学習するロボット：
強化学習システムを用いた適応的制御の枠組み

○木村 元，　小林 重信
東京工業大学 大学院総合理工学研究科

Learning Robots: A Framework of Adaptive Control using Reinforcement Learning System

○ Hajime Kimura,　Shigenobu Kobayashi
Interdisciplinary Graduate School of Science and Eng., Tokyo Institute of Technology

Abstract: In this paper, we introduce a new adaptive control framework composed of *reinforcement learning (RL)*. The RL is on-line learning of an input-output mapping through a process of trial and error designed to improve its behavior. We show the features of this framework through a demonstration of learning control rules in real robots.

1　はじめに

本稿では，強化学習という新しい枠組みを解説し，ロボットの制御規則獲得問題への適用例を紹介する．

1.1　強化学習：適応的学習制御の枠組み

強化学習とは，試行錯誤を通じて環境へ適応する学習制御の枠組である．教師付き学習 (supervised learning) と異なり，状態入力に対する正しい行動出力を明示的に示す教師が存在しない．その代わり報酬というスカラーの情報を手がかりにして学習するが，報酬にはノイズや遅れがある．そのため，行動を実行した直後の報酬をみるだけでは，学習主体はその行動が正しかったかどうかを判断できないという困難を伴う．強化学習の枠組を Fig.1 に示す．学習の主体「エージェント」と制御対象「環境」は以下のやりとりを行う．

1. エージェントは時刻 t において環境の状態観測 s_t に応じて意志決定を行い，行動 a_t を出力
2. エージェントの行動により，環境は s_{t+1} へ状態遷移し，これに応じた報酬 r_t をエージェントへ与える．
3. 時刻 t を $t+1$ に進めてステップ 1 へ戻る．

形式的には，エージェントは利得（return: 最も単純な場合，報酬の総計）の最大化を目的として，状態観測から行動出力への写像（政策（policy）と呼ばれる）を獲得する．環境とエージェントには一般に下記の性質が想定される．

- エージェントは予め状態遷移の知識を持たない．
- 環境の状態遷移は確率的．
- 報酬の与えられ方は確率的．
- 状態遷移を繰返した後，やっと報酬にたどり着くような，段取り的な行動を必要とする環境（報酬の遅れ）．

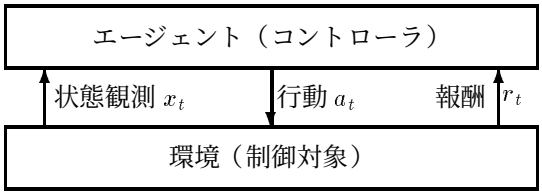

Figure 1: 強化学習の枠組．エージェントは試行錯誤を通じて適切な制御規則を獲得していく．

1.2　制御の視点から見た強化学習の特徴

強化学習が期待される理由の一つは，不確実性のある環境を扱っている点にある．多くの実世界の制御問題において，厄介な不確実性の扱いが求められる．もう一つの理由は，報酬に遅れが存在し，離散的な状態遷移も含んだ段取り的な制御規則の獲得を行う点にある．設計者がゴール状態で報酬を与えるという形で，させたいタスクをエージェントに指示しておけば，ゴールへの到達方法はエージェントの試行錯誤学習によって自動的に獲得される．つまり，設計者は「何をすべきか」をエージェントに報酬という形で指示しておくだけで「どのように実現するか」をエージェントが学習によって自動的に獲得する枠組となっている．

1.3　応用上期待できること

1.3.1　制御プログラミングの自動化・省力化

環境に不確実性や計測不能な未知のパラメータが存在すると，タスクの達成方法やゴールへの到達方法は設計者にとって自明ではない．よってロボットへタスクを遂行するための制御規則をプログラムすることは設計者にとって重労働である．ところが，達成すべき目標を報酬によって指示することは前記に比べれば遥かに簡単である．そのため，タスク遂行のためのプログラミングを強化学習で自動化することにより，設計者の負担軽減が期待できる．十分に優れた性能を持つ強化学習エージェントをコントローラとして１つだけ開発しておけば，あとはロボットの目的に応じて報酬の与え方だけを設計者が設定するだけで，あらゆる種類のロボット制御方法を同一のコントローラによって自動的に獲得できる．

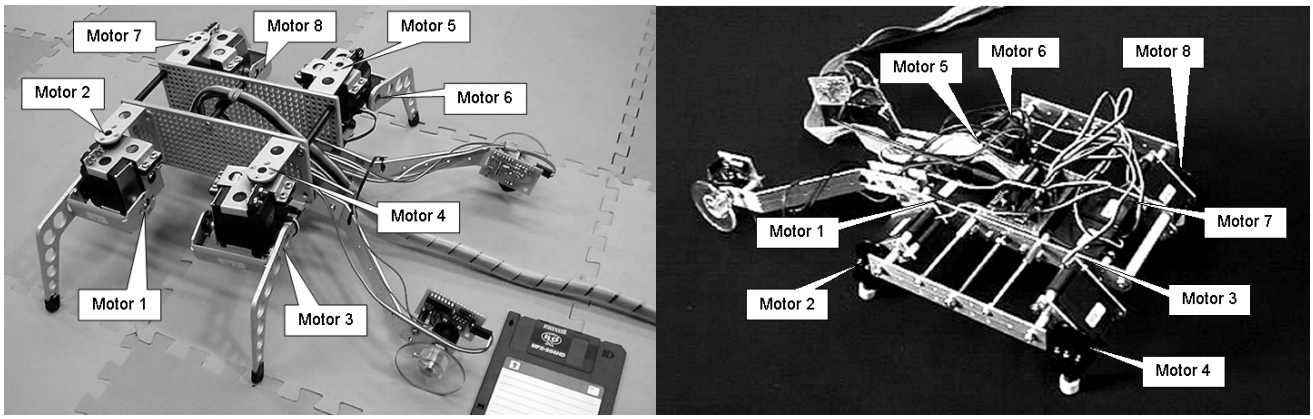

Figure 2: 学習対象のロボット（左 OCT1, 右 OCT2）各足は角度指令で動作するサーボモータ２個を持つ.

1.3.2 ハンドコーディングよりも優れた解

試行錯誤を通じて学習するため，人間のエキスパートが得た解よりも優れた解を発見する可能性がある．特に不確実性（摩擦やガタ，振動，誤差など）や計測が困難な未知パラメータが多い場合，人間の常識では対処し切れないことが予想され，強化学習の効果が期待できる．エキスパートの制御規則を学習初期状態に設定して，それを改善する場合と，全くのゼロから学習を開始し，設計者にとっては意外な新しい解を発見する場合とが考えられる．

1.3.3 自律性と想定外の環境変化への対応

機械故障などの急激な変化やプラントの経年変化のような緩慢な変化など，予め事態を想定してプログラミングしておくことが困難な環境の変化に対しても自動的に追従することが期待できる．特に宇宙や海底など，通信が物理的に困難な場合や，通信ネットワークの制御のように現象のダイナミクスが人間にとって速すぎる場合において，強化学習の自律的な適応能力が特に威力を発揮する．

2 強化学習の適用例：ロボットの歩行動作獲得

前章で説明した強化学習の利点について，具体例を挙げて説明する．Fig.2 に示すように，２種類の４足ロボット（クモ型の OCT1 とイヌ型の OCT2）に対し，完全に同一の強化学習アルゴリズムを適用し，効率よく前進する動作を獲得する [4]．エージェントすなわちロボットのコントローラが獲得すべき制御規則は，現在の関節の角度を状態入力として与えられたとき，前進するような動きとなるようにモータの目標値とすべき関節の角度を出力することである．ロボットの学習目標は，効率よくまっすぐ前進することなので，各時刻におけるボディの前進速度と曲がり具合の評価値をエージェントが報酬として受け取るよう設定する．

エージェントとロボットは以下のやりとりを行う．(1) エージェントは状態観測としてロボットの関節の角度を受け取る．(2) エージェントは行動出力として関節モータの角度の目標値を出力．(3) ロボットは目標角度の方向へ各モータを動かす．(4) 約 0.3〜0.5 秒後，ロボットはボディが移動した距離や曲がり具合を計測し，その値を報酬としてエージェントに与える．(5) ステップ 1 に戻って繰り返す.
上記の設定により，ロボットを効率よく前進させる学習問題は，エージェントが利得（報酬の総計）を最大化するよう政策を探索する最適化問題へ帰着される．

ここで注目すべき点は，ロボット OCT1 と OCT2 がメカニズム的に全く異なるにもかかわらず，強化学習問題として見ると同じになる点と，求めるべき制御規則が比較的複雑である割に，報酬の設定が極めて簡単な点である．よって，OCT1 へ適用可能な強化学習アルゴリズムが，何も変更することなく OCT2 にも適用できるという意味と，設計者が極めて簡単な報酬設定をするだけで複雑な制御規則を自動的に得られるという２つの意味において，強化学習による制御規則プログラミングの自動化・省力化を実現している．

ロボットの各脚は Fig.2 に示すように２個の位置制御サーボモータで駆動され，全体で８自由度である．クモ型ロボットの OCT1 では，各脚の２つのモータはそれぞれリフト脚，スイング脚を受け持つ．リフト脚は上下方向，スイング脚は前後方向へ足先を運ぶ役割を持つ．これに対しイヌ型ロボットの OCT2 では，各脚の２つのモータが駆動する関節軸は平行であり，足先を運ぶ方向について役割分担がなく，互いの動作が干渉するため，制御規則の構築は OCT1 よりも困難である．各モータと関節は図中で示すとおり，OCT1 では反時計回りに対応付けられているのに対し，OCT2 では左側前後，右側前後の順番で割り当てられている点でも構造的に異なる．Fig.2 のロボット後方に見える 2 個の車輪は，ボディの移動を検出し報酬信号を生成する．２つの車輪の移動速度平均はロボットの前進速度を示し，２つの車輪の移動速度の差分はロボットが旋回していることを示す．ロボットの学習目標はまっすぐに前進することなので，各時間ステップでの報酬は，２つの車輪の移動速度平均から差分の絶対値を引いた値とした．

実時間でおよそ 80〜120 分学習後の様子を Fig.10 に示す．OCT1 では Fig.10 左側に示すようにカメのような歩行動作を獲得した．OCT1 では右側の前足 (leg4) と左側の後ろ足 (leg2) はほぼ同位相で，また左側の前足 (leg1) と

右側の後ろ足 (leg3) がほぼ同位相で同期して動いており，歩容はトロット (trot) に近い．カーペット上とゴムマット上では OCT1 で獲得された動作にはやや違いが見られた．カーペット上では，ボディは容易に床をすべることが可能なのでしばしばボディを引きずる動作が見られた．これは，支持脚で体重を支える動作よりも前進させる動作を優先して学習した結果と考えられる．一方，摩擦の大きなゴムマット上では，ロボットのボディが床から離れた状態を保持し続ける傾向が見られた．これらの結果は，ロボットがそれぞれ周囲の状況に応じて適応的に動作を獲得できたことを示している．OCT2 でも Fig.10 右側に示すように足先以外の脚部やボディをひきずる動作となったが，左右に大きく傾きながらも高速に直進する動作が得られた．このような動作は実際に試行錯誤しない限り，獲得するのは困難である．

3 強化学習の基礎

強化学習アルゴリズムの解析は，マルコフ決定過程 (Markov decision process: MDP) で表現される環境において数多く示されている．この強化学習の基礎理論により，環境がマルコフ決定過程で表現できることさえ分かれば，エージェントが予め遷移確率や報酬についての知識を持っていなくても学習できることが示されている．強化学習を実問題へ適用する場合，問題表現がマルコフ決定過程となるように状態表現や報酬の与え方を設計することが重要である．以下，強化学習の基礎について，マルコフ決定過程と代表的な強化学習アルゴリズムを簡単に説明する．

3.1 マルコフ決定過程による環境モデル化

環境のダイナミクスを以下のようにモデル化したのが MDP である．状態空間を $\mathcal{S}$，行動空間を $\mathcal{A}$，実数の集合を $\mathcal{R}$ と表す．各離散時間ステップ t において，エージェントは状態 $s_t \in \mathcal{S}$ を観測して行動 $a_t \in \mathcal{A}$ を実行し，状態遷移の結果，報酬 $r_t \in \mathcal{R}$ を受け取る．一般に報酬と遷移先の状態は確率変数だが，MDP ではその分布は s_t と a_t だけに依存する．遷移先の状態 s_{t+1} は遷移確率 $T(s_t, a, s_{t+1})$ に従い，報酬 r_t は期待値 $r(s_t, a)$ に従って与えられる．

学習エージェントは $T(s_t, a, s_{t+1})$ や $r(s_t, a)$ についての知識は事前に持たない．強化学習の目的はエージェントのパフォーマンスを最大化する政策 (policy) を生成することである．無限期間のタスクにおいて自然な評価規範として，以下の割引報酬合計による評価がある．

$$V_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \ , \tag{1}$$

ただし割引率 $0 \leq \gamma \leq 1$ は未来の報酬の重要度を示し，V_t は時刻 t の評価値を表す．MDP では，評価値は以下のように定義できる：

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \, r_t \middle| s_0 = s, \pi \right] \ , \tag{2}$$

ただし $E\{\cdot\}$ は期待値演算を表す．MDP では式 2 で定義される各状態 s における評価値を最大化する政策を探すことが目標である．本稿のロボット学習問題では，状態空間が上下界の存在する 8 次元の連続な空間であり，行動空間も同じである．

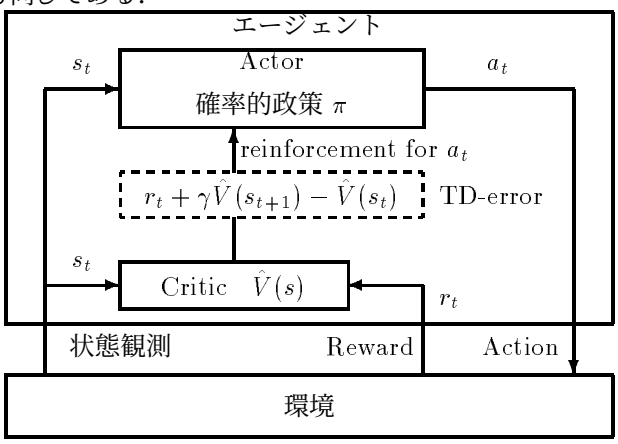

Figure 3: 一般的な actor-critic アルゴリズムの枠組．Critic は状態評価値を推定し actor に TD-error を与える．Actor は確率的政策に基づいて行動選択すると同時に，TD-error を手がかりに政策を改善する．

1. エージェントは環境において状態 s_t を観測
   Actor は確率的政策 π に従い行動 a_t を実行
2. Critic は報酬 r_t を受取り，次状態 s_{t+1} を観測し，actor への強化信号として以下の TD-error を計算：
   $$(\text{TD-error}) = \left[r_t + \gamma \hat{V}(s_{t+1}) \right] - \hat{V}(s_t) \ ,$$
   γ は割引率，
   $\hat{V}(s)$ は critic が推定した割引報酬の期待値．
3. TD-error を用いて actor の行動選択確率を更新
   (TD-error) > 0 ならば，実行した行動 a は比較的好ましいと考えられるので，この選択確率を増やす．
   逆に (TD-error) < 0 ならば，実行した行動 a は比較的好ましくないものと考えられるので，この選択確率を減らす．
4. TD 法を用いて critic の value の推定値を更新
   例えば TD(0) ならば以下のように計算する．
   $\hat{V}(s_t) \leftarrow \hat{V}(s_t) + \alpha \, (\text{TD-error})$, ただし α は学習率．
5. 手順 1. から繰り返す．

Figure 4: 一般的な actor-critic アルゴリズムの処理手順

4 強化学習アルゴリズム

4.1 Actor-Critic アルゴリズム

Actor-critic 法 [1](Fig.3) は基本となる仕組みが単純な上，連続空間への適用が容易で，ロボットの制御規則獲得において有望な手法である．処理手順の概要を Fig.4 に示す．Actor は，状態から行動への確率分布である確率的政策 π に従ってランダムに行動を選び実行する．Critic は，actor の政策のもとでのそれぞれの状態の評価値を推定する．ある状態 s_t で行動 a_t を選択後，報酬 r_t を受け取って状態

s_{t+1} へ遷移した時，critic は以下の TD 法に従って状態評価の推定値 $\hat{V}(s_t)$ を更新する：

$$\text{TD\_error} = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \quad (3)$$

$$\hat{V}(s_t) \leftarrow \hat{V}(s_t) + \alpha \text{TD\_error} \quad (4)$$

ただし $0 < \alpha \leq 1$ は学習率，$0 \leq \gamma < 1$ は割引率を表す．Actor は，critic で計算される TD_error を手がかりにして政策を改善する．TD_error が正のとき，エージェントは良い状態へ遷移したと考えられるので，状態 s_t における行動 a_t の選択確率を高くするよう修正する．逆に TD_error が負のとき，エージェントは悪い状態へ遷移したと考えられるので，状態 s_t における行動 a_t の選択確率を低くするよう修正する．この原理は行動空間が連続であってもそのまま拡張できる．

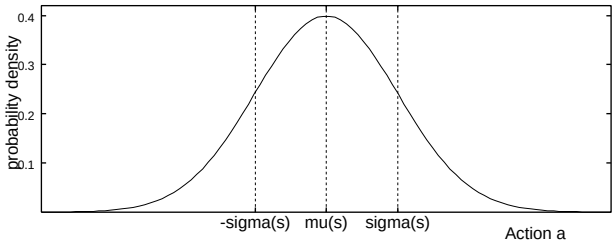

Figure 5: Actor において 1 次元連続値行動を扱う例．中心値 mu(s)，標準偏差 sigma(s) の正規分布に従ったランダムサンプルによって行動 a を選択．状態遷移の結果，TD_error が正なら，中心値 mu(s) を a の方向へ修正．行動 a が ±sigma(s) の内側だったなら，sigma(s) を小さくする方向へ，外側なら大きくする方向へ修正する．TD_error が負なら逆の操作を行う．

4.2 連続な行動空間を扱う Actor-Critic

前述のように，actor の確率的政策 π を確率分布関数とすることで，actor-critic を連続行動空間へ拡張できる．Actor では，一般に確率的政策をパラメータ関数表現し，そのパラメータを勾配法によって更新していく．Fig.5 は確率的政策として正規分布を用いた例を示す．この actor を用いたエージェントは，中心値 mu(s)，標準偏差 sigma(s) の正規分布に従ったランダムサンプリングによって行動 a を選択する．状態遷移後，式 3 に従って TD_error を計算し，この値に基づいて actor の政策パラメータを更新する．TD_error が正のとき，エージェントは良い状態へ遷移したと考えられるので，状態 s_t における行動 a_t の選択確率を高める方向へ修正するが，正規分布の場合は中心値 mu(s) を a の方向へ修正し，行動 a が ±sigma(s) の内側だったなら，sigma(s) を小さくする方向へ，外側なら大きくする方向へ修正することで実現できる．TD_error が負の場合は逆の操作を行う．本ロボットの問題においては，行動空間に上下界が存在するため，この正規分布による行動選択に修正を加えている．実装方法の詳細は文献 [4] を参照していただきたい．

5 実験結果

本実験では，Fig.6 に示すように各ロボットをネットワーク上のサーバとして設定し，LAN を介して TCP プロトコルによってそれぞれのサーバへ接続して通信する．同一の学習エージェントプログラムによってそれぞれ異なるロボットを学習させた．

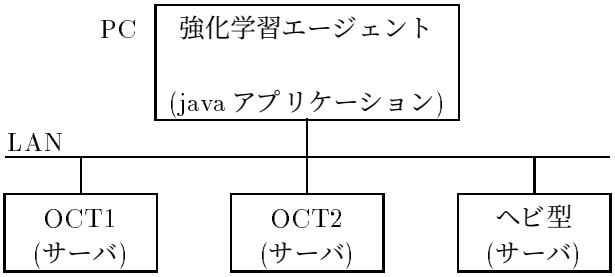

Figure 6: 実験に用いたシステムのネットワーク構成．各ロボットは，ネットワーク上の異なるサーバに見える．

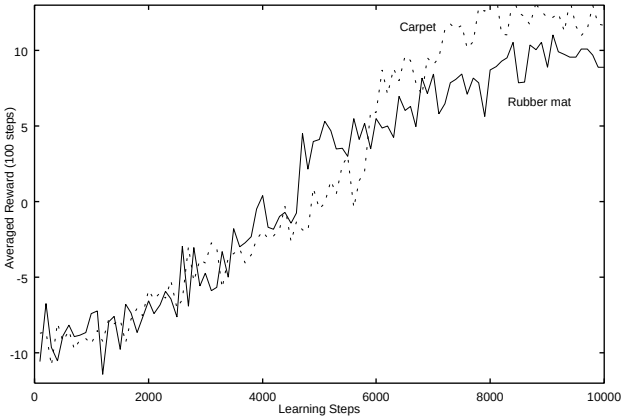

Figure 7: OCT1 の学習の様子．10000 ステップ（約 80 分）

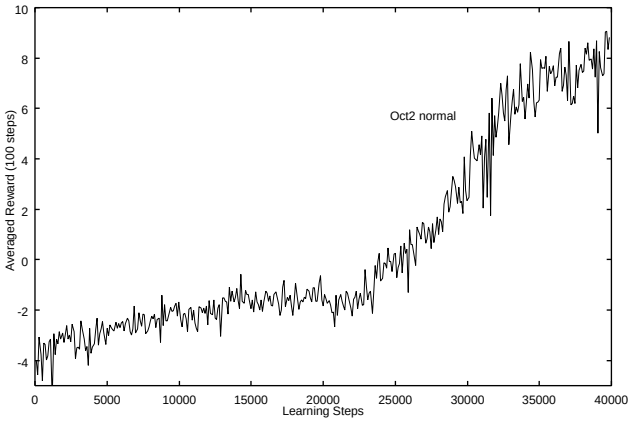

Figure 8: OCT2 の学習の様子．40000 ステップ（約 160 分）

OCT1 は比較的よく滑べるカーペット上および摩擦の大きなゴムマット上で実験を行った．OCT2 はカーペット上のみで実験を行った．

Fig.7 は OCT1 における学習曲線を表す．縦軸は 100 ステップ毎の報酬の平均値を表すが，報酬は (1 ステップで

２つの車輪が前進した平均値) − (１ステップでの２つの車輪の差分の絶対値) という形式で与えられるため，学習初期において値が負の値を示しているのはロボットが後進しているせいではなく，左右に旋回しているためである．OCT1 ロボットはカーペットおよびゴムマット上の両方の設定において 8000 ステップ (約１時間) 以下という実時間で良好な挙動を獲得している．10000 ステップ後の実際の移動速度は約 5cm/sec である．

Fig.8 は OCT2 における学習曲線を表す．OCT2 では各脚の２つのモータは足先を運ぶ方向について役割分担がなく，互いの動作が干渉するという困難さから OCT1 に比べ動作獲得までに３〜４倍の学習ステップ数を要している．それでも約２時間という実時間で動作獲得できた．

さらに我々は，OCT ロボットに用いた強化学習エージェントを，Fig.9 に示すヘビ型ロボットへ適用し，適切な移動制御規則が得られるかどうかを試した [5]．このロボットは，2 自由度の関節 4 個で構成されており，計 8 個のモーターを用いているため OCT ロボットと同じ入出力を持っている．胴体中央の横に，移動距離と方向を計測する車輪が取り付けてあり，側面方向移動タスク学習時は，真横に直進した場合に最大の報酬が与えられる．また，旋回動作タスク学習時は，旋回時に報酬が最大になるよう報酬が設定される．学習の結果得た動作の一例を Fig.11 に示す．当初，側面方向への移動動作としてサイドワインダー（ガラガラヘビ）のように体をくねらせながら移動する動作の獲得を期待していたのだが，学習させた結果，頭としっぽをオールのように動かす動作を獲得した．これは，胴体中央に速度と方向検出用車輪を取り付けたため，体をくねらせると報酬が負になってしまうためであり，報酬獲得最大化の意味で合理的な動作を獲得している．

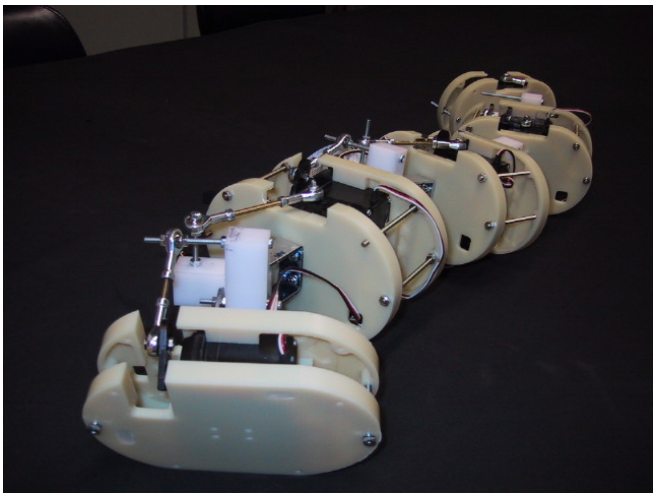

Figure 9: ヘビ型ロボット

6　おわりに

本稿では，強化学習の特徴について解説することに重点を置き，ロボットの制御規則獲得問題へ適用した例を紹介した．実験において獲得された動作では，ボディを引きずるなど好ましくないように思える動作が見られたが，ボディが床についているかどうかなどを報酬に反映させれば，よりロボティクス的に好ましい動作の獲得が期待できるものであり，強化学習の特長を損なうものではない．

しかし，実問題では「試行錯誤」が許されない場合が多く，満足な動作を獲得する前にロボットが壊れてしまうなど問題も多い．そのため，強化学習に対して批判的な意見があるのも事実だが，他の学習のテクニックと組み合わせることによって学習速度が向上していくことが期待される．さらに今後，強化学習の使用を前提としたハードウエア設計がなされれば，強化学習のポテンシャルを十分に生かした新しい製品やサービスが出現する可能性がある．

References

[1] Barto, A.G., Sutton, R.S. & Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC–13, no.5, pp. 834–846 (1983).

[2] 木村 元，宮崎 和光，小林 重信：強化学習システムの設計指針，計測と制御, Vol.38, No.10, pp.618–623 (1999).

[3] 木村 元，小林 重信：Actor に適正度の履歴を用いた Actor-Critic アルゴリズム– 不完全な Value-Function のもとでの強化学習，人工知能学会誌, Vol.15, No.2, pp.267–275 (2000).

[4] 木村 元，山下 透，小林 重信：強化学習による４足ロボットの歩行動作獲得，電気学会 電子情報システム部門誌, Vol.122-C, No.3, pp.330–337 (2002).

[5] 楠 義寛，木村 元，小林 重信：蛇型ロボットの強化学習による制御規則獲得，計測自動制御学会 第 30 回知能システムシンポジウム, pp.161-166 (2003).

[6] Sutton, R.S. & Barto, A.: Reinforcement Learning: An Introduction, *A Bradford Book*, The MIT Press (1998).

[7] Sutton, R.S., McAllester, D., Singh, S. & Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation, *Advances in Neural Information Processing Systems 12 (NIPS12)*, pp. 1057–1063 (2000).

[8] Williams, R.J.: Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning, *Machine Learning 8*, pp. 229–256 (1992).

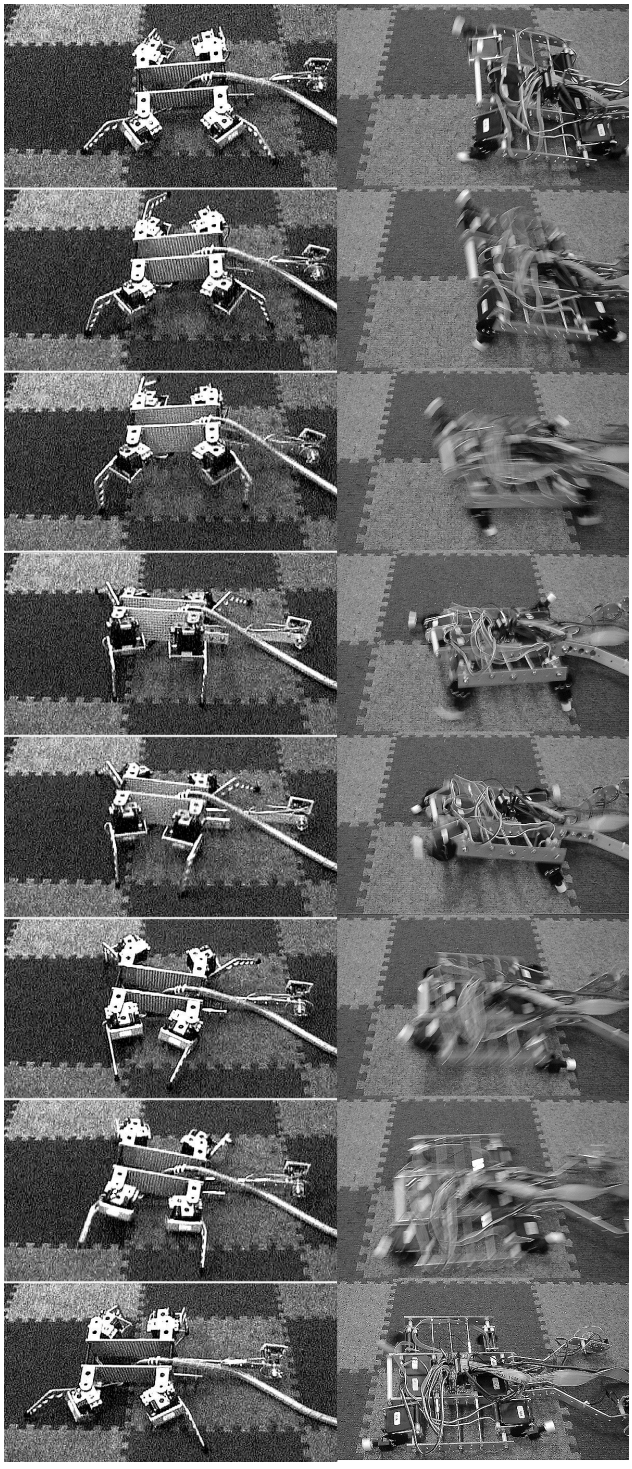

Figure 10: カーペット上にて 10000 ステップ（約 80 分）学習後に得た動作の一例．左側：約 10000 ステップ後（80 分），右側：約 30000 ステップ後（120 分）．

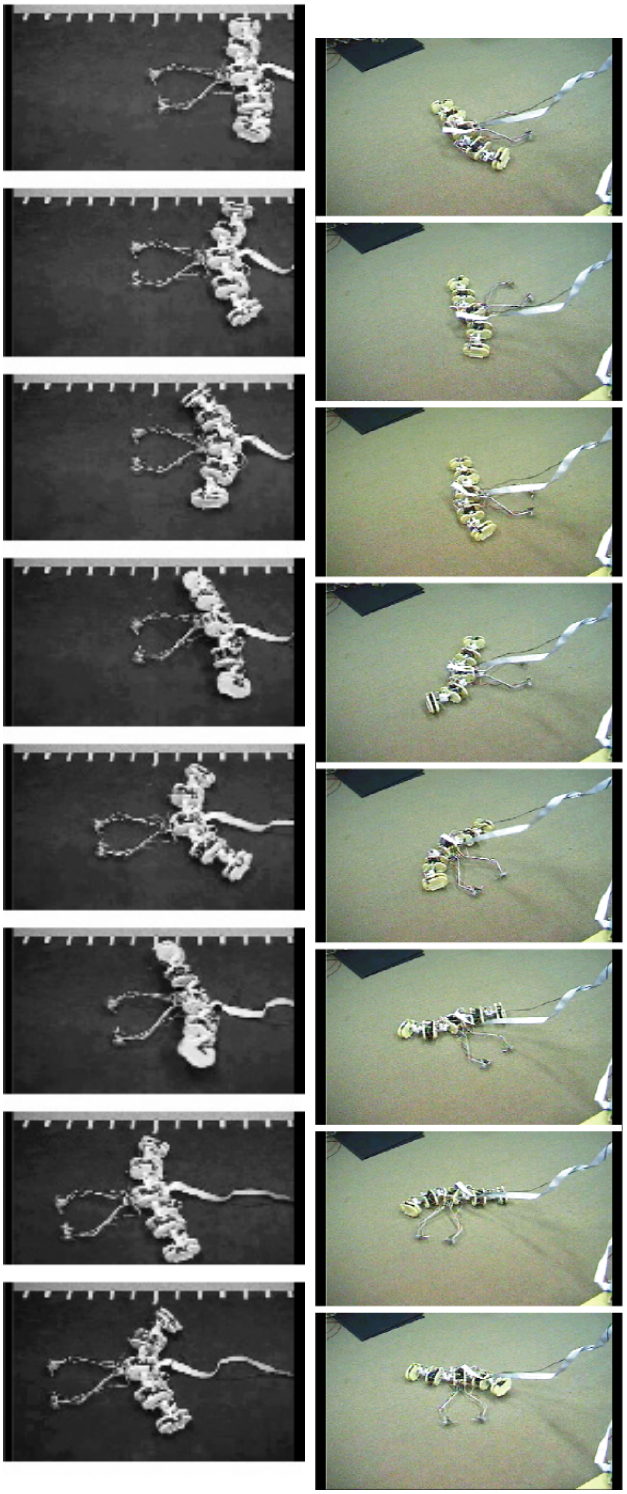

Figure 11: ヘビ型ロボットが得た動作の一例．左側：側面方向移動動作，右側：旋回動作．