得を行い、「ROS を使ったカメラから得た視覚情報を使用した物体追跡をするロボット」が作られた [4][5]。

しかし、従来のロボットでは、行為に失敗した際にそれを認識せず、その後の行動が適切にできなかった。ロボットが目的の達成に失敗したときにはまず、その失敗をロボット自らが認識する必要がある。また、その後その失敗を認識した上で、次の目標に向かって適切な行動をとる必要がある。

よって、我々は失敗を認識して失敗時に柔軟な回復行動がとれることを目指した。その手段として BDI という考え方を使う。BDI とは、まず目標の達成手段（プラン）を大まかに定め（それを意図と呼ぶ）、その過程で必要に応じて副目標（サブゴール）の達成方法を決めていくという考え方である。この考え方により、失敗からの回復時を含めた目標達成の過程において、その時点での状況に応じた手段の決定ができる。BDI によるエージェントを構築するため、そのプラットフォームとして 3 節で述べるように Jason[6] を用いた。

2　NXTとROS

2.1　LEGO Mindstorms NXT

本研究では小型ロボットとして教育用 LEGO MINDSTORMS NXT(以下 NXT と記載)（図 1） を使用した。付属の LEGO ブロックを組み合わせて自在にロボットを成形することが可能で、今回フロント部分に Xtion カメラとコンパスセンサーを搭載しており、その機能を活用した。

図 1: コンパスセンサーを搭載した NXT

NXT と PC の接続には USB または Bluetooth による通信を用いる。本研究では Bluetooth によってワイヤレスで制御することにより NXT を動作させた。また、Xtion と PC の接続には USB 接続を用いた。

2.2　NXT Python+

NXT の行動制御にはこれまでの研究で構築された NXT Python+ライブラリ [3] を使用している。これは、既存の NXT Python ライブラリを基に、より上位レベルの記述が可能になるよう作られたものである。NXT Python

上に、直進や回転などの基本行為のメソッド群、その基本行為を組み合わせたメソッド群、パラメーター入力の GUI が用意されている。

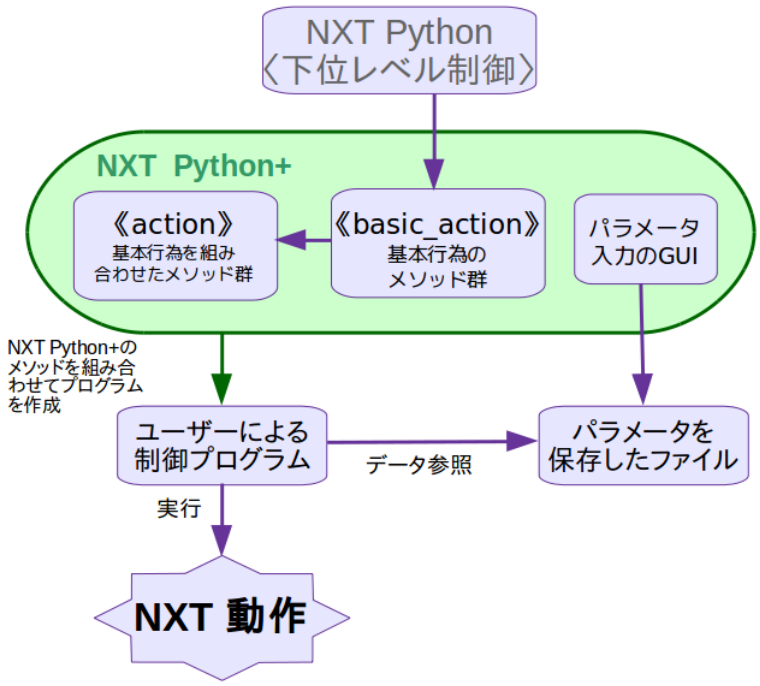

図 2: NXT Python+

- basic_action
  下位レベルの制御命令を持つ NXT Python ライブラリ上に、直進や回転などの基本行為のメソッド群を実装したもので、basic_action.py というファイルにまとめられている。NXT の起動、Bluetooth 通信などプログラムの実行までに必要な処理もまとめられている。

- action
  basic_action の基本行為を組み合わせた、さらに上位の行為のメソッド群を実装したもの。action.py というファイルにまとめられている。

- パラメータ入力の GUI
  メソッドに必要なパラメータの入力を容易に行うために構築された。GUI をユーザーが選んで使用することができる。

ユーザーが NXT Python+にあるメソッドを自在に組み合わせて作成したプログラムが実行されると、パラメータを保存したファイルからデータを参照し、NXT が動作する（図 2）。

2.2.1 使用したクラスとメソッド

本研究では、basic_action.py と action.py ファイル内に定義されている以下のクラスとメソッドを使用した。

- basic_action.py
  - Basic_Action クラス — compass_value()
    コンパスセンサーの値を得るメソッド。
- action.py
  - Action クラス — compass_angle_decl(t1, t2)
    t1 からからみて t2 ほど左の角度を得るメソッド。

2.3 ROS

ROS（Robot Operating System）[7] とは世界中で広く使用されているロボット用ソフトウェアフレームワークである。米国のロボットベンチャー企業である Willow Garage 社が開発を開始し、現在は Open Source Robotics Foundation という NPO が開発・無料公開しており、ソフトウェア開発者のロボット・アプリケーション作成を支援するライブラリとツールを提供している。具体的には、ハードウェア抽象化、 デバイスドライバ、ライブラリ、視覚化ツール、 メッセージ通信、パッケージ管理などが提供されている。

ROS はそれぞれの機能を有したソフトウェアをノードとして複数同時に実行し、それぞれのノードがお互いにデータをやりとりするシステムになっている。トピックを経由してメッセージがノード間でやりとりされる（図 3）。それぞれのノードが独立しているため、ソフトウェアの再利用性が高いという利点がある。

本研究では ROS を動作させるプラットフォームとしては公式サポートされている Ubuntu を選択している。商用の環境に依存せずに、ROS を使用した開発が可能である。ROS は OpenCV などの画像処理ライブラリとも連携しやすい。プログラミング言語は、プログラムの実装やメモリ管理が容易な Python を選択した。

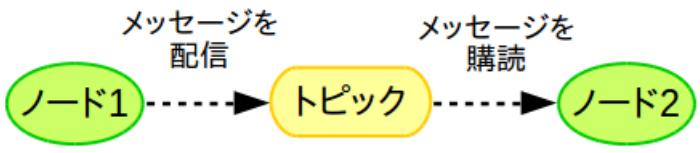

図 3: トピックによる通信モデル

3　実装

3.1　Jason と ROS の通信

このロボットは、NXT Python+で実装した基本行為を用いたプランによって行動するエージェントとして Jason[6] で構築されている。ROS 側で物体検出から行動選択がなされるとその結果が Jason 側に伝わり、エージェントが適切な行動をとる（図 4）。

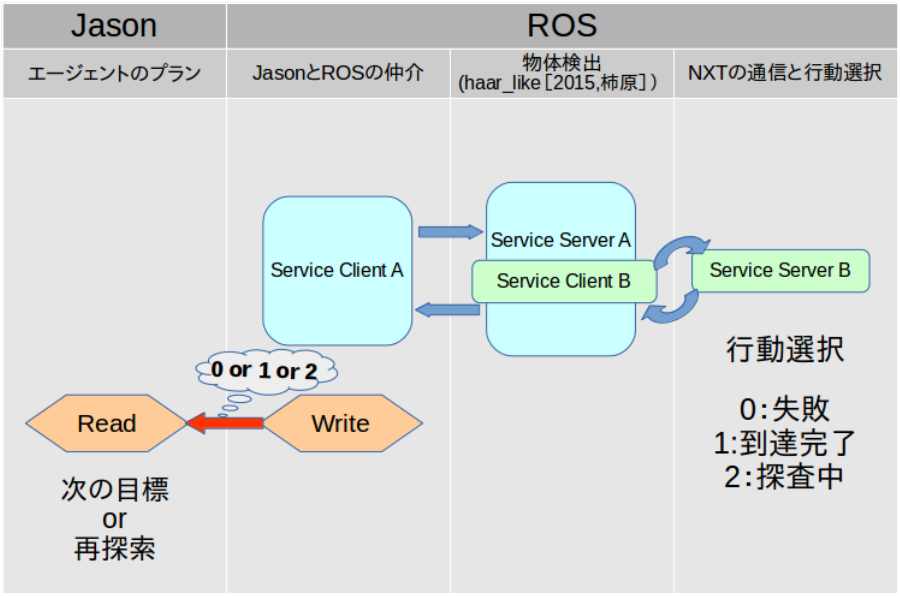

図 4: Jason と ROS の通信

3.2　失敗の認識について

我々は以下の 2 つの行為に対する失敗の認識を可能にした。

1. ロボットが目標物を認識する行為
2. ロボットの目標物への到達行動
   ここでの到達行動とは、ロボットが探索をはじめてから認識行為、目標物に向かう移動の一連の行動を指す。

1. の失敗の認識は基本行為の失敗と捉えられるため ROS 側で、2. の失敗の認識はプランの実行の失敗と捉えられるため Jason 側で行った。次にそれぞれの失敗認識の方法を述べる。

3.2.1 目標物認識の失敗

従来、ロボットは画像認識の誤りにより目標物を見失った場合、その場で向きを変えながら目標物の再認識を試行していたが、見つからない場合は向きを変えながらその場で回転を際限なく続けていた。そこで我々は、ROS 側では、ロボットが探索を始める時の位置から３６０度探索した場合、目標物を認識せず、到達しなかったと判断するようにした。

まず、探索を始めるときの方向（以下 A と記載）とそこから 180 度反対の方向（以下 B と記載）の値をコンパスセンサーを使用して取得する。ロボットは左回転で探索を行う。その際、回転の１つの動きをするごとにそのときの方位の値（以下 n とする）を取得する。これを繰り返し行い、n の値が 1 回目に B を通った上で A を通ると、初めの位置から 360 度探索を行ったということになる。したがって到達失敗となる（図 5）。

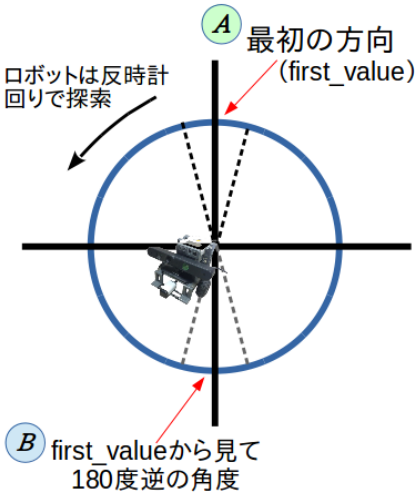

図 5: ROS 側の実装

3.2.2 到達行動の失敗

現実世界では、モータの回転の不正確さによる行動の不正確さなどに起因して目標物に到達できず、そのまま時間が経過し続けることがあり得る。そこで我々は、Jason 側で、目標物到達のプランの実行に時間がかかりすぎる場合は、それを失敗と見なすようにした。

時間の計測は内部アクション .time[6] を用いて行う。まず、NXT と通信を始めた時間を取得し、信念に記憶させる。次に現在の時間を信念に記憶させながら探索行動を行う。そして、NXT と通信を始めた時間からある一定の時間が経過すれば到達失敗とする。

3.3　実行結果

3.2.1 で述べた、目標物の認識に失敗した場合のロボットの動作を図 6 に示す。目標物がないので見つけられない場合、ロボットは、１周回って失敗を判断し一旦到達行動を止めた。なお、失敗を認識した場合 “Failed” とコンソールに表示するようにエージェントを構築しており、一周回った際に “Failed” と表示されたので、この段階で失敗を認識したということがわかった。従来は、ロボットは、目標物を見つけるまでこのまま回り続けていたが、失敗を認識するようになった。さらに失敗した後、別の行動をとることもできるようになった [2]。

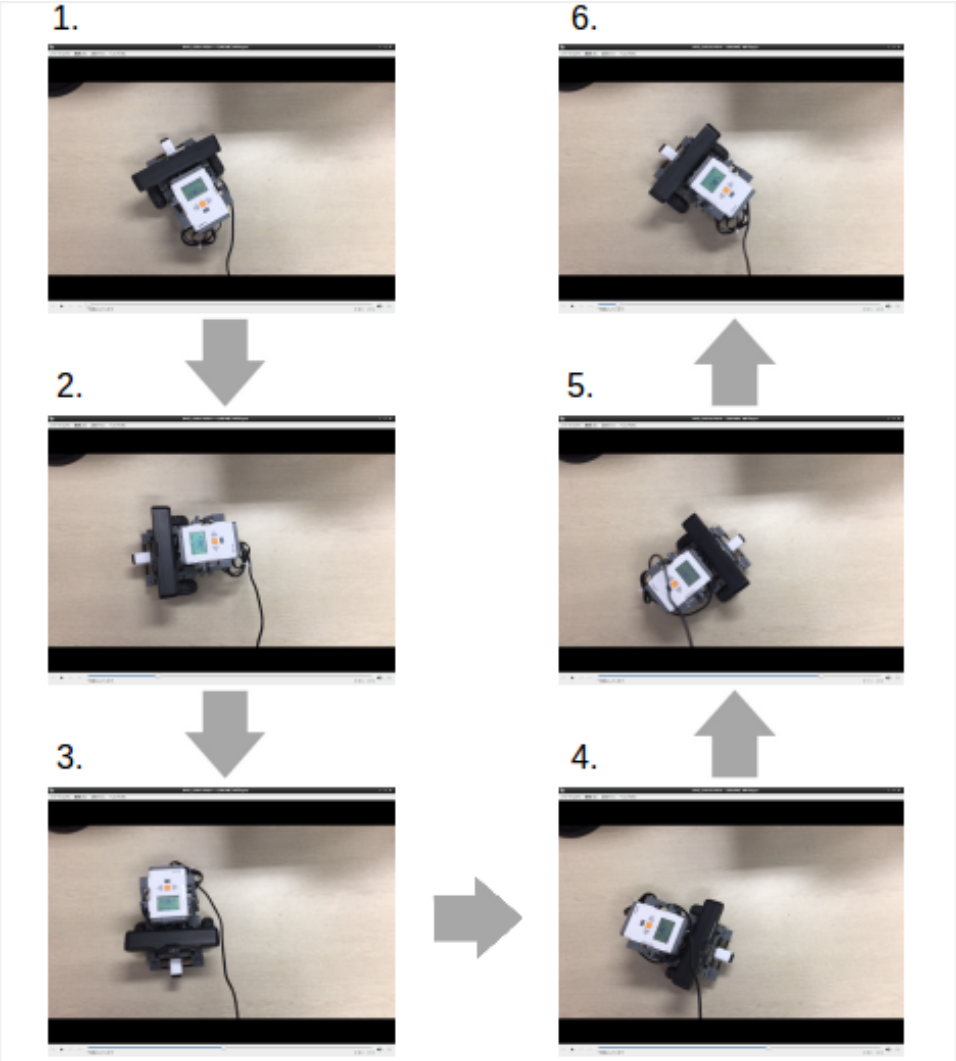

図 6: ROS 側　実行結果

また、3.2.2 に述べた、ロボットの目標物への到達行動のプランの実行に時間がかかった場合、Jason 側でそのことを認識することができた。失敗と認識するまでの経過時間は任意に設定できる。

目標物を lena、失敗と見なすまでの時間を３０秒として実行し、図 7 にその時の Jason のコンソール出力を示す。動作をコンソール画面で確認できるよう、エージェントは随時その時の状態に関する情報をコンソールに出力するようになっている。“Connecting to NXT” は Jason が NXT に接続中であることを表し、接続されればその時刻を “connect time” で示す。“Arriving at lena” は目標物 lena に接近中であることを、“now” とそれに続く “=” は現在の時刻と０時からの秒数を表す。また、“Failed” “time out” は経過時間が所定の秒数以上となって失敗と見なされたことを表す。

図では、NXT と通信を始めた時刻が “connect time” で表示されており、その時刻から３０秒以上経った時点で “Failed” “time out” と表示し失敗と見なされていることがわかる。

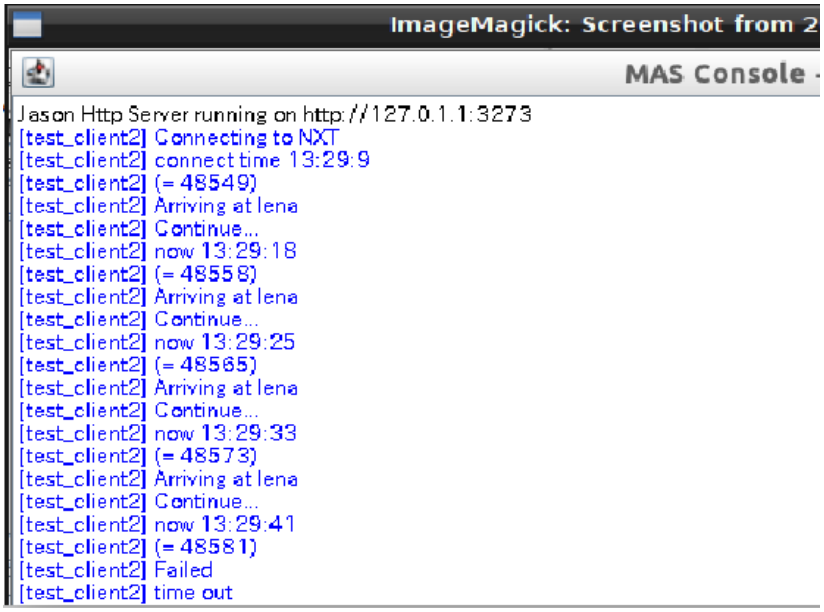

図 7: Jason　コンソール画面

4　まとめ

コンパスセンサーを用いたロボットの向き、および Jason の内部アクションを用いた時間を取得することで、ロボットの行動選択に失敗を取り入れ、ゴールの失敗を認識できるようになった。

今後の課題として、本研究は、ロボットが行為に失敗した場合の振る舞いに関するものであるが、それとは別にロボットの目標達成の成功率を上げるには、行為の失敗をできるだけ少なくする必要があり、それには目標物の認識率を今より上げることが望まれる。また、今回はカメラ画像の取得に ROS を用いているが、今後はより複雑なロボットの行動制御にも ROS を使用することも課題として挙げられる。

5 謝辞

本研究を遂行するにあたり、熱心にご指導して下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、研究室配属時から丁寧なご指導、助言を与えてくださっている、奈良女子大学大学院人間文化研究科情報科学専攻の兼松明未さん、そして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] 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, 奈良女子大学大学理学部情報科学科, 2017.

[2] 石井あすか. 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. 2016 年度卒業論文, 奈良女子大学大学理学部情報科学科, 2017.

[3] 小島侑子. 小型ロボット制御のための汎用ライブラリの構築. 2011 年度修士論文, 奈良女子大学大学院人間文化研究科, 2012.

[4] 柿原怜奈. 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, 奈良女子大学大学理学部情報科学科, 2016.

[5] 兼松明未. ROS による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, 奈良女子大学大学理学部情報科学科, 2016.

[6] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming Multi-Agent Systems in agentSpeak using Jason*. John Wiley & Sons, 2007.

[7] ROS.org. `http://www.ros.org/`.