2014 年度 卒業論文

小型ロボット制御のための
汎用ライブラリの拡張

奈良女子大学 理学部 情報科学科 ４回生 新出研究室
11251631 樽井志織

平成 27 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求められている。このようなロボットの実現を目指し、小型ロボット制御のための実験をより円滑に行うために、先行研究において NXT Python+と呼ばれるライブラリの構築が行われた。しかしこれには、動的環境下での正確な動作に問題があった。

本研究は、構築された NXT Python+を更に発展させ、動的環境下においても小型ロボットが活動できるように、新たに学習機構の追加を行うものである。

1 はじめに

1.1 研究背景

今日、人間の雑務を代行するロボットや、コミュニケーションをとることの出来るロボットなど、様々な機能を持ったロボットが普及している。我々はそのなかでも、自律的に目標を達成するロボットの実現を目指し、研究を行っている。しかし、人間が活動する実世界では、つねに環境が動的に変化しており、現代では、動的な環境下においても、自律的に目的を達成できるロボットの実現が望まれている。

先行研究では、自律的に目標を達成するロボットの実現のため、比較的安価である程度の性能を持つロボットを使用して実験を行う足掛かりとして、既存のライブラリからより上位レベルのライブラリの構築を行った。[3] しかし、2.1 節で述べるように、実世界でさまざまな環境で動作するにあたって、モーターに与えるパワーなどのパラメータの調整が必要であり、これに手間がかかる難点があった。

そこで、本研究では、先行研究で構築された NXT Python+を拡張し、動的環境下においても自律的に目標を達成するロボットの実験を行えるようにするために、新しく学習機構を追加し、ライブラリの拡張を行う。

なお、本研究は奈良女子大学 4 回生江川との共同研究である。本論文では、NXT Python+に追加した学習機構の実装について述べ、ライブラリを使用した検証実験については [2] で述べられている。

1.2 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）[1] を使用した。NXT は、ロボットの頭脳の役割をするインテリジェントブロックに、その他の様々な形のブロックを組み合わせてロボットを成形する。インテリジェントブロックには、出力ポートが 3 つ、入力ポートが 4 つ存在し、モーターやセンサーなどを接続して、自由にロボット製作を行う事が可能である。例として、実験に使用した NXT の 1 つ（図 1）には、サーボモーターが 3 つ出力ポートに、コンパスセンサーが 1 つ、ライトセンサーが 2 つ、超音波センサーが 1 つ、計 4 つのセンサーが入力ポートに接続されている。

NXT を動かすための制御プログラムは、USB の接続または Bluetooth の通信によって、PC から転送される。本研究では、Bluetooth による通信を使用して、NXT を動作させた。

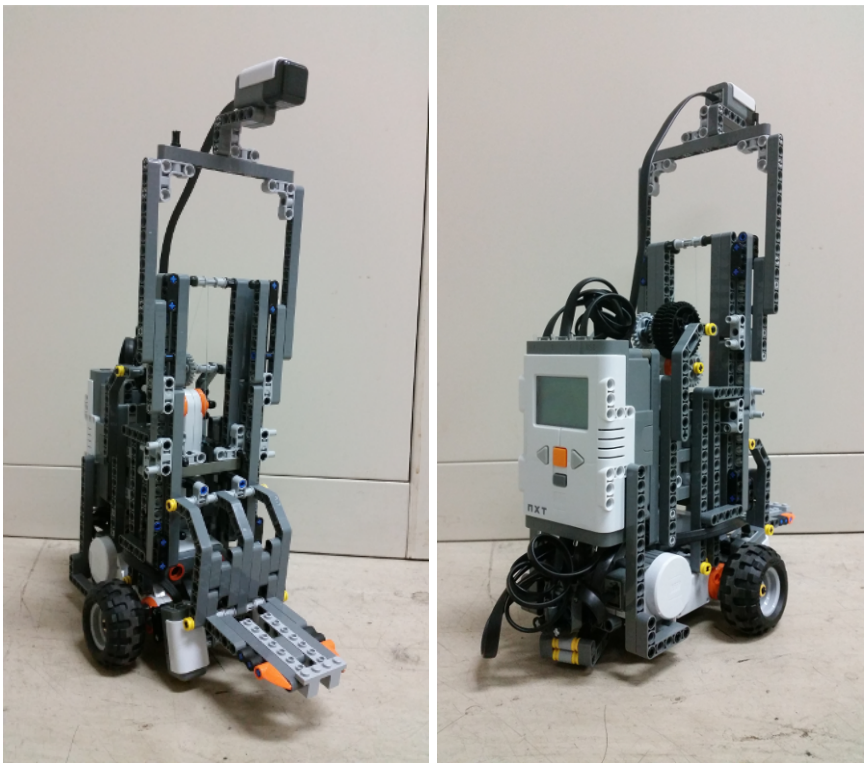

図 1: LEGO Mindstorms NXT

1.2.1 NXT

本節には、本研究で使用する NXT の機能について述べる。

- インテリジェントブロック

  前述の通り、NXT の脳の役割を果たすブロックである。上部に出力ポートが 3 つ、下部に入力ポートが 4 つあり、PC からの制御プログラムを USB または Bluetooth で受け取る。

- モーター

  パワーと動作時間を指定して回転を行う。インテリジェントブロックの出力ポートに接続し、主にタイヤを動かす。本研究で使用した NXT には 3 つめのモーターが接続されており、このモーターはリフトに繋がっている。

- ライトセンサ

  光度を計測するセンサー。本研究では、NXT の下部に取り付けられ、床の色を識別している。このセンサーがラインの黒と、床の白を識別して、ライントレースを行っている。

- コンパスセンサ

  方位を計測するセンサー。NXT 上部に取り付けられ、NXT の向いている方角を計測している。本研究での 90 度回転における強化学習で、90 度回転を行えているかどうかの判定に使用されている。

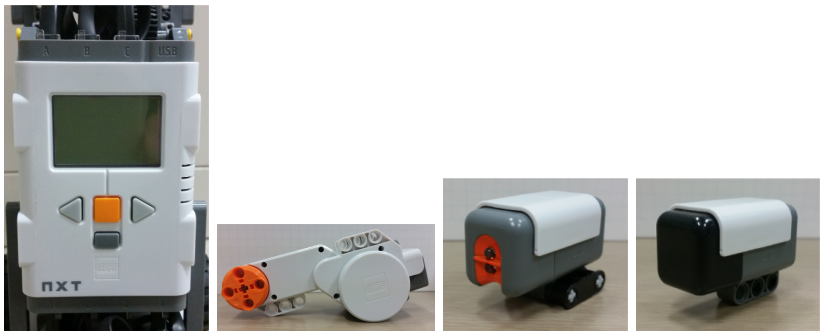

図 2: 左からインテリジェントブロック、モーター、ライトセンサ、コンパスセンサ

1.3 NXT Python+について

先行研究では、プログラミングに慣れていない人でも視覚的にポートなどの設定が行えるようにするため、スライダやボタンなどを使用した入力操作で設定が行える GUI を設計した。これを受けて、GUI が容易に設計出来るよう、ライブラリ作成の言語は Python が選択された。

NXT には、既にいくつかのライブラリが存在しており、Python を使ったライブラリには NXT Python と呼ばれるライブラリがあった。しかし、このライブラリは、モーターの回転数を直接制御するといった下位レベルの制御命令が基になっており、複雑な命令を NXT に与えようとした場合、プログラムの記述が長くなるという特徴があった。これは、自律的に目標を達成するロボットの実験を行う場合、下位レベルの記述のみではプログラムの記述が長くなり、記述ミスが多くなるということや改良がし辛いという欠点になる。よって、複雑な命令がある程度短い記述で行えるようにするため、NXT Python を基にした、下位レベルの制御命令を組み合わせたライブラリである NXT Python+が先行研究にて構築された。

1.3.1 機能

NXT Python+には、NXT Python の制御命令を組み合わせて構築された、直進や回転などの基本行為のメソッド群である basic_action と、その basic_action の基本行為を組み合わせ構築された、“ライントレースを行う” など basic_action より上位の行為のメソッド群である action が設計されている。

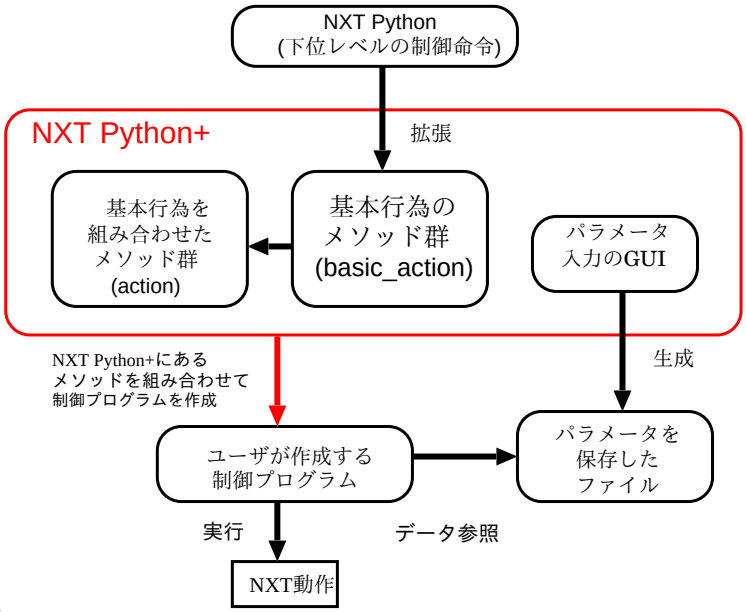

図 3: NXT Python+

さらに、その 2 つのメソッド群で使用されるモーターを回転させるパワーの値や、どのポートにどのセンサーが接続されているかなどの情報を容易に入力出来るようにするための GUI も設計されている。(図 4) この GUI は、各パラメータをスライダやボタンで入力し、その情報をデータファイルに保存しておく機能を備えており、制御プログラムを実行する際はこのデータファイルを参照する。これにより、パラメータを変更した場合でも、プログラムファイルを修正する必要がなくなった。

2 NXT Python+の拡張について

2.1 動的環境下における問題点

NXT Python+を構築したことにより、小型ロボット制御のプログラムを、従来より短く簡潔に記述できるようになった。また、GUI を設計したことで、センサーの接続などが変更された場合でも、パラメータを容易に調節できるようになった。

しかし、NXT Python+を動的環境下において使用する場合、環境が変わる毎に GUI を使用し、パラメータを調節する必要があった。例えば、リフトに荷物を乗せるなどの動作で、NXT の重量が変化した場合、パラメータを変更せずに 90 度回転を行うと、正確に曲がる事ができないことがある。よって、環境が変化した時は、GUI でのパラメータの調節は必須である。しかし、設定にはある程度の時間がかかるため、長期間の連続的な実験などでは、GUI による設定を省略する手段が必要であった。

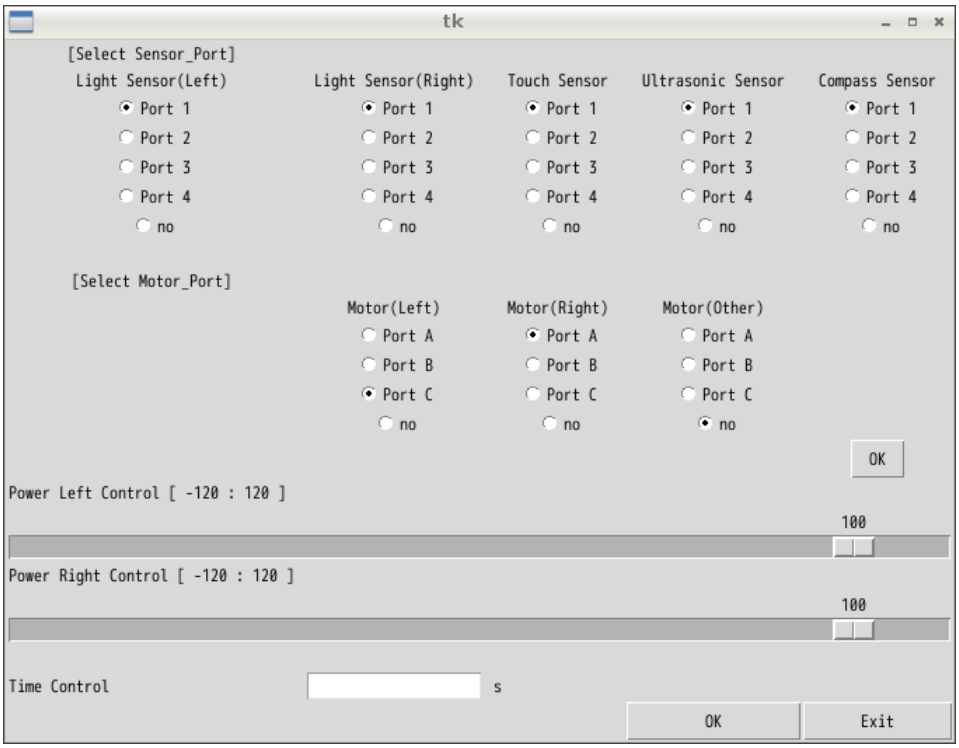

図 4: GUI

本研究では、上記の問題を解決するために、NXT Python+に学習機構を追加した。特に、NXT を動作させるにあたって、使用頻度の高いメソッドである turn_right_90() および turn_left_90() に注目した。この 2 つのメソッドは、GUI で調節したモーターのパワーの値と走行時間を使用し、左右に 90 度回転を行うメソッドである。従来、このメソッドを使用する時は、GUI にてパラメータの設定を行う際に、使用する NXT に合わせてパワーの値を調節していた。この手間を省くため、強化学習を使用し、90 度回転に最も適切なパラメータを学習するライブラリとして Learn クラスを作成した（図 5)。

なお、本研究では Python のバージョンは Python2.7.6 を使用した。

2.2 強化学習

学習機構で行う強化学習には ε-greedy 法を使用した。

まず、NXT にはいくつかのパラメータが選択肢として与えられる。このパラメータは、

1. 90 度回転を行うためのモーターのパワーの値
2. そのパワーでどの程度回転することができるのかを値にした平均報酬値
3. その選択肢を何回試行したのかを表す試行回数

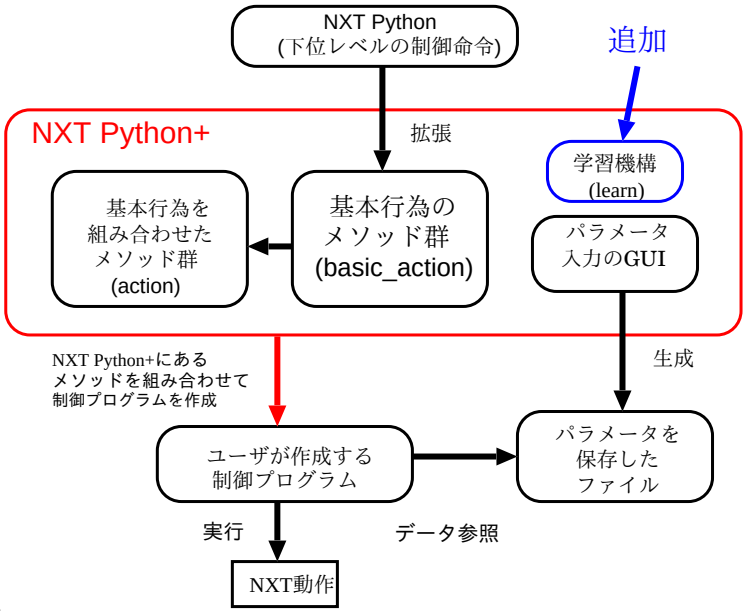

図 5: 学習機構の追加

の 3 つから成る。初期状態では報酬値と試行回数は 0 になっている。

選択肢が与えられると、NXT は交差点まで前進し、一時停止する。そして与えられた選択肢の中から、最適なパラメータを使用して 90 度回転を行うが、初期の状態では、どの選択肢が最も適切であるのかといった情報が存在しない。この情報を学習するために、パラメータを選ぶ際には以下のような方法を取る。

1. 確率 ε で、選択肢からランダムに 1 つパラメータを選ぶ
2. 確率 $1-\varepsilon$ で、その時点で最適値と推測されるパラメータを選ぶ

上記のように、確率 ε でランダム選択を行い、$1-\varepsilon$ で最適値を選択するような方法を ε-greedy 法と呼ぶ。

選ばれたパラメータで試行した NXT には、その都度、“どの程度回転を行えたのか” という評価が与えられる。これを報酬と呼び、この報酬値の平均が最も高い値が、90 度回転を行うためのモーターのパワーの最適値となる。

2.3 実装

2.3.1 評価方法

NXT は ε-greedy 法で繰り返し試行を行い、どのパワー値でどの程度回転を行うことが出来るのかを学習する。学習を行うためには、この試行の度に、90 度回転に評価を与える必要があった。

評価方法には人手による評価方法と、コンパスセンサによる評価方法がある。極力人手を介さず、正確に評価を行うことができるため、本研究ではコンパスセンサを使用して 90 度回転の評価を行った。

まず、回転を行う前の NXT の位置から、正確に 90 度回転した場合の評価を “1” とする。その位置から離れていくごとに評価の値は下がり、全く回転しない場合と 180 度回転した場合の評価が “0” となる。例えば、試行で NXT が 30 度回転した場合、90 度からは 60 度離れているので、評価は “0.3333” となる。また、120 度回転した場合も、90 度から 60 度離れているので、30 度の場合と同じ評価が与えられる。

以上の処理を行う comp_reward() メソッドを使用して、90 度回転における評価を行った。

2.3.2 機能

Learn クラスは、先行研究にて構築された action や basic_action を基にして作られたメソッド群を持つクラスである。学習に必要な機能が、個々の機能ごとに細分化されたメソッドとなっており、個々のメソッドを単体で使用することも可能である。ただし、学習のみを行う場合は main メソッドを実行するだけでよい。

learn ファイルにまとめられたメソッドは以下の通りである。

- __init__()

  前回の学習結果をもとにして、学習に使用する選択肢とイプシロンの設定を行うメソッド（コンストラクタ）。前回の学習結果が存在しない場合はデフォルトの選択肢が設定される。

- greedy_select(param)

  平均報酬の推定値が最も良い選択肢の番号を返すメソッド。param は選択肢のリストを示す。main メソッドで使用される。

- random_select(size)

  ランダムに選んだ選択肢の番号を返すメソッド。size は選択肢の個数を示す。main メソッドで使用される。

- random_LR()

  ランダムに右か左の値を返すメソッド。main メソッドで使用される。

- trial(param, selection, LR)

  選んだ選択肢のパラメータを使用して 1 回試行した後、その結果に応じて報酬を返し、平均報酬の推定値と選択肢を選んだ回数を更新するメソッド。param は選択肢のリスト、selection は選んだ選択肢の番号、LR は左右のどちらかを示す。

  main メソッドで使用される。

- main()

  強化学習を実行するメソッド。

  ライントレースを行い、交差点で試行を行うかどうかを確認した後に学習を行う。試行しない場合は左右のどちらかに曲がるか、直進するかを選択し、再び交差点まで前進する。

  一定回数試行を行った後に停止する。また、一定回数毎に結果をファイルに出力する。

- turn_right_kg()

  学習したパラメータで右折するメソッド。データファイルに保存されているパラメータを、モーターのパワーに使用して 90 度回転を行う。

- turn_left_kg()

  学習したパラメータで左折するメソッド。

- make_para(param, cut)

  前回の学習結果から、学習に使用するパラメータの選択肢を生成するメソッド。パラメータの値を中心とした選択肢を生成する。param は選択肢のリストを、cut は選択肢に使用されるパラメータの間隔を示す。なお、生成されるパラメータの数は、引数の param に依存する。

- comp_reward(be)

  コンパスセンサの値から学習の報酬を取得し、報酬値を返すメソッド。be は移動前のコンパスセンサの値を示す。

- ep_cut(ep, cut)

  選択肢に使用されるパラメータの間隔をもとに、 ε の値を変更し、その値を返すメソッド。結果の平均報酬値が高ければ ε は低く設定され、平均報酬値が低ければ ε は高く設定される。

  これによって NXT は、平均報酬値が低い場合にランダムで選択肢を選びやすくなるため、不適切なパラメータの選択肢を連続して選ぶ確率が低くなる。

  ep は ε の値を、cut は選択肢に使用されるパラメータの間隔を示す。

- cut_set()

  前回の学習結果から、選択肢に使用されるパラメータの間隔を決定し、その値を返すメソッド。

  前回の結果の平均報酬値が高ければ間隔を小さくし、平均報酬値が低ければ間隔を大きくする。例えば、平均報酬値が低く前回の結果の近似値では十分に回転が行えない場合には、前回の結果から離れたパラメータで選択肢を生成するように間隔を設定する。

2.3.3 補助メソッド

通常の学習機構の他に、繰り返し学習を行うことを想定し、補助を目的としたメソッドの構築も行った。例えば、一度学習を行ったが、元々選択肢の中に適切なパラメータが存在しなかった時などは、最も 90 度に近い値で回転できるパラメータを選ぶことは出来るが、正確に 90 度の回転を行うことは出来ない。このような場合、学習そのものを繰り返すことで、問題点を解決することが出来る。この補助として、以下の 3 つのメソッドを構築した。なお、この 3 つのメソッドは 2.3.2 節にて一度詳細を述べている。

- make_para(param, cut)
- ep_cut(ep, cut)
- cut_set()

この 3 つは、前回の結果をもとに、次の学習の調節を行うメソッドである。

2.3.4 メソッドの使用

以下に、上記メソッド使用の流れを述べる。

1. learn クラスのインスタンスを生成する。この時、コンストラクタである__init__() が呼び出され、選択肢の生成が行われる。以下は__init__() での流れである。
2. __init__() 内の cut_set() が、選択肢に使用するパラメータの間隔（以下刻み幅と記載）を決定するため、前回の結果の中で最も高い平均報酬値をデータファイルから読み込む。上記に記した通り、この値が高ければ、刻み幅は小さくなる。もし前回の結果がなければ、刻み幅は 7 となる。
3. 同じく__init__() 内の make_para(param, cut) が、パラメータの選択肢を生成する。cut_set() で決められた刻み幅の値を引数で受け取り、前回の結果を中心とした選択肢を生成する。前回の結果がなければ、74 を中心とした選択肢が生成される。例えば、刻み幅 7 で、パラメータの数が 3 つの場合は、選択肢は (67,74,81) となる。
4. 同じく__init__() 内の ep_cut(ep, cut) が、刻み幅から ε の値を決定する。刻み幅は前回の結果をもとにして決まっているので、ep_cut(ep, cut) も cut_set() と同じく、前回の結果から ε を決定していることになる。ただし、前回の結果がなければ、ε は 0.3 となる。

   ここまでが__init__() 内での流れである。
5. 学習を行うメソッドである main() を実行する。以下は main() 内での流れである。
6. action クラスの linetrace_forward() を使用し、交差点まで直進する。ここで試行を行わない場合は、進行方向を決定し、再び交差点まで直進する。

実行する場合、random_LR() で曲がる方向を決定し、試行に使用するパラメータの決定を行う。ep_cut(ep, cut) で設定した ε の確率でランダムにパラメータを決定する random_select(size) を実行し、残りの確率で、平均報酬が高いと推測されるパラメータを選ぶ greedy_select(param) を実行する。

7. trial(param, selection, LR) に、選ばれた選択肢の番号と左右のどちらかが与えられ、試行が行われる。回転前の方位をコンパスセンサで計測し、comp_reward(be) に与えることによって、行った試行に対しての評価を行う。このメソッドによって出された報酬値を過去の報酬の合計に合算し、平均報酬の推定値を更新する。同時に、選んだ選択肢の選択回数も更新される。

以上の 5 から 7 を繰り返し、NXT は学習を行う。学習自体を繰り返す場合は、一度学習が終わった後に、もう一度 1 から学習を行う。

3 まとめ

本研究では、動的環境に適応していなかった NXT Python+に学習機構を追加し、動的な環境下においても 90 度回転を行えるように、NXT Python+の拡張を行った。また、このライブラリを使用し、実際に学習を行って、NXT が異なった環境に適応できることを確認した。これにより、環境が変化した時でも、なるべく人手を介することなく、NXT のパラメータの調節を行い、90 度の右折または左折を行うことができるようになった。

今後の課題としては、本研究で学習機構を追加し学習を行えるようにしたのは 90 度回転のみであるため、その他の動作においても動的環境に適応できるように、ライブラリの拡張を行うことで、ロボットの動きをより現実世界での動きに近付けることができると考えられる。また、本研究では NXT の進路選択などは人間の手によって行われているため、より人手を介さない学習方法を使い、学習機構を改良することが望ましい。他にも、学習機構だけでなく、action ファイルなどに新たなメソッドを追加することによって、ライブラリを発展させることができる。

4 謝辞

本研究の遂行および本論文の執筆にあたり、丁寧にご指導して下さった新出尚之准教授に深く感謝致します。また、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] Lego Group. Lego.com エデュケーションオフィシャルサイト. http://education.lego.com/.

[2] 江川鈴菜. 強化学習による環境別のロボットの基本行為の獲得について. 2014 年度卒業論文, 奈良女子大学理学部情報科学科, 2015.

[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.