SkyAI: 高度にモジュール化された強化学習ライブラリ
— スクリプトインターフェイス —

○山口 明彦（学振特別研究員，奈良先端大）　小笠原 司（奈良先端大）

SkyAI: Highly Modularized Reinforcement Learning Library
— Script Interface —

*Akihiko Yamaguchi (JSPS Research Fellow, NAIST), Tsukasa Ogasawara (NAIST)

Abstract— This paper introduces an open source library of reinforcement learning (RL) methods, named SkyAI (available from `skyai.sourceforge.net`). SkyAI is a highly modularized RL library for real/simulated robots to learn behaviors. Our ultimate goal is to develop an artificial intelligence (AI) program with which the robots can learn to behave as their users wish. In this paper, we describe an overview of SkyAI, and its script interface. We also demonstrate the applications to crawling tasks of both a humanoid robot in simulation and real robots.

Key Words: Reinforcement Learning, Open Source Library, Robot Control, Script Interface

1. はじめに

強化学習手法は，報酬関数によって表現された行動目的のみから，ロボット自身に行動を獲得させる手法であり，将来のロボットに不可欠な機能のひとつである．強化学習の実用化に向けて多くの研究が行われているが [1, 2, 3, 4]，現状では「ロボット制御に応用するための強化学習の研究」が大半であり，「強化学習を用いて実用的なシステムを構成する研究」は少数であると考えられる．前者から後者への移行を促進するためには，前者の研究で開発された手法群（次元縮約 [2]，モデルの利用，階層化，模倣学習 [3]．学習した知識の再利用 [4] など）を容易に統合し，実システムに組み込める手段が必要である．

我々は，強化学習を始めとする機械学習手法群を実システムへ容易に実装できるようにするため，SkyAI の開発を進めている．SkyAI は高度にモジュール化されたオープンソースの強化学習ライブラリであり（**Fig.**1），多様な場面に柔軟に対応できる工夫がほどこされている．その一方で，実機に適用可能なレベルの実行速度を維持している．本稿では，SkyAI の柔軟さを実現しているスクリプトインターフェイスを中心に解説する．

以下，2. 章で SkyAI の概略を述べ，3. 章でスクリプトインターフェイスについて解説する．4. 章でシミュレーション及び実機の小型ロボットの運動学習への適用例を示す（これらはベンチマークタスクとして SkyAI に含まれる）．5. 章でまとめる．

2. SkyAI の概略

SkyAI は（1）高度なモジュール化，(2）高い実行速度と柔軟性，(3）開発の容易性，を実現するように開発されている．これらを実現するため，コンパイル言語（具体的には C++）で開発して実行速度の要求を満たし，スクリプトインターフェイスを導入することで，プログラムをコンパイル後もモジュール構造の再構成を行えるようにする，という方針で実装している．また，モジュールの開発を容易にするためのソースコードを生成するような外部ツールは使用せず，標準のプリプロセッサとコンパイラのみで処理できるように実装している．以下，詳細に述べる．

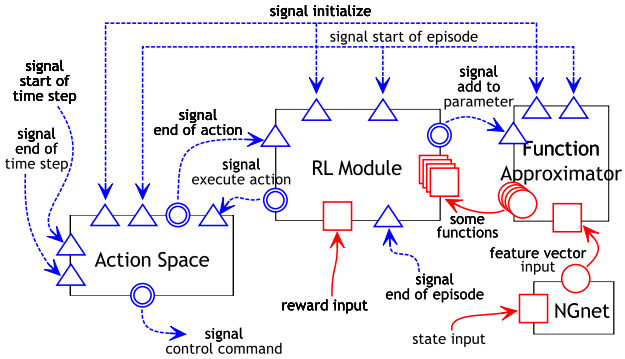

Fig.1 Example module structure around an RL module

2·1 SkyAI の特徴

2·1.1 高度なモジュール化

強化学習などの機械学習手法を高度に（適切に，細かく）モジュール化することによって，以下の利点が得られる．

拡張性： あるモジュールを拡張して新しいモジュールを作成するなどの拡張が容易になる．

実装の再利用性： モジュール構造を採用すると，タスク（問題）固有の実装と，汎用的な実装を分離することができる．例えばロボットの下位の制御モジュールと，汎用的な強化学習モジュールを分離できる．これによって強化学習モジュールはほかのタスクでも利用可能となり，実装の再利用性が向上する．

学習した知識の再利用性： 例えば，複数のタスク間で，学習したダイナミクスモデルを共通に使えるようになる．具体的には，モデルベース型強化学習をモジュールとして実装する際に，モデル学習の機能をモジュールの内部に埋め込まず，独立したモ

ジュールとして実装することで，複数の強化学習モジュール間で共有できる．

2·1.2 高い実行速度と柔軟性

SkyAI は実機のロボットシステムへ実装されることを想定しているため，高い実行速度が要求される．一方で，高度にモジュール化されたアルゴリズム群を柔軟に組み合わせられる必要がある．そこで，モジュール群の実装はコンパイル言語で開発し，さらにスクリプトインターフェイスを提供することで，コンパイル後も柔軟にモジュール構造を再構成できるようにする．

2·1.3 開発の容易性

モジュールは，C++のクラスに相当するものを想定している．すなわちデータを内部に持ち，データを操作する関数群を外部に公開する．よってモジュール間の通信は関数の提供（「ポート」と呼ぶ）によって実現される．このもっともシンプルな実装方法はいわゆる関数ポインタの利用である．しかし，この機構はスクリプトによって操作できる必要があるため，より複雑な実装が必要となる．このため，新しいモジュールの開発が容易ではなくなる．

類似のソフトウェア（例えば [5]）ではソースコードの生成プログラムを提供することで，この問題を回避している．しかし，そのような方式は，モジュールの改変を複雑にすることが多い．

そこで SkyAI では，マクロやテンプレートクラス群を提供することで，簡潔に SkyAI と整合性が取れたソースコードの記述を可能とする．これらは，標準のプリプロセッサやコンパイラのみで処理可能である．

2·2 モジュールの構成

各モジュール（**Fig.**1 RL module —強化学習モジュール— 参照）は C++のクラスに相当し，メンバ関数をほかのモジュールに対して公開できるようにカプセル化した「ポート」を複数個持つ．ポートとして公開する関数の型（引数，戻り値の型）は任意であり，ひとつのモジュールが持つことができるポートの数も任意である．ポートは，関数を提供する側とされる側でそれぞれ設置され，さらに能動的に関数を呼び出すタイプと，受動的に関数を呼び出されるタイプの 2 種類がある．つまり計 4 種類のポート（signal/slot ポート，in/out ポート）がある．

モジュールはパラメタを保持し，スクリプトから変更できる．また，モジュールが保持するパラメタをスクリプトとして書き出すことも可能であり，学習したパラメタはこの方法でファイルに保存される．

2·3 SkyAI の適用

SkyAI を用いたシステムの中心は,「エージェントクラス」である．エージェントクラスはすべてのモジュール群を管理し，スクリプト言語のパーサを提供する．エージェントクラスは汎用的なものであり，どのようなアプリケーションにでも適用できる．基本的には，プログラムの main 関数相当箇所でエージェントクラスを用いる．

SkyAI は強化学習モジュールや汎用的な行動空間を提供するが，ロボットやタスク固有のモジュールは提供しない（できない）．このためユーザが自分のロボットに適用する際，少なくともロボットの下位制御モジュールを，SkyAI のモジュールと同様の方法で実装する必要がある．

このため SkyAI を適用するには,（1）ロボットやタスク固有のモジュールを実装,（2）エージェントクラス，標準モジュール群，ユーザのモジュール群を併せてプログラムをビルド,（3）タスクを学習するスクリプトを記述，の流れで開発を行う．

3. スクリプトインターフェイス

スクリプトインターフェイスは，プログラムのコンパイル後も柔軟にモジュール構造を変更できることを主目的としている．よってスクリプトで記述できる必要があるのは，モジュール群のインスタンス化，ポート間の接続，モジュールのパラメタの操作である．加えて，柔軟性のみでなく，モジュール群の利用を容易にする手段もスクリプトインターフェイスが提供する．さらに，適切にモジュール群を実装しておくことで，タスクの定義もスクリプト上で記述できる．

3·1 基本機能

モジュールのインスタンス化とは，C++のクラスをメモリを確保し変数として生成する，という意味である．よってこの操作にはモジュール名（クラス名）とインスタンス名を指定する必要がある．ポートの接続には，ふたつのポートの指定が必要である．各ポートはモジュールのインスタンス名とポート名によって指定される．モジュールのパラメタの操作は，ベクタなどへの代入が簡潔に記述できるように工夫されている．以下にスクリプトの一例を示す．

```
// instantiate a module:
module MBasicLearningManager lm
module MHumanoidEnvironment  env
// connect ports:
connect lm.signal_initialization, env.slot_initialize
// set parameters:
env.config ={
    FPS = 50.0
    PDGainKp = (2.5, 2.5, 5.0)  // assign to a vector
  }
```

3·2 複合モジュール

再利用性を高めようとすると，モジュール群は細分化される．例えば強化学習モジュールから価値関数の関数近似モジュールを分離して実装すると，それぞれのモジュールの汎用性が高まる．しかしモジュール群が細分化されていると，モジュール構造が複雑になり多くの接続が要求され，利用しにくくなる．そこで，いくつかのモジュールをまとめた「複合モジュール」をスクリプト上で定義できるようにすることで，細分化されたモジュール群の利用を容易にする．

Fig.2 に複合モジュールの一例を示す．いくつかのモジュールを内部で定義し，一部のポートを外部に公開する．パラメタについても，指定したもののみ公開できる．複合モジュールは，通常のモジュールと同様の方法でインスタンス化，接続，パラメタ設定できる．複合モジュールを，ほかの複合モジュールに組み込むことも可能である．

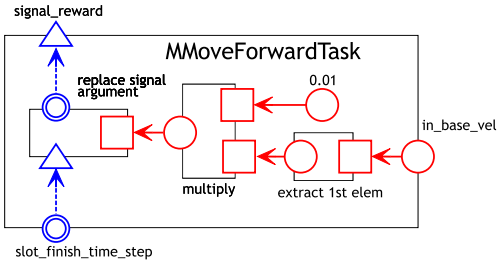

Fig.2 Example of a composite module

3·3 タスク定義

強化学習において，タスクを定義するとは，エピソードの開始・終了タイミング，及び報酬関数を定義することである．SkyAI では，これらをスクリプトで記述できるようにするため，演算モジュール群も提供している．例えば **Fig.**2 は匍匐タスク用の報酬を計算する複合モジュールである．このモジュールは，外部からボディリンクの重心速度を受け取って報酬を計算し，時間ステップごとに報酬シグナルを発生させている．

タスクをスクリプト上で記述できると，タスクごとにモジュールを実装する必要がなくなり，プログラムの再コンパイルが不要となる．このためロボットのエンドユーザもタスクを指定できるようになる．

4. 適用例

本章では，SkyAI をシミュレーションの小型ヒューマノイドの匍匐タスクに適用した例，及び実機の小型ロボット群（2 種類に組み立てた ROBOTIS 社製の Bioloid）の匍匐タスクに適用した例を示す．なお，これらのタスクはベンチマークとして SkyAI に含まれている．

4·1 小型ヒューマノイドの匍匐タスク

この実験で用いたヒューマノイドロボット，及び匍匐タスクの定義は [1] で述べられているものと同等である．

4·1.1 強化学習モジュール群

強化学習手法としては，Q(λ)-learning モジュール（連続状態・離散行動用）と価値関数の線形関数近似モジュールを組み合わせて用いる．さらに，強化学習モジュールの行動出力（離散値）を，DCOB [1] モジュールを用いて制御入力（連続値ベクトル）に変換する．線形関数近似モジュールには，状態出力に NGnet モジュールを通して変換した特徴ベクトルを入力する．

4·1.2 タスクモジュール群

ロボット（環境）モジュールは，Open Dynamics Engine (ODE) を用いて作成した．このモジュールには，リセット（ロボットを初期状態に戻す），目標制御角の入力，時間ステップシグナル，状態の出力，などのポートを持たせた．

このタスクでは，簡単のため，ロボットの自由度を 5 に制約する．この変換はスクリプト上で「定数乗算モジュール」を用いて行われる（**Fig.**3）．

タスクモジュールは，いくつかのモジュール群を組み合わせて複合モジュールとしてスクリプト上で定義

```
module  MConstMultiplier_TRealMatrix_TRealVector
        action_converter
// DCOB's output is connected to the converter:
connect bftrans.signal_execute_command ,
        action_converter.slot_x
// output of the converter is connected to the robot's
// target angle port:
connect action_converter.signal_y ,
        env.slot_execute_command_des_q
// set the converter's parameter:
action_converter.config ={
    Factor ={
        resize(17, 5)
        //  j   0  1  2  3  4
        [0]=  ( 0, 0, 0, 0, 0)
        [1]=  ( 1, 0, 0, 0, 0)
        [2]=  ( 0, 0, 0, 0, 0)
        [3]=  ( 0, 1, 0, 0, 0)
        ...
        [15]= ( 0, 0, 0, 0, 1)
        [16]= ( 0, 0, 0, 0, 0)
      }
  }
```

Fig.3 Script to constrain the robot's DoF from 17 to 5

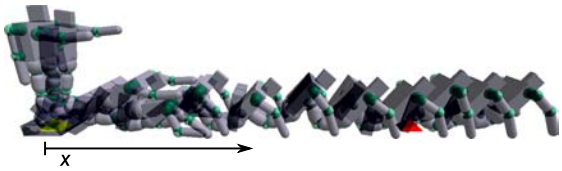

Fig.4 Sequence of an acquired crawling motion

した．報酬はロボットの重心の x 軸速度に比例して与えられ，エピソードは一定時間経過もしくは転倒による累積ペナルティが一定値を下回れば終了する．

4·1.3 学習結果

Fig.4 に得られた匍匐の様子を示す．学習曲線は省略するが，[1] の DCOB と同等である．

4·2 Bioloid の匍匐タスク

次に，実機のロボットへの適用例として，ROBOTIS 社製の小型ロボット Bioloid の匍匐学習を示す．Bioloid は多種の形状に組み立てられ，以下の実験では King Spider 型と Dinosaur 型を用いた．King Spider の匍匐タスクは [6] で示したものと同等である．報酬は，**Fig.**5 に示すように，赤外線距離センサを用いて壁からの距離を計測，微分し速度に変換して与える．

Dinosaur についても，アクチュエータの数以外は King Spider と同様の設定で実験を行ったが，King Spider と比較して手足が短く速度が出にくいため，与える報酬を 5 倍に増幅している．また，距離センサの位置を，腹部から頭部に移動している．さらに，Dinosaur は重心位置が比較的高く，転倒することがあったため，転倒した場合（操作者が目視で判定）は負の報酬を与えるようにした．

強化学習手法は，前節と同様に Q(λ)-learning と DCOB を用いたものである．King Spider は計 4 回，Dinosaur は計 3 回試行を行った．**Fig.**6 に King Spider で得られた学習曲線（エピソード-収益）を，**Fig.**7 に得られた匍匐の一例をそれぞれ示す．学習の後半で収益（累積報酬）が低下しているエピソードがあるのは，

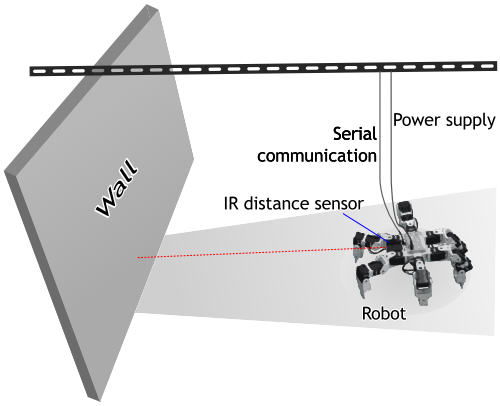

Fig.5 Experimental environment of the crawling task of the spider robot

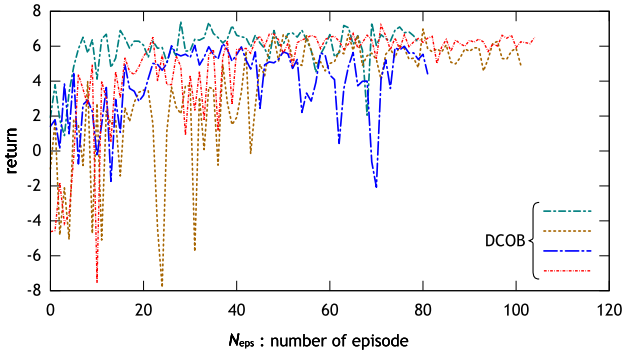

Fig.6 Resulting learning curves of the crawling task of a bioloid robot. Each curve shows the return per episode in a run

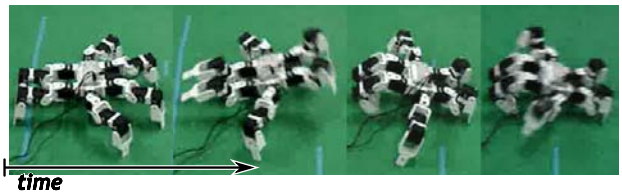

Fig.7 Snapshots of an acquired crawling motion of the spider robot (4.8[s], 5.4[s], 6.0[s], 7.2[s])

計算機との通信トラブル，ロボットのネジが外れるなどのトラブルが原因である．**Fig.**8 は Dinosaur で得られた学習曲線，**Fig.**9 は得られた匍匐の一例である．King Spider よりも学習にやや時間が掛かっているのは，転倒を含みタスクの難易度が上がっているからであると考えられる．

5. まとめ

本稿では，高度にモジュール化された強化学習ライブラリ SkyAI を，スクリプトインターフェイスを中心に紹介した．SkyAI はオープンソースで，現在は GNU General Public License で公開している．`skyai.sourceforge.net` より入手可能である．依存しているライブラリは，Boost C++ ライブラリ，Octave（liboctave とヘッダ），ODE（ベンチマークのみ）で，いずれもフリーで入手できる．本稿で示した実機のベンチマークタスクは，Bioloid の購入のみで再現可能である（別途のセンサは不要）．

Fig.8 Resulting learning curves of the crawling task of the dinosaur robot. Each curve shows the return per episode in a run

Fig.9 Snapshots of an acquired crawling motion of the dinosaur robot (7.2[s], 7.6[s], 8.0[s], 8.6[s])

謝辞 本研究の一部は科研費（22·9030）の助成を受けて行った．

参考文献

[1] A. Yamaguchi, J. Takamatsu and T. Ogasawara: “Constructing action set from basis functions for reinforcement learning of robot control”, the IEEE Internactional Conference in Robotics and Automation (ICRA’09), Kobe, Japan, pp. 2525–2532 (2009).

[2] J. Morimoto, S. Hyon, C. Atkeson and G. Cheng: “Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction”, the IEEE Internactional Conference in Robotics and Automation (ICRA’08), pp. 2711–2716 (2008).

[3] J. Kober and J. Peters: “Learning motor primitives for robotics”, the IEEE Internactional Conference in Robotics and Automation (ICRA’09), pp. 2509–2515 (2009).

[4] J. Zhang and B. Rössler: “Self-valuing learning and generalization with application in visually guided grasping of complex objects”, Robotics and Autonomous Systems, **47**, 2-3, pp. 117–127 (2004).

[5] N. Ando, T. Suehiro and T. Kotoku: “A software platform for component based rt-system development: OpenRTM-Aist”, Simulation, Modeling, and Programming for Autonomous Robots, **5325**, pp. 87–98 (2008).

[6] 山口, 高松, 小笠原:“強化学習によるロボットの動作獲得のための基底関数に基づく行動空間生成手法 DCOB –実機多自由度ロボットの匍匐動作への適用–”, 日本機械学会ロボティクス･メカトロニクス講演会 2010 (ROBOMEC2010), 旭川, pp. 2P1–G10 (2010).