2017 年度 卒業論文

自律ロボット
行動制御ライブラリの拡張

奈良女子大学 生活環境学部 情報衣環境学科
生活情報通信科学コース 4 回生
新出研究室 14480253 小松芙美子

平成 30 年 2 月

目次

概要

近年、環境の変化に対して、自律的に目標を達成するロボットの実現が望まれている。当研究室では、カメラや機械学習を利用した目標物の認識や BDI エージェントと ROS を用いた到達行動の実現、失敗の概念や失敗に対する回復行動の構築を行い、自律ロボットの実現を目指した。しかし、目標物ではない物体を目標物と判断したり、動作が不正確なために目標物と衝突する問題があった。そこで、問題を改善するために目標物認識率や行動精度の向上を目指し、先行研究で用いたロボットよりも高精度なロボットを使用し、行動制御ライブラリの拡張を行った。また、深層学習による画像認識能力を持つ自律ロボットの実装を行った。本論文では、高精度なロボットを使用するための行動制御ライブラリの拡張について述べる。

1 はじめに

近年、自律ロボットが実現され、実世界で活用されている。例えば、災害救援ロボットやお掃除ロボットなど、幅広い分野で使用されつつある。これらのロボットは、人間による指令や判断がいつも与えられる状況下にあるわけではない。そのため、人間のように何らかの環境変化を感知し、自ら行動選択を行う自律性が必要になってくる。当研究室では、環境の変化に対応した意図的な行動選択を行わせるために、BDI モデルによる意思決定を行う自律ロボットの実現を目指した研究を行っている。

先行研究 [1][2][3][4] では、Haar-Like 特徴量とカラーヒストグラムを用いて目標物を認識し、そこへ到達する行動や失敗の概念、失敗に対する回復行動の構築が行われた。しかし、目標物ではない物体を目標物と判断したり、目標物の認識が不正確なために衝突するという問題があった。

そこで、我々は、先行研究で用いたロボットである教育用 LEGO Mindstorms NXT よりも高精度なロボットである教育用 LEGO Mindstorms EV3 を使用するために行動制御ライブラリを拡張し、さらに深層学習による画像認識能力の実現を行うことで、目標物認識率や行動の精度の向上を目指した。なお、本研究は奈良女子大学大学院人間文化研究科 2 回生兼松との共同研究である。深層学習による物体認識能力や認識精度の結果については、兼松の修士論文 [5] に掲載する。本論文では、行動制御ライブラリの拡張について述べる。

2 使用するロボット

2.1 LEGO Mindstorms EV3

本研究では、ロボットとして教育用 LEGO Mindstorms EV3[6](以下 EV3 と記載) を使用した。付属のブロックを組み合わせて自在にロボットを作成できる。先行研究で用いたロボット NXT よりもインテリジェントブロックやセンサの性能が向上しており、SD カードによる OS の搭載が可能になった。今回は、図 1 のように、EV3 に、Xtion カメラと超音波センサ、ジャイロセンサ等を搭載し、行動実行の際に活用している。

EV3 は、USB や Bluetooth の通信を用いてコンピュータ側から遠隔制御することが可能である。USB による通信では、EV3 が動作出来る範囲が限定されてしまうため、本研究では、Bluetooth による通信を用いた。

図 1: Xtion カメラとセンサを搭載した EV3

2.2 ev3dev-lang-python

本研究では、EV3 に Debian GNU/Linux ベースの OS である ev3dev[7] を搭載している。この OS 上で EV3 を制御するライブラリはいくつか存在している。我々は、メモリ管理が不要であり、開発が容易である Python を開発言語としたため、Python 言語を使用しているライブラリ ev3dev-lang-python[8] を選択した。このライブラリは、個々のモータやセンサを制御するために作成されたものである。それらのデバイスに対する下位レベルの制御は行えるが、複数のモータやセンサを用いた直進や右折左折などの上位レベルの行動制御の機能は持っていない。

3 ロボット制御方法

EV3 の制御には、意図的な行動選択を行う BDI エージェントと、ロボット開発で利用される ROS を使用している。

3.1 BDI エージェント

BDI エージェントとは信念、願望、意図の 3 つの心理状態を使用し意志選択を行うエージェントである。エージェントは、信念、願望から目標を設定し、目標を達成する手段を意図とする。意図を実行した後の環境の知覚や信念、願望による目標の再設定を繰り返すことで目標を達成する。1 つの目標に対して、複数のプランを持つことができ、目標を並列して複数持つことが可能なため、環境の変化に対して柔軟な対応ができる利点がある。

ライブラリや ROS によって実現される基本行為を用いて自律的に行動する BDI エージェントの構築には Jason[9] を用いた。Jason は、BDI エージェントの構築プラットフォームであり、目標を達成する行動プランを記述している。Jason は、開発言語として AgentSpeak を使用し、外部環境を Java で実装している。

3.2 ROS

ROS(Robot Operating System)[10] は、ロボット開発用ソフトウェアフレームワークである。ROS には、プログラムを動かすためのパッケージがあり、その内部にライブラリを実行するプログラムであるノードが存

在する。このノード同士が通信を行い、やりとりすることで、ロボットのそれぞれの機能を独立したプログラムとして記述でき、修正の際に全体を修正することなくそのノードの修正だけで対処することができる。また、作成したプログラムの再利用がしやすく、画像処理ライブラリとの連携が容易である。ROS は、開発環境に依存せず多くの言語が使用できるため、本研究では、動作プラットフォームとして Ubuntu を利用し、開発言語として Python を使用した。

3.2.1 ROS の通信

ROS ノード間の通信方法として、図 2 のようなトピックを用いた一方向の通信がある。トピックはノード間のメッセージの受け渡しに用いられる、名前の付いた通信路である。また、メッセージには、受け渡す情報のデータ構造を定義したメッセージ型が用意されている。ユーザがメッセージ型を定義することも可能であり、本研究でも新たにメッセージ型の定義を行った。ノードは Publisher(配信者) と Subscriber(購読者) を用いて通信を行い、同じトピックに対する Publisher と Subscriber は互いに同じメッセージ型を送受信する。トピックを用いた通信は、Publisher から Subscriber へ一方向にメッセージを受け渡す。また、1 つのノードに複数の Publisher と Subscriber を定義したり、1 つのトピックに複数の Publisher や Subscriber が通信することもできる。

また、他の通信方法として、サービスを経由した双方向通信がある。サービスは、受けとった情報を受け渡したり、受け渡した情報に対して処理を行った結果を返すことが出来る。サービスには、受け渡す情報と返り値のデータ構造を定義したサービス型が用意されている。ユーザがサービス型を定義することも可能である。

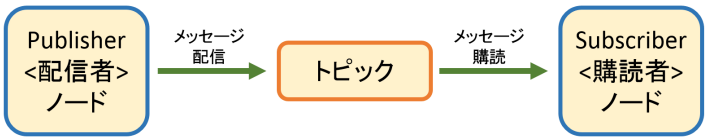

図 2: ROS メッセージを用いた一方向通信

4 実装

4.1 行動制御ライブラリの拡張

今回使用するロボット EV3 で、行動の精度向上を行った上で、先行研究で実現した複雑な行動を実行させる際に、先行研究で構築された NXT 向けの行動制御ライブラリ NXT Python+[11] をそのまま使用できない。また、2.2 節で述べたように、ev3dev-lang-python ライブラリには、個々のセンサやモータを制御することが出来るが、モータやセンサを複数同時に使用する直進や右折左折などの複雑な行動制御を行うことができない。したがって、EV3 の行動制御ライブラリの拡張を行った。下位レベルの制御をまとめた ev3dev-lang-python をもとに、EV3 を起動し行動制御プログラムを実行するまでに必要なセンサやモータの使用準備を行う命令と直進や回転、センサの値取得などの基本行動、基本行動を組み合わせた行動を行う命令を構築した。以下に、NXT Python+ に存在した命令を改善して定義したメソッドを示す。詳しくは、4.2.2 節で述べる。

- 基本行動
  - move(speed, time, angle)
  - rotate(left_speed, right_speed, time)
- 基本行動を組み合わせた行動
  - rotate_left_until_angle(angle, speed)
  - rotate_right_until_angle(angle, speed)
  - adjust_heading(angle, goal_angle, speed)

また、先行研究では、前進する回数を指定し直進していたが、この直進行動を用いて到達行動を行うと、到達に成功したかどうかを判断する前に目標物に衝突する危険があった。そのため、EV3 に搭載している超音波センサを用いて目標物との距離を測定し、測定した距離をもとに直進する行動の構築を行った。この行動を実現するためには、距離を指定してモータを動作させる能力が必要であるが、ev3dev-lang-python には、用意されていない。しかし、相対的にタイヤの回転角度を指定しモータを動作させる能力は存在しているため、指定した距離からタイヤの回転角度を計算することで、距離を指定して直進する行動を実現し、新たなメソッドとして定義した。タイヤの回転角度は下記の式を用いて求めた。新たに定義したメソッドは、基本行動を組み合わせた行動 move_until_cm(distance, speed) とした。詳しくは、4.2.2 節で述べる。

タイヤの回転数 = 指定した距離 / (タイヤの直径 × 円周率)
タイヤの回転角度 = タイヤが 1 回転する角度 × タイヤの回転数

4.2 行動制御ライブラリの ROS 化

先行研究で使用したロボットは、図 3 のように、意図的な行動選択を行うエージェントとして Jason で実装され、Jason やロボットとの通信、物体検出、行動制御を行うノードからなる ROS によって制御を行っていた。行動制御を行う際は、ROS から行動制御ライブラリ NXT Python+ を呼び出し利用している。同様の方法で EV3 の制御を行うと、ある行動命令と別の行動命令が同時に同じセンサの値を取得しようとする際に、競合状態が発生し、Python プログラムのエラーが発生して行動を停止するため、到達行動を続行できない場合があった。

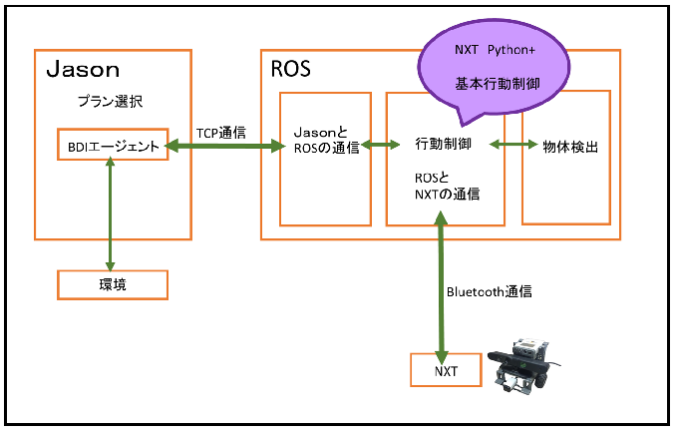

図 3: 先行研究でのロボット制御図

そこで、拡張した行動制御ライブラリをプログラムパッケージ ev3dev_ros、既存のノードをプログラムパッケージ robo17 として ROS 上にまとめた。図 4 のように、パッケージ同士でデータをやりとりすることで、競合を解消した。なお、EV3 の既存のライブラリ ev3-lang-python は EV3 上に存在するが、拡張した行動制御ライブラリをまとめたプログラムパッケージが存在する ROS や Jason も同様に搭載すると、動作がかなり低速になり開発しにくい。そのため、ROS や Jason は EV3 の外部にある計算機上で動作させることで、動作の低速化を防止し、開発を容易にしている。

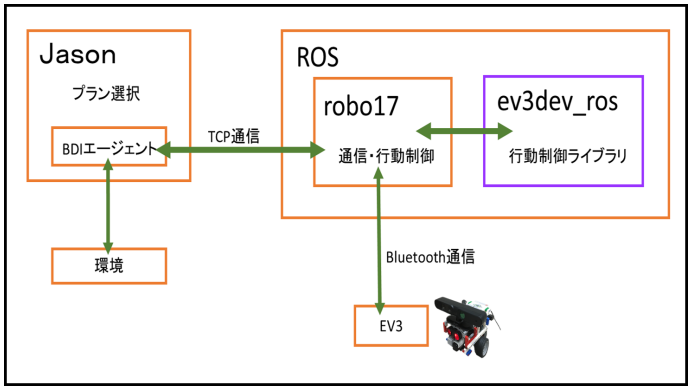

図 4: EV3 制御図

4.2.1 robo17 パッケージ

EV3 の制御を行うパッケージ。物体検出を行う detect_object ノード、EV3 との通信・行動制御を行う ev3 ノード、BDI エージェントを構築する Jason と ROS の通信を行う tcp ノードをまとめている。

robo17 パッケージでは、図 5 のようにトピックを用いた通信を行っている。まず、Jason からの行動命令を tcp ノードが受け取った後、ev3 ノードを呼び出して行動制御を開始する。ev3 ノードでは、ev3dev_ros パッケージの ev3_action ノードを呼び出し、行動動作命令を EV3 へ伝え、行動させる。行動制御の過程で、物体検出や認識をする必要があるため、detect_object ノードから ev3 ノードへ Objects トピックを経由し、メッセージを配信している。ここでのメッセージは、自作のメッセージ型を定義した。メッセージのデータ構造は次のようになっている。

- Objects トピックを経由するメッセージのデータ構造
  - data: 物体認識の結果。データ型は Object[ ]。
  - frame_width: Xtion カメラから取得した画像枠の高さ。データ型は int16。
  - frame_height: Xtion カメラから取得した画像枠の幅。データ型は int16。

ev3 ノードでの行動制御が終わった後、ev3 ノードから行動が成功したかどうかの情報を tcp ノードに返し、tcp ノードから Jason へその情報を伝える。

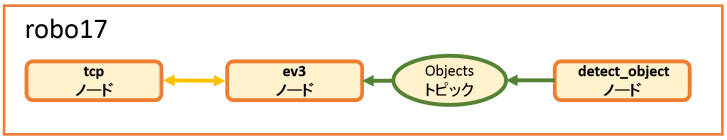

図 5: robo17 パッケージ

4.2.2 ev3dev_ros パッケージ

4.1 節で拡張を行った行動制御ライブラリをノードとしてまとめている。リアルタイムに値をやりとりするため、センサやモータを個々のノードに分けた。パッケージには、以下のノードが存在する。

- ev3_action ノード
  基本行動や基本行動を組み合わせた行動を行うノード。EV3 の既存ライブラリ ev3dev-lang-python を基にして作られたメソッド群を持つ Action クラスが定義されている。EV3 が行動する際に必要な機能がそれぞれ分けられたメソッドとなっており、メソッドを単体で使用することができる。
  - __init__(ev3)
    行動に必要なセンサの受信やモータ動作情報の配信設定を行う init メソッド。
    引数の ev3 には、既存ライブラリ ev3dev-lang-python のモジュールを指定する。
  - compass_cb(msg)
    コンパスセンサの値を受信するメソッド。
    引数の msg には、compass_sensor ノードから受け取ったメッセージを指定する。
  - gyro_cb(msg)
    ジャイロセンサの値を受信するメソッド。
    引数の msg には、gyro_sensor ノードから受け取ったメッセージを指定する。
  - touch_cb(msg)
    タッチセンサの値を受信するメソッド。
    引数の msg には、touch_sensor ノードから受け取ったメッセージを指定する。
  - ultrasonic_cb(msg)
    超音波センサの値を受信するメソッド。
    引数の msg には、ultrasonic_sensor ノードから受け取ったメッセージを指定する。

  ○基本行動メソッド
  - move(speed, time, angle)
    直進するメソッド。speed はモータの回転速度、time は稼働時間、angle はタイヤの回転角度を示す。speed、angle は正の値を指定すると前進、負の値を指定すると後退する。稼働時間やタイヤの回転角度を指定し直進する能力を追加した。また、前進する前や前進途中で物体との距離がある一定の値以下になると、衝突を防ぐために前進を中止する能力も追加した。
  - rotate(left_speed, right_speed, time)
    回転するメソッド。left_speed は左モータの回転速度, right_speed は右モ ータの回転速度, time は稼働時間を示す。稼働時間を指定し回転する能力を追加した。
  - rotate_left(speed, time)
    左回転用に作成したメソッド。speed はモータの回転速度、time は稼働時間を示す。稼働時間を指定し左回転する能力を追加した。
  - rotate_right(speed, time)
    右回転用に作成したメソッド。speed はモータの回転速度、time は稼働時間を示す。稼働時間を指定し右回転する能力を追加した。

– stop()
モータ停止させるメソッド。

○基本行動を組み合わせた行動メソッド

– adjust_heading(angle, goal_angle, speed)
回転した後の角度誤差の修正を行うメソッド。主に後述する rotate_until_angle で用いる。angle は指定した回転角度、goal_angle は目標とする位置、speed はモータの回転速度を示す。例えば、EV3 の現在の向きが 30、rotate_until_angle(60, 100) である場合、adjust_heading の引数は、angle = 60、goal_angle = 90 となる。angle は正の値を指定すると右回転、負の値を指定すると左回転する。先行研究 [3] では、0〜4 度の誤差があったが、誤差を約 0〜2 度に修正できるようになった。

– move_until_cm(distance, speed)
現在地から指定した距離 (cm) まで直進するメソッド。distance は指定した距離、speed のモータの回転速度を示す。distance は正の値を指定すると前進、負の値を指定すると後退する。前進の場合、現在測定した距離よりも指定した距離が大きいと、衝突する可能性があるため、前進を行わない。

– rotate_left_until_angle(angle, speed)
現在地から指定した角度になるまで左回転するメソッド。angle は回転角度、speed はモータの回転速度を示す。angle は正の値のみ指定できる。

– rotate_right_until_angle(angle, speed)
現在地から指定した角度になるまで右回転するメソッド。angle は回転角度、speed はモータの回転速度を示す。angle は正の値のみ指定できる。

– rotate_until_angle(angle, speed)
現在地から指定した角度になるまで回転するメソッド。angle は回転角度、speed はモータの回転速度を示す。angle は正の値を指定すると右回転、負の値を指定すると左回転する。

- gyro_sensor ノード
ジャイロセンサの使用準備や値取得を行うノード。EV3 の既存ライブラリ ev3dev-lang-python を基にして作られたメソッド群を持つ Gyro クラスが定義されている。ジャイロセンサに関する機能がそれぞれ分けられたメソッドとなっており、メソッドを単体で使用することもできる。センサの値は現在の角度を返す。

– __init__(ev3)
ジャイロセンサの設定を行う init メソッド。
引数の ev3 には、既存ライブラリ ev3dev-lang-python のモジュールを指定する。

– publish()
ジャイロセンサの値を配信するメソッド。

– gyro_value()
ジャイロセンサの値を返すメソッド。

– gyro_reset()
ジャイロセンサの値を初期化するメソッド。

- ultrasonic_sensor ノード
  超音波センサの使用準備や値取得を行うノード。EV3 の既存ライブラリ ev3dev-lang-python を基にして作られたメソッド群を持つ Ultrasonic クラスが定義されている。超音波センサに関する機能のメソッドであり、メソッドを単体で使用することもできる。センサの値は物体との距離 (cm) を返す。
  - __init__(ev3)
    超音波センサの設定を行う init メソッド。
    引数の ev3 には、既存ライブラリ ev3dev-lang-python のモジュールを指定する。
  - publish()
    超音波センサの値を配信するメソッド。
  - gyro_value()
    超音波センサの値を返すメソッド。

- touch_sensor ノード
  タッチセンサの使用準備や値取得を行うノード。EV3 の既存ライブラリ ev3dev-lang-python を基にして作られたメソッド群を持つ Touch クラスが定義されている。タッチセンサに関する機能のメソッドであり、メソッドを単体で使用することもできる。センサの値はタッチしているかの真理値を返す。
  - __init__(ev3)
    タッチセンサの設定を行う init メソッド。
    引数の ev3 には、既存ライブラリ ev3dev-lang-python のモジュールを指定する。
  - publish()
    タッチセンサの値を配信するメソッド。
  - gyro_value()
    タッチセンサの値を返すメソッド。返り値は True か False。

- compass_sensor ノード
  コンパスセンサの使用準備や値取得を行うノード。EV3 の既存ライブラリ ev3dev-lang-python を基にして作られたメソッド群を持つ Compass クラスが定義されている。コンパスセンサに関する機能のメソッドであり、メソッドを単体で使用することもできる。センサの値は現在の方角を返す。
  - __init__(ev3)
    コンパスセンサの設定を行う init メソッド。
    引数の ev3 には、既存ライブラリ ev3dev-lang-python のモジュールを指定する。
  - publish()
    コンパスセンサの値を配信するメソッド。
  - gyro_value()
    コンパスセンサの値を返すメソッド。

- tacho_motor_l ノード
  左モータの使用準備や左モータの稼働設定を行うノード。EV3 の既存ライブラリ ev3dev-lang-python を基にして作られたメソッド群を持つ Motor_L クラスが定義されている。左モータに関する機能のメソッドがある。
  - __init__(ev3)
    左モータの設定を行う init メソッド。
    引数の ev3 には、既存ライブラリ ev3dev-lang-python のモジュールを指定する。
  - set_motor(msg)
    モータの稼働コマンドを受け取り、コマンドで指定された動作を行うメソッド。 引数の msg には、ev3_action ノードから受け取ったメッセージを指定する。

- tacho_motor_r ノード
  右モータの使用準備や右モータの稼働設定を行うノード。EV3 の既存ライブラリ ev3dev-lang-python を基にして作られたメソッド群を持つ Motor_R クラスが定義されている。右モータに関する機能のメソッドがある。
  - __init__(ev3)
    右モータの設定を行う init メソッド。
    引数の ev3 には、既存ライブラリ ev3dev-lang-python のモジュールを指定する。
  - set_motor(msg)
    モータの稼働コマンドを受け取り、コマンドで指定された動作を行うメソッド。 引数の msg には、ev3_action ノードから受け取ったメッセージを指定する。

robo17 パッケージと同様に ev3dev_ros パッケージでもトピックを用いた通信を行っている。通信の様子を図 6 に示す。gyro_sensor、ultrasonic_sensor、touch_sensor、compass_sensor ノードから、それぞれ Gyro トピック、Us トピック、Touch トピック、Compass トピックを通り、ev3_action ノードへメッセージを配信している。これらのメッセージは、既存のメッセージ型を利用し、Gyro トピックと Compass トピックを経由する通信ではデータ型 int64 のメッセージ、Us トピックを経由する通信ではデータ型 float64 のメッセージ、Touch トピックを経由する通信ではデータ型 bool のメッセージとして定義されている。

また、ev3_action ノードは tacho_motor_l、tacho_motor_r ノードへ Motors トピックを経由しメッセージを配信することで、モータを動作させる。ここでのメッセージは、自作のメッセージ型を定義した。メッセージのデータ構造は次のようになっている。

- Motors トピックを経由するメッセージのデータ構造
  - cmd: モータの動作コマンド。データ型は string。
  - left_speed: 左モータの速度。データ型は float64。
  - right_speed: 右モータの速度。データ型は float64。
  - time: 動作時間。データ型は float64。
  - position: タイヤの回転角度。データ型は int64。

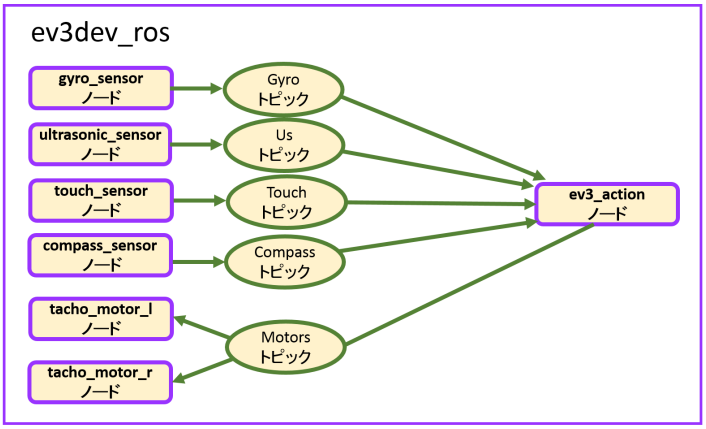

図 6: ev3dev_ros パッケージ

5　行動制御ライブラリの検証

5.1　基本行動を組み合わせた行動による精度検証

拡張した行動制御ライブラリ内にある基本行動を組み合わせた行動について、精度検証を行った。この検証では、行動制御ライブラリにある行動のみを対象とし、BDI エージェントや robo17 パッケージとの通信を用いた目標到達行動は行っていない。検証は、研究室内の平らな卓上で行った。

5.1.1　move_until_cm

指定した距離まで直進するメソッド move_until_cm の検証結果について述べる。ロボットを 9 回ずつ指定した距離まで前進させた際の誤差を測定した。前進した場合の結果を表 1 に示す。前進した時の誤差は平均すると 0.7cm になることが分かる。この誤差はロボットが動作を続行するのに影響を与える可能性は低いため、十分な行動精度であるといえる。

回数	1	2	3	4	5	6	7	8	9
目標とする距離	10	10	10	20	20	20	30	30	30
実際に進んだ距離	9.5	9.4	9.6	19.2	19.4	19	29	29	28.9
誤差	0.5	0.6	0.4	0.8	0.6	1	1	1	1.1

表 1: 前進した場合の誤差

5.1.2　rotate_until_angle

指定した角度まで回転するメソッド rotate_until_angle の検証結果について述べる。ロボットを左右に 6 回ずつ指定した角度まで回転させた際の誤差を測定した。回転したの結果を表 2 に示す。表 2 から、右回転左回転のどちらの場合においても誤差が 2 度以下になることが分かる。先行研究 [3] では、この誤差が 4 度であったため、行動精度が向上したといえる。

	右回転						左回転					
回数	1	2	3	4	5	6	1	2	3	4	5	6
目標位置	45	45	180	180	360	360	45	45	180	180	360	360
実際位置	45	46	180	180	359	361	45	44	178	180	360	359
誤差	0	1	0	0	−1	1	0	−1	−2	0	0	−1

表 2: 右回転・左回転した場合の誤差

5.2 目標到達行動による性能検証

拡張した行動制御ライブラリを用いた目標到達行動による性能検証を行った。この検証では、ある条件下で目標物を探索、認識し、目標物へ到達するという目標を設定した。前提条件として、目標物と EV3 の間に障害物が存在しないとした。目標物については、事前に正解画像と不正解画像を収集し、学習を行っており [5]、物体によって認識できる距離が異なっているため、認識できる距離の平均である 1.2m を認識可能距離とした。

条件を、条件 1. 目標物との距離が十分に離れている場合、条件 2. 目標物との距離が近い場合として検証を行った。また、検証開始時に目標物を認識している場合と認識していない場合について、目標到達回数を計測し、分析を行った。EV3 と物体の距離が 3cm 以内である場合、カメラが物体の全体像を取得出来ず、物体検出や物体認識を行うことが不可能であるため、条件 2 では、検証開始時に目標物を認識していない場合のみを想定した。

5.2.1 条件 1. 目標物との距離が十分に離れている場合

目標物との距離を 1.3m とし、到達させたい目標物を設定した。まず、EV3 は衝突を防ぐために物体との距離を測定する。探索行動可能な距離であると判断し、検証開始時に目標物を認識している場合は、その目標物に向かって到達行動を行った。到達行動には、2 種類の行動がある。目標物が視界の右側または左側にある際は、目標物が視界の中央に位置するように回転を行い、目標物が視界の中央にある際は、前進を行う。

探索行動可能な距離であると判断し、検証開始時に目標物を認識していない場合は、目標物を探索する行動を行った後、目標物を発見・認識できた場合は到達行動を行った。探索行動では、右回転を行い周囲を探索する。到達行動は、目標物を認識している場合と同様である。

目標到達回数は、検証開始時に目標物を認識している場合と認識していない場合の両方とも 10 回中 10 回であった。検証では、到達率が高いという結果になったが、物体が認識可能圏外に存在する場合では、探索と到達行動を交互に繰り返すため、既に認識している場合と比べると到達に時間がかかった。

5.2.2 条件 2. 目標物との距離が近い場合

目標物との距離を 2cm とし、到達させたい目標物を設定した。まず、EV3 は衝突を防ぐために物体との距離を測定する。物体との距離が 3cm 以内であった場合、物体との距離が近すぎるため、物体の全体像を取得できない。よって、物体検出や認識ができず、このまま探索行動を行うと回転時に物体と衝突する可能性がある。この問題を回避するため、EV3 は物体検出可能になる距離まで後退する。その後、条件 1 と同様に目標物の認識状況によって、行動選択し到達行動を行った。

目標到達回数は、10 回中 10 回であった。このような結果になった要因は、後退行動をとることで衝突や認識できない問題を回避できたためだと考えられる。

6 まとめ

行動制御ライブラリを拡張し、ROS で利用可能になったため、BDI モデルを用いた行動プランやセンサを用いたリアルタイムな行動が実行可能になった。さらに別途行った深層学習による画像認識能力との結合でより高度な目標認識を可能にした。これらによって、目標到達率が高い行動能力を持つ自律ロボットの構築を行うことができた。

今後の課題として、探索時に得た情報を用いて目標物を探索する行動の構築が必要である。例えば、目標物 A を探索している途中で、物体 B を発見し、物体 B の位置を記憶しておく。その後、物体 B を目標物として与えられた際、新たに探索行動を行うのではなく、記憶した物体 B の位置情報を用いることで、効率的な探索行動を実現できるだろう。また、本研究で実装した自律ロボットは、目標物がカメラの視界内にない場合、探索に失敗し、目標到達できない可能性がある。目標到達の失敗が起こる状況として、目標物との間に障害物が存在する場合や視界内に目標物は見当たらないが目標物の位置を把握している場合などが想定できる。この問題を解決するために効率的な到達行動の構築が必要である。さらに、正確な到達行動を行うための物体検出や物体認識の改善の必要がある。

7 謝辞

本研究および本論文の執筆にあたり、丁寧にご指導をしてくださった新出尚之准教授、研究結果に関して議論していただいた近畿大学理工学部の高田司郎教授に深く感謝致します。また、本研究に関して、お忙しい中、熱心にご助言していただいた兼松先輩、そして多くのご指摘をくださいました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] 樽井志織. 目標到達機構を持つ自律的な小型ロボット制御を行う BDI エージェントの構築. 2016 年度修士論文, 奈良女子大学大学院人間文化研究科, 2017.

[2] 石井あすか. 自律ロボットの目標達成における行動の柔軟化について. 2016 年度卒業論文, 奈良女子大学理学部情報科学科, 2017.

[3] 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.

[4] 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.

[5] 兼松明未. 自律ロボットの目標物到達機構の精度向上とそれを用いて行動する BDI エージェントの構築. 2017 年度修士論文, 奈良女子大学大学院人間文化研究科, 2018.

[6] 教育版レゴ®マインドストーム®EV3 | LEGO Education. http://www.legoedu.jp/ev3/.

[7] ev3dev HOME. http://www.ev3dev.org/.

[8] ev3dev-lang python. https://github.com/ev3dev/ev3dev-lang-python/.

[9] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. Programming multi-agent systems in Agentspeak using Jason. Wiley, 2007.

[10] ROS.org. http://www.ros.org/.

[11] 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.